Journal of Basic Sciences (JBS)
مجلة العلوم الأساسية
Volume 39, Issue 1, January – March 2026
Page no.: 75 - 98
Email: jbs@journals.asmarya.edu.ly ISSN: 3105-8728

Performance Evaluation of Machine Learning Algorithms in Detecting Network Intrusion Attacks

Nadia Attiya Algaddar[1], Melad Mohamed Al-Daeef [2*], Alhadi A. Klaib[3]

[1] The Libyan Academy for Postgraduate Studies, Tripoli, Libya
[2] Department of Internet Technologies, Elmergib University, Al-Khomes, Libya
[3] Libyan Authority for Scientific Reseaarch, Tripoli, Libya

*** Corresponding author: mmaldaeef@elmergib.edu.ly**

Received: Mar. 04, 2026	Accepted: Mar. 15, 2026	Published: Mar. 25, 2026

Abstract:
Computer networks are at the heart of today's information society and are crucial for providing key services such as e-commerce, cloud computing, healthcare systems, and smart infrastructures. These services are playing a vital role in enhancing and improving the lifestyle and connectivity. However, these services are vulnerable to network-based attacks and cybercrimes like DDoS, brute-force login attacks, botnets, and web-based attacks. Intrusion Detection Systems (IDSs), as their name suggests, are designed to monitor the traffic in a network and detect any intrusion or malicious activities. These systems are broadly classified into two types: Signature-based and Anomaly-based IDSs. Signature-based IDSs is based on a set of predefined rules, they can detect known types of attacks. Anomaly-based IDSs, on the other hand, can detect unknown types of attacks. This paper discusses how Machine Learning (ML) techniques are employed in IDSs for sorting network traffic and for automatically detecting intrusion attacks. Four supervised learning algorithms, namely Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LR), are evaluated and compared using the CICIDS2017 benchmark dataset. The dataset includes realistic network traffic that includes both legitimate user activities and various types of cyber-attacks. Many preprocessing steps are implemented on the dataset, including data cleaning, correlation analysis of features, dimensionality reduction, handling class imbalance using a hybrid approach that includes down sampling and Synthetic Minority Over-sampling Technique (SMOTE). The performance of the classifier is measured using various performance parameters like Accuracy, Precision, Recall, F1 score, Confusion Matrix, and Execution Time. From the results, it can be inferred that the DT algorithm provides better results with an accuracy of 99.94%, indicating its suitability for IDSs.

Keywords: Intrusion Detection Systems, Machine Learning, Classification Algorithms, Network Security.

DOI: https://doi.org/10.59743/jbs.v39i1.348

تقييم أداء خوارزميات التعلم الآلي في كشف عن هجمات اختراق الشبكات

نادية عطية القدار[1]، ميلاد محمد الضعيف[2*]، الهادي علي كليب[3]

[1] الأكاديمية الليبية للدراسات العليا، طرابلس، ليبيا
[2] قسم تقنيات الإنترنت، كلية تقنية المعلومات، جامعة المرقب، الخمس، ليبيا
[3] الهيئة الليبية للبحث العلمي، طرابلس، ليبيا

*** لمراسلة المؤلف:** mmaldaeef@elmergib.edu.ly

استلمت: 04 مارس، 2026	قبلت: 15 مارس، 2026	نشرت: 25 مارس، 2026

الملخص:
تعد شبكات الحاسوب ركيزة أساسية لمجتمع المعلومات المعاصر، فهي ضرورية لتوفير خدمات حيوية كالتجارة الإلكترونية، الحوسبة السحابية، أنظمة الرعاية الصحية، والبنى التحتية الذكية. تلعب هذه الخدمات دوراً محورياً في تحسين نمط الحياة وطرق التواصل. مع ذلك، هذه الخدمات عرضة لأنواع مختلفة من الهجمات الإلكترونية مثل هجوم حجب الخدمة الموزع (DDoS)، هجمات التخمين العشوائي لكلمات المرور، شبكات البوت أو الروبوت، وهجمات الويب. صممت أنظمة كشف التسلل (IDSs)، كما يوحي اسمها، لمراقبة حركة البيانات في الشبكة وكشف أي تسلل أو أنشطة ضارة، حيث تصنف هذه الأنظمة إلى نوعين رئيسيين: أنظمة كشف التسلل القائمة على البصمة، وأنظمة كشف التسلل القائمة على الشذوذ. تعتمد أنظمة كشف التسلل القائمة على البصمة على مجموعة من القواعد المحددة مسبقاً وتستطيع كشف أنواع الهجمات المعروفة. أما أنظمة كشف التسلل القائمة على الشذوذ، فتستطيع كشف أنواع جديدة وغير معروفة مسبقاً من الهجمات. تتناول هذه الورقة البحثية كيفية توظيف تقنيات التعلم الآلي في أنظمة كشف التسلل لفرز حركة البيانات الشبكية والكشف التلقائي عن هجمات التسلل. تم تقييم ومقارنة أربع خوارزميات للتعلم الخاضع للإشراف والتحكم، وهي: شجرة القرار (DT)، آلة المتجهات الداعمة (SVM)، خوارزمية أقرب الجيران (KNN)، والانحدار اللوجستي (LR)، باستخدام مجموعة بيانات CICIDS2017 المعيارية. تتضمن هذه المجموعة من البيانات حركة مرور شبكية واقعية تشمل أنشطة المستخدمين المشروعة وأنواعاً مختلفة من الهجمات الإلكترونية. تم تطبيق مسار المعالجة المسبقة على البيانات، حيث تتضمن خطوات المعالجة تنظيف البيانات، وتحليل ارتباط الميزات، وتقليل الأبعاد، ومعالجة عدم توازن الفئات باستخدام نهج هجين يجمع بين تقليل حجم العينة وتقنية زيادة حجم الأقلية الاصطناعية (SMOTE). تم قياس أداء نموذج التصنيف باستخدام معايير أداء مختلفة مثل الدقة، الضبط، الاستدعاء، مقياس F1، مصفوفة الارتباك، ووقت التنفيذ. تشير النتائج إلى أن خوارزمية شجرة القرار (DT) يقدم نتائج أفضل بدقة 99.94%، مما يدل على ملاءمته لأنظمة كشف التسلل (IDSs).

الكلمات المفتاحية: أنظمة كشف التسلل، التعلم الآلي، خوارزميات التصنيف، أمن الشبكات.

1. Introduction

Computer networks form the backbone of today's digital society, supporting critical services such as e-commerce, cloud computing, healthcare systems, and smart infrastructures. While these services play a significant role in improving quality of lifestyle and connectivity, they also increase vulnerability to cyberattacks. Network-based attacks such as distributed

denial-of-service (DDoS), brute-force login attempts, botnet activities, and web-based exploits continue to evolve in both scale and complexity.

Intrusion Detection Systems (IDSs) are designed to monitor network traffic and identify unauthorized or malicious activities [1]. IDSs are commonly categorized into signature-based and anomaly-based systems. Signature-based IDSs rely on predefined attack patterns and are effective only against known threats. In contrast, anomaly-based IDSs detect deviations from normal network behavior, enabling the identification of unknown and zero-day attacks [2].

Machine learning (ML) techniques have become central to anomaly-based intrusion detection due to their ability to automatically learn traffic patterns from large datasets. ML-based IDSs reduce manual rule definition and improve detection accuracy. However, the effectiveness of an IDS heavily depends on the selected learning algorithm, dataset quality, class balance, and computational efficiency [3], [4], [5]. Several studies reported detection accuracies above 98%, however, many of them suffer from notable limitations including; 1) class imbalance which lead to biased classification models, 2) limited evaluation metrics focusing only on the accuracy, 3) high computational complexity unsuitable for real-time deployment [6]. This study aims to compare the performance of four classical ML algorithms under consistent experimental conditions, focusing on both detection accuracy and real-time performance requirements. Evaluated algorithms include, Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression (LR).

2. Literature Review

Extensive research has investigated the use of ML techniques for network intrusion detection and attack classification. Early studies relied on traditional datasets which are now considered outdated due to unrealistic traffic patterns and redundant records. More recent studies utilize modern datasets such as UNSW-NB15, CSE-CIC-IDS2018, and CICIDS2017 that provide more realistic attack scenarios and richer feature sets. Researchers have applied a variety of algorithms, including DT, SVM, KNN, LR, Neural Networks, and Deep Learning models to identify malicious traffic, classify attack types and improve detection accuracy.

In [7], KNN classification and k-means clustering were applied to the CIDDS-001 dataset to evaluate NIDs. Results show that anomaly-based NIDS outperform signature-based approaches in detecting new attacks, and that traditional datasets such as KDD99 are insufficient. KNN achieved nearly 99% accuracy with a low false positive rate, highlighting the effectiveness of modern datasets and distance-based methods. However, the study did not address the issue of data imbalance.

Researchers in [8] proposed a federated learning-based approach for wireless intrusion detection using the AWID dataset, combining Federated Averaging (FedAvg) with LR for classification and stacked autoencoders for anomaly detection. The study demonstrated that federated learning can achieve good accuracy, 88% - 95% while preserving privacy and reducing communication and computation costs through collaborative model training on edge devices. However, the work was limited to a simulated environment, used a small set of algorithms, and did not address data imbalance, focusing mainly on accuracy metric.

In [9], researchers proposed an IDS for smart home networks to detect traffic anomalies at the IoT gateway using the Artificial Immune System-Extreme Learning Machine (AIS-ELM) approach. A clonal algorithm was applied to optimize input parameters, while ELM was used for anomaly detection. The system was evaluated on a custom dataset collected from a Mozilla Gateway-based smart home using a Raspberry Pi. Results show about 99% accuracy. However, the study has several limitations, including a small and limited dataset, restricted attack types, insufficient analysis of class imbalance, and the absence of detection time evaluation.

A study in [10] evaluated several ML techniques for intrusion detection in IoT networks through empirical comparisons. DT, RF, Naïve Bayes, and SVM algorithms were evaluated using feature selection methods, including Pearson correlation and Fisher score, as well as feature extraction using Principal Component Analysis (PCA) on the NSL-KDD dataset. Results show that a DT combined with the Fisher score achieved the best performance with 99.26% accuracy and a prediction time of 0.4 seconds. Despite these results, the study is limited by its lack of explicit handling of class imbalance and its reliance on the NSL-KDD dataset, which has limited realism for contemporary network environments.

Researchers in [11] presented a ML-based IDS to monitor network performance and identify abnormal activities. The proposed model involved data acquisition, preprocessing, and feature selection to obtain a relevant subset of features. After which the refined dataset was analyzed using the Konstanz Information Miner KNIME analytics platform. Three classifiers, SVM, Resilient Backpropagation, and DT were evaluated using the CICIDS2017 dataset. Experimental results show that the model effectively improved detection accuracy and reduced false alarms, achieving a maximum accuracy of 98.6% and an average accuracy of 90.59%. These

findings highlight the potential of ML techniques in intrusion detection. Nevertheless, the study has several limitations including, the lack of explicit handling of class imbalance, the absence of detailed per-class performance analysis, and no evaluation of detection time.

In [12], the researchers proposed a Realtime Intrusion Detection System (RIDS) for enterprise networks using the WPA3 protocol to address security vulnerabilities in WPA2 and WPA3. A new WPA3-specific attack dataset was generated using a testbed network, and a two-stage IDS architecture was introduced, where lightweight detection is performed at the access point and suspicious traffic is forwarded to a WLAN controller for ML-based analysis. LR, DT, and RF classifiers were evaluated. DT has achieved the best performance at 99.98% accuracy and a false positive rate of 0.00054. Although the results demonstrated strong detection capability, the experimental is limited by the narrow range of attack types, the lack of class imbalance handling, restricted evaluation metrics, and the absence of per-class performance analysis.

The study in [13] proposed a hybrid intrusion detection approach that integrates network-based and host-based IDSs using ML to improve detection rate, particularly for advanced persistent threats. It transforms host data with the standard Bidirectional Encoder Representations from Transformers (BERT) model, combines host and network features, and uses a two-stage classifier, binary filtering plus multi-class classification. The proposed approach was evaluated on CICIDS 2018 and NDSec-1 datasets. It improved macro F1 scores by 8.1% and 3.7%, respectively, and significantly boosts detection for certain attacks such as DoS-LOIC-UDP and DoS-SlowHTTPTest. However, the study limitations include the lack of systematic handling of data imbalance, limited class-level analysis, omission of rare categories, and no measurement of detection time.

In [14], the researchers evaluated how well different ML models and feature extraction techniques generalize across multiple NIDS datasets. It compared deep learning methods (Deep Feed Forward, CNN, RNN) and shallow learning methods (DT, LR, Naive Bayes), alongside feature extraction approaches such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and autoencoder. The approach was tested on UNSW-NB15, ToN-IoT, and CSE-CIC-IDS2018 datasets. Results show that performance varies significantly, and no single model-feature combination consistently performs best across all datasets. The findings emphasized that dataset choice strongly affects detection performance and highlight the need for a universal feature set. Remaining limitations include weak handling of rare classes, lack of evaluation against adversarial attacks, and no measurement of detection time.

Overall, the reviewed studies demonstrate that ML-based IDSs consistently achieve high accuracy and outperform traditional approaches, particularly when trained on modern datasets. However, many works share common limitations, including inadequate handling of class imbalance, limited attack diversity, and insufficient evaluation metrics such as detection time and per-class performance. In addition, reliance on specific datasets or simulated environments raises concerns about generalizability and real-world applicability. These gaps highlight the need for more robust, balanced, and comprehensive evaluation frameworks in future intrusion detection research. This study contributes to existing literature by performing a balanced, metric-rich, and time-aware comparative evaluation using a realistic dataset.

3. Methodology

This section outlines the steps taken to evaluate ML algorithms for intrusion detection. It describes the dataset selection, preprocessing procedures,

feature optimization, and class balancing techniques applied to ensure data quality and reliable model training. It also presents the experimental design, including dataset partitioning and implementation conditions, followed by the description of the selected classifiers. This structured methodology ensures fair comparison, reduces bias, and provides a robust basis for assessing model performance in terms of accuracy, generalization, and computational efficiency.

3.1 The Used Dataset

The CICIDS2017 dataset which is available online at [15] was developed by the Canadian Institute for Cybersecurity to provide realistic network traffic for intrusion detection research, and it was collected over multiple days in a controlled environment designed to simulate real-world network behavior. It contains 543,591 records with 88 features representing both benign and malicious traffic. It includes diverse attack categories such as DoS/DDoS, port scanning, brute-force attempts, web attacks, and botnet activity. The dataset integrates flow-based statistical attributes with protocol-level information, making it well suited for ML-based classification tasks.

CICIDS2017 dataset was preprocessed before being used for training and testing models. Data preprocessing is a crucial stage because raw network traffic data often contain noise, redundancy, and class imbalance which occur when benign traffic instances vastly outnumber malicious ones causing ML models to become biased toward the majority class [16]. To ensure data quality, several steps were applied to clean the dataset from such issues. These steps include the removal of missing and infinite values, elimination of duplicate records, normalization of numerical attributes, and encoding of categorical features. In addition, feature reduction was performed through correlation analysis to identify highly correlated

attributes, allowing redundant features to be removed in order to reduce dimensionality and improve training efficiency. Since intrusion detection datasets typically contain a much larger proportion of benign traffic than attack traffic, class imbalance can negatively affect classifier performance. To address this issue, a hybrid balancing approach was used, it represented in combining downsampling of majority classes with the application of Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for minority classes [17]. This has improved class representation and enhanced the system's ability to detect rare attacks. This also helped in comparing multiple ML algorithms to evaluate their effectiveness in intrusion detection and identify the most suitable model based on accuracy, robustness, and computational efficiency.

In this study, four supervised ML classifiers were implemented, they are, DT, SVM, KNN, and LR. These algorithms were evaluated under identical experimental conditions to ensure a fair, unbiased, and the comparison of their performance in network intrusion detection. These models were selected because they represent diverse learning paradigms and have been widely applied in intrusion detection research due to their proven effectiveness and complementary strengths. To ensure reliable evaluation and robust generalization, the processed dataset was partitioned into three subsets: 70% for training, 15% for validation, and 15% for testing as illustrated in Table I. This structured division allows the models to learn patterns from the training data, fine-tune parameters using the validation subset, and undergo final performance assessment on an independent test subset that was not exposed during the training phase. This strategy helps in reducing overfitting, enhancing model reliability, and providing a realistic estimate of real-world performance.

TABLE I. RESULTS FOR DATA SPLIT

Split type	percentage	Approximate size
Training data	70%	24500
Validation data	15%	5250
Testing data	15%	5250

Each classifier was implemented according to its underlying learning principle, following subsections introduced and describe each classifier individually.

3.2 Decision Tree (DT)

DT classifier is a supervised ML model that works as a hierarchical, rule-based structure for classification. It builds a tree-like model by repeatedly splitting the dataset into smaller subsets based on the values of input features.

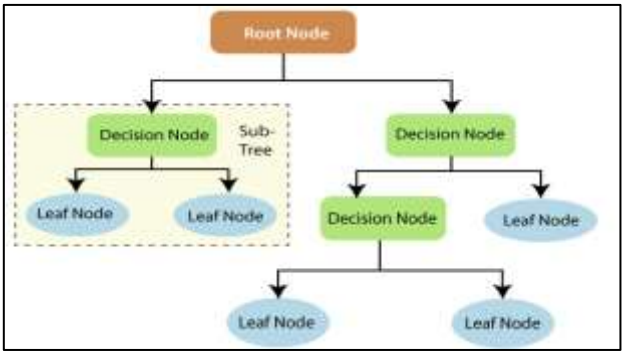

Fig 1. Decision Tree [18]

At each step, the algorithm selects the feature that best separates the data into different classes, creating decision rules that guide the classification process. This recursive splitting allows the model to learn complex and nonlinear relationships between variables while still remaining easy to understand. One of the main advantages of DTs is their high interpretability, as the final model can be visualized as a set of clear conditional -if then-

rules. This transparency makes DT especially useful in security applications such as IDSs where it is important to understand how and why a particular decision was made.

3.3 Support Vector Machine (SVM)

SVM is a margin-based classification algorithm that works by finding the best possible boundary, called a hyperplane, that clearly separates data points belonging to different classes. It chooses this boundary so that the distance between the classes is as wide as possible, which helps the model make more accurate predictions on new data. This approach makes SVM especially useful for complex problems with many features, such as network traffic datasets where each data record may contain a large number of variables that need to be analyzed at the same time.

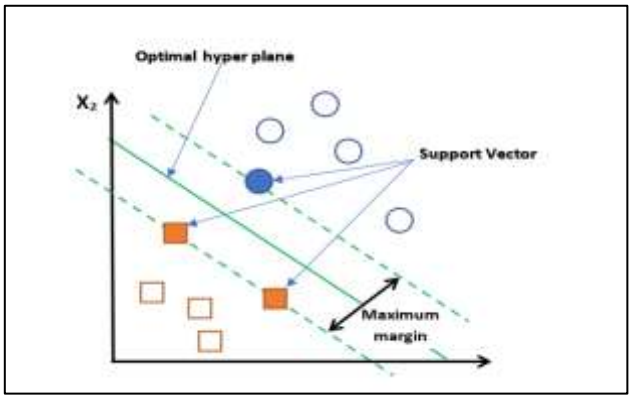

Fig 2. Support Vector Machine [19]

In this study, SVM is implemented to identify network intrusions; the SVC function from Scikit-learn was used with default parameters. Despite their effectiveness, the generalization ability of SVM models is significantly affected by the choice of parameters, including the regularization parameter (C), the kernel function, and the gamma parameter (γ) [20]. SVM belongs to

linear learning that uses a plane to classify instances into varying categories. A hyperplane can be used as a prediction, as presented in Equation (1) [21].

$$g(x) = \begin{cases} +1, \ if \ w.x + b \geq 0 \\ -1, \ if \ w.x + b \leq \ 0 \end{cases} \qquad (1)$$

The class is predicted as normal if g(x) is more significant than zero; otherwise, the class is predicted as abnormal if g(x) is less than zero.

3.4 K-Nearest Neighbors (KNN)

KNN algorithm is a distance-based classification method that determines the class of a new instance by analyzing the labels of the closest data points in the feature space. It measures similarity using a distance function, often the Euclidean distance, typically denoted as D (a, b) as shown in Equation (2) [22] where each feature of one instance is compared with the corresponding feature of another across all r features in the dataset. Based on these computed distances, the algorithm assigns the class that appears most frequently among the selected nearest neighbors. Unlike model-based techniques, KNN does not build an explicit predictive model; instead, it directly compares new data with stored training examples, making it simple to understand and easy to implement. Because it relies on this instance-based, non-parametric approach and does not construct a decision boundary in advance, KNN is particularly effective for pattern recognition tasks involving complex or irregular class boundaries that cannot be separated using simple linear rules.

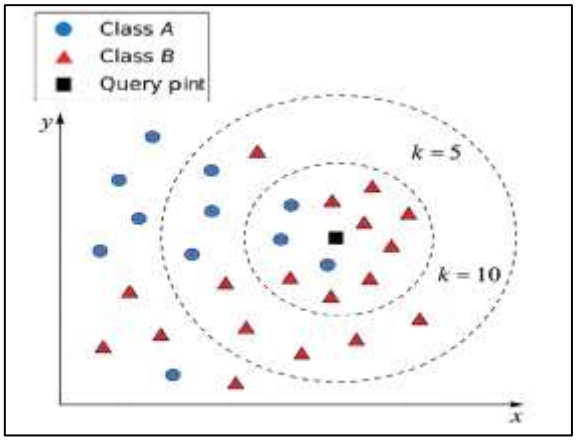

Fig 3. K-Nearest Neighbors (K-NN) [23]

In Equation (2), the i^{th} featured element of the instance a, bi is the i^{th} featured element of the instance b, while r represent the entire feature quantity of the dataset.

$$D(a,b)=\sqrt{\sum_{i=1}^{r}(ai - bi)} \quad (2)$$

3.5 Logistic Regression (LR)

LR is a statistical learning algorithm used for classification tasks that works well when the relationship between input features and class labels can be separated using a linear decision boundary. It estimates the probability that a data point belongs to a particular class by applying a logistic function to a weighted combination of input features. Because of its simple mathematical structure, LR is computationally efficient and can be trained quickly even on large datasets, making it a practical choice for real-time or large-scale applications. This efficiency, along with its interpretability and stable performance, makes LR especially useful in problems where fast prediction and clear understanding of feature influence are important. Mathematically, LR computes the weighted sum of input features and adds a bias term b. The resulting value z represents the linear combination of the feature vector x and weight vector w, as shown in Equation (3) [24].

$$z = w \cdot x + b \quad (3)$$

In Equation (3), nothing forces z to be a legal probability, that is, to lie between 0 and 1. In fact, since weights are real-valued, the output might even be negative, z ranges from $-\infty$ to ∞.

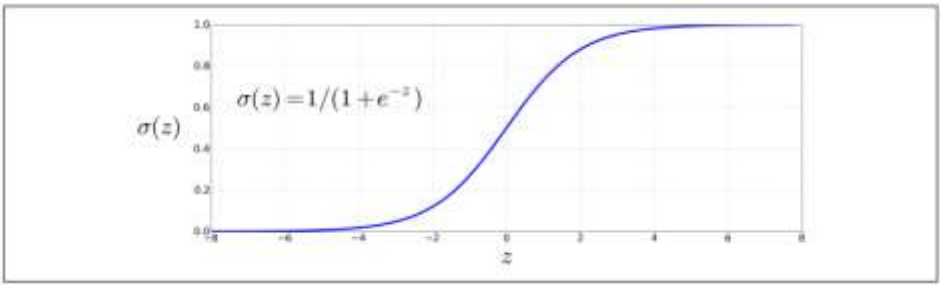

Fig 4. The sigmoid function [24]

To convert the value z into a probability, we apply the sigmoid function, also known as the logistic function, which gives LR its name. The sigmoid function (4), takes a real value and maps it to the range (0,1). It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

$$\sigma(Z)=\frac{1}{1+e^{-Z}} \quad (4)$$

By training and testing these four models under consistent conditions, the study enables a comprehensive comparative analysis to determine which algorithm achieves the best balance between detection accuracy, generalization capability, and computational efficiency for intrusion detection applications.

4. Research Design

The proposed approach consists of a set of sequence steps, starting with the original dataset, followed by data preprocessing. The dataset is then divided into three subsets, including training, validation, and testing. After that, four models are trained and evaluated, after which the best-performing model is

selected to classify the network traffic as either normal or malicious. These steps are illustrated in Figure 5.

4.1 Models Training

The four selected ML algorithms, DT, SVM, KNN, and LR, were trained on the prepared training subset to learn classification patterns that distinguish between normal and malicious network traffic. The training phase involved learning from labelled instances to build a predictive model capable of identifying intrusions.

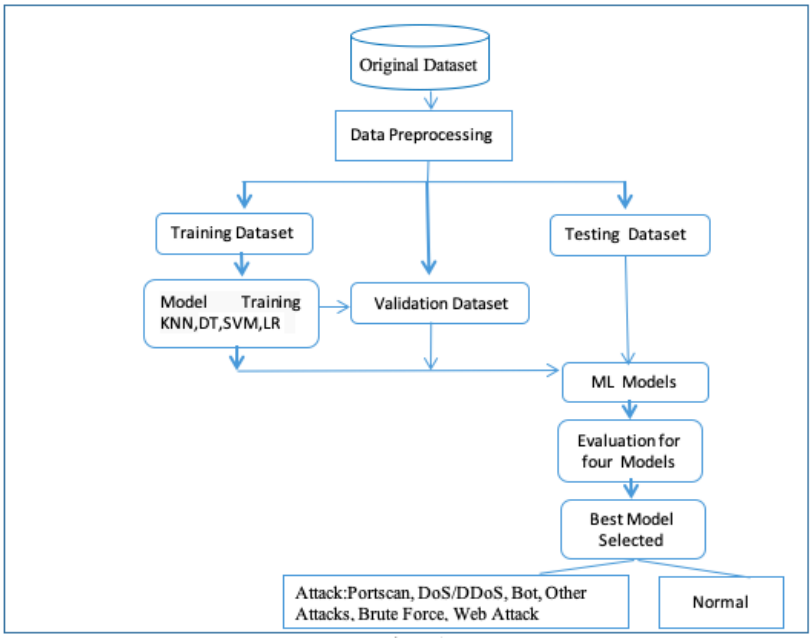

Fig 3. The sigmoid function [24]

After training, the models were validated using a separate validation subset to fine-tune parameters and reduce the risk of overfitting. This multi-model learning allows for comparative analysis of the performance of each classifier under equivalent conditions with the aim of identifying the algorithm that produces the most accurate and generalizable results for network intrusion detection. Subsequently, every trained model was

assessed using unseen test dataset -data that wasn't used during training and validation phases- to ensure fair and unbiased assessment of model performance.

4.2 Evaluation of the Four Algorithms

The performance of each ML model was assessed using both validation and testing datasets. The validation set was used during training to monitor generalization and adjust hyperparameters, while the unseen testing set was reserved for final evaluation. Multiple metrics include; accuracy, precision, recall, specificity, and F1 score [25], [26] were calculated to provide a comprehensive assessment since relying on accuracy alone is insufficient for IDSs. Using consistent evaluation criteria across all models allowed for a detailed comparison of their ability to correctly classify normal and attack traffic and to identify which algorithm achieved the best balance between detection performance and generalizability.

4.3 Selecting the Best Algorithm

After evaluating all models, the best-performing one was selected based on accuracy, precision, recall, and F1 score measured on both validation and testing datasets. This model was chosen for its ability to reliably detect intrusions, consistent with prior studies emphasizing the selection of top-performing algorithms [27], [28]. The study's goal was to compare multiple ML algorithms to achieve accurate, real-time intrusion detection and identify the model most capable of distinguishing between normal and malicious traffic.

4.4 Performance Measures

The confusion matrix used to generate a multi-class classification dataset, serves as the foundation to compute the key performance measures. It

provides a summary of classification results by presenting the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These components are essential for deriving statistical performance metrics, including accuracy, precision, recall, and F1-score, that are used to evaluate the classifier's effectiveness. The confusion matrix for the multi-class classification task is presented in Table II.

TABLE II. CONFUSION MATRIX OF A MULTI-CLASS CLASSIFICATION ALGORITHMS

Actual \ Predicted	Normal	Portscan	DoS/DDoS	Bot	Other Attacks	Brute Force	Web Attack
Normal	TP	FP	FP	FP	FP	FP	FP
Portscan	FP	TP	FP	FP	FP	FP	FP
DoS/DDoS	FP	FP	TP	FP	FP	FP	FP
Bot	FP	FP	FP	TP	FP	FP	FP
Other Attacks	FP	FP	FP	FP	TP	FP	FP
Brute Force	FP	FP	FP	FP	FP	TP	FP
Web Attack	FP	FP	FP	FP	FP	FP	TP

After the four possible outcomes of a classification model were introduced, it is important to understand their practical meanings and implications. TP refers to cases that belong to a class and are correctly classified, while FP are cases that do not belong to a class but are incorrectly labeled as such. TN represents instances correctly identified as not belonging to a class, and FN are instances that belong to a class but are wrongly classified as another. Together, these four measures describe how well a model performs in classification tasks.

The performance metrics now can be calculated using these four outcomes as following [29], [30]. The accuracy is calculated as in Equation (5). It measures the ability of classifier to diagnose classes in the dataset.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (5)$$

The recall or sensitivity measure is computed as in Equation (6). It indicates the accuracy measure of the target class's occurrence.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (6)$$

The precision measure is computed as in Equation (7) to measure the probability that a positive prediction is correct.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (7)$$

Lastly, the F-Score measure is calculated as in Equation (8). It is characterized by striking a balance between the proportion of the correct values expected for precision (the model's accuracy) and the proportion of the correct values expected to recall (the extent of the model's completeness).

$$\text{F1-Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (8)$$

5. Experimental Results and Result Analysis

This section presents the experimental evaluation of four ML classifiers - DT, SVM, KNN, and LR – that were trained and tested on the CIC-IDS2017 dataset using a 70%, 15%, and 15% split, as illustrated in section number and Table I. Prior to training, several preprocessing steps were applied, including correlation-based feature reduction, class balancing using downsampling and SMOTE, and feature scaling for SVM, all of which enhanced model reliability and reduced overfitting.

The results indicate that the DT classifier achieved the highest performance, attaining 99.94% testing accuracy with very low training and testing times. This was followed by SVM, which achieved 99.58% accuracy. KNN also

showed strong performance with 98.97% accuracy but required longer prediction time, whereas LR recorded the lowest accuracy, 89.26%, and the longest training time. Table III. and Table IV demonstrate that DT and SVM consistently maintained high precision, recall, and F1-scores across attack categories with minimal misclassification.

Table III highlights clear performance differences in accuracy, recall, precision, F1-score, and computational efficiency. Overall, DT achieved the best balance between predictive performance and execution speed, followed closely by SVM, while KNN showed competitive accuracy but suffered from high computational cost during testing. LR showed fast execution but limited capability in identifying complex attack patterns, thus ranked as lowest in overall effectiveness. These findings indicate that tree-based and margin-based classifiers are particularly well suited for complex intrusion detection tasks, with the DT model emerging as the most practical candidate for real-time IDSs deployment due to its superior accuracy, stability, and efficiency. Class-level analysis as illustrated in Table IV, indicated that DT consistently outperformed other classifiers in detecting minority attack categories. Table V shows the aggregated confusion-matrix metrics for all classifiers across all classes.

TABLE III. OVERALL PERFORMANCE OF ALL CLASSIFIERS

Classifier	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Testing Time (s)	Rank
DT	99.94%	99.94%	99.94%	99.94%	0.4092	0.0119	1st
SVM	99.58%	99.58%	99.58%	99.58%	2.6790	0.6911	2nd
KNN	98.97%	98.97%	98.97%	98.97%	0.0551	1.4974	3rd
LR	89.26%	90.83%	89.26%	89.07%	61.4226	0.0079	4th

TABLE IV. CLASS-LEVEL ANALYSIS ACROSS ALL CLASSIFIERS

Metric \ Class	Bot	Brute	DoS/DDoS	Normal	Other	Portscan	Web
DT Accuracy	100%	99.96%	99.94%	99.96%	99.96%	99.92%	99.98%
DT Precision	100%	99.73%	99.73%	99.87%	99.87%	99.87%	100%
DT Recall	100%	100%	99.87%	99.87%	99.87%	99.60%	99.87%
DT F1	100%	99.86%	99.90%	99.87%	99.87%	99.73%	99.93%
SVM Accuracy	99.90%	99.92%	99.87%	99.71%	99.89%	99.92%	99.98%
SVM Precision	99.60%	99.87%	99.73%	98.29%	100%	99.87%	99.87%
SVM Recall	99.73%	99.60%	99.33%	99.73%	99.20%	99.60%	100%
SVM F1	99.67%	99.73%	99.53%	99.01%	99.60%	99.73%	99.93%
KNN Accuracy	99.66%	99.92%	99.73%	99.24%	99.81%	99.73%	99.81%
KNN Precision	98.16%	99.87%	98.94%	97.72%	99.20%	99.73%	99.07%
KNN Recall	99.47%	99.60%	99.20%	96.93%	99.47%	98.40%	99.60%
KNN F1	98.81%	99.73%	99.07%	97.32%	99.33%	99.06%	99.33%
LR Accuracy	94.63%	99.90%	99.49%	94.72%	95.09%	96.59%	99.01%
LR Precision	77.08%	99.73%	97.88%	93.08%	74.65%	100%	95.68%
LR Recall	88.80%	99.60%	98.53%	68.13%	99.33%	76.13%	97.47%
LR F1	82.53%	99.67%	98.21%	78.68%	85.24%	86.45%	96.57%

TABLE V. AGGREGATED CONFUSION-MATRIX METRICS FOR ALL CLASSIFIERS ACROSS ALL CLASSES

	DT				SVM				KNN				LR			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Bot	750	4500	0	0	748	4497	3	2	746	4486	14	4	666	4302	198	84
Brute Force Attack	750	4498	2	0	747	4499	1	3	747	4499	1	3	747	4498	2	3
DoS/ DDoS Attack	749	4498	2	1	745	4498	2	5	744	4492	8	6	739	4484	16	11

	DT				SVM				KNN				LR			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Normal	749	4499	1	1	748	4487	13	2	727	4483	17	23	511	4462	38	239
Other Attacks	749	4499	1	1	744	4500	0	6	746	4494	6	4	745	4247	253	5
Portscan	747	4499	1	3	747	4499	1	3	738	4498	2	12	571	4500	0	179
Web Attack	749	4500	0	1	750	4499	1	0	747	4493	7	3	731	4467	33	19

6. Conclusion

The experimental results demonstrate that balanced datasets and lightweight algorithms are essential for practical intrusion detection. Proper preprocessing of utilized dataset and class balancing significantly enhance detection reliability and prevent bias toward majority classes, which otherwise leads to poor recognition of rare intrusions. Furthermore, detection accuracy alone is insufficient for real-world deployment since computational efficiency and response time are equally critical in high-speed network environments. Using the CICIDS2017 dataset, four classical ML classifiers were evaluated under identical experimental conditions, confirming that traditional models remain highly effective for intrusion detection tasks. Among them, the DT classifier achieved superior performance in terms of accuracy, robustness, and computational efficiency, providing the best trade-off between detection capability and execution speed.

Future research will focus on enhancing the comparison of classifiers by integrating ensemble and deep learning models to improve performance and robustness. Evaluations will be extended across multiple datasets to assess generalizability, while feature optimization will be explored to maximize classifier efficiency. Additionally, the implementation of real-time testing

will provide insights into the practical effectiveness of these classifiers under dynamic conditions.

References

[1] S. Kannadhasan and R. Nagarajan, "Intrusion detection in machine learning based E-shaped structure with algorithms, strategies and applications in wireless sensor networks," Heliyon, vol. 10, no. 9, p. e30675, May 2024, doi: 10.1016/j.heliyon.2024.e30675.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," J. Inf. Secur. Appl., vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[3] R. O. Ogundokun, J. B. Awotunde, P. Sadiku, E. A. Adeniyi, M. Abiodun, and O. I. Dauda, "An Enhanced Intrusion Detection System using Particle Swarm Optimization Feature Extraction Technique," Procedia Comput. Sci., vol. 193, pp. 504–512, 2021, doi: 10.1016/j.procs.2021.10.052.

[4] V. Ganesh, M. Sharma, and S. K. Henge, "Particle Swarm Optimization Feature Extraction Technique for Intrusion Detection System," Jan. 04, 2023, In Review. doi: 10.21203/rs.3.rs-2412032/v1.

[5] N. Ben Henda, A. Msolli, I. Haggui, A. Helali, and H. Maaref, "Attack Detection in IoT Network Using Support Vector Machine and Improved Feature Selection Technique," J. Netw. Syst. Manag., vol. 32, no. 4, p. 92, Oct. 2024, doi: 10.1007/s10922-024-09871-3.

[6] E. A. Al-Qarni and G. A. Al-Asmari, "Addressing Imbalanced Data in Network Intrusion Detection: A Review and Survey," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 2, 2024, doi: 10.14569/IJACSA.2024.0150215.

[7] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning," Procedia Comput. Sci., vol. 125, pp. 709–716, 2018, doi: 10.1016/j.procs.2017.12.091.

[8] B. Cetin, "Wireless Network Intrusion Detection and Analysis using Federated Learning.," Master's Thesis, Youngstown State University, 2020.

[9] E. Alalade, D. ,., "Intrusion Detection System in Smart Home Network Using AIS/ELM Hybrid Approach," University of Cincinnati, 2020.

[10] M. Baich, T. Hamim, N. Sael, and Y. Chemlal, "Machine Learning for IoT based networks intrusion detection: a comparative study," Procedia Comput. Sci., vol. 215, pp. 742–751, 2022, doi: 10.1016/j.procs.2022.12.076.

[11] A. S. Jaradat, M. M. Barhoush, and R. S. B. Easa, "Network intrusion detection system: machine learning approach," Indones. J. Electr. Eng. Comput. Sci., vol. 25, no. 2, p. 1151, Feb. 2022, doi: 10.11591/ijeecs.v25.i2.pp1151-1158.

[12] R. Saini, D. Halder, and A. M. Baswade, "RIDS : Real-time Intrusion Detection System for WPA3 enabled Enterprise Networks," Jul. 06, 2022, arXiv: arXiv:2207.02489. doi: 10.48550/arXiv.2207.02489.

[13] Z. Chen, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Machine learning-enabled hybrid intrusion detection system with host data transformation and an

advanced two-stage classifier," Comput. Netw., vol. 250, p. 110576, Aug. 2024, doi: 10.1016/j.comnet.2024.110576.

[14] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in IoT networks," Digit. Commun. Netw., vol. 10, no. 1, pp. 205–216, Feb. 2024, doi: 10.1016/j.dcan.2022.08.012.

[15] "https://www.unb.ca/cic/datasets/ids-2017.html." [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[16] M. Tahir, A. Abdullah, N. I. Udzir, and K. A. Kasmiran, "A novel approach for handling missing data to enhance network intrusion detection system," Cyber Secur. Appl., vol. 3, p. 100063, Dec. 2025, doi: 10.1016/j.csa.2024.100063.

[17] P. Verma et al., "A Novel Intrusion Detection Approach Using Machine Learning Ensemble for IoT Environments," Appl. Sci., vol. 11, no. 21, p. 10268, Nov. 2021, doi: 10.3390/app112110268.

[18] B. Jijo T, and A. Abdulazeez M, "Classification Based on Decision Tree Algorithm for Machine Learning," J. Appl. Sci. Technol. Trends, vol. 2(1), pp. 20–28, 2021.

[19] C. Savas and F. Dovis, "The Impact of Different Kernel Functions on the Performance of Scintillation Detection Based on Support Vector Machines," Sensors, vol. 19, no. 23, p. 5219, Nov. 2019, doi: 10.3390/s19235219.

[20] F. Nie, Z. Hao, and R. Wang, "Multi-Class Support Vector Machine with Maximizing Minimum Margin," Proc. AAAI Conf. Artif. Intell., vol. 38, no. 13, pp. 14466–14473, Mar. 2024, doi: 10.1609/aaai.v38i13.29361.

[21] Z. K. Maseer, Q. K. Kadhim, B. Al-Bander, R. Yusof, and A. Saif, "Meta-analysis and systematic review for anomaly network intrusion detection systems: Detection methods, dataset, validation methodology, and challenges," IET Netw., vol. 13, no. 5–6, pp. 339–376, Sep. 2024, doi: 10.1049/ntw2.12128.

[22] ChitkaraUniversity and D. Kaur, "A Comparative Study of Various Distance Measures for Software fault prediction," Int. J. Comput. Trends Technol., vol. 17, no. 3, pp. 117–120, Nov. 2014, doi: 10.14445/22312803/IJCTT-V17P122.

[23] K. Taha, "Big Data Analytics in IoT, social media, NLP, and information security: trends, challenges, and applications," J. Big Data, vol. 12(1), p. 150, Jun. 2025, doi: 10.1186/s40537-025-01192-9.

[24] D. Jurafsky and J. Martin, Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition with language models, 3rd edn draft. Stanford University, 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/?utm_source=chatgpt.com

[25] V. Vandana and R. Verma, "Evaluating Effectiveness: A Critical Review of Performance Metrics in Intrusion Detection System," J. Eng. Sci. Technol. Rev., vol. 18, no. 1, pp. 199–209, 2025, doi: 10.25103/jestr.181.20.

[26] J. Liedgren, "Comparison of machine-learning algorithms for intrusion detection systems," Master's Thesis, Stockholm University, https://su.diva-portal.org/smash/get/diva2:1971980/FULLTEXT01., 2025.

[27] S. Singhal, "Comparative analysis of traditional machine learning models for network intrusion detection," 2025.

[28] I. Acheme, D. and A. Wasiu, A., "A Comparative Study of Machine Learning Algorithms Used for Network Intrusion Detection," vol. 8(1), pp. 494–500, 2024.

[29] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," Aug. 13, 2020, arXiv: arXiv:2008.05756. doi: 10.48550/arXiv.2008.05756.

[30] G. Zeng, "Invariance Properties and Evaluation Metrics Derived from the Confusion Matrix in Multiclass Classification," Mathematics, vol. 13, no. 16, p. 2609, Aug. 2025, doi: 10.3390/math13162609.