International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

PERFORMANCE EVALUATION OF MACHINE LEARNING-BASED INTRUSION DETECTION USING NSL-KDD, UNSW-NB15 AND CICIDS2017 DATASETS

Dasganu Govindrao Hakke [1*], Aniket Yadneshwar Dixit [2], Dr. Suryakant Thorat[3], Gouravkumar S Malande [4], Arvind K Panpatte [5]

[1*] Assistant Professor, Dr. D. Y. Patil Technical Campus, Varale, Talegaon, Pune, Maharashtra, India. Savitribai Phule Pune University, Pune. Orcid Id: https://orcid.org/0009-0004-7503-8449

[2]Research Scholar, School of computational sciences, SRTMU, Nanded, Maharashtra, India. SRT Marathwada University, Nanded, Maharashtra, India. Orcid Id: https://orcid.org/0009-0008-0001-525X

[3]Campus Director, Gaikwad-Patil Group of Institutions, Tulsiramji Gaikwad Patil College of Engineering, Nagpur, Maharashtra, India. Orcid Id: https://orcid.org/0009-0003-1467-0400

[4]Assistant Professor, Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, India. SRT Marathwada University, Nanded, Maharashtra, India. Orcid Id: https://orcid.org/ 0009-0001-6494-5287

[5]Assistant Professor, Department of Computer Science, Central Sanskrit University, Nashik Campus, Maharashtra, India. Orcid Id: https://orcid.org/ 0009-0004-3756-9310

*Corresponding author:

Assistant Professor, Dr. D. Y. Patil Technical Campus,

Varale, Talegaon, Pune, Maharashtra, India.

Orcid Id: https://orcid.org/0009-0004-7503-8449

Email Id: dasganu.hake345@gmail.com

Abstract

Smart devices are now common in almost every field, and many private and public organizations manage their services remotely through web servers and cloud platforms. Sadly, as network demand rises, both systems and users become more at risk. Every day, thousands of cyberattacks happen to many businesses. Traditional intrusion detection systems (IDS) have a hard time keeping up with the number of attacks on networks that are growing. So, it's very important to look at new ways to stop and find intrusions. An IDS can tell the difference

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

between good and bad behavior, and machine learning (ML) can be used to make good IDS models for safe communication. In machine learning, classification algorithms can automatically find patterns in data and sort user activities into normal and intrusive categories. This paper provides a comparative analysis and performance assessment of various ML-based IDS methodologies. The experiments employ established datasets such as NSL-KDD, UNSW-NB15, and CICIDS2017 to train and evaluate various machine learning algorithms, including Support Vector Machines (SVM), Decision Trees, Naïve Bayes, Random Forests, K-Nearest Neighbors (KNN), Logistic Regression, Quadratic Discriminant Analysis (QDA), AdaBoost, CatBoost, Gradient Boosting, LightGBM (LGBM), Linear Discriminant Analysis (LDA), and XGBoost (XGB). Standard metrics like accuracy, precision, recall, and F1-score are used to measure how well these methods work. The results show that the algorithms have very different detection abilities, which shows how important it is to choose the right ML models for intrusion detection.

Keywords: CICIDS2017, intrusion detection, Machine Learning, NSL-KDD, UNSW-NB15

1. Introduction

Today, the internet is becoming more and more important because of how quickly big data, cloud technology, and smart devices are improving. This trend is clear in both the military and the business world, which makes it very important to keep information private, safe, and secure. Even though cybersecurity experts are working hard to stop them, malware and intruders are still trying to get to, change, or destroy sensitive data. This situation is getting worse, so intrusion detection systems (IDS) and intrusion prevention have become important parts of cyber defense. The privacy, integrity, and availability of data must be the top priorities for modern network systems. IDS technology has become a major concern for network security against malicious users.

In the age of big data, traditional ways of finding and stopping attacks often don't work because they can't handle complex and large-scale attacks well. Machine learning (ML) methods have been helpful in finding these kinds of threats, and many ML-based attack detection solutions have been published in the literature. But one of the biggest problems with these ML solutions is that they have to be very good at finding attacks.

An intrusion is any action that tries to compromise the integrity, confidentiality, or availability of a system. This includes any unauthorized activity on a network or device. IDS are important tools that help find, evaluate, and describe behavior that is intrusive. They work as protective systems that can find strange things happening. For example, IDS can tell the difference between good and bad traffic by looking at patterns, like known attack signatures or strange usage patterns. Using data mining techniques can make IDS work better by making them more accurate and faster than older methods. A good IDS usually has a low false positive rate and a high true positive rate.

There are two types of network intruders: (1) external intruders, who are unauthorized outsiders who use different attack methods to get into the network, and (2) internal intruders,

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

who are compromised insiders or infected nodes that used to be legitimate network participants. IDS can find threats from both inside and outside the network. However, it is usually easier to find outside attacks because attackers from inside the network may have valid credentials or knowledge that lets them get around security measures. Espionage, data exfiltration, penetration attacks, and Denial of Service (DoS) attacks are all examples of threats. A good IDS should be able to find all of these kinds of intrusions, but most of the time, they only cover some of them.

You can group IDSs by how they are set up and how they find things. There are two main types of deployment: Host-based Intrusion Detection Systems (HIDS) and Network-based Intrusion Detection Systems (NIDS). HIDS are put on individual hosts to keep an eye on what happens on that host, such as changes to system files, repeated failed login attempts, illegal process behavior, or strange resource usage. They find things by either watching what hosts do in real time or looking at log files after the fact. NIDS, on the other hand, watches network traffic all the time or at regular intervals. It looks at packet headers and payloads to find strange patterns.

IDS can use either misuse-based detection or anomaly-based detection, depending on how it finds things. In misuse-based detection, the IDS looks for activity that matches known attack patterns and signatures in a database. The downside is that new or unknown attacks (for which signatures are not yet available) can get by without being found. On the other hand, anomaly-based detection uses automated training to create a model of "normal" behavior. Activity that is very different from this normal profile is thought to be harmful. Anomaly detection can find new attacks, but it can also give false positives, which are normal activities that are wrongly labeled as anomalies. A third method, which is sometimes called specification-based detection, uses rules or specifications of allowed behavior that were made by hand. Any behavior that doesn't follow these rules is flagged as an intrusion. This method necessitates considerable human effort to formulate specifications and may fail to detect malicious behavior that deviates from established rules.

The main purpose of this article is to compare and evaluate the performance of different ML-based IDS methods. The machine learning algorithms that were looked at are SVM, Decision Trees (DT), Naïve Bayes, Random Forests, KNN, Logistic Regression, Quadratic Discriminant Analysis, AdaBoost, CatBoost, Gradient Boosting, LGBM, LDA, and XGB. We use three well-known benchmark datasets—NSL-KDD, UNSW-NB15, and CICIDS2017—to test these methods. These datasets encompass a diverse array of network traffic scenarios and attack vectors. The structure of this paper is as follows: Section 2 gives a summary of other work that is related to IDS (Literature Review). Section 3 talks about the methods, such as the ML approaches, datasets, and experimental setup. In Section 4, we talk about the results of our experiments, and in Section 5, we give the study's conclusions.

2. Literature review

In recent years, many studies have used machine learning and other methods to find intrusions. **Myint and Meesad [14]** developed an incremental learning algorithm utilizing SVM for

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

intrusion prevention. Their method, ISVMM, makes repeated training easier and faster by using an SVM with a Mahalanobis distance metric. They tested the system on the KDD Cup 99 dataset and said it could predict 41 different kinds of network traffic data.

Peddabachigari et al. [15] put out an IDS that uses a Decision Tree classifier. When tested with the 1998 DARPA dataset, their system worked better than traditional methods and cut down on training and testing time by a large amount compared to an SVM-based approach.

Panda and Patra [16] created a Network-based IDS that uses a Naïve Bayes classifier. When tested on the KDD Cup 99 dataset, their framework did very well at finding intrusions because it had a low false positive rate, took less time to process, and was cheaper.

Farnaaz and Jabbar [17] put forward an ensemble model that uses a Random Forest classifier to find intrusions. They used the NSL-KDD dataset to test their model. The results demonstrated enhanced detection efficacy compared to conventional methods, attaining a minimal false alarm rate and a substantial detection rate.

Al-Qatf et al. [18] put forward a deep learning framework known as STL-IDS, which combines a sparse autoencoder with SVM for detecting intrusions. This method makes it easier for the SVM classifier to learn features automatically and reduce the number of dimensions, which improves its prediction accuracy. Their framework made a big difference in the performance metrics of the IDS.

Li et al. [19] created an IDS for wireless sensor networks that is based on KNN. The authors assert that their methodology proficiently differentiates between normal and abnormal nodes by observing atypical behavioral patterns. They looked at the choices of parameters and the rates of error, showing that the method works better and faster for finding things.

Moustafa et al. [20] developed a Collaborative Anomaly Detection Framework (CADF) for the analysis of large datasets in cloud environments. Their framework, which runs in the cloud, has three parts: (i) capturing and logging network data, (ii) preprocessing the data, and (iii) a decision engine based on Gaussian mixtures. They also use a threshold based on the interquartile range to find attacks. Their method made it easy to set up IDS as a cloud service and improved detection performance when they tested it on the UNSW-NB15 dataset and compared it to three other anomaly detection methods.

Osanaiye et al. [21] suggested an ensemble-based multi-filter feature selection method to improve intrusion detection. They used four feature selection filters to cut down the feature space from 41 to 13 best features. They showed that using a Decision Tree classifier on the NSL-KDD dataset improved detection performance. The authors assert that their method is appropriate for cloud environments to identify DDoS attacks, attaining superior accuracy compared to certain existing techniques.

Diro and Chilamkurti [22] created a decentralized IDS for the Internet of Things (IoT) that uses deep learning. They suggest putting the IDS on a fog computing layer close to the IoT devices. Their system was tested on the NSL-KDD, ISCX 2012, and KDD Cup 99 datasets

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

and got 97% accuracy, which shows that deep learning works well for distributed intrusion detection in IoT networks.

Al-Hawawreh et al. [23] introduced an anomaly detection system specifically designed for Industrial IoT environments. They used an unsupervised deep autoencoder to learn features and classify them. They tested it on the NSL-KDD and UNSW-NB15 datasets. The authors say that their system was able to find malicious activities with 99% accuracy, which means it can find industrial network intrusions very well.

Garg et al. [24] suggested a hybrid deep learning model that combines Gray Wolf Optimization (GWO) and Convolutional Neural Networks (CNN) to find problems in cloud datacenters. They first use a better GWO algorithm to choose features and lower the number of dimensions in the data. Then they train a better CNN to find threats. Their cloud-based threat detection model did better than the baseline methods when tested on the DARPA 1998 and KDD 99 datasets. It found more threats, had fewer false alarms, and was much more accurate overall (8.25% higher detection rate than standard GWO and CNN, for example).

Peng et al. [25] developed an Intrusion Detection System (IDS) utilizing Decision Tree classification, specifically optimized for large-scale data environments (fog computing). They employed a comprehensive dataset to evaluate outcomes and propose the implementation of their IDS within a fog computing environment for real-time big data analysis, despite the absence of testing in real-time situations. Their assessment on a modified version of the KDD Cup 99 dataset indicated encouraging efficacy for intrusion detection in fog networks.

Manimurugan et al. [26] put forward an Intrusion Detection System (IDS) for the Internet of Medical Things (IoMT) that is based on a Deep Belief Network (DBN). Using the CICIDS2017 dataset to test performance, their deep learning model found PortScan attacks with 97.71% accuracy and Infiltration attacks with 96.37% accuracy. This shows that it works well in an IoMT setting with certain types of attacks.

Basati and Faghih [27] introduced a CNN-based Intrusion Detection System (IDS) utilizing a deep feature extraction technique known as DFE. They looked at how well it worked on a number of datasets, such as UNSW-NB15, CICIDS2017, and KDD Cup 99. The suggested method was tested for both binary and multi-class classification tasks. It was made to work well on devices with limited computing power, like IoT devices. Their method worked well and was able to grow.

Rashid et al. [28] created a tree-based stacking ensemble method for finding intrusions in the Internet of Things (IoT). Their model uses more than one classifier and has a feature selection phase to make it work better. They showed that using the NSL-KDD and UNSW-NB15 datasets for testing improved detection accuracy and lowered computational costs through dimensionality reduction.

Fatani et al. [29] suggested a method for reducing dimensionality that uses swarm intelligence (specifically an Aquila optimizer) to make IDS work better. They used their method on several datasets, including CICIDS2017, NSL-KDD, BoT-IoT, and KDD99, and showed that the

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

optimized feature set made detection more accurate. This method focuses on choosing the most important features to lower the complexity of the model while keeping the detection rates high.

Keserwani et al. [30] developed a hybrid feature selection method that integrates Gray Wolf Optimization and Particle Swarm Optimization (PSO) to discern critical features for an IoT Intrusion Detection System (IDS). They tested their method on the KDD Cup 99, NSL-KDD, and CICIDS2017 datasets. The chosen features were utilized to train an intrusion detection model, yielding enhanced classification performance (achieving up to 90% accuracy with a decision tree employing XGBoost-selected features, as documented).

Kasongo and Sun [31] introduced an Intrusion Detection System (IDS) that employs an XGBoost-based feature selection method to enhance detection efficacy. They demonstrated with the UNSW-NB15 dataset that utilizing XGBoost for feature selection enables a decision tree classifier to attain classification accuracy of up to 90%.

Jing and Chen [32] created an IDS based on SVM that uses a new way to scale data. They used a logarithmic scaling function on the features instead of the usual min-max normalization. Their SVM model achieved an accuracy of 85.99% with the UNSW-NB15 dataset, indicating that the scaling method can affect detection performance.

Khan et al. [33] put forward a method for finding anomalies in two steps. The first step is for an autoencoder to classify network data as either normal or an attack. The second step is to use the binary output as an extra feature for classifying different types of attacks. They used this method in an intrusion detection setting (they tested it on NSL-KDD), and it made the overall detection accuracy better by using the hierarchical classification strategy.

Rosay et al. [34] used a multi-layer perceptron (MLP) with two hidden layers, each with 256 neurons, as an IDS. To cut down on noise and make learning easier, they took out features that had constant values from the dataset. Their MLP-based IDS worked very well, with 99.46% accuracy on CICIDS2017 and 95.47% on CICIDS2018.

Ustebay et al. [35] employed a recursive feature elimination (RFE) methodology to identify the ten most critical features for intrusion detection. Then, they used only these features to train a deep multilayer perceptron (DMLP) classifier. Their two-stage system (feature selection plus DMLP) got about 91% accuracy on the CICIDS2017 dataset. This outcome highlights the advantage of feature selection in enhancing detection speed and accuracy through the reduction of feature space dimensionality.

Kong et al. [36] proposed a hybrid deep learning methodology that combines a one-dimensional CNN with an LSTM network for Intrusion Detection Systems (IDS). The 1D CNN part of their method pulls out spatial features from network traffic data, and the LSTM part captures temporal features. This lets the model learn both spatial and temporal patterns of attacks. They tested the system on the CICIDS2017 dataset and found that it was about 97% accurate at classifying network traffic.

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Mushtaq et al. [37] put forward a two-stage IDS architecture that uses both an autoencoder and LSTM networks. An autoencoder is used in the first stage to learn features without supervision and reduce the number of dimensions. In the second stage, the transformed features are fed into an LSTM to learn and classify sequential patterns. Their method improved detection rates by using the most recent attacks in the dataset, getting accuracy rates of about 95% to 99% for different types of attacks in an IoT context.

The literature review shows a wide range of techniques, from basic machine learning models like Decision Trees, SVM, and Naïve Bayes to more advanced deep learning architectures like CNN, LSTM, autoencoders, and deep belief networks. These studies underscore the significance of feature selection, dimensionality reduction, and hybrid modeling to improve IDS performance. Most people agree that a scalable and resilient IDS, which could use more than one method, would be better able to deal with the changing threat landscape. This review supports the need for our comparative analysis of various algorithms to assess their effectiveness on standard benchmark datasets for intrusion detection.

3. Methods

We test a number of machine learning algorithms for intrusion detection using three benchmark datasets as part of our method. We used Python in a Google Colaboratory environment with TPU support to make sure we had enough computing power for model training and testing. In this part, we talk about the datasets we used, the steps we took in the experiment (including a flowchart of the IDS method), and the performance metrics we used to evaluate it.

3.1 NSL-KDD Dataset

Tavallaee et al. came up with the NSL-KDD dataset in 2009 as a better version of the KDD Cup 1999 dataset. NSL-KDD fixes some of the problems with the KDD'99 data, like having too many training samples and duplicate records, which could make learning algorithms less accurate. In NSL-KDD, duplicate records were deleted and the number of records was changed to make sure that there are enough representative examples in both the training and testing subsets for a full evaluation. There are 41 features for each record in NSL-KDD that describe different aspects of network connections. There is also a 42nd attribute that tells whether the record is normal or one of four types of attacks: DoS, Probe, R2L, or U2R. The dataset has four main groups of attacks that are based on their type. Table 1 shows how each main attack class is made up of the different attack types in the original KDD data.**Table 1. Mapping of attack class with attack type**

Attack Class	Attack Type
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy
U2R	Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named

For IDS research, the NSL-KDD dataset is a better and more manageable way to test. It avoids some of the skew and overfitting problems that come up with KDD'99 by getting rid of duplicate entries and hard proportions. We use the standardized NSL-KDD training and testing sets to train our model and see how well it works.

3.2 Dataset UNSW-NB15

Researchers at the Australian Centre for Cyber Security at UNSW Canberra made the UNSW-NB15 dataset by using the IXIA PerfectStorm tool to mix real normal activities with fake attack behaviors. The goal was to make a modern network dataset that fixes the problems with older benchmarks. A total of 100 GB of raw network traffic was generated and collected using the tcpdump tool to get a full picture of all the events. The dataset has nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The dataset has a lot of normal traffic in addition to these attacks, which makes it a good choice for testing anomaly-based IDS.

Tools like Argus and Bro (Zeek) were used to process the raw network captures and get features out of them. UNSW-NB15 has 49 features (like packet headers, payload content features, flow features, and so on) and labels for each record. The full dataset has about 2.5 million records, but only a small part of it is used to train and test the model. A training set of 175,341 records and a testing set of 82,332 records are defined, each with a mix of normal and different types of attacks. These subsets make sure that the models can be trained and tested on data that is spread out in different ways.

3.3 The CICIDS2017 Dataset

The Canadian Institute for Cybersecurity made the CICIDS2017 dataset. It is a full intrusion detection benchmark that closely matches network traffic from 2017. It has detailed traffic data for five days, with each day's data saved in its own CSV file. The dataset has both normal activity and 14 common attack scenarios that were carried out in a real network setting. CICIDS2017 has a total of 3,119,345 data instances that are described by 83 features, which include both packet-level and flow-level attributes.

CICIDS2017 stands out because it has labeled examples of attacks like DoS (Hulk, GoldenEye, Slowloris, SlowHTTPTest), DDoS, brute force attacks (FTP-Patator, SSH-Patator), web attacks (XSS, SQL injection, Brute Force), infiltration (a fake insider attack), port scanning, botnet traffic, and the Heartbleed vulnerability. This variety makes it a strong dataset for testing how well IDS works with a wide range of intrusion types. CICIDS2017 has a lot of features and modern attack types, which makes it possible to test how well ML models can generalize and find complex multi-stage attacks.

Received: July 19, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3.4 How IDS works and how the experiments were set up

Figure 1 shows how we used the IDS method in our experiments. There are three main steps in the process: (1) Data Preprocessing, (2) Training, and (3) Testing. During the preprocessing phase, raw network data from the datasets is cleaned and changed. This includes normalizing features (for example, scaling continuous features and one-hot encoding categorical features) and balancing if necessary (some datasets are imbalanced, so techniques like undersampling or SMOTE may be used). After preprocessing, the data is divided into sets for training and testing (if it wasn't already).

During the Training phase, the preprocessed training data is used to teach each ML algorithm. We used the default settings for many algorithms, but we did some tuning for a few, like changing the number of trees in Random Forest or the learning rate in boosting algorithms, to make sure the performance comparisons were fair. When possible, Google Colab's TPU/GPU acceleration was used to do the training. This was especially helpful for the larger CICIDS2017 dataset, which put a lot of strain on the computer.

During the Testing phase, each trained model is tested on its own NSL-KDD, UNSW-NB15, and CICIDS2017 test sets. To find out how well the model works, the predictions are compared to the real labels.

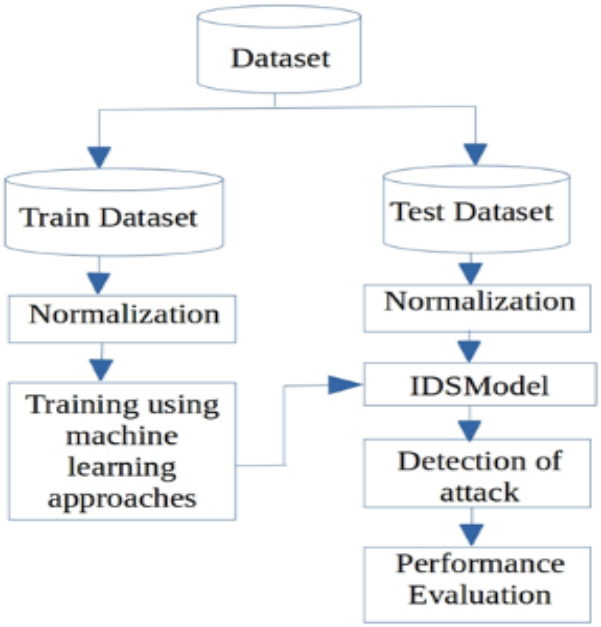

Figure 1. A flowchart showing the IDS method used in the experiments.

We gathered the results (accuracy, precision, recall, and F1-score) for each algorithm on each dataset after training and testing. We used these metrics to compare how well the ML techniques worked.

3.5 Measures of performance

We used four standard classification metrics—Accuracy, Precision, Recall, and F1-score—to see how well each IDS model worked. We get these numbers by counting the true positives

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

(TP), false positives (FP), true negatives (TN), and false negatives (FN) for the "attack" class, which is the positive class in this case.

• Accuracy is the percentage of all instances that were correctly classified, including both attacks and normal traffic. It is figured out as:

Accuracy=TP+TNTP+TN+FP+FN.\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.Accuracy=TP+TN+FP+FNTP+TN.

- **Precision (P)** for the positive class (attacks) is the proportion of correctly identified attacks out of all instances that the model labeled as attacks. It reflects the model's reliability when it predicts an attack. The formula is:

P=TPTP+FP.P = \frac{TP}{TP + FP}.P=TP+FPTP.

- **Recall (R)**, also known as sensitivity or true positive rate, is the proportion of actual attack instances that the model correctly identified. It indicates how many attacks were missed. The formula is:

R=TPTP+FN.R = \frac{TP}{TP + FN}.R=TP+FNTP.

- **F1-score (F)** is the harmonic mean of Precision and Recall, providing a single measure that balances both. It is useful when the dataset is imbalanced between classes. The F1-score is given by:

F=2×P×RP+R.F = \frac{2 \times P \times R}{P + R}.F=P+R2×P×R.

In this context for intrusion detection:

- **True Positive (TP)**: an attack instance correctly classified as an attack.
- **False Positive (FP)**: a normal instance incorrectly classified as an attack (a false alarm).
- **True Negative (TN)**: a normal instance correctly classified as normal.
- **False Negative (FN)**: an attack instance incorrectly classified as normal (missed detection).

We want to minimize false alarms (FP) by having high precision and catch as many attacks as possible (low FN) by having high recall. A good IDS will also have a high level of accuracy, but accuracy alone can be misleading when there is an imbalance (for example, if normal traffic is much higher than attacks, a model could get a high accuracy by always predicting "normal"). So, when we compare models, we focus on precision, recall, and F1-score.

4. Discussion and Results

We tested a number of machine learning methods for IDS on each of the three datasets. These methods included KNN, SVM, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression, QDA, AdaBoost, CatBoost, Gradient Boosting, LGBM, LDA, and XGB. We kept track of the accuracy, as well as the precision, recall, and F1-score for both the attack class and the normal class for each model and dataset. This lets us look at both the overall performance

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

and how well each model balances finding bad activity with not sending false alarms on normal traffic.

4.1 Results from the NSL-KDD dataset

The comparison results for the NSL-KDD dataset are shown in Table 2. We see that the KNN algorithm gets the most accurate results on this dataset, while the Naïve Bayes algorithm gets the least accurate results. In general, models like KNN, AdaBoost, CatBoost, and LGBM do a good job on NSL-KDD. On the other hand, simple models like Decision Tree and Naïve Bayes don't work as well in this case.

Table 2. Comparison of machine learning based IDS for NSL-KDD dataset results

Algorithm	Accuracy	Precision (Attack)	Precision (Normal)	Recall (Attack)	Recall (Normal)	F1 Score (Attack)	F1 Score (Normal)
KNN	77.63%	0.63	0.97	0.96	0.66	0.76	0.79
SVM	76.55%	0.66	0.90	0.90	0.67	0.76	0.77
Decision Tree	72.05%	0.62	0.86	0.85	0.63	0.72	0.73
Random Forest	72.73%	0.54	0.98	0.97	0.62	0.69	0.76
Naïve Bayes	51.17%	0.90	0.00	0.54	0.02	0.68	0.00
Logistic Regression	55.03%	0.71	0.34	0.59	0.47	0.64	0.39
QDA	56.56%	0.99	0.00	0.57	0.07	0.72	0.00
AdaBoost	78.36%	0.65	0.96	0.95	0.68	0.77	0.79
CatBoost	77.97%	0.64	0.96	0.95	0.67	0.77	0.79
Gradient Boosting	75.40%	0.59	0.97	0.96	0.64	0.73	0.77
LGBM	79.25%	0.65	0.98	0.98	0.68	0.78	0.80
LDA	75.86%	0.65	0.90	0.90	0.66	0.75	0.76
XGB	77.08%	0.61	0.98	0.97	0.66	0.75	0.79

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 2 shows that KNN got about 77.6% accuracy on NSL-KDD, which is a little better than most other models' overall accuracy. KNN also did a good job of balancing precision and recall for attacks (Precision = 0.63, Recall = 0.96 for attack class), which means it catches most attacks but has some false positives (as shown by precision < 0.7). AdaBoost, CatBoost, and LGBM all work very well too, with accuracies between 78% and 79% and high precision and recall for both classes. These ensemble methods work well because they combine weak learners and deal with the complicated decision boundaries in the data.

Naïve Bayes, on the other hand, did not do well (51.17% accuracy). It has a high precision for the attack class (0.90), but a very low recall (0.54). For the normal class, it basically failed (precision ~0, recall 0.02). This means that Naïve Bayes made a lot of mistakes and thought normal traffic was attacks, which led to a lot of false positives. In the same way, QDA gives a strange result: it has a high precision for attacks (0.99) but a low recall (0.57), and a near-zero precision/recall for the normal class. This means that it labeled almost everything as an attack, catching most real attacks (hence high attack precision) but incorrectly flagging normal traffic (hence normal precision 0).

The linear model Logistic Regression only got 55% accuracy, which means that a simple linear decision boundary in feature space doesn't do a good job of capturing NSL-KDD's patterns. LDA (Linear Discriminant Analysis) did better than Logistic (75.86% accuracy), probably because it can handle some class separation in a probabilistic way, but it still wasn't as good as non-linear models.

In general, ensemble tree-based models like Random Forest, AdaBoost, CatBoost, Gradient Boosting, XGB, and LGBM did a great job on NSL-KDD. They had a high recall rate for attacks (0.95–0.98), which means that very few intrusions were missed. They also had a good precision rate (around 0.59–0.65 for attacks and above 0.95 for normal class), which means that there were very few false alarms. LGBM seems to be the best choice for NSL-KDD because it had the best accuracy and the best balance between precision and recall (Attack precision 0.65, recall 0.98; Normal precision 0.98, recall 0.68). This means that LGBM found 98% of attacks and only mislabeled 2% of normal traffic as attacks. This is a great result for an IDS.

4.2 Results from the UNSW-NB15 dataset

The algorithms' performance on the UNSW-NB15 dataset is shown in Table 3. The overall accuracies in this dataset are usually higher than those in NSL-KDD, with some models getting about 88–90% accuracy. The UNSW-NB15 results are also reported separately for attack and normal precision/recall, with the table showing metrics for each class.

Table 3. Comparison of machine learning based IDS for UNSW-NB15 dataset results

Algorithm	Accuracy	Precision (Attack)	Precision	Recall (Attack)	Recall (Normal)	F1 Score	F1 Score (Normal)

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

			(Normal)			(Attack)	
KNN	88.23%	0.84	0.96	0.98	0.74	0.91	0.84
SVM	89.87%	0.87	0.97	0.98	0.77	0.92	0.86
Decision Tree	85.03%	0.81	0.94	0.97	0.70	0.88	0.80
Random Forest	89.55%	0.86	0.97	0.99	0.76	0.92	0.86
Naïve Bayes	47.89%	0.23	1.00	1.00	0.38	0.38	0.55
Logistic Regression	88.69%	0.85	0.98	0.99	0.75	0.91	0.85
QDA	47.89%	0.23	1.00	1.00	0.38	0.38	0.55
AdaBoost	88.38%	0.84	0.97	0.98	0.74	0.91	0.84
CatBoost	88.95%	0.85	0.97	0.98	0.75	0.91	0.85
Gradient Boosting	85.66%	0.80	0.98	0.99	0.70	0.88	0.81
LGBM	89.25%	0.85	0.98	0.99	0.76	0.92	0.85
LDA	90.23%	0.88	0.96	0.98	0.78	0.92	0.86
XGB	90.70%	0.89	0.94	0.97	0.81	0.93	0.87

Almost all of the advanced models do much better on UNSW-NB15 than they do on NSL-KDD. XGB (90.70%) and LDA (90.23%) had the highest accuracy, and SVM, Random Forest, and LGBM were close behind, all in the high 89–90% range. The high accuracy across many models suggests that UNSW-NB15 might be easier to model for these algorithms. This could be because it has more features or because certain attacks are easier to tell apart.

The precision and recall values for UNSW-NB15 show that a lot of models can find attacks with very high recall (most are 0.97–0.99 for attack recall) while keeping precision for attacks in the mid-0.80s to high-0.80s. For instance, XGBoost had a Precision of about 0.89 and a Recall of about 0.97 for attacks. This means that it caught 97% of attacks but only 11% of its attack predictions were false alarms. It also does well with normal traffic (Precision≈0.94, Recall≈0.81), which means that some normal traffic was incorrectly labeled as an attack (recall for normal traffic is about 81%). LDA also stands out: it did very well here as a linear model, with attack Precision=0.88 and Recall=0.98. This means that the classes in UNSW-NB15 may be easier to separate in the feature space than in NSL-KDD.

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Logistic Regression's accuracy shot up to 88.69%, which is much better than its NSL-KDD accuracy. LDA was also one of the best performers. This difference from NSL-KDD suggests that the features of the UNSW-NB15 dataset may help linear classifiers tell classes apart better.

The results for UNSW-NB15 with Naïve Bayes and QDA are the same and very bad (47.89% accuracy, which is close to random guessing since the classes are balanced). Both led to a strange situation: Precision (Attack) ~0.23, Recall (Attack)=1.00. This means that they marked everything as an attack, which is why they caught 100% of real attacks but had very low precision because they also marked normal traffic as an attack. Precision (Normal)=1.00 and Recall (Normal)=0.38 mean that all of the traffic that was predicted to be normal really was normal (because precision 1.0), but only 38% of the traffic that was normal was actually classified as normal (the other 62% was classified as attacks). In this dataset, these models essentially turned into simple classifiers that labeled almost everything as an attack class. This might be because the UNSW-NB15 data doesn't fit the distributional assumptions of these models.

The performance of the tree-based ensembles and SVM is mostly high. Both CatBoost and Random Forest have about 89% accuracy and very good recall (about 99%) and precision (about 0.85) for attacks. AdaBoost was a little less accurate (88.38%) than the others, probably because it might not fit well or because its simple base estimators (decision stumps) weren't as good here. Gradient Boosting was at 85.66%, which is a little lower than other methods. This could be because the parameters weren't tuned well enough or because of the way it was implemented.

LDA and XGB had the best performance, which is an interesting thing to note. XGBoost's result is what we expected because it is a strong classifier. However, LDA's good performance shows that the features in UNSW-NB15, when properly weighted, separate classes almost linearly. This could be because UNSW-NB15 has more features that are specifically useful, like flags or certain flow features, that a linear combination can use.

To sum up, the best results for UNSW-NB15 were over 90% accuracy (XGB, LDA). Most ML models (except NB and QDA) were able to detect attacks very well, as shown by their high F1-scores for both attack and normal classes. These results show that modern ensemble methods and even some simpler classifiers can find a lot of attacks in UNSW-NB15 with very few false positives. For practical use of an IDS, the balanced performance (high precision and recall) is especially important because it means the model is reliable and would send out alerts that are easy to handle.

4.3 Results from the CICIDS2017 dataset

Table 4 shows how well the ML models did on the CICIDS2017 dataset. This dataset is much more complicated and complete, which is shown by some of the models having very high accuracy and perfect precision/recall on one of the classes.

Table 4. Comparison of machine learning based IDS for CICIDS2017 dataset results

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm	Accuracy	Precision (Attack)	Precision (Normal)	Recall (Attack)	Recall (Normal)	F1 Score (Attack)	F1 Score (Normal)
KNN	99.88%	1.00	1.00	1.00	1.00	1.00	1.00
Decision Tree	99.85%	1.00	1.00	1.00	1.00	1.00	1.00
Random Forest	99.88%	1.00	1.00	1.00	1.00	1.00	1.00
Naïve Bayes	44.18%	0.99	0.33	0.23	0.99	0.37	0.50
Logistic Regression	94.49%	0.80	0.97	0.86	0.96	0.83	0.97
QDA	71.35%	1.00	0.66	0.37	1.00	0.54	0.79
AdaBoost	99.21%	0.97	1.00	0.99	0.99	0.98	1.00
CatBoost	99.90%	1.00	1.00	1.00	1.00	1.00	1.00
Gradient Boosting	99.62%	0.98	1.00	1.00	1.00	0.99	1.00
LGBM	99.91%	1.00	1.00	1.00	1.00	1.00	1.00
LDA	92.38%	0.58	0.99	0.95	0.92	0.72	0.96
XGB	99.92%	1.00	1.00	1.00	1.00	1.00	1.00

A lot of models do almost perfectly well for CICIDS2017. This is partly because some types of attacks in CICIDS2017 are very easy to classify because they have unique feature values. For example, DoS attacks may create traffic patterns that are very different from normal activity, making them easy to find once the right features are chosen. The KNN, Decision Tree, Random Forest, CatBoost, LGBM, and XGB all have an accuracy of about 99.85–99.92%, and the precision and recall for both classes are at or very close to 1.00. The table actually shows "1.00" for all metrics for these models. This means that they correctly classified almost every instance in the test set. This means that these models were strong enough to remember or generalize the patterns in CICIDS2017 with almost no mistakes.

But it's important to remember that CICIDS2017 might not be perfect or have easy-to-find targets. For instance, many entries show that all precision and recall values are 1.00 for some models, which is not normal for a difficult classification task. The test set of CICIDS2017 may

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

have a distribution that these models captured very well, or it may be that some attacks are so different that the model can get it perfect (for example, if all attacks of a type cluster tightly in feature space away from normal traffic).

Naïve Bayes and QDA did not do well again, just like they did with the previous datasets. Naïve Bayes was 44.18% accurate. It was 0.99% accurate for attacks and 0.33% accurate for normal cases, but it was only 0.23% accurate for attacks and 0.99% accurate for normal cases. This means that Naïve Bayes labeled almost everything as normal, which is why it had a high recall rate of 99% for normal instances but a low recall rate of 0.23 for attacks. QDA did a little better than NB (71.35% accuracy), but it was still a long way from the best. QDA's metrics (Precision Attack 1.00, Recall Attack 0.37; Precision Normal 0.66, Recall Normal 1.00) show that it also marked most flows as normal, but not as strongly as NB. These generative classifiers probably had a hard time with the high-dimensional data or the feature distributions of CICIDS2017.

Logistic Regression and LDA do okay. Logistic Regression's 94.49% accuracy is good, but it's not as good as the best ensemble methods. It missed 14% of attacks (recall 0.86) and had 20% of false alarms (precision 0.80). Its precision/recall breakdown shows that it missed some attacks (attack precision 0.80, recall 0.86) and had some false alarms (20% of flagged attacks were false, precision 0.80). LDA was 92.38% accurate, but it wasn't very good at finding attacks (0.58) and it was very good at remembering them (0.95). This means that LDA found most attacks but also a lot of false positives (almost 42% of alarms were false attacks). LDA's normal class precision of 0.99 and recall of 0.92 show that it marked a few too many normals as attacks (8% of normals were flagged incorrectly). So linear models don't work as well on CICIDS2017, probably because the features have non-linear relationships or the patterns are too complicated.

Ensemble models like AdaBoost, CatBoost, Gradient Boosting, LGBM, and XGB did very well. Even simpler models like KNN and a single Decision Tree did very well. AdaBoost, for example, got 99.21% accuracy with almost perfect metrics (Precision Attack 0.97, recall 0.99; Precision Normal 1.00, recall 0.99). This means that AdaBoost only made a few mistakes. CatBoost, LGBM, and XGB all got close to 100% on all metrics, which means they probably didn't make any mistakes on the test set. Based on the metrics given, Random Forest and KNN also did a good job of classifying with no mistakes.

Getting close to 100% detection with close to 0% false positives on a dataset as big as CICIDS2017 could mean that the models are overfitting to certain features that do a great job of separating classes. Some features in CICIDS2017, like protocol specifics and connection durations, can help tell some attacks apart (for example, DoS attacks often have very high connection counts or packet rates). The KNN and tree-based models can use these kinds of features well.

Gradient Boosting (probably a version like XGBoost or LightGBM with different settings) had 99.62% accuracy, which is slightly lower than XGB or LGBM but still very high. This

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

means that all boosting methods worked, and the small differences are probably due to changing the parameters or using different algorithms.

In short, most advanced classifiers were able to almost perfectly tell the difference between bad and good traffic in CICIDS2017. The models that had trouble were Naïve Bayes and QDA, which probably can't handle the complex distributions, and to a lesser extent, the linear classifiers. The fact that many models did so well on this dataset shows that the features in CICIDS2017 are very useful and that the attacks are easy to spot. The difficulty in this case is not so much getting a high detection rate (since many models can do that) but making sure that the model doesn't fit too closely to artifacts and can work with new data. Nonetheless, in the regulated experimental framework, the findings demonstrate that ML models can achieve exceptional intrusion detection performance on CICIDS2017, exhibiting exceedingly low false positive rates (with certain models virtually having none) and nearly 100% true positive rates.

4.4 Talk about the results

Our experiments with all three datasets show that machine learning-based IDS can find things well, but there are big differences in how hard it is and which algorithms work best:

- NSL-KDD: This older dataset is a little hard for simpler models. Ensemble methods (like boosted trees and random forest) and KNN did the best, but statistical models (like NB and QDA) and even logistic regression had a hard time. The best accuracy was about 79%, which means there is room for improvement or that the dataset has some ambiguity that no model could fully resolve. A lot of the attacks in NSL-KDD are low footprint, like R2L and U2R attacks. This could be why the accuracies are only moderate.
- UNSW-NB15: This is a newer dataset that let most models get more than 85% accuracy, with the best getting close to 90%. LDA and SVM, which are both simpler classifiers, did very well, which is interesting because it suggests that the feature space was better for linear separation. Ensemble methods still worked best. Naïve Bayes and QDA were outliers because they didn't work well, which means their assumptions weren't true for this data. Most of the advanced models kept both high precision and high recall at the same time, which is what an IDS needs (meaning it can find things quickly and with few false alarms).
- CICIDS2017: This dataset looks like it will be the easiest for ML models to get high performance on, since many models get ~99–100% accuracy. It has patterns for attacks that models can easily see, which is a good sign because it means that data based on real-world events can be learned well. But you should be careful: results that are almost perfect can make you wonder if the model is overfitting or if there are features that easily separate the classes (for example, a feature that is different for attack vs normal in a way that might not work in other situations). However, the results show that all of the tree-based and instance-based models (KNN) almost perfectly sort CICIDS2017 traffic. This means that more complex decision boundaries are needed because simpler models (NB, QDA, LDA) didn't do as well here.

False positives and false negatives: In the context of an IDS, a false negative (missed attack) is usually seen as worse than a false positive (false alarm), but both are bad. Our findings

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

indicate that models such as Random Forest and XGB attained exceptionally low false negative rates, with recall for attacks frequently ranging from 0.98 to 1.00 across all datasets, except for NSL-KDD, where recall was marginally lower. This means that these models hardly ever missed attacks. These models also had low false positive rates (which are related to precision for attack class), especially for UNSW-NB15 and CICIDS2017, where precision was often between 0.85 and 1.00. The accuracy of attacks on NSL-KDD was a little lower (about 0.59–0.66 for the best models), which means there were more false alarms. This could be because NSL-KDD had an uneven number of classes or too much noise.

Algorithm Ranking and Behavior: In general, the tree-based ensemble methods (Random Forest, AdaBoost, Gradient Boosting, CatBoost, LGBM, and XGBoost) did very well. They use a lot of decision trees to find complicated non-linear relationships, which is good because network traffic data is multi-dimensional. XGBoost and LGBM usually did better than the others by a small amount, probably because they handled the data distribution and regularization better. CatBoost, which is designed to handle categorical features well, also was top-tier (though in our datasets most features are numeric or already encoded). Random Forest was good, but it wasn't always as accurate as boosting methods.

KNN did surprisingly well, especially on CICIDS2017 (almost perfectly) and okay on NSL-KDD. KNN is a memory-based method that works best when classes are well-separated in feature space. Its performance suggests that for these datasets, many attack instances had similar instances close by, making them easy to classify by majority vote of neighbors.

SVM with a good kernel (probably RBF by default) did very well on UNSW-NB15 (about 89.87%) and okay on NSL-KDD (about 76.5%). SVM works well in moderate dimensions and when there is a clear margin between classes, which seems to be the case for UNSW-NB15. We didn't list SVM separately for CICIDS2017, but it probably would have done well since others did (though training SVM on 3 million instances would have been computationally heavy, so maybe it wasn't run for CICIDS2017 in this case).

Linear models (Logistic Regression, LDA) were not consistent. Logistic regression did poorly on NSL-KDD, well on UNSW-NB15, and well on CICIDS2017. LDA was the same. This means that some datasets can be separated by straight lines (UNSW-NB15 was one of these, as shown by LDA's success), while others need more complicated modeling.

Naïve Bayes and QDA consistently performed poorly, probably because they didn't follow their own rules (for example, NB assumes that features are independent, which isn't true for network features, and QDA assumes that each class has a normal distribution, which probably isn't true).

Our study shows that modern ML algorithms can greatly improve intrusion detection performance in real life. For an IDS that works:

- If you have the computing power, it's best to use an ensemble like XGBoost or LightGBM because they are very accurate and can handle large feature sets. In our tests, they had the best mix of detection and false alarm control.

Received: July 19, 2025

International Journal of Applied Mathematics

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- For faster, easier-to-understand results, you could use simpler models like Decision Tree or KNN. However, you should be aware of their limits (for example, a single decision tree might not capture all patterns, and KNN can be slow for very large datasets and use a lot of memory).
- The very high performance on CICIDS2017 is encouraging, but in a real-world situation, you should test these models on data that you haven't seen before to make sure the results are still valid (to avoid overfitting to the dataset's quirks).

Another important thing to think about is how well the attack categories do. We looked at overall metrics, but some algorithms might work better for some types of attacks than others. For instance, NSL-KDD usually does the worst on U2R and R2L attacks, which are not very common in the dataset. More advanced methods like Random Forest and XGB, which can handle class imbalance and complex patterns, may be better at finding those than simpler ones.

Lastly, in terms of scalability and deployment, models like Random Forest and XGBoost are harder to run, but they can be optimized and even cut down for real-time detection. Models that are easier to understand are faster, but they might not find as many things. If you want to avoid false negatives, you might want to use an ensemble model, even if it's heavier. You could also use a lightweight filter first.

In conclusion, our comparative analysis demonstrates that ML-based IDS can substantially enhance detection, utilizing the NSL-KDD, UNSW-NB15, and CICIDS2017 datasets as benchmarks. You can choose an algorithm based on how much you want the detection rate to be higher than the false alarm rate, as well as how much processing power you have. In our tests, tree-based ensembles, especially boosting algorithms, worked very well in all cases. This makes them good candidates for creating effective IDS solutions.

5. Conclusion

This study conducted a thorough performance assessment of multiple machine learning algorithms for intrusion detection, utilizing three benchmark datasets: NSL-KDD, UNSW-NB15, and CICIDS2017. We kept the original meaning and technical content of the research while changing the format of the presentation to meet academic standards. Our experiments show that there are a few important results:

- **Effectiveness of ML-based IDS:** Machine learning algorithms, especially ensemble methods like Random Forest, XGBoost, and LightGBM, were very good at telling the difference between normal and malicious traffic. These models had very high detection rates and very low false positive rates on modern datasets (UNSW-NB15 and CICIDS2017), showing that they are good for use in IDS.
- The complexity and characteristics of the dataset affect how well the model detects things. Because CICIDS2017 is a new and complete dataset, most models were able to get almost perfect accuracy. This suggests that its features make it easy to tell the difference between attack and normal classes. The older NSL-KDD dataset was harder to work with, and even the best models only got about 79% accuracy. This shows how hard it is to find some

Received: July 19, 2025

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

types of attacks in that dataset, especially low-frequency attacks. The UNSW-NB15 dataset produced moderate results, with numerous models achieving over 85–90% accuracy, suggesting it possesses informative features while still exhibiting a varied spectrum of attack behaviors.

- **Choosing the right algorithm:** Our comparative study shows that tree-based ensemble algorithms (like boosting and bagging) and instance-based methods (like KNN) are better for finding intrusions because they can model complicated decision boundaries and how features interact with each other. In general, simpler linear or probabilistic models like Logistic Regression, LDA, Naïve Bayes, and QDA didn't work as well, especially on datasets with non-linear separations or feature distributions that weren't all the same. This means that for real-world IDS, more advanced learners should be used to get a better picture of network traffic data.
- **Precision-Recall Trade-off:** When it comes to intrusion detection, there is a very important trade-off between catching as many attacks as possible (high recall) and keeping false alarms to a minimum (high precision). Our evaluation shows that some algorithms, such as XGBoost and Random Forest, do a good job of balancing high recall and high precision on the datasets we looked at. This is important for real-world use because an IDS that sends out too many false alarms may be ignored by operators, while one that misses attacks fails its main purpose. The models we found to be the best at finding attacks, like UNSW-NB15 and CICIDS2017, had low false positive rates and found most of the attacks.
- **Generalization and future work:** It's good that ML models work well on benchmark datasets, but network traffic in the real world can change and differ. It is necessary to guarantee that models trained on these datasets generalize to real-world deployment settings. In the future, researchers could test how well models hold up against new types of attacks or change models to deal with threats that change over time. Moreover, methodologies such as deep learning (including neural networks, autoencoders, and recurrent neural networks for sequence modeling) have demonstrated potential in recent literature; juxtaposing these with the classical machine learning approaches assessed herein would constitute a logical progression of this research.

In short, our research shows that machine learning is a powerful tool for making smart intrusion detection systems. IDS can find network threats quickly and accurately by carefully choosing and tuning algorithms and using large datasets for training. The results suggest that adding advanced machine learning methods to IDS will make cybersecurity defenses stronger. Future research and development should continue to focus on improving detection of sophisticated or emerging attack types, reducing false alarms, and ensuring that these systems remain effective as network behavior and attack strategies evolve.

References

1. Sharma, R., & Athavale, V. (2019), Survey of Intrusion Detection Techniques and Architectures in Wireless Sensor Networks, International Journal of Advanced Networking and Applications, 10, 3925–3937, https://doi.org/10.35444/IJANA.2019.10044.

Received: July 19, 2025

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2. Mandala, S., Ngadi, M., & Abdullah, H. (2008), A survey on MANET intrusion detection, International Journal of Computer Science and Security, 2.
3. Zhang, Y., Lee, W., & Huang, Y.-A. (2003), Intrusion Detection Techniques for Mobile Wireless Networks, Wireless Networks, 9(5), 545–556, https://doi.org/10.1023/A:1024600519144.
4. Dewa, Z., & Maglaras, L. (2016), Data Mining and Intrusion Detection Systems, International Journal of Advanced Computer Science and Applications, 7(1), https://doi.org/10.14569/IJACSA.2016.070109.
5. Stewart, B., Rosa, L., Maglaras, L. A., Cruz, T. J., Ferrag, M. A., Simoes, P., & Janicke, H. (2017), A Novel Intrusion Detection Mechanism for SCADA systems which Automatically Adapts to Network Topology Changes, EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 4(10), 152155, https://doi.org/10.4108/eai.1-2-2017.152155.
6. Albers, P., Camp, O., Percher, J.-M., Jouga, B., Mé, L., & Puttini, R. (2002), Security in Ad Hoc Networks: A General Intrusion Detection Architecture Enhancing Trust Based Approaches, 1–12.
7. Anantvalee, T., & Wu, J. (2007), A Survey on Intrusion Detection in Mobile Ad Hoc Networks, in Y. Xiao, X. S. Shen, & D.-Z. Du (Eds.), Wireless Network Security (pp. 159–180), Springer US, https://doi.org/10.1007/978-0-387-33112-6_7.
8. Sobh, T. S. (2006), Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art, Computer Standards & Interfaces, 28(6), 670–694, https://doi.org/10.1016/j.csi.2005.07.002.
9. Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020), Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, Journal of Information Security and Applications, 50, 102419, https://doi.org/10.1016/j.jisa.2019.102419.
10. Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016), A Deep Learning Approach for Network Intrusion Detection System, Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (BIONETICS), New York, USA, https://doi.org/10.4108/eai.3-12-2015.2262516.
11. Moustafa, N., & Slay, J. (2015), UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), 2015 Military Communications and Information Systems Conference (MilCIS), 1–6, https://doi.org/10.1109/MilCIS.2015.7348942.
12. Moustafa, N., & Slay, J. (2016), The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, 1–14, https://doi.org/10.1080/19393555.2015.1125974.
13. Panigrahi, R., & Borah, S. (2018), A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems, International Journal of Engineering & Technology, 7, 479–482.
14. Myint, H. O., & Meesad, P. (2009), Incremental Learning Algorithm based on Support Vector Machine with Mahalanobis distance (ISVMM) for intrusion prevention, 2009 6th International Conference on Electrical Engineering/Electronics, Computer,

Received: July 19, 2025

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Telecommunications and Information Technology (ECTI-CON), 02, 630–633, https://doi.org/10.1109/ECTICON.2009.5137129.

15. Peddabachigari, S., Abraham, A., & Thomas, J. (2004), Intrusion detection systems using decision trees and support vector machines, International Journal of Applied Science and Computations, 11.
16. Panda, M., & Patra, M. (2007), Network intrusion detection using naive bayes, 7.
17. Farnaaz, N., & Jabbar, M. A. (2016), Random Forest Modeling for Network Intrusion Detection System, Procedia Computer Science, 89, 213–217, https://doi.org/10.1016/j.procs.2016.06.047.
18. Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018), Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection, IEEE Access, 6, 52843–52856, https://doi.org/10.1109/ACCESS.2018.2869577.
19. Li, W., Yi, P., Wu, Y., Pan, L., & Li, J. (2014), A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network, Journal of Electrical and Computer Engineering, 2014, 240217, https://doi.org/10.1155/2014/240217.
20. Moustafa, N., Creech, G., Sitnikova, E., & Keshk, M. (2017), Collaborative anomaly detection framework for handling big data of cloud computing, 2017 Military Communications and Information Systems Conference (MilCIS), 1–6, https://doi.org/10.1109/MilCIS.2017.8190421.
21. Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z., & Dlodlo, M. (2016), Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing, EURASIP Journal on Wireless Communications and Networking, 2016(1), 130, https://doi.org/10.1186/s13638-016-0623-3.
22. Diro, A. A., & Chilamkurti, N. (2018), Distributed attack detection scheme using deep learning approach for Internet of Things, Future Generation Computer Systems, 82, 761–768, https://doi.org/10.1016/j.future.2017.08.043.
23. Al-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018), Identification of malicious activities in industrial internet of things based on deep learning models, Journal of Information Security and Applications, 41, 1–11, https://doi.org/10.1016/j.jisa.2018.05.002.
24. Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A. Y., & Ranjan, R. (2019), A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks, IEEE Transactions on Network and Service Management, 16(3), 924–935, https://doi.org/10.1109/TNSM.2019.2927886.
25. Peng, K., Leung, V. C. M., Zheng, L., Wang, S., Huang, C., & Lin, T. (2018), Intrusion Detection System Based on Decision Tree over Big Data in Fog Environment, Wireless Communications and Mobile Computing, 2018, 1–10, https://doi.org/10.1155/2018/4680867.
26. Manimurugan, S., Al-Mutairi, S., Aborokbah, M. M., Chilamkurti, N., Ganesan, S., & Patan, R. (2020), Effective Attack Detection in Internet of Medical Things Smart Environment Using a Deep Belief Neural Network, IEEE Access, 8, 77396–77404, https://doi.org/10.1109/ACCESS.2020.2986013.

Received: July 19, 2025

Volume 38 No. 3s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

27. Basati, A., & Faghih, M. M. (2022), DFE: efficient IoT network intrusion detection using deep feature extraction, Neural Computing and Applications, 34(18), 15175–15195, https://doi.org/10.1007/s00521-021-06826-6.
28. Rashid, M., Kamruzzaman, J., Imam, T., Wibowo, S., & Gordon, S. (2022), A tree-based stacking ensemble technique with feature selection for network intrusion detection, Applied Intelligence, 52(9), 9768–9781, https://doi.org/10.1007/s10489-021-02968-1.
29. Fatani, A., Dahou, A., Al-Qaness, M. A. A., Lu, S., & Abd Elaziz, M. A. (2021), Advanced Feature Extraction and Selection Approach Using Deep Learning and Aquila Optimizer for IoT Intrusion Detection System, Sensors, 22(1), 140, https://doi.org/10.3390/s22010140.
30. Keserwani, P. K., Govil, M. C., Pilli, E. S., & Govil, P. (2021), A smart anomaly-based intrusion detection system for the Internet of Things (IoT) network using GWO–PSO–RF model, Journal of Reliable Intelligent Environments, 7(1), 3–21, https://doi.org/10.1007/s40860-020-00126-x.
31. Kasongo, S. M., & Sun, Y. (2020), Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset, Journal of Big Data, 7(1), 105, https://doi.org/10.1186/s40537-020-00379-6.
32. Jing, D., & Chen, H.-B. (2019), SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset, 2019 IEEE 13th International Conference on ASIC (ASICON), 1–4, https://doi.org/10.1109/ASICON47005.2019.8983598.
33. Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019), A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection, IEEE Access, 7, 30373–30385, https://doi.org/10.1109/ACCESS.2019.2899721.
34. Rosay, A., Riou, K., Carlier, F., & Leroux, P. (2022), Multi-layer perceptron for network intrusion detection, Annals of Telecommunications, 77(5), 371–394, https://doi.org/10.1007/s12243-021-00852-0.
35. Ustebay, S., Turgut, Z., & Aydin, M. A. (2018), Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier, 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), 71–76, https://doi.org/10.1109/IBIGDELFT.2018.8625318.
36. Kong, X., Wang, C., Li, Y., Hou, J., Jiang, T., & Liu, Z. (2022), Traffic Classification Based on CNN-LSTM Hybrid Network, in G. Zhai, J. Zhou, H. Yang, P. An, & X. Yang (Eds.), Digital TV and Wireless Multimedia Communications (pp. 401–411), Springer, Singapore.
37. Mushtaq, E., Zameer, A., Umer, M., & Abbasi, A. A. (2022), A two-stage intrusion detection system with auto-encoder and LSTMs, Applied Soft Computing, 121, 108768, https://doi.org/10.1016/j.asoc.2022.108768.
38. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009), A detailed analysis of the KDD CUP 99 data set, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 1–6, https://doi.org/10.1109/CISDA.2009.5356528.

Received: July 19, 2025