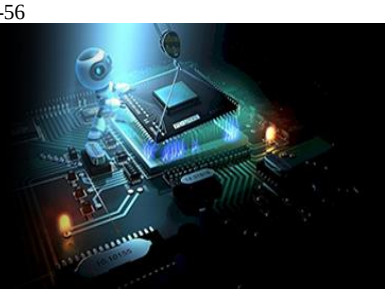
International Journal of Engineering in Computer Science



E-ISSN: 2663-3590
P-ISSN: 2663-3582
Impact Factor (RJIF): 5.52
www.computersciencejournals.com/ijecs
IJECS 2025; 7(2): 45-56
Received: 10-06-2025
Accepted: 14-07-2025

Ahmed Jaber Joudah
Department of Computer Engineering Techniques, Engineering Technical College, University of Imam Jaáfar Al-Sadiq, Thi-Qar Branch, Thi-Qar, Iraq

Huda Yass Khudhair
Department of Computer Engineering Techniques, Engineering Technical College, University of Imam Jaáfar Al-Sadiq, Thi-Qar Branch, Thi-Qar, Iraq

Corresponding Author:
Ahmed Jaber Joudah
Department of Computer Engineering Techniques, Engineering Technical College, University of Imam Jaáfar Al-Sadiq, Thi-Qar Branch, Thi-Qar, Iraq

Detecting network security attacks using ensembling techniques

Ahmed Jaber Joudah and Huda Yass Khudhair

DOI: https://www.doi.org/10.33545/26633582.2025.v7.i2a.201

Abstract
The number of devices using the Internet, as well as the tasks performed through it, increases every day. So do the attacks against the availability, integrity and confidentiality of the information they handle. Intrusion Detection Systems (IDS) are one of the most effective security mechanisms to protect network systems against computer attacks, whether or not there is prior knowledge of them. The application of Artificial Intelligence and, more specifically, of automatic learning in this type of program stands out. Most of these IDS analyse network traffic and its normal behaviour so that they can activate an alarm when part of said traffic differs from the usual. In this way, some types of attacks can be detected even if they have never been carried out before. The study investigates ensemble learning for detecting network security attacks, intending to improve machine learning algorithm performance by combining their results. This study evaluates the efficacy of different machine learning algorithms in intrusion detection utilizing the NSL KDD dataset, employing *Recursive Feature Elimination* (RFE) for feature selection. Results indicate that *Random Forest* (RF) and *Gradient Boosting Machine* (GBM) are the foremost performers, while *Decision Tree* (DT) also demonstrates commendable balanced accuracy and precision. Although *Support Vector Machine* (SVM) has demonstrated favourable outcomes in previous studies, the research indicates that multiple classifiers ought to be evaluated for forthcoming *intrusion detection system* (IDS) implementations.

Keywords: Intrusion Detection Systems, IDS, Security, Intrusion Detection Systems, Ensembling Techniques

1. Introduction
Identifying malware and attacks through network traffic analysis continues to pose a significant challenge for network security and incident management professionals. Despite the existence of advanced detection technologies capable of distinguishing between malicious and benign behaviour, effective detection continues to pose a challenge [1].
One of the best ways to protect a network, whether against known attacks or those that are not yet known, is the use of Intrusion Detection Systems (IDS). IDS can monitor information about an entire network or just about a computer. Those that monitor only the device on which they are installed are called Host-based Intrusion Detection Systems (HIDS). HIDS can monitor, in addition to the packets that arrive at a certain network interface, very different elements, such as log files or running processes, so they are quite dependent on the veracity of this information [2]. On the other hand, Network-based IDS (NIDS) analyze information about an entire network. They are generally installed on different computers located at certain points of the same computer network to have a global view of what is happening in the network. These systems can analyze such traffic using a variety of techniques to determine whether its purpose is illegitimate [3]. When the IDS detects a packet or set of packets that may correspond to a computer attack, it generates an alarm, which typically has to be attended to by a security technician. It would be desirable for the program to react only to packets that correspond to an attack, but this is impossible. The IDS will never know for sure whether a packet corresponds to an attack or not; the purpose of this document is to determine to what extent machine learning algorithms can be applied to intrusion detection and which algorithms are most appropriate for detecting which types of attacks based on certain metrics. To do this, a practical study will be carried out by applying a series of machine learning algorithms to a dataset containing information corresponding to a set of network packets. Part of these will correspond to legitimate network use and part to

computer attacks [4]. The packets corresponding to attacks will be marked in advance to evaluate the effectiveness of each algorithm. The aim is to determine which algorithms are most suitable based on which metrics are considered most important (false positive rate, false negative rate, precision, etc.) and which types of attacks are a priority to detect [5].

Numerous types of attacks can affect networked systems and there are many different ways to classify them. Probing/Scanning are attacks that attempt to scan a network or machine for information and possible vulnerabilities [6]. User to Root (U2R) attacks in which the attacker already has access to a resource on a system and tries to exploit a vulnerability at the application or operating system level to obtain higher privileges. Remote to Local (R2L) are attacks in which an attempt is made to exploit vulnerabilities in a system remotely in an attempt to obtain local user privileges. They are generally preceded by a scan of network resources. Denial of Service (DoS) are attacks that seek to compromise the availability of a system's data. They generally block a machine or network resource by saturating it with illegitimate requests so that its legitimate users do not have access to it [7].

There are many other more exhaustive classifications for computer attacks. From the point of view of network security, one of the most widely used is the one proposed by The Mitre Corporation, which classifies the different techniques used by attackers into 12 groups based on their objectives [7]. This classification is public and is currently used by companies and organizations around the world to evaluate their degree of protection against the different types of known computer attacks. It is therefore important to keep in mind at all times that not all attacks are detectable and that an IDS should always be part of a broader network security infrastructure, which includes other security mechanisms that can prevent other types of attacks.

1.1 Machine learning and IDS Classification algorithms

Detecting intrusions in NIDS using anomaly detection is often considered a classification problem, and several very different approaches to it have been studied in recent years [9].

When detecting intrusions, one may try to determine a binary classification of the detected events, classifying them as malicious or legitimate. Since typically the events classified as malicious will trigger an alarm and will require subsequent review by a security technician, this distinction is not sufficient. It is necessary to be able to classify the event as belonging to a computer attack family whenever possible. So that the technician can correctly review the security incident and determine if an attack has occurred. It will be necessary to know at least the type of attack for this system to be applied in practice and to be able to determine if the attack has had an impact, if it was a false positive, etc. Without this characteristic, the detection of anomalies could not be implemented effectively in real Intrusion Detection Systems.

In addition to classification techniques, the use of clustering techniques [9] for intrusion detection is also quite common. With this in mind, different approaches can be used when applying machine learning algorithms in a NIDS. Simple classifiers. A single ML algorithm can be used to try to group the different events analysed as legitimate or malicious and, in the latter case, to determine the type of attack. Hybrid classifiers. Another option that is widely used in the literature [10] is to combine several different algorithms, applying one to the result of the previous one. A common option is to apply a clustering algorithm before applying a classifier. Several classifiers are also often applied (either different algorithms or the same one several times) to carry out successive classifications on each type of attack. In this way, each classifier only has to make a binary classification.

Ensemble classifiers: This technique consists of applying several classifiers on the same dataset and combining the results of all of them. The most common way of doing this is to classify each element by the category chosen by most algorithms.

Similarly, as already mentioned, the detection of anomalies by an IDS can be carried out both in isolation and in conjunction with signature-based detection. The specific techniques that will be implemented for the study will be described later. The dataset will be selected from among those available on the Internet for this purpose. With a focus on Random Forest (RF) and Gradient Boosting Machine (GBM) as the best-performing machine learning algorithms for intrusion detection on the NSL KDD dataset, this work novelty resides in its thorough evaluation of competing algorithms.

2. Methodology

The first step will be to select the metrics that will be taken into account to determine the suitability of a machine learning algorithm for classifying a packet or packet flow into the five categories that interest us: Legitimate, Probe, U2R, R2L or DoS.

Secondly, a dataset will be selected with a capture of the traffic of a certain network, in which each packet appears marked with the category to which it corresponds, to apply supervised learning techniques. There are datasets available on the Internet that meet these characteristics. They will be studied and one will be selected based on the format of its data, the amount of information it contains, the variety of attacks it shows and how recent its data is. An analysis and presentation of the main relevant characteristics of the dataset will also be carried out, the distribution of attacks for the total number of entries, etc. Next, the attributes of said dataset that are to be taken into account for the detection of attacks will be selected. This selection will be made automatically using certain algorithms designed for this purpose. The experiment will be used to study the effectiveness of the classification algorithms based on the chosen characteristics.

Thirdly, a set of classification algorithms will be selected to be considered in the experiment. They will be selected based on their effectiveness as general classification algorithms and their use in other similar studies, after a review of the available literature. Many of these algorithms will have parameters that will need to be determined before applying them to the data.

Finally, each of the algorithms will be applied to the data in the dataset. The dataset will be divided into two parts. A majority of this data will be used to train the classification algorithm, i.e. to allow it to form a model of legitimate network traffic from the labels of each packet. The remaining fraction will be used to see the effectiveness of this algorithm when classifying once the model has been created. For each algorithm, the results will be collected and

the performance metrics mentioned above will be calculated. These metrics will also be calculated for each of the main types of attack marked in the dataset. This information will be compared between the different algorithms and conclusions will be drawn about the effectiveness of these in each case.

2.1 Technology used

Execution of a Python script that applies the machine learning algorithm to the dataset for each of the algorithms. Initially, train the model, followed by evaluating its efficacy. A validation phase between the two parties will be undertaken. This script will utilise the *Scikit-Learn* and *Plotly* libraries. Scikit-Learn (*Sklearn*) is an open-source machine-learning library for Python. This library encompasses the primary machine learning algorithms, along with tools for the preliminary and subsequent processing of data. These tools encompass utilities for dataset partitioning and the extraction and analysis of data derived from a test. *Plotly* is a library for graphical data representation that we will utilise to depict the test results.

2.2 Metric selection

Various metrics may be evaluated to ascertain the appropriateness of a classification algorithm. We must evaluate the algorithm's efficacy in classifying the analysed elements [11]. We will employ the subsequent metrics for measurement [12]:

True Positive (TP) denotes the quantity of malicious packets accurately identified as such. True Negative (TN) denotes the number of legitimate packets accurately identified as such. False Negative (FN) denotes the quantity of malicious packets erroneously identified as legitimate. False Positive (FP) denotes the quantity of legitimate packets erroneously identified as malicious.

Accuracy: Represents the fraction of correctly classified packets compared to total classified packets. It is also sometimes known as precision, not to be confused with accuracy which we will discuss later. It is one of the most basic performance metrics in a classification algorithm. It is calculated using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (Eq\ 1)$$ [11-12]

Precision (P, precision or positive predictive value) [1]: Represents the percentage of positives that correspond to an attack. High precision means few false positives.

$$Precision = \frac{TP}{TP + FP} \quad (Eq\ 2)$$ [11-12]

Sensitivity (S, True Positive Rate or recall): Represents the percentage of attacks that have been correctly detected. High FFN means few false negatives.

$$Sensitivity = \frac{TP}{TP + FN} \quad (Eq\ 3)$$ [11-12]

F-score: Harmonic mean of precision and sensitivity. Like accuracy, it serves to get a general idea of how good a classifier is. It may not be the best metric for intrusion detection, since it assumes that precision and sensitivity are equally important. In our case, this is not the case and for this reason, we will first look at each one separately, giving special importance to precision. It consists of a beta parameter to regulate the importance of each one, although in this case, we will leave it at 1 so that both have the same. In this way, the F-score would be calculated as:

$$F - score = \frac{2(P \times S)}{P + S} \quad (Eq\ 3)$$ [11-12]

A classification algorithm that maximizes the values of accuracy, precision and sensitivity will therefore be desired. The importance of each of these metrics will depend on the type of system being considered. In general, in an IDS, maximizing precision will be prioritized, because a high rate of false positives can imply an excess of work for security technicians when checking alarms. Therefore, the relative priority of one or another metric will also depend on how critical the security of the system is. In our case, and with the main objective being the integration of anomaly detection with the signature-based techniques already used, the main objective will be to maximize precision, so that when security technicians receive an alert generated by AI, they know that it is most likely a real alert and examine it carefully.

For multi-class classification (3 or more), which we will attempt in this study, these metrics will be calculated for each of the classes. Only in the case of accuracy will we try to approximate its value for the classifier as a whole. Since the proportion of network traffic generated by the different types of attacks is very different, we will use balanced accuracy, which gives a different weight to the examples of each of the classes depending on the presence of that type of examples in the dataset. In this way, the classes that appear in a lower proportion will have a greater weight, so that the fact that the IDS can classify them or not will consequently affect the value of the metric. We are not interested in an IDS that can perfectly classify Probe and DoS attacks, which generate a much larger amount of traffic if it is not also able to classify R2L and U2R attacks.

On the other hand, when evaluating an algorithm for its application in an IDS, it is also necessary to analyze its computational performance when creating a model of a network and classifying the events that occur in it. The main metrics used to determine this are the time taken by the algorithm to create a model of the system (Training Time, TRT) and, above all, to classify the packets once the model has been created (Testing Time, TST) [13]. Even if an algorithm were able to perfectly classify network traffic, a high processing time could make it useless for application in an IDS.

2.3 Network traffic datasets

In the case of datasets for evaluating intrusion detection systems, this translates into whether each packet or packet flow has the type of traffic associated with it: whether it is normal traffic, whether it corresponds to an attack, the type of attack, etc. Only a dataset containing these labels can be used in a supervised learning process.

There are multiple datasets with network traffic information that can be used to evaluate the performance of an IDS. Not all of them are publicly available; for this study, we focus on analysing some of those that are, and that have been used in similar experiments [14]. The characteristics of each dataset (format and labelling) will determine how the

machine learning algorithms process them to obtain information, and the performance of the algorithms when processing them. These were the datasets considered:

- **KDD CUP 1999:** KDD CUP 99 contains flow-based data enriched with data from system logs and network packets involved in the communication. It contains more than 20 types of attacks along with normal traffic and is already divided into a training subset for ML algorithms and another for testing. It is discarded because it is outdated, although it has good characteristics for the experiment.
- **NSL KDD:** It is an update to KDD CUP 99 that addresses some of the redundancy issues raised in [12], but it still contains some of the issues raised by McHugh in [15]. It, like KDD CUP 99, contains data based on flows that have been enriched with information from the hosts and packets involved in the communication. It has the advantage of allowing comparison of studies conducted on it because its use is quite standardised.
- **CIC-IDS2017:** Another of the datasets offered by the CIC. It contains information based on bidirectional flows, with more than 80 characteristics in total. It has been generated in an emulated network and is divided into different files corresponding to the periods in which the different types of attacks were emulated. Contains the most recent and common attack types in real networks at the time of its publication.
- **CSE-CIC-IDS2018:** The latest of the datasets offered by the CIC in collaboration with the Communications Security Establishment (CSE). It offers both raw data (network traffic capture and equipment logs) and flow-based data. It focuses on dividing the dataset into profiles that can be combined in the desired proportions to simulate the behaviour of a given network. This can be quite useful for different entities to experiment with a dataset similar to their network, although in principle it does not offer any advantage for our experiment. It is preferable to use flow-based datasets that have been enhanced with additional data and claims [16]. The NSL KDD dataset was chosen for the current study due to its standard nature and this reason. It is labelled as well, which is essentially a prerequisite for intrusion detection.
- **The NSL KDD dataset:** The dataset contains data in a flow-based format, enriched with data on the packets and devices involved in the communication. The dataset has a total of 41 features in addition to the label and the difficulty level. These features can be divided into 4 large groups. Table 1 shows the basic attributes of a packet flow. Table 2 shows the attributes related to the content of the packets that make up the flow, i.e. information related to the application layer. On the other hand, tables 3 and 4 show information on other network packet flows that share some characteristic with the flow to which the entry corresponds. This information is of vital importance to detect certain types of attacks (network scans, DDoS attacks, etc.). In the case of Table 3, this is information about other connections temporally related to the given one, while in Table 4 they are related by IP addresses or port numbers.
- In this way, characteristics can be distinguished in text string format, others in binary format, and others in numerical format. Later, it will be necessary to process the dataset to have a homogeneous type of data with which to feed the machine learning algorithms.
- Finally, each entry in the dataset includes a label with the flow classification. The flows are classified as normal or as corresponding to a computer attack. The dataset includes flows corresponding to 40 different types of attacks, each of which can be classified into one of the categories. This correspondence is shown in Table 5. On the other hand, a final characteristic is included that indicates the degree of difficulty when classifying each entry. For this study, we will ignore this characteristic.

Table 1: Basic flow attributes

No.	Name	Description
1	*Duration*	Duration of the communication
2	*Prpe*	Transport layer protocol used in the communication (text string)
3	*Service*	Network service used by the client (text string)
4	*Flag*	State of the connection (text string)
5	*Src bytes*	Number of bytes sent from the source to the destination during the connection.
6	*Dst bytes*	Number of bytes sent from the destination to the source during the connection.
7	*Land*	Indicates whether the source and destination IP addresses and port numbers are equal (1) or not (0)
8	*Wrong*	fragment Number of erroneous fragments counted in the connection
9	*Urgent*	The number of packets with the urgent bit activated counted in the connection.

Table 2: Attributes related to package content

No.	Name	Description
10	*Hot*	A number of compromise indicators detected in the packet content, such as: 'entering a system'
11	*Num_failed_logins*	Number of failed login attempts
12	*Logged_in*	Indicates whether a login was successful (1) or not (0)
13	*Num_compromised*	Number of possible compromise indicators detected in the packet content
14	*root_shell*	Indicates whether an administrator console was obtained (1) or not (0)
15	*You_attempted*	Indicates whether the command 'su root' was attempted to be executed during the connection (1) or not (0)
16	*Root_number*	Number of accesses and operations performed as root during the connection
17	*Num_file_creations*	Number of file creation operations performed during the connection
18	*Num_shells*	Number of command interpreters used during the connection
19	*Num_access_files*	Number of control file access operations
20	*Num_outbound_cmds*	Number of external connections made in a session FTP
21	*Is_hot_login*	Indicates whether the login corresponds to one of those considered potentially dangerous (such as root or admin users) (1) or not (0)
22	*Is_guest_login*	Indicates whether the login is as a guest (1) or not (0)

Table 3: Attributes about traffic temporally related to the flow

No.	Name	Description
23	*Count*	Number of connections to the same destination host as the current one in the last 2 seconds
24	*Srv_count*	Number of connections to the same service (destination port) as the current one in the last 2 seconds
25	*Serror_rate*	Percentage of connections with the flag (4) s0, s1, s2 or s3 activated among those counted in count (23)
26	*Srv_serror_rate*	Percentage of connections with the flag (4) s0, s1, s2 or s3 activated among those counted in srv count (24)
27	*Rerror_rate*	Percentage of connections with the flag (4) REJ activated among those counted in count (23)
28	*Srv_rerror_rate*	Percentage of connections with the flag (4) REJ activated among those counted in count (23) srv count (24)
29	*Same_srv_rate*	Percentage of connections against the same service (destination port) as the current one, among those counted in count (23)
30	*Diff_srv_rate*	Percentage of connections against different services (destination port) than the current one, among those counted in count (23)
31	*Srv_diff_host_rate*	Percentage of connections against different destination host than the current one, among those counted in srv count (24)

Table 4: Host-Related Traffic Attributes

No.	Name	Description
32	*dst_host_count*	Number of connections with the same destination IP address as the current one
33	*dst_host_srv_count*	Number of connections with the same port number as the current one
34	*dst_host_same_srv_rate*	Percentage of connections to the same destination port, among those counted in dst host count (32)
35	*dst_host_diff_srv_rate*	Percentage of connections to different destination ports, among those counted in dst host count (32)
36	*dst_host_same_src_port_rate*	Percentage of connections from the same source port, among those counted in dst host srv count (33)
37	*dst_host_srv_diff_host_rate*	Percentage of connections to different destination hosts, among those counted in dst host srv count (33)
38	*dst_host_serror_rate*	Percentage of connections with flags (4) s0, s1, s2 or s3 enabled, among those counted in dst host count (32)
39	*dst_host_srv_serror_rate*	Percentage of connections with flags (4) s0, s1, s2 or s3 enabled s3 enabled, among those counted in dst host srv count (33)
40	z	Percentage of connections with the (4) REJ flag enabled, among those counted in dst host count (32)
41	*dst_host_srv_rerror_rate*	Percentage of connections with the (4) REJ flag enabled, among those counted in dst host srv count (33)

Table 5: Correspondence between attacks collected in NSL KDD and their category

Category	Attacks
Probe	*Satan, Ipsweep, portsweep, ipsweep, mscan, Nmap, Portsweep, Mscan, Saint*
R2L	*Guess Password, Ftp write, imap, xsnoop, named, Phf, Multihop, Warezmas ter, Warezclient, Spy, X_lock, X_snoop, Snmpguess, Snmpge attack, Httptunnel, Sendmail,*
U2R	*sqlattack, perl, xterm, Buffer overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps*
DoS	*teardrop, winnuke, syndrop, newtear, bonk, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Ud storm, Process table, Worm, Mailbomb*

2.4 Dataset processing

2.4.1. Dataset splitting: Training Set and Test Set: Scikit-Learn provides tools to automatically split a dataset into two random subsets for learning and testing. By default, the Training Set will comprise 75% of the total, with the remaining 25% used to evaluate the classifier's effectiveness. Table 6 shows the number of instances of each class in both the Training Set and the Test Set using this configuration.

Table 6: Number of instances in the split NSL KDD dataset

Class	Instances in Training Set	Instances in the test set
Normal	57196	19087
Probe	10423	3514
R2L	2863	978
U2R	89	29
DoS	39702	13151

2.4.2. Feature selection: Not all features offered by the datasets offer the same information when classifying the data. Using all the features of the data can negatively impact both the computational performance and the effectiveness of the classification algorithms. Before the training phase of a dataset, it is common to preprocess it to remove redundant or non-relevant features.

Feature selection is not always performed in the field of intrusion detection [17], since many datasets offer their data in formats that show only the features considered relevant when detecting attacks. Nevertheless, in this paper, we will consider some methods to remove irrelevant features and apply some of them if relevant. It has been observed that, in our case, feature selection significantly improves the training time (TRT) of some classifiers. Furthermore, if it is adequate, it also improves the effectiveness in some cases. Therefore, it is also intended to carry out a study of how the number of features and the method to select them affect the effectiveness of the classifiers.

Two automatic methods of feature selection will be applied that allow processing the labelled dataset and determining which features of it are less relevant to classify the nature of the network events. The first method chosen is Recursive Feature Elimination (RFE). The algorithm performs several iterations in which it calculates the relevance of the features of the dataset. It starts by doing so with all of them and eliminates the least relevant in each iteration. It has two important parameters: the classification method used to evaluate the relevance of the features (estimator) and the desired number of features. In our case, the classification method will be Decision Trees, since it is one of the fastest classifiers and many of the composite methods that we are going to use are based on it (we will see this later). To determine the optimal number of features, one option would be to use a variation of RFE that allows this to be done automatically. RFECV is a version of RFE that performs cross-validation using the data in the dataset to determine the optimal number of features, so it is not necessary to specify it in advance. However, we want to evaluate the performance of the classifiers for each feature and selection method, so this is not useful, we will use RFE for each case.

The other feature selection method is the statistical method of Principal Component Analysis (PCA). The idea of this method is to convert the initial set of features, probably correlated, into a smaller set of features without linear correlation that preserves as much information as possible with respect to the original set. It has the advantage that it is not necessary to select a classifier as an estimator of the relevance of each feature so that no biases are introduced when applying it. It has the disadvantage that it does not take into account the classification problem being addressed when selecting the features.

Both RFE and PCA will be applied to the dataset to select 5, 10, 15, 20 and 25 features. The average effectiveness of the classifiers will be evaluated in all cases with respect to the same metric so that the optimal number of features and the best method to select them are determined. The metric with respect to which we will compare them will be the average balanced accuracy of all the classifiers. Once several features and a selection method have been selected, we will proceed to evaluate the effectiveness of the classifiers in that case, taking into account all the metrics described above.

2.4.3. Preprocessing: The classification algorithms implemented in *ScikitLearn* need to process numeric data. Therefore, all the features in the dataset that are in string format (including the label) will be replaced by a unique integer for each string. All the features will be stored in 32-bit float format. In addition, all NaNs that can be found will be replaced by numbers.

On the other hand, some of the classifiers require that the data they work with has been scaled or normalized, so the data will be treated in the appropriate way in each case.

In addition, the labels corresponding to all groups will be grouped to replace those corresponding to specific attacks by the family to which they belong, according to the information collected in Table 5.

2.4.4. ML algorithms to be studied: For this study, only those algorithms that have obtained the best results in the studies carried out so far will be considered, for which we will look at the information collected in [16-17].

2.5 Simple classifiers

2.5.1 Algorithms selected

- **Decision Trees (DT) [18]:** The algorithm implemented by Scikit-Learn is CART (Classification and Regression Trees). This is a modified version of the ID3 and C4.5 algorithms. This algorithm creates a binary decision tree during the supervised training phase, which is interpreted as if-then rules. At each

node, a decision is made based on the value of a certain attribute. The decisions and their order are determined based on their relevance to classify the data during the training phase [7].

- **Support Vector Machines (SVM):** SVM is one of the most widely used approaches in supervised machine learning [19]. To perform the calculation efficiently, kernel functions are used, which allow obtaining a measure of the degree of similarity of two points in a space of greater dimensionality than the given data, without having to calculate the coordinates of the points in said space. In the classification using SVM, different kernel functions can be used (linear, Gaussian, polynomial...) with different parameters. The suitability of one or the other will depend largely on the input data. As for the other parameters of the classifier, the penalty parameter C or cost is important. C determines the degree of importance of the misclassified points. A higher C implies that more importance is given to correctly classifying all the points in the Training Set than to leaving space in the models of each class for the classification of future data. The selection of these parameters will be done automatically with cross-validation tools, also looking at the parameters used in other studies such as [20] and [21]. We tested a series of parameters considered in these studies using the cross-validation tool Randomized Search CV so that the optimal ones were Gaussian kernel, C=5000 and gamma = 0.1. Later on, the selection of features will be seen in more detail.
- **Artificial Neural Networks (ANN):** This is a family of machine learning algorithms that process information using units that imitate the behaviour of neurons in the human brain. Each of these units (called neurons) takes a set of data as input and produces a single output data, which can be used as input by other processing units. We are going to focus on the most widely used architecture: feedforward neural networks, where the neural network can be represented as a directed graph without cycles, such that each node is a neuron and each link represents that the output of one neuron is the input of another. For simplicity, it is considered that the network is made up of layers, such that the neurons in layer i receive as input the output of those in layer i-1 (except in layer 0) and pass their output to layer i+1 (except in the last layer). The first layer has n+1 neurons, where n is the dimensionality of the input data. The output of each neuron in the first layer after feeding it with the vector x is each of the attributes of the vector, except for the vector n+1 which is always 1. The neurons in the intermediate layers or hidden layers receive as input the weighted sum of the outputs of the neurons in the previous layer that connect to them, and calculate the output by applying a function (activation function) to that value. The architecture of a neural network is called the set of its nodes, its links and the activation function applied to its neurons. The learning process is carried out by applying different weights to the links of a fixed architecture. Each combination of possible weights in the links is a hypothesis. During the supervised learning process, they will be adjusted so that the output layer, with a single node, produces the desired output [22]. One of the most widely used architectures is the Multilayer Perceptron (MLP), which will be used for this study.
- **Naive Bayes Classifier (NB):** This is a generative model [22]. The problem is that it is usually more complicated to determine the distribution followed by the input data than to create an adequate classifier without doing so, as occurred in the algorithms explained above. Naive Bayes is a simplification of the Optimal Bayes Classifier (based on Bayes' Theorem) in which the number of parameters to be estimated is significantly reduced by making an assumption. The parameters are estimated using the maximum likelihood principle.
- It will be necessary to determine the optimal parameters for these algorithms.

2.5.2 Composite classifiers

Composite classifiers are those in which the results of several simple classifiers, whether of the same or different type, are used to perform the final classification based on the results of these.

There are different methods to pool the results of the strong classifiers to make a final decision.

- **Bagging:** Different models are built on different portions of the original dataset. Generally, it is done with simple classifiers of the same type.
- **Boosting:** Different models are built. The first one learns to classify the data, while the following ones try to learn to correct the prediction errors of the previous one.
- **Voting:** Different models are built, generally with simple classifiers of different types, and the results of these are combined through a voting process or some simple statistical calculation.

For this study, we will consider some examples of each of these methods, of those implemented by Scikit Learn. These have been the ones we have selected.

- **Random Forest (RF):** This is a classifier that consists of a collection of decision trees (DT), in which each tree is built by applying an algorithm to a subset of the Training Set from a random vector so that this vector determines how the tree is generated. A bagging technique is therefore used. The final results are obtained by voting on the predictions of each of the trees.
- **Adaptive Boosting or AdaBoost:** The AdaBoost classifier is based on the creation of a set of weak classifiers, typically decision trees (DT). First, the first tree is built according to a classification algorithm (for example ID3 or CART) from the data in the Training Set. Then, the effectiveness of this model is evaluated and a second tree is created, giving greater weight to those elements of the Training Set that have not been correctly classified by the first one. This process is repeated successively.
- **Gradient Boosting Machine (GBM):** This classifier differs from AdaBoost in the way it tries to correct the error of simple classifiers. While AdaBoost does this by adjusting the weights of the elements in the Training Set, GBM does this by adjusting the parameters of the loss function. In classification, a loss function calculates the value of the penalty due to the incorrect classification of data. In this way, the classification

problem is treated as an optimization problem in which the aim is to minimize the loss of the model by adding simple classifiers to the set. Different standard loss functions can be used for this, or a specific one can be defined.

- **Voting Classifier (VC):** This is one of the simplest ways of combining several classifiers. The final classification of each example is simply determined by a majority vote of all classifiers. The class can also be predicted by calculating the average of the probabilities given by each simple classifier to each class and choosing the one with the highest average. This is what we will do in this study. Different types of classifiers can be used.

2.6 Implementation

The implementation will consist of a main script in Python3 based on the Scikit Learn libraries and a directory structure. In addition, it will use an auxiliary script for each evaluated classifier and another auxiliary script for graph extraction. All the necessary files are available in GitLab 1.

The main script, *test.py*, will read the data and execute other auxiliary scripts contained in these directories; the script itself will create, if they do not exist, the output directories where it will save the results of the evaluation of the classifiers. Each execution of the script will overwrite this data. To replicate the experiment carried out, it is only necessary to clone the Gitlab repository and execute the Test.py script.

The directory structure is the following, as it is available in GitLab and represented in Figure 1. A base directory ./ with the main script, which carries out the entire study regarding the classifiers. Two subdirectories are required: the *./data* subdirectory should contain the KDD NSL dataset in CSV format and the ./classifiers subdirectory, with a script for each of the classifiers to be evaluated, following the same structure in all cases. In addition, the main script itself will create, if they do not exist, 3 output subdirectories where it will write data. In *./models*, the Scikit Learn models that should be used in the future are saved. In *./results*, the files with the performance metrics resulting from the study will be saved. At the root, there will be a file that allows comparing the efficiency of the classifiers according to the selection of characteristics, and a series of subdirectories with the metrics corresponding to the study with each subset of characteristics. Finally, in *./graphs* the same structure will be replicated to store the graphs and tables that allow visualizing this data.

2.6.1. Selection of parameters

To implement the script, we must first select the appropriate parameters for all the classifiers. In our case, the classifiers in which it was necessary to consider some parameters a priori are ANN and SVM.

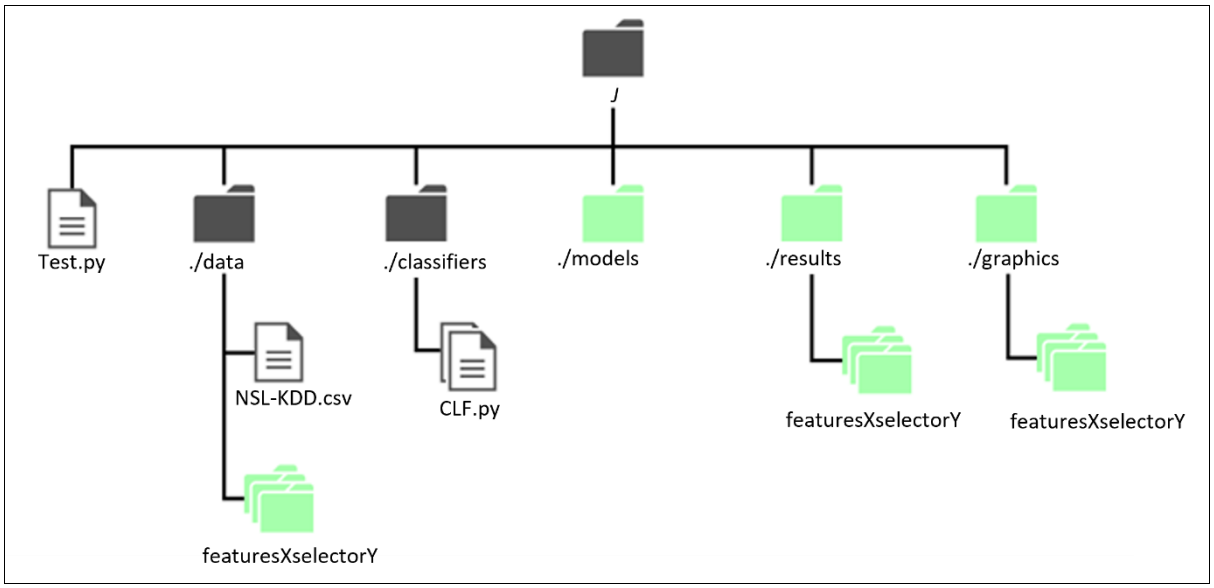


Fig 1: Directory tree used by the script. In a lighter color, the directories are created at runtime.

In the case of ANN, the parameter to be determined is the structure of the neural network. A classic three-layer structure was chosen, selecting the number of neurons in the hidden layer according to what was explained in [19]. In this way, and since the effectiveness of the classifiers with different numbers of characteristics in the input is going to be evaluated, the number of nodes in the hidden layer will be equal to this. In the case of SVM, we will use one of the tools provided by Scikit Learn to perform cross-validati, *RandomizedSearchCV*, Through this, we test different types of kernels (linear, polynomial, Gaussian, sigmoid), different values for the penalty parameter C (10, 100, 1000, 5000, 10000) and different values for the coefficient of the RBF, polynomial and sigmoid kernels, gamma (0.00001, 0.001, 0.1). These test values were obtained by reviewing the literature and the values typically used in intrusion detection. Using this random search tool, these parameters were optimized in terms of the accuracy of the classifier, not in terms of the time taken, resulting in the selection of the RBF (Radial Basis Function) kernel, C=5000 and gamma = 0.1. The RBF kernel seemed to fit the data distribution better. In addition, it allows its treatment in a much shorter training time than the other two kernels that offer better results: the polynomial and the linear. The value of the penalty parameter has been automatically selected, giving a value much higher than that used in the consulted bibliography [21].

2.6.2. Steps performed by the script

Once the script associated with each of the classifiers has been created, the main script can be executed. The steps followed by the program are the following.

Loading the dataset and processing the dataset. The CSV file with the KDD NSL dataset located in ./data is opened, under the name 'NSL-KDD.csv'. The labels and the characteristics that are in string format are encoded so that the dataset can be applied to the machine learning algorithms, which require numerical data. The labels of the dataset are grouped. The model used to encode them is saved so that the original values can be recovered later.

Selection of characteristics. The features with the most information are selected so that 5, 10, 15, 20 and 25 features remain; using two feature selection algorithms, RFE and PCA, as explained in section 6.5.2. Versions of the dataset with the selected features are saved in subdirectories of the *./data* folder. One subdirectory for each number of features evaluated and for each selection algorithm.

Classifier evaluation. The auxiliary scripts in the *./classifiers* directory are executed, and each classifier is evaluated for each of the selected feature subsets, saving the extracted metrics to disk in the *./results* subdirectory. The evaluation of each classifier consists of three steps. First, the selected dataset is preprocessed, scaling or normalizing the data as necessary for each classifier. Second, the classifier is trained (training phase) with the Training Set. Next, we try to predict the classes of the Test Set (classification phase) and extract the performance metrics, saving them on disk in the corresponding subdirectory (again, one for each selector and number of features).

2.6.3. Extracting Results

Extracting results: An additional script has been developed that allows extracting graphs and tables from the data generated by the test. It is located in the root folder under the name `ProcessResults.py`.

This script will read the files created by each classifier in the *./results* folder and its subdirectories; and will create tables and graphs from them, which will make it easier for us to compare the different classifiers. To create the different graphs, we will use the Plotly Python libraries. The format of the tables and graphs will be the one offered by them.

3. Results and Discussion

First, analyse the influence of the number of features and their selection method on the effectiveness of the classifiers. Figure 2 shows the average balanced accuracy of the classification algorithms for each number, according to what was defined; the selection methods being RFE and PCA. That is, the balanced accuracy of all the classifiers has been obtained for each number of features and selection method and the average of all of them has been obtained, to have a single metric with which to compare all the cases.

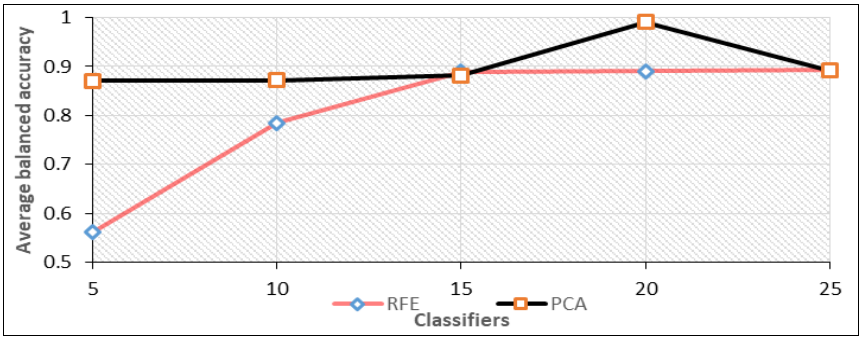


Fig 2: Average balanced accuracy of classifiers as a function of the number of features, selected using RFE

Looking at the graphs we see that in general, the accuracy increases as the number of features increases, especially using PCA. Reviewing the data we see that the accuracy values are usually higher for RFE in most cases, especially for small numbers of features. In the cases where it is better for PCA, there is hardly any difference with respect to RFE. This also occurs for those algorithms not based on Decision Trees, which is the estimator used to select the features. There are also differences with respect to training times. For the same number of features, they are usually higher in the case of PCA. This may be because classifiers have more difficulties in creating a model using features that are less relevant to the problem. Thus, we consider that the best selection method is RFE, and a good number of features (20), for which in this case the highest accuracy was obtained. Table 7 shows the attributes selected from those previously presented.

Table 7: Attributes selected by RFE

No.	Name	Description
1	Duration	Communication duration
2	Protocol_type	Transport layer protocol used in the communication
3	Service	Network service used by the client
4	Flag	Connection status
5	Src_bytes	Number of bytes sent from source to destination during the connection.
6	Dst_bytes	Number of bytes sent from destination to source during the connection.
11	Num_failed_logins	Number of failed login attempts
22	Is_guest_login	Indicates whether the login is as a guest (1) or not (0)
23	Count	Number of connections to the same destination host as the current one in the last 2 seconds
30	Diff_srv_rate	Percentage of connections (23) against different services (destination port) than the current one.
32	Dst_host _ount	Number of connections with the same destination IP address as the current one

33	Dst_host_srv_count	Number of connections with the same port number as the current one
34	Dst_host_same_srv rate	Percentage of connections from (32) to the same destination port.
35	Dst_host_diff_srv rate	Percentage of connections from (32) to different destination ports.
36	Dst_host_same_src port rate	Percentage of connections from (33) from the same source port.
37	Dst_host_srv_diff_host_rate	Percentage of connections from (33) to different destination hosts.
38	Dst_host_error_rate	Percentage of connections from (32) with the (4) s0, s1, s2 or s3 flags set.
39	Dst_host_srv_serror rate	Percentage of connections from (33) with the (4) s0, s1, s2 or s3 flags set.
40	Dst_host_error_rate	Percentage of connections from (32) with the (4) REJ flag set.
41	Dst_host_srv_error rate	Percentage of connections from (33) with the (4) REJ flag set.

Once these parameters are set, we can analyze the effectiveness of all the classifiers with respect to the metrics. We will focus mainly on four metrics: accuracy (balanced, used as an indicator of the overall effectiveness of a classifier), precision (to try to minimize the number of false positives by a possible ML-based IDS) and training and classification times (to make it possible to apply the classifier in a real-time IDS).

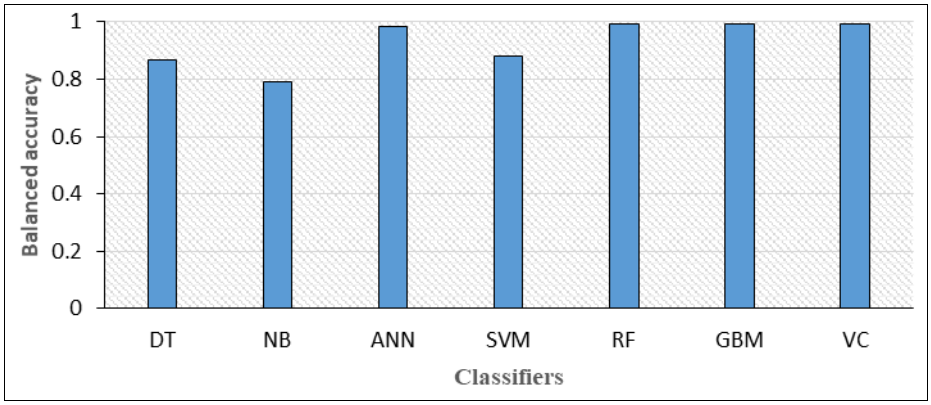


Fig 3: Balanced accuracy of each classifier, for 20 features selected by RFE

We start with accuracy, which is what gives us a general idea of how well each algorithm has classified the samples in the 5 categories. Figure 3 gives us an idea of the value for each of the classifiers. In this way, we see that the classifiers that obtained the highest accuracy were ANN, RF, and GBM. However, reviewing the data, we see that ANN failed to classify any instance of U2R attacks, which appear in a lower proportion, which rules it out as a good classifier. We will see this in more detail later.

We now look at the accuracy, the fraction of really positive positives. A low accuracy would indicate that a large number of network flows have been classified as attacks when they were not, or that they have been classified as the wrong type of attack. It does not make sense to take into account the overall accuracy of each classifier, but rather we should focus on reviewing the accuracy of each classifier for each class. More specifically, when classifying the 4 types of attacks, which is what would generate false alarms and an overload of work for security technicians. Therefore, we will look at tables 8, 9 and 10; which show us the performance metrics for each class of the ANN, RF and GBM classifiers, respectively. Here we see that ANN has not been able to classify U2R-type attacks in any case and that its sensitivity and precision when classifying R2L-type attacks are also significantly lower.

Table 8: Performance metrics of the DT, NB and ANN classifier, for 20 features selected by RFE

	DT classifier				NB classifier				ANN classifier			
Class/metrics	**F Score**	**Precision**	**Sensitivity**	**Total**	**F Score**	**Precision**	**Sensitivity**	**Total**	**F Score**	**Precision**	**Sensitivity**	**Total**
DOS	0.999	0.998	0.999	13151	0.828	0.751	0.908	13151	0.987	0.996	0.979	13151
Normal	0.995	0.995	0.995	19087	0.872	0.87	0.873	19087	0.982	0.976	0.978	19087
Probe	0.994	0.994	0.995	3514	0.388	0.553	0.298	3514	0.971	0.964	0.978	3514
R2L	0.944	0.947	0.941	978	0.045	1	0.023	978	0.881	0.896	0.865	978
U2R	0.525	0.5	0.552	28	0	0	0	28	0	0	0	28

Table 9: Performance metrics of the SVM and RF classifier, for 20 features selected by RFE

	SVM classifier				RF classifier			
Class/metrics	**F Score**	**Precision**	**Sensitivity**	**Total**	**F Score**	**Precision**	**Sensitivity**	**Total**
DOS	0.987	0.996	0.979	13151	0.998	0.997	0.979	13151
Normal	0.982	0.976	0.978	19087	0.951	0.972	0.991	19087
Probe	0.971	0.964	0.978	3514	0.964	0.987	0.943	3514
R2L	0.881	0.896	0.865	978	0.918	0.908	0.929	978
U2R	0	0	0	28	0.326	0.5	0.241	28

Table 10: Performance metrics of the GBM and VC classifier, for 20 features selected by RFE

	GBM classifier				VC classifier			
Class/Metrics	**F Score**	**Precision**	**Sensitivity**	**Total**	**F Score**	**Precision**	**Sensitivity**	**Total**
DOS	0.998	0.998	0.998	13151	0.998	0.997	0.999	13151
Normal	0.994	0.993	0.995	19087	0.993	0.992	0.995	19087
Probe	0.986	0.984	0.978	3514	0.987	0.987	0.978	3514
R2L	0.942	0.971	0.914	978	0.935	0.959	0.911	978
U2R	0.731	0.826	0.655	28	0.176	0.6	0.103	28

Finally, we look at the training and classification times, shown in Figure 4. The algorithms with the longest training time have been, by far, ANN and VC; followed far behind by GBM and SVM. This training time may not greatly condition the application of the algorithm to an IDS if the algorithm is not intended to be trained regularly. In that case, it would only affect the installation and start-up time of the IDS. As for the classification times, the only one with a significantly higher time is SVM, with 2.8 seconds for the 37,130 samples of the Test Set. This could make it a poor candidate for its application in a real-time IDS. Of the remaining classifiers, the composites are the ones with the highest classification times, being up to 0.3 seconds in the case of RF. However, it is not considered a time that in any way prevents its application in an IDS.

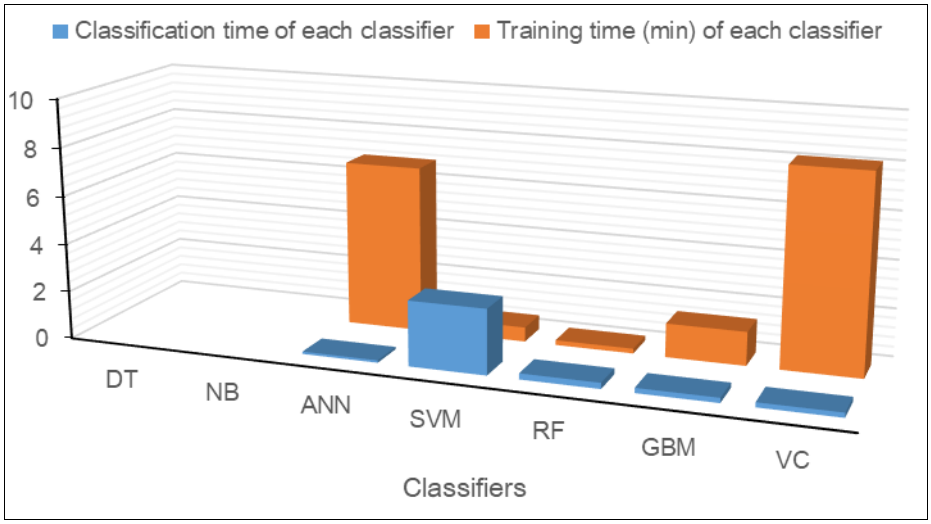


Fig 4: Training time (min) and Classification time (s) of each classifier, in minutes. For 20 features selected by RFE

4. Conclusion

According to the analysis, Random Forest (RF) and Gradient Boosting Machine (GMB) are the best algorithms for detecting intrusions. Always remember that this occurs for the data format of the NSL KDD dataset and the features selected by the RFE algorithm, but it may not be applicable in other situations. Other classifiers, especially DT, also obtained adequate values regarding balanced accuracy and precision. In addition, it should be remembered that classifiers such as SVM obtained auspicious results in other studies on intrusion detection. It is considered that many of the algorithms should continue to be taken into account in future similar studies, and could even be offered as options in future implementations of an IDS.

References

1. Khraisat A, Gondal I, Vamplew P, Kamruzzaman J. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur. 2019;2:20. https://doi.org/10.1186/s42400-019-0038-7
2. Khraisat A, Alazab A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecur. 2021;4:18. https://doi.org/10.1186/s42400-021-00077-7
3. Abbas A, Khan MA, Latif S, Latif S, Zaib A, Alnumay WS, *et al*. A new ensemble-based intrusion detection system for internet of things. Arab J Sci Eng. 2022;47:1805-1819. https://doi.org/10.1007/s13369-021-06086-5
4. Herrmann D, Pridöhl H. Basic concepts and models of cybersecurity. In: Christen M, Gordijn B, Loi M, editors. The ethics of cybersecurity. Cham: Springer; 2020. p. 19-36. https://doi.org/10.1007/978-3-030-29053-5_2
5. Shanker R, Agrawal P, Singh A, Bhatt M. Framework for identifying network attacks through packet inspection using machine learning. Nonlinear Eng. 2023;12. https://doi.org/10.1515/nleng-2022-0297
6. Wang H, Li Y. Overview of DDoS attack detection in software-defined networks. IEEE Access. 2024;12:38351-38381. https://doi.org/10.1109/ACCESS.2024.3375395
7. Asharf J, Moustafa N, Khurshid H, Debie E, Haider W, Wahab A. A review of intrusion detection systems using machine and deep learning in internet of things: challenges, solutions and future directions. Electronics. 2020;9(7):1177. https://doi.org/10.3390/electronics9071177
8. García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur. 2009;28:18-28. https://doi.org/10.1016/j.cose.2008.08.003

9. Lee K, Lee J, Yim K. Classification and analysis of malicious code detection techniques based on the APT attack. Appl Sci. 2023;13(5):2894. https://doi.org/10.3390/app13052894
10. Ibrahim HAH, Zuobi ORAL, Abaker AM, Alzghoul MB. A hybrid online classifier system for internet traffic based on statistical machine learning approach and flow port number. Appl Sci. 2021;11(24):12113. https://doi.org/10.3390/app112412113
11. Orozco-Arias S, Piña JS, Tabares-Soto R, Castillo-Ossa LF, Guyot R, Isaza G. Measuring performance metrics of machine learning algorithms for detecting and classifying Transposable Elements. Processes. 2020;8(6):638. https://doi.org/10.3390/pr8060638
12. Altulaihan E, Almaiah MA, Aljughaiman A. Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms. Sensors. 2024;24(2):713. https://doi.org/10.3390/s24020713
13. Talpur F, Korejo IA, Chandio AA, Ghulam A, Talpur MSH. ML-based detection of DDoS attacks using evolutionary algorithms optimization. Sensors. 2024;24(5):1672. https://doi.org/10.3390/s24051672
14. Khanan A, Abdelgadir Mohamed Y, Mohamed AHHM, Bashir M. From bytes to insights: a systematic literature review on unraveling IDS datasets for enhanced cybersecurity understanding. IEEE Access. 2024;12:59289-59317. https://doi.org/10.1109/ACCESS.2024.3392338
15. McHugh J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans Inf Syst Secur. 2000;3(4):262-294.
16. Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor. 2016;18(2):1153-1176. https://doi.org/10.1109/COMST.2015.2494502
17. Musa US, Chakraborty S, Abdullahi MM, Maini T. A review on intrusion detection system using machine learning techniques. In: 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS); 2021. p. 541-549. https://doi.org/10.1109/ICCCIS51004.2021.9397121
18. Mishra P, Varadharajan V, Tupakula U, Pilli E. A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun Surv Tutor. 2018;PP:1-1. https://doi.org/10.1109/COMST.2018.2847722
19. Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach. 2018;19:305-307. https://doi.org/10.1007/s10710-017-9314-z
20. Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification. Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei; 2003. p. 1-12.
21. Chen WH, Hsu SH, Shen HP. Application of SVM and ANN for intrusion detection. Comput Oper Res. 2005;32:2617-2634. https://doi.org/10.1016/j.cor.2004.03.019
22. Shalev-Shwartz S, Ben-David S. Understanding machine learning: from theory to algorithms. Cambridge: Cambridge University Press; 2014. https://doi.org/10.1017/CBO9781107298019