Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.039687
Article

Ensemble-Based Approach for Efficient Intrusion Detection in Network Traffic

Ammar Almomani[1,2,*], **Iman Akour**[3], **Ahmed M. Manasrah**[4,5], **Omar Almomani**[6], **Mohammad Alauthman**[7], **Esra'a Abdullah**[1], **Amaal Al Shwait**[1] **and Razan Al Sharaa**[1]

[1]School of Computing, Skyline University College, University City of Sharjah, P. O. Box 1797, Sharjah, United Arab Emirates
[2]IT-Department-Al-Huson University College, Al-Balqa Applied University, P. O. Box 50, Irbid, Jordan
[3]Information Systems Department, College of Computing & Informatics, University of Sharjah, United Arab Emirates
[4]Comp. Info Sciences (CIS) Division, Higher Colleges of Technology, Sharjah, United Arab Emirates
[5]Computer Sciences Department, Yarmouk University, Irbid, Jordan
[6]Computer Network and Information Systems Department, The World Islamic Sciences and Education University, Amman, 11947, Jordan
[7]Department of Information Security, Faculty of Information Technology, University of Petra, Amman, Jordan

*Corresponding Author: Ammar Almomani. Email: ammarnav6@bau.edu.jo

Received: 11 February 2023; Accepted: 18 May 2023; Published: 23 June 2023

Abstract: The exponential growth of Internet and network usage has necessitated heightened security measures to protect against data and network breaches. Intrusions, executed through network packets, pose a significant challenge for firewalls to detect and prevent due to the similarity between legitimate and intrusion traffic. The vast network traffic volume also complicates most network monitoring systems and algorithms. Several intrusion detection methods have been proposed, with machine learning techniques regarded as promising for dealing with these incidents. This study presents an Intrusion Detection System Based on Stacking Ensemble Learning base (Random Forest, Decision Tree, and k-Nearest-Neighbors). The proposed system employs pre-processing techniques to enhance classification efficiency and integrates seven machine learning algorithms. The stacking ensemble technique increases performance by incorporating three base models (Random Forest, Decision Tree, and k-Nearest-Neighbors) and a meta-model represented by the Logistic Regression algorithm. Evaluated using the UNSW-NB15 dataset, the proposed IDS gained an accuracy of 96.16% in the training phase and 97.95% in the testing phase, with precision of 97.78%, and 98.40% for taring and testing, respectively. The obtained results demonstrate improvements in other measurement criteria.

Keywords: Intrusion detection system (IDS); machine learning techniques; stacking ensemble; random forest; decision tree; k-nearest-neighbor

1 Introduction

Network security has become vital to its successful operation with the emerging Internet and communication technologies. Therefore, organizations are forced to invest in the security of their sensitive information and functions by adopting different security controls such as firewalls, anti-virus software, and Intrusion Detection Systems (IDS) to ensure the security of the network and all associated assets. The IDS is essential for identifying abnormal traffic and notifying the network administrator [1].

Machine learning techniques have garnered substantial popularity in network security over the last decade due to their ability to learn relevant features from network data and perform accurate categorization based on learned patterns [2]. Furthermore, due to its deep architecture, Deep Learning (DL)-based IDS rely on the automated learning of intricate characteristics from raw data [3].

Ensemble learning, on the other hand, is a machine learning paradigm in which several models, such as classifiers or experts, are developed and integrated strategies to address a specific computational intelligence issue. Ensemble learning is primarily used to improve a model's performance (Classification, prediction, function approximation, etc.) or to lessen the risk of an unintentional poor model selection. The architecture of the ensemble model is depicted in Fig. 1.

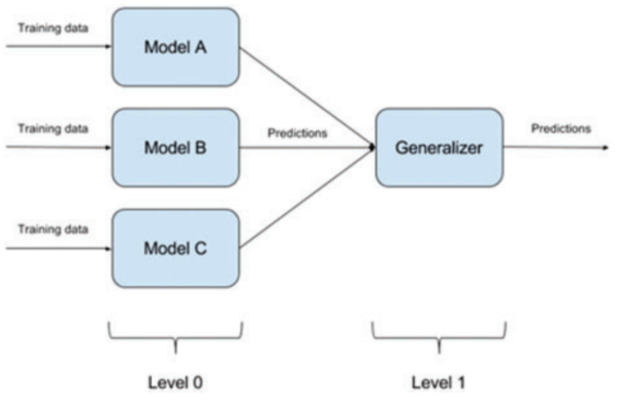

Figure 1: The basic architecture of ensemble classifiers

This paper proposes an IDS that utilizes a diversity of individual experts to analyze network traffic requests and responses. The system tracks the activity from when it enters the network to when it causes harm or performs unauthorized actions. The system achieves classifier diversity by employing different training parameters for each classifier, allowing individual classifiers to produce various decision boundaries. Combining these diverse errors leads to a lower overall error and higher accuracy than other A.I. and DL algorithms.

The research aims to develop an effective and efficient IDS (IDS) for detecting security breaches in complex network traffic. The objectives of this research are:

- To evaluate the performance of several machine learning algorithms for intrusion detection.
- To use stacking ensemble learning to improve the performance of the IDS.
- To evaluate the proposed IDS system using the UNSW-NB15 dataset and to compare its performance with other existing IDS.

The contributions of this research are:

- The proposed IDS system uses the stacking ensemble learning technique to improve the performance of intrusion detection.
- The system is evaluated using the UNSW-NB15 dataset and shows high accuracy and improved performance compared to other IDS.
- The study provides valuable insights into using machine learning algorithms and stacking ensemble learning for intrusion detection, which can contribute to advancing this field.

This paper is organized as follows: Section 2 presents the related work. Section 3 proposed an intrusion detection model to which different machine learning techniques are applied is described. While Section 4 provides the Implementation, and the results are discussed. Finally, Section 5 concludes the paper and presents future work.

2 Related Work

Jim Anderson first proposed the idea of IDS in 1980 [4]. Since then, many IDS products have been developed and matured to meet network security needs [5]. A combination of "intrusion" and "detection systems" is known as an IDS. The term "intrusion" refers to unlawful access to computer systems or a network's internal data to compromise its integrity, confidentiality, or availability [6]. In contrast, the detection system serves as a safeguard against illicit activities. Because of this, IDS is a security tool that constantly monitors host and network traffic to detect any suspicious behavior that violates the security policy and compromises the network's confidentiality, integrity, and availability. It is connected to a network adapter configured with port mirroring technology, as shown in Fig. 1.

The IDS will flag the host or network administrator's malicious behavior. Scientists have investigated machine learning and DL approaches to meet the needs of successful IDS. Aiming to acquire meaningful information from large datasets, both Machine Learning (ML) and DL) are being explored. Over the past decade, the development of extremely powerful Graphics Processing Units (GPUs) has led to the widespread adoption of these technologies in network security [7,8]. ML and DL are powerful methods for network traffic analysis and traffic prediction. To derive valuable information from network traffic, ML-based IDS significantly relies on engineering features [2]. In contrast, DL-based IDS relies on automatically learning complex features from raw data due to its deep architecture [3].

When designing an IDS, four functional modules,1-Event-boxes; 2-Database-boxes; 3-Analysis-boxes; and 4-Routine boxes, are used to build the overall architecture, as shown in Fig. 2. For the most part, Event-Boxes are sensors that monitor the system and gather data for subsequent analysis. This gathered data must be kept for processing. Data-base-box elements serve this purpose by storing the information received from the event-Boxes. The Analysis-boxes processing module is where harmful conduct is detected by examining events. The most critical step is to stop the hostile conduct once it has been identified. On the other hand, the Response-boxes take action immediately if any intrusion is detected. In Fig. 2, an example of the IDS framework is shown [1]. A Host-Based IDS (HIDS) and a Network-based IDS (NIDS) can be characterized by information source Event-Boxes (NIDS). System calls and process identifiers are the focus of HIDS, while network events are the focus of NIDS (I.P. address, protocols, service ports, traffic volume, etc.). IDS can be divided into signature-based IDS (misuse-based) and anomaly-based IDS based on the analysis done in Analysis-boxes. (Signature-based IDS) SIDS utilizes a database of known attack signatures to detect intrusions by comparing captured data against the database [9]. Only known assaults can be detected using this method; new threats cannot be detected using this method (previously unseen attacks). Signature-based IDS has a significantly reduced false-positive rate.

On the other hand, Anomaly-based IDS (AIDS) aims to understand the system's usual behavior and establishes a threshold. An anomaly alarm is sounded when a certain observation deviates from its regular pattern [10]. Anomaly-based IDS is useful for detecting previously unnoticed assaults since it seeks to find unusual occurrences. In contrast to SIDS, AIDS has a greater false-positive rate for intrusion detection [1].

To effectively detect network-based attacks, many researchers attempted to combine Stack ensemble learning, a collection of ML algorithms. For example, the stacked ensemble technique presented was assessed using.

The performance of the suggested method was compared to that of other well-known ML algorithms such as Artificial neural network ANN, CART, random forest, and Support vector machines SVM. The experimental results show that stacked ensemble learning is a proper technique for classifying network-based attacks. Similarly, the authors in [8] applied a group of learning algorithms over the UNSW-NB15 dataset using the stacking classifier method. The mixed method for feature selection also includes Lasso regression using SVM. The accuracy of the Lasso was evaluated using the R2 score evaluation, which was 59%. Similar to the research in [11], Gao et al. [12] analyze the NSL-KDD dataset by exploring different training ratios and creating multiple decision trees to conclude their MultiTree detection algorithm with adaptive voting in multiple classifier algorithms. Fig. 2 shows the IDS for network traffic monitoring, and Fig. 3 show IDS Architecture.

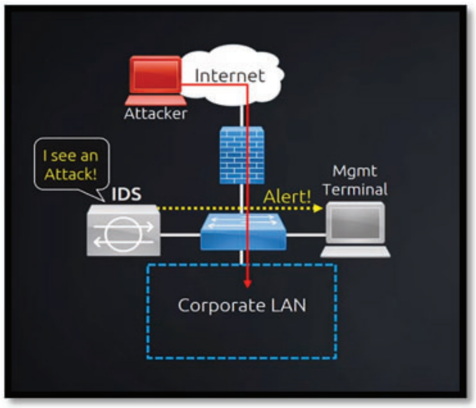

Figure 2: IDS(IDS)—monitors network traffic [13]

For this reason, the authors adopted different classifiers, including Decision trees, random forests, K-Nearest Neighbors KNN, DNN. The adaptive voting algorithm increases the detection accuracy from 84.1% to 85.2%. Therefore, Larriva-Novo et al. [14] suggested using a Dynamic classifier multiclass to obtain the best capabilities for each cyber-attack detection model. They tested their approach on a UNSW-NB15 dataset, in which the dataset was split into training data (75%) and tests (25%). The dynamic classifier improved results by 5.3% and 2.3% compared to the best static model for the balanced and unbalanced datasets. Their work is achieving a noticeable increase in performance in terms of detection rate for IDS based on multiple classes of attacks.

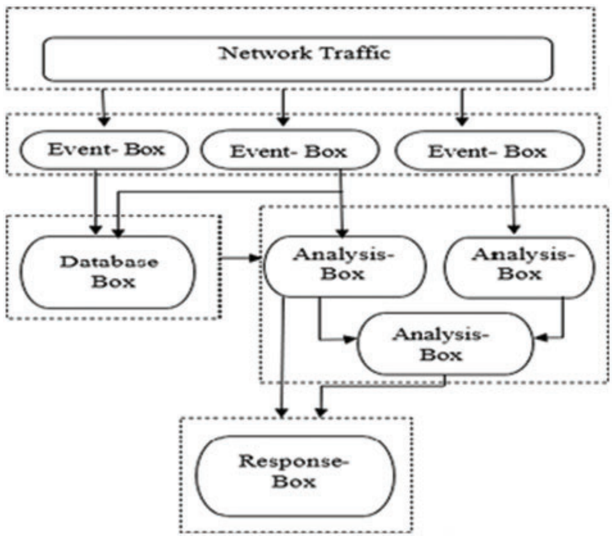

Figure 3: Common intrusion detection architecture for IDS [1]

Slay [15] proposed using a Hyper-Clique-based Improved Binary Gravitational Algorithm (HC-IBGSA) to identify the optimal model parameters and feature set for SVM. The proposed approach was evaluated using two benchmark intrusion datasets, the NSL-KDD CUP and the UNSW-NB15 dataset. The evaluation was carried out over two feature sets for SVM training: (1) SVM trained with all features and (2) SVM trained with the optimal features obtained from HC-IBGSA. The proposed approach shows a 94.11% classification accuracy using the UNSW NB15 dataset with features extracted using the HC-IBGSA algorithms. Similarly, Slay [15] presents a feature selection for rare cyber-attacks based on the UNSW-NB15 dataset using the Random Committee technique. The proposed approach evaluation of the multiclass Classification obtains an accuracy of 99.94%. However, the high accuracy rate was reported as the best case for a work attack.

Consequently, Correctly classifying network flows as benign or malicious traffic is the way the authors [16] have adopted. Their approach depends on classifying network traffic flows using R.F., MLP, and LSTM. The proposed approach evaluations using the CIDDS-001 dataset yield 99.94% accuracy. As a result, many researchers have started to adopt various classifiers into their intrusion detection techniques to accommodate the different types of input data. For instance, In [17] the authors attempted to unite the strengths of SIDS and an AIDS-based IDS into a new hybrid IDS system (HIDS). The new HIDS combines the C5 decision tree classifier and a single class support vector machine (OC-SVM). Using the Network Security Lab Knowledge Discovery in databases (NSL-KDD) and Australian Defense Force Academy (ADFA) datasets, the authors confirm that they achieve low alarm rates.

To summarize, proposing an intrusion detection model would entail using multiple datasets to demonstrate detection capability and extensibility in various environmental settings. Table 1 shows that the NSL-KDD, KDD Cup 99, and UNSW-NB15 datasets are widely used in IDS research.

Table 1: Summary of selected studies utilizing ensemble methods for IDSs

Reference	Algorithm	Accuracy	Classifier	Dataset	Adv.	Limitation
Jing et al. [18]	SVM	85.99% 75.77%	Binary Multiclass	UNSW-NB15	The proposed SVM method accuracy outperforms the Naive Bayes (N.B.) method with a score of 75.77%.	Compared to the UNSW-NB15 dataset, the KDDCUP99 dataset lacks some common examples for NIDS evaluation.
Ahmad et al. [19]	SVM RF ELM	99% 97.5% 99.5%	Binary	NSL–KDD	These techniques are well-known for their classifiers, and ELM outperforms other approaches (SVM, R.F.).	SVM outperforms other approaches on small datasets, whereas EML outperforms others on large datasets.
Rajadurai et al. [11]	Ensemble ANN RF Naïve Bayes SVM	91.06% 97.74% 74.00% 74.40% 74.00%	Binary	NSL-KDD	Staked ensemble learning is suitable for classifying attacks, and the proposed system outperforms other intrusion detection models in terms of accuracy.	The precision and recall values of the RNN and ANN approaches are significantly lower than those of the proposed approach.
Abirami et al. [8]	RF SVM Naive Bayes Logistic regression	93% 85% 79% 80%	Binary	KDD Cup99 NSK-KDD Kyoto2006	The clustering classifier and ensemble algorithm yielded better results.	The decision tree classification algorithm and the regression algorithms achieved less precision.
Gao et al. [12]	DeciTree RF LR KNN DNN Adaboost SVM	73.58% 79.71% 74.09% 76.02% 81.6%	Binary	NSL-KDD	The ensemble model significantly improves detection accuracy.	The deep neural network has some advantages in detection, but it takes time.
Ring et al. [20]	NBTree Fuzzy SVM FS+GAR-forest TDTC FSSL EM-FS FSSL-EL TSE-IDS NBTree FSSL FSSL-EL TSE-IDS	82.02% 82.74% 82.37% 85.05% 84.86% 84.25% 84.54% 85.79% 66.16% 68.82% 71.29% 72.52% 87.37% 73.57%	Binary	NSL-KDD UNSW-NB15	Under several criteria, the proposed CFS-BA-Ensemble method outperforms other relevant techniques.	Difficult to identify high attack detection rates (ADR) while minimizing false alarm rates (FAR).

(Continued)

Table 1: Continued

Reference	Algorithm	Accuracy	Classifier	Dataset	Adv.	Limitation
Bamakan et al. [21]	XGBoost KNN Logistic Regression Stacking	83.54% 84.00% 63.54% 85.42%	Binary Multiclass	MLPAT ELM	APT attacks are conducted with high planning levels and a high degree of target recognition.	The cost of implementing APT is too high.
Larriva-Novo et al. [14]	Dynamic Classifier	87.6%	Binary Multiclass	UNSW-NB15	The dynamic classifier model improves the detection accuracy of each model.	TPR can be reduced by up to 40% using category exploit.
Pang et al. [22]	CFS-BA	99.81%	Binary Multiclass	NSL-KDD AWID ClC-IDS2017	The proposed CFS-BA-Ensemble method outperforms other approaches on different metrics.	Efficient while keeping FAR under control
Sindhu et al. [23]	XGBoost KNN Logistic Regression Stacking	83% 84% 52% 85%	Binary	KDD'99	The stack classifier achieved the best result compared to otherclassifiers.	Logisticregression is the worst inaccuracy.
Lee et al. [24]	DNN SHAP BRCG CEM	90.82% 87.96% 82.71% 92.82%	Binary	NSL-KDD ClC-IDS2017 KDDT	It gives a much better insight to the security analyst on why the alert was flagged.	The model may learn that demand leads to poverty performance.
Raman et al. [25]	C4.5 Naïve Bayes RF Multilayer Perception SVM CART KNN	81% 76.56% 80.67% 77.41% 69.52% 80.3% 79.4%	Binary Multiclass	NSL-KDD CIDDS-001	Possibility of gaining access to a high level of electronic resilience against malicious activity and unauthorized identification.	These technologies may be incapable of generating and updating a new malware signal due to high alarms or low detection rates.
Raman et al. [26]	HC-IBGSA	94.11%	Binary Multiclass	NSL-KDD, UNSW-NB15	Using more recent IDS datasets to assess algorithm performance before and after feature selection	Not using different classifiers, only SVM
Slay [15]	Feedforward NN	99.94%	Binary Multiclass	UNSW-NB15	The DL model achieves very high accuracy.	Traditional MLalgorithms are inefficient at classifying Network Intrusions.
Khraisat et al. [16]	RF MLP LSTM	99.94%	Attack type	CIDDS-001	The multi-flow method is appropriate for detecting anomalies in the CIDDS-001 dataset.	As the length of the sequence increases, the radiofrequency decreases dramatically.

(Continued)

Table 1: Continued

Reference	Algorithm	Accuracy	Classifier	Dataset	Adv.	Limitation
Rashid et al. [27]	k-NN Naïve Bayes SVM NN DNN Auto-encoders	99.80% 98.60% 100% 99.90% 99.90% 98.60%	Binary	NSL-KDD CIDDS-001	SVM, DNN, and k-NN classifiers all perform similarly.	Since the multiclass Classification was not addressed, the types of attacks in the CIDDS-001 dataset cannot be discovered.
Rababah et al. [17]	C4.5 Naïve Bayes SVM CART KNN	81% 76.56% 80.67% 69.52% 80.3%	Binary	NSL-KDD	Compared to SIDS and AIDS, HIDS has a higher detection rate and lower alarm rate.	Single algorithms give in accurate results.
Yang et al. [28]	RBFN Naïve Bayes DT RI K-NN	92.17% 91.23% 91.38% 91.81% 91.24%	Binary Multiclass	CIDDS-001	Correlation rules and group analysis are used to detect illegal activities in database usage patterns.	The emphasis was on reducing false alarms rather than increasing the detection rate.
Gautam et al. [29]	KNN SVM DT RF ET XGBoost Stacking FSXGBoost FS Stacking	96.6% 98.01% 99.72% 98.37% 93.43% 99.78% 99.86% 99.7% 99.82%	Binary	NSL-KDD CIDDS-001	The proposed system has a high detection rate and a low computing cost.	Require high computation and storage requirements.

3 Data Acquisition (Collection, Information Gathering)

To test and evaluate the proposed approach, we have used the UNSW-NB15 dataset from ACCS [15], a modern NIDS benchmark data set for Network IDS. The dataset has 2.5 million records that are divided into 45 features. We have modified the original UNSW-NB15 dataset to reduce the number of features to 43 instead of 45, including flow-based and packet-based features. These features are further subdivided into four categories: content, fundamental, flow, and time-based features. The number of selected data instances from the UNSW-NB15 dataset is 257,673, divided into training data instances (175,341 records) and testing data instances (82,332 records). The record distribution and class distribution of the UNSW-NB15 dataset are shown in Fig. 4 [30].

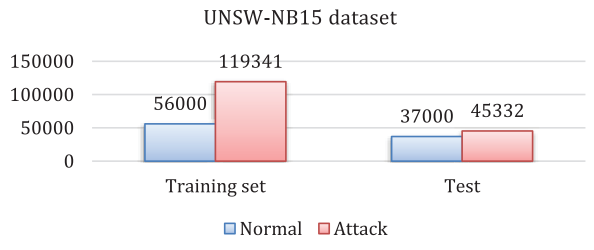

Figure 4: UNSW-NB15 dataset distribution

The UNSW-NB15 dataset provides two labeled features (i.e., cat-attack and label). Label characteristics were only utilized when the data was either normal or attacked (binary data). The features list and their names are given in Table 2.

Table 2: Features of the UNSW-NB15 [15]

No.	Name	T	group	Description
1	dur	F	Basic	Total record length
2	Proto	N	Flow	Transaction protocol
3	service	N	Basic	HTTP, FTP, SSH, DNS .., else (-)
4	state	N	Basic	The state and its dependent protocol, e.g., ACC, CLO, else (-)
5	spkts	I	Basic	Count of packets sent from the source to the destination
6	dpkts	I	Basic	Count of packets from source to destination
7	sbytes	I	Basic	Bytes from the source to the destination
8	dbytes	I	Basic	Source bytes to destination bytes
9	Rate		Basic	This term is in the training/test sets but not defined anywhere.
10	sttl	I	Basic	Source to live time of destination.
11	dttl	I	Basic	Time to live in the destination.
12	sload	F	Basic	Bits of source per second.
13	dload	F	Basic	Bits per second destination.
14	sloss	I	Basic	Retransmitted or dropped source packets.
15	dloss	I	Basic	Retransmitted or dropped destination packets.
16	sintpkt	F	Time	Time of arrival for the interpacket source (mSec).
17	dintpkt	F	Time	Time of arrival between packets (mSec).
18	sjit	F	Time	Jitter Source (mSec).
19	djit	F	Time	Jitter Destination (mSec).
20	swin	I	Content	Advertising Content Source TCP window.
21	dwin	I	Content	Advertisement in the TCP window for the destination
22	stcpb	I	Content	TCP sequence number from the source
23	dtcpb	I	Content	TCP sequence number for the destination
24	tcprtt	F	Time	The TCP's 'synack' and 'ackdat' are added together.
25	synack	F	Time	The interval between the TCP packets SYN and SYN ACK.
26	ackdat	F	Time	The interval between the TCP's SYN ACK and ACK packets.
27	smeansz	I	Content	Means the flow packet's src size.
28	dmeansz	I	Content	The average size of the dst sent flow packets.
29	trans_depth	I	Content	The connection's depth in the http request/response transaction.
30	res_bdy_len	I	Content	The magnitude of the data transmitted by the server's http service.

(Continued)

Table 2: Continued

No.	Name	T	group	Description
31	ct_srv_src	I	Connection	No, this is the 100th time in 100 connections with the same service and source address.
32	ct_state_ttl	I	General	No, for each state, based on a defined source/destination value range. to live.
33	ct_dst_ltm	I	Connection	No. connections of the same destination address Last time in 100 connections.
34	ct_src_dport_ltm	I	Connection	No Last time there were no connections with the same source address and destination port at 100 connections.
35	ct_dst_sport_ltm	I	Connection	No of the Last time in 100 connections no connections of the same destination address and source port.
36	ct_dst_src_ltm	I	Connection	No of the Last time there were 100 connections with the same source and destination address.
37	is_ftp_login	B	General	If a user and password are used for the ftp session, then another 1 is 0.
38	ct_ftp_cmd	I	General	There are no flows with ftp command.
39	ct_flw_http_mthd	I	General	No. flows with methods like Get and Post in HTTP.
40	ct_src_ltm	I	Connection	The number of connections with the same source address in the last time 100 connections.
41	ct_srv_dst	I	Connection	No. of the Last time connections of the same source address in 100 connections.
42	is_sm_ips_ports	B	General	This variable will have a value of 1 other 0 if the source is identical to destination I.P. and port numbers.
43	attack_cat	N		Every type of assault is given a name. This data collection has nine categories (e.g., Fussers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms).
44	Label	B		Normal records have a value of 0 while attack records have a value of 1.

Note: Type (T.), N: nominal, I: integer, F: float, T: timestamp, and B: binary.

The table above shows the different types of available features. These are called nominal features and are grouped into proto, service, state, and attack_cat features. The proto feature has 133 values related to the TCP and UDP protocols. The service feature has 13 values related to the different network services, such as DNS, HTTP, SMTP, FTP data, FTP, SSH, POP3, DHCP, SNMP, SSL, IRC, and radius, as illustrated in Fig. 5. The state feature has 9 values related to different transport layer protocol flags such as INT, FIN, CON, REQ, RST, ECO, PAR, no, and URN, as illustrated in Fig. 6. The other category has a "-" followed by a long DNS name. For the conversion of nominal features into a numerical format, we employed one-hot encoding. This method represents each category as a binary vector, which is compatible with our feature selection process and the machine learning algorithms used in this study.

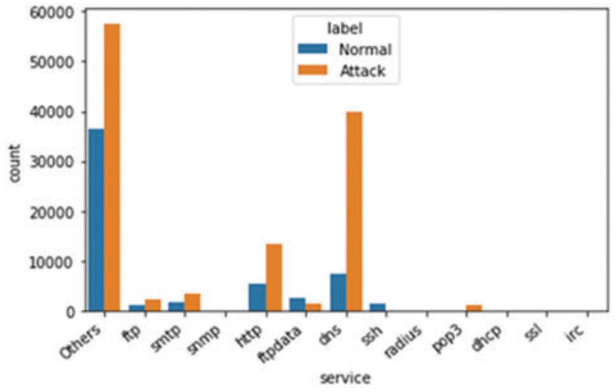

Figure 5: The distribution of the service feature

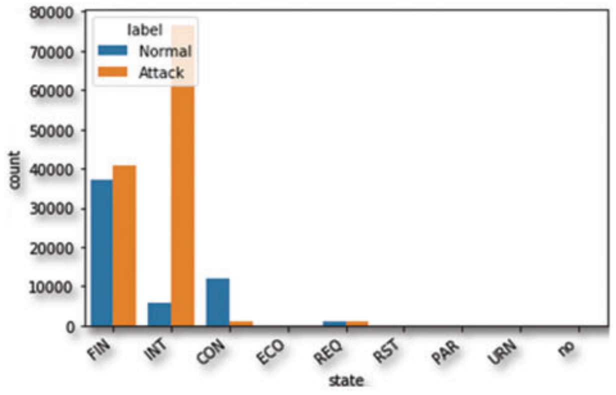

Figure 6: The distribution of the state feature

4 Proposed Ensemble Learning System Description

Ensemble learning is a widely used machine learning technique that combines the capabilities of individual classifiers to create a classification model with superior overall Classification or predictive power. In the context of intrusion detection systems (IDS), ensemble learning has demonstrated better performance compared to standalone classifiers. This study aims to develop a robust network intrusion detection system using an ensemble classifier approach. The proposed solution requires pre-processing of datasets for training purposes. In this work, we primarily focus on the UNSW-NB15 dataset, which comprises 45 features, including four nominal and 41 numerical features, after converting them to numerical values. Data normalization is essential, as the dimensions and units used in data collection vary, necessitating the scaling of different feature values within a specific range. In this study, we employ the maximum and minimum algorithms to normalize the data. The MinMaxScaler algorithm scales and translates each feature individually to lie between a given minimum and maximum value (typically between zero and one), as described in Eq. (1).

$$y = \frac{x - min}{\max - min} \tag{1}$$

The primary objective of this work is to detect network intrusions effectively. The proposed approach is structured into two main layers: the Base-learner layer and the Combining-Module layer, as depicted in Fig. 6. The Base-learner layer consists of multiple classifiers that learn from the data to identify intrusion patterns. In contrast, the Combining-Module layer serves as an aggregator that combines the output from the individual base classifiers, thereby enhancing the overall predictive performance. This layered architecture helps improve intrusion detection accuracy by leveraging the strengths of different classifiers and ensuring a more robust and reliable outcome.

In the first layer, we have selected three base binary classifiers to create a base module for the proposed system: (1) the random forest classifier (R.F.) [31], a method of combining multiple classifiers to tackle complex problems by using several decision trees on different subsets of a dataset and averaging the results to increase prediction accuracy; (2) the decision tree classifier (D.T.) [32], which identifies valuable information from large amounts of random data through predicting values, requiring a training dataset to create a tree and a test dataset to assess the decision tree's accuracy; and finally, (3) the K-NN classifier (K-Nearest Neighbors), a data classification method that calculates the probability of a data point being part of the nearest group, as depicted in the literature.

The ensemble output serves as input to the meta-classifier through the stack generalization process, ultimately yielding the final decision based on logistic regression to predict the probability of a specific class or event (e.g., pass/fail, win/lose). Ensemble learning can be categorized into three main types: bagging, boosting, and stacking. Bagging is the most prevalent method for predicting test outcomes. During the boosting training phase, models are rigorously trained on misclassified data. Stacking, or stacked generalization, is a highly-regarded ensemble technique for enhancing classification performance by combining multiple classifiers, as shown in Fig. 7.

In contrast to bagging and boosting, stacking comprises two levels: the base learner (level 0) and the meta-learner (level 1) [33]. At the first level, heterogeneous classification models learn from the training data, and their outputs generate a new dataset for the stacking learner. Each new instance in the dataset is assigned to the correct one. Class to be predicted (with a level 0 prediction as an attribute). The meta-learner then utilizes the newly formed dataset to produce the outcome [34], as illustrated in Fig. 8.

5 Experimental Results and Analysis

This section presents and analyses the suggested IDS findings based on ML methodologies and approaches. After applying seven algorithms to the UNSW-NB15 dataset, we used numerous assessment criteria to determine the efficacy of the proposed IDS system. The following assessment and performance metrics are used to evaluate the proposed IDS' performance. The anticipated outcomes are between 0 and 1. Accuracy, Precision, Recall, AUC, F1-Measure, and Mean squared error (MSE) [35] are used the evaluate the proposed approach.

- Accuracy is the number of correct predictions per classifier.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2}$$

- Precision: is a measure of genuine positive results derived from all positive findings in the dataset during Classification.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

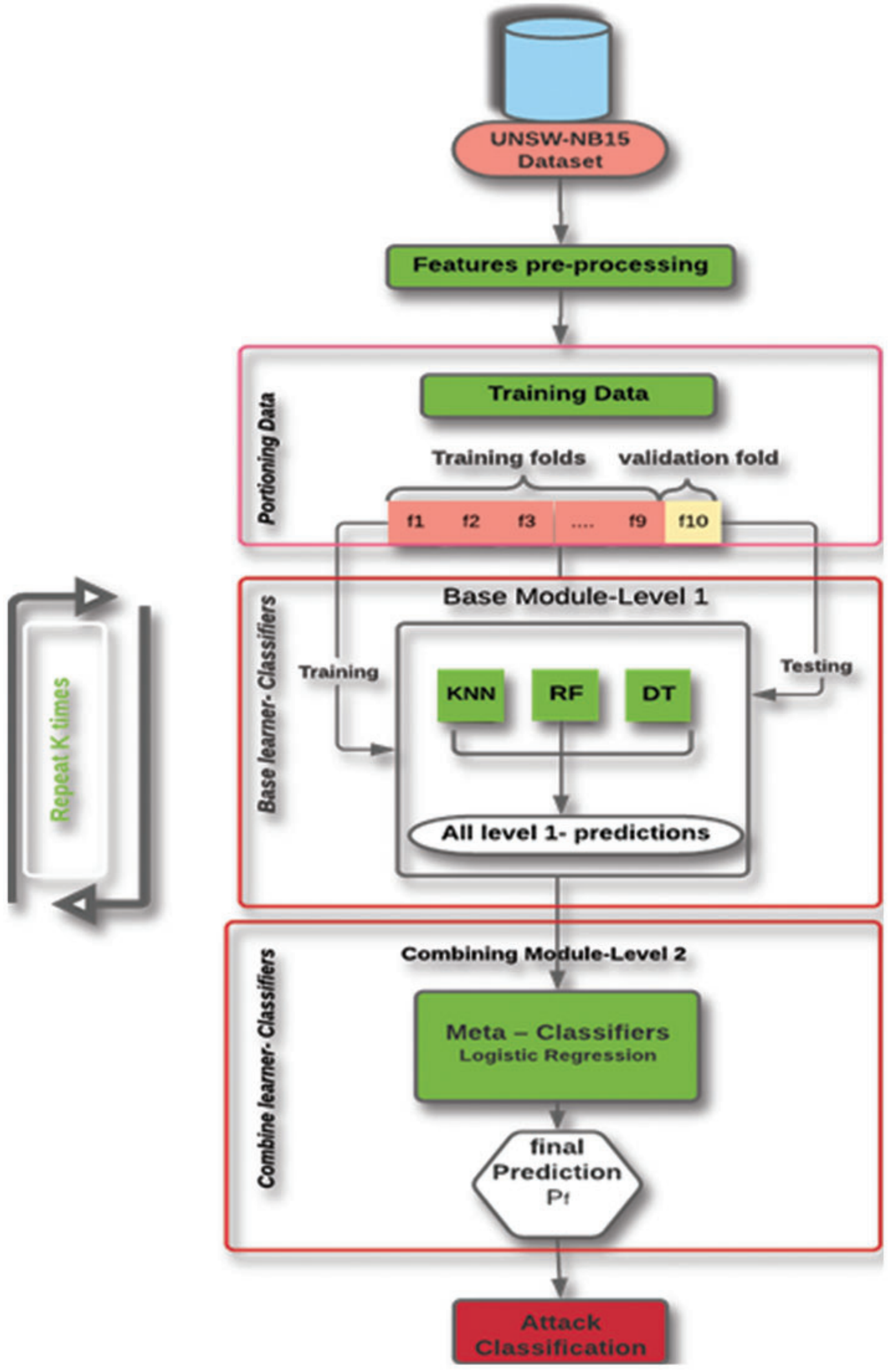

Figure 7: IDS based on stacking (schematic overview)

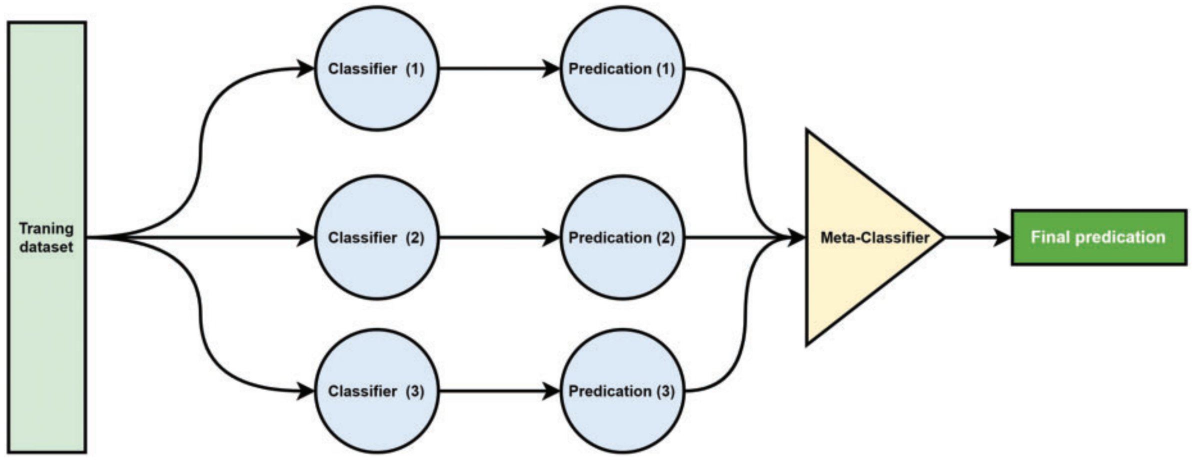

Figure 8: Stacking Classifiers [36]

- Recall: is the measurement of data points predicted to be positive by all the classifiers.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

- AUC—ROC curve: is a performance metric for classification tasks at various threshold levels. It is defined as the chance that the model rates a random positive sample higher than a random negative sample.

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

- F1-Measure: It is the harmonic relationship between Precision and Recall. F1 score is regarded as a superior performance statistic to accuracy. The greater F1-Score rate indicates that the MLmodel is doing better.

$$F1 - Measure = 2 \times \frac{Precision \times Rrcall}{Precision + Rrcall} \tag{7}$$

- Mean squared error (MSE): represents the expected value of the squared error loss. It is never negative; therefore, numbers close to zero are preferred.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\text{Yi} - \text{Yi}^{\wedge}\right)^2 \tag{8}$$

Collaboratory (Colab) was used for all evaluations, a cloud service based on Jupyter Notebooks for distributing ML teaching and research [37]. The models, pre-processing processes, and measurements were all implemented using the Python programming language in the same Colab environment. The system was running at high speed, and it could be classified the full data quickly so that it could be implemented as a real-world system in the future. We can improve the level of speed if we use collab pro [38] The fastest GPUs are reserved for Colab Pro and Pro+ customers.

Seven classifiers were examined to assess the performance of the proposed method: Naïve Bayes (N.B.), linear Support Vector Machine (SVM), SVM with Radial Basis Function (RBF) kernel, k-Nearest Neighbors (KNN), Logistic Regression (L.R.), Decision Tree (D.T.), and Random Forest (R.F.). The proposed method's accuracy was evaluated on the UNSW-NB15 dataset with a full feature space (42 features) for binary Classification. Stratified 10-fold cross-validation was employed to train all models on the training dataset. The 10-fold cross-validation process divides the dataset into ten subsets, with nine used for training the classifiers and one for testing.

Table 3 presents the accuracy, precision, recall, F1-Score, AUC, and MSE results of the proposed ML algorithms on the training dataset. R.F. achieves the highest accuracy of 96.12%. Linear SVM exhibits the highest precision score of 99.79% but the lowest recall score of 91.21%. While NB has a recall score of 93.41%, its accuracy is considerably lower. Conversely, D.T. demonstrates the highest recall value of 96.38% and minimal variance in its accuracy and recall values, resulting in a higher F1 score than the N.B. model. The R.F. algorithm yields the highest F1 score of 97.18% and the lowest MSE error of 0.0388. To enhance performance, we employed the stacking ensemble method, which comprises two or more base models (level-0 models) and a meta-model that aggregates the base models' predictions (level-1 model). The base models include R.F., D.T., and KNN, with the L.R. algorithm as the meta-model. The stacking ensemble achieves 96.16% accuracy.

Table 3: Performance comparison of ML classifiers on the training dataset

Method	Accuracy	Precision	Recall	F1-score	AUC	MSE
SVM (RBF)	93.60	99.63	91.69	95.49	95.36	0.0640
Random forest RF	96.12	97.98	96.38	97.18	95.96	0.0388
Logistic regression LR	93.40	99.13	91.83	95.34	94.79	0.0660
Linear SVM	93.31	99.79	91.21	95.31	95.33	0.0669
Naïve Bayes NB	81.38	78.16	93.41	85.10	79.44	0.1862
Decision tree DT	95.00	96.27	96.38	96.33	94.23	0.0500
KNN	93.76	95.90	94.98	95.44	93.02	0.0624
Stacking ensemble	96.16	97.78	96.62	97.20	95.88	0.0384

The test dataset results for accuracy, precision, recall, F1-Score, AUC, and MSE of the proposed ML algorithms are displayed in Table 4. R.F. obtains the highest values for accuracy, precision, recall, F1-score, and AUC, with 97.94%, 98.52%, 97.72%, 98.12%, and 97.96%, respectively. D.T. follows with values of 96.74% for accuracy, 96.98% for precision, 97.11% for recall, 97.04% for F1-score, and 96.70% for AUC. KNN ranks third with values of 93.33% for accuracy, 95.56% for precision, 92.17% for recall, 93.84% for F1-score, and 93.46% for AUC. R.F. achieves the best MSE error of 0.0206, followed by D.T. at 0.0326 and KNN at 0.0667. The stacking ensemble attains 97.95% accuracy.

More details can be shown clearly in Fig. 9, which discusses the Performance comparison of ML classifiers on the test dataset.

Table 4: Performance comparison of ML classifiers on the test dataset

Method	Accuracy	Precision	Recall	F1-score	AUC	MSE
SVM (RBF)	93.10	92.65	95.01	93.81	92.89	0.0690
Random forest RF	97.94	98.52	97.72	98.12	97.96	0.0206
Logistic regression LR	90.70	91.54	91.58	91.56	90.61	0.0930
Linear SVM	91.04	92.70	90.89	91.78	91.06	0.0896
Naïve Bayes NB	76.43	86.28	68.02	76.07	77.38	0.2357
Decision tree DT	96.74	96.98	97.11	97.04	96.70	0.0326
KNN	93.33	95.56	92.17	93.84	93.46	0.0667
Stacking ensemble	97.95	98.40	97.87	98.13	97.96	0.0205

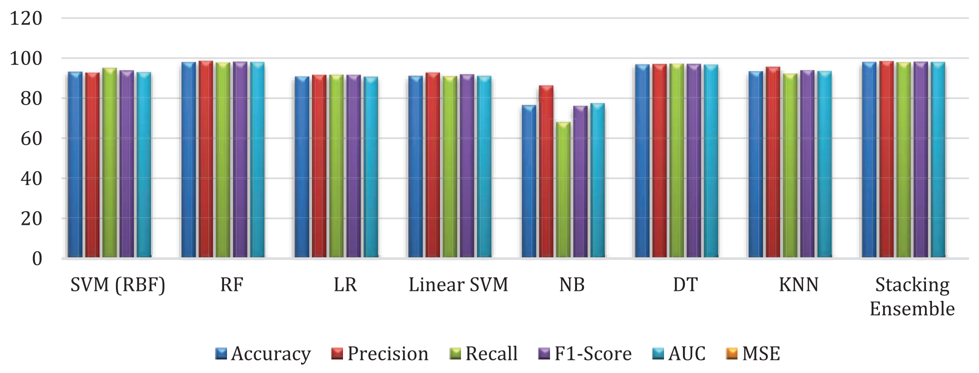

Figure 9: Performance comparison of ML classifiers on the test dataset

In conclusion, this study addresses the challenge of identifying the most suitable classifier for a specific classification task, such as intrusion detection in network data. The research compares the performance of various classifiers, including Multilayer Perceptron (MLP), SVM, Decision Trees, and Naïve Bayes. The findings reveal that employing an ensemble approach, which combines multiple classifiers, mitigates the risk of suboptimal selection compared to relying on a single classifier. The study applies the stacking ensemble technique with base models Random Forest (R.F.), Decision Tree (D.T.), k-Nearest Neighbors (KNN), and the meta-model.

6 Conclusion and Future Work

In conclusion, this study has addressed the issue of determining the best classifier for a specific classification task, such as intrusion detection in network data. The study has compared the performance of several classifiers, including Multilayer Perceptron (MLP), SVM (SVM), Decision Trees, and Naive Bayes. The results indicate that an ensemble approach, combining multiple classifiers, reduces the risk of making a poor selection compared to relying on a single classifier. The study has applied the stacking ensemble technique using the base models Random Forest (R.F.), Decision Tree (D.T.), k-Nearest Neighbors (KNN), and the meta-model Logistic Regression (L.R.), and achieved an accuracy of 97.95% in the testing phase. For future research, it is recommended to explore the applicability of this technology in real-world production systems. Additionally, further studies could investigate the combination of different pre-processing techniques and various ensembles to improve the overall performance of IDS.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare they have no conflicts of interest to report regarding the present study.

References

[1] R. A. Disha and S. Waheed, "Performance analysis of machine learning models for intrusion detection system using gini impurity-based weighted random forest (GIWRF) feature selection technique," *Cybersecurity*, vol. 5, no. 1, pp. 1–22, 2022. https://doi.org/10.1186/s42400-021-00103-8

[2] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald *et al.,* "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015. https://doi.org/10.1186/s40537-014-0007-7

[3] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *8th IEEE Int. Conf. on Communication Software and Networks (ICCSN)*, Beijing, China, pp. 581–585, 2016.

[4] J. P. Anderson, "Computer security threat monitoring and surveillance," Technical Report, James P. Anderson Company, 1980.

[5] H. Debar, M. Dacier and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999. https://doi.org/10.1016/S1389-1286(98)00017-6

[6] S. Mukkamala, G. Janoski and A. Sung, "Intrusion detection using neural networks and support vector machines," in *2002 Int. Joint Conf. on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, Honolulu, HI, USA, vol. 15, pp. 1702–1707, 2002.

[7] J. Lew, D. A. Shah, S. Pati, S. Cattell, M. Zhang *et al.,* "Analyzing machine learning workloads using a detailed GPU simulator," in *IEEE Int. Symp. on Performance Analysis of Systems and Software (ISPASS)*, Madison, WI, USA, pp. 151–152, 2019.

[8] M. Abirami, U. Yash and S. Singh, "Building an ensemble learning based algorithm for improving intrusion detection system," In: M. Abirami (Ed.), *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pp. 635–649, Singapore: Springer, 2020.

[9] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019. https://doi.org/10.1186/s42400-019-0038-7

[10] J. H. Liao, C. H. R. Lin, Y. C. Lin and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013. https://doi.org/10.1016/j.jnca.2012.09.004

[11] H. Rajadurai and U. D. Gandhi, "A stacked ensemble learning model for intrusion detection in wireless network," *Neural Computing and Applications*, pp. 1–9, 2020.

[12] X. Gao, C. Shan, C. Hu, Z. Niu and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019. https://doi.org/10.1109/ACCESS.2019.2923640

[13] B. Lutkevich, *What is an intrusion detection system (IDS)?* USA: Techtarget, 2020. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/intrusion-detection-system

[14] X. Larriva-Novo, C. Sánchez-Zas, V. A. Villagrá, M. Vega-Barbas and D. Rivera, "An approach for the application of a dynamic multi-class classifier for network intrusion detection systems," *Electronics*, vol. 9, no. 11, pp. 1759, 2020. https://doi.org/10.3390/electronics9111759

[15] M. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *IEEE Military Communications and Information Systems Conf. (MilCIS)*, Canberra, ACT, Australia, pp. 1–6, 2015.

[16] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman and A. Alazab, “Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine,” *Electronics*, vol. 9, no. 1, pp. 173, 2020. https://doi.org/10.3390/electronics9010173
[17] B. Rababah and S. Srivastava, “Hybrid model for intrusion detection systems,” *ArXiv Preprint arXiv:2003.08585*, 2020.
[18] D. Jing and H. -B. Chen, “SVM based network intrusion detection for the UNSW-NB15 dataset,” in *IEEE 13th Int. Conf. on ASIC (ASICON)*, Chongqing, China, pp. 1–4, 2019.
[19] I. Ahmad, M. Basheri, M. J. Iqbal and A. Rahim, “Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection,” *IEEE Access*, vol. 6, pp. 33789–33795, 2018. https://doi.org/10.1109/ACCESS.2018.2841987
[20] M. Ring, S. Wunderlich, D. Scheuring, D. Landes and A. Hotho, “A survey of network-based intrusion detection data sets,” *Computers & Security*, vol. 86, no. 1, pp. 147–167, 2019. https://doi.org/10.1016/j.cose.2019.06.005
[21] S. M. H. Bamakan, H. Wang, T. Yingjie and Y. Shi, “An effective intrusion detection framework based on mclp/svm optimized by time-varying chaos particle swarm optimization,” *Neurocomputing*, vol. 199, no. 10, pp. 90–102, 2016. https://doi.org/10.1016/j.neucom.2016.03.031
[22] Z. -H. Pang, G. -P. Liu, D. Zhou, F. Hou and D. Sun, “Two-channel false data injection attacks against output tracking control of networked systems,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3242–3251, 2016. https://doi.org/10.1109/TIE.2016.2535119
[23] S. S. S. Sindhu, S. Geetha and A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach,” *Expert Systems with Applications*, vol. 39, no. 1, pp. 129–141, 2012. https://doi.org/10.1016/j.eswa.2011.06.013
[24] W. Lee, S. J. Stolfo and K. W. Mok, “A data mining framework for building intrusion detection models,” in *IEEE Symp. on Security and Privacy (Cat. No. 99CB36344)*, Oakland, CA, USA, pp. 120–132, 1999.
[25] M. G. Raman, N. Somu, K. Kirthivasan, R. Liscano and V. S. Sriram, “An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine,” *Knowledge-Based Systems*, vol. 134, no. 5, pp. 1–12, 2017. https://doi.org/10.1016/j.knosys.2017.07.005
[26] M. G. Raman, N. Somu, S. Jagarapu, T. Manghnani, T. Selvam *et al.,* “An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm,” *Artificial Intelligence Review*, vol. 53, no. 5, pp. 1–32, 2019.
[27] A. Rashid, M. J. Siddique and S. M. Ahmed, “Machine and deep learning based comparative analysis using hybrid approaches for intrusion detection system,” in *3rd Int. Conf. on Advancements in Computational Sciences (ICACS)*, Lahore, Pakistan, pp. 1–9, 2020.
[28] H. Yang, G. Qin and L. Ye, “Combined wireless network intrusion detection model based on deep learning,” *IEEE Access*, vol. 7, pp. 82624–82632, 2019. https://doi.org/10.1109/ACCESS.2019.2923814
[29] R. K. S. Gautam and E. A. Doegar, “An ensemble approach for intrusion detection system using machine learning algorithms,” in *2018 8th Int. Conf. on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, pp. 14–15, 2018.
[30] Z. Zoghi and G. Serpen, “UNSW-NB15 computer security dataset: Analysis through visualization,” *arXiv preprint arXiv:2101.05067*, 2021.
[31] S. S. Nikam, “A comparative study of classification techniques in data mining algorithms,” *Oriental Journal of Computer Science and Technology*, vol. 8, no. 1, pp. 13–19, 2015.
[32] S. Kaur and H. Kaur, “Review of decision tree data mining algorithms: Cart and c4. 5,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 4, pp. 436–439, 2017.
[33] I. Syarif, E. Zaluska, A. Prugel-Bennett and G. Wills, “Application of bagging, boosting and stacking to intrusion detection,” in *Int. Workshop on Machine Learning and Data Mining in Pattern Recognition*, Berlin, Germany, pp. 593–602, 2012.

[34] M. Graczyk, T. Lasota, B. Trawiński and K. Trawiński, “Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal,” in *Asian Conf. on Intelligent Information and Database Systems*, Hue, Vietnam, pp. 340–350, 2010.

[35] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning (No. 4)*. New York, USA: Springer, 2006.

[36] F. Ceballos, *Stacking classifiers for higher predictive performance*. USA: Medium, 2022. [Online]. Available: https://towardsdatascience.com/stacking-classifiers-for-higher-predictive-performance-566f963e4840

[37] J. Introne, R. Laubacher, G. Olson and T. Malone, “The climate colab: Large scale model-based collaborative planning,” in *Int. Conf. on Collaboration Technologies and Systems (CTS)*, Philadelphia, PA, USA, pp. 40–47, 2011.

[38] E. Bisong, “Google colaboratory,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Berkeley, CA: Apress, pp. 59–64, 2019.