Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

Multi-Class Network Anomaly Detection Using Machine Learning Techniques

Satyanarayana Gunupusala, Shahu Chatrapathi Kaila*

Department of Computer Science and Engineering, Jawaharlal Nehru Technological University Hyderabad, Kukatpally, Hyderabad-500085, Telangana, India
E-mail: shahujntu@gmail.com

Received: 28 September 2023; **Revised:** 4 November 2023; **Accepted:** 6 November 2023

Abstract: Computer networks rely on Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs) to ensure the security, reliability, and availability of an organization. In recent years, various approaches were developed and implemented to create effective IDSs and IPSs. This paper specifically focuses on IDSs that utilize Machine Learning (ML) techniques for improved accuracy. ML-based IDSs have verified to be successful in discovering network attacks. However, their performance tends to decline when dealing with high-dimensional data spaces. It is essential to develop a suitable feature extraction strategy that could identify and remove irrelevant features that do not significantly classification process to address this issue. Additionally, many ML-based IDSs exhibit high false positive rates and poor detection accuracy when trained on unbalanced datasets. In this study, we analyze the UNSW-NB15 IDS, which will serve as the training and testing data for our models. In order to reduce the feature space and improve the efficiency of our analysis, we leverage a filter-based feature reduction method utilizing the Pearson correlation coefficient algorithm. By identifying and selecting only the most relevant features, we are able to streamline our dataset and focus on the variables that have the highest impact on our analysis. This approach not only reduces computational complexity but also improves the interpretability of our results by eliminating unnecessary noise from the data. After applying the feature reduction technique, we proceed to implement a range of machine learning methods to perform our classification task. These include well-known algorithms such as Stacking, Extra Trees, Multi-Layer Perceptron, XGBoost, K-Nearest Neighbors, Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest, and Decision Tree. By employing a diverse set of algorithms, we are able to explore different modeling approaches and evaluate their effectiveness in accurately classifying the various types of assaults. In order to assess the performance of our classification models, we utilize a range of specialized evaluation metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), R2-Score, Mean Squared Error (MSE), Precision, F1-Score, Recall, and Accuracy. These metrics provide us with a comprehensive understanding of how well our models are performing across different dimensions, including the accuracy of predictions, the level of precision in classifying different assault types, and the overall goodness-of-fit of our models. By considering multiple evaluation metrics, we are able to gain a more nuanced understanding of the strengths and weaknesses of each algorithm and make informed decisions about their suitability for our classification task. These metrics deliver a complete evaluation of the classifiers' effectiveness in detecting community intrusions.

***Keywords*:** machine learning algorithms, UNSW-NB15, MSE, Intrusion Detection System (IDS), network attacks

MSC: 32C05, 32C07

1. Introduction

The Internet of Things (IoT) has emerged as a groundbreaking technology, fueling automation in a wide range of applications. According to IoT Analytics [1], there are an astounding 21.2 billion connected devices, with 11.6 billion of them being IoT devices. It is anticipated that the number of IoT devices will reach 22 billion by 2025, driven by a 10 percent rise in the number of coupled devices [2]. Nonetheless, this rapid expansion has also resulted in a heightened risk of network threats, underscoring the need for a robust emphasis on network security [3]. In this context, ensuring network security becomes crucial, and Intrusion Detection (ID) plays a vital role in safeguarding networks against cyber-attacks. An IDS serves as the first line of defense, typically consisting of a device or software strategically placed within a network to observe all traffic [4]. The primary functions of IDS include reporting malicious behaviors to network administrators, halting such behaviours, and identifying intruders [5]. Figure 1 illustrates the picture of IDS in the IoT network.

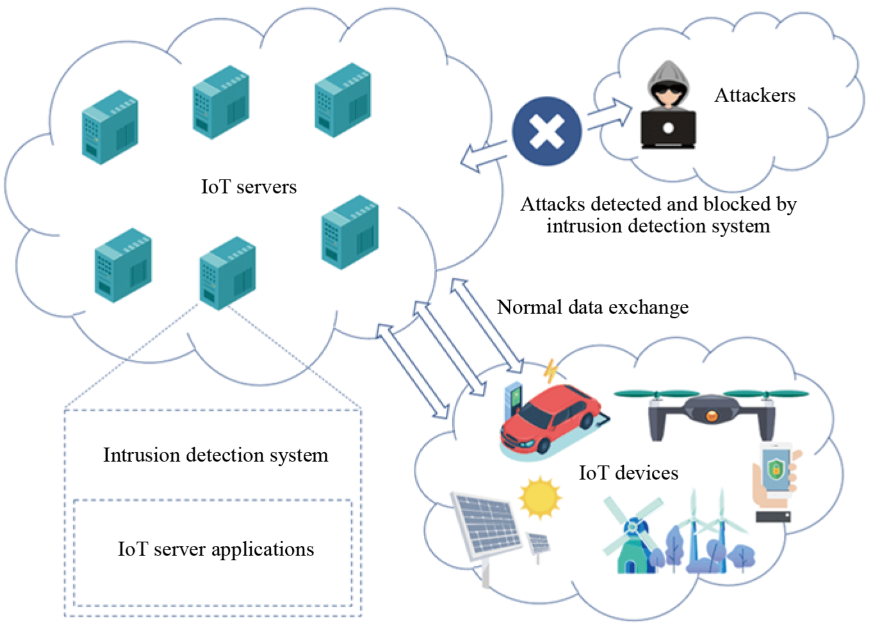

IoT servers
Attackers
Attacks detected and blocked by intrusion detection system
Normal data exchange
Intrusion detection system
IoT server applications
IoT devices

Figure 1. Intrusion Detection System (IDS) deployed in IoT networks

IDS can be categorized into three main forms: hybrid detection, signature, and anomaly-based detection. Anomaly-based detection relies on behavioral methods to define normal and abnormal behaviors. Any deviation from the established normal behaviors is considered suspicious, making anomaly detection particularly effective in identifying unknown or zero-day attacks. On the other hand, signature-based detection relies on known attack signatures to identify and detect future threats [6]. By leveraging a database of recognized attack signatures, this method can quickly identify and mitigate known attacks. Hybrid detection combines elements of both anomaly and signature-based approaches, utilizing the strengths of each to provide a comprehensive intrusion detection capability. This hybrid approach aims to enhance detection accuracy by incorporating the advantages of both methods.

ML plays a major role in the development of ID to effectively detect and mitigate threats. ML approaches are typically classified as supervised, unsupervised, or semi-supervised. In the context of IoT networks, which utilize diverse protocols and standards, the formation of heterogeneous networks poses a challenge for a single model to learn and adapt to different traffic patterns. To address this challenge, an ensemble model which integrates multiple learning models can leverage the strengths of each model to enhance classification accuracy. By aggregating the outputs of different models, the risk of selecting a weak classifier is reduced. Moreover, combining classifiers with several decision boundaries can lead to the creation of a superior decision boundary compared to individual classifiers. Nonetheless, it is imperative to acknowledge that the effectiveness of an ensemble classifier cannot always be guaranteed to surpass that of the top-performing individual classifier, nor is progress always guaranteed. Despite this, an ensemble classifier can mitigate the overall risk associated with making a suboptimal selection [7]. In the realm of Internet of Things (IoT) networks, Ensemble Learning (EL) techniques prove to be particularly advantageous for addressing intricate and nonlinear problems. EL involves amalgamating multiple models, of similar nature, that were trained on diverse partitions of the training dataset. Two common methods for combining the decisions of ensemble models are soft and hard voting. In hard voting (majority voting), each classifier casts a single vote for a class, and the result that receives the most votes is chosen. Soft voting, on the other hand, predicts the class on the basis of the predicted probabilities of the classifiers. Soft voting may provide an improvement over hard voting by taking into account the uncertainty of each classifier in the final decision, thus incorporating more information.

The field of intrusion detection has seen substantial developments in recent years with the incorporation of several machine learning (ML) algorithms. These algorithms play a crucial role in identifying and preventing potential cyber-attacks by analyzing patterns and anomalies in network traffic. Notwithstanding the profusion of scholarly investigations in this domain, a notable void persists in the form of a comprehensive comparative analysis regarding the efficacy of distinct machine learning (ML) techniques for detecting intrusions. Hence, the primary aim of this research endeavour is to meticulously examine and compare the performance of various ML algorithms by utilizing the UNSW-NB15 dataset. This dataset, encompassing a wide spectrum of network traffic data, serves as an exceptional basis for assessing the efficacy and efficiency of diverse ML methods in identifying intrusions. By analyzing this dataset, we aim to identify the strengths and weaknesses of each algorithm, allowing us to make informed recommendations on which ML methods are most suitable for intrusion detection in different network environments. Moreover, this research aims to shed light on the areas that require further improvement and research in the field of intrusion detection using ML algorithms.

The work is structured into 7 sections to systematically convey its content. Section 2 provides an in-depth exploration of the background relevant to the subject matter. In Section 3, an overview of various machine learning approaches is presented. Section 4 introduces the UNSW NB15 dataset, serving as a concise introduction to the dataset used for the experimental work. Following that, Section 5 outlines the proposed approach, while Section 6 delves into the details of the feature selection methods employed. The actual experimental work is elaborated upon in Section 7, and Section 8 serves as the conclusion, summarizing the findings and elucidating future research prospects.

2. Related works

This chapter presents a comprehensive literature survey focusing on ML methods in the area of intrusion detection. Its primary objective is to offer an overview of the study performed in the area. Extensive efforts have been made by researchers to explore various ML algorithms, and the following highlights some of their significant contributions.

In their research, Khammassi et al. [8] utilized the Genetic Algorithm (GA) and the LR wrapper-based feature selection method to analyze the datasets of KDDCup99 and UNSW-N15. The Weka simulation program was used for this study, with numerous simulations being conducted. The results indicated that the combination of GA-LR and Decision Tree (DT) classifier led to a detection accuracy of 81.42% on the UNSW-NB15 dataset, with a relatively low False Alarm Rate (FAR) of 6.39%. This was achieved by utilizing a subset of 20 features out of the original 42. Moreover, for the KDDCup99 dataset, the GA-LR approach and the DT classifier achieved a FAR rate of 0.105% and a detection score of 99.90%, utilizing only 18 features. These findings demonstrate the effectiveness of the GA-LR approach in feature selection for improving detection accuracy and reducing the false alarm rate in network intrusion detection systems.

In their study, Osanaiye et al. [9] put forth a filter-based approach to detect DDoS attacks, employing multiple filter techniques such as Relief, Gain Ratio, Chi-Square, and Information Gain. Utilizing the "NSL-KDD attack detection" dataset, the authors successfully demonstrated the effectiveness of their system by achieving a remarkable 99.67% detection accuracy with a minimal False Alarm Rate of 0.42%. The Decision Tree (DT) method was used for classification, which was trained and validated through a 10-fold cross-validation approach. However, it is important to note that the multiclass classification aspect of the tNSL-KDD dataset was not thoroughly investigated in this study. Nevertheless, the results obtained from this research present a promising solution for detecting DDoS attacks with high accuracy and minimal false alarms.

Ambusaidi et al. [10] an IDS combined with a filter-stimulated input reduction approach became explored. The researchers evaluated exceptional datasets, along with KDDCup99, Kyoto 2006, and NSL-KDD, to assess the performance in their approach. To decide the correlation among exclusive I/P variables, the authors employed a Flexible Mutual Information (FMI) approach, which is a non-linear correlation degree. The classifier employed of their experiments became the LS-SVM, which stands for Least Square Support Vector Machine. The observe's results revealed that when the use of the LS-SVM FMI to pick 18 capabilities on the NSL-KDD dataset, the technique completed an outstanding accuracy of 99.94% even as retaining a remarkably low False Alarm Rate (FAR) of 0.28%. The "LS-SVM FMI" finished with a whole accuracy of 78.86% for the KDD Cup ninety-nine dataset. On the 10th new release, the "LS-SVM FMI" in the instance of Kyoto 2006 received a detection rate of 97.80% and a FAR charge of 0.43%.

Zhou et al. [11] several machine learning algorithms were tested on a proposed feature embedding method. The algorithms evaluated were K-Nearest Neighbors (KNN), DT, Logistic Regression, Gaussian Naïve Bayes (NB), and Support Vector Machine (SVM). The performance of these models was assessed with the NSL-KDD & UNSW-NB15 datasets. On the NSL-KDD dataset, the models obtained the following accuracy scores: SVM-98.86%, Gaussian NB-98.8%, LR-98.85%, DT-98.77%, and KNN-98.82%. On the dataset of UNSW-NB15, the models obtained the following accuracy scores: SVM-92.32%, Gaussian NB-92.52%, Logistic Regression-90.35%, DT-92.29%, and KNN-91.9%.

Miller et al. [12] 3 decision combination approaches have been tested. The most accurate approach obtained an accuracy of 84.1%. In this approach, two different Naïve Bayes (NB) models' decisions were combined using the NB model. However, there is a concern regarding the results because using NB for classification and combining the models' decisions may have influenced the accuracy, potentially leading to an inflated result.

In their study, Tama et al. [13] proposed a two-stage classifier that combines Rotation Forest and bagging techniques. This approach utilizes Principal Component Analysis (PCA) to establish feature subsets, resulting in improved classification performance. However, when tested on the NSL-KDD and UNSW-NB15 datasets, the suggested model achieved an accuracy of only 85.7% and 72.52%, respectively. While these results may be considered satisfactory, they do not match the state-of-the-art performance on these datasets. Further analysis and improvements may be necessary to enhance the accuracy of this proposed classifier.

In their paper "A Two-Stage Intrusion Detection System Utilizing Deep Belief Networks and DT", Aloqaily et al. [14] proposed a novel approach for detecting intrusions in computer networks. The system utilized Deep Belief Networks (DBN) to reduce the dimensionality of the data, which was then passed to a Decision Tree (DT) for intrusion detection. The results of their evaluation on the NSL-KDD dataset showed an impressive accuracy of 99.43%. However, it should be noted that the use of DBN did introduce a delay which increased as the number of nodes in the network increased. These findings highlight the trade-off between accuracy and processing time in intrusion detection systems.

In their study, Siddiqui et al. [15] utilized multiple Support Vector Machine (SVM) models along with the bagging technique. The quantity of SVM models employed corresponded to the number of classes present in the dataset. The researchers exclusively concentrated on utilizing flow-based features to train the SVM models. Remarkably, the proposed Intrusion Detection System (IDS) achieved exceptional F-scores of 0.98 and 0.99, respectively, when subjected to evaluation using the CICIDS2017 and BotNetIoT datasets.

Connelly et al. [16] advised an Intrusion Detection System (IDS) that employed an Incremental approach of the Extreme Learning Machine (IELM) in aggregate with the Advanced Principal Component Analysis (APCA) algorithm. The APCA set of rules became used to dynamically select the most significant capabilities needed by the IELM, enabling optimum attack prediction. To investigate the performance of their counseled IDS framework, the researchers applied the us-NB15 dataset. The primary metric considered for assessing the overall performance became the accuracy completed on the test statistics. Additionally, they taken into consideration other metrics like FAR and DR. The experimental findings tested that the IELM-APCA IDS carried out an accuracy of 70.51%. The detection rate was measured at 77.36%, while the FAR has been determined to be 35.09%. It's really worth noting that the overall performance of the cautioned IDS framework becomes assessed using those metrics at the dataset of UNSW-NB15.

Gao et al. [17] an IDS adapted to the Edge-of-Things (EoT) model was introduced. EoT expands the functionality of the Internet of Things (IoT) by allowing fast data processing and transmission from "cloud servers" to "edge devices" including laptops, mobile phones, portable computers, and IoT devices. The EoT-IDS has a feature selection module that chooses the most relevant features by using the correlation approach. Once the feature subset has been selected, intrusion predictions are made using a "Deep Belief Network" (DBN) classifier. To evaluate the effectiveness of its suggested system, the investigator used the dataset of UNSW-NB15. The main performance parameter was the accuracy attained on the test data. The best configuration, which had 64 "hidden units" in the first layer, 60 hidden units in the 2nd layer, and a serial number of 28 for the DBN architecture, produced accuracy for the binary classification process of 85.73 percent during the experiments the authors carried out using various DBN network structures.

Almogren et al. [18] proposed a Network Intrusion Detection System (NIDS) framework that incorporated various techniques to optimize the performance of intrusion detection. The framework utilized the One-Side Selection (O-SS) strategy to reduce noisy data records in the majority classes, thereby improving the quality of the dataset. To tackle the issue of imbalanced data, the authors employed a synthetic method called Synthetic Minority Over Sampling (SMOS) to increase the number of minority instances. For feature extraction, Convolutional Neural Networks (CNN) were utilized to extract spatial attributes, while Bidirectional Long-Short Term Memory (Bi-LSTM) models were utilized to capture temporal features. The combination of CNN and Bi-LSTM served as the deep learning model for predictive tasks. To evaluate the proposed system, two intrusion detection datasets, namely UNSW-NB15 and NSL-KDD, were used. The accuracy achieved on the test data was considered the primary performance metric.

Jiang et al. [19] a feature reduction approach on the basis of the RF algorithm was implemented to create the Feature Importance (FI) scores for every attribute in the dataset of UNSW-NB15. Based on their significance in the classification process w.r.t. target variable, the features are ranked using the FI algorithm (class). A feature subset with 11 attributes was chosen after multiple trials were performed. The author considered several ML approaches, including the above-mentioned parameters for the classification procedure. To evaluate the efficiency of such approaches, the accuracy (AC) and F-Measure (FM) scores attained on the test data have been utilized as the primary metrics. Among the evaluated approaches, the RF method achieved the best findings, with an accuracy of 75.56 percent and an F-Measure of 73.00 percent. These findings indicate that the RF algorithm, in combination with the selected feature subset, performed well in classifying instances within the UNSW-NB15 dataset.

In a study conducted by Satyanarayan et al. [20], a framework of GA-EMs was proposed and evaluated for its effectiveness in utilizing Ensemble Machine Learning techniques. The authors explored multiple algorithms and assessed their overall performance. The findings revealed that Random Forest emerged as the top-performing technique, achieving an impressive accuracy rate of 98.06%. Following closely behind, XGboost demonstrated a commendable accuracy rate of 97.99%. Additionally, Extra-Trees and AdaBoost showcased competitive accuracy rates of 97.80% and 97.57% respectively. Stacking, while not the highest, still achieved a respectable accuracy rate of 97.30%. These results highlight the varying levels of performance exhibited by different ensemble techniques within the GA-EMs framework.

In their study, Rani et al. [21] recommended employing the COMSOL software to simulate and adjust the properties of arterial tissue. They conducted initial exploratory analysis to gain insights into the behavior of variables and target functions. Subsequently, they trained machine learning regression models using data obtained from a theoretical human carotid artery. To assess the accuracy of these models, they compared the results with flow division ratios documented in current literature. Mean Absolute Error values were calculated for the test dataset to evaluate the performance of the models. The obtained values were as follows: polynomial regressor (0.0106), hyper-tuned support vector regressor (0.0487), decision tree regressor (0.000), random forest regressor (0.0156), Adaboost (0.0508), gradient-boosting (0.0044), and XGboost (0.0043). Additionally, the authors incorporated a quantile loss feature to evaluate the prediction uncertainty.

Karmakonda et al. [22] have introduced a novel relay selection scheme that combines learning automata and Particle Swarm Optimization (PSO) to achieve energy-efficient relay selection and ensure reliable data delivery. In this innovative approach, the network undergoes clustering utilizing the well-known Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol. To enhance the stability of Cluster Heads (CH), a departure from the traditional LEACH protocol is made, replacing random number assignment with sensor node energy levels. Each sensor node within the network employs the sophisticated PSO algorithm to estimate the most optimal routes to the sink node. A noteworthy aspect of this method is the introduction of learning automata for successor node selection in cases of packet loss, eliminating the need for retransmissions.

3. Overview of ML approaches

The supervised ML methods that were applied in this work are reviewed in the sections that follow.

3.1 *Logistic regression*

Logistic regression [23], despite its name, is a classification technique that can also be used for regression tasks. It relies on a "sigmoid predictive function", described as

$$h(z) = \frac{1}{1+e^{-Z}} \tag{1}$$

In (1), z indicates a linear function. This function returns a "probability score", P, ranging from 0 to 1. To map these probabilities to 2 discrete classes (0 or 1), a threshold value, θ, is set. The predicted class is assigned as 1 if P is greater than or equal to θ, and 0 otherwise. In this way, logistic regression could be used to conduct binary classification tasks.

3.2 *Suppport vector machine*

Similar to the aforementioned approaches, SVM [24, 25] is intended to categorize data points by locating a hyperplane in an N-dimensional space. SVM aims to create a clear decision boundary that distinctly separates data points of different classes. To achieve this, SVM calculates a margin distance between data points belonging to different classes. The area between the hyperplane, as well as the closest data points for each class, is known as the margin. The SVM seeks to optimize this margin to discover the best decision boundary that successfully divides the classes. By identifying the hyperplane with the maximum margin, SVM creates a robust classifier that can accurately classify future data points based on their position relative to the decision boundary. SVM is particularly useful when dealing with complex datasets where a clear separation between classes is not easily achieved.

3.3 *Decision tree*

A DT [26] is a method that aims to divide individuals into groups based on their similarities in relation to the target variable. The method constructs a tree structure that represents hierarchical links between variables. The decision tree-building process is iterative. At every iteration, the algorithm selects an explanatory variable that provides the best separation of individuals into distinct groups. This variable is used as a splitting criterion to create branches in the tree, resulting in subsets of individuals that exhibit similar characteristics. The process continues recursively, with each subset of individuals being further divided based on the most informative variables until a stopping criterion is met. The stopping criterion is reached when no further splits can be made, indicating that the tree has captured the patterns and relationships in the data to the best extent possible.

3.4 *Random forest*

RF algorithms are powerful techniques used for classification and regression tasks [27, 28]. They consist of an ensemble of multiple Decision Trees that serve as individual predictors. The fundamental concept behind Random Forest is to generate a significant number of Decision Tree models instead of relying on a single optimized model. Every Decision Tree (DT) within the forest undergoes training on a distinct, randomly selected portion of the dataset, employing a diverse assortment of features. This deliberate approach generates variation in the predictions offered by the individual trees. When it comes to the prediction phase, each DT independently furnishes its own forecast. The ultimate forecast in classification problems, determined through a majority vote, is selected based on the class that garners the highest number of votes from the constituent trees. In the case of regression tasks, the estimated value is generally calculated as the average or median of the estimated values from every tree. By combining the predictions from multiple trees and considering their collective wisdom, Random Forests can provide robust and accurate predictions. They are known for their ability to handle complex datasets, mitigate overfitting, and provide insights into variable importance.

3.5 *K-nearest neighbour*

The KNN method is a versatile ML technique that could be applied for supervised as well as unsupervised processes. It serves as the foundation for many clustering methods in use today [29]. In the present study, we employed the kNN approach in its supervised machine learning form. The technique relies on the Standard Euclidean Metric (EM), which measures the distance between 2 points or instances in a given space [30]. Let's consider two instances, p, and q, in a "Euclidean" space denoted as Z.

$$\Delta(p,\, q) = \sqrt{\sum_{i=1}^{r} (qk - qk)^2} \tag{2}$$

The EM between p and q, denoted as $(p,\, q)$, is computed as the sum of the squared differences between the corresponding attributes of the instances. To determine the identity or label of a given example, $r0$, in the space Z, the kNN approach calculates the EM between $r0$ and its k closest cases in Z. The label or class of $r0$ is then determined since the majority class among its k nearest neighbours.

3.6 *Extra-tress*

Extra Trees (ET), also known as "extremely random trees", is an advanced supervised machine learning technique that goes beyond traditional decision tree algorithms. By utilizing multiple decision trees (DTs) for decision making, ET provides a powerful and versatile approach for both classification and regression tasks. One key distinction of ET compared to other methods, such as Random Forest (RF), is its use of the entire training data set to build individual DTs. Unlike RF, which constructs each DT from a subset of the training data, ET leverages the full data set, allowing for a more comprehensive analysis. Additionally, the ET methodology introduces an extra layer of randomness by randomly selecting split points for each node. This increased randomness enhances the diversity of the individual trees, leading to improved accuracy and robustness in the final prediction. Overall, ET offers a unique and effective approach to machine learning by combining the strengths of decision trees with additional randomness. Its ability to handle both classification and regression tasks, along with its utilization of the entire data set, sets it apart from other methods. By incorporating this advanced technique into predictive models, researchers and practitioners can achieve more accurate and reliable results in various domains [31, 32].

3.7 *XGBoost*

The XGBoost algorithm stands as a potent boosting technique designed to enhance the capabilities of tree-based algorithms. Although XGBoost, Extra Trees (ET), and Random Forest (RF) all fall within the category of ensemble methods in machine learning, notable differences exist between XGBoost and RF/ET. Unlike RF and ET, where decision trees operate as independent entities, XGBoost takes a different approach. In XGBoost, each decision tree is constructed by extending an existing tree and learning from the errors made by the prior trees in the ensemble. This sequential learning process empowers XGBoost to effectively capture intricate patterns and elevate predictive accuracy [33–35].

3.8 *Stacking*

Wolpert [36] introduced stacking as an ensemble strategy to diminish error rates in generalization. This strategy accomplishes the objective by amalgamating various primary learners and integrating meta-learners into the ensemble. The ensuing steps elucidate the stacking process in the Figure 2.

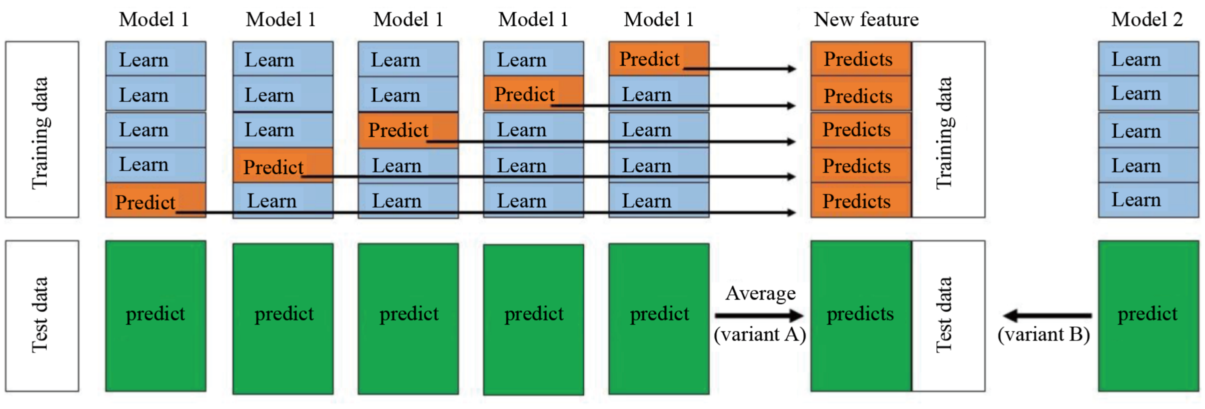


Figure 2. The process of stacking

3.8.1 *Data splitting*

Initially, the dataset is partitioned into a training dataset and a testing dataset.

3.8.2 *K-fold split*

Typically, K-fold cross-validation technique is used to segregate the training dataset into K equal segments. For illustration, if we set K to five, we will subdivide the dataset into five equitably sized portions.

3.8.3 *Primary learner training*

In the stacking paradigm, the primary learner uses four of these training segments to train itself. The primary learner incorporates knowledge and improves its predictive abilities by considering the input features and target labels within these four segments.

3.8.4 *Validation set*

The trained primary learner utilizes the remaining portion of the training dataset, known as the validation, set, to generate predictions. These predictions, along with the original input features, serve as fresh input features for the meta-learner.

3.8.5 *Meta-learner training*

The meta-learner, a unique machine learning algorithm, undergoes training by employing the projected values from the primary learner as input features, in conjunction with the factual target labels derived from the validation set. As a result, it acquires the aptitude to generate predictions based on this amalgamation of information.

3.8.6 *Predictive application*

After successfully mastering the training of the stacking process, individuals are able to skilfully utilize it for the essential tasks of forecasting and categorizing novel instances or data points. This powerful technique allows the primary learner to generate accurate predictions on fresh data, which are then seamlessly fed into the meta-learner. As a result, the meta-learner is able to analyse and synthesize the predictions from multiple primary learners, ultimately yielding an optimal prediction or risk estimation for the assigned tasks. The stacking process offers immense value in the field of data analysis and decision-making. By combining the strengths and expertise of various primary learners, it enhances the accuracy and reliability of predictions on new data points. This technique empowers professionals to make well-informed decisions based on robust and comprehensive insights. Whether it is in financial forecasting, medical diagnosis, or customer behaviour prediction, the stacking process has proven to be an effective tool in handling complex and diverse datasets. With its ability to generate reliable predictions, it has become an indispensable asset in many industries, ensuring informed decision-making and ultimately driving success and growth.

The stacking methodology facilitates the integration of the collective merits possessed by multiple models, leveraging the predictions of one model to inform the inputs of another. This approach frequently yields performance improvements and elevates the level of precision in predictions, surpassing the solitary capabilities of a single model.

3.9 *Multi-level perceptron (MLP)*

A MLP [37, 38] is a kind of Artificial Neural Network (ANN) made up of many linked layers of perceptron-like neurons. As a feedforward neural network, it only allows only one direction of data flow-from the I/P layer to the O/P layer. An I/P layer, one or more hidden layers, and an O/P layer make up the MLP. The neurons in neighboring layers are completely linked, which indicates that each neuron in one layer is linked to every neuron in the layer above it. Each layer is made up of a collection of neurons, also known as nodes. Each neuron in an MLP gets inputs from the neurons in the layer below, processes the weighted sum of these inputs using an “activation function”, and then creates an output. The network may learn complex patterns as well as correlations in the data due to the activation function’s introduction of non-linearity. The MLP is trained using a process called backpropagation, which involves forward propagation of input data through the network to produce an output, comparison of the output with the desired output, calculation of the error, and adjustment of the weights in the network to minimize the error. This process is repeated iteratively until the network learns to make accurate predictions. MLPs are widely utilized for various tasks like classification, regression, and pattern recognition. They have proven to be effective in solving complex problems and can handle large amounts of data. Nevertheless, they may suffer from overfitting if the model becomes too complex or if the dataset is insufficient.

4. UNSW NB15 dataset

In our experimental procedures, we utilized the ”UNSW-NB15” attacks dataset [39]. Initially, this dataset consisted of 45 features, as detailed in Table 1.

4.1 *Handling to types of attack*

There are nine groups in which attack kinds may be classified.

4.1.1 *Fuzzers*

These attacks involve the attacker attempting to discover security loopholes in a program, network, or operating system. They accomplish this by inundating the system with copious volumes of unstructured data, resulting in its destabilization leading to a system failure.

4.1.2 *Analysis*

This category includes various intrusions that target web applications through different means such as port scans, spam emails, and web scripts like HTML files. The goal is to gain unauthorized access or exploit vulnerabilities in the system.

Table 1. List features of the UNSW-NB15 dataset

S.NO	Feature	Dtype	S.NO	Feature	Dtype
1	id	int64	24	dwin	int64
2	dur	float64	25	tcprtt	float64
3	proto	object	26	synack	float64
4	service	object	27	ackdat	float64
5	state	object	28	smean	int64
6	spkts	int64	29	dmean	int64
7	dpkts	int64	30	trans_depth	int64
8	sbytes	int64	31	response_body_len	int64
9	dbytes	int64	32	ct_srv src	int64
10	rate	float64	33	ct_state_ttl	int64
11	stti	int64	34	ct dst 1tm	int64
12	dttl	int64	35	ct_src_dport_1tm	int64
13	sload	float64	36	ct_dst_sport_1tm	int64
14	dload	float64	37	ct_dst src 1tm	int64
15	sloss	int64	38	is_ftp_login	int64
16	dloss	int64	39	ct_ftp_cmd	int64
17	sinpkt	float64	40	ct_flw_http_mthd	int64
18	dinpkt	float64	41	ct_src 1tm	int64
19	sjit	float64	42	ct_srv_dst	int64
20	djit	float64	43	is_sm_ips_ports	int64
21	swin	int64	44	attack cat	object
22	stcpb	int64	45	label	int64
23	dtcpb	int64			

4.1.3 *Backdoor*

Backdoor attacks involve bypassing normal authentication mechanisms to gain unauthorized remote access to a device. Attackers create hidden entry points that allow them to access the system discreetly, often seeking plain text information without being detected.

4.1.4 *Denial of Service (DoS)*

DoS attacks aim to disrupt computer resources, typically by overwhelming them with excessive demands. These attacks make the system extremely busy, rendering it unable to process authorized requests, effectively denying access to legitimate users.

4.1.5 *Exploit*

Exploits are sequences of instructions that take advantage of glitches, bugs, or vulnerabilities in a host or network. Attackers leverage these unintentional or unsuspected behaviors to gain unauthorized access or compromise the system.

4.1.6 *Generic*

This technique targets any block cipher with a hash function to cause a collision, regardless of the block-cipher configuration. The objective is to undermine the security of the encryption algorithm by exploiting weaknesses in the hash function.

4.1.7 *Reconnaissance*

Also known as probing, reconnaissance attacks involve gathering data about a "computer network" to evade its security controls. Attackers collect valuable data to understand the system's vulnerabilities and plan subsequent attacks.

4.1.8 *Shellcode*

In shellcode attacks, the attacker injects a small piece of code, often starting from a shell, to gain control over the compromised machine. This enables them to execute malicious actions and potentially take full control of the system.

4.1.9 *Worm*

Worm attacks involve self-replication to spread to other computers. They often utilize computer networks to propagate themselves, relying on security vulnerabilities in target computers to gain access and continue spreading. The dataset in Table 2 provides a comprehensive breakdown of each attack class, offering detailed information and illustrating the distribution of values across dissimilar data subsets.

Table 2. Types of attacks in the UNSW-NB15 dataset

Attack type	UNSW-NB15	UNSW-NB15-TRAIN-1	UNSW-NB15-VAL	UNSW-NB15-TEST
Normal	56,000	41,911	14,089	37,000
Generic	40,000	30,081	9,919	18,871
Exploits	33,393	25,034	8,359	11,132
Fuzzes	18,184	13,608	4,576	6,062
DoS	12,264	9,237	3,027	4,089
Reconnaissance	10,491	7,875	2,616	3,496
Analysis	2,000	1,477	523	677
Backdoor	1,746	1,330	416	583
Shellcode	1,133	854	279	378
Worms	130	99	31	44

5. The proposed approach

The following list includes the steps to create the models:

i. Begin by pre-processing the dataset to prepare it for analysis.

ii. Utilize the Pearson correlation coefficient method to identify the most significant features.

iii. Divide the dataset into 2 separate parts: the training set and the testing set.

iv. Construct classifier models using various algorithms like.

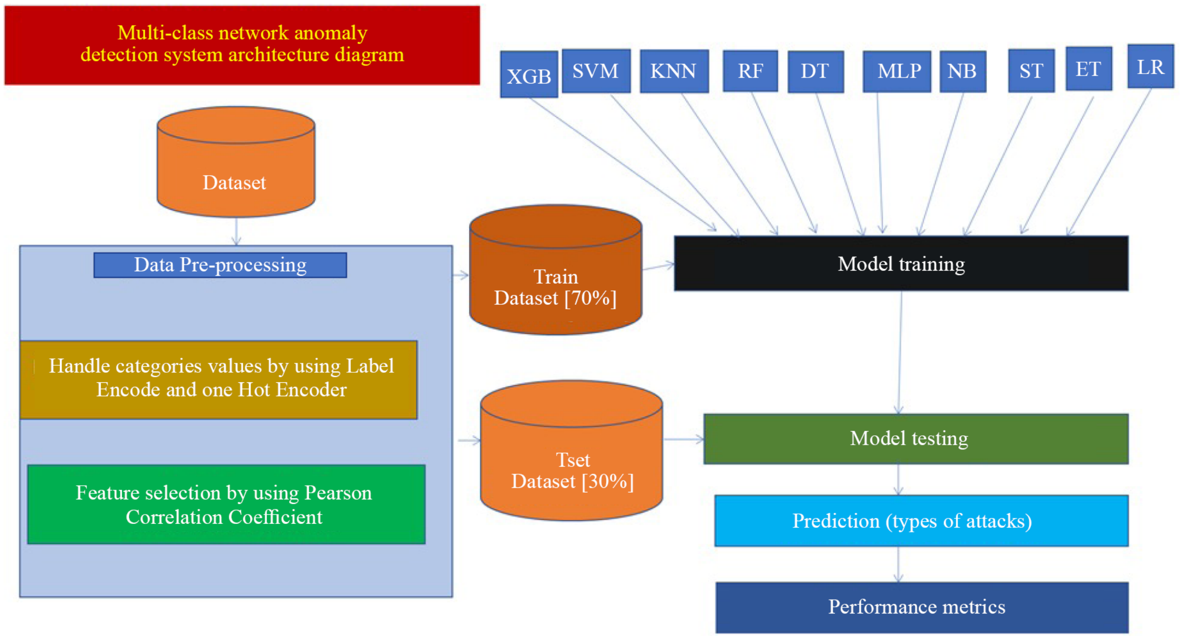
v. ET, AdaBoost (ADAB), XGB, NB, MLP, LR, DT, RF, SVM, KNN, and Stacking based on the training data.

vi. Acquire the test data to evaluate the performance of the constructed models.

vii. Test the Multi-classifier models using the test data to assess their effectiveness.

viii. Compute and compare metrics such as F1-Score, Precision, Recall, Accuracy, and Mean Squared Error (MSE) for the chosen models to evaluate their performance.

Figure 3 presents a comprehensive system's architectural design suggested in the study. In the initial phase of our approach, we remove the "id" and "Label" attributes as they represent serial numbers and the Label is not relevant to the analysis. The pre-processing stage begins by addressing any null values within the dataset and eliminating them. Among the most utilized approaches for normalizing data is "min-max" normalization. The lowest and highest values of each characteristic are separately changed to 0 and 1, and all other values are converted to a decimal between 0 & 1. Next, the categorical data is transformed into a numerical format using a label encoder. Subsequently, a one-hot encoder is applied to eliminate any inherent ordering between the values obtained from the label encoder. Following pre-processing, the next steps involve selecting significant features using the Pearson correlation coefficient. Here's an algorithm1 that outlines the steps to create and evaluate classifier models using the specified process.



Multi-class network anomaly detection system architecture diagram
XGB
SVM
KNN
RF
DT
MLP
NB
ST
ET
LR
Dataset
Data Pre-processing
Train Dataset [70%]
Model training
Handle categories values by using Label Encode and one Hot Encoder
Tset Dataset [30%]
Model testing
Feature selection by using Pearson Correlation Coefficient
Prediction (types of attacks)
Performance metrics

Figure 3. Block diagram of the proposed method

Algorithm 1 Algorithm for proposed approach

```
Step 1: Begin
Step 2: Pre-processing:
# Perform necessary data preprocessing steps (e.g., handling missing values, scaling).
Step 3: Feature Selection:
significant_features = Identify Significant Features (dataset)
# Utilize Pearson correlation coefficient or other methods to identify significant features.
Step 4: Data Splitting:
training_set, testing_set = SplitDataset (dataset, test_size)
# Divide the dataset into training and testing sets, specifying the test set size.
Step 5: Model Training:
classifier_models = TrainClassifierModels (training_set, algorithms)
# Train classifier models using various algorithms (ET, AdaBoost, XGB, NB, MLP, LR, DT, RF, SVM, KNN, Stacking).
Step 6: Test Data Acquisition:
test_data = Load TestData ()
# Acquire test data for evaluation.
Step 7: Model Evaluation:
results = Evaluate Classifier Models (classifier_models, test_data)
# Test the classifier models using the test data to assess their effectiveness.
Step 8: Performance Metrics:
metrics ComputeMetrics (results, test_data)
# Compute evaluation metrics such as F1-Score, Precision, Recall, Accuracy, and MSE.
```

```
Step 9: Comparison and Reporting:
ReportMetrics (metrics)
Step 10: End
# Display or log the computed metrics to evaluate model performance.
```

6. Feature selection technique

This section provides an overview of various feature selection approaches utilized for important features for model construction [40]. These methods can be classified into hybrid, wrapper, and filter methods. Filter approaches evaluate every feature independently of the classifier, rank them, and choose the most superior ones. An example of a filter-based method is the Pearson correlation coefficient method. Wrapper methods, on the other hand, depend on the selected classifier. These approaches select a subset of the "feature set", determine the performance of the classifier, and then assess the next subset. The subset that achieves the best performance is chosen. An example of a wrapper method is Recursive Feature Elimination (RFE). Hybrid approaches combine the advantages of filter and wrapper techniques. They leverage the strengths of each approach to select the most relevant features. For the experimental purposes in this paper, Pearson's correlation coefficient [41, 42] from the filter method is employed. Figure 4 illustrates the various feature selection techniques used.

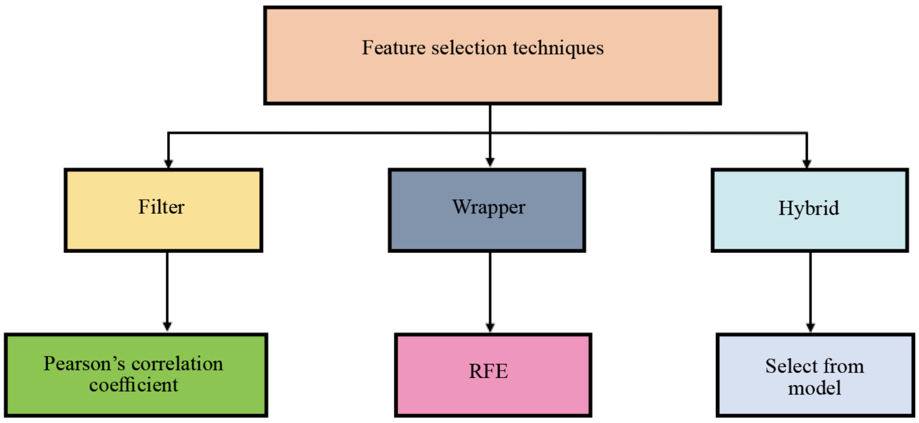

Feature selection techniques
Filter
Wrapper
Hybrid
Pearson's correlation coefficient
RFE
Select from model

Figure 4. Feature selection techniques

The "Pearson correlation coefficient", usually referred to as Pearson's "*r*" or just the "correlation coefficient", is a statistical indicator that defines the magnitude and the direction of the linear connection between 2 continuous variables. It has the symbol "*r*" and has a value between -1 and 1. The covariance of the two variables is divided by the total of their standard deviations to determine the Pearson correlation coefficient. Pearson's correlation coefficient is calculated using the following formula (3).

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x^2)][n\sum y^2 - (\sum y^2)]}} \tag{3}$$

In (3), r indicates Pearson Coefficient.
n denotes the number of pairs of the stock.
Σxy signifies the sum of products of the paired stocks.

Σx represents of the x scores.
Σy presents the sum of the y scores.
Σx^2 denotes the sum of the squared x scores.
Σy^2 indicates the sum of the squared y scores.
Interpreting the magnitude of the correlation coefficient:
$|r| < 0.3$: Weak correlation.
$0.3 \leqslant |r| < 0.7$: Moderate correlation.
$|r| \geqslant 0.7$: Strong correlation.

Pearson's correlation coefficient is commonly applied in feature selection to evaluate the linear correlation between features and the target variable. Features with higher correlation coefficients may be considered more relevant and informative for the model. It is important to note that correlation does not imply causality and that other factors should be considered while selecting model attributes. In the context of the experimental purposes described in the paper, Pearson's correlation coefficient value is considered as indicating a moderate correlation. The specific threshold for determining what constitutes a "moderate" correlation may vary depending on the specific field or application. In general, correlation coefficients falling within the range of 0.3 to 0.7 (inclusive) are often considered as indicating a moderate level of correlation.

Through this process, we reduce the original 43 features to 7 using feature selection techniques. Here's Algorithm 2 for calculating the Pearson correlation coefficient to identify significant features in a dataset.

Algorithm 2 Algorithm for calculating the Pearson correlation coefficient

```
Function Calculate Pearson Correlation (dataset, feature_x. feature_y):
n = length(dataset) # Number of data points
sum x = 0
sum_y = 0
sum_xy = 0
sum_x_squared = 0
sum_y_squared = 0
# Calculate the sums of $X, Y, XY, X^2$, and $Y^2$
for each data_point in dataset:
sum_x += data_point[feature_x]
sum_y+= data_point[feature_y]
sum_xy += data_point[feature_x] * data_point[feature_y]
sum_x_squared += data_point[feature_x] * data_point[feature_x]
sum_y_squared += data_point[feature_y] * data_point[feature_y]
# Calculate the Pearson correlation coefficient
numerator = (n * sum_xy) (sum_x * sum_y)
denominator = sqrt((n * sum_x_squared - sum_x * sum_x) * (n * sum_y_squared-sum_y * sum_y))
r = numerator/denominator
retum r
```

After pre-processing the dataset, we apply several popular machine learning techniques to gain insights into their performance on this dataset. To evaluate the multi-classification of different types of attacks, we measure various performance metrics like Accuracy, MSE, Recall, Precision, F1-Score, R2-Score, Mean Absolute Error (MAE), Root Mean Square Error (RMSE). These metrics provide a comprehensive evaluation of the classifiers' effectiveness in detecting network intrusions.

7. Experimental work

The experimental work is performed on a system with an "Intel Core (TM) i3-1005G1 CPU @1.20 GHz" and 4 GB RAM. The Scikit-Learn ML Python framework was used to develop, train, assess, and test the ML models. A robust and adaptable open-source platform called Scikit-Learn is constructed on top of other libraries like matplotlib and NumPy. It provides a comprehensive set of tools and functionalities for machine-learning tasks.

The ML algorithms ET, XGB, NB, MLP, DT, RF, KNN, SVM, LR, and Stacking (ST) classifiers, are examined on the dataset of UNSW-NB15, which is a new dataset for intrusion detection. Performance evaluation is based on multiple parameters on selected features, as presented in Table 3.

Table 3. Comparison of the performance of chosen classifiers with the UNSW-NB15 dataset

Classifiers	Accuracy	Precision	Recall	F1-Score	MAE	MSE	RMSE	R2-Score
Logistic Regression (LR)	97.58	0.97	0.98	0.97	0.06	0.18	0.42	87.87
Support Vector Machine (SVM)	97.59	0.97	0.98	0.98	0.05	0.17	0.42	87.93
K-Nearest Neighbors (KNN)	97.37	0.97	0.97	0.97	0.06	0.19	0.44	86.95
Random Forest (RF)	97.31	0.97	0.97	0.97	0.06	0.19	0.44	86.63
Decision Tree (DT)	97.19	0.97	0.97	0.97	0.06	0.20	0.45	86.17
Multi-Layer Perceptron (MLP)	97.54	0.97	0.98	0.97	0.06	0.17	0.42	87.97
Naïve Bayes (NB)	95.28	0.95	0.95	0.95	0.20	0.99	0.99	35.29
XGBoost (XGB)	97.42	0.97	0.97	0.97	0.06	0.18	0.42	87.68
Extra Trees (ET)	97.21	0.97	0.97	0.97	0.06	0.20	0.45	86.73
Stacking (ST)	97.27	0.97	0.97	0.97	0.06	0.19	0.44	86.73

Performance evaluation is based on multiple parameters on selected features, as presented in Table 3. From the findings in Table 3, it could be found that the SVM classifier achieves the highest accuracy of 97.58%, and lowest MSE of 0.18 among all the methods. On the other hand, the NB classifier exhibits the lowest accuracy of 95.28% and the highest MSE of 0.99 among the selected classifiers. The accuracy of selected classifier using UNSW-NB15 is shown in Figure 5.

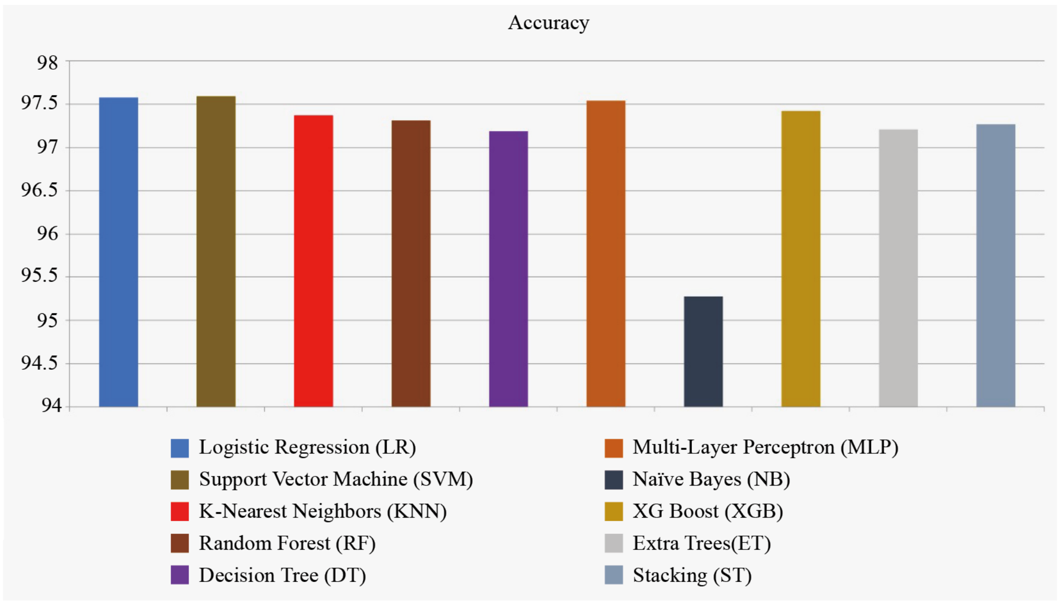


Figure 5. Accuracy of selected classifiers when using the dataset of UNSW-NB15 with the selected features

8. Conclusion and future scope

In the conducted experimental work, the performance of ML classifiers, like LR, SVM, KNN, RF, DT, MLP, NB, XGB, ET, and ST, was evaluated for intrusion detection using the UNSW-NB15 dataset. The classifiers were compared based on various metrics like RMSE, MAE, R2-Score, F1-Score, Recall, Precision, MSE, and Accuracy. The results of the evaluation indicate that the SVM classifier outperformed the other classifiers on the UNSW-NB15 dataset, considering the chosen parameters. Specifically, the SVM classifier achieved an accuracy of 97.58% when using selected features. This demonstrates the effectiveness of the SVM classifier in accurately identifying instances of intrusion within the dataset. Furthermore, we plan to implement the Transfer learning Based Framework for Zero-Day Attack Detection.

The future of machine learning in renewable energy applications [43, 44] is promising, with continued advancements in forecasting, grid management, energy storage, and resource allocation. These developments will contribute to a more sustainable and environmentally friendly energy landscape, helping to mitigate the impact of climate change.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Lueth KL. *State of the IoT 2018: Number of IoT Devices Now at 7B-Market Accelerating*. 2018. Available from: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ [Accessed 30th October 2023].

[2] McKay RM, Pendleton B, Britt J, Nakhavanit B. Machine learning algorithms on botnet traffic: ensemble and simple algorithms. In *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*. New York, NY, USA: Association for Computing Machinery; 2019. p.31-35. Available from: https://doi.org/10.1145/3314545.3314569.

[3] Aldwairi M, Mardini W, Alhowaide A. Anomaly payload signature generation system based on efficient tokenization methodology. *International Journal on Communications Antenna and Propagation*. 2018; 8(5): 421. Available from: https://doi.org/10.15866/irecap.v8i5.12794.

[4] Mohamed T, Otsuka T, Ito T. Towards machine learning based IoT Intrusion Detection service. In *Lecture Notes in Computer Science*. Cham: Springer International Publishing; 2018. p.580-585. Available from: https://doi.org/10.1007/978-3-319-92058-0_56.

[5] Butun I, Morgera SD, Sankar R. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*. 2014; 16(1): 266-282.

[6] Cha Z, Ma Y. *Ensemble Machine Learning*. New York, NY: Springer; 2012.

[7] Raschka S, Mirjalili V. *Python Machine Learning*. 2nd ed. Birmingham, UK: Packt; 2012.

[8] hammassi C, Krichen S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers & Security*. 2017; 70: 255-277. Available from: https://doi.org/10.1016/j.cose.2017.06.005.

[9] Osanaiye O, Cai H, Choo KR, Dehghantanha A, Xu Z, Dlodlo ME. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*. 2016; 2016(1): 130. Available from: https://doi.org/10.1186/s13638-016-0623-3.

[10] Ambusaidi MA, He X, Nanda P, Tan Z. Building an intrusion detection system using a FilterBased feature selection algorithm. *Transactions on Computers/IEEE Transactions on Computers*. 2016; 65(10): 2986-2998. Available from: https://doi.org/10.1109/tc.2016.2519914.

[11] Zhou Y, Han M, Liu L, He JS, Wang Y. Deep learning approach for cyberattack detection. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Honolulu, HI, USA: IEEE; 2018. p.262-267. Available from: https://doi.org/10.1109/infcomw.2018.8407032.

[12] Miller ST, Busby-Earle C. Multi-perspective machine learning a classifier ensemble method for intrusion detection. In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*. New York, NY,

USA: Association for Computing Machinery; 2017. p.7-12. Available from: https://doi.org/10.1145/3036290.3036303.

[13] Tama BA, Comuzzi M, Rhee K. TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access*. 2019; 7: 94497-94507. Available from: https://doi.org/10.1109/access.2019.2928048.

[14] Aloqaily M, Otoum S, Ridhawi IA, Jararweh Y. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*. 2019; 90: 101842. Available from: https://doi.org/10.1016/j.adhoc.2019.02.001.

[15] Siddiqui AJ, Boukerche A. TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of Things. *Cluster Computing*. 2020; 24(1): 17-35. Available from: https://doi.org/10.1007/s10586-020-03153-8.

[16] Connelly L. Logistic regression. *Medsurg Nursing*. 2020; 29(5): 353-354.

[17] Gao J, Chai S, Zhang B, Xia Y. Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis. *Energies*. 2019; 12(7): 1223. Available from: https://doi.org/10.3390/en12071223.

[18] Almogren A. Intrusion detection in Edge-of-Things computing. *Journal of Parallel and Distributed Computing*. 2020; 137: 259-265. Available from: https://doi.org/10.1016/j.jpdc.2019.12.008.

[19] Jiang K, Wang W, Wang A, Wu H. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access*. 2020; 8: 32464-32476. Available from: https://doi.org/10.1109/access.2020.2973730.

[20] Satyanarayana G, Chatrapathi KS. Improving intrusion detection performance with genetic algorithm-based feature extraction and ensemble machine learning methods. *International Journal of Intelligent Systems and Applications in Engineering*. 2023; 11(4): 100-112.

[21] Rani TR, Al Shibli A, Siraj M, Srimal W, Al Bakri NZS, Radhika TSL. ML-based approach to predict carotid arterial blood flow dynamics. *Contemporary Mathematics*. 2023; 5(1). Available from: https://ojs.wiserpub.com/index.php/CM/article/view/3224 [Accessed 30th October 2023].

[22] Karmakonda K, Das MS, Ravi G. An energy-efficient learning automata and cluster-based routing algorithm for wireless sensor networks. *Contemporary Mathematics*. 2023; 4(3): 488-504. Available from: https://doi.org/10.37256/cm.4320232654.

[23] Khan NM, Madhav CN, Negi A, Thaseen IS. Analysis on improving the performance of machine learning models using feature selection technique. In *Advances in Intelligent Systems and Computing*. Cham: Springer; 2019. p.69-77. Available from: https://doi.org/10.1007/978-3-030-16660-1_7.

[24] Wang H, Xiong J, Yao Z, Lin M, Ren J. Research survey on support vector machine. In *10th EAI International Conference on Mobile Multimedia Communications*. Chongqing, China: EAI; 2017. p.95-103. Available from: https://doi.org/10.4108/eai.13-7-2017.2270596.

[25] Rahman MM, Islam MM, Manik MMH, Islam MR, Al-Rakhami MS. Machine learning approaches for tackling novel Coronavirus (COVID-19) pandemic. *SN Computer Science*. 2021; 2(5): 384. Available from: https://doi.org/10.1007/s42979-021-00774-7.

[26] Patel BR, Rana KK. A survey on decision tree algorithm for classification. *International Journal of Engineering Development and Research*. 2014; 2(1): 1-5.

[27] Breiman L. Random forests. *Machine Learning*. 2001; 45(1): 5-32. Available from: https://doi.org/10.1023/a:1010933404324.

[28] Hastie T, Tibshirani R, Friedman JH. *The Elements of Statistical Learning*. New York, NY: Springer; 2009. Available from: https://doi.org/10.1007/978-0-387-84858-7.

[29] Belouch M, Hadaj SE, Idhammad M. A two-stage classifier approach using RepTree algorithm for network intrusion detection. *International Journal of Advanced Computer Science and Applications*. 2017; 8(6): 389-394. Available from: http://dx.doi.org/10.14569/IJACSA.2017.080651.

[30] Gao J, Chai S, Zhang B, Xia Y. Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis. *Energies*. 2019; 12(7): 1223. Available from: https://doi.org/10.3390/en12071223.

[31] Ahmad MW, Reynolds J, Rezgui Y. Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. *Journal of Cleaner Production*. 2018; 203: 810-821. Available from: https://doi.org/10.1016/j.jclepro.2018.08.207.

[32] Alsariera YA, Adeyemo VE, Balogun AO, Alazzawi AK. AI Meta-Learners and Extra-Trees algorithm for the detection of phishing websites. *IEEE Access*. 2020; 8: 142532-142542. Available from: https://doi.org/10.1109/access.2020.3013699.

[33] Devan PAM, Khare N. An efficient XGBoost-DNN-based classification model for network intrusion detection system. *Neural Computing & Applications*. 2020; 32(16): 12499-12514. Available from: https://doi.org/10.1007/s00521-020-04708-x.

[34] Scikit-Learn. *Ensemble Gradient Boosting Classifier*. Available from: https://scikit-learn.org/stable/modules/generated/ [Accessed 30th October 2023].

[35] Scikit-Learn. *Gradient Boosting Classifier*. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html [Accessed 30th October 2023].

[36] Wolpert DH. Stacked generalization. *Neural Networks*. 1992; 5(2): 241-259.

[37] Amato F, Mazzocca N, Moscato F, Vivenzio E. Multilayer perceptron: an intelligent model for classification and intrusion detection. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. Taipei, Taiwan: IEEE; 2017. p.686-691. Available from: https://doi.org/10.1109/waina.2017.134.

[38] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. Canberra, ACT, Australia: IEEE; 2015. p.1-6. Available from: https://doi.org/10.1109/milcis.2015.7348942.

[39] Anwer HM, Farouk M, Abdel-Hamid A. A framework for efficient network anomaly intrusion detection with features selection. In *2018 9th International Conference on Information and Communication Systems (ICICS)*. Irbid, Jordan: IEEE; 2018. p.157-162. Available from: https://doi.org/10.1109/IACS.2018.8355459.

[40] Scikit-Learn. *Machine Learning in Python*. Available from: https://scikit-learn.org/stable [Accessed 30th October 2023].

[41] Hauke J, Kossowski T. Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*. 2011; 30(2): 87-93. Available from: https://doi.org/10.2478/v10117-011-0021-1.

[42] Scikit-Learn. *machine learning in Python-scikit-learn 142 documentation*. Available from: https://scikit-learn.org/stable [Accessed 30th October 2023].

[43] Krishna VM, Vuddanti S, Narendra B, Prasad K. Experimental study on self-excited induction generator for small-scale isolated rural electricity applications. *Results in Engineering*. 2023; 18: 101182. Available from: https://doi.org/10.1016/j.rineng.2023.101182.

[44] Krishna VM, Vuddanti S. Identification of the best topology of delta configured three phase induction generator for distributed generation through experimental investigations. *International Journal of Emerging Electric Power Systems*. 2021; 23(3): 329-341. Available from: https://doi.org/10.1515/ijeeps-2021-0064.