Received: 8 June 2022 | Revised: 6 July 2022 | Accepted: 11 July 2022 | Published online: 13 July 2022

Journal of Computational and Cognitive Engineering
2023, Vol. 2(2) 88–97
DOI: 10.47852/bonviewJCCE2202270

BON VIEW PUBLISHING

RESEARCH ARTICLE

Machine Learning-Based Intrusion Detection System: An Experimental Comparison

Imran Hidayat[1], **Muhammad Zulfiqar Ali**[2] **and Arshad Arshad**[3,*]

[1]*School of Computing, Edinburgh Napier University, UK*

[2]*James Watt School of Engineering, University of Glasgow, UK*

[3]*School of Computing, Glasgow Caledonian University, UK*

Abstract: Recently, networks are moving toward automation and getting more and more intelligent. With the advent of big data and cloud computing technologies, lots and lots of data are being produced on the internet. Every day, petabytes of data are produced from websites, social media sites, or the internet. As more and more data are produced, a continuous threat of network attacks is also growing. An intrusion detection system (IDS) is used to detect such types of attacks in the network. IDS inspects packet headers and data and decides whether the traffic is anomalous or normal based on the contents of the packet. In this research, ML techniques are being used for intrusion detection purposes. Feature selection is also used for efficient and optimal feature selection. The research proposes a hybrid feature selection technique composed of the Pearson correlation coefficient and random forest model. For the machine learning (ML) model, decision tree, AdaBoost, and K-nearesrt neighbor are trained and tested on the TON_IoT dataset. The dataset is new and contains new and recent attack types and features. For deep learning (DL), multilayer perceptron (MLP) and long short-term memory are trained and tested. Evaluation is done on the basis of accuracy, precision, and recall. It is concluded from the results that the decision tree for ML and MLP for DL provides optimal accuracy with fewer false-positive and false-negative rates. It is also concluded from the results that the ML techniques are effective for detecting intrusion in the networks.

Keywords: MLP, LSTM, KNN, IDS, machine learning

1. Introduction

A network intrusion detection system (NIDS) is used to detect unwanted or malicious traffic in the network. An intrusion detection system (IDS) detects anomalies or attacks in real time. Nowadays, mostly applications are moving to the cloud. Due to rapid and fast growth of network devices, security risks got increased. For that reason, the security of cloud infrastructure and network resources is the main priority in the modern world. Therefore, IDS should be accurate, error-free, and efficient. Due to the advent of cloud computing, the Internet of Things (IoT) and quantum computing huge amount of data are being created every day, which are known as big data. This big data also helps in training the machine learning (ML) model for security purposes. Network security is a challenging field nowadays. IDS provides promising results in determining the intrusion in the network.

IDS mainly consists of two types: anomaly-based and signature-based. Anomaly-based IDS used ML or deep learning (DL) techniques to detect data patterns. Signature-based IDS works on predefined attacks and rules. IDS detects threats by monitoring traffic data in computer networks and issues alert after detecting threats. IDS can be passive or active depending on its alert system.

Today, many researchers are working in the field of IDS. ML and DL techniques are used to detect network anomalies by using historical data and standard datasets. Alkhatib et al. (2021) proposed ML algorithms for intrusion detection purposes. Naïve Bayes algorithm is used in the research. The results obtained from Naïve Bayes algorithm are compared with support vector machines (SVM). Tao et al. (2018) used SVM and a genetic algorithm for attack detection. Detection accuracy is increased by optimizing the selection parameters and weights. Kim et al. (2014) used K-nearesrt neighbor (KNN) and K-means for intrusion detection purposes. The detection accuracy got increased in this research. Shapoorifard (2017) proposed a novel technique to detect the attacks. First, data are segmented into smaller clusters using C 4.5 algorithm, and then multiple SVM models are created from the subset of the data. This technique reduces the time complexity of the model. Zhao et al. (2017) proposes a work based on deep belief networks. The dimensionality of data is reduced by using probabilistic models. The probabilistic neural network is used for the classification of data.

However, several problems exist in the IDS domain like low accuracy, high false-positive rates, and relevant feature selection problem. The contributions of this research article described below:

*__Corresponding author:__ Arshad Arshad, School of Computing, Glasgow Caledonian University, UK. Email: arshad.100@strath.ac.uk

1. It provides a ML algorithm for the purpose of detecting intrusion in the network.
2. It provides an efficient and effective feature selection technique based on the correlation among features.
3. It provide comparative analysis with other ML techniques.
4. It increase the detection accuracy of the ML model.

The paper is organized as follows: Section 2 presents the literature review and related work, Section 3 is dedicated to methodology, and Sections 4 and 5 presents results and conclusion, respectively.

2. Related Work

According to Bashir and Chachoo (2014), organizations are facing security threats every day in the form of malware and cyberattacks. IDS and intrusion prevention system detect and prevent the network from these malwares. Raghunath and Mahadeo (2008) propose a NIDS that detects the attacks in the network by using ML techniques and associated pattern analysis technique to detect anomaly in the network.

Gadze et al. (2021) proposes IDS for software-defined networks and detecting distributed denial-of-service (DDoS) attack in the network. DL-based convolutional neural networks (CNN) and long short-term memory (LSTM) models are presented and evaluated. Overall, 89.63% accuracy is achieved in this research. The performance of the model is also compared with other state-of-the-art ML algorithms. Maseer et al. (2021) use the CICIDS 2017 dataset for making of IDS. This is one of the new and flow-based dataset with new attack categories. The authors utilized DL and proposed a new technique, namely AIDS. The researcher in this study evaluates the performance by using true-positive and true-negative rates. KNN-AIDS and decision tree (DT)-AIDS obtain the best results in this research.

Wang et al. (2020) uses the NSL-knowledge data discovery (KDD) dataset for IDS. Several ML algorithms are used in the study. A new framework named SHAP is proposed in the research. This algorithm combines local and global explanations for IDS. Vinayakumar et al. (2019) use the DL approach along with KDD CUP 99 datasets for the making of IDS. In this research, 1,000 epochs are set for each experiment. This model is also applied to different datasets like NSL-KDD, UNSW-NB 15, and CICIDS 2017 to measure the performance. In this research, high-dimensional features are also learned by the model. This model also provides optimal accuracy.

Rajagopal et al. (2021) uses Azure ML platform for IDS. Meta-classification approach is used for both binary and multi-classification purposes. Three datasets are used in the research such as UNBSW, CICIDS, and CICDOS. 99.8% accuracy is achieved on UNSW, whereas 99% on CICIDS and 98% on CICDOS. Train and test split ratio of 40:60 is used in the research Ahmed et al. (2020). The DL technique is used for IDS. UNSW-NB 15 dataset is used for training and testing purposes. In this research, CNN is used with regularized multilayer perceptron (MLP) instead of fully connected layers. Keras library is used for development purposes. The model is trained on GPU. Early stopping is also used to prevent the model from overfitting.

Saranyaa et al. (2019) uses KDD CUP 99 datasets with several ML algorithms like linear discriminate analysis (LDA), classification and regression tree (CART), and random forest. Random forest achieves the highest accuracy with 99.8%, LDA with 98%, and CART with 98.1%. Zhang and Ran (2021) uses DL for IDS. CNN algorithm is proposed in this research along with Google Net inception to detect network packets binary problem. Overall, 99.63% accuracy is achieved. Gao et al. (2019) use NSL-KDD dataset. A new ML model called the adaptive ensemble learning model is proposed. Multi-tree algorithm is proposed to increase the overall performance of the algorithm. 84.2% accuracy is achieved in this research.

Ring et al. (2021) proposed host-based IDS. In this research, the DL model is presented. A new algorithm called ALAD is also proposed in this research. This new model detects application-level attacks in the network. Optimal accuracy is achieved through this model. This model is also compared against other state-of-the-art algorithms. Devarakonda et al. (2022) use the NSL-KDD dataset in the research. In this DL model, the autoencoder is proposed. Both NIDS and host-based IDS are proposed in this research.

The application of DL is widely used in the field of IDS. DL provides promising results when there is a huge amount of data which need to be processed. In the security field, an enormous amount of data is sometimes received from different sources and there is a need to process that data quickly or efficiently.

Ashiku and Dagli (2021) proposes DL-based IDS to detect network attacks. They developed a flexible IDS which also detects zero-day attacks. UNSW-NB 15 dataset is used for this purpose. Overall, 95.4% accuracy is achieved in this research. Tang et al. (2020) used IDS 2018 dataset for research purposes. In their study, a novel attention-based CNN-LSTM model is proposed which is based on DL. Several experiments are performed, and optimal accuracy is achieved in this research

Vladimir (1967) used NSL-KDD and UNSW-NB 15 datasets are used for training. The deep reinforcement learning approach is used. The new type of network traffic attack is detected automatically. The proposed model can process a million records of network traffic. Paper et al. (2016) proposes a new DT technique called self-taught learning. This technique learns features automatically from the data and feeds them to the model. They used NSL-KDD dataset for training. Optimal accuracy is achieved in this research.

Faker and Dogdu (2019) uses three classifiers to detect anomalies in the network. One is deep feed forward neural network (DNN), and the other is an ensemble technique based on random forest and gradient boosting. UNSW-NB and CICIDS 2017 datasets are used. Five cross-fold validation is also used for evaluation purposes. Experimentation is done using the spark library. 99.16% accuracy is achieved on the UNSW-NB dataset and 99.99% on the CICIDS dataset. Park et al. (2021) proposes a technique called HIIDS, which is hybrid intelligent IDS. This technique learns important and most relevant features from the dataset. LSTM and autoencoder are used. ISCX-UNB dataset is used for training. 97.52% accuracy is achieved in this research.

Istiaque et al. (2021) uses KDD CUP 99 datasets for training. Fifteen features are used along with the MLP algorithm. 95% accuracy is achieved in this research. 95% accuracy is achieved in this research. Alkhatib et al. (2021) uses the recurrent neural networks (RNN) model, which is based on the sequence model. They used their own generated dataset. Area under the curve (AUC) value of greater than 0.8 is achieved in this research.

Fu et al. (2022) used ML techniques for the IDS. Information gain and gain ratios are used for the selection of features. IoTID20 and NSL-KDD datasets are used in the research. Several ML algorithms like MLP, J48, IBK, and bagging are used in the research. 99% accuracy is achieved in the research. Tang et al. (2022) used DL in the research. NSL-KDD dataset is used in the research. Stacking-based model is used in the study which is the combination of various classification models to improve the accuracy. 86.8% accuracy is achieved in the research. The results of the research were also compared with four ML algorithms. This technique improves the overall detection accuracy of the detection model. Ullah et al. (2022) used DL to improve the accuracy of the intrusion detection model. CIC-IDS, CIC-DOS, and CSE-CIC-IDS 2018 datasets are used in the research. LSTM and GRU are used in the research. Overall, 99% accuracy is achieved in the research. Albulayhi et al. (2022) proposed an ML-based model to detect zero-day attacks in the network. A DL-based model consisting of CNN, autoencoder, and LSTM is proposed. CSE-CIC-IDS 2018 dataset is used in the research. The principal component analysis technique is used to select relevant features from the dataset. Better accuracy is achieved in the research. Halbouni et al. (2022) proposed the ML and DL models for intrusion detection purposes. The authors reviewed several approaches used for intrusion detection purposes. Recent ML and DL algorithms are also discussed by the authors which are used for IDS purposes. Kim and Pak (2022) proposed an ML model for intrusion detection purposes. Several algorithms are used, like AdaBoost, random forest, ELM, DNN, CNN, and XGBoost. 95% accuracy is achieved in the research.

3. Materials

Research methodology is discussed in this section. The effectiveness of using the ML technique is also discussed in this section.

3.1. Proposed framework

In ML model, we have a block or model diagram for development purposes. The methodology proposed for making an IDS is discussed below:

Figure 1 represents the proposed methodology of the ML model. In ML model, firstly, we take data from a source and then apply preprocessing techniques to that data. The data collected from different sources are not clean and sometimes may include null values, so in preprocessing, we remove null values and replace them with suitable ones. After null values removal, data need standardization and normalization. The standardization and normalization techniques put data in the range between 0 and 1. Step-by-step discussion about the model is discussed below.

Figure 2 represents the proposed model for the project. First step is the collection of the relevant data where data for the development of the proposed model is collected. Second step is the standardization of the data where mean is set to zero and standard deviation to one. Normalization is the process where all the values in the data are getting scaled to a certain range. In feature selection relevant features are getting selected for the development of the model. Training and testing is performed after the selection of the relevant features. Most effective features are get selected to train or test the performance of the ML model.

3.1.1. Dataset

The TON_IoT dataset used in the study is new dataset and includes all the latest network attacks. TON_IoT contains features related to the IoT traffic.

These are all the features which are present in the TON_IoT dataset. Not all these features are used for training purposes because not all are necessary for predicting attacks. For that purpose, a feature selection technique is used to select relevant features from the data. In feature selection techniques, relevant and most important features are selected, and the rest of the features are removed for training the model.

Table 1 represents the features of the dataset used for the model development. These are the standard features used for the development of the ML model. ts, date, time all features are collected from the real traffic and saved in the form CSV format for ML models. The features are captured in the form of packets and saved in the excel format to use by the ML model.

3.2. Preprocessing

The data which are collected for model training and testing purpose contain outliers and null values. These values need to be removed for the efficient working of the ML model.

The data contain categorical values and numerical values. The data are collected from real-time environments and saved as a comma separated values (CSV) to use for model-building purposes. Preprocessing involves several steps like normalization, standardization, label encoding, one-hot encoding, and feature scaling. All these steps are necessary for the development of the ML model. The details about these steps are described below

3.3. Data standardization

Data standardization is one of the most important parts of preprocessing. Standardization rescales the data so that its standard deviation becomes 1 and the mean becomes 0. Standardization brings down all features of the data to the common scale. The dataset which we used in ML mostly has

Figure 1
General framework for IDS

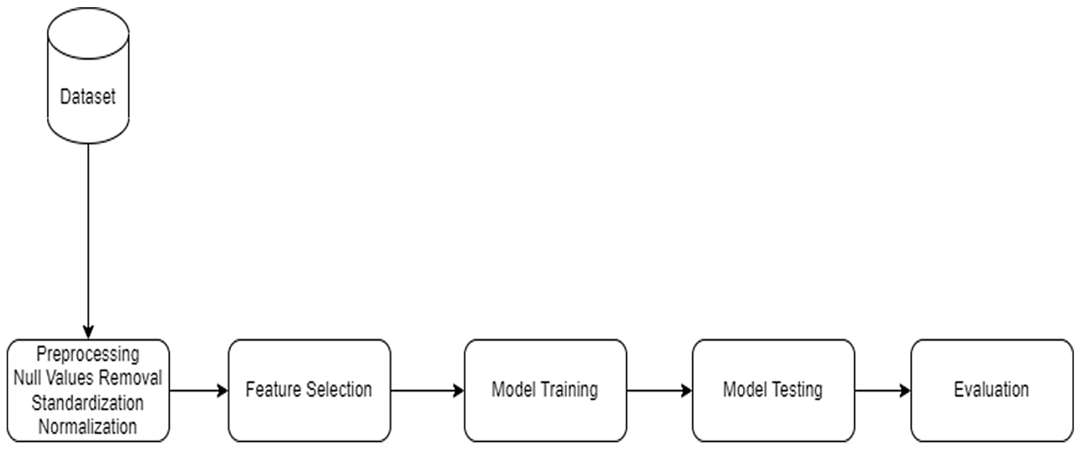
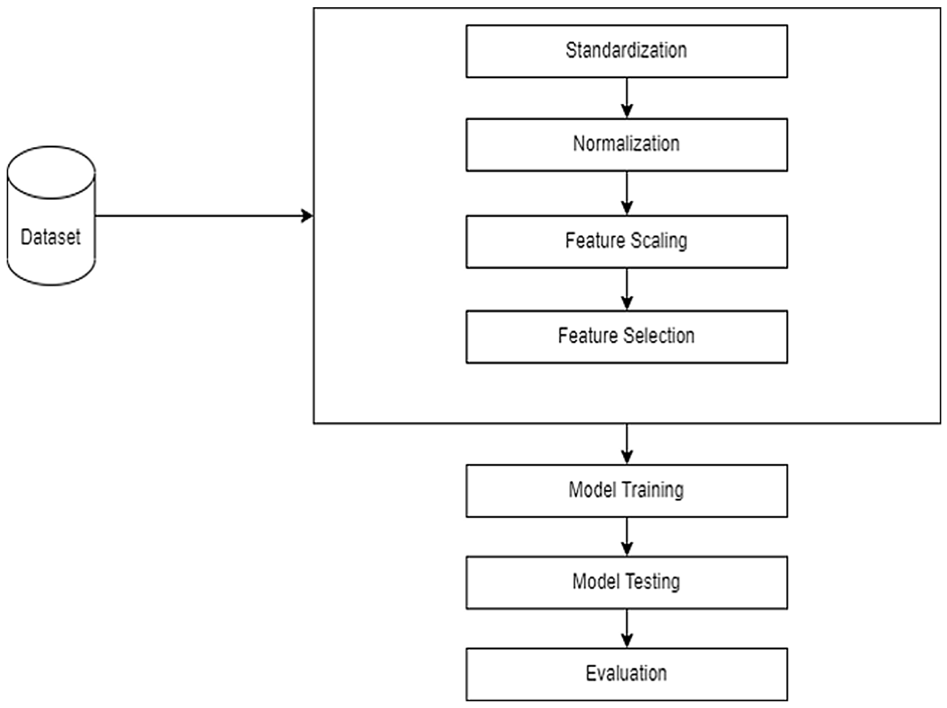


Figure 2
Proposed methodology



many features. The value of these features lies on a different scale. Consider an example of house price prediction in which the area of the house is 200 square meters, and the number of rooms is 1, 2, or 3. If we use this data without scaling, then ML gives more importance to the features with high values. ML models will learn faster when the data are on the same scale. One solution in ML for this problem is standardization. In standardization, mean value of the column is subtracted from each value and then divided by the standard deviation. In this way, data are normally distributed. In our work, we also do standardization. The resultant data obtained by standardization is shown below:

$$X = x - \mu/\sigma \tag{1}$$

Equation (1) is the standard equation of standardization, where μ is the mean of the data and σ is the standard deviation of the data.

3.4. Data normalization

Normalization is the second step in the process. The main purpose of normalization is to transform data in such a manner that the data are either dimensionless or similar distribution. Due to normalization, equal weight is given to each of the variables in the dataset:

$$X[:,i] = x[:,i] - \min(x[:,i])/\max(x[:,i]) - \min(x[:,i]) \tag{2}$$

In equation (2), min is the minimum absolute value of *a*, whereas max is the maximum absolute value of *a*.

3.5. Label encoding

The label encoder technique is used to convert categorical features to numerical. This technique converts each and every categorical value present in dataset to a number.

Table 1
Dataset features

Feature	Description
Ts	Timestamp
Date	Date of logging sensor data
Time	Time of logging sensor data
Fridge_temperature	Fridge sensor temperature measurement
Temp_condition	Fridge sensor temperature condition
Label	Normal or attack traffic data
Type	Normal or attack traffic type like DoS or DDoS.
Src_ip	IP address of source
Src_port	Port number of source computer
Dst_ip	IP address of destination
Dst_port	Port number of destination
Proto	Protocol either transmission control protocol (TCP) or user datagram protocol (UDP)
Duration	Connection duration
Src_bytes	Bytes sent by a source computer
Dst_bytes	Bytes received by a destination computer
Conn_state	State of connection
Missed_bytes	Bytes missed by destination
Src_pkts	Packets sent by a source
Src_ip_bytes	Number of IP bytes by a source
Dst_pkts	Destination packets
Dst_ip_bytes	Destination IP bytes
Dns_query	Type of Domain name system (DNS) query
http_response	Response generated by http
http_response	Response generated by http
http_status_code	Status code of http
http_version	Version of http
Weird_name	Whether a transmission control protocol (TCP) is bad or not

3.6. Data classes

After label encoding, we need to prepare our target column. For that purpose, we assign our data label to normal or abnormal for binary classification, and for multi-class classification, all the attacks are defined.

3.7. Data distribution

Data distribution plays a very important role in ML model training and testing purpose. If our data are imbalanced, then the results of ML might not be good. So balanced data distribution plays a vital role in ML. If dataset is not balanced, then we do synthetic minority oversampling technique (smote) to balance our dataset classes. In our case, our dataset is balanced, so we do not need any smote technique.

Figure 3 represents the data distribution of the TON_IoT dataset. It is evident from the figure that the data are balanced in the target class. Sixty-five percent of normal data is present, along with 35% of attack data. Normal distribution of data is mandatory for obtaining high accuracy because all the classes need to participate equally in model training.

Figure 4 represents the class distribution of multi-class data. In our target class, we have 10 types of attacks. Scanning, denial-of-service (DoS), DDoS, and man-in-the-middle (MITM) attack. These all are network attacks. ML models classify the traffic on the basis of these attacks. Scanning, DoS, MITM, injection, ransomware, backdoor, and XSS are all types of network attacks. The ML model is trained on data features to effectively and efficiently detect these attacks.

3.8. Feature selection

Feature selection is one of the most important tasks in the ML domain. Not all the features are used for model training. If so many features are present in the dataset, then they may increase the training

Figure 3
Data distribution

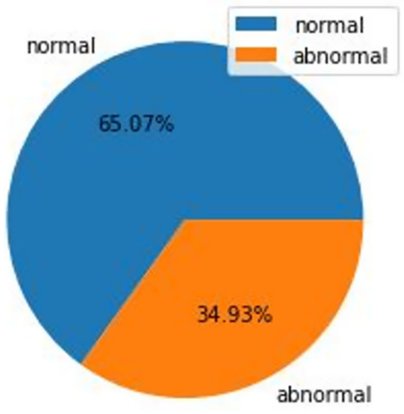
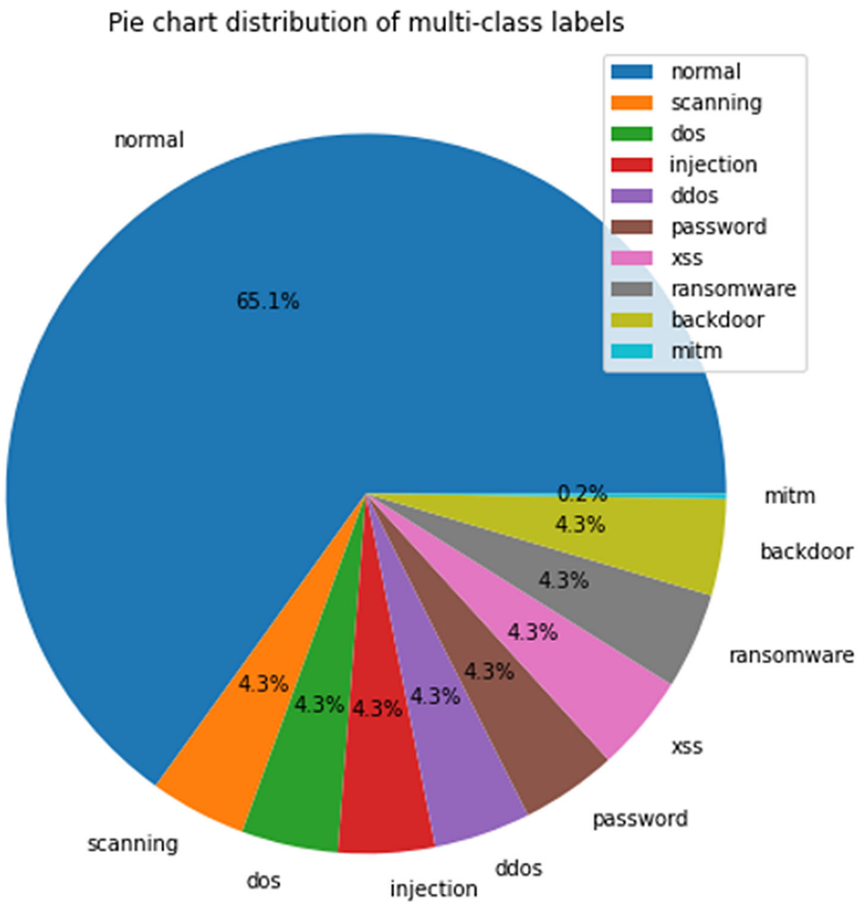


Figure 4
Overall attack distribution



time and complexity of the model. Sometimes, obtained data are very high dimensional, and we need to convert it to the lower dimension for efficiency and effective attack detection. So efficient feature selection technique is necessary to cope with this problem. Also, relevant feature selection is very important in ML. Sometimes, we remove some most important features and may get low accuracy. We need to cope with all these problems. Various feature selection techniques are used in ML like recursive feature elimination, chi-square, or backward feature selection techniques. These techniques are used based on datasets, dimensionality, and correlation. In our case, we use the Pearson correlation coefficient technique for feature selection. This technique works based on the correlation among variables.

3.9. Pearson's correlation coefficient

This technique works based on correlation. This technique depicts the linear relationship among the variables in the dataset. This technique can take a range of values between −1 and +1. A value of 0 indicates no relationship among variables, whereas −1 indicates a negative relationship and +1 indicates a positive relationship among the variables. If the relationship between two values is stronger, the correlation is close to +1.

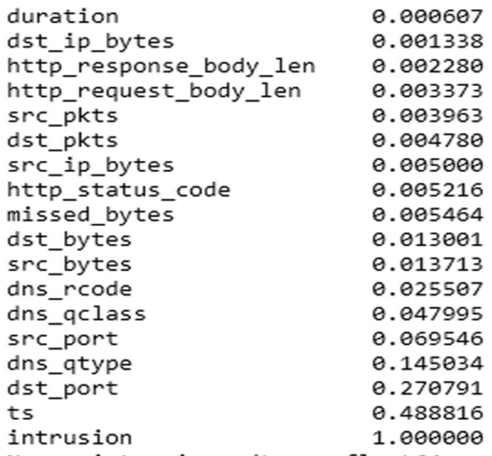
Figure 5 represents the features selected on the basis of the correlation score in our dataset. These are the final features which are selected for model training purposes. These features are further joined with one-hot encoded variable to form complete data.

3.10. Random forest feature scoring

In the research, random forest feature scoring is also utilized. The features obtained from the Pearson correlation technique are given to the random forest for further selection. The random forest model selects features on the basis of their importance. Seventeen features are selected with the Pearson correlation technique. These features are further reduced to 14 after utilizing the random forest technique. The features which are obtained from the random forest model are shown below.

Table 2 represents features that the random forest model selects. The features which are selected by the random forest model are referred to as true, and the features which are not selected are referred to as false.

Figure 5
Selected features



```
duration                    0.000607
dst_ip_bytes                0.001338
http_response_body_len      0.002280
http_request_body_len       0.003373
src_pkts                    0.003963
dst_pkts                    0.004780
src_ip_bytes                0.005000
http_status_code            0.005216
missed_bytes                0.005464
dst_bytes                   0.013001
src_bytes                   0.013713
dns_rcode                   0.025507
dns_qclass                  0.047995
src_port                    0.069546
dns_qtype                   0.145034
dst_port                    0.270791
ts                          0.488816
intrusion                   1.000000
```

Table 2
Random forest feature selection

Feature	Selection
Duration	True
Dst_ip_bytes	True
http_response_body_len	True
http_request_body_len	True
Src_pkts	True
Dst_pkts	True
Src_ip_bytes	True
Missed_bytes	True
Src_bytes	True
Dst_bytes	True
Dns_rcode	True
Src_port	True
Dst_port	True
ts	True

3.11. ML training

After all the preprocessing is done, then we have the model training phase. The features which are obtained from preprocessing stage is given to the ML model for training purpose. In the training phase, the data or features which are obtained from preprocessing stage are given to the model, and the model starts training on these features. Some ML algorithms take so much time for training, while some require less time. Sometimes, hyperparameter tuning is required if desired results are not obtained.

3.12. Model evaluation

After the testing phase, we need to evaluate our ML model on the basis of some parameters. For classification problems, confusion matrix, accuracy, and receiver operating characteristic (ROC) curve are used to measure the model's performance, whereas root mean square error is used for regression problems. Accuracy and precision are considered the benchmark in binary classification problems. If our mode obtains accuracy greater than 95% with less false-positive rate, then we conclude that the performance of our model is good and it is ready for a real-time production environment. Sometimes, we also consider precision, recall, and accuracy.

3.12.1. ROC curve

ROC curve is also used to measure the effectiveness of the binary classification problem. ROC curve plots two parameters, true-positive and false-positive. ROC curve is also appropriate when the class data is balanced, whereas, for imbalanced data, precision, recall, and f score are feasible.

4. Results

In this section, the results obtained from each model are described. Models are evaluated on the basis of accuracy, detection time, and testing time. The confusion matrix and ROC curve are used to evaluate model performance. ML models are tested on preprocessed dataset, and accuracy is calculated.

4.1. Decision tree results

After data preprocessing, ML model is applied to the preprocessed data. Data are trained and tested in the ratio of 75:25. Seventy-five percent of data is used for training and 25% is used for testing the model.

The accuracy obtained from the decision tree model is 99.6%, which is optimal. Accuracy means how our model is accurate, and its predictions are correct. If the ML model achieves an accuracy of 90% or greater, then we conclude that its performance is considered as good.

A decision tree is ideal for classification problems because it mostly gives maximum accuracy if data preprocessing is done properly. The decision tree is composed of nodes and branches, and besides feature selection, it automatically makes feature selection on each node when splitting occurs. So decision tree often is used in classification-related problems.

Figure 6 is the ROC curve obtained from decision tree results. ROC curve illustrates the capability of a binary classifier. A higher AUC value tells us that the model performance is better. The ROC curve is plotted against the model's true- and false-positive rates.

From the classification and ROC curve, it is concluded that the performance and accuracy of decision tree are optimal and accurate. So, decision tree can be used for IDS in real network environments.

4.3. KNN results

KNN algorithm is also used in our research. The results obtained from this model is described below.

The accuracy obtained from KNN algorithm is 99%. Although the accuracy obtained from KNN model is good, it takes so much time to train or test the model making it inappropriate for deployment purpose. When dealing with IDS, we also consider the time taken by the model to train or test the algorithm.

Figure 7 represents the RoC Curve for KNN algorithm. The auc value achieved by the KNN model is 1.00 which depicts the performance of the model. If auc score is near to 1 then it is concluded that the ML model is suitable for the real time deployment and it also achieves less false positive rates. The RoC curve is used mostly in binary classification tasks where true positive and false negative rates calculated. In order to make an effective classifier high RoC score is required.

Figure 6
ROC curve decision tree

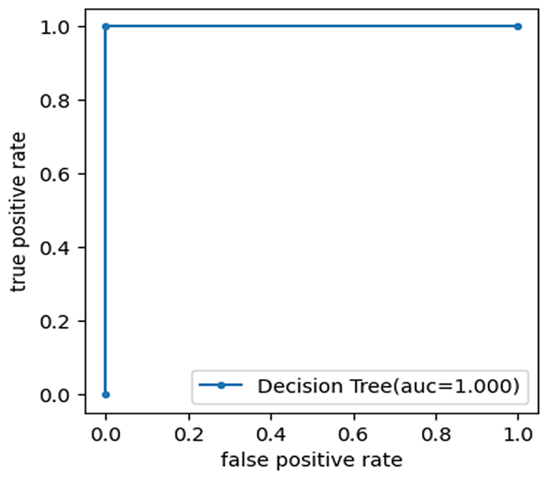
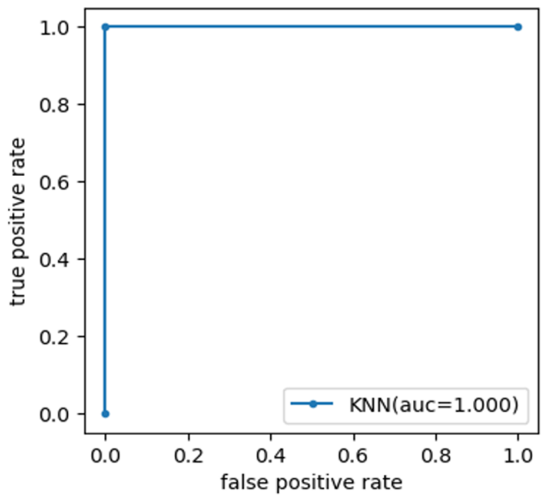


Figure 7
ROC curve KNN



4.4. AdaBoost accuracy

The accuracy achieved by the AdaBoost algorithm is 99.8%. The results obtained from this model are shown below. The accuracy obtained from the AdaBoost algorithm is 99.8%. Although the accuracy obtained from the AdaBoost model is good, it takes so much time to train or test the model making it inappropriate for deployment purposes.

The ROC area is 1.00 in the case of AdaBoost. This shows that the model performs well in terms of positive predictions. AUC area is not relevant to accuracy, and it only refers to the positive predictions of the model.

ROC and AUC curves determine the model's efficiency in case of true-positive and false-positive predictions. The area between the true-positive and false-positive rate is being determined by the ROC curve in the case of binary classification problems; however, in the case of multi-class classification problems, other parameters are considered.

From Table 2, it is evident that the accuracy of the ML models is good, but the testing time and training time of the decision tree are less than the other algorithms making it more appropriate for real-time detection and deployment.

Figure 8 represents the RoC Curve for AdaBoost model. The auc value achieved by the AdaBoost model is 1.00 which depicts the performance of the model. If auc score is near to 1 then it is concluded that the ML model is suitable for real time deployment and it also achieves less false positive rates. Boosting algorithms generally achieves good accuracy in classification tasks. Boosting involves the stacking classifiers in which different classifiers are stack on each other to perform classification.

Figure 9 represents the overall accuracy of the ML models trained in the research. From figure it is concluded that the supervised learning algorithms are suitable for the attack detection tasks and detect attacks with higher accuracy. Tree based

Figure 8
ROC curve AdaBoost

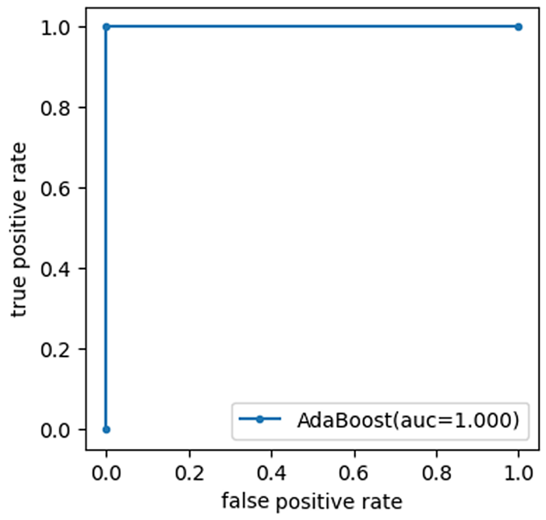


Figure 9
Comparison graph

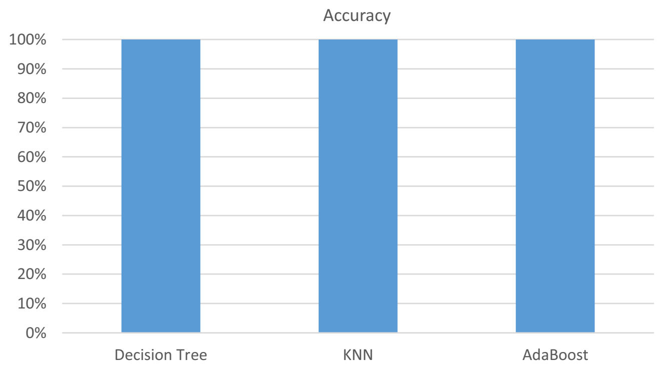


Table 3
Accuracy

Algorithm	Accuracy	Precision	Recall	F1 score
Decision tree	99.6%	99%	98%	99.8%
KNN	99%	99.2%	99.4%	99%
AdaBoost	99.8%	99.8%	99.2%	99.9%

algorithms mostly provides high accuracy in classification and prediction tasks.

Table 3 represents the overall accuracies of the ML models. It is concluded from the table that all the supervised learning algorithms achieves high accuracy in detection and prediction tasks.

4.5. DL results

The results of the DL model are evaluated on the basis of the ROC curve and model loss and accuracy curves. The results obtained from DL models on TON_IoT dataset are discussed below.

Figure 10
MLP accuracy curve

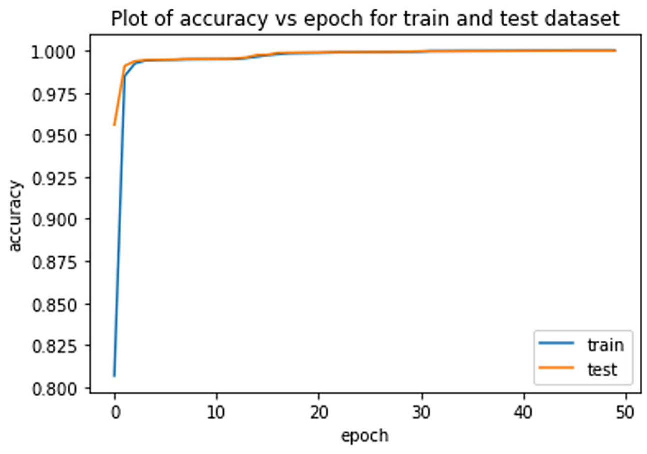
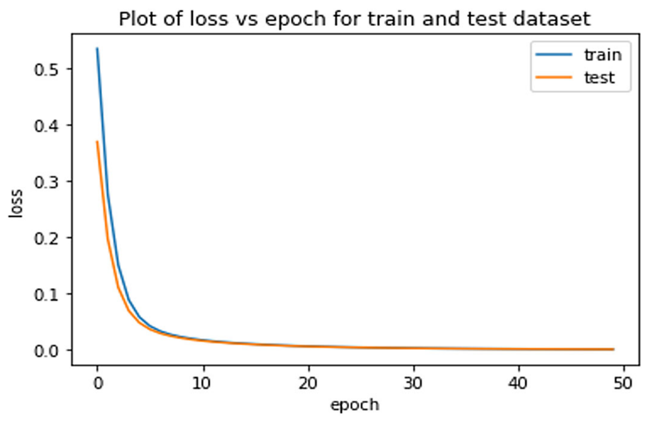


Figure 11
MLP loss curve



4.5.1. MLP results

Figure 10 represents the accuracy curve for MLP. This curve shows us that with the increase in the number of epochs, the accuracy of the model increases. At the start, the accuracy becomes low, but with the increase in epoch, accuracy increases.

Accuracy to epoch curve is the most important evaluation parameter for DL algorithms. With the increase in the number of epochs, the accuracy tends to be increased.

Figure 11 is the loss curve loss of the MLP model. It is evident from the figure that the loss of the model tends to be low with the increase in the number of epochs. So the number of epochs plays a significant role in determining the accuracy and reducing the loss of the model.

4.5.2. LSTM results

It is evident from the comparison table that the accuracy, precision, and recall of the MLP model are better than the other algorithm in determining the attack in the network. The accuracy, precision, recall, and f1 score of the MLP model provide optimal results as compared to the LSTM model.

However, LSTM also provides good results. LSTM usually works well with sequential or temporal data, whereas MLP works well with numerical data.

Figure 12 represents the Accuracy curve of the LSTM model. It is concluded from the figure that with the increase in number of epochs the accuracy of the model gets increased. High number of epochs means high accuracy but very large number of epochs may lead to the over fitting problem while training Deep Learning models.

Figure 13 represents the loss curve of the LSTM model. It is concluded from the figure that with the increase in number of epochs the loss of the model gets decreased. High number of epochs means low loss of the model. LSTM makes use of epochs in order to train the model. High number of epochs sometime leads to the model overfitting.

Table 4 represents the overall accuracies of the deep learning models.it is concluded from the figure that MLP achieves an accuracy of 99.2% and the accuracy achieved by the LSTM model is 99%. Deep Learning models are also considered as effective for attack and intrusion detection tasks. The accuracy is the main predictor to determine the performance of ML and DL models. DL algorithms makes use of activation functions in order to train the model.

Figure 12
LSTM accuracy curve

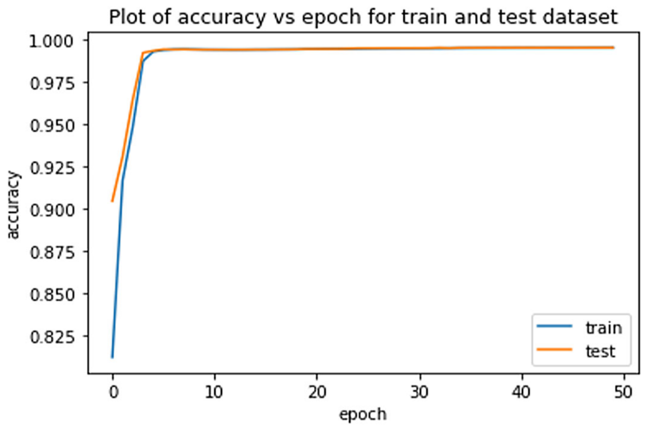


Figure 13
LSTM loss curve

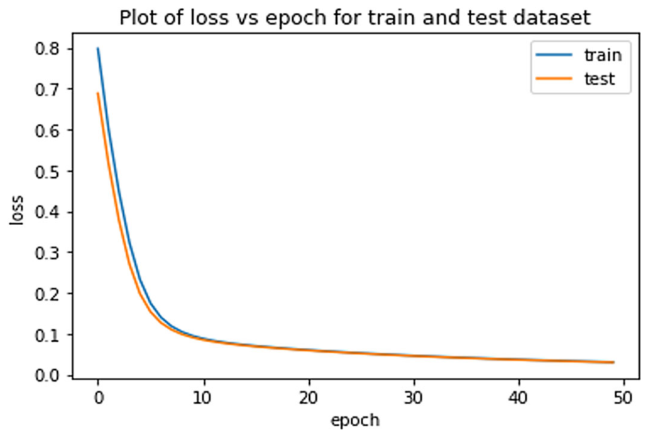


Table 4
Deep learning accuracy

Algorithm	Accuracy	Precision	Recall	F1 score
MLP	99.2%	99%	99.4%	98%
LSTM	99%	99%	99.4%	99%

5. Conclusions

In this research, ML techniques are used for the detection of intrusion in computer networks. TON_IoT dataset is used in this research for IDSs. Pearson's correlation coefficient feature selection technique is used for efficient feature selection technique. Several ML algorithms are applied to data like decision tree, AdaBoost, and KKN. These algorithms are evaluated on the basis of accuracy, precision, recall, and ROC curve. The accuracy achieved by decision tree on the TON_IoT dataset is nearly 99.6% followed by AdaBoost which is also near 99.8%. KNN achieves an accuracy of 99. From this research, we concluded that the use of ML algorithms for IDS is optimal and ML techniques provide accurate results with very less false-positive rates and false-negative rates. DL techniques are also being applied to the two datasets. The results obtained on the TON_IoT dataset are optimal and accurate. MLP obtained an accuracy of nearly 99.2% on the TON_IoT dataset, whereas LSTM obtained 99% on TON_IoT. It is evident from the results that the decision tree for ML and MLP and LSTM for DL provide accurate optimal results.

ML provides efficient and accurate techniques for detecting intrusion in the network. The algorithms like decision tree, KNN, and MLP provide good results along with accuracy. So, we can say that ML provides a good basis for intrusion detection in the network. Moreover, proposed model could be implemented for the detection of unknown attacks in the network in real time.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Informed Consent

Informed consent was obtained from all individual participants included in the study.

References

Ahmed, H., Elsayed, G., Chaffar, S., & Belhaouari, S. B. (2020). A two-level deep learning approach for emotion recognition in Arabic news headlines. *International Journal of Computers and Applications*, *44*(7), 604–613, https://doi.org/10.1080/1206212X.2020.1851501.

Albulayhi, K., Abu Al-Haija, Q., Alsuhibany, S. A., Jillepalli, A. A., Ashrafuzzaman, M., & Sheldon, F. T. (2022). IoT intrusion detection using machine learning with a novel high performing feature selection method. *Applied Sciences*, *12*(10), 5015, https://doi.org/10.3390/app12105015.

Alkhatib, N., Ghauch, H., & Danger, J. L. (2021). SOME/IP intrusion detection using deep learning-based sequential models in automotive ethernet networks. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference*, 2021, 0954–0962.

Ashiku, L., & Dagli, C. (2021). Network intrusion detection system using deep learning. *Procedia Computer Science*, *185*, 239–247, doi.org/10.1016/j.procs.2021.05.025.

Bashir, U., & Chachoo, M. (2014). Intrusion detection and prevention system: Challenges & opportunities. In *2014 International Conference on Computing for Sustainable Global Development*, 806–809, https://doi.org/10.1109/IndiaCom.2014.6828073.

Devarakonda, A., Sharma, N., Saha, P., & Ramya, S. (2022). Network intrusion detection: A comparative study of four classifiers using the NSL-KDD and KDD'99 datasets. In *Journal of Physics: Conference Series*, *2161*, 012043, https://doi.org/10.1088/1742-6596/2161/1/012043.

Faker, O., & Dogdu, E. (2019). Intrusion detection using big data and deep learning techniques. In *Proceedings of the 2019 ACM Southeast Conference*, 86–93, https://doi.org/10.1145/3299815.3314439.

Fu, Y., Du, Y., Cao, Z., Li, Q., & Xiang, W. (2022). A deep learning model for network intrusion detection with imbalanced data. *Electronics*, *11*(6), 898, https://doi.org/10.3390/electronics11060898.

Gadze, J. D., Bamfo-Asante, A. A., Agyemang, J. O., Nunoo-Mensah, H., & Opare, K. A. B. (2021). An investigation into the application of deep learning in the detection and mitigation of DDOS attack on SDN controllers. *Technologies*, *9*, *14*, https://doi.org/10.3390/technologies9010014.

Gao, X., Shan, C., Hu, C., Niu, Z., & Liu, Z. (2019). An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, *7*, 82512–82521. https://doi.org/10.1109/ACCESS.2019.2923640.

Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). Machine learning and deep learning approaches for cybersecurity: A review. *IEEE Access*, *10*, 19572–19585. https://doi.org/10.1109/ACCESS.2022.3151248.

Istiaque, S. M., Khan, A. I., Al Hassan, Z., & Waheed, S. (2021). Performance evaluation of a smart intrusion detection system (IDS) model. *European Journal of Engineering and Technology Research*, *6*(2), 148–152, https://doi.org/10.24018/ejers.2021.6.2.2371.

Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, *41*(4), 1690–1700.

Kim, T., & Pak, W. (2022). Robust network intrusion detection system based on machine-learning with early classification. *IEEE Access*, *10*, 10754–10767. https://doi.org/10.1109/ACCESS.2022.3145002.

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access*, *9*, 22351–22370. https://doi.org/10.1109/ACCESS.2021.3056614.

Paper, C., Javaid, A. Y., & Sun, W. (2016). A deep learning approach for network intrusion detection system. Security and Safety, 3, e2. https://doi.org/10.4108/eai.3-12-2015.2262516.

Park, D., Kim, S., & Kwon, H. (2021). Host-based intrusion detection model using siamese network. *IEEE Access*, *9*, 76614–76623. https://doi.org/10.1109/ACCESS.2021.3082160.

Raghunath, B. R., & Mahadeo, S. N. (2008). Network intrusion detection system (NIDS). In *2008 First International Conference on Emerging Trends in Engineering and Technology*, 1272–1277, https://doi.org/10.1109/ICETET.2008.252.

Rajagopal, S., Kundapur, P. P., & Hareesha, K. S. (2021). Towards effective network intrusion detection: From concept to creation on Azure Cloud. *IEEE Access*, *9*, 19723–19742. https://doi.org/10.1109/ACCESS.2021.3054688.

Ring IV, J. H., Van Oort, C. M., Durst, S., White, V., Near, J. P., & Skalka, C. (2021). Methods for host-based intrusion detection with deep learning. *Digital Threats: Research and Practice*, *2*, 1–29, https://doi.org/10.1145/3461462.

Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. A. (2020). Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Computer Science*, *171*, 1251–1260, https://doi.org/10.1016/j.procs.2020.04.133.

Shapoorifard, H. (2017). Intrusion detection using a novel hybrid method incorporating an improved KNN. *International Journal of Computer Applications*, *173*(1), 5–9. https://doi.org/10.5120/ijca2017914340.

Tang, C., Luktarhan, N., & Zhao, Y. (2020). SAAE-DNN: Deep learning method on intrusion detection. *Symmetry*, *12*, 1–20. https://doi.org/10.3390/sym12101695.

Tang, Y. (2022). Deep stacking network for intrusion detection. *Sensors*, *22*(1), 25. https://doi.org/10.3390/s22010025.

Tao, P., Sun, Z. H. E., & Sun, Z. (2018). An improved intrusion detection algorithm based on GA and SVM. *IEEE Access*, *6*, 13624–13631. https://doi.org/10.1109/ACCESS.2018.2810198.

Ullah, S., Khan, M. A., Ahmad, J., Jamal, S. S., e Huma, Z., Hassan, M. T., . . . & Buchanan, W. J. (2022). HDL-IDS: A hybrid deep learning architecture for intrusion detection in the internet of vehicles. *Sensors*, *22*(4), 1340, https://doi.org/10.3390/s22041340.

Vladimir, V. F. (1967). Candirenggo. *Gastronomía Ecuatoriana y Turismo Local*, *1*, 5–24.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334.

Wang, M. Z., Wang, Y. J., Wang, T. Y., Hou, L. Z., & Li, M. (2020). New approach for information security evaluation and management of IT systems in educational institutions. *Journal of Shanghai Jiaotong University (Science), 25*(6), 689. https://doi.org/10.1007/s12204-020-2231-y.

Zhang, Y., & Ran, X. (2021). A step-based deep learning approach for network intrusion detection. *Computer Modeling in Engineering and Science*, *128*(3), 1231–1245. https://doi.org/10.32604/cmes.2021.016866.

Zhao, G., Zhang, C., & Zheng, L. (2017). Intrusion detection using deep belief network and probabilistic neural network. In *2017 IEEE International Conference on Computational Science and Engineering and IEEE International Conference on Embedded and Ubiquitous Computing*, *1*, 639–642, https://doi.org/10.1109/CSE-EUC.2017.119.

How to Cite: Hidayat, I., Zulfiqar Ali, M., & Arshad, A. (2023). Machine Learning-Based Intrusion Detection System: An Experimental Comparison. *Journal of Computational and Cognitive Engineering* 2(2), 88–97, https://doi.org/10.47852/bonviewJCCE2202270