# ENHANCING IIOT SECURITY WITH MACHINE LEARNING AND DEEP LEARNING FOR INTRUSION DETECTION

***Omer Fawzi Awad[1,2*], Layth Rafea Hazim[1,2], Abdulrahman Ahmed Jasim[1,3], and Oguz Ata[1]***

[1]Electrical and Computer Engineering Department, Altinbas University, Istanbul, Turkey
[2]Computer Science Department, Computer Science and Mathematics College, Tikrit University, Iraq
[3]Collage of Engineering, Al-Iraqia University, Baghdad, Iraq

Emails: omer.fawzi@tu.edu.iq[1*], layth.r.hazim@tu.edu.iq[2], abdulrahman.alsalmany@aliraqia.edu.iq[3], oguz.ata@altinbas.edu.tr[4]

***ABSTRACT***

*The rapid growth of the Internet of Things (IoT) and digital industrial devices has significantly impacted various aspects of life, underscoring the importance of the Industrial Internet of Things (IIoT). Given its importance in industrial contexts that affect human life, the IIoT represents a key subset of the broader IoT landscape. Due to the proliferation of sensors in smart devices, which are viewed as points of contact, as the gathering of data and information regarding the IIoT systems and devices operating on the IoT, there is an urgent requirement for developing effective security methods to counter such threats as well as protecting IIoT systems. In this study, we develop and evaluate a well-optimized intrusion detection system (IDS) based on deep learning (DL) and machine learning (ML) techniques to enhance IIoT security. Leveraging the Edge-IIoTset dataset, specifically designed for IIoT cybersecurity evaluations, we focus on detecting and mitigating 14 distinct attack types targeting IIoT and IoT protocols. These attacks are categorized into five threat groups: information collection, malware, DDoS, man-in-the-middle attacks, and injection attacks. We conducted experiments using machine learning algorithms (k-nearest neighbors, decision tree) and a neural network on the KNIME platform, achieving a remarkable 100% accuracy with the decision tree model. This high accuracy demonstrates the effectiveness of our approach in protecting industrial IoT networks.*

***Keywords: IIoT; Cybersecurity; Intrusion Detection System; Machine Learning; KNIME; Deep Learning.***

## 1.0 INTRODUCTION

Living in the era of AI and information collection technologies, the Internet of Things (IoT) has become an indispensable part of our daily lives, impacting various sectors including residential, industrial, smart cities, healthcare, and beyond [1]. According to the IoT Analytics report, the number of connected IoT devices reached 14.4 billion in 2022, and it is projected to exceed 27 billion by 2025 [2]. The IoT ecosystem encompasses networks, devices, and physical objects interconnected online, interacting with the environment both internally and externally. This connectivity facilitates data collection and analysis through wireless communication, driving applications in home automation, smart energy, industrial automation, and environmental monitoring [3].

Applications of the Industrial Internet of Things (IIoT) bridge the gap between physical and informational technology with the goal of improving productivity and efficiency in industrial processes. But as IPv4 to IPv6 protocols have made it easier to link things to the Internet of Things, worries about network security have grown [4]. IIoT devices are vulnerable to cyberattacks that jeopardize the confidentiality, integrity, and availability of data because of their constrained power, storage, and computational capabilities [5].

Intrusion detection systems (IDS) are essential for keeping an eye on and stopping illegal network activity in order to reduce these dangers. IDS systems can be divided into two categories: host-based (HIDS) and network-based (NIDS), each of which performs particular monitoring tasks [6]. Intelligent threat detection systems (IDS) that can precisely identify and categorize cyber threats in real-time have been made possible by recent developments in machine learning (ML) and deep learning (DL) [7].

This study focuses on enhancing IIoT security by developing and evaluating an optimized IDS leveraging ML and DL techniques. Specifically, we aim to detect and mitigate a diverse range of cyber threats targeting IIoT and IoT protocols using advanced algorithms. The experimental framework includes machine learning models such as k-nearest neighbors (KNN), neural networks (NN), and decision trees (DT), integrated with feature reduction and data cleaning techniques to improve classification accuracy. Through comprehensive performance analysis and comparisons with prior research, we seek to contribute valuable insights to the field of IIoT security.

## 2.0 LITERATURE REVIEW

Many studies were investigating the IoT and IDS fields in order to develop the new security mechanism. This section covered various IDSs which have lately been put forth; Table 1 lists the IIoT and IoT datasets that are now accessible for cyber security. The Edge-IIoTset dataset, which is divided into two categories for DL and ML and for a comprehensive cybersecurity dataset related to IIoT and IoT which could be utilized by ML depending on IDSs, was utilized by M. A. Ferrag et al. [8]. For classifying 14 attacks with the use of three centralized models, they suggest 61 characteristics with a high correlation of 1176 features utilizing RF, DT, KNN, SVM, and DNN: Lastly, the best ACC in 2 classes with SVM, RF, DNN, and KNN. 15 classes with the best ACC 94.67 of DNN; 6 classes with the best ACC 96.01 in additionally DNN. According to Baich et al. [9], the minimum prediction time is 0.4 and the ACC is 99.26 when utilizing ML for supporting and validating the suggested state of the art, a comparative study between two feature selection methods, and feature intrusion detection for this case in binary class and multiclass (5 classes: DOS, normal, U2R, Probe, and R2L). The "DeL-IoT" is a software-defined networking (SDN)-based IDS, according to Tsogbaatar et al. [10]. Deep and stack autoencoders were used for extracting significant features. The projected model demonstrated greater accuracy in identifying attacks, with an F-score range of (99.5-99.9%) and accuracy of (91.04-99.95%). In order to reduce a few information leakages for testing data, Yakub Y. et al. [11] proposed applying ML supervised algorithms depending on IDSs to UNSW-NB15 datasets. PCA could after that be used for performing dimensional reduction.

Zhang et al. [12] proposed a deep belief network (DBN), a genetic algorithm (GA) built IDS, demonstrating that DBN intrusion detection models could detect intrusions more effectively through adaptably increasing the number of hidden layers and neurons throughout several iterations. A distributed IDS with a dependable architecture is recommended by Prazeres N, Costa R, Santos L, et al. [13] for use in fog computing. They created models (RF and NB for IoT-23 and LR and DT for MQTT-IoT-IDS2020) using IoT-23 and MQTT-IoT-IDS2020 datasets, which they then contrasted with three IoT-flow IDS architectures for evaluating the terms precision, recall, and F1-score. Ge et al. [14] proposed a DL approach for identifying IoT cyber risks that uses feed-forward networks to distinguish between various intrusions. This made use of a feed-forward neural network model (FNN) with layering that uses sampling-truncated normalization and initialized randomized weights. Milon Islam M., Hasan M., Ishrak Islam Zarif M., et al. [15] used three ML algorithms (LR, SVM, DT, RF), also DL algorithms including ANN, for finding the best performance in terms of precision, accuracy, F1-score, recall, and the area under the receiver operating characteristic curve (ROC curve). The best accuracy was found to be 99.4% for RF, DT, and ANN.

Using two datasets (Bot-IoT and CSE-CIC-IDS2018), Maglaras L, Ferrag M, Moschoyiannis S, et al. [16] proposed analyzing seven DL models (deep neural networks (DNN)s, recurrent neural networks (RNNs), deep belief networks, restricted Boltzmann machines, deep Boltzmann machines, convolutional neural networks (CNNs), and deep autoencoders) for each model for performing multiclass and binary classification and used these datasets for performing best accuracy, false alarm, and detected rate through assessing such approaches' efficiency. Both DNNs and RNNs get the highest accuracy in CSE-CIC-IDS2018 dataset; the CNN achieves an accuracy of 97.376% in the case when there are 100 hidden nodes and a learning rate of 0.5. Deep auto-encoders achieve a greater accuracy of 98.394% in the case when there are 100 hidden nodes and the learning rate is 0.5, according to the accuracy as well as training time regarding generative and unsupervised models in Bot-IoT dataset. In this paper, Ahmad J., Ullah S., Khan M., et al. [17] suggest a deep convolutional neural network (DCNN) depending on the IDS which comprises of three fully connected dense layers and two convolutional layers. Additionally, they used IoT20 datasets to carry out the experiment. The suggested model's analytical performance depends on high levels of precision, accuracy, recall, and F1-score.

This model employs the following methods: AdaMax, Adam, and Nadm. Lastly, the study's proposal contrasts several sorts of cutting-edge DL and ML approaches. Investigations reveal that the proposed method is more precise compared to three ML-based conventional intrusion detection networks. Along with detecting DDoS attacks, Aamir et al. [18] developed a semi-supervised intrusion detection model depending on PCA which made use of the subset benchmark dataset with new attack vectors, utilized clustering for labeling the data as well as obtaining the classes for distinguishing the attackers for normal traffic, and lastly utilized three ML algorithms (SVM, k-NN, and RF) following labeling for obtaining the results with accuracy (92%, 95%, and finally 96.66%). Hara et al. [19] suggested an autonomous encoding intrusion detection method depending on semi-supervised learning. The 2-class classification was found to be more accurate compared to an intrusion detection model using DNN in simulated trials on an NSL-KDD dataset. By Pacheco et al. [20], a ML attack detection system was introduced. The model shows good accuracy for IIoT cyber-attack intrusion detection.

Table 1: Available IoT and IIoT Datasets for Cyber Security

| Dataset | Description | Features | ML & DL Technique | Threats | Traffic | |
|---|---|---|---|---|---|---|
| | | | | | IoT | IIoT |

| | | | | | | |
|---|---|---|---|---|---|---|
| Edge-IIoTset [8] | This dataset is brand-new for IoT and IIoT cybersecurity applications that use it to test ML based IDSs. | 61 | RF, DT, KNN, SVM, and DNN | 14 | Yes | Yes |
| NSL-KDD [9] | Which was suggested for addressing some of KDD'99 datasets idea, and it can be classified in tow categorize binary and multiclass classification and machine learning and based on intrusion detection systems | 41 | DT, RF, NB, and SVM | 5 | Yes | No |
| N-BaIoT [10] | This dataset depends on network-based detection regarding IoT Botnet attacks using deep autoencoders, and it is gathered from the port mirror of IoT devices. | 115 | Deep ensemble of PNN, DAE, SAE | 10 | Yes | Yes |
| UNSW-NB15 [11] | Is a network intrusion detection, this dataset contains raw of networks packets, this dataset developed by IXIA tool to extract modern, and behavior conducted by ACSC. | 49 | KNN, SVM, QDA and NB | 9 | Yes | No |
| NSL-KDD [12] | Which was suggested for addressing some of KDD'99 datasets idea, and it can be classified in tow categorize binary and multiclass classification and machine learning and based on intrusion detection systems | 41 | FC-ANN, TANN, SA-DA-SNMS, BPNN and GA-DBN | 17 | Yes | No |
| IoT-23 and MQTT-IoT-IDS2020 [13] | Real dataset not simulated used to produce benign traffic and networks attacks that can be based on known the botnet. | 23 | RF, NB for IoT-23 and LR with LR and DT for MQTT-IoT-IDS2020 | 2 | Yes | No |
| BoT-IoT[14] | Is a legitimation and malicious traffic datasets that can simulated IoT devices that can testbed including attacks and targeted of virtual machines with network devices. | 43 | Feed-forward Neural Network (FNN) | 10 | Yes | Yes |
| DS2OStraffi traces[12], [15] | This dataset includes traces that were recorded in the DS2OS IoT environment. They are considerably distinct from traditional network traces because they come from the application layer. | 13 | LR, SVM, DT, RF and ANN | 7 | Yes | No |
| CSE-CIC-IDS2018 and Bot-IoT[16] | Seven different attacks, including Brute-force, Heartbleed, DDoS, DoS, botnets, web attacks, and infiltration, are included in these datasets. The CIC Flow Meter tool is utilized for extracting 80 network flow features from the generated network traffic, same like it was done with the CICDS2017 dataset. | - | RNN, DNN, deep belief networks, restricted Boltzmann machines, CNNs, deep autoencoders, and deep Boltzmann machines | 14 | Yes | Yes |
| IoTID20[17] | These datasets are developed to identification cyberattacks for IoT networks devices and generated in smart home network that can provide normal and anomaly networks flow, the advantage of this dataset it includes modern communicated data on networking interference detection and contain binary, categories and subcategories of labeled. | 83 | Deep Convolutional Neural Network (DCNN) | 4 | Yes | Yes |
| CICIDS2017 [18] | the dataset contains common, recent, benign attacks that closely mimic true real-world data (PCAPs). It also contains the outcomes of CIC-Flow Meter network traffic analysis, labeled flows depending on the time stamp, source and destination IP addresses, source and destination ports, protocols, and attacks (CSV files). The | 84 | Clustering to labeled classes, kNN, SVM and RF | 17 | No | No |

| | | | | | | |
|---|---|---|---|---|---|---|
| | definition of extracted characteristics is additionally available. | | | | | |
| NSL-KDD [19] | Which was suggested for addressing some of KDD'99 datasets idea, and it can be classified in tow categorize binary and multiclass classification and machine learning and based on intrusion detection systems | 41 | AAE and DNN | 17 | No | No |

## 3.0 METHODOLOGY

To give a summary of the experimental findings, the suggested method for conducting classification as well as detection tasks is presented in this part. With the use of DT, k-NN, and NN algorithms, we conducted two experiments: one for the attack label (binary classification), and the other for the attack type. DT has an effective core architecture depending on binary classification, which is utilized as the foundation for the suggested approach to show how well (DT) models work. We have created the next methods for such three types of models for comparing the performance and accuracy of our suggested model: the structure of DL and ML algorithms is depicted in Fig. 1.

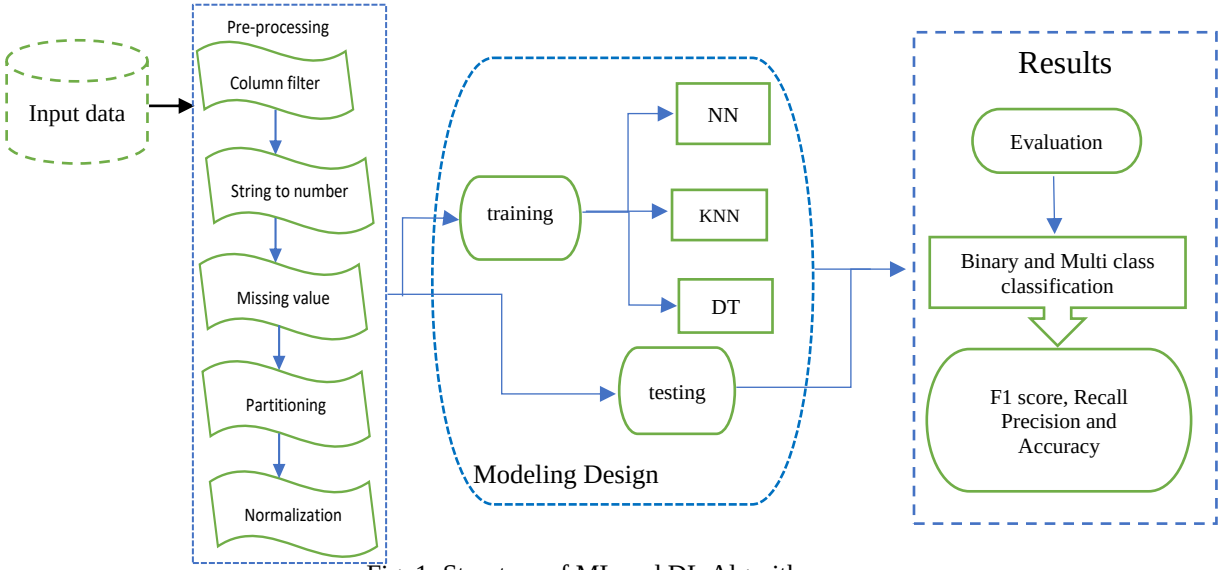



Fig. 1: Structure of ML and DL Algorithms

### 3.1 Dataset

The Edge-IIoTset dataset was introduced by Ferrag et al. [20]. A new cybersecurity dataset for IoT and IIoT applications, the Edge_IIoT dataset was developed in 2022 with IIoT environments in mind as a realistic cyber security dataset. It has 14 different attack types and 62 features. This dataset [20] contains 14 different attacks against IIoT and IoT protocols, which are divided into five different threats: information collection, malware, DDoS, man-in-the-middle attacks, and injection attacks. 61 of the 1176 features have a substantial link with one another. A total of 2,219,201 instances, 1,615,643 of which are regarded as regular, and 603,558 of which are regarded as attacks, are reported by Edge_IIoT [20]. The distribution of instances for Edge-IIoTset dataset is shown in Fig. 2. We utilized a stratification option to ensure that the percentages were the same across all classes, and we kept 80% of the sample for training and used 20% of it for testing. A few features from Edge_IIoT dataset were removed throughout pre-processing; however, such features had no impact on the accuracy regarding the result. High accuracy depends on two factors: (1) the classes should be equally important; and (2) the dataset should be balanced. Depending on such conditions, several preprocessing processes should be carried out.

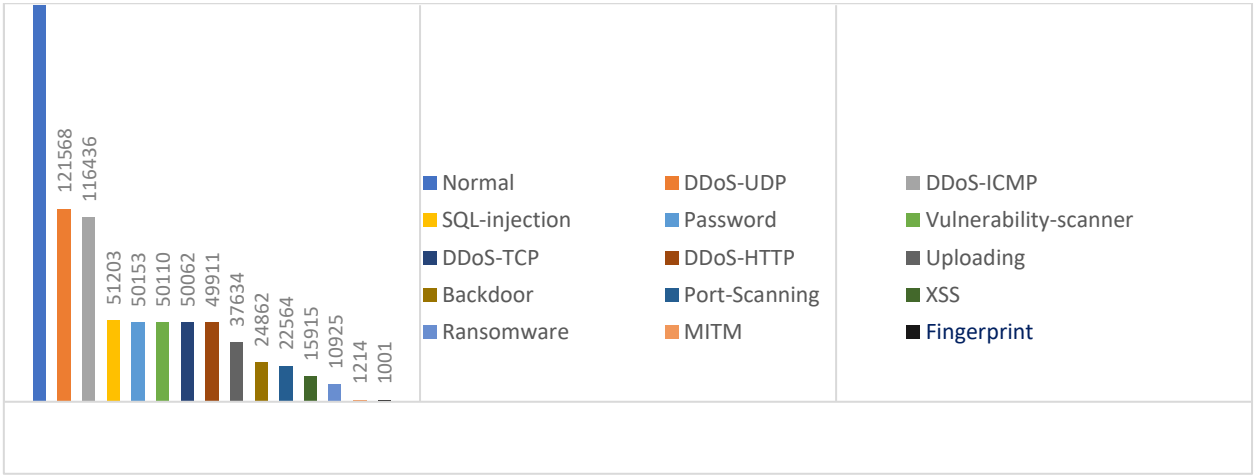



Fig. 2: Distribution of Instance for Edge_IIoT Dataset

### 3.2 KNIME

Users can combine, access, analyze, and visualize data using Konstanz Information Miner, an open-source software platform for data integration and data science. With regard to experienced users, its low-code, no-code interface provides a sophisticated suite of data science tools [18]. Workflows are built using desktop-based KNIME Analytics Platform by developers and analysts [19]. KNIME Server is business software created for process automation, group collaboration, management, and deployment. Here are some of the benefits of using KNIME: Ease of use: KNIME has a user-friendly interface that makes it easy to get started with data science. Powerful features: KNIME offers a wide range of features for data science, including data wrangling, machine learning, and data visualization. Open source: KNIME is free to modify and use because it is open-source software. Community: KNIME has a large and active community of users and developers who are constantly sharing tips and tricks. Here are some of the drawbacks of using KNIME: (1) Learning curve: KNIME has a steep learning curve, which can be a challenge for beginners. (2) Complexity: KNIME can be complex software, which can be a challenge for some users. (3) Dependency on third-party tools: KNIME relies on several third-party tools, such as R and Python, which can be a challenge for some users. Overall, (4) KNIME is a powerful and versatile data science platform that is suitable for a wide range of users. Yet, it is of high importance to be aware of the learning curve and complexity of the software before you decide to use it. Here are some of the things you can do with KNIME: Data wrangling: KNIME can be used to clean, transform, and prepare data for analysis. Machine learning: KNIME can be used to build and train machine learning models. Data visualization: KNIME can be used to create interactive data visualizations. Deployment: KNIME can be used to deploy data science models and workflows.

### 3.3 Pre-processing

A critical stage in creating a ML model is data pre-processing. To ensure that we develop our DL and ML models without any problems, we should do a few preparations. Our ML and DL models won't function effectively without data pre-processing. There are certain techniques that leverage nodes in the KNIME platform to improve accuracy while building a classification model for intrusion detection depending on the combination of k-NN, NN, and DT. Internal rules in the original data information are broken, which results in subpar data processing and analysis. As a result, it is necessary to clean "dirty data" and turn it into data that meets the standards for data quality. The main challenges of data cleaning include missing values, inaccurate data, distortion, and misfits. The answer is removing or replacing the current special symbols and use the same constant for filling the missing data. We categorize the procedures that are used with pre-processing into two parts depending on the format of the datasets [20]. For binary classification pre-processing, filter some features. In the Edge_IIoT dataset, the features are considered very important in order not to affect the accuracy of the results in terms of training and testing conducted by the trained algorithms. Some features (ip.src_host, frame.time, http.tls_port, ip.dst_host, and tcp.payload) were dropped during the pre-processing stage because data is an imbalance with regard to normal class and to reduce the bias during training. For multiclass classification, we filter some features from the Edge_IIoT dataset, and here we drop the same features as in binary classification, but the difference is that in multiclass classification we drop the Attack label feature and still the Attack type feature. String to number: through converting strings in a column (or set of columns) to numbers using wildcard selection, all features, and exclude (Attack type feature), we are parsing a few options which are utilized on this node in this instance. Missing Value Node: this node assists in handling missing values discovered in input table cells. In the case when a double missing value is displayed, choose a fixed value for the missing value and enter (0.1) in it. By doing this, you can

make all the values almost equal without impacting the accuracy of the results which are displayed, and when the mean yields an integer value, The input table's partitioning node is divided into two partitions which are allocated 20% for testing and 80% for training. Both the two partitions and the split dataset (1,775,360 for training and 443841 for testing) are accessible at the two output ports. The Min-Max scaler is the final preprocessing step that normalizes the input features and variables for this node (aside from the standardization step that adjusts the features to have values between 0 and 1). This will transform all features into the range [0, 1], where the minimum and maximum values of each feature or variable will be 0 and 1, respectively. Averaging, minmax scaling, and standard scaling are some of the methods for normalizing data. In this study, the data were normalized using a standard scalar. A standard scalar utilizes the standard normal distribution (SND), thus its mean and variance are 0 and 1, respectively. We have 41 features following applying one hot encoding; to standardize the feature space, we employed a standard scalar. It could be modeled mathematically as Eq 1.

$$z = \frac{x - \mu}{sd} \quad (1)$$

Here, $z$ denotes the standard feature space regarding $x$ input data samples, $\mu$ represent the mean, and $sd$ represent the standard deviation. The mathematical representation of the mean is Eq. 2.

$$z = \frac{1}{N}\sum_{i=1}^{N}(x)_i \quad (2)$$

and standard deviation is Eq. 3.

$$sd = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \quad (3)$$

Here $x\ i$ represent the input sample. The preprocessing structure is the same for multiclass and binary classification, yet the rule engine node is utilized in a different step of DL algorithm. This node "takes a list of user defined rules and attempts to match them to each one of the rows in the input table, if a rule matches its outcome value added into a new column and the first matching rule of order to definition determine of outcome". The second step is to create a collection of numbers by aggregating numbers to put the column in set, which could after that be safely split into the original column content. The name of attacks feature is after that converted to a number by being utilized and replaced with the column of attack type to outcome column number.
For fine-tuning parameters, it is tried for finding the best parameters for algorithms. Practically, the first model utilized to find the best accuracy is the k-NN algorithm, where only the numeric column is utilized and the Euclidean distance is implemented, and the test data is also forward as the output. The k-Nearest Neighbor (k-NN) is configured by standard settings for the column with class labels (Attack_label), and the number of neighbors is 3. This number is used to classify a new instance, and the weight neighbors for the distance includes the query pattern that can be stored for training patterns into classification of the closer neighbors that have greater influence on the class. Same options for multiclass classification, just the class labels (Attack_type). Fig. 3 Simple architecture of preprocessing for binary and multiclassification.

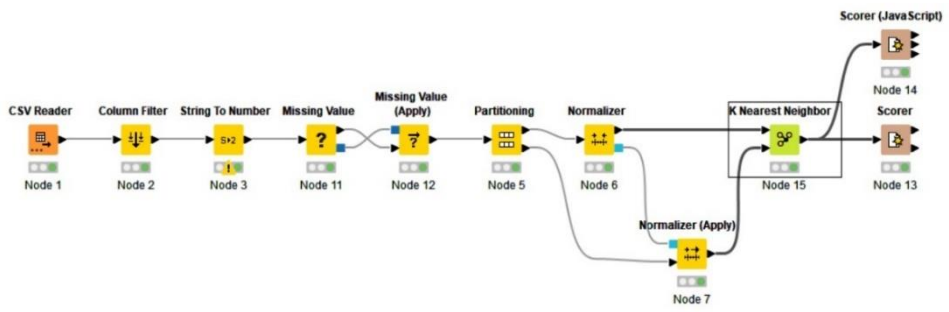



Fig. 3: Simple Architecture of Binary and Multiclass Classification Utilized K-NN Algorithm

With regard to NN in binary classification, we utilized a keras input layer with the following options: batch size (1500), shape (41), a keras dense layer (3), an input hidden layer with units (64, 128, 256), and the last for the output layer. The attack label, which is utilized in Keras Network, is the target feature for the learner employing unit (1) and the activation function (sigmoid) for binary classification. The accuracy of the training procedure is equivalent to 98.26, as can be seen in Fig. (4a), and the loss rate for the Keras log output is 0.0497 with 15 epochs and a batch size of 1184, as shown in Fig. (4b).

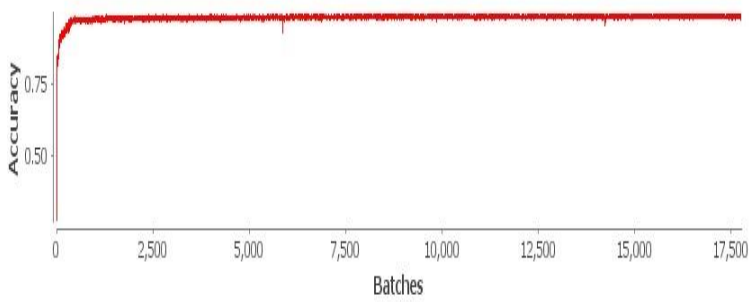



Fig. (4a): Accuracy for Binary Training

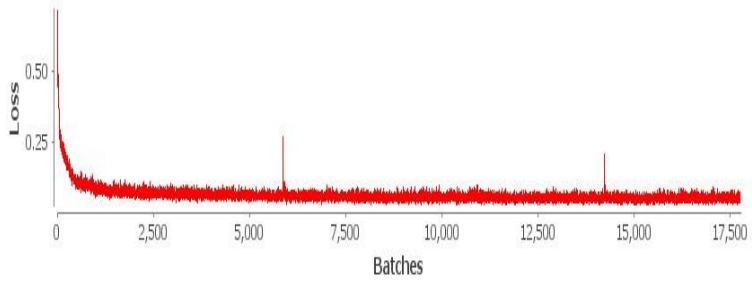



Fig. (4b): Gradient Descent for Binary Training

For multiclass classification, we used a keras input layer where the options are shape (56) and batch size (512), and a keras dense layer (3) for the input hidden layer where units (64, 128, 256) are activated using the activation function (ReLU). ReLU function as well as its derivative are monotonic. Any negative input causes the function to return (0); any positive input causes it to return the value x. The output therefore has a range from 0 to infinite. ReLU is the activation function that NNs employ the most frequently and is the default activation function. The target feature the learner wants to learn is (Attack_type), and Figure 5 depicts the architecture of NN preprocessing. Eq. (4) describes how to obtain the ReLU value, and Eq. (5) indicates how to find the Softmax value for output layer using unit (15) with activation function (Softmax). ReLU function as well as its derivative are both monotonic; if any negative input is given, the function returns (0); if any positive input is given, the function returns x.

$$ReLU\ formula\ is: f(x) = max(0, x)\ (4)$$

$$\sigma(\overrightarrow{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}\ (5)$$

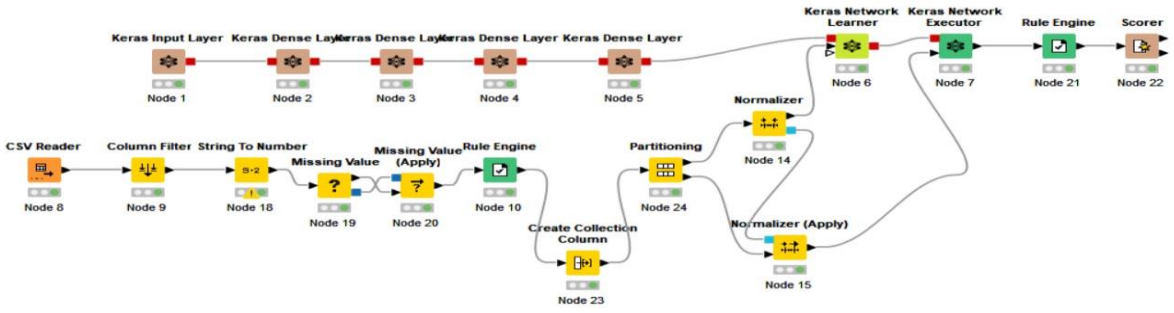



Fig. 5: Simple Architecture of Binary and Multiclass Classification Utilized (NN) Algorithm

The output therefore has a range from 0 to infinite. ReLU is the activation function that NNs employ the most frequently and is the default activation function. The keras network learner for training our utilized number of neurons equals (56) and the shape is (56) and the conversion is "from number (double)"; the target data column is Attack_type; the number of neurons equals (15) and the shape equals (15); the conversion is from collection of number (integer) to one-hot tensor; and the target column is aggregated value. The standard loss function is categorical, see Fig. (6a) learner monitor for Keras network learner to show the accuracy for the training algorithm is equal (92.57); Fig. (6b) loss rate; and the keras log output is for current value percentage (0.23) with (4) epochs and (3468) batch size.

The general settings of the option for Keras Networks learners the back end is keras (tensor flow) with epochs (4), the validation batch size is 512, the training batch size is 512, and the Adam optimizer is utilized with a learning rate equal to 0.001. Decision Tree Learner: the target column is Attack_type, the quality measures are the Gini index, the Minimum Description Length (MDL) pruning approach (2), the number of records (10,000), and the number of threads (12). In Fig. 7, the structure of the DT algorithm is shown. The tree root of the DT can be divided into sub-roots for each instance, and each of these roots takes weight.

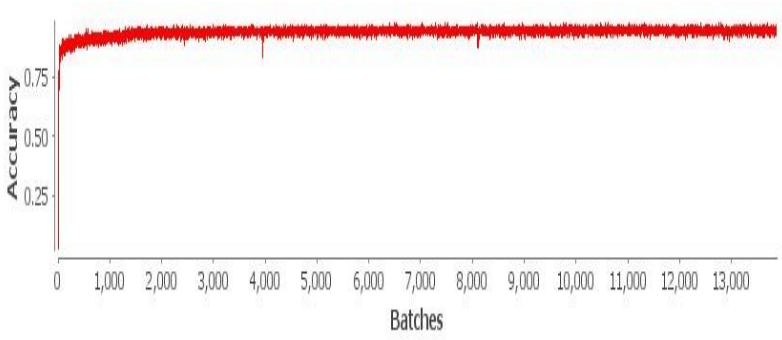



Fig. (6a): Accuracy for Binary Training

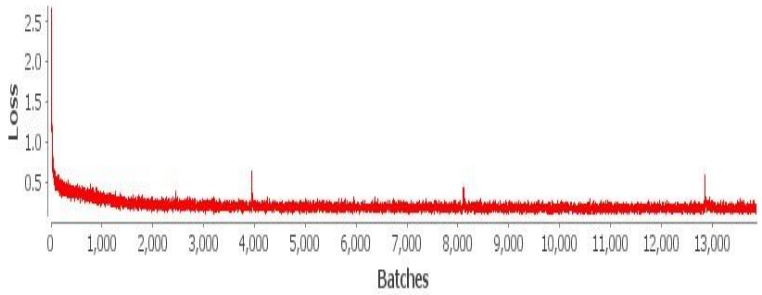



Fig. (6b): Gradient Descent for Binary Training

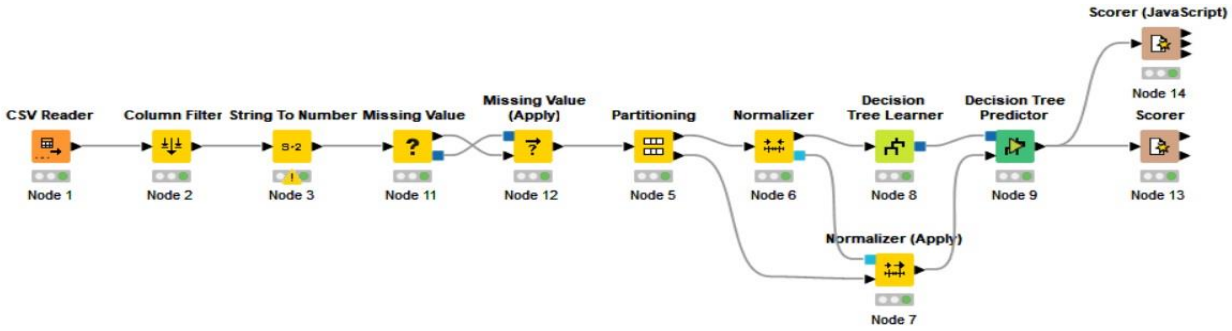



Fig. 7: Architecture of Multiclass Classification Utilized (DT) Algorithm

## 4.0 RESULTS

As described before, k-NN, NN, and DT algorithms are implemented. Then, the mean regarding the results is taken from the precision as Eq. (4), recall as Eq. (5), F-measure as Eq. (6), and accuracy as Eq. (7) of each algorithm.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \quad (7)$$

The experiment used the Edge_IIoT dataset for intrusion detection systems using several classifiers and the following sampling types: random samples with an 80% training and 20% testing dataset size.
For binary classification, the first model can be used in our study, the k-NN algorithm from Fig. 8. As can be seen from the confusion matrix (TP = 321966), FN = 1163, FP = 6576, and TN = 114136), the correct classification is (436102), the wrong classification is (7739), the error rate is (1.74%), Cohen's kappa (k) is (99.5%), and the accuracy is (98.26%).

K Nearest Neighbor

Confusion Matrix

| Rows Number : 443841 | 0 (Predicted) | 1 (Predicted) | |
|---|---|---|---|
| 0 (Actual) | 321966 | 1163 | 99.64% |
| 1 (Actual) | 6576 | 114136 | 94.55% |
| | 98.00% | 98.99% | |

Class Statistics

| Class | True Positives | False Positives | True Negatives | False Negatives | Recall | Precision | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 321966 | 6576 | 114136 | 1163 | 99.64% | 98.00% | 99.64% | 94.55% | 98.81% |
| 1 | 114136 | 1163 | 321966 | 6576 | 94.55% | 98.99% | 94.55% | 99.64% | 96.72% |

Overall Statistics

| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 98.26% | 1.74% | 0.955 | 436102 | 7739 |

Fig. 8: Confusion Matrix of (K-NN) Algorithm

In the second experiment NN, from Fig. 9, we can see the confusion matrix of this model: the results of (TP = 323212), (FN = 374), (FP = 6445), and (TN = 113810), the correct classification (437022), the wrong classification (6819), the error rate (1.54%), Cohen's kappa (k) (96%), and the accuracy (98.46%).

Neural Network

Confusion Matrix

| Rows Number : 443841 | 0 (Predicted) | 1 (Predicted) | |
|---|---|---|---|
| 0 (Actual) | 323212 | 374 | 99.88% |
| 1 (Actual) | 6445 | 113810 | 94.64% |
| | 98.04% | 99.67% | |

Class Statistics

| Class | True Positives | False Positives | True Negatives | False Negatives | Recall | Precision | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 323212 | 6445 | 113810 | 374 | 99.88% | 98.04% | 99.88% | 94.64% | 98.96% |
| 1 | 113810 | 374 | 323212 | 6445 | 94.64% | 99.67% | 94.64% | 99.88% | 97.09% |

Overall Statistics

| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 98.46% | 1.54% | 0.960 | 437022 | 6819 |

Fig. 9: Confusion Matrix of (NN) Algorithm

The final experiment in binary classification DT, from Fig. 10, can be seen from the confusion matrix of this model (TP = 323181), (FN = 0), (FP = 0), and (TN = 120660), the correct classification (443841), the wrong classification ((0)), the error rate (0), Cohen's kappa (k) (100%), and the accuracy (100%).

Decision Tree

Confusion Matrix

| Rows Number : 443841 | 0 (Predicted) | 1 (Predicted) | |
|---|---|---|---|
| 0 (Actual) | 323181 | 0 | 100.00% |
| 1 (Actual) | 0 | 120660 | 100.00% |
| | 100.00% | 100.00% | |

Class Statistics

| Class | True Positives | False Positives | True Negatives | False Negatives | Recall | Precision | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 323181 | 0 | 120660 | 0 | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |
| 1 | 120660 | 0 | 323181 | 0 | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

Overall Statistics

| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 100.00% | 0.00% | 1.000 | 443841 | 0 |

Fig. 10: Confusion Matrix of DT Algorithm

Figure 11 shows the contrast between binary classification techniques which could be used and the outcomes from classifiers like NN, k-NN, and DT from above. With regard to binary classification, DT had a greater classification accuracy of 100%, whilst k-NN had the lowest accuracy of 98.256%. The higher value (100) of the F1-score, which displays the total result of the recall and precision ratios, indicates that a model DT is more capable of making accurate classifications.

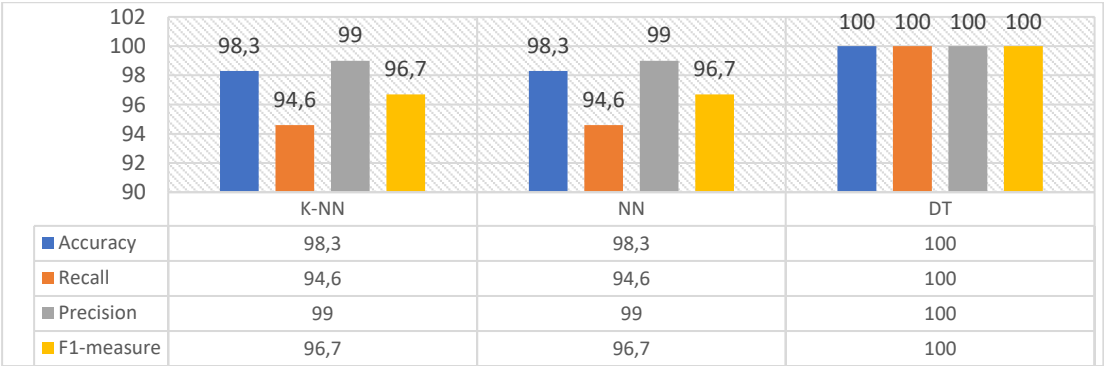



Fig. 11: Comparison Between Three Algorithms for Binary Classification

The second experiment for the multiclass classification used the same models with the same sampling types of random samples with 80% training data size and 20% testing dataset size. We utilized three algorithms: k-NN, NN, and DT. The k-NN algorithm result of the confusion matrix can be seen in Fig. 12. For this experiment, the high accuracy of the relation between the actual and the prediction can be seen on the DDoS-UDP (100%) and on the MITM (100%). The lowest accuracy of the relation can be seen on the XSS, where the percentage of accuracy is 54.37%.

k-NN multi class

Confusion Matrix

| Rows ... | Backd... | DDoS_... | DDoS_... | DDoS_... | DDoS_... | Finger... | MITM (... | Normal... | Passw... | Port_S... | Ranso... | SQL_in... | Upload... | Vulner... | XSS (P... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Backd... | 4792 | 0 | 0 | 15 | 0 | 0 | 0 | 154 | 0 | 41 | 0 | 0 | 9 | 0 | 0 | 95.63% |
| DDoS_... | 0 | 5600 | 0 | 2 | 0 | 0 | 0 | 134 | 942 | 1 | 0 | 2016 | 622 | 265 | 317 | 56.57% |
| DDoS_... | 0 | 0 | 23160 | 0 | 0 | 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 99.83% |
| DDoS_... | 0 | 49 | 0 | 8794 | 0 | 0 | 0 | 1 | 16 | 1164 | 0 | 12 | 2 | 7 | 3 | 87.52% |
| DDoS_... | 0 | 0 | 0 | 0 | 24593 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100.00% |
| Finger... | 0 | 0 | 48 | 0 | 0 | 107 | 0 | 20 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 58.47% |
| MITM (... | 0 | 0 | 0 | 0 | 0 | 0 | 80 | 170 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 32.00% |
| Normal... | 1 | 6 | 0 | 0 | 0 | 1 | 0 | 322624 | 51 | 79 | 0 | 3 | 8 | 0 | 0 | 99.95% |
| Passw... | 0 | 1766 | 0 | 2 | 0 | 0 | 0 | 989 | 2596 | 2 | 0 | 2946 | 698 | 865 | 324 | 25.48% |
| Port_S... | 0 | 0 | 0 | 753 | 0 | 0 | 0 | 399 | 0 | 3312 | 0 | 0 | 0 | 0 | 0 | 74.19% |
| Ranso... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 193 | 1 | 59 | 1841 | 0 | 71 | 0 | 0 | 85.03% |
| SQL_i... | 0 | 1019 | 0 | 0 | 0 | 0 | 0 | 8 | 853 | 1 | 0 | 8398 | 0 | 1 | 0 | 81.69% |
| Upload... | 3 | 943 | 0 | 0 | 0 | 0 | 0 | 204 | 483 | 1 | 31 | 8 | 5427 | 5 | 551 | 70.89% |
| Vulner... | 1 | 61 | 0 | 0 | 0 | 0 | 0 | 192 | 363 | 1 | 0 | 3 | 11 | 9396 | 0 | 93.70% |
| XSS (A... | 1 | 547 | 0 | 0 | 0 | 0 | 0 | 230 | 196 | 5 | 0 | 0 | 700 | 1 | 1424 | 45.88% |
| | 99.87% | 56.05% | 99.79% | 91.93% | 100.00% | 72.79% | 100.00% | 99.17% | 47.19% | 70.86% | 98.34% | 62.74% | 71.90% | 89.15% | 54.37% | |

Fig. 12: Confusion Matrix of (K-NN) Algorithm

Class statistics for this model can be seen on Fig. 13, as these figures show class statistics and overall statistics of the k-NN algorithm that found the percentage of accuracy (95.11%) for this model, the error rate (4.89%), Cohen's kappa (k) (89.3%), correctly classified (422144), and incorrectly classified (21697).

Class Statistics

| Class | True Positives | False Positives | True Negatives | False Negatives | Recall | Precision | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| Backdoor | 4792 | 6 | 438824 | 219 | 95.63% | 99.87% | 95.63% | > 99.99% | 97.71% |
| DDoS_HTTP | 5600 | 4391 | 429551 | 4299 | 56.57% | 56.05% | 56.57% | 98.99% | 56.31% |
| DDoS_ICMP | 23160 | 48 | 420594 | 39 | 99.83% | 99.79% | 99.83% | 99.99% | 99.81% |
| DDoS_TCP | 8794 | 772 | 433021 | 1254 | 87.52% | 91.93% | 87.52% | 99.82% | 89.67% |
| DDoS_UDP | 24593 | 0 | 419248 | 0 | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |
| Fingerprinting | 107 | 40 | 443618 | 76 | 58.47% | 72.79% | 58.47% | 99.99% | 64.85% |
| MITM | 80 | 0 | 443591 | 170 | 32.00% | 100.00% | 32.00% | 100.00% | 48.48% |
| Normal | 322624 | 2694 | 118374 | 149 | 99.95% | 99.17% | 99.95% | 97.77% | 99.56% |
| Password | 2596 | 2905 | 430748 | 7592 | 25.48% | 47.19% | 25.48% | 99.33% | 33.09% |
| Port_Scanning | 3312 | 1362 | 438015 | 1152 | 74.19% | 70.86% | 74.19% | 99.69% | 72.49% |
| Ransomware | 1841 | 31 | 441645 | 324 | 85.03% | 98.34% | 85.03% | 99.99% | 91.21% |
| SQL_injection | 8398 | 4988 | 428573 | 1882 | 81.69% | 62.74% | 81.69% | 98.85% | 70.97% |
| Uploading | 5427 | 2121 | 434064 | 2229 | 70.89% | 71.90% | 70.89% | 99.51% | 71.39% |
| Vulnerability_scanner | 9396 | 1144 | 432669 | 632 | 93.70% | 89.15% | 93.70% | 99.74% | 91.37% |
| XSS | 1424 | 1195 | 439542 | 1680 | 45.88% | 54.37% | 45.88% | 99.73% | 49.76% |

Overall Statistics

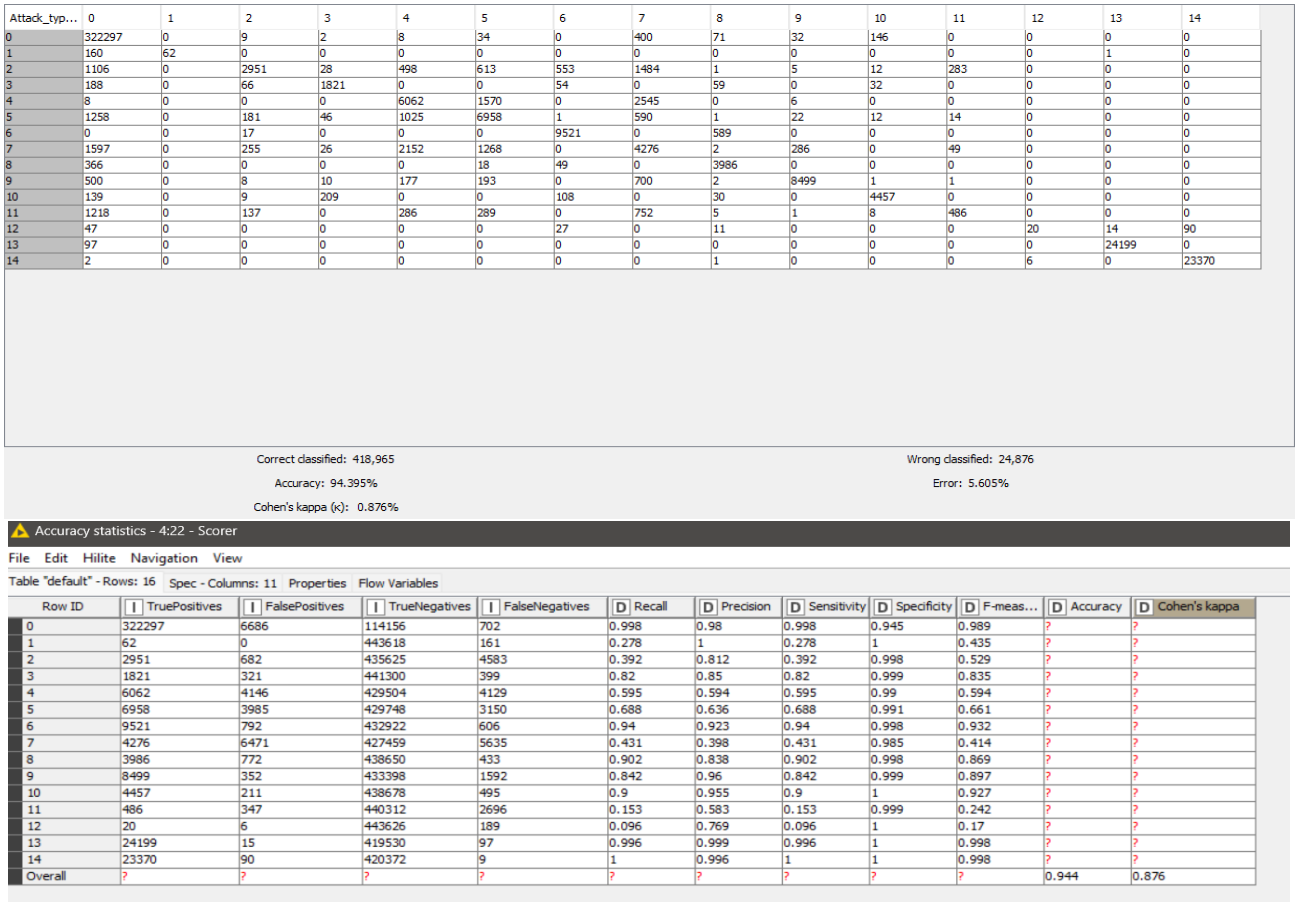
| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 95.11% | 4.89% | 0.893 | 422144 | 21697 |

Fig. 13: Class and Overall Statistics of (K-NN) Algorithm

The NN algorithm is the second experiment. On this experiment, using the rule engine node to convert the classes that contain normal and the names of attacks to be read by the Keras learner node as a number, this is a rule that can be utilized on this node:
$Attack_type$ = "Normal" => 0
$Attack_type$ = "MITM" => 1
$Attack_type$ = "Uploading" => 2
$Attack_type$ = "Ransomware" => 3
$Attack_type$ = "SQL_injection" => 4

$Attack_type$ = "DDoS_HTTP" => 5
$Attack_type$ = "DDoS_TCP" => 6
$Attack_type$ = "Password" => 7
$Attack_type$ = "Port_Scanning" => 8
$Attack_type$ = "Vulnerability_scanner"=> 9
$Attack_type$ = "Backdoor" =>10
$Attack_type$ = "XSS" => 11
$Attack_type$ = "Fingerprinting" =>12
$Attack_type$ = "DDoS_UDP" =>13
$Attack_type$ = "DDoS_ICMP" => 14

The result of the confusion matrix and the accuracy statistics for NN can be seen in Fig. 14. For this experiment, the high (TP) of the relation between actual and prediction can be seen on the number (0=normal) (322297), which means a high percentage of accuracy. The lowest (TP) of the relation between actual and prediction can be seen on the number (12=fingerprinting attack) (20), which means the lowest accuracy of this relation.



| Attack_typ... | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 322297 | 0 | 9 | 2 | 8 | 34 | 0 | 400 | 71 | 32 | 146 | 0 | 0 | 0 | 0 |
| 1 | 160 | 62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 1106 | 0 | 2951 | 28 | 498 | 613 | 553 | 1484 | 1 | 5 | 12 | 283 | 0 | 0 | 0 |
| 3 | 188 | 0 | 66 | 1821 | 0 | 0 | 54 | 0 | 59 | 0 | 32 | 0 | 0 | 0 | 0 |
| 4 | 8 | 0 | 0 | 0 | 6062 | 1570 | 0 | 2545 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1258 | 0 | 181 | 46 | 1025 | 6958 | 1 | 590 | 1 | 22 | 12 | 14 | 0 | 0 | 0 |
| 6 | 0 | 0 | 17 | 0 | 0 | 0 | 9521 | 0 | 589 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1597 | 0 | 255 | 26 | 2152 | 1268 | 0 | 4276 | 2 | 286 | 0 | 49 | 0 | 0 | 0 |
| 8 | 366 | 0 | 0 | 0 | 0 | 18 | 49 | 0 | 3986 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 500 | 0 | 8 | 10 | 177 | 193 | 0 | 700 | 2 | 8499 | 1 | 1 | 0 | 0 | 0 |
| 10 | 139 | 0 | 9 | 209 | 0 | 0 | 108 | 0 | 30 | 0 | 4457 | 0 | 0 | 0 | 0 |
| 11 | 1218 | 0 | 137 | 0 | 286 | 289 | 0 | 752 | 5 | 1 | 8 | 486 | 0 | 0 | 0 |
| 12 | 47 | 0 | 0 | 0 | 0 | 0 | 27 | 0 | 11 | 0 | 0 | 0 | 20 | 14 | 90 |
| 13 | 97 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24199 | 0 |
| 14 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 23370 |

Correct classified: 418,965 Wrong classified: 24,876
Accuracy: 94.395% Error: 5.605%
Cohen's kappa (κ): 0.876%

Accuracy statistics - 4:22 - Scorer

File Edit Hilite Navigation View

Table "default" - Rows: 16 Spec - Columns: 11 Properties Flow Variables

| Row ID | TruePositives | FalsePositives | TrueNegatives | FalseNegatives | Recall | Precision | Sensitivity | Specificity | F-meas... | Accuracy | Cohen's kappa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 322297 | 6686 | 114156 | 702 | 0.998 | 0.98 | 0.998 | 0.945 | 0.989 | ? | ? |
| 1 | 62 | 0 | 443618 | 161 | 0.278 | 1 | 0.278 | 1 | 0.435 | ? | ? |
| 2 | 2951 | 682 | 435625 | 4583 | 0.392 | 0.812 | 0.392 | 0.998 | 0.529 | ? | ? |
| 3 | 1821 | 321 | 441300 | 399 | 0.82 | 0.85 | 0.82 | 0.999 | 0.835 | ? | ? |
| 4 | 6062 | 4146 | 429504 | 4129 | 0.595 | 0.594 | 0.595 | 0.99 | 0.594 | ? | ? |
| 5 | 6958 | 3985 | 429748 | 3150 | 0.688 | 0.636 | 0.688 | 0.991 | 0.661 | ? | ? |
| 6 | 9521 | 792 | 432922 | 606 | 0.94 | 0.923 | 0.94 | 0.998 | 0.932 | ? | ? |
| 7 | 4276 | 6471 | 427459 | 5635 | 0.431 | 0.398 | 0.431 | 0.985 | 0.414 | ? | ? |
| 8 | 3986 | 772 | 438650 | 433 | 0.902 | 0.838 | 0.902 | 0.998 | 0.869 | ? | ? |
| 9 | 8499 | 352 | 433398 | 1592 | 0.842 | 0.96 | 0.842 | 0.999 | 0.897 | ? | ? |
| 10 | 4457 | 211 | 438678 | 495 | 0.9 | 0.955 | 0.9 | 1 | 0.927 | ? | ? |
| 11 | 486 | 347 | 440312 | 2696 | 0.153 | 0.583 | 0.153 | 0.999 | 0.242 | ? | ? |
| 12 | 20 | 6 | 443626 | 189 | 0.096 | 0.769 | 0.096 | 1 | 0.17 | ? | ? |
| 13 | 24199 | 15 | 419530 | 97 | 0.996 | 0.999 | 0.996 | 1 | 0.998 | ? | ? |
| 14 | 23370 | 90 | 420372 | 9 | 1 | 0.996 | 1 | 1 | 0.998 | ? | ? |
| Overall | ? | ? | ? | ? | ? | ? | ? | ? | ? | 0.944 | 0.876 |

Fig. 14: Confusion Matrix and Accuracy Statistics Results for (NN) Model

The final experiment for the multiclass classification utilized DT algorithms. The result of confusion matrix can be seen in Fig. 15. For this experiment, the high accuracy of the relation between the actual and the prediction can be seen on the backdoor, where DDoS_ICMP, DDoS_TCP, DDoS_UDP, and MITM equal (100%), while

the lowest accuracy of the relation can be seen on DDoS_HTTP, where the percentage of accuracy is 83.64%.

DT

Confusion Matrix

| Rows ... | Backdo... | DDoS_... | DDoS_I... | DDoS_... | DDoS_... | Fingerp... | MITM (... | Normal... | Passw... | Port_S... | Ranso... | SQL_in... | Uploadi... | Vulnera... | XSS (P... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Backd... | 4753 | 0 | 0 | 0 | 0 | 0 | 0 | 143 | 0 | 30 | 1 | 0 | 1 | 0 | 0 | 96.45% |
| DDoS_... | 0 | 8450 | 0 | 0 | 0 | 0 | 0 | 1 | 493 | 0 | 0 | 703 | 207 | 36 | 93 | 84.64% |
| DDoS_I... | 0 | 0 | 23231 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | > 99.99% |
| DDoS_... | 0 | 0 | 0 | 9919 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100.00% |
| DDoS_... | 0 | 0 | 0 | 0 | 24247 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100.00% |
| Finger... | 0 | 0 | 0 | 0 | 0 | 147 | 0 | 27 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 79.46% |
| MITM (... | 0 | 0 | 0 | 0 | 0 | 0 | 87 | 166 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 34.39% |
| Normal... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 323121 | 0 | 70 | 0 | 0 | 1 | 0 | 0 | 99.98% |
| Passw... | 0 | 590 | 0 | 0 | 0 | 0 | 0 | 14 | 8317 | 1 | 0 | 585 | 165 | 137 | 82 | 84.09% |
| Port_S... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 384 | 0 | 4085 | 0 | 0 | 0 | 0 | 0 | 91.41% |
| Ranso... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 199 | 0 | 57 | 1893 | 0 | 0 | 0 | 0 | 88.09% |
| SQL_in... | 0 | 707 | 0 | 0 | 0 | 0 | 0 | 8 | 524 | 1 | 0 | 9317 | 0 | 0 | 0 | 88.25% |
| Upload... | 0 | 195 | 0 | 0 | 0 | 0 | 0 | 9 | 125 | 2 | 1 | 0 | 6949 | 0 | 271 | 92.02% |
| Vulner... | 0 | 64 | 0 | 0 | 0 | 0 | 0 | 10 | 127 | 0 | 0 | 0 | 0 | 9874 | 0 | 98.00% |
| XSS (A... | 0 | 97 | 0 | 0 | 0 | 0 | 0 | 155 | 50 | 7 | 0 | 0 | 273 | 1 | 2625 | 81.83% |
| | 100.00% | 83.64% | 100.00% | 100.00% | 100.00% | 98.66% | 100.00% | 99.66% | 86.31% | 95.80% | 99.89% | 87.85% | 91.48% | 98.27% | 85.48% | |

DDoS_HTTP (Actual)

Class Statistics

Fig. 15: Confusion Matrix of (DT) Algorithm

Class statistics for this model can be seen in Fig. 16. This figure shows class statistics and overall statistics of the DT algorithm that found the percentage of accuracy (98.46%) for this model, the error rate (1.54%), Cohen's kappa (k) (96.7%), correctly classified (437015), and incorrectly classified (6826).

Class Statistics

| Class | True Positives | False Positives | True Negatives | False Negatives | Recall | Precision | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| Backdoor | 4753 | 0 | 438913 | 175 | 96.45% | 100.00% | 96.45% | 100.00% | 98.19% |
| DDoS_HTTP | 8450 | 1653 | 432205 | 1533 | 84.64% | 83.64% | 84.64% | 99.62% | 84.14% |
| DDoS_ICMP | 23231 | 0 | 420609 | 1 | > 99.99% | 100.00% | > 99.99% | 100.00% | > 99.99% |
| DDoS_TCP | 9919 | 0 | 433922 | 0 | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |
| DDoS_UDP | 24247 | 0 | 419594 | 0 | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |
| Fingerprinting | 147 | 2 | 443654 | 38 | 79.46% | 98.66% | 79.46% | > 99.99% | 88.02% |
| MITM | 87 | 0 | 443588 | 166 | 34.39% | 100.00% | 34.39% | 100.00% | 51.18% |
| Normal | 323121 | 1116 | 119532 | 72 | 99.98% | 99.66% | 99.98% | 99.07% | 99.82% |
| Password | 8317 | 1319 | 432631 | 1574 | 84.09% | 86.31% | 84.09% | 99.70% | 85.18% |
| Port_Scanning | 4085 | 179 | 439193 | 384 | 91.41% | 95.80% | 91.41% | 99.96% | 93.55% |
| Ransomware | 1893 | 2 | 441690 | 256 | 88.09% | 99.89% | 88.09% | > 99.99% | 93.62% |
| SQL_injection | 9317 | 1288 | 431996 | 1240 | 88.25% | 87.85% | 88.25% | 99.70% | 88.05% |
| Uploading | 6949 | 647 | 435642 | 603 | 92.02% | 91.48% | 92.02% | 99.85% | 91.75% |
| Vulnerability_scanner | 9874 | 174 | 433592 | 201 | 98.00% | 98.27% | 98.00% | 99.96% | 98.14% |
| XSS | 2625 | 446 | 440187 | 583 | 81.83% | 85.48% | 81.83% | 99.90% | 83.61% |

Overall Statistics

| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 98.46% | 1.54% | 0.967 | 437015 | 6826 |

Fig. 16: Class and Overall Statistics of (K-NN) Algorithm.

As shown in Fig. 17, the results of classifiers, such as NN, k-NN, and DT is that while DT had a higher classification accuracy of (98.46), the F1-score, which presents the combined result of precision and recall ratios, had a higher value of (100), which represents the better classification capability of a model DT for multiclass classification.

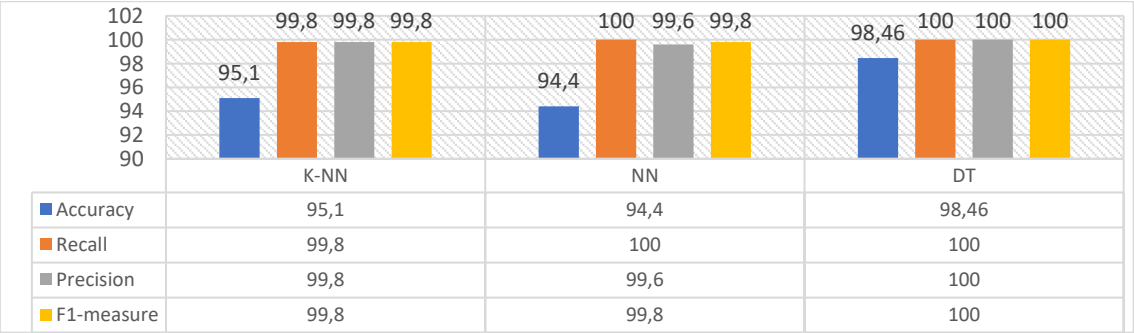



Fig. 17: Comparison Between Three Algorithms for Multiclass Classification

## 5.0 CONCLUSION

This study uses a realistic network dataset traffic to assess the dependability as well as effectiveness of a DT model of binary classification for network intrusion detection, ensuring the accuracy of the suggested models. Edge-IIoTtest dataset, which made use of the KNIME platform, was used to test our models (NN, k-NN, and DT); DT and k-NN for ML, and NN for DL for binary and multiclass classification. DT model's accuracy has increased as a result of the combination of the significant strengths of each model when the models were evaluated in terms of accuracy situations. With regard to multiclass and binary classification, the results were compared depending on recall, accuracy, F1-measure, and precision. With the two experiments (multiclass and binary), we achieved the maximum accuracy in DT model for binary classification (100%), and the highest accuracy in the DT model for multiclass (98.46%), with the highest precision, recall, and F1-measure of such percentages (100%). For enhancing IDS, the proposed DT architecture model is ideal for the system administrator as well as the networking designer or industry. Yet, this study investigates new datasets for testing and training and simultaneously applies DL and ML to the dataset. However, more study is required to look into attack behavior patterns and use the information gathered to improve prevention and prediction models. In order to improve accuracy and efficiency with new models, this work might be enhanced in the future by utilizing various optimization as well as feature selection approaches, using the model in real-time to categories, and capturing additional network traffic. Implementing a classification of incoming traffic as anomalous or normal is the goal. Finally, we compared the high accuracy of three previous studies with our study to obtain the highest accuracy for binary and multiclass classification models from the Table. 2. Our study obtained high accuracy (100%) for binary and high accuracy (98.46%) in multiclass classification for the DT model.

Table 2: Comparison Between Studies Depending on High Accuracy.

| **Studies** | **Year** | **High Accuracy of Classification** | | | |
|---|---|---|---|---|---|
| | | **Binary (%)** | **Algorithm** | **Multiclass (%)** | **Algorithm** |
| [16] | 2022 | 99.99 | DNN<br>k-NN<br>SVM<br>RF | 94.67 | DNN |
| [21] | 2022 | 97.27 | Poly BR<br>Poly PCA | - | - |
| [22] | 2023 | 99 | MEC-based architecture | - | - |
| Our study | 2023 | 100 | DT | 98.46 | DT |

## REFERENCES


[1] Ramya Mohanakrishnan, “Top 10 Applications of IoT in 2022.” Accessed: Apr. 07, 2024. [Online]. Available: https://www.spiceworks.com/tech/iot/articles/top-applications-internet-of-things/

[2] IoT.Business.News, “State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally.”

[3] Prof. Sathish and Dr. S. Smys, “A Survey on Internet of Things (IoT) based Smart Systems,” *Journal of ISMAC*, vol. 2, no. 4, pp. 181–189, Sep. 2020, doi: 10.36548/jismac.2020.4.001.

[4] M. Nuaimi, L. C. Fourati, and B. Ben Hamed, “Intelligent approaches toward intrusion detection systems for Industrial Internet of Things: A systematic comprehensive review,” *Journal of Network and Computer Applications*, vol. 215. Academic Press, Jun. 01, 2023. doi: 10.1016/j.jnca.2023.103637.

[5] V. Hnamte and J. Hussain, “DCNNBiLSTM: An Efficient Hybrid Deep Learning-Based Intrusion Detection System,” *Telematics and Informatics Reports*, vol. 10, Jun. 2023, doi: 10.1016/j.teler.2023.100053.

[6] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, “Challenges and Opportunities in Securing the Industrial Internet of Things,” *IEEE Trans Industr Inform*, vol. 17, no. 5, pp. 2985–2996, May 2021, doi: 10.1109/TII.2020.3023507.

[7] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, “IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices,” *IEEE Internet Things J*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019, doi: 10.1109/JIOT.2019.2935189.

[8] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, “Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning,” *IEEE Access*, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[9] M. Baich, T. Hamim, N. Sael, and Y. Chemlal, “Machine Learning for IoT based networks intrusion detection: a comparative study,” *Procedia Comput Sci*, vol. 215, pp. 742–751, 2022, doi: 10.1016/j.procs.2022.12.076.

[10] E. Tsogbaatar *et al.*, “DeL-IoT: A Deep Ensemble Learning Approach to Uncover Anomalies in IoT,” 2021. [Online]. Available: https://www.airbnb.com/

[11] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, and R. Colomo-Palacios, “A machine learning-based intrusion detection for detecting internet of things network attacks,” *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9395–9409, Dec. 2022, doi: 10.1016/j.aej.2022.02.063.

[12] Y. Zhang, P. Li, and X. Wang, “Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network,” *IEEE Access*, vol. 7, pp. 31711–31722, 2019, doi: 10.1109/ACCESS.2019.2903723.

[13] N. Prazeres, R. L. de C. Costa, L. Santos, and C. Rabadão, “Engineering the application of machine learning in an IDS based on IoT traffic flow,” *Intelligent Systems with Applications*, vol. 17, Feb. 2023, doi: 10.1016/j.iswa.2023.200189.

[14] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep Learning-Based Intrusion Detection for IoT Networks,” in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, IEEE, Dec. 2019, pp. 256–25609. doi: 10.1109/PRDC47002.2019.00056.

[15] M. Hasan, M. Milon Islam, M. Ishrak Islam Zarif, and M. Hashem, “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches,” 2019, doi: 10.1016/j.iot.2019.10.

[16] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[17] S. Ullah *et al.*, “A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering,” *Sensors*, vol. 22, no. 10, May 2022, doi: 10.3390/s22103607.

[18] M. Aamir and S. M. Ali Zaidi, “Clustering based semi-supervised machine learning for DDoS attack classification,” *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 4, pp. 436–446, May 2021, doi: 10.1016/j.jksuci.2019.02.003.

[19] K. Hara and K. Shiomoto, “Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder,” in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2020, pp. 1–8. doi: 10.1109/NOMS47738.2020.9110343.

[20] J. Pacheco, V. H. Benitez, L. C. Felix-Herran, and P. Satam, “Artificial Neural Networks-Based Intrusion Detection System for Internet of Things Fog Nodes,” *IEEE Access*, vol. 8, pp. 73907–73918, 2020, doi: 10.1109/ACCESS.2020.2988055.

[21] P. Dini *et al.*, “Design and Testing Novel One-Class Classifier Based on Polynomial Interpolation with Application to Networking Security,” *IEEE Access*, vol. 10, pp. 67910–67924, 2022, doi: 10.1109/ACCESS.2022.3186026.

[22] Z. A. El Houda, B. Brik, A. Ksentini, and L. Khoukhi, “A MEC-Based Architecture to Secure IoT Applications using Federated Deep Learning,” *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 60–63, Mar. 2023, doi: 10.1109/iotm.001.2100238.