IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

Performance Analysis of Tree-Based and Deep Learning Algorithms for Developing Distributed Secure Systems in IoT: A Comparative Study

Aziz Ullah Karimy[1*] and Dr. P Chandra Sekhar Reddy[2]
[1,2] Department of Electronics and Communication Engineering,
University College of Engineering , Science & Technology, Hyderabad
Jawaharlal Nehru Technological University, Hyderabad, Telangana, India.
Corresponding author:*azizullah.karimy91@gmail.com, drpcsreddy@jntuh.ac.in

Abstract: *Notably, IoT device utilization has experienced a substantial wave recently, and ensuring these devices' privacy and security has become a critical concern. ML-based security approaches are promising for IoT network protection against security concerns. This study provides a proximate analysis of tree-based and deep-learning algorithms for securing IoT domains. Specifically, we evaluate Decision Tree, RandomForest, XGBoost, Catboost, Extreme Tree, Light GMB, Adaptive Boosting, CNN, LSTM, MLP, GRU, and Autoencoder on four publicly available datasets - IoT23, CICID2017, EdgeIIoT, BotnetIoT and Contiki OS and Cooja simulation were used to generate a dataset featuring various RPL attacks. To assess the performance of a model, we measure its accuracy, precision, recall, and F1-score metrics. Our discoveries indicate that tree-based algorithms outperform deep learning algorithms regarding training time, memory usage, and interpretability while gaining comparable or even better detection accurateness. Conversely, deep-learning algorithms exhibit higher detection rates for rare or previously unseen attacks; their proficiency in detecting complex patterns and relationships within a given dataset has demonstrated remarkable efficacy in data analysis and classification tasks. We conclude that both tree-based and deep learning algorithms have their strengths and weaknesses, and in the IoT environment, one should base the choice of the algorithm on requirements and constraints. Our research shows hybrid approaches combining algorithm strengths can establish secure, distributed IoT systems.*

Keywords: Internet of things; machine learning; distributed secure system; deep learning; hybrid approaches

I. INTRODUCTION

1.1 Background

IoT has revolutionized how we interact with the digital world by enabling seamless connectivity between various devices. However, this interconnectivity also raises consequential security challenges as IoT devices become potential targets for malicious activities and cyber-attacks. The securing system is a vital shield for monitoring and identifying suspicious or anomalous activities within IoT networks. The effectiveness and efficiency of the algorithms directly impact the overall security in IoT systems. ML is a versatile data analysis tool crucial in IoT applications, allowing for autonomous decision-making and real-time processing[1].In this research, we focus on two categories of ML algorithms: tree-based and deep learning. This choice is motivated by the specific needs of IoT security research. Tree-based algorithms have garnered significant attention due to their interpretability and capability to handle heterogeneous data types, specifically tabular data. These algorithms, including decision tree, random forest, extreme trees, and gradient boosting, have shown exceptional performance in various domains, ranging from finance and healthcare to environmental monitoring[2]. The efficiency of tree-based algorithms makes them an excellent choice for IoT apps with limited data on account of resource constraints. Furthermore, their less demanding computational needs align well with the constraints of IoT devices, allowing for local processing at edge of the network, reducing latency, and minimizing data transmission to central servers[3].

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

Conversely, DL has made significant headway in image recognition, natural language processing, and audio analysis. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are highly effective in learning complex patterns and representations from large datasets. However, applying these techniques to the IoT poses resource challenges, and their computational complexity & memory requirements often need to be more practical for resource-limited IoT devices. Further, transmitting considerable volumes of data to central servers raises concerns about privacy and security[4].When choosing algorithms for IoT applications, it is essential to consider the constraints. Tree-based algorithms are highly efficient and are a good fit for IoT apps with limited data. On the other hand, Deep Learning techniques are compelling but can pose resource challenges due to their computational complexity and high memory requirements. As a result, they often require optimization or adaptation for resource-limited IoT devices.By narrowing our focus to these categories, a broad analysis of the performance of tree-based and deep-learning algorithms is needed. Understanding their strengths, weaknesses, and resource requirements is essential for developing effective securing mechanisms that ca ter to resource-constrained nature of IoT devices. By conducting this proximate study, we aim to provide an understanding of the most suitable algorithms for different IoT scenarios and contribute to advancing IoT security.

1.2 Objectives

This article embarks on a novel approach to IoT security by thoroughly examining the performance of Tree-based and Deep Learning algorithms. We take into account the resource limitations of IoT devices and provide a detailed analysis of algorithm effectiveness. This helps in developing practical and effective secure distributed systems that cater to the specific requirements of the IoT environment. We will rigorously evaluate tree-based algorithms, including Decision Trees, Random Forest, XGBoost, Catboost, Extreme Trees, and LightGBM. In addition, we will subject a spectrum of deep-learning algorithms, such as CNNs, LSTMs, MPL, GRU, and Autoencoders, to meticulous evaluation. Additionally, we introduce evaluation metrics optimized for IoT applications, encompassing accuracy, precision, recall, F1-score, and resource utilization.

1.3 Contribution

The contributions of this paper encompass:

- A comprehensive evaluation of tree-based and deep-learning algorithms for intrusion detection in IoT applications.
- The utilization of multiple datasets, including IoT23, CICID2017, EdgeIIoT, BotnetIoT, and a bespoke dataset generated from Contiki OS and Cooja simulation, featuring various routing attacks.
- A comparative assessment of algorithm performance employing a diverse set of evaluation metrics, facilitating the identification of strengths and weaknesses in intrusion detection systems.

1.4 Paper Organization

Section 2 of the paper presents an extensive overview of existing approaches in intrusion detection, employing both tree-based and deep-learning algorithms. Section 3 delves into the experimental setup, covering the selection of algorithms and datasets. In Section 4, we elaborate on our methodology, encapsulating preprocessing steps and algorithm testing. Sections 5 and 6 offer an in-depth analysis of performance results, complemented by meaningful discussions. The paper concludes with insights into the efficacy of tree-based and deep-learning algorithms in securing IoT applications and provides recommendations for future research.

II. REVIEW OF RELEVENT STUDIES

The following section covers the research closely related to attack detection using ML- and DL-based traffic classifiers. We focus on existing studies published within the last three years. [5] concentrated on assessing the performance of various techniques for catching intrusions in IoT apps. They used the latest datasets (TON-IOT) created in the "Cyber-Range & IoT Lab at UNSW University." These datasets comprised various data sources like telemetry from IoT sensors, operating systems (Windows 7 & 10, Ubuntu 14), and network traffic. DT, RF, adaboost, XGBoost, ANN, & MLP algorithms were trained and tested to classify unknown & normal network traffic. [6] suggested and deployed an

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

exciting approach to detecting and preventing malicious activity in an IoT apps using DT classifier. After preprocessing, discarding some nominal attributes, specific attributes, missing values, and correlated features, they down-sampled Avast IoT-23[7]dataset in two samples of 80% & 20% to train and test it. They achieved 99.9% accuracy in classifying each label. [8] In this experimental study of the "UNSW-NB15" dataset, several classifiers were analyzed to classify attacks. SVM, XGBoost, Cat Boost, KNN, QDA, & NB classifiers were tested. Dataset was first normalized using the min-max concept to limit information leakage in test data. The dataset was subjected to PCA to reduce its dimensionality. Dataset contains nine types of attacks: Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Fuzzers, Shellcode, and Worm. The top-performing classifiers were XGBoost & Cat Boost.

[9] Assessed the performance of DNN for intrusion detection in an IoT apps; model used three datasets "KDD99, NSL-KDD, & UNSW-NB15" for training. Eight features were extracted & used as input. They divided the datasets into 70%, 15% & 15% to train, validate and test. Implemented this model on 6BR (IPv6 Border Router) to monitor traffic, and based on their training, it would detect attacked behaviour. UNSW-NB15[10] used with DNN produced a higher performance of 99.2% at epoch 19, and NSL-KDD [11] used with DNN produced same performance of 99.2% at epoch 19, & achieved an accuracy of 91.5% attack detection at epoch 43. [12]Assessed (GRU, CNN, RNN, LSTM) with KDD-CUPP to forecast intrusions in MQTT. Attackers often exploit vulnerabilities in the MQTT protocol due to its lightweight nature, making it a popular target in IoT applications. In this experimental study, the LSTM algorithm outperforms other algorithms. Enriching security in a multi-cloud IoT domain is a recent research breakdown by [13], which suggested a model for intrusion detection in the IoT field, employing DL techniques to identify & categorize security breaches. The NSL-KDD was preprocessed, and one-hot encoding was applied to extract 41 features, which were then mapped to a 122-dimensional LeNet-based model selected for detecting intrusions in the IoT domain; the proposed model outperformed some of the existing neural network-based IDS obtaining overall of 97.5% accuracy.

The analysis shows that suitable algorithm to utilize relies on the particular use case, the characteristics of the IoT ecosystem, the computing power of edge devices, and data that is accessible. More research is required to investigate these algorithms' scalability, adaptability, and robustness in extensive IoT deployments. Federated learning approaches improve efficiency and privacy when incorporated into a secure IoT apps.

III. METHODOLOGY

3.1 Tree-Based Algorithms

Tree-based algorithms are powerful and versatile methods widely employed in different domains of ML applications. These algorithms, based on decision trees, offer an intuitive and interpretable way to model complex data relationships. A hierarchical, tree-like arrangement recursively splits the input space into subsets. Each inner node corresponds to a decision based on a particular feature, while each leaf node characterizes a predicted outcome.

A. Decision Tree

The decision tree symbolizes a non-parametric ML technique in supervised learning tasks. This method selects attributes for nodes in two ways: information gain and gini impurity.

Entropy counts the impurity or randomness in the target variable's class labels in a dataset, and we calculate it as follows:

$$Information\ Gain\ (S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|}.Entropy(Sv).....i$$

$$Entropy(S) = -\sum_{c \in classes} \frac{|S_c|}{|S|}.lan(\frac{|S_c|}{|S|}).....ii$$

In equations 1,2, and 3, D is original dataset, A denotes features considered for split of data, values(A) represent distinct values of feature A in dataset D, and Dv is a subset ofdata

The Gini impurity formula count the probability of misclassifying an instance in dataset D by randomly picking an example and labelling it according to class distribution[14].

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

$$Gini\ (S)\ =\ 1 - \sum_{c \in classes} \left(\frac{|S_c|}{|S|}\right) \ldots\ldots\ldots\ldots iii$$

B. Random Forest

RF is an ensemble learning technique leveraging numerous Decision Trees for prediction. Every decision tree is trained on a random subset of the data and features. The ultimate prediction is constructed by amalgamating the outcomes of all individual Decision Trees. Prediction is computed[14].

$$y\ =\ \frac{1}{T}\sum_{t=1}^{T} y_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots iv$$

Where yt is the prediction of tree t, and S is the total number of trees in the Random Forest.

C. Extem Gradient Boosting (XGBoost)

XGBoost is a Boosting-based algorithm. Boosting is the process by which weak decision trees are constructed in series to generate a strong learner/estimator, with each consecutive tree attempting to learn from its predecessor's mistakes and working to minimize those mistakes[15].

$$Objective\ Function\ =\ \sum_{i=1}^{n}\left[S(y_i, y_i\hat{}) + \sum_{k=1}^{K}\Omega(f_k)\right] \ldots\ldots\ldots\ldots v$$

In equation 5, n is the number of training samples, $S(y_i, y_i)\hat{}$ is the loss function, K is the number of trees in ensemble, $\&\Omega(f_k)$ is the regularization term that penalizes complex models to prevent overfitting.

D. Light Gredient Boosting Machine (LightGBM)

LightGBM is a gradient boosting technique that employs a tree-based learning strategy; this method grows trees leaf-by-leaf, and other algorithms build levels-by-level, making it faster & memory-efficient.

E. Adaptive Boosting (AdaBoost)

AdaBoost is an ensemble technique primarily designed for classification problems. Several weak classifiers are combined to form a robust and accurate model. AdaBoost's basic principle is to concentrate on misclassified instances throughout the training stage, allowing succeeding classifiers to fix the mistakes made by prior ones.

$$Prediction\ (i)\ =\ sing\left(\sum_{t=1}^{T}\alpha_t\,.\,h_t(i)\right) \ldots\ldots\ldots\ldots vi$$

Where T is the total number of weak classifiers, $h_t(i)$ is the prediction of the t^{th} weak classifier for input i, and α_t is weight of t^{th} weak classifier.

F. Categorical Boosting (CatBoost)

CatBoost is open-source boosting library designed by Yandex. It's intended for use on issues with many independent variables, such as regression and classification. The CatBoost method gradually builds an ensemble of decision trees, with each tree correcting mistakes of the preceding ones.

G. Extreme Randomized Classifier (Extra Tree)

ExtraTrees is ensemble ML strategy that trains a many decision trees and combines their results to provide a forecast. Extra Trees add a randomization level to tree-building process, making it more robust and less prone to overfitting.

In this research, we fine-tuned the hyperparameters of the selected algorithms. With carefully tailored parameters to match each algorithm's unique requirements and characteristics. For Random Forest and Extra Trees classifiers, we set n_estimators to 100 to construct ensembles of 100 trees, while max_depth was fixed at 25 to control tree depth. Additionally, we specified as 'entropy' to enable information gain-based splitting. As for XGBoost, CatBoost, and

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

LightGBM, we configured n_estimators to 100 to indicate the number of boosting rounds, and we set learning_rate to 0.001 to regulate the step size in gradient descent optimization.

3.2 Deep Learning Algorithms

DL a subfield of ML, uses computational algorithms to learn and improve independently. Utilizing it in cybersecurity improves security, detects/prevents cyber threats & protects data. DL methods have shown great promise in fighting the increasingly complex nature of cyber threats in cybersecurity, and they employ ANNs, which are intended to mimic how people think and learn. The general formula is as follows:

$$output = f(x) = \sum_{i=1}^{n} w_i x_i + bias \ldots\ldots\ldots vii$$

where w_i illustrate the weights associated with each input feature x_i, & xi represents input data for individual nodes. The function f (x) represents the activation function, which is applied to the sum of weighted inputs and the bias to produce the final output.

A. Convolution Neural Network (CNN)

CNNs are a discriminative DL model widely utilized to handle massive training datasets by employing hierarchical attribute extraction and articulation. To utilize the 2-D input data structure, the network employs local connections and shared weights instead of fully connected networks. When implementing a CNN model, the selection of the loss function is of utmost importance. For multi-class problems, we used categorical cross-entropy to assess classification accuracy. The Nadam optimizer was chosen to handle the weight updates, dynamically adjusting the model parameters to minimize the selected loss function. In our code, we have implemented this approach in the following manner:

nadam = Nadam(lr=0.008, beta_1=0.9, beta_2=0.999, epsilon=1e-08, schedule_decay=0.004)
model.compile(loss='categorical_crossentropy', optimizer=nadam, metrics=['accuracy'])

During the training process, we used Keras to manages the backpropagation, which iteratively updates the model's parameters to improve its performance based on errors.

B. Multi-Layer Perceptron (MLP)

The MLP is a versatile and robust architecture consisting of multiple layers of nodes (neurons) interconnected by weighted edges. The input layer acquires input data, which is then processed through the hidden layers before producing the output in final layer. Each node in hidden layers involves a non-linear function to the weighted sum of its intakes, allowing the MLP to learn complex non-linear relationships between the input and output data. It is used for classification, regression, and other tasks. MLP is trained by optimizing its weights and biases using a suitable optimization technique, for this study we used Nandam optimizer, to minimize the training data's prediction error (loss function). Back-propagation is a procedure that incorporates forward propagation (to compute predictions) & backward propagation (to update weights and biases) and It is handled using keras, during the entire training process, allowing the model to learn from its errors.

C. Long-Short Term Memory (LSTM)

LSTM networks are RNNs that can learn order dependency in sequence prediction challenges. LSTM networks were developed to overcome the vanishing gradient problem in standard RNNs, making them more suitable for sequential input applications. During training, we utilized Adam optimizer with lr of 0.01, dynamically adjust the model parameters & minimize loss function. To prevent overfitting, 'early stopping' is deployed, callback that stops training if validation loss remains stagnant for number of consecutive epochs.

D. Gated Recurrent Unit (GRU)

It's an alternative RNN architecture type. GRU was presented as alternative to the classic LSTM model, simplifying the LSTM architecture while maintaining comparable performance in sequential data workloads. In this experiment, we used the same optimizer and loss function for both the GRU and LSTM models.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

E. Autoencoder

Autoencoders are a DL algorithm that acquires input and converts it into new replica. Its immediate applications are dimensionality reduction and feature learning. Autoencoders comprise two major components: encoder & decoder.

$$encoding = Z = f(x) = \sigma(W_{encode}.X + b_{encode}).....viii$$

Where X is input data, Z is encoding of input, W_{encode} is weight matrix, & b_{encode} is bias vector of the encoder. Function f (·) represents the activation function used in the encoder, such as ReLU or sigmoid, & σ(·) denotes activation function.

$$decoding = X' = g(Z) = \sigma(W_{decode}.Z + b_{decode}).....ix$$

X′ is reconstructed output, W_{decode} is weight matrix, & b_{decode} is bias vector of the decoder, & g(·) rep resents activation function used in decoder. for this study, after extensive experimentation and evaluation of various optimizers and loss functions, we found that for this particular autoencoder model, the com bination of SGD optimizer with sparse categorical cross-entropy loss function yielded the most optimal results.

3.3 Datasets

Examining datasets is critical to validating any ML techniques since it allows us to determine how effectively the proposed method can detect invasive activities. Public datasets extensively serve as benchmarks.

A. CICIDS-2017

CICIDS-2017 dataset was created and included labelled flows examined using CICFlowMeter and real-world data (PCAPs) of standard and benign attacks. Realistic background traffic generation was a crucial consideration when beginning the CICIDS-2017 dataset. The suggested B-Profile system was used to simulate the behaviour of 25 users across different protocols, including HTTP, HTTPS, FTP, SSH, and email, producing naturalistic benign traffic[16].

The five-day data collection phase included regular daytime traffic (Monday) and actual attacks (Tuesday to Friday). Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS were all included in the attacks. They used a complete assessment approach with eleven criteria to ensure dataset's dependability because CICIDS-2017 dataset met these criteria, distinguished as a trustworthy benchmark for IDS evaluations.

B. IoT-23

IoT-23 is new network traffic dataset from IoT devices. It includes 20 malware catches in IoT, and three benign IoT device traffic grabs. Malicious & benign scenarios are executed within a controlled network environment, simulating unrestrained internet connections akin to real-world IoT devices[7].

C. Edge-IIoTset

Edge-IIoTset is a unique and comprehensive cyber security dataset created exclusively for IoT and IIoT applications. ML-based intrusion detection systems find this resource valuable in centralized & federated learning modes. Dataset comprises a testbed with seven layers, each integrating cutting-edge technology to meet the specific requirements of IoT and IIoT environments. Dataset encompasses information from various IoT devices, including temperature & humidity sensors, ultrasonic sensors, and water level detectors. It covers wide range of fourteen attacks against IoT & IoT communication protocols divided into five categories of threats, including DoS/DDoS, data collection, man-in-the-middle attacks, injection attacks, malware attacks, and man-in-the-middle attacks[17].

D. N-BaIoT

The dataset addresses lack of accessible botnet data, particularly for IoT apps. It includes traffic information from nine business IoT devices infected by Mirai & BASHLITE botnets[18]. Dataset aims to differentiate between genuine and malicious communications via the use of anomaly detection techniques. However, as the malicious data comprises ten different types of attacks from two other botnets, it can also be utilized for multiclass classification, which involves ten attack categories and one "benign" category. The feature headers in dataset contain several statistics that summarise

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

most recent traffic to and from particular IP addresses and ports, as well as time-frame data. Statistics include weight, mean, standard deviation, radius, magnitude, covariance, and Pearson correlation coefficient.

E. RPL-Attacks

We used Contiki OS & Cooja simulation to create a dataset explicitly aimed towards RPL attacks to increase the diversity of datasets for our study. We used benign and malicious nodes to mimic three different attack types: flooding attack, decreased rank attack, and version number increase attack. As a consequence, these simulations produced a total of 6 raw datasets.

Raw data must be preprocessed before applying ML for intrusion detection. We noticed that regarding packet counts, message categories, total packet lengths, and rates, raw data from simulations with weak nodes varied greatly from those with normal nodes. This was addressed by breaking up raw data into 1-second frames and extracting different calculated values from each frame to produce a new dataset with sixteen features as follow:

1-source node, 2-destination node, 3-packet count, 4-source node ratio, 5-destination node ratio, 6-source node duration, 7-destination node duration, 8-total packet duration, 9-total packet length, 10-source packet ratio, 11-destination packet ratio, 12-DIO message count, 13-DIA message count, 14-DIS message count, 15-non protocol message count, 16-label.

IV. EXPERIMENTAL EVALUATION

4.1 Data Pre-Processing

An effective pre-processing strategy is crucial for optimal performance of ML algorithms on diverse datasets. In preparing data for model training, we used following method:

- Handling Missing Values: Datasets' missing values were dealt with utilizing appropriate, dataset-specific techniques before any additional processing. We ensure dataset is complete and consistent by adding or imputing missing values.
- Handling Categorical Values: To process categorical data in datasets, we utilized OneHotEncoder to convert them into binary vectors. This eliminates assumptions & ensures that specific categories dominate during model training.
- Label Encoding: Label encoding was used to translate target labels into numerical values in classification tasks. This numerical form makes it easier to train models for categorization issues.
- Numerical Feature Scaling: Scaling of features is necessary to avoid numerical features dominating others only because of their scale. Scaling the numerical features within a given range was done using MinMaxScaler. This normalization procedure ensures that each feature contributes equally to model training.
- Addressing Class Imbalance: We noticed unbalanced class distributions in some of datasets used for our research. This can have a significant impact on performance of the model. To address class imbalance, we employed Synthetic Minority Over-sampling Technique (SMOTE) to generate instances of minority class artificially. This helped to improve its representation & reduce the effects of class inequality. Additionally, we used RandomUnderSampler to remove instances from majority class to address class imbalance and reduce bias. To make these preprocessing steps more effective, we implemented a Pipeline to chain them. This made it easier to transform data cohesively and systematically, ensuring it was handled consistently.
- Feature Selection: Datasets with many features may lead to overfitting and complicated calculations. Feature selection based on ANOVA F-value f_classif can solve these problems.The approach identified primary features that have most significant impact on target variable. Dimensionality was decreased by keeping only most valuable features, which improved the model's capacity to be generalized and understood.

4.2 Testing

In this section, we outline the testing methodology, performance findings, and in-depth analyses based on evaluating tree-based and DL algorithms on various datasets. We aimed to assess the efficiency of tree-based and DL algorithms & compare their efficacy.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

We extensively evaluated each algorithm on five different datasets: IoT23, CICID2017, EdgeIIoT, BotnetIoT, and custom dataset produced using Contiki OS & Cooja simulation for RPL protocol. The structured method described in the "Preprocessing" section was used to preprocess datasets. We carried out 10-fold cross-validation for each algorithm-dataset combination and repeated procedure three times to guarantee a fair assessment.

$$Accuravy = \frac{TP + TN}{TP + TN + FP + FN} \dots\dots\dots\dots x$$

$$Precision = \frac{TP}{TP + FP} \dots\dots\dots\dots xi$$

$$Recall = \frac{TP}{TP + FN} \dots\dots\dots\dots xii$$

$$f1 - score = \frac{TP}{TP + {}^{1}/_{2}\,(FP + FN)} \dots\dots\dots\dots xiii$$

Where: True positive (TP) = number of samples correctly detected as intrusion samples; True negative (TN) = number of samples correctly detected as normal samples; False positive (FP) = number of samples incorrectly detected as intrusion samples; False negative (FN) = number of samples incorrectly detected as normal samples.

Performance Metrics: Accuracy, F-Score, Recall, and Precision were performance metrics utilized for evaluation. For each method and dataset, we calculated these measures, capturing models' capacity to categorize cases accurately, their accuracy in recognizing genuine positives, and their recall in catching all positive instances. Training score was also measured, reflecting the algorithms' ability to fit training data. Based on (Table 1) Decision Tree has displayed excellent performance in both binary and multiclass classification tasks, achieving high performance on most datasets with comparably minimal training time. Random Forest performs better than Decision Tree and has acceptable training times. Similarly, XGBoost demonstrated flawless training scores and outstanding performance across all measures to stand out as a top performer in each category. Although LightGBM likewise showed remarkable performance in both classifications with modest training time and a balance between efficiency and accuracy, its training duration was very high. AdaBoost performed very well in both instances, displaying competitive accuracy, precision, and recall levels. However, compared to other methods, its training time was substantially more extended, which may limit its applicability to large datasets and limited computational capabilities. CatBoost demonstrated strong performance in binary & multiclass classification, offering excellent accuracy, precision, and recall. It was a practical option for various classification tasks because of its reasonable training duration.

According to (Table 2)., CNN performed well in classifying data from various datasets. On the CICIDS 2017, it earned good accuracy, precision, and recall scores, demonstrating its efficacy in detecting true positives. Its performance on IoT-23, however, was noticeably worse, indicating a possible need for model or data augmentation. The training period was brief. Conversely, MLP provided competitive performance with reasonable accuracy, precision, and recall on most datasets, notably CICIDS 2017. Its performance on IoT 23, however, varied considerably, indicating sensitivity to data distribution and attributes.

To carry on exploring DL, LSTM has shown varying performance across datasets. Its performance on IoT-23 could have been better, demonstrating difficulty in capturing sequential patterns and recollection, despite achieving outstanding accuracy, precision, and recall on datasets, and the training period was lengthy.

Similar to LSTM, GRU's performance changed depending on dataset. It demonstrated strong recall, accuracy, and precision on Edge-IIoT, but it had limits on IoT-23, possibly making it impossible to capture long-term dependencies despite its short training period. To carry on exploring DL, LSTM has shown varying performance across datasets. Its performance on IoT-23

could have been better, demonstrating difficulty in capturing sequential patterns and recollection, despite achieving outstanding accuracy, precision, and recall on datasets, and the training period was lengthy.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

Similar to LSTM, GRU's performance changed depending on dataset. It demonstrated strong recall, accuracy, and precision on Edge-IIoT, but it had limits on IoT-23, possibly making it impossible to capture long-term dependencies despite its short training period. On most datasets, AutoEncoder showed competitive performance, obtaining good accuracy, precision, and recall. Its performance on IoT23, however, was noticeably worse, indicating a need for more feature engineering or model advancements

Table 1. Tree based algorithms, Binary and Multiclass Classification

Algorithms	Binary Classification						Multiclass Classification				
	Datasets	Training Score	Accuracy	Precision	Recall	Training Time	Training Score	Accuracy	Precision	Recall	Training Time
Decision Tree	BotNetIoT	1.00	0.9999	0.9999	0.9996	18.656	0.91314	0.9103	0.9198	0.9103	5.5742
	CICIDS2017	0.9989	0.9982	0,9983	0.9982	109.25	0.9998	0.9986	0.9987	0.9986	45.839
	IoT23	0.9233	0.8840	0.8657	0.8440	18.405	0.7569	0.5445	0.5596	0.5445	99.337
	EdgeIIoT	1.00	0.9999	1.00	1.00	3.5852	0.9969	0.9636	0.9638	0.9636	6.2994
	RPL	1.00	0.8060	0.8062	0.8060	0.0367	1.00	0.9014	0.9094	0.9041	0.0199
Random Forest	BotNetIoT	1.00	0.9999	0.9999	0.9999	100.57	0.9131	0.9106	0.9557	0.9106	39.350
	CICIDS2017	0.9989	0.9982	0.9983	0.9982	676.24	0.9990	0.9987	0.9988	0.9987	198.18
	IoT23	0.8010	0.8952	0.8900	0.8952	240.91	0.7007	0.7238	0.6527	0.7238	1261.5
	EdgeIIoT	1.00	0.9999	1.00	0.9998	68.284	0.9654	0.9603	0.9645	0.9603	106.90
	RPL	1.00	0.8525	0.8566	0.8525	0.7628	1.00	0.9417	0.9425	0.9417	0.4388
XGBoost	BotNetIoT	1.00	0.9999	0.9999	0.9999	80.574	1.00	0.9998	0.9998	0.9998	352.98
	CICIDS2017	0.9994	0.9992	0.9993	0.9992	944.43	0.9995	0.9990	0.9991	0.9990	2161.5
	IoT23	0.9112	0.9328	0.9370	0.9328	213.79	0.7317	0.7377	0.7163	0.7377	1731.3
	EdgeIIoT	1.00	0.9999	0.9998	1.00	158.26	0.9817	0.9851	0.9854	0.9851	4091.8
	RPL	0.9917	0.8903	0.8928	0.8903	0.6139	1.00	0.9441	0.9429	0.9441	1.2524
LightGMB	BotNetIoT	1.00	1.00	1.00	1.00	10.042	0.9998	0.9989	0.9989	0.9989	44.040
	CICIDS2017	0.9992	0.9990	0.9991	0.9990	46.760	0.9673	0.9757	0.9810	0.9757	96.400
	IoT23	0.9025	0.9194	0.9291	0.9194	13.961	0.6253	0.6401	0.6218	0.6401	87.500
	EdgeIIoT	1.00	0.9999	0.9998	1.00	11.762	0.6738	0.7085	0.7888	0.7085	107.89
	RPL	0.9747	0.8700	0.8730	0.8700	0.1735	1.00	0.9490	0.94478	0.9490	0.5832
AdaBoost	BotNetIoT	1.00	0.9999	0.9999	0.9999	139.50	0.4427	0.4399	0.3940	0.4399	60.118
	CICIDS2017	0.9886	0.9879	0.9883	0.9879	425.72	0.7739	0.8914	0.8185	0.8914	174.59
	IoT23	0.7070	0.8783	0.8605	0.8783	78.930	0.6435	0.6697	0.5410	0.6697	92.116
	EdgeIIoT	1.00	1.00	0.9999	0.9998	5.0257	0.7547	0.8332	0.7934	0.8332	107.89
	RPL	0.7845	0.7691	0.7700	0.7691	02942	0.7487	0.7439	0.8129	0.7439	0.2318
CatBoost	BotNetIoT	1.00	0.9999	0.9999	0.9999	111.035	0.9999	0.9998	0.9998	0.9998	1562.6
	CICIDS2017	0.9991	0.9990	0.9990	0.9991	572.038	0.9992	0.9989	0.9990	0.9989	1601.3
	IoT23	0.9034	0.9215	0.9300	0.9215	311.027	0.7193	0.7278	0.7046	0.7278	1547.1
	EdgeIIoT	1.00	1.00	0.9999	0.9998	302.35	0.9755	0.9834	0.9852	0.9834	475.80
	RPL	0.9585	0.8787	0.8825	0.8787	4.9963	0.9976	0.9417	0.9399	0.9417	16.105
Extra Tree	BotNetIoT	1.00	0.9999	0.9999	0.9999	34.971	1.00	0.9997	0.9997	0.9997	15.317
	CICIDS2017	0.9999	0.9986	0.9986	0.99855	408.47	0.9998	0.9982	0.9983	0.9982	106.89
	IoT23	0.9330	0.8778	0.8801	0.8778	202.30	0.7784	0.5727	0.5859	0.5727	285.36
	EdgeIIoT	1.00	1.00	1.00	0.9999	56.9444	0.9969	0.9729	0.9728	0.9729	104.27
	RPL	1.00	0.8467	0.8487	0.8467	0.5413	1.00	0.9344	0.9320	0.9344	0.4718

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

Table 2. Deep learning Classification

Algorithms	Datasets	Accuracy	Precision	Recall	F1-socre	Training Time
CNN	BotNetIoT	0.8461	0.8953	0.8461	0.8132	689.78
	CICIDS2017	0.9961	0.9962	0.9961	0.9960	1230.9
	IoT23	0.6902	0.6068	0.6902	0.5980	2185.2
	EdgeIIoT	0.9535	0.9593	0.9535	0.9509	2966.8
	RPL	0.9519	0.9502	0.9503	0.9496	42.98
MLP	BotNetIoT	0.9067	0.9222	0.9067	0.8775	43.343
	CICIDS2017	0.9966	0.9967	0.9966	0.9966	1284.1
	IoT23	07086	0.6646	0. 7086	00.6809	705.38
	EdgeIIoT	0.9526	0.9641	0.9526	0.9480	387.17
	RPL	0.9305	0.9271	0.9305	0.9267	8.0258
LSTM	BotNetIoT	0.4523	0.4345	0.4567	04456	1271.2
	CICIDS2017	0.9249	0.9060	0.9249	0.9100	6982.8
	IoT23	0.5750	0.4659	0.5750	0.4198	5633.20
	EdgeIIoT	0.7809	0.7371	0.7809	0.7444	7180.3
	RPL	0.9134	0.9101	0.9134	0.9115	93.132
GRU	BotNetIoT	0.7985	0.7737	0.7985	0.7430	145.87
	CICIDS2017	0.9249	0.9060	0.9249	0.9100	5894.1
	IoT23	0.6560	0.5784	0.6560	0.5354	6039.1
	EdgeIIoT	0.7262	0.5274	0.7262	0.6110	639.14
	RPL	0.7841	0.7858	0.7841	0.7687	38.938
AutoEncoder	BotNetIoT	0.8886	0.8462	0.8886	0.8593	1130.1
	CICIDS2017	0.9787	0.9776	0.9787	0.9770	5833.2
	IoT23	0.6898	0.6071	0.6898	0.5975	7233.7
	EdgeIIoT	0.9199	0.8972	0.9199	0.8992	6340.5
	RPL	0.9049	0.8663	0.9049	0.8815	300.54

V. RESULT AND DISCUSSION

To establish an IoT system focused on security and incorporating federated learning, our investigation in this study centered on assessing the effectiveness of tree-based and DL algorithms in context of binary and multiclass classification tasks. We additionally factored in computing resources available within the IoT ecosystem, data volume at edge devices, and power consumption when determining suitability of these algorithms for conducting localized training at edge.

5.1 Tree-Based Algorithms Analysis

This work looked at tree-based algorithms performing well in binary and multiclass classification tasks. The performance metrics, as summarized in (Table 1) demonstrate that these algorithms received favorable assessment results on most datasets, positioning them as promising candidates for federated learning-based secure IoT systems as evident in (Fig. 1 and Fig 2).Tree-based methods are often small and quick to compute, making local training at the edge levels possible. The resource limitations of edge devices in the IoT ecosystem are well matched by their simplicity in handling relatively limited datasets and their lower computing demands.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

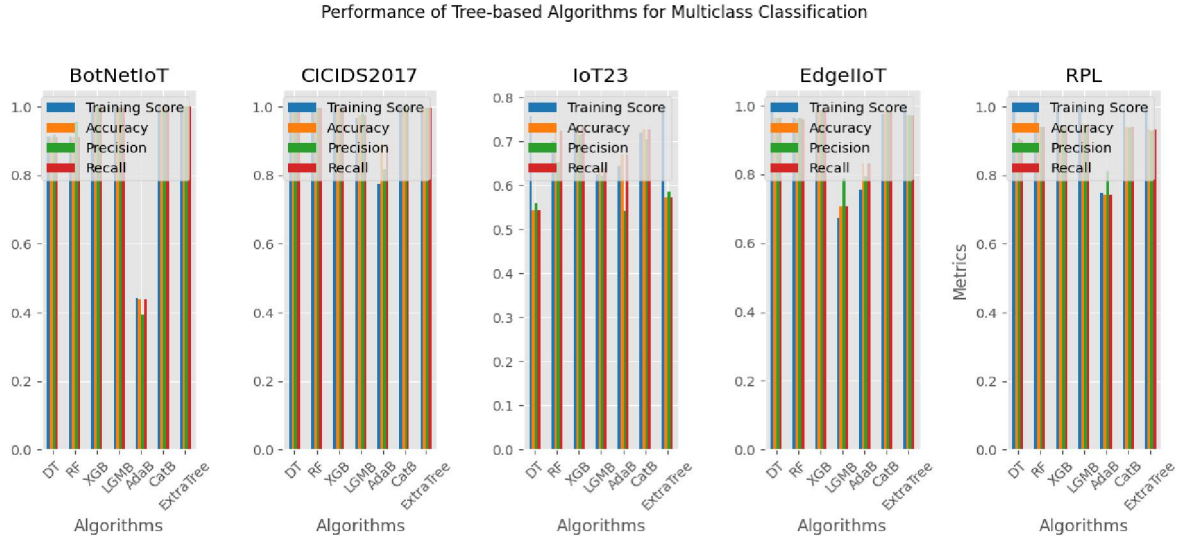

Fig 1: Evaluation metrics for Multi-Class Classification.

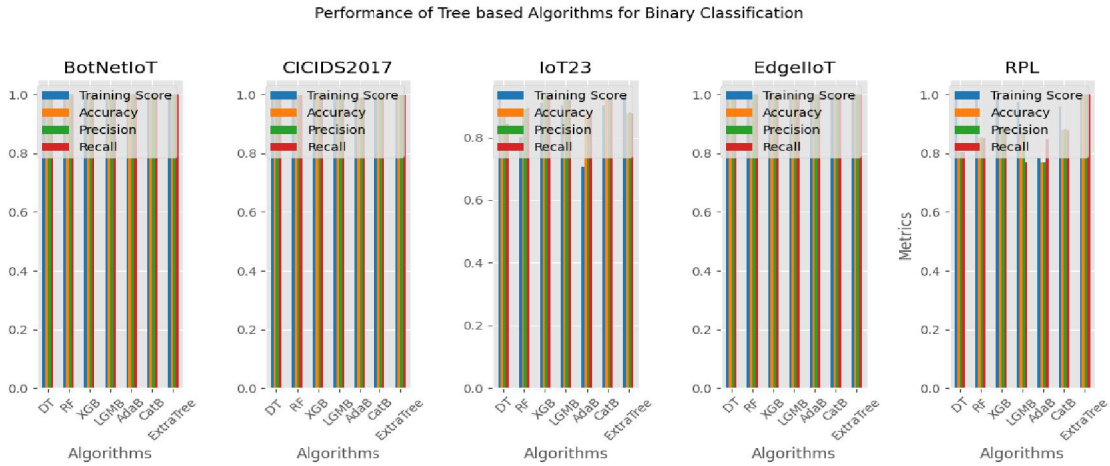

Fig 2. Evaluation metrics for Binary Classification.

5.2 Deep Learning Algorithms Analysis

The deep learning algorithms, such as CNN, MLP, LSTM, GRU, and AutoEncoder, also demonstrated promising outcomes for binary and multiclass classification problems. These algorithms captured complicated patterns with excellent precision and accuracy across most IoT datasets. DL techniques, however, often need an immense volume of data for training, which may be difficult for edge devices with constrained storage and connection options. Additionally, training times for DL algorithms were noticeably more prolonged than those for tree-based algorithms, highlighting the necessity for careful consideration of computational resources at the edge level of IoT applications.

These algorithms, in particular Random Forest and XGBoost as show in (Fig 3.), have proven reliable, making them appropriate for federated learning in a decentralized IoT environment. Their quick training times make them more suitable for implementation on edge devices with limited resources. Although DL algorithms demonstrated promising performance, their more demanding computing needs and demand for larger datasets may make it challenging to implement effectively at edge devices.Transferring computational load to more powerful cloud or centralized servers could represent a viable strategy for specific tasks, particularly when complexity of data patterns requires the utilization of DL techniques.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

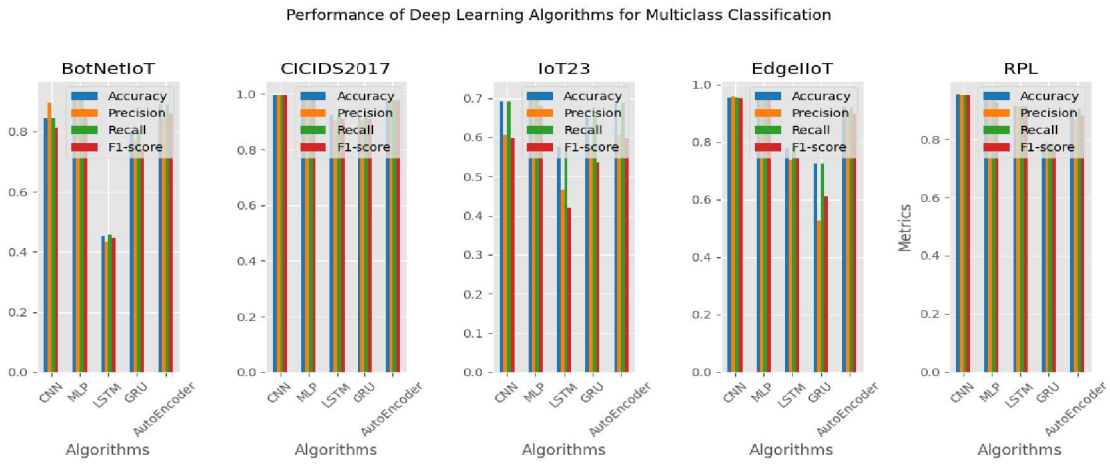

Fig 3. Evaluation metrics for Deep Learning algorithms.

VI. CONCLUSION AND FUTURE PROSPECTIVES

In this research, we compared two types of algorithms - tree-based and deep learning - to determine the ideal approach for creating a secure and distributed system for IoT apps. Our analysis involved testing multiple techniques using publicly accessible data sets and a unique dataset that we created by simulating Contiki OS & Cooja under various RPL attacks. We evaluated the performance of the algorithms based on several indicators such as accuracy, precision, recall, F1-score, and resource utilization.

The research findings have demonstrated that tree-based algorithms, including Random Forest and XGBoost, have shown better performance than deep learning methods concerning factors such as training time, memory usage, and interpretability. Additionally, these tree-based algorithms deliver comparable or even superior detection accuracy. Tree-based algorithms are especially suitable for local training at the edge level due to their capability to handle small datasets and minimal computational requirements, which makes them an excellent choice for IoT devices with limited resources.

Based on our study, a combination of deep learning algorithms and tree-based algorithms could be advantageous for future research. Although deep learning algorithms are effective in capturing complex patterns, and tree-based algorithms are efficient and easy to interpret. By utilizing both types of algorithms, we can create a reliable and flexible system for IoT apps.

To effectively implement these algorithms in real-world IoT applications, more research is necessary. The focus should be on optimizing resources and achieving scalability. Additionally, analyzing their performance within a federated learning framework would provide valuable insights for secure distributed IoT systems.

REFERENCES

[1]. Aziz UllahKarimy1, ,Dr. P Chandrasekhar Reddy2. Securing the Internet of Things: A Study on Machine LearningBased Solutions for IoT Security and Privacy Challenges. ZKG International [Internet]. 2023;8(2). Available from: https://zkginternational.com/archive/volume8/Securing-the-Internet-of-Things-A-Study-on-Machine-Learning-Based-Solutions-for-IoT-Security-and-Privacy-Challenges.pdf

[2]. Lokesh Babu C VM. Why do tree-based models still outperform deep learning on tabular data? 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT). 2022;417–22.

[3]. C LB, M V. Review on Various Machine Learning Algorithms Implemented in IoT Security. In: 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT) [Internet]. Kannur, India: IEEE; 2022 [cited 2023 Nov 8]. p. 417–22. Available from: https://ieeexplore.ieee.org/document/9917738/

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 5, March 2024

[4]. Jamalipour A, Murali S. A Taxonomy of Machine-Learning-Based Intrusion Detection Systems for the Internet of Things: A Survey. IEEE Internet Things J. 2022 Jun 15;9(12):9444–66.
[5]. Tasnim A, Hossain N, Parvin N, Tabassum S, Rahman R, Iqbal Hossain M. Experimental Analysis of Classification for Different Internet of Things (IoT) Network Attacks Using Machine Learning and Deep learning. In: 2022 International Conference on Decision Aid Sciences and Applications (DASA) [Internet]. Chiangrai, Thailand: IEEE; 2022 [cited 2023 Nov 8]. p. 406–10. Available from: https://ieeexplore.ieee.org/document/9765108/
[6]. Bekkouche R, Omar M, Langar R, Hamdaoui B. Ultra-Lightweight and Secure Intrusion Detection System for Massive-IoT Networks. In: ICC 2022 - IEEE International Conference on Communications [Internet]. Seoul, Korea, Republic of: IEEE; 2022 [cited 2023 Nov 8]. p. 5719–24. Available from: https://ieeexplore.ieee.org/document/9838257/
[7]. Garcia S, Parmisano A, Erquiaga MJ. IoT-23: A labeled dataset with malicious and benign IoT network traffic [Internet]. Zenodo; 2020 [cited 2023 Nov 8]. Available from: https://zenodo.org/record/4743746
[8]. Kayode Saheed Y, Idris Abiodun A, Misra S, Kristiansen Holone M, Colomo-Palacios R. A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Engineering Journal. 2022 Dec;61(12):9395–409.
[9]. Choudhary S, Kesswani N. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. Procedia Computer Science. 2020;167:1561–73.
[10]. Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS) [Internet]. Canberra, Australia: IEEE; 2015 [cited 2023 Nov 8]. p. 1–6. Available from: http://ieeexplore.ieee.org/document/7348942/
[11]. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications [Internet]. Ottawa, ON, Canada: IEEE; 2009 [cited 2023 Nov 8]. p. 1–6. Available from: http://ieeexplore.ieee.org/document/5356528/
[12]. Thavamani S, Sinthuja U. LSTM based Deep Learning Technique to Forecast Internet of Things Attacks in MQTT Protocol. In: 2022 IEEE Fourth International Conference on Advances in Electronics, Computers and Communications (ICAECC) [Internet]. Bengaluru, India: IEEE; 2022 [cited 2023 Nov 8]. p. 1–4. Available from: https://ieeexplore.ieee.org/document/9716585/
[13]. Selvapandian D, Santhosh R. Deep learning approach for intrusion detection in IoT-multi cloud environment. Autom Softw Eng. 2021 Nov;28(2):19.
[14]. Mitchell, Tom M. Machine learning. Vol. 1. McGraw-hill New York; 1997.
[15]. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. 2016 [cited 2023 Nov 8]; Available from: https://arxiv.org/abs/1603.02754
[16]. Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization: In: Proceedings of the 4th International Conference on Information Systems Security and Privacy [Internet]. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications; 2018 [cited 2023 Nov 8]. p. 108–16. Available from: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116
[17]. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. IEEE Access. 2022;10:40281–306.
[18]. Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, et al. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Comput. 2018 Jul;17(3):12–22.