A Distributed Intrusion Detection System using Machine Learning for IoT based on ToN-IoT Dataset

Abdallah R. Gad[1], Mohamed Haggag[2], Ahmed A. Nashat[3], Tamer M. Barakat[4]
Department of Communication and Electronics Engineering
October High Institute for Engineering & Technology, 6th of October City 12596, Egypt[1]
Department of Electronics and Communication Engineering
Misr University for Science and Technology, 6th of October City 12566, Egypt[2]
Electrical Engineering Department, Faculty of Engineering, Fayoum University, Fayoum, 63514, Egypt[1, 3, 4]

***Abstract*—The internet of things (IoT) is a collection of common physical things which can communicate and synthesize data utilizing network infrastructure by connecting to the internet. IoT networks are increasingly vulnerable to security breaches as their popularity grows. Cyber security attacks are among the most popular severe dangers to IoT security. Many academics are increasingly interested in enhancing the security of IoT systems. Machine learning (ML) approaches were employed to function as intrusion detection systems (IDSs) to provide better security capabilities. This work proposed a novel distributed detection system based on machine ML approaches to detect attacks in IoT and mitigate malicious occurrences. Furthermore, NSL-KDD or KDD-CUP99 datasets are used in the great majority of current studies. These datasets are not updated with new attacks. As a consequence, the ToN-IoT dataset was used for training and testing. It was created from a large-scale, diverse IoT network. The ToN-IoT dataset reflects data from each layer of the IoT system, such as cloud, fog, and edge layer. Various ML methods were tested in each specific partition of the ToN-IoT dataset. The proposed model is the first suggested model based on the collected data from the same IoT system from all layers. The Chi2 technique was used to pick features in a network dataset. It reduced the number of features to 20. Another feature selection tool employed in the windows dataset was the correlation matrix, which was used to extract the most relevant features from the whole dataset. To balance the classes, the SMOTE method was used. This paper tests numerous ML approaches in both binary and multi-class classification problems. According to the findings, the XGBoost approach is superior to other ML algorithms for each node in the suggested model.**

Keywords*—Intrusion detection system (IDS); internet of things (IoT); ToN-IoT dataset; machine learning (ML)

I. INTRODUCTION

The internet of things (IoT) is a network of everyday physical objects which can communicate and synthesize data utilizing the current network infrastructure by connecting to the internet. These things are networked digital appliances or sensors that can collect data and transmit it across the internet. New applications and services are created as a result of the interplay between sensors, connectivity, people, and processes. In the IoT, these digital sensors or devices are known as "things" [1].

As IoT networks become more prevalent, they become more prone to security breaches. Cyber security attacks are one of the most common serious threats to IoT security. These attacks take several forms and target various resources on a wide range of IoT devices. These cyberattacks frequently target a large number of devices in an IoT network, which may then be utilized as a "resource" or "platform" for attacks such as distributed-denial-of-service (DDoS) and fraudulent operations such as ransomware and password attacks. As a result, securing IoT devices and building malicious (intrusion) detection models for IoT systems is becoming increasingly vital for protecting sensitive data[2].

Because the security of broad IoT is vital, it is critical for identifying IoT risks and defining current prevention methods. This work outline and categorize IoT security threats categories and common defense methods to give the reader the security foundation they need to understand the work[2]. The following are some of the reasons why IoT security procedures vary from those used in traditional security systems:

- IoT systems' computational power, memory capacity, battery life, and network bandwidth are all restricted. As a result, existing standard security solutions, which are often expensive in terms of resources, cannot be installed.
- IoT systems are widely scattered and heterogeneous. As a result, traditional centralized solutions may be ineffective. Furthermore, the distributed nature of IoT introduces new obstacles and restrictions to its security.
- IoT solutions are employed in an ever-changing physical context. Physical attacks, as a result, have been added to the list of typical security issues.

One of the IoT security protection solutions is IDS. An IDS [3] is software and/or hardware that monitors a network or system for hostile activity and issues quick alarms. Since 1970 IDSs have been used. They are divided into two categories: i) deployment and ii) detecting methods. There are two types of IDS deployments: HIDS and NIDS. HIDS (Host-based Intrusion Detection System) is an intrusion detection system installed on a host machine (i.e., a device or a Thing). They keep track of and evaluate system application files and the operating system. Insider intrusion detection and prevention are best accomplished through HIDS. NIDS (Network-based

Intrusion Detection System) captures and analyzes network packet traffic. To put it another way, they are sniffing packets. NIDSs are effective against external intrusion attacks. After the intrusion has occurred, the following situation will explore. An exemplary detection system is one that rapidly recognizes the compromised condition and minimizes the loss (s). IDSs come in many different forms. all categories of IDSs are presented in [4].

Signature detection(knowledge-based)[5], Anomaly detection (behavior-based) [6], and Hybrid detection are the different detection types of IDSs.

An intrusion prevention system (IPS) [7] is used to keep off intruders. An IPS reacts quickly and prevents harmful traffic from passing through by deleting sessions, restarting sessions, blocking packets, or proxying traffic. On the other hand, an IDS replied once an attack has been detected. Inline detection, layer seven switches, deceptive systems, application firewalls, and hybrid switches are all examples of IPS.

During the study experimentation, the following stages will be followed:

1) Choosing ToN-IoT as a new dataset [8]. The dataset was thoroughly analyzed by removing the flow identification attributes to eliminate bias and overfitting and preprocessing the data.

2) The ToN-IoT dataset also has many issues, including class imbalance, categorical attributes, and missing values. For challenges using the ToN-IoT dataset, a hybrid approach was provided.

3) The ToN-IoT dataset was utilized to test several machine learning (ML), which are: naïve bias (NB), logistic regression (LR), decision tree (DT), k-Nearest Neighbor (KNN), support vector machine (SVM), random forest (RF), XGBoost, Adaboost.

The subsequent are the research's major contributions:

1) Propose a distributed machine learning IDS for IoT with comparison to another research.

2) Most existing detection algorithms are evaluated using the NSL-KDD, KDD-CUP99, and UNSW-NB15 datasets. These databases are out of date and do not cover current IoT threats. However, the effectiveness of the proposed model is evaluated using an actual ToN-IoT dataset. As assessment measures, accuracy, precision, recall, F1-Score, and false-positive rate (FPR) are utilized.

3) Resolving issues with the ToN-IoT dataset, such as class imbalance, missing values, and irrelevant attributes that impact the IDS model's performance.

4) The Chi^2 and correlation matrix were used to select the most important attributes.

5) The class imbalance problem was solved using the SMOTE approach.

The paper organization is as follows: The sections below provide a short overview of IDS for IoT. The ToN-IoT dataset is briefly described in Section III. In Section IV is described the experimental techniques. In Section 5, the findings of the experiments are discussed. Finally, in Section VI, the conclusion is offered.

II. Related Work

Previously, many machine learning methods were used to malicious datasets in malicious intrusion detection research. IoT devices, as previously stated, are lightweight and low-powered devices with limited computing ability to run traditional antimalware solutions [9]. As a result, research is undertaken to address these issues.

Alhanahnah et al [10] set out to solve this problem by developing IoT malware detection technologies that could operate effectively on any platform while being lightweight despite resource limits. Lightweight signatures were created from high-level code to create the suggested solution. The investigation proved that the signature generating mechanism is effective. The proposed method was found to have an 85.2 % detection rate with zero false positives using analytical approaches.

Ngo and Nguyen [11] also investigated the increase of malware targeting IoT sensors and enhanced the efficacy of current malicious software detection techniques. The research looked at several prior studies on IoT security. The pros and cons of various malware detection technologies were examined and contrasted. The study discovered that the ELF-header approach had a low false detection rate of 0.2 % using tabular comparisons. Furthermore, employing the coding scheme to combine malware samples enhanced detection accuracy to over 98 %, according to the data.

Su et al. [12] suggested a lightweight technique to detect and classify DDoS malware and normal IoT applications. The research used a convolutional neural network to conduct experiments, allowing resource-constrained IoT devices to function normally. The correctness of the suggested design was tested using a five-fold validation technique. With an average accuracy of 94 %, the recommended design predicted malware.

Nguyen et al. [13] also assessed the efficacy of three deep learning-based techniques in identifying IoT malicious software. The models were developed using attributes 1) fixed-size-byte series, 2) fixed-size-color image, and 3) variable sized-sequence data. The fixed-size byte sequence strategy was less accurate than the variable-sized and fixed-size color picture approaches, with an accuracy of 90.58 %. However, the study was regarded as initial, and the authors recommended that further tests be conducted to increase the accuracy.

Alasmary et al. [14] used multiple datasets to see the relationships and variations between malicious software on various systems. A method was constructed and utilized for categorizing android malware, IoT malware, and non-threatening samples using control flow graphs. A 10-fold validation procedure was used to assess the performance of these models. The convolutional neural network (CNN) model was shown to have a 99.66 % accuracy in detecting IoT malware from normal samples in the study.

Hasan et al. [15] used artificial neural networks (ANN), SVM, and LR as machine learning algorithms to identify attacks and abnormalities in IoT sensors at IoT sites. With a

99.4 % accuracy, the experiment suggests that CNN is the preferable approach to apply in IoT for intrusion detection systems.

Authors [16] suggested a revolutionary real-time, distributed, and lightweight IDS, efficiently combining edge, fog, and cloud computing. It enables sophisticated data processing at the intermediaries' level, decreasing the amount of data sent to the cloud. As a result, processing occurs at hubs, routers, or gateways. The IDS's AIS architecture is made up of three parts:

A training engine: trains detectors using data from an initial learning dataset. Because it necessitates complicated and powerful processing units, this stage is handled on the cloud layer.

An analyzer examines abnormalities provided by detectors in order to warn and reject false positive signals. To increase precision, the authors apply memory cell detectors and genetic algorithms. The analyzer engine is installed at the fog layer since this stage necessitates a greater connection between the infected edge nodes and the main engine.

Detector sensors: Each node in the network is equipped with detecting logic. Various detectors might recognize each form of attack in the suggested IDS, which is intelligent and distributed. When a threshold is exceeded, the anomaly is transmitted to the analyzer engine, resulting in an intrusion warning.

The essential work strengths are as follows: a) Combination of innovative analysis in the cloud with lightweight analysis in the fog-layer; b) botnet attacks are detected using a smart strategy, and c) detection of zero-day attacks and unknown attacks based on an online self-training method. The lightweight IDS efficiency was evaluated using two datasets: SSH Brute Force dataset, and KDD-Cup99. According to testing data, the three-layered suggested approach achieves a 3.51 % false-positive rate (FPR) with 98.35% and 97.83% precision.

A real-time combination of specification-based and anomaly-based IoT IDS was proposed by Bostani and Sheikhan [17]. It may be used to identify sinkholes and selective-forwarding attacks in 6LowPAN networks. This IDS operates in two stages: router-level specification detection and root-level anomaly detection. First, the routers examine aspects of the traffic of a network and host nodes on a local level. The first phase's findings are forwarded to the root node for the second step and then deleted from routers to save memory and CPU resources. At the root node, the second phase is global intrusion detection, which involves performing anomaly-based analysis on entering data packets. To demonstrate appropriate real-time detection, they use three main experimental tests, each with ten simulations: the first contracts with assessment values, the second with network scale (small and medium-size) to confirm independent scale-network IDS, and the third with the option to extend detected attacks such as wormhole. According to the results of simulated situational experiments, the proposed hybrid technique may reach a true positive rate of 76.19 % and a FPR of 5.92 %.

Moustafa et al. [9] introduced an ensemble NIDS based on existing statistical characteristics to reduce harmful events, including botnet cyberattacks against HTTP, MQTT, and DNS protocols in IoT systems. The model has three phases: a) Using a thorough study of the TCP/IP model, a collection of attributes is derived from the network traffic protocols. The authors used the Bro-IDS tool for the basic characteristics and created a new extractor module (that collaborates with Bro-IDS) to derive further statistical aspects of transactional processes. b) The correlation coefficient is applied to the result attributes in this step-in order to obtain the essential ones. This phase allows NIDS's computational cost to be reduced. An ensemble technique using the AdaBoost (Adaptive Boosting) algorithm disperses the network data. Then, to identify attacks, Decision Tree (DT), NB, and ANN ML algorithms are used. When compared to individual ML algorithms, the AdaBoost technique improves detection performance. It is capable of dealing with any situation through the computation of an error function, the minor differences of the feature vectors are used to learn and select which learners can correctly categorize each instance of the input data, and the error function is assigned to each occurrence. They used the UNSW-NB15 and NIMS botnet datasets. The ensemble approach achieved between 95.25 % and 99.86 % of DR and 0.01% to 0.72 % of FPR.

Nguyen et al. [18] presented a self-learning anomaly-based IDS (DoT) that was autonomous. Their solution consists of Security Gateways that monitor system devices, as well as an IoT security service (which might be a service provider) that detects abnormalities in a device-type-specific mode. Network devices are automatically grouped based on their manufacturer's hardware and software specifications. Then, for each device type, anomaly models will be created. According to the authors, the used algorithm is Gated Recurrent Units (GRU), which is a recurrent neural network (RNN) that can train efficiently with little training data. As a result, final GRU models result from a collective learning process involving multiple Security Gateways while maintaining privacy. This IDS approach appears to be communication-efficient and ideal for distributed systems such as the internet of things. In order to identify the Mirai virus, the authors test their approach in a real-world smart home deployment. They have a DR with 95.6 % and no false alarms in 257 ms.

Illy et al. [19] presented a fog-to-things IDS architecture. The detecting method is implemented on two levels: the fog and cloud layers of the system. On the one hand, this design enables the authors to deal with their computationally demanding ML detection induced by ensemble learning (a combination of ML algorithms). On the other side, owing to fog detection provides for low latency detection and, consequently, quick reaction. As a result, anomaly detection is conducted first in the fog layer; if the traffic is detected as an attack, an alert is delivered to the security administrator. Additional analysis is performed in the cloud to categorize the kind of attack and provide it to him. They used a multi-expert mode and a multi-stage technique to evaluate alternative ML combinations. On the NSL-KDD dataset, they achieved 85.81 % overall accuracy for binary classification and 84.25 % overall accuracy for attack classification, respectively.

TABLE I. A SUMMARY OF APPROACHES FOR IOT MALICIOUS DETECTION

Authors	Study purpose	ML-methods	Data used for Evaluation
Alhanahnah [10]	Developing IoT malware detection technologies that could operate effectively on any platform while being lightweight despite resource limits.	Statistical technique	IoT malware dataset
Su [12]	Detecting and classifying DDoS	CNN	IoT DDoS malware dataset
Nguyen [13]	Identifying IoT malware.	Deep learning	They prepare their own data
Alasmary [14]	Detection of IoT network attacks.	KNN- ID3- Random Forest- AdaBoost- Multi-layer perceptron (MLP)- Naïve Bayes (NB)	Bot-IoT
Hasan [15]	Identifying attacks and anomalies	Neural Networks (NN), SVM, and logistic regression	Data collected from Kaggle
Hosseinpour [16]	Distributed IDS	Artificial Immune System (AIS)	KDD99 and SSH Brute Force from ISCX
Bostani [17]	A mixture of specification-based and Anomaly-based IDS	Unsupervised-Optimum Path Forest	NSL-KDD
Moustafa[9]	An ensemble NIDS	Decision Tree (DT), NB, and ANN	UNSW-NB15
Nguyen [18]	A self-learning anomaly-based IDS	a recurrent neural network	Real-world smart home
Illy [19]	A fog-to-things IDS	ensemble learning	NSL-KDD
Gonzalo et al. [20]	Detecting IoT attacks	CNN, LSTM	KDD99, NSL-KDD, CISC2010
Shafiq [21]	A new-feature selection algorithm.	Decision Tree (C4.5)- SVM-RF-NB	BoT-IoT

Gonzalo et al. [20] demonstrated an embedded IoT micro-security that uses a CNN prototype to identify URL-based cyberattacks on a client's IoT devices. For botnet detection, the add-on works in concert with an RNN-LSTM model housed on the back end servers. With an accuracy of 94.30 % and an F-1 score of 93.58 %, CNN can detect phishing attacks. Botnet attacks are detected using LSTM with a 94.80% accuracy when all malicious in the dataset are utilized.

Shafiq et al. [21] demonstrated a malicious intrusion model for IoT. this study developed a new feature selection technique and tested the newly developed technique on several machine learning techniques such as s, C4.5 decision tree, and Random Forest classifiers; they got more than 95% accuracy. A summary of algorithms that are used to construct IDS for IoT is presented in Table I.

III. STATE-OF-THE-ART DATASETS

KDD99 and NSL-KDD are the most extensively used NIDS/HIDS datasets. For assessment and testing, public attack datasets such as CAIDA [22], DEFCON [23], ADFA IDS[24] , KYOTO [25], and ISCX 2012 [26] are accessible. The most recent are either unlabeled data or unavailable data from certain nations, or data exclusive to a domain.

KDD99 [27] is a dataset used for constructing the robust NIDS for detecting "dangerous" connections from "great" ones. The dataset is a feature-extracted edition of the DARPA dataset. KDD99 comprises data from a military network with inserted attacks that are divided into four categories: i) DoS; ii) remote to user; iii) user to root; iv) probing. Using the Bro-IDS tool, KDD99 is based on 41 attributes for each sample+, as well as the class label. The attributes are divided into four categories [27]:

- 1–9: the fundamental attributes of each TCP connection.
- 10-22: attributes suggested by domain knowledge.
- 23-31: a two-second time frame was used to calculate traffic attributes.
- 32-42: host capabilities are intended to evaluate cyberattacks lasting longer than two seconds.

KDD99 is common and widely used for experimental analysis by security researchers. Various efforts [28, 29] were created to minimize the number of characteristics by picking the most important ones from the initial 41. However, numerous studies, such as [30, 31], have found KDD99 to have drawbacks, a few of the most notable ones:

The probability distributions of the testing and training sets diverge. To put it another way, KDD99 has imbalanced categorization.

- The data set is no longer current (1999).
- New attacks are not available.
- The data collected are not from an IoT system.

NSL-KDD [32] is an improved version of KDD99 that addresses its shortcomings. First, duplicate entries are deleted from the whole dataset. Second, a range of samples from the original KDD99 was chosen to acquire accurate findings from classifier systems. Third, the issue of an uneven probability distribution is no longer an issue. The lack of current low-footprint attack scenarios is a fundamental flaw in this collection.

UNSW-NB15 [33] was built by the Australian Centre for Cyber Security in 2015. Its purpose is to create a mixture of modern real-world activities and synthetic modern attacks behaviors. There are around two million and 540,044 records in four CSV files. Those records were created from 100GB of raw data recorded using the tcpdump utility (in pcap files).

The IoT dataset by Sivanathan et al. [34, 35]. Deals with categorization for IoT sensors based on network traffic characteristics. The authors provide a brilliant ecosystem for 28 IoT sensors, including cameras, lighting, plugs, motion sensors, appliances, and health-monitoring devices. They use statistical analysis to provide essential insights into network traffic patterns utilizing attributes such as port numbers, activity cycles, signaling patterns, and encryption suites. They also synthesized network traffic traces from their infrastructure for six months and provided them to the scholarly community.

The CICIDS database [36] is a recent Intrusion dataset provided by the Canadian Institute for Cyber-security, to represent the most recent attacks similar to real-world data. It was created using HTTP, HTTPS, FTP, SSH, and e-mail protocols to model the abstract behavior of 25 users. CIC-FlowMeter examines the data, including labeled data based on timestamps, starting, and ending IP addresses, ports, protocols, and attacks. The authors developed the B-Profile technique to describe the behavior of FTP, SSH, HTTP, HTTPS, and e-mail protocols in order to simulate realistic traffic. While capturing the data, the authors used Brute Force FTP, SSH Heartbleed, and DDoS attacks. Unlike current standard IDS datasets, the assessment system [37] identified eleven critical attributes required to develop a valid benchmark dataset.

The CSE-CIC-IDS 2018 [38] dataset is a one-of-a-kind IDS dataset that has emerged to replace poor datasets that restrict IDS/NIDS experimental assessments. CSE-CIC-IDS2018 is an anomaly-based dataset containing intrusions in the network to overcome the usage of signature datasets: a) dos; b) Heartbleed; c) botnet; d) brute-force; e) DDoS; f) and web attacks were among the seven attack scenarios mentioned by the authors. The attack architecture consists of 50 nodes, whereas the target organization is divided into five departments, each with 30 servers and 420 hosts. CICFlowMeter-V3 was used to extract 80 characteristics from network traffic and system logs.

BoT-IoT [39] The ACCS Cyber Range Lab created a network environment based on IoT that includes both regular and botnet traffic. The Ostinato and Node-red tools were used to create IoT and non-IoT network traffic, respectively. The Argus program was used for extracting the dataset's original 42 attributes from a total of 69.3GB of pcap files. The collection comprises 477 normal flows (0.01 %) and 3,668,045 assault flows (99.99 %), totaling 3,668,522 flows. The dataset contains traffic from DDoS, DoS, OS, Data exfiltration, Keylogging attacks, and further DDoS and DoS operations put up on the protocol utilized. The main lack in this dataset is that it comprises over 99 % botnet traffic but just about 1% regular traffic.

TABLE II. AVAILABLE DATASETS

Dataset	Dataset Advantages	Dataset Disadvantages
KDD99	KDD99 is the most widely deployed dataset. • Data that has been labeled and has 41 attributes for each connection and the class description. • DOS, remote-to-user, user-to-root, and probing attacks are all used. • (PCAP) network traffic is provided.	• The dataset is out of date, and KDD99 has imbalanced classes. • Not for the internet of things (IoT) systems.
NSL-KDD	• It is an upgraded version of KDD99 that addresses the limitations of KDD99. • no duplicated records in the training and test sets.	•There are not enough current low-footprint attack scenarios. • Not for the internet of things (IoT).
UNSWNB15	It offers real-world modern regular activities and synthetic modern attack behaviors. • Network traffic (PCAP) and CSV files are available.	Because recent attacks and typical network traffic behave similarly, it is more complicated than the KDD99 dataset.
Sivanathan Dataset	• This IoT network traffic dataset is based on a real-world IoT network. • CSV and PCAP files are available.	• Data that has not been tagged. • No attack data is required for the IoT device proliferation and traffic characterization.
CICIDS	•labeled network flows used for building IDS based on machine learning. • PCAP and CSV files are available. • Brute Force, DoS, Heartbleed, Web-Attack, Botnet, and DDoS cyberattacks.	• Not public. • Not for the internet of things (IoT).
CSE-CICIDS2018	• PCAP, CSV, and log files are available. • Brute-force, Botnet, DoS, DDoS, and Web attacks are all implemented. • It is a dynamically produced dataset that may be modified, extended, and replicated.	• Not public. • Not for the internet of things (IoT).
BoT-IoT	• IoT network traffic dataset. • PCAP and CSV files are available.	• The main lack in this dataset is that it comprises over 99 % botnet traffic but just about 1% regular traffic.

KDD99 is the most often used network dataset, as indicated in Table II. Since 1999, it has been in use. It is, unfortunately, out of date. NSL-KDD was built to overcome the limitations of KDD99. There are no duplicate records, and the data is balanced. UNSW-NB15 was proposed as a replacement for NSL-KDD, which lacks contemporary attacks. It is a well-known dataset that has been subjected to recent attacks. Meanwhile, when it comes to similarities between new attacks and normal activities, it is more complicated than KDD99. The Sivanathan et al. dataset, CSE-CIC-IDS 2018, and CICIDS are examples of more recent network datasets. Compared to the other datasets offered, Sivanathan's work is the only one that includes IoT network traffic. It is, however, intended for the proliferation of IoT devices rather than intrusion detection. CICIDS and CSE-CIC-IDS 2018 have labeled records but do not target IoT system security despite having an up-to-date attack list.

IV. TON-IOT DATASET

The ToN-IoT dataset was used in this investigation. The ToN-IoT includes telemetry data from linked devices, Linux operating system data, Windows operating system logs, and IoT network traffic, among other data sources acquired from the entire IoT system. A medium-scale IoT network provides diverse data. ToN-IoT was designed by the UNSW Canberra IoT Labs and the Cyber Range. The ToN-IoT repository contains the ToN-IoT dataset [40]. Furthermore, the ToN-IoT was represented in CSV format with a labeled column indicating attack or normal and a sub-category attack-type. Various types of cyberattacks, such as ransomware, password attack, scanning, denial of service (DoS), distributed denial of service (DDoS), data injection, backdoor, Cross-site Scripting (XSS), and Man-In-The-Middle (MITM) were represented. Various IoT and IIoT sensors were targeted in these attacks, launched, and gathered across the IIoT network. The dataset's details may be found in [40].

1) ToN-IoT network dataset: The network ToN-IoT dataset contains 44 attributes and a label classified as normal or attack for each data point. Fig. 1 shows the statistics for network data samples in the train-test ToN-IoT dataset.

2) ToN-IoT Linux dataset: Linux datasets are partitioned into three categories disk, memory, and process CSV files. The first CSV file contains attributes for disk usage in normal behavior and attack. The second CSV file is related to memory activity, containing ten attributes, a label column labeled as normal or attack, and an attack-type containing attack type (DoS, DDoS). The last file belongs to processes in Linux operating system. The Linux process ToN-IoT contains 14 attributes and an attack type for each data point. Fig. 1 shows the statistics for all Linux data records in ToN-IoT.

3) ToN-IoT Windows dataset: Windows datasets are contained records for windows 7 & 10. Windows 7 CSV file contains 133 attributes, labeled as normal or attack, and attack-type containing attack type (DoS, DDoS). Windows 10 CSV file contains 125 attributes, labeled as normal or attack and attack-type, which contain attack type (DoS, DDoS). The statistics for all Windows data in the ToN-IoT were presented in Fig. 1.

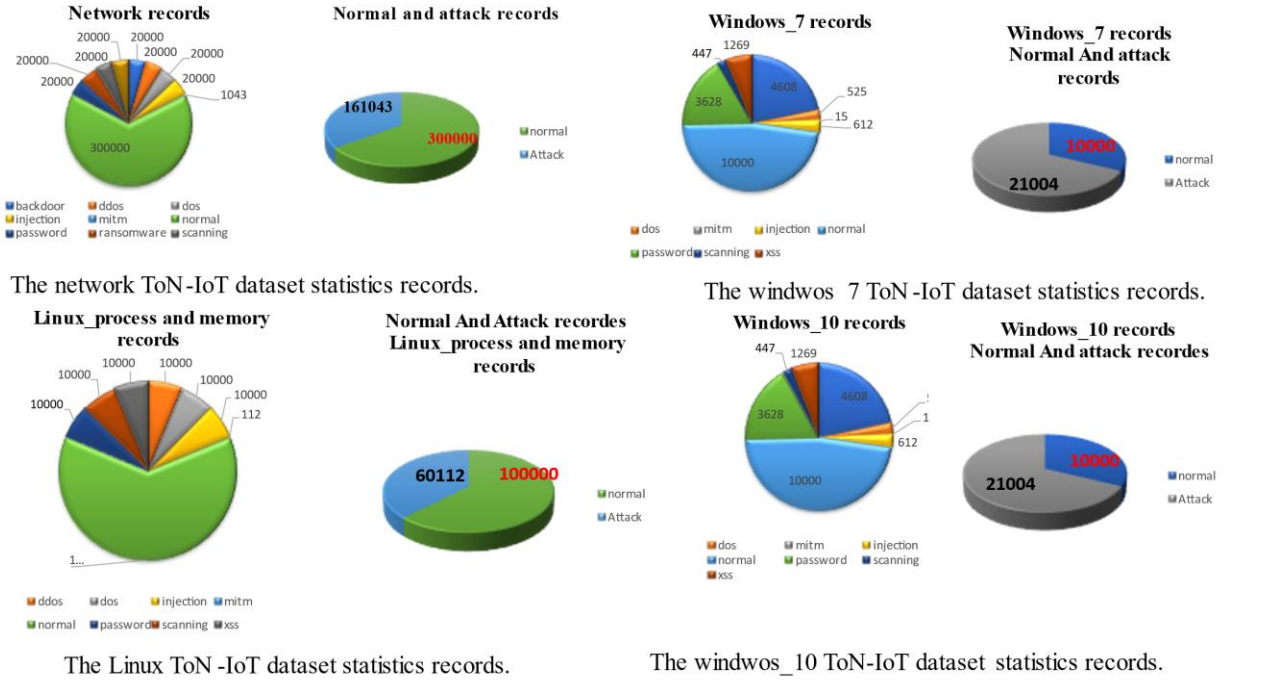

Fig. 1. ToN-IoT Statistics.

V. DISTRIBUTED IDS

This part of work discusses the architecture of the proposed model, system components of the proposed detection model, and the various detection nodes in proposed model.

A. Model Architecture

The primary purpose of the suggested detection system for IoT networks is to make on-demand security services more convenient while also preventing attacks. The proposed detection system employs a machine learning for detecting attacks in the network traffic within the IoT network and in all other nodes in IoT systems. As shown in Fig. 2, the proposed detection system functions primarily in various stages/phases - cloud phase, fog-network-detection phase, and a fog-host detection phase.

B. Machine Learning Methods

The ToN-IoT dataset has been used to evaluate various machine learning (ML) approaches. The chosen algorithms are utilized for training, and testing ML approaches with various parameters in the preprocessing phase of data for intrusion detection. The accuracy, precision, recall, F1-Score, false-positive rate (FPR), and confusion matrix were used to evaluate the different classifiers. The methodologies utilized have demonstrated great performance in the production of IDSs and have proven to be successful in various industries. This study look at the logistic regression (LR), naïve bias (NB), support vector machine (SVM), decision tree (DT), random forest (RF), k-Nearest Neighbor (KNN), Adaboost, and XGBoost techniques, among others [41], [42].

C. Model Nodes

In this section each node in the proposed model will discussed.

1) Malicious network detection node (Cloud layer): The framework for malicious/intrusion detection comprises the default procedure in machine learning:

a) Data preprocessing and feature engineering.

b) Training machine learning models.

c) Evaluate the selected model.

The deployed IDS in the cloud was established using various feature-engineering techniques discussed in the next section.

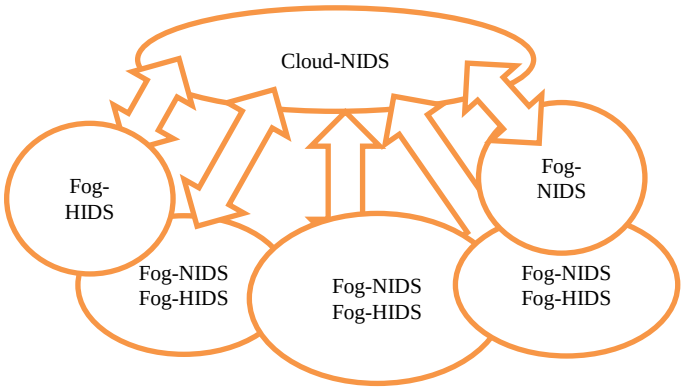

Fig. 2. The Proposed Detection System.

ToN-IoT preprocessing: Filtering and preparing are the most critical steps before supplying data into machine learning in order to achieve high performance. The used dataset has numerous obstacles, including missing values, categorical characteristics, and class imbalance. Unnecessary attributes may impact the performance of the chosen ML algorithms. Using permutations of multiple preprocessing and normalizing strategies, this work tested the selected ML algorithms using several preprocessing techniques.

- Missing value imputation: Missing values are common in the ToN-IoT. These missing values must be addressed appropriately. In the proposed model, the imputation of missing values is substituted with the most frequent value in each feature containing missing data. A second an approach was imputed numerical features using mean value.
- Converting categorical attributes to numerical: There are various categorical attributes in the ToN-IoT dataset. Numerical values must be assigned to the category characteristics. One-hot encoding was used to achieve this goal.
- Class-imbalance: The SMOTE technique was utilized to balance the classes in the used dataset. The ToN-IoT dataset is plagued with class imbalance distributions. Solutions to the imbalanced problem, oversampling, under-sampling, and hybrid techniques were proposed. Oversampling is the practice of duplicating the minority class points. Several researchers utilized it. However, this approach has the drawback of overfitting these spots. Others employ under-sampling, which reduces the dominating class's score. The problem with this method is that some of the elements that have been removed may be necessary for accurately portraying the class. A hybrid strategy was used; it duplicates minority class points while removing certain majority class points. Synthetic Minority Oversampling Technique (SMOTE) [43], [44] enhances basic random oversampling by providing synthetic minority class samples, addressing the overfitting problem that can occur with simple random oversampling. SMOTE creates new data points instead of replicating old ones. A linear combination of two comparable minority samples is used to generate extra minority data points.
- Several attributes such as timestamp, IP-address, source-port, and destination_port were removed from the dataset since they may cause overfitting.

The two key steps used during feature-engineering development are preprocessing based on the mentioned dataset challenges and data normalization. For high performance, ML approaches are evaluated using a variety of feature-engineering techniques:

Feature selection: Various aspects must be checked for intrusion detection, some of features will be valuable while others will be useless. The feature selection procedure is assigning a score to each potential feature and picking the best (k) attributes. A function of both is obtained by counting the frequency of a feature in training for each positive and negative

class occurrence separately. Non-essential attributes are removed, increasing accuracy, decreasing computation time, and reducing the overfitting problem, resulting in better performance. The used feature selection technique was Chi^2, which is a filter method.[31], [45].

$$X^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(O_i-E_i)^2}{E_i} \quad (1)$$

where m indicates the no of attributes, n indicates the no of classes, O_i is the observed frequency, and E_i is the expected frequency.

- Data normalization: The ToN-IoT contains attributes with varying values, and some attributes have larger values than others. Because a technique may be slanted toward characteristics with larger values, differing values out of range might lead to inaccurate results. As a result, data normalization is critical in preventing outweighing attributes with greater values over attributes with smaller values by scaling the feature vector. Min-Max is used to scale data between [0:1] as presented in Eq. (2).

$$Z = \frac{x-x_{min}}{x_{max}-x_{min}} \quad (2)$$

where x reflects the feature-value, Z reflects the feature-value after normalization, the maximum and minimum values of the feature are x_{max} and x_{min}.

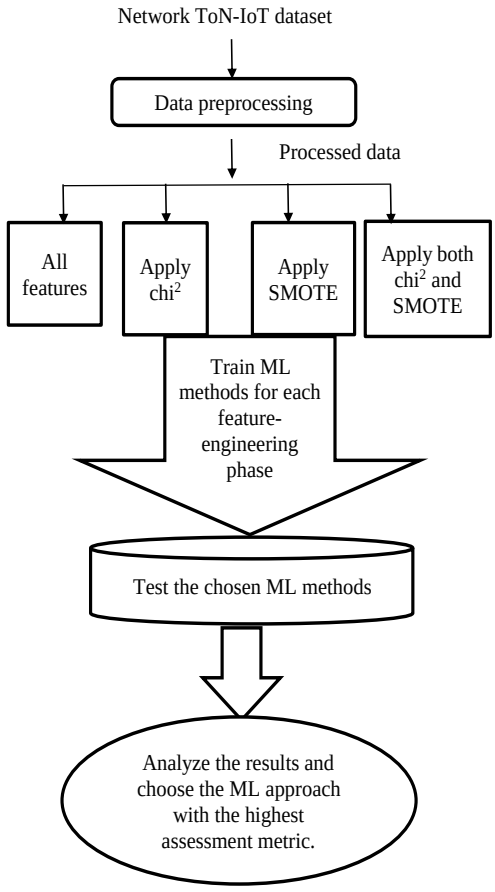

Fig. 3. The Procedure to Evaluate Network Dataset [41].

The training process: All ToN-IoT datasets are in CSV extension; initially, the ToN-IoT dataset was divided into two sets. The first set comprises training with 70% of the dataset. The second set contains unseen data for evaluating the performance of the selected ML algorithms. Before employing any preprocessing to the ToN-IoT, the splitting step was completed to avoid data leaking. The effectiveness of the chosen machine learning algorithms is evaluated using a variety of assessment measures, which will be provided in the next section. The previous steps associated with evaluating the performance of various ML algorithms utilizing ToN-IoT datasets are summarized in Fig. 3 and 4.

Classifier performance evaluation: Based on the ToN-IoT dataset, numerous metrics were utilized to assess the efficacy of various machine learning approaches. The chosen assessment techniques were chosen because they provide a detailed explanation of the outcomes for machine learning-based malicious detection [46].

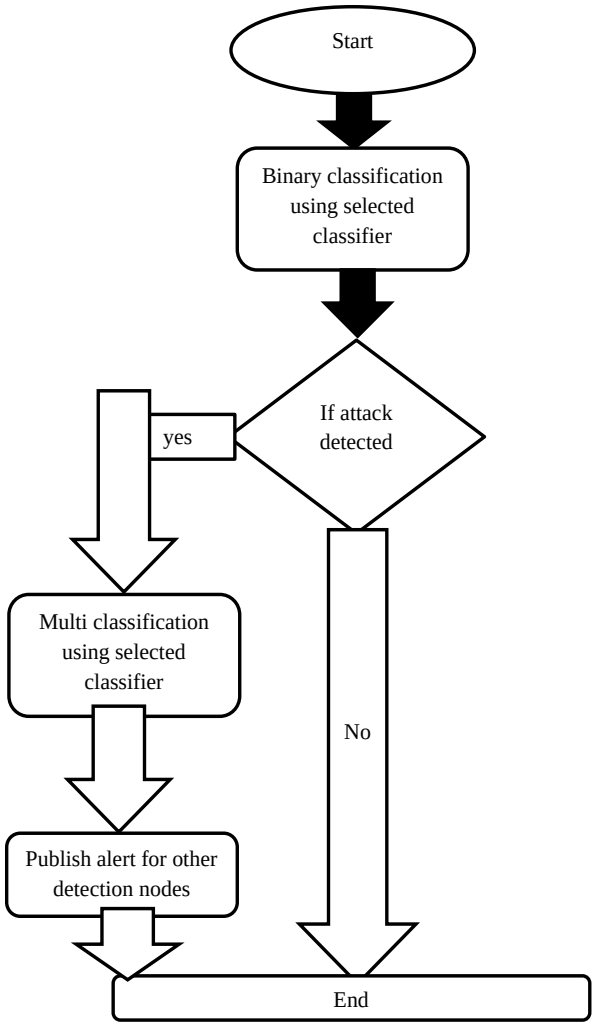

Fig. 4. The Overall Process for Malicious Detection.

The first metric is accuracy, which measures a technique's overall efficiency as a proportion of instances accurately identified as normal or attacks. The precision metric, which shows the proportion of accurately recognized attacks out of all

detected attacks, is the second metric. The third metric is recall, which represents the proportion of properly recognized attacks in the test dataset as a fraction of all attacks. The fourth metric is the F1-score. The final metric was the false-positive rate (FPR) [46]. These carefully chosen metrics are defined as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

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

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

$$\text{F1} - score = 2 \times \frac{(precison * recall)}{(precision + recall)} \quad (6)$$

$$\text{FPR} = \frac{FP}{FP+TN} \quad (7)$$

Where true positive (TP) is the total number of actual attack records that are correctly identified as attacks. True Negative (TN) refers to the total number of real records that are correctly classified as normal records. False Negative (FN) refers to the total number of real attack samples that are incorrectly detected as normal. False Positive (FP) refers to the total number of normal samples that are incorrectly identified as attacks.

2) Malicious network detection node (Fog layer): In the fog computing layer, this work offer an IDS in the proposed architecture. Devices in this tier have more effective attributes than those on the IoT edge layer. It is feasible to use intrusion\ malicious detection to monitor the IoT system without sending the data to the cloud, eliminating the latency issue that has plagued several studies advocating cloud layer analysis. The fog layer has processing nodes nearest to the physical IoT system, processing instruments, and edge storage to identify threats more quickly. A binary classification approach and a multi-class classification method are used in the architecture to detect intrusions.

The process is shown in Fig. 5.

1) The fog node connects each terminal device to the network using various protocols and collects data created by each terminal sensor in real-time[47].

2) The original data is preprocessed and trained by the cloud server: The entire training dataset is collected on the cloud server, and the entire training procedure is completed there, including the generation, and saving of a training model.

3) The fog node transmits a detection command: After establishing a communication link with the terminal device, the fog node gathers a considerable quantity of network data and sends a detection instruction to the cloud server.

4) The cloud server provides the training phase: The cloud server sends the data preprocessing pipeline and the trained classification prototype to the fog node after receiving the detection instruction.

5) Fog-node detection: The fog-node receives the model and utilizes it for data preprocessing and detection, producing detection results.

6) Malicious response: The discovered anomalous data are forwarded to the malicious response module, which performs the necessary processing [47].

In the fog layer, the same malicious network IDS model was used, it was trained in the cloud layer.

3) The host malicious detection node (fog layer): The fog layer contains the operating system (OS) devices in the IoT system. a malicious detection model was deployed for each device in the fog layer in the IoT system. The deployed model is called a host intrusion detection system. A host IDS is considered to run on a single machine and protect it from interruptions or malicious attacks that could harm the device – or data. A HIDS uses the measurements in the host environment. These supplies are sent into the HIDSs as input.

Based on the selected ToN_IoT dataset, two operating systems (Linux and Windows 10&7) were included in the dataset. Based on these data, an intrusion model was designed for each partition of data to detect included attacks.

a) Windows dataset (preprocessing): The correlation study significantly influences the applicability of attributes in defining security events using machine learning models. We built a correlation coefficient function [43] for ranking the attributes powers into a range of [-1, 1] in order to estimate the correlation coefficient between the attributes without the label characteristics on the Windows 7 and 10 datasets. The direction of the link is indicated by the sign of the correlation coefficient, while the magnitude (i.e., how near it is to -1 or +1) reflects the strength of the relationships between the characteristics [43]. The correlation matrix was tweaked to find the most associated attributes with a cut-off value of 0.85 % or above. Table III shows the top ten most associated traits in each dataset. As presented in Table III illustrate the most linked attributes of the Windows 7 and 10 datasets, respectively. Machine/deep learning algorithms would.

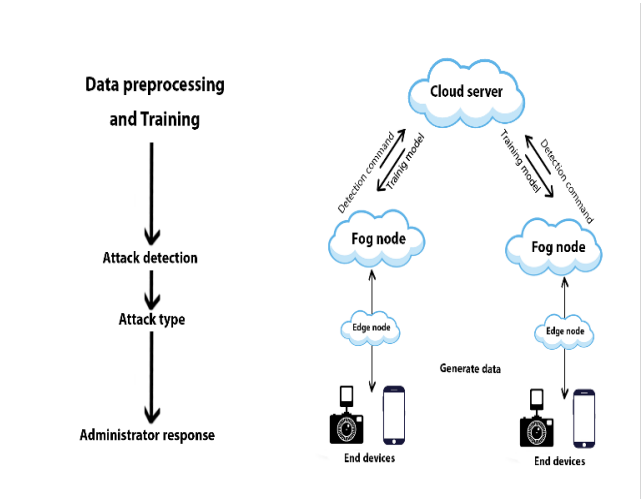

Fig. 5. Cloud-Fog Intrusion Detection Scheme.

TABLE III. THE TOP TEN ATTRIBUTES

Attributes (win_7)	Attributes (win_10)
Process _Total _IO Other _Bytes_sec	Network_I.Intel_R.._82574_L_GNC_ Current_Bandwidth
Network -_I.Intel_R_ Pro-_000MT)_Bytes_Received_sec	Network_I_Intel.R_82574L GNC_Packets_Sent_Unicast_sec
Process_Total _IO_ Other_Operations_sec	Memory_Pool_Paged_Bytes
Process_Total _IO Data_Bytes _sec	Logical_Disk_Total Disk_Read_Bytes.sec
Process_Total_IO_Read_Bytes _sec	Memory_Page__Reads__sec
Network I.Intel.R_ Pro_1000MT_Bytes_Received_sec	Network I.Intel.R 82574L_GNC.Packets_Sent.sec
Process _Pool_Paged_Bytes	Memory.Modified_Page_List_Bytes
Process _Pool_Paged_Bytes	Process _IO _Data_Operations_sec
Network _I.Intel.RPro_ 1000MT.Packets_Received_sec	LogicalDisk_Total._Avg.Disk.Bytes_ Transfer
Process__Total__IO_Data__Operations_sec	Processor _pct_Processor _Time

b) Preprocessing of Linux dataset: Linux datasets are partitioned into three categories: disk, memory, and process. Timestamp and CMD attributes were eliminated from the dataset for obtaining high performance and far away from overfitting. The Correlation-matrices of the most critical attributes in the Win_7 dataset and Win_10 are extracted from another paper [48].

VI. DATASET EVALUATION

In this part, the architecture of the suggested model was evaluated; the evaluation is based on binary classification and multi-class classification.

The presented results in this section are based on the best outcomes from the experimentations, other outcomes for classifiers such LR, NB, Adaboost, and SVM are neglected since these classifiers have poor results.

Malicious detection for network dataset evaluation (cloud layer):

Based on the newly available ToN-IoT dataset, the efficiency of the deployed machine learning approaches were investigated for malicious identification. The best parameters stated in the literature were selected [49], [50]. The experiments for this work were carried out in Python 3.8. All trials were run on a Windows 10 computer with a Core i7 processor and 16 GB of RAM. An experimental methodology was utilized to evaluate the effectiveness of the selected ML algorithms using the ToN-IoT network.

- Binary classification [41]: The results for the network dataset are introduced in this section. In addition to the confusion matrix, the accuracy, precision, recall, F1-score, and FPR are offered to evaluate the chosen ML algorithms. In general, the XGBoost produces considerable results for binary classification depending on multiple feature engineering strategies applied to the dataset.

Using all attributes, for XGBoost, the training accuracy is 0.992 %, the testing accuracy is 0.991 %, the recall is 0.984 %, the precision is 0.991 %, and the F1-score is 0.987 %, according to the results. With a significance of 0.007, k-Nearest Neighbor (KNN) exhibits relevance in the scenario of false-positive rate (FPR). The findings for the best ML techniques are shown in Table IV. The kNN, on the other hand, was the second-best technique. The training accuracy is 0.989 %, the testing accuracy is 0.988 %, the recall is 0.986 %, the precision is 0.979 %, and the F1-score is 0.983 %, according to the kNN findings. Naive bias is the poorest technique (NB). The heterogeneity of data in ToN-IoT datasets might explain the ML technique's performance variances. The findings of RF and DT are practically identical.

As a feature selection strategy, the Chi^2 was used. After using Chi^2 to evaluate ML algorithms based on a variety of criteria, because the optimum assessment measure is obtained with only 20 features, the best 20 attributes were selected from the total 108 attributes. After Chi^2, XGBoost provides considerable results, almost identical to testing with all characteristics. The training accuracy is 0.984 %, the testing accuracy is 0.983 %, the recall is 0.984 %, the precision is 0.967 %, the F1-score is 0.975 %, and the FPR is 0.008 %, according to the findings. The KNN approach was the second-best technique. Naive bias is the poorest technique (NB).). The findings for the ML techniques are shown in Table IV.

Because ToN-IoT has an issue with class imbalance, another testing approach based on SMOTE was used; XGBoost and KNN also have the same best outcome with 0.990 % accuracies. Recall is 0.976 %, accuracy is 0.997 %, F1-score is 0.986 %, and FPR is 0.013 % for XGBoost in terms of other assessment criteria. KNN has a recall of 0.981 %, accuracy of 0.990 %, F1-score of 0.985 %, and FPR has the highest score of 0.001 % to other ML algorithms. XGBoost is superior to KNN since it requires less training and testing time.

Finally, the selected ML algorithms were assessed using Chi^2 and SMOTE for binary classification. With Chi^2 and SMOTE, KNN produces considerable results. The findings for the ML techniques are shown in Table IV.

- Multi-class classification: The dataset contains an attribute type that displays the attack sub-class for multi-class classification tasks, as stated before. There are ten sub-classes in ToN-IoT. In this part, candidates' ML methods will be analyzed for evaluating multi-classification tasks. When assessing prospective ML methods for a multi-class classification task, several factors must be considered. To begin, LR is most commonly employed to solve binary classification task and cannot be immediately used for multi-class classifications. As a result, to construct LR for multi-class classification, the one-vs-rest (OvR) approach is applied. Accuracy, precision, recall, F-score, and confusion matrix are the assessment measures used to compare all models. The multi-classification findings are summarized in Table V. When comparing all ML algorithms, XGBoost achieves decent results. The training accuracy for XGBoost is 0.986 percent, the testing accuracy is 0.983 %, the recall is 0.953 %, the

F1-score is 0.949 percent, and the FPR is 0.008 %. KNN comes in second with scores of 0.981 % for training accuracy and 0.979 % for testing accuracy, and the AdaBoost classifier has the worst metrics of all tested ML methods. SVM is the most time-consuming in terms of training and testing.

The Chi^2 feature selection methodology after was used evaluating all of the specified ML algorithms over the whole dataset. Chi^2 will assess all ML algorithms with the best 20 attributes from all 108 attributes. XGBoost achieves considerable results in binary classification, for example. The training accuracy is 0.985 percent, the testing accuracy is 0.982 %, the recall is 0.950 %, the precision is 0.943 %, the F1-score is 0.946%, and the FPR is 0.008 %. Table V. displays the results of all ML techniques applied with the Chi2 approach. KNN was the second-best approach, AdaBoost is the poorest model.

Another approach based on the SMOTE technique was used. As seen in Table V, XGBoost outperforms other commonly used ML algorithms. Finally, the selected ML algorithms were assessed using the Chi^2 and SMOTE methodologies on the basis of the multi-class classification issue. With the Chi^2 and SMOTE methods, XGBoost achieves considerable results. The training accuracy of XGBoost is 0.980 %, while the FPR is 0.019 %. Table V. displays the outcomes of chosen ML approaches using the Chi^2 and SMOTE techniques.

In the cloud layer, the network intrusion detection was suggested to be deployed in the cloud for binary and multi-class classification problems, as shown in Fig. 6.

TABLE IV. THE RESULTS OF THE BINARY CLASSIFICATION (NETWORK_DATA)

Data	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR	Confusion Matrix
Network All attributes	DT	0.981	0.980	0.960	0.982	0.971	0.022	[[88032 1966] [851 47464]]
	RF	0.980	0.979	0.959	0.984	0.971	0.023	[[87949 2049] [792 47523]]
	KNN	0.989	0.988	0.986	0.979	0.983	0.007	[[89325 673] [995 47320]]
	XGB	**0.992**	**0.991**	**0.984**	**0.991**	**0.987**	**0.009**	**[[89220 778] [442 47873]]**
Network Chi^2	KNN	0.984	0.982	0.983	0.965	0.974	0.009	[[89170 828] [1694 46621]]
	XGB	**0.984**	**0.983**	**0.984**	**0.967**	**0.975**	**0.008**	**[[89247 751] [1598 46717]]**
Network SMOTE	KNN	0.991	0.990	0.981	0.990	0.985	0.001	[[89067 931] [504 47811]]
	XGB	0.993	0.990	0.976	0.997	0.986	0.013	[[88789 1209] [125 48190]]
Network Chi^2 & SMOTE	KNN	0.985	0.982	0.959	0.989	0.974	0.023	[[87960 2038] [515 47800]]
	XGB	**0.986**	**0.982**	**0.954**	**0.996**	**0.975**	**0.026**	**[[87670 2328] [190 48125]]**

TABLE V. THE OUTCOMES OF THE MULTI-CLASS CLASSIFICATION (NETWORK_DATA)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR
Network All attributes	AdaBoost	0.399	0.399	0.339	0.229	0.274	0.505
	KNN	0.981	0.979	0.933	0.925	0.929	0.009
	XGB	**0.986**	**0.983**	**0.945**	**0.953**	**0.949**	**0.008**
Network Chi^2	AdaBoost	0.498	0.497	0.352	0.363	0.358	0.424
	KNN	**0.980**	**0.977**	**0.929**	**0.928**	**0.929**	**0.014**
Network SMOTE	KNN	0.976	0.976	0.901	0.956	0.928	0.018
	XGB	**0.980**	**0.979**	**0.907**	**0.968**	**0.937**	**0.018**
Network Chi^2 & SMOTE	KNN	0.971	0.976	0.899	0.956	0.927	0.019
	XGB	**0.980**	**0.978**	**0.911**	**0.967**	**0.939**	**0.019**

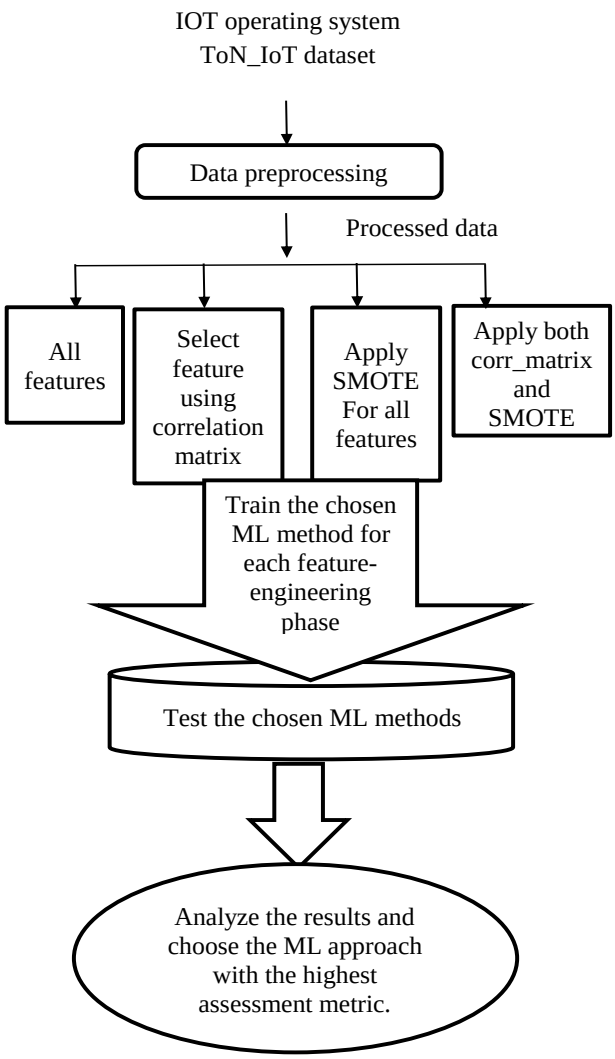

Fig. 6. The Process for Evaluating ML Methods on the OS ToN-IoT Dataset.

After applying various feature engineering techniques for the network ToN_IoT dataset, the XGBoost classifiers obtains optimal results using chi2 and SMOTE techniques that were used for class balancing and feature selection.

- Malicious detection for Linux dataset evaluation (fog layer): The same intrusion model was deployed in the fog layer with the same specification as discussed in earlier sections.
- Linux dataset evaluation (fog layer): This section focuses on the effectiveness of the used ML techniques for host malicious detection using the newly released Linux ToN-IoT dataset. Linux dataset was partitioned into three categories process, disk, and memory.
- Process_Linux (Binary classification): This section showed the findings for the Linux ToN-IoT dataset. In general, XGBoost is useful for binary classification. This experiment begin by impute missing values and then use the Min-Max normalization approach. Training accuracy is 0.994 %, testing accuracy is 0.993 %, recall is 0.990 %, precision is 0.991 %, and F1-score is 0.991 %, according to the findings. In terms of false-positive rate (FPR), XGBoost is significant at 0.005%. Table VI displays the results of all ML techniques applied without the SMOTE approach. The KNN approach, on the other hand, was the second-best methodology.

Because ToN-IoT has a class imbalance issue, another testing methodology based on SMOTE approach was used; RF and DT had the same better outcome with 0.992 % accuracies. In terms of other assessment measures, F1-score is 0.989 % for DT and RF, and FPR is 0.007 % for DT, which is superior to RF. Table VI displays the outcomes for several ML algorithms.

Multi-class classification (Process_Linux): For multi-class classification tasks, the Linux ToN-IoT dataset includes a feature type that displays the attack sub-category. When assessing prospective ML algorithms for a multi-class classification issue, several factors must be taken into account. The multi-classification findings are summarized in Table VII.

When compared to other ML algorithms, XGBoost achieves decent results. The training accuracy for XGBoost is 0.962 %, the testing accuracy is 0.954 %, the recall is 0.870 %, the precision is 0.909 %, the F1-score is 0.889 %, and the FPR is 0.004 %. DT and RF come in second with scores of 0.981 % for training accuracy and 0.979 % for testing accuracy, and the AdaBoost classifier has the worst statistics of all tested ML methods.

Another procedure was used based on the SMOTE method. As stated in Table VII. DT has the greatest outcomes compared to other used ML techniques.

- Memory_Linux (Binary classification): In this section, the results for memory-based Linux ToN-IoT dataset were presented. In general, RF is significant for binary classification. To begin with, the results reveal that the training accuracy is 0.999 %, the testing accuracy is 0.997 %, the recall is 0.993 %, the precision is 0.997 %, the F1- score is 0.995 %, and the FPR is 0.001 percent, with XGBoost achieving the best second result. Table VIII displays the results of all ML techniques applied without the SMOTE approach. NB is the worst approach.

Because ToN-IoT has a class imbalance problem, another approach based on SMOTE was used; RF produced the best result with 0.997 % accuracy.

- Multi-class classification (Memory_Linux): For multi-class classification tasks, the Linux memory ToN-IoT dataset includes a feature type that displays the attack sub-class. The multi-classification findings are summarized in Table IX.

When compared to other ML algorithms, XGBoost achieves decent results. The training accuracy for XGBoost is 0.986 %, the testing accuracy is 0.982 %, the recall is 0.922 %, the precision is 0.965 %, the F1-score is 0.943 %, and the FPR is 0.001 %. RF comes in second with scores of 0.988 % for training accuracy and 0.982 % for testing accuracy.

TABLE VI. THE OUTCOMES OF THE BINARY CLASSIFICATION (LINUX_PROCESS_DATA)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR	Confusion Matrix
process All attributes	KNN	0.992	0.990	0.988	0.985	0.986	0.007	[[29549 215] [281 17989]]
	XGB	**0.994**	**0.993**	**0.991**	**0.990**	**0.991**	**0.005**	**[[29606 158] [181 18089]]**
process SMOTE	DT	0.997	0.992	0.989	0.989	0.989	0.007	[[29555 209] [192 18078]]
	RF	**0.997**	**0.992**	**0.988**	**0.990**	**0.989**	**0.008**	**[[29540 224] [181 18089]]**
	XGBOOST	0.994	0.992	0.985	0.995	0.990	0.009	[[29483 281] [88 18182]]

TABLE VII. THE OUTCOMES OF THE MULTI-CLASS CLASSIFICATION (LINUX_PROCESS_DATA)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR
process All attributes	DT	0.969	0.952	0.901	0.889	0.895	0.005
	RF	0.969	0.950	0.875	0.839	0.857	0.006
	XGBOOST	**0.962**	**0.954**	**0.909**	**0.870**	**0.889**	**0.004**
Process SMOTE	DT	0.943	0.950	0.838	0.881	0.859	0.01
	RF	0.943	0.948	0.831	0.871	0.851	0.011
	XGBoost	**0.934**	**0.949**	**0.811**	**0.884**	**0.846**	**0.017**

TABLE VIII. THE OUTCOMES OF THE BINARY CLASSIFICATION. (MEMORY_LINUX)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR	Confusion Matrix
Memory All attributes	DT	0.999	0.996	0.995	0.991	0.993	0.002	[[29867 66] [104 11997]]
	RF	**0.999**	**0.997**	**0.997**	**0.993**	**0.995**	**0.001**	**[[29891 42] [83 12018]]**
	XGBoost	0.998	0.997	0.996	0.992	0.994	0.002	[[29887 46] [93 12008]]
Memory SMOTE	DT	0.999	0.996	0.992	0.993	0.993	0.003	[[29837 96] [79 12022]]
	RF	0.999	0.997	0.995	0.995	0.995	0.002	[[29867 66] [64 12037]]
	XGBoost	**0.998**	**0.996**	**0.991**	**0.996**	**0.993**	**0.004**	**[[29818 115] [49 12052]]**

TABLE IX. THE RESULTS OF THE MULTI-CLASS CLASSIFICATION (LINUX MEMORY)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR
process All attributes	DT	0.988	0.982	0.947	0.928	0.937	0.002
	RF	0.988	0.982	0.964	0.918	0.941	0.001
	XGBoost	**0.986**	**0.982**	**0.965**	**0.922**	**0.943**	**0.001**
Process SMOTE	DT	0.972	0.981	0.874	0.947	0.909	0.003
	RF	**0.972**	**0.981**	**0.878**	**0.938**	**0.907**	**0.003**
	XGBoost	0.968	0.978	0.877	0.933	0.904	0.008

Another approach was used based on the SMOTE method. As stated in Table IX, DT has the top outcomes compared to other employed ML methods.

- Malicious detection for windows dataset evaluation (fog layer): This work focuses on the efficiency of the chosen ML methods for malicious detection using the newly released Windows ToN-IoT dataset. The Windows dataset was partitioned into two categories, Win_10 and Win_7.
- Win_10 dataset (Binary classification): In this part, the outcomes for the Windows ToN-IoT were presented.

In general, XGBoost shows significance. Training accuracy is 1.0%, testing accuracy is 1.0%, and F1-score is 1.0%, according to the results. In the case of false-positive rate (FPR), XGBoost shows significance with 0.0. Table X. shows the results for the selected ML methods. Since ToN-IoT suffers from a class imbalance problem, another testing methodology was done based on SMOTE technique, XGBoost has the best result. Table X. shows the results for all used ML methods using SMOTE technique.

Another testing technique was based on the best selected attributes that were selected from a correlation matrix. XGBoost obtains the best result with and without SMOTE. Tables X shows the outcomes for all selected ML methods.

- Multi-class classification (Win_10): For multi-class classification issues, the ToN-IoT dataset includes a feature type that displays the attack sub-category. The multi-classification findings are summarized in Table XI.

When compared to other ML algorithms, XGBoost achieves decent results. XGBoost training accuracy is 1.0, its testing accuracy, recall, precision, and F1-score are all 1.0%, and its FPR is 0.00 %. Another testing approach based on the SMOTE technique was used. As seen in Table XI. XGBoost outperforms other commonly used ML algorithms.

- Win_7 dataset (Binary classification (Win_7): In general, for binary classification, XGBoost shows significant results for windows 7 dataset. Table XII displays the outcomes for the selected ML method.

ToN-IoT has a class imbalance problem, another testing methodology was done based on SMOTE technique, XGBoost has the best result.

XGBoost obtains the best result with and without SMOTE for multi-class classification. XGBoost shows significant outcomes. Table XIII displays the outcomes for the selected ML method.

TABLE X. THE OUTCOMES OF THE BINARY CLASSIFICATION. (WIN_10)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR	Confusion Matrix
WIN_10 All attributes	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**[[3045 0] [0 3287]]**
WIN_10 All attributes (SMOTE)	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**[[3045 0] [0 3287]]**
(10) Selected attributes	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.001**	**[[3043 2] [1 3286]]**
(10) Selected attributes SMOTE	**XGBoost**	**1.0**	**0.999**	**0.999**	**1.000**	**0.999**	**0.001**	**[[3042 3] [1 3286]]**

TABLE XI. THE RESULTS OF THE MULTI-CLASS CLASSIFICATION FOR NORMAL RECORDS AGAINST ATTACK RECORDS (WIN_10)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR
WIN_10 All attributes	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**
WIN_10 All attributes (SMOTE)	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**
(10) Selected attributes	**XGBoost**	**1.0**	**0.989**	**0.977**	**0.911**	**0.943**	**0.001**
(10) Selected attributes SMOTE	**XGBoost**	**1.0**	**0.986**	**0.868**	**0.912**	**0.890**	**0.001**

TABLE XII. THE OUTCOMES OF THE BINARY CLASSIFICATION. (WIN_7)

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR	Confusion Matrix
WIN_7 All attributes	**XGBoost**	**1.0**	**0.999**	**0.999**	**0.998**	**0.999**	**0.00**	**[[3000 1] [3 1790]]**
WIN_7 All attributes (SMOTE)	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**[[3000 1] [0 1793]]**
(10) Selected attributes	**XGBoost**	**1.0**	**0.987**	**0.988**	**0.977**	**0.982**	**0.007**	**[[2979 22] [42 1751]]**
(10) Selected attributes SMOTE	**XGBoost**	**0.999**	**0.986**	**0.979**	**0.983**	**0.981**	**0.012**	**[[2964 37] [31 1762]]**

TABLE XIII. THE RESULTS OF THE MULTI-CLASS CLASSIFICATION (WIN_7).

DATA	Models	Train Acc	Acc	Precision	Recall	F1-score	FPR
WIN_7 All attributes	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**
WIN_7 All attributes (SMOTE)	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**
(10) Selected attributes	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**
(10) Selected attributes SMOTE	**XGBoost**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**

VII. CONCLUSION

Based on the ToN-IoT dataset, this study proposed a unique malicious detection technique for IoT. The ToN-IoT dataset was employed for training and testing, and the suggested model incorporates several essential aspects. There is a problem with a class imbalance in the ToN-IoT dataset, as well as missing values. this work which deal with ToN-IoT can cover more attacks than prior work with obsolete datasets like KDD-CUP99, NSL-KDD, and UNSW-NB15.The ToN-IoT contains nine types of attacks (Scanning, Cross-Site Scripting (XSS), Denial of Service (DoS), Distributed Denial of Service (DDoS), Backdoor, Injection Attack, Password Cracking Attack, Man-In-The-Middle (MITM), Ransomware.

Exploring, preprocessing, feature selection, class imbalance solution, training ML methods, and testing ML methods are the various system blocks. In a network dataset, the Chi^2 approach was utilized to select attributes. It lowered the number of attributes to 20, resulting in a quicker training time, lower model complexity, and the highest performance over the whole dataset. Another feature selection methodology was the correlation matrix which was used in the windows dataset to obtain the most relevant attributes from the whole dataset. The SMOTE approach was utilized to balance the classes. It improved performance by reducing dominant class bias, reducing overfitting, and improving the overall performance. A good evaluation metric was achieved by using Chi2, SMOTE, and correlation matrix as preprocessing approaches. For evaluating the performance of the deployed ML algorithms, many evaluation metrics (accuracy, precision, recall, F1-score, FPR, and confusion matrix) were used. The results determined that XGBoost outperforms all other ML approaches in binary classification and multi-class classification tasks after assessing the selected ML methods. The main contributions of this work are that it uses a new benchmark dataset that is updated with new attacks and gathered from a real IoT system. The gathered dataset reflects data from each layer of the IoT system, such as (the cloud, fog, and edge layers). The proposed model is a distributed malicious model based on a multi-layer for IoT system. Various ML methods were tested in each specific parathion of the ToN-IoT dataset. The prosed model is the first suggested model that is based on the collected data from the same IoT system from all layers and devices (sensors).

In the future, Deep learning methods such as (recurrent neural network, auto-encoder, and convolution neural network) will be used in the ToN-IoT dataset. More work might be done in the future to enhance the performance of the baseline techniques on the datasets. Advanced parameter optimization approaches (for example, Bayesian optimization and the genetic algorithm) can be used to optimize the model's hyperparameters and get superior outcomes.

REFERENCES

[1] M. Lombardi, F. Pascale, and D. Santaniello, "Internet of Things: A General Overview between Architectures, Protocols and Applications," Information, vol. 12, no. 2, p. 87, 2021.

[2] X. Yao et al., "Security and privacy issues of physical objects in the IoT: Challenges and opportunities," Digital Communications, vol. 7, no. 3, pp. 373-384, 2021.

[3] A. S. Ashoor and S. Gore, "Difference between intrusion detection system (IDS) and intrusion prevention system (IPS)," in International Conference on Network Security and Applications, 2011, pp. 497-501: Springer.

[4] E. P. Nugroho, T. Djatna, I. S. Sitanggang, A. Buono, and I. Hermadi, "A Review of Intrusion Detection System in IoT with Machine Learning Approach: Current and Future Research," in 2020 6th International Conference on Science in Information Technology (ICSITech), 2020, pp. 138-143: IEEE.

[5] L. Thomas and S. Bhat, "Machine Learning and Deep Learning Techniques for IoT-based Intrusion Detection Systems: A Literature Review," International Journal of Management, Technology Social Sciences vol. 6, no. 2, pp. 296-314, 2021.

[6] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6481-6494, 2019.

[7] V. Pandu, J. Mohan, and T. P. Kumar, "Network intrusion detection and prevention systems for attacks in IoT systems," in Countering Cyber Attacks and Preserving the Integrity and Availability of Critical Systems: IGI Global, 2019, pp. 128-141.

[8] N. M. T.-I. D. [Online]. Available: Available: https:// cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i.

[9] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," IEEE Internet of Things Journal vol. 6, no. 3, pp. 4815-4830, 2018.

[10] M. Alhanahnah, Q. Lin, Q. Yan, N. Zhang, and Z. Chen, "Efficient signature generation for classifying cross-architecture IoT malware," in 2018 IEEE Conference on Communications and Network Security (CNS), 2018, pp. 1-9: IEEE.

[11] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," ICT Express, vol. 6, no. 4, pp. 280-286, 2020.

[12] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in 2018 IEEE 42Nd annual computer software and applications conference (COMPSAC), 2018, vol. 2, pp. 664-669: IEEE.

[13] K. D. T. Nguyen, T. M. Tuan, S. H. Le, A. P. Viet, M. Ogawa, and N. Le Minh, "Comparison of three deep learning-based approaches for IoT malware detection," in 2018 10th international conference on Knowledge and Systems Engineering (KSE), 2018, pp. 382-388: IEEE.

[14] H. Alasmary et al., "Analyzing and detecting emerging internet of things malware: A graph-based approach," IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8977-8988, 2019.

[15] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," Internet of Things vol. 7, p. 100059, 2019.

[16] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen, "An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach," International Journal of Digital Content Technology its Applications vol. 10, 2016.

[17] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach," Computer Communications, vol. 98, pp. 52-71, 2017.

[18] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 756-767: IEEE.

[19] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in 2019 IEEE Wireless Communications and Networking Conference (WCNC), 2019, pp. 1-7: IEEE.

[20] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," journal of Network Computer Applications vol. 163, p. 102662, 2020.

[21] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," Computers Security vol. 94, p. 101863, 2020.

[22] CAIDA:. Center for Applied Internet Data Analysis. CAIDA Data - Overview of Datasets, Monitors, and Reports Available: (https://www.caida.org/data/overview/index.xml).

[23] D. R. H. C.-C. t. F. Archive. Available: (https://www.defcon.org/html/links/dc-ctf.html).

[24] A. -IDS-DATASET. Available: (https://www.unsw.adfa.edu.au/unsw canberracyber/cybersecurity/ADFA-IDS-Datasets/).

[25] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," in Proceedings of the first workshop on building analysis datasets and gathering experience returns for security, 2011, pp. 29-36.

[26] D. R. C. I. f. C. UNB. Available: (https://www.unb.ca/cic/datasets/index.html).

[27] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The UCI KDD archive of large data sets for data mining research and experimentation," ACM SIGKDD explorations newsletter, vol. 2, no. 2, pp. 81-85, 2000.

[28] N. Chandolikar and V. Nandavadekar, "Selection of relevant feature for intrusion attack classification by analyzing KDD Cup 99," International Journal of Computer Science Information Technology vol. 2, no. 2, pp. 85-90, 2012.

[29] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets," in Proceedings of the third annual conference on privacy, security and trust, 2005, vol. 94, pp. 1723-1722: Citeseer.

[30] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the internet of things: a survey of existing protocols and open research issues," IEEE Communications Surveys Tutorials vol. 17, no. 3, pp. 1294-1312, 2015.

[31] A. A. Olusola, A. S. Oladele, and D. O. Abosede, "Analysis of KDD'99 intrusion detection dataset for selection of relevance features," in Proceedings of the world congress on engineering and computer science, 2010, vol. 1, pp. 20-22: WCECS.

[32] NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity |UNB.

[33] N. Moustafa and J. Slay, "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), 2015, pp. 1-6: IEEE.

[34] A. Sivanathan et al., "Characterizing and classifying IoT traffic in smart cities and campuses," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 559-564: IEEE.

[35] A. H. A. Sivanathan, Hassan Habibi, and V. Sivaraman., " UNSW Proliferation Dataset.," ed.

[36] Canadian Institute for Cybersecurity (CIC). IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, ed.

[37] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in 2016 International Conference on Information Science and Security (ICISS), 2016, pp. 1-6: IEEE.

[38] CSE-CIC-IDS2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, ed.

[39] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," Future Generation Computer Systems, vol. 100, pp. 779-796, 2019.

[40] T.-I. D. N. Moustafa, 2020, [online] Available: https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i.

[41] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset," IEEE Access, vol. 9, pp. 142206-142217, 2021.

[42] A. R. Gad, N. Hassan, R. A. A. Seoud, and T. M. J. A. Nassef, "Automatic machine learning classification of Alzheimer's disease based on selected slices from 3D magnetic resonance imagining," vol. 67, pp. 10-15.

[43] J. Wang, M. Xu, H. Wang, and J. Zhang, "Classification of imbalanced data by using the SMOTE algorithm and locally linear embedding," in 2006 8th international Conference on Signal Processing, 2006, vol. 3: IEEE.

[44] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," Journal of Big Data, vol. 8, no. 1, pp. 1-41, 2021.

[45] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, 1995, pp. 388-391: IEEE.

[46] M. Haggag, M. M. Tantawy, and M. M. El-Soudani, "Implementing a deep learning model for intrusion detection on apache spark platform," IEEE Access vol. 8, pp. 163660-163672, 2020.

[47] R. Du, Y. Li, X. Liang, and J. Tian, "Support vector machine intrusion detection scheme based on cloud-fog collaboration," Mobile Networks Applications pp. 1-10, 2022.

[48] N. Moustafa, M. Keshk, E. Debie, and H. Janicke, "Federated TON_IoT Windows datasets for evaluating AI-based security applications," in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 848-855: IEEE.

[49] J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques. Elsevier, 2011.

[50] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," IEEE Access, vol. 8, pp. 165130-165150, 2020.