IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2020, accepted August 31, 2020, date of publication September 9, 2020, date of current version September 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022862

TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems

ABDULLAH ALSAEDI[1], NOUR MOUSTAFA[2], (Senior Member, IEEE), ZAHIR TARI[1], ABDUN MAHMOOD[3], (Senior Member, IEEE), AND ADNAN ANWAR[4]

[1]School of Science, RMIT University, Melbourne, VIC 3000, Australia
[2]School of Engineering and Information Technology, University of New South Wales at ADFA, Campbell, ACT 2612, Australia
[3]School of Computer Science and Information Technology, La Trobe University, Bundoora, VIC 3086, Australia
[4]School of Information Technology, Centre for Cyber Security Research and Innovation (CSRI), Deakin University, Geelong, VIC 3220, Australia

Corresponding author: Abdullah Alsaedi (abdullah.mohammed.alsaedi@rmit.edu.au)

This work was partly supported by the Research grants of the Australian Research Data Commons (ARDC) RG192500 and UNSW Canberra PS51776.

ABSTRACT Although the Internet of Things (IoT) can increase efficiency and productivity through intelligent and remote management, it also increases the risk of cyber-attacks. The potential threats to IoT applications and the need to reduce risk have recently become an interesting research topic. It is crucial that effective Intrusion Detection Systems (IDSs) tailored to IoT applications be developed. Such IDSs require an updated and representative IoT dataset for training and evaluation. However, there is a lack of benchmark IoT and IIoT datasets for assessing IDSs-enabled IoT systems. This paper addresses this issue and proposes a new data-driven IoT/IIoT dataset with the ground truth that incorporates a label feature indicating normal and attack classes, as well as a type feature indicating the sub-classes of attacks targeting IoT/IIoT applications for multi-classification problems. The proposed dataset, which is named TON_IoT, includes Telemetry data of IoT/IIoT services, as well as Operating Systems logs and Network traffic of IoT network, collected from a realistic representation of a medium-scale network at the Cyber Range and IoT Labs at the UNSW Canberra (Australia). This paper also describes the proposed dataset of the Telemetry data of IoT/IIoT services and their characteristics. TON_IoT has various advantages that are currently lacking in the state-of-the-art datasets: i) it has various normal and attack events for different IoT/IIoT services, and ii) it includes heterogeneous data sources. We evaluated the performance of several popular Machine Learning (ML) methods and a Deep Learning model in both binary and multi-class classification problems for intrusion detection purposes using the proposed Telemetry dataset.

INDEX TERMS Internet of Things (IoT), Industrial Internet of Things (IIoT), cybersecurity, intrusion detection systems (IDSs), dataset.

I. INTRODUCTION

The Internet of Things (IoT) is an emerging paradigm that enables the interconnection of physical objects and computing capabilities to connect to the Internet. The IoT can help to build flexible and efficient applications in various domains such as health care, environmental monitoring, and industrial control systems [1], [2]. Although IoT can increase productivity and efficiency through intelligent and remote management, it also increases the risk of cyber attacks due to a lack of security measures in the IoT ecosystem that exposes IoT devices to malicious attacks from both inside and outside of enterprise networks [3]. The potential threats to IoT applications and the need to reduce risks have recently become a hot topic in the cyber security area. Some IoT-based applications, also commonly known as Industrial IoT (IIoT) in the Industry 4.0 revolution, involve mission-critical tasks such as industrial control and infrastructure systems which require a high level of security. Reportedly, in the latest attack on the IIoT applications, several power substations in Ukraine were compromised resulting in a power black-out which affects approximately 225,000 customers [4]. A Supervisory

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale.

Control and Data Acquisition (SCADA) system that controls and monitors the smart grid's IIoT devices was compromised by an attacker who successfully acquired privileges to access SCADA systems through an IT network and took the power offline [4]. Another example of IoT attacks is the Mirai botnet in late 2016, which consisted primarily of compromised smart cameras, causing Internet-wide outages when it overwhelmed several high-profile companies with massive distributed denial-of-service (DDoS) attacks [5]. Hence, an efficient and accurate security measure is required to secure IoT/IIoT applications.

Several security tools (e.g., firewalls, encryption, and intrusion detection systems) have been extensively used in traditional IT-based systems. However, such measures cannot be directly deployed for IoT/IIoT-based applications without considering their different nature and characteristics [2], [6], [7]. In general, IoT and IIoT applications consist of lightweight communication protocols and resource-constrained devices that have limited computation power and storage capacity [2], [7]. Consequently, security applications and cryptographic solutions that require high computational capabilities cannot be applied to Iot/IIoT. Moreover, none of these security measures is sufficient to entirely protect IoT applications from potential threats. Yet, a full complement of these security tools can design a robust security system. In order to identify cyber threats against IoT/IIoT services, it is vital that cyber-security applications such as intrusion detection systems (IDS) be developed, specifically designed for IoT and IIoT applications to meet their special requirements [2], [6], [7].

Intrusion detection systems (IDSs) are widely used as a second line of defence to monitor systems or network events to detect possible malicious activities that successfully evade security perimeters (e.g., firewalls) [8]. The evaluation of intrusion detection methods is essential and the use of IoT-related datasets that reflect real-world IoT applications plays an essential role in evaluating the accuracy as well as the efficiency of IoT security methods. However, the **lack of availability of real-world datasets** for IoT and IIoT applications presents a major obstacle to the evaluation of intrusion detection methods tailored to IoT/IIoT applications. The scarcity of these datasets hinders the design and development of IoT-based intrusion detection methods since the empirical validation and evaluation of such methods should meet performance expectations [7], [9], [10]. This lack of availability is mainly due to privacy issues which is why many large companies who create these IoT datasets do not show interest in sharing their data with research communities [9], [10]. Finally, Buczak and Guven [11] conducted a survey of cyber-security research using data mining and machine learning methods for intrusion detection systems. They confirmed that the unavailability of a labelled dataset is a significant gap in the literature that must be addressed in order to develop a promising anomaly-based intrusion detection method.

Various network datasets, for example, KDDCUP99, NSL-KDD [12], UNSW-NB15 [13] and ISCX [14], were generated for evaluating IDSs; however, they do not include any specific characteristics of IoT/IIoT applications as these datasets contain neither sensors' reading data (i.e., telemetry or measurement data) nor IoT network traffic. The LWSNDR [15] dataset contains only a homogeneous data collected from a single and multi-hop Wireless Sensor Networks (WSNs), and it does not include any attack scenarios. Another known dataset is the AWID [16] dataset, which only contains the network features extracted from Media Access Control (MAC) layer frame from 802.11 wireless network. It also does not have the telemetry data of IoT devices.

Sivanathan *et al.* [17] proposed IoT-based datasets for IoT device classifications based on network traffic characteristics; however, these datasets did not include attack scenarios. Responding to the aforementioned issues, Koroniotis *et al.* [18] and Hamza *et al.* [19] recently proposed network-based IoT datasets that include attack scenarios. However, the datasets did not have a variety of attack types such as ransomware and Cross-site Scripting (XSS); nor they contain sensor measurement data of IoT devices.

Most of the recently published datasets [12]–[14], [17]–[19] are network-based datasets, which primarily contain packet-level and flow-level information or a combination of both, for detecting attacks on the IoT network. However, they do not have the actual data generated from sensor readings. While these types of datasets could assist in detecting network-based attacks targeting IIoT systems, they cannot adequately detect sensor attacks that manipulate sensory data or compromise IoT devices [20]–[22]. Therefore, there is a real need for real-world datasets that not only contain sensors' reading data but also includes various types of attacks to enable a comprehensive evaluation of data-driven IoT intrusion detection solutions. These issues have motivated us to come up with an IoT-related dataset that contains sensors' reading data as an information source for data-driven IoT-based IDS to properly monitor the internal behaviour of IoT applications, hereby protecting them from malicious activities that are intended to sabotage the functionality of the targeted applications.

The aim of this paper is to provide a representative and recent dataset that can be used to accurately design and evaluate IoT/IIoT defence solutions. We propose a **new data-driven IIoT-based dataset** (i.e., **T**elemetry data) gathered from a representative scale-down testbed for each IoT device. The proposed datasets are made publicly available for use by the research community [23]. The testbed includes seven IoT and IIoT sensors, such as weather and Modbus sensors, that were used to capture their telemetry data. The datasets are made publicly available for the use of the research community. Moreover, a description of the datasets and their characteristics are provided here. We also evaluated the performance of data-driven intrusion detection methods as a binary classification problem, based on several supervised machine learning methods using the proposed IIoT-based datasets. Then, all IoT device datasets are combined into a single dataset, named *combined_IoT_dataset*, and evaluated

on binary and multi-class classification problems. Throughout the paper, we will use the terms IoT and IIoT interchangeably since the proposed datasets have a variety of IoT and IIoT sensors.

The key contributions of this study are listed as follows:

1) A representative medium-scale testbed for the generation and collection of datasets is introduced. The testbed was designed based on interacting network elements and IoT/IIoT systems with the three layers of Edge, Fog and Cloud to simulate a realistic representation of IoT/IIoT network configuration.
2) New datasets are proposed with new data features for IoT services based on the proposed testbed. These datasets are named TON-IoT as they contain **T**elemetry data, **O**perating systems' data, and **N**etwork data from the testbed IoT/IIoT network.
3) A first-hand evaluation of seven popular machine learning methods as well as a deep learning model on the proposed datasets is also provided as a baseline for further research.

This paper is organised as follows. Section II discusses current studies related to existing security datasets. Section III presents an overview of the testbed architecture used for the generation and collection of the proposed datasets. A detailed description of TON_IoT datasets as well as the methodology used to develop the proposed datasets are given in Section IV. Also, normal and attack scenarios are discussed in this Section. An overview of candidate machine learning (ML) methods used to evaluate their performance on the proposed datasets is presented in Section V, and the evaluation and experimental results of these methods are shown in Section VI. Finally, the conclusion and possible future directions are given in Section VII.

II. RELATED WORK

The evaluation of intrusion detection methods tailored to IIoT applications is vital and the use of IoT-related datasets reflecting real-world IoT applications plays an important role in evaluating the accuracy and efficiency of IIoT security methods. However, the lack of availability of real-world datasets for IIoT applications is a major obstacle to the evaluation of intrusion detection solutions tailored to IoT/IIoT applications. The absence of such datasets hinders the design and development of IIoT-based intrusion detection methods since the empirical validation and evaluation of such methods should be showing promising performance [9], [10]. Buczak and Guven [11] conducted a survey of cyber-security research using data mining and ML methods for IDSs. They stated that the unavailability of labelled datasets is a significant gap found in the literature to develop promising anomaly-based intrusion detection solutions. This is mainly due to privacy issues, as most the IoT datasets from large companies are not made available nor shared with research communities [10].

Well-known datasets (e.g., KDDCUP99, NSL-KDD [12], UNSW-NB15 [13] and ISCX [14]) were then designed to fill this gap for evaluation purposes; these datasets however do not include the specific characteristics of IoT/IIoT applications, as they do not contain either sensor measurements nor IoT network traffic. Even though several studies did use such datasets to evaluate their IoT-related intrusion detection solutions [24]–[28], one can argue that these datasets do not reflect IIoT characteristics since none of which contains any IoT device in their testbeds. The LWSNDR [15] dataset contains only homogeneous data collected only from humidity-temperature sensor deployed in a single and multi-hop Wireless Sensor Networks (WSNs). Although it has some anomalous points since the author used a hot water kettle to introduce the anomalies, the dataset does not include any attack scenarios. Sivanathan *et al.* [17] came up later with IoT-based datasets for the classification of IoT devices based on network traffic characteristics. They developed a smart home testbed, where IoT traffic is collected and relied on flow-based characteristics to classify each IoT device. They assumed that each IoT device exhibits identifiable patterns in their traffic flows such as activity cycles and volume patterns. Nonetheless, such datasets do not have any attack scenario since they were generated for the purpose of device classification.

Addressing the aforementioned issues, [16], [18], [19] proposed network-based IoT datasets that included attack scenarios. Kolias *et al.* [16] proposed Aegean WiFi Intrusion Dataset (AWID) dataset for intrusion detection in wireless networks. This AWID dataset was collected from a Small Office/Home Office SOHO 802.11 wireless network containing the following devices: a desktop machine, two laptops, two smartphones, one tablet and a smart TV. However, The dataset contains only the traces from Media Access Control (MAC) layer frame, and it does not have the telemetry data of IoT devices.

Koroniotis *et al.* [18] generated a BoT-IoT dataset collected from a realistic representation of an IoT network that comprises both legitimate and attack traffic, the attacks included are DDoS, DoS, service scan, keylogging, and data exfiltration. The BoT-IoT dataset contains over 72 million records of network traffic collected from a simulated IoT environment. The author also provided a scaled-down version of the dataset with roughly 3.6 million records for evaluation purposes. Hamza *et al.* [19] proposed an IoT-based dataset to detect DoS attacks in an IoT network, where they collected normal and various type of DoS attacks traffic (e.g., TCP SYN flooding, Ping of Death, and SNMP/TCMP flooding). They imitated a smart home environment to collect their data. Nevertheless, these datasets neither do have a variety of attack types (e.g., ransomware and XSS-Cross-site Scripting) nor contain sensor readings of IoT devices.

In summary, the majority of recently published IoT datasets were designed to validate IoT network-based IDSs. They mostly contain packet/flow-level information or a combination of both to detect attacks on IoT networks; they do not however have the actual data generated from sensor readings (i.e., measurement/telemetry data). While this can assist in detecting network-based attacks targeting IIoT systems, it is

TABLE 1. A comparison of the popular and publicly available datasets with the proposed datasets.

Dataset	Creation Year	IoT telemetry data	Diverse attack scenarios	representative testbed with IoT scenarios	Label	Heterogeneity of IoT data sources
KDDCUP99 [29]	1998	✗	✗	✗	✓	✗
NSL-KDD [12]	1998	✗	✗	✗	✓	✗
LWSNDR [15]	2010	✓	✗	✓	✓	✗
UNSW-NB15 [1] [13]	2015	✗	✓	✗	✓	✗
AWID [16]	2015	✗	✓	✓	✓	✗
ISCX [14]	2017	✗	✓	✗	✓	✗
UNSW-IoT trace [17]	2018	✗	✗	✓	N/A	✗
UNSW-IoT [19]	2019	✗	✓	✓	✓	✗
BoT-IoT [18]	2018	✗	✓	✓	✓	✗
Proposed T-IIoT Datasets	2019	✓	✓	✓	✓	✓

still not sufficient to detect those attacks that either manipulate sensory data or compromise IoT devices [20]–[22]. This has motivated the community to come up with IoT-related datasets that contain sensor readings as an information source for data-driven IoT IDSs to monitor the internal behaviour of the given IoT applications as well to protect them from malicious activities intended to sabotage the functionality of targeted applications. Mohammadi *et al.* [10] conducted an extensive survey about the use of deep learning in IoT and big data for streaming analysis, where twenty-five (25) IoT-based datasets generated from sensor readings in different domains and used in DL (Deep Learning) implementations. Yet, none of these contained attack data. Therefore, there is a need for a real-world dataset that not only contains sensor measurements but also has various types of attacks to enable the evaluation of data-driven intrusion detection approaches.

This paper addresses the aforementioned limitations and proposes a new T-IIoT dataset to accurately design and evaluate IIoT defence solutions. To the best of our knowledge, this is first time in the security area where the (proposed) datasets include IoT telemetry data collected from heterogeneous IoT/IIoT data sources, network traffic and audit traces of operating systems. Moreover, the proposed dataset contains a variety of IoT-related attacks and legitimate scenarios, including the ground truth of both attack and legitimate instances. Table 1 provides a summary of the unique properties of the new proposed dataset when compared to existing datasets. As can be seen from the table, both KDDCUP99, NSL-KDD are outdated compared to the other datasets. The KDDCUP99, NSL-KDD, UNSW-NB15 and ISCX datasets do not include the specific characteristics of IoT/IIoT applications, as they do not contain either IoT telemetry data or IoT network traffic. The LWSNDR only contains a homogeneous data, and it does not contain any attack scenarios. The AWID dataset contains only the network features extracted from Media Access Control (MAC) layer frame from 802.11 wireless network, and it does not have the telemetry data. Furthermore, the UNSW-IoT trace dataset does not have any attack scenario since it was generated for device classification purposes only. The UNSW-IoT and Bot-IoT datasets do not contain sensor readings of IoT devices. In contrast, the new proposed dataset has new properties compared with the existing datasets: (i) it has various normal and attack events for several IoT/IIoT services, (ii) it includes heterogeneous data sources, and (iii) it was collected from a testbed with a realistic representation of a IoT architecture for communicating Edge, Fog and Cloud layers.

[1]UNSW-NB15 was generated at University of New South Wales (UNSW) in Canberra. The UNSW-IoT trace and UNSW-IoT were generated at UNSW in Sydney.

III. DETAILS OF THE TESTBED

This section describes the testbed environment created for the generation and collection of the proposed TON_IoT telemetry datasets. As depicted in Figure 1, a new systematic testbed of Industry 4.0/Industrial IoT (IIoT) networks was designed to create new representative datasets comprising data collected from several normal and cyber-attack events in a realistic representation of IoT networks. The testbed was designed based on interacting network elements and IoT/IIoT systems with the three layers of *Edge, Fog* and *Cloud* to simulate a real-world execution of current production IoT/IIoT networks. Software-defined Network (SDN) [30] and Network Function Virtualisation (NFV) [31] were used to facilitate the management of the interaction between the three layers, and included physical and simulated systems. To do that, the NSX-VMware platform [32] was utilised to provide the features of SDN and NFV.

Briefly, NSX-VMware platform [32] is a network virtualisation technology used to facilitate the implementation of virtual networks on a physical network and within the virtual server infrastructure. The SDN feature in the NSX-VMware platform allows network administrators to automatically control, change, and manage network and security behaviour in a dynamic way with minimal human intervention [30]. In the proposed testbed, the NSX-VMware platform with SDN enables us to automatically manage and control the programmable IoT/IIoT applications and network devices. Moreover, the NSX-VMware platform was deployed with VMware vSphere hypervisor NFV [33] to allow the creation and management of various virtual machines (VMs) that simultaneously operate to offer the IoT/IIoT and network services. Furthermore, the vCloud NFV [32]

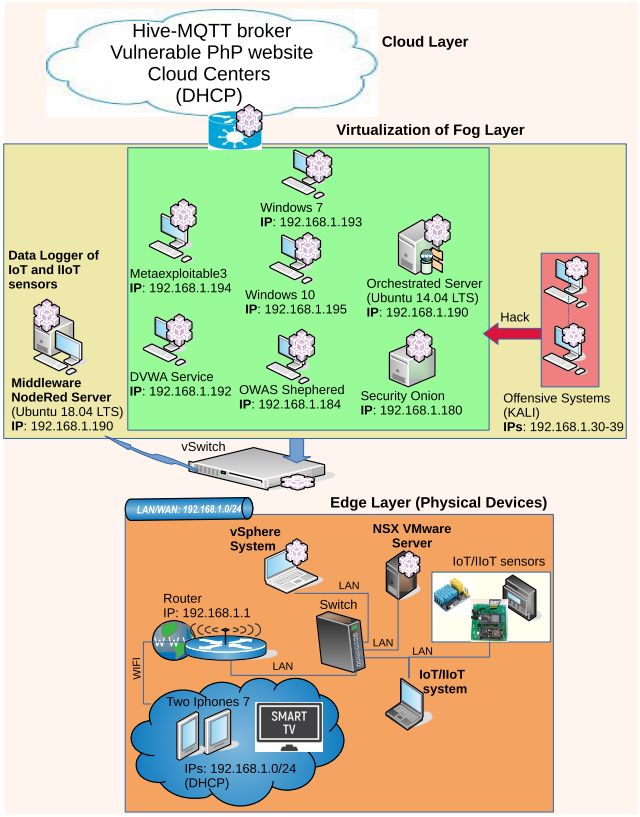


FIGURE 1. Testbed environment for generating and collecting the proposed datasets from the IIoT networks.

platform in NSX-VMware was utilised to provide a modular design with abstractions that facilitates multi-domain, hybrid physical, and VM deployment [31]. The benefit of using the vCloud NFV platform is that it provides and manages orchestrated VMs of the proposed testbed. Finally, using the NSX-VMware, we designed a blueprint architectural design that emulates and controls multiple VMs in the testbed for both hacking and normal operations, allowing the interconnections between the three layers. The *Edge* layer of the testbed includes a set of simulated and physical IoT/IIoT services, a NSX-VMware Server and different network devices (e.g., switches and routers). The *Fog* layer consists of the virtualisation technology, Offensive Kali systems [34], several VMs and Middleware Node-RED Server [35]. Finally, the *Cloud* layer contains various cloud services such as a Hive-MQTT broker [36], a vulnerable PHP website [37] and Cloud centres services (e.g., Microsoft Azure IoT Hub [38] and Amazon Web Services Lambda [39]). A detailed description of these three layers configured in the testbed is provided in the following subsections.

A. EDGE LAYER

This layer is the fundamental one of the entire IoT/IIoT applications since it can acquire data by directly measuring real-world physical conditions and sending information to the Fog or Cloud for further analysis [40]. As shown in Figure 1, the Edge layer includes physical devices, a NSX-VMware Server [41] and different network equipment (e.g., switches and routers) that are all connected to different LAN interfaces as well as linked to the Fog layer through the vSwitch. It includes a set of simulated and physical IoT/IIoT services and sensors, such as a thermostat and weather sensors, connected to a Message Queue Telemetry Transport (MQTT) gateway [42] to publish and subscribe to different topics, such as measuring temperature and pressure. Additionally, the NSX-VMware platform [32] was installed on the host server (i.e., NSX-VMware Server) to allow interaction between network elements and IoT/IIoT systems with the three layers of Edge, Fog and Cloud; thus it can lead to the realistic execution of current production IoT/IIoT networks. The hypervisor technology (VMware vSphere hypervisor NFV) of NSX-VMware [33] was installed on a host server (i.e., vSphere System) to manage the VMs created at the Fog layer. Finally, the testbed network contains other devices in this layer (e.g., two iPhone 7 and Smart TV) and their patterns have been recorded in the network traffic of the datasets.

B. FOG LAYER

The original idea of this layer is to extend the Cloud computing and services to the Edge of the network to provide limited computing capacity and storage near to the data sources [40], [43]. As it can be seen from Figure 1, the Fog layer consists of the virtualisation technology, the offensive Kali systems [34], various VMs and Middleware Node-RED Server [35]. The virtualisation technology provides us with a control on the VMs and their services using the NSX-VMware [41] and vCloud [32] platforms to offer the framework of executing SDN and NFV in the proposed testbed. The NSX vCloud NFV platform provides the design of a dynamic IoT/IIoT network testbed of the ToN_IoT dataset with creating and controlling several VMs for attacks and normal operations, allowing the interconnections between the three layers via vSwitches and gateways. Moreover, the Fog layer includes Offensive Kali systems (Linux-based OS) [34] that contains ten Kali Linux VMs with static IP addresses (i.e., 192.168.1.30-39) and various bash and python scripts of attacks scenarios that exploit vulnerable systems in the IIoT network. More details are provided in Section IV-B.

The Virtual Machines (VMs) shown in the green box were used to offer vulnerabilities such as the VMs of DVWA service [44], Metasploitable3 [45] and OWASP Security Shepherd [46], and others such as Windows 7 and 10 were employed as the remote web connection of the Node-RED tool. The Damn Vulnerable Web Application (DVWA) [44] is used as a vulnerable Web application coded in PHP/MySQL that contains the most common Web vulnerabilities. The DVWA VM (IP: 192.168.1.192) was utilised to make security vulnerabilities through Web applications that were attacked by the Offensive Kali systems. The Metasploitable3 [45] is a free vulnerable VM with various security

vulnerabilities. The Metasploitable3 VM (IP: 192.168.1.194) was deployed in the testbed to increase the vulnerability of Fog devices as well as to attack them using various hacking methods by the Offensive Kali systems. The Open Web Application Security Project (OWASP) Security Shepherd [46] (IP: 192.168.1.184) is a security platform that exploits common security vulnerabilities in Web and mobile applications being exploited of the offensive systems. Finally, the remaining VMs are the Orchestrated server and the Security Onion VM [47]. The Orchestrated server (OS:Ubuntu 14.04 LTS with IP: 192.168.1.190) is configured to offer some orchestrated services, such as FTP, Kerberos, HTTPS, and DNS to reflect real production networks. The Security Onion (IP: 192.168.1.180) was used to log network data from all the active systems in the testbed.

The Middleware Node-RED Server is the IoT/IIoT virtualised server deployed in the testbed. This server included the scripts that run IoT/IIoT services through public and local MQTT gateways utilised in the testbed and linked with the Cloud layer to subscribe and publish the telemetry data of IoT/IIoT sensors. Also, a data logger on the Middleware Node-RED Server was used to store telemetry data of IoT/IIoT sensors, and a Node-RED tool [35] used for simulating various IoT/IIoT sensors.

The IoT and IIoT services on the Middleware Node-RED Server were simulated through a Node-RED tool [35] and Modbus of Node-RED tools [48]. The Node-RED is an open-source development tool developed by IBM Emerging Technology used for the integration of IoT physical devices with APIs and their backend cloud server [35]. The main benefit of this tool is to enable developers/researchers to easily design and configure real-time IoT/IIoT applications on end-devices and connect them with their corresponding Cloud infrastructure. As mentioned in Section IV-A, using the Node-RED tool, various JavaScript scripts were designed to simulate various IoT/IIoT services (e.g., Smart Fridge, temperature, GPS, weather and Modbus) to present a variety of the most popular real-world IoT/IIoT applications found in smart home, smart cities and smart manufacturing. The various pieces of scripts were triggered for publishing and subscribing to a specific topic as explained later in Section IV-A. The data generated by several sensors were transferred to their corresponding backend cloud server using a Message Queue Telemetry Transport (MQTT) protocol [42]. The MQTT was mainly designed for resource-constrained devices such as IoT/IIoT sensors as a lightweight messaging transportation protocol that links machine-to-machine (M2M) communications and it also operates via a topic-based publish-subscribe mode. Accordingly, when a sensor (i.e., a publisher) publishes a message to a broker (server-side) under a particular topic, all the sensors (i.e., subscribers) that have subscribed to the same topic can receive this message. The broker is the primary component that completes the transfer process based on one-to-many connections [42].

C. CLOUD LAYER

This layer generally hosts large-size data centres with significant capacity for both computation power and storage to support IoT/IIoT applications and satisfy the resource requirements for big data analysis [43]. In the testbed, the Cloud layer contains various Cloud services such as a Hive-MQTT broker [36], a vulnerable PHP website [37] and Cloud centres services (e.g., Microsoft Azure IoT Hub [38] and Amazon Web Services Lambda [39]). The Hive-MQTT broker receives the IoT data from the lower layer for further analysis. As mentioned in Section III-B, IoT/IIoT sensors send the data to the corresponding Hive-MQTT broker using MQTT protocol. The Hive-MQTT broker is a public IoT MQTT-based messaging platform designed for fast and effective data transportation to and from the connected IoT/IIoT devices [36]. The Hive-MQTT broker has a list of subscribers, which receive the data sent by publishers. The use of the public IoT platform (i.e., Hive-MQTT broker) in the testbed ensures that it reflects a realistic IIoT network configuration. The vulnerable public PHP website was used to launch injection attacks events against websites as discussed later in Section IV-B. Finally, Microsoft Azure IoT Hub [38] and AWS Lambda [39] were used to subscribe and publish IoT/IIoT topics between them and the Fog VMs through the MQTT protocol. These Cloud services were used due to their popular use in most IoT applications, and to determine the variations of IoT legitimate samples when applying ML methods.

IV. IIoT DATASETS AND EXPERIMENTAL SETUP

A. OVERVIEW OF THE DATASETS

The ToN-IoT datasets include heterogeneous data sources gathered from the Telemetry data of IoT/IIoT services, as well as the Operating Systems logs and Network traffic of IoT network, which were collected from a realistic representation of a medium-scale network designed at the Cyber Range and IoT Labs at the UNSW Canberra. The main focus of this work is on the proposed dataset of the Telemetry data of IoT/IIoT services and their characteristics. The ToN-IoT datasets can be accessed at ToN-IoT repository [23]. Moreover, the proposed datasets were labelled with a *label feature* (indicating whether an observation is normal or attack) and a *type feature* (indicating the attacks sub-classes for multi-class classification problems). Nine (9) types of cyber-attacks (e.g., Scanning, DoS, DDoS, ransomware, backdoor, data injection, Cross-site Scripting (XSS), password cracking attack and Man-in-The-Middle (MITM)) were launched against various IoT and IIoT sensors across the IIoT network. Details of the dataset can be accessed in [23].

The generated data were stored in log and CSV files, and seven (7) IoT and IIoT sensors (e.g. weather, temperature and Modbus sensors) were used to capture their telemetry data, and two smartphones and a smart TV were logged in network traffic. The two main folders of IIoT datasets are 'Processed_datasets' and 'Train_Test_datasets'. The 'Processed datasets' folder contains a processed and filtered

version of the datasets with their standard features and label in the format of CSV files. In the 'Train_Test_datasets' folder, samples of the datasets are used as train-test datasets in a CSV format selected for evaluating the accuracy and efficiency of new cybersecurity applications and machine learning methods. Each IoT device has its own CSV file such as 'Train_Test_IoT_*device_name*.csv'. The total number of Train-Test IIoT datasets is seven (7), which covers each of the IIoT devices: Fridge, GPS Tracker, Motion Light, Garage Door, Modbus, Thermostat and Weather.

The testbed has a combination of physical and simulated IoT/IIoT services. The real devices include two smartphones and a smart TV configured with dynamic IP addresses using the DHCP server installed on the Orchestrated Server. They also include physical ESP8266 weather sensors [49]. These sensors were deployed at the edge layer and linked with the Node-RED to subscribe and publish sensory data to the public MQTT broker in the cloud layer. The testbed also has six simulated IoT services found in most popular IoT/IIoT applications. These services were developed using Javascript in the Node-RED API and linked to the public MQTT broker in the cloud layer. To generate the telemetry data for the simulator, JavaScript codes were written to publish the data to the broker, and this data was logged in order to evaluate cyber security solutions. In what follows, three algorithms (i.e, JavaScript code) of the simulated IoT devices are provided namely: Fridge, GPS and Thermostat sensors. The remaining JavaScript codes of all the simulated IoT/IIoT services are provided in this link [50].

Algorithms 1, 2 and 3 show three of the simulated IoT/IIoT services (i.e., Fridge, GPS and thermostats) used in the testbed. Algorithm 1 describes the script of the simulated fridge sensor that runs against the message payload that is passed through it as an input. The main idea of the script is to measure the fridge's temperature (*fridge_temperature*), given as an input from the message payload, and adjust it below a predefined threshold. First, the *timestamp* is retrieved from the *date* function, which will be sent with the telemetry data. As shown in the Algorithm 1, the *if* statement is used as a threshold to ensure that the *fridge_temperature* does not exceed the predefined value and adjust it accordingly. *temp_condition* is set to high or low based on the *fridge_temperature* value. Then, the generated values of *fridge_temperature* and *temp_condition* together with the *timestamp* will be published to the MQTT broker as the telemetry data of the Node-RED fridge sensor. Algorithm 2 gives the script of the simulated a GPS sensor where *latitude* and *longitude* values of GPS tracker sensor are given as inputs. Then, calculations are performed on both values to generate the telemetry data of the GPS sensor. Finally, Algorithm 3 gives the script of simulated thermostats' sensors, where *currentTemperature* and *sensorState* are retrieved from the message payload as input values. *currentTemperature* is the current temperature reading of a thermostat sensor connected to the testbed environment, where *sensorState* is the status of a thermostat sensor which is either on or off. The task of this script is to maintain the temperature between 25 and 26 $^{\circ}C$, the *default* value of the temperature is set to 25 $^{\circ}C$. Then, the generated values of *currentTemperature*, *sensorState* and *timestamp* will be published to the MQTT broker as the telemetry data of the thermostat sensor. This is just a brief explanation of simulated IoT devices, and each scenario of IoT devices together with a description of their features is provided below in more details.

For the generation of the dataset, various IoT/IIoT scenarios were simulated in the testbed. These scenarios could be applied to the following popular IoT/IIoT applications: smart homes, smart cities, and smart manufacturing. For instance, Smart Fridge, garage door and Motion lights sensors can be found in most smart homes. The GPS sensor can be found in the smart cities. The Modbus, Thermostat, and weather

Algorithm 1 Fridge Simulation Script

Input: *fridgeTemp*, *tempCondition*
Output: *data*

timeStamp ← *date*()
fridgeTemp ← *retrieve temperature from message payload* or *set it to a random number*
if (*RandomNumber* ≥ 0.5 or *fridgeTemp* ≥ 6) **then**
    fridgeTemp = *fridgeTemp* + (*randomNumber* × *fridgeTemp* ≥ 6)
end if
if *fridgeTemp* ≥ 6 **then**
    tempCondition ← *high*
else
    tempCondition ← *low*
end if
data ← *timeStamp*, *fridgeTemp*, *tempCondition*
return *data*

Algorithm 2 GPS Simulation Script

Input: *latitude*, *longitude*
Output: *data*

timeStamp ← *date*()
$x \leftarrow 0$
$y \leftarrow 10$
$z \leftarrow 0.01$
$sign_1$ ← set to a random number
i_1 ← set to (a random number ×9)
latitude ← retrieve it from message payload $+sign_1 * i_1 * z$ or x
$sign_2$ ← set to a random number
i_2 ← set to (a random number ×9)
longitude ← retrieve it from message payload $+sign_2 * i_2 * z$ or y
data ← *latitude*, *longitude*, *timeStamp*
return *data*

Algorithm 3 Thermostat Simulation Script

Input: *currentTemperature*, *sensorState*
Output: *data*

timeStamp ← *date*()
default ← 25
Initialize *currentTemperature* and *difference*
difference = *currentTemperature* − 25
currentTemperature ← retrieve it from message payload or set it to *default*
sensorState ← retrieve it from message payload or set it to *false*
if *currentTemperature* ≠ 25 **then**
  sensorState ← *true*
else
  sensorState ← *false*
end if
if *sensorState* ← *true* **then**
  if 0 < *difference* < 1 **then**
    currentTemperature = *currentTemperature* − *difference*
    sensorState ← *false*
  else if −1 < *difference* < 0 **then**
    currentTemperature = *currentTemperature* − *difference*
    sensorState ← *false*
  else
    set the *currentTemperature* to a random number
  end if
end if
data ← *currentTemperature*, *sensorState*, *timeStamp*
return *data*

monitoring can be found in most smart manufacturing and industrial applications. We extracted the features in these scenarios based on the sensing functionality of the simulated IoT/IIoT services. The detail of the IoT/IIoT scenarios and the extracted features are as follows:

- *Smart fridge* measures the temperature and adjusts it below a threshold when needed. Table 2 shows the feature column in IoT fridge dataset with their descriptions.
- A remotely activated *garage door* enables opening or closing of the door based on a probabilistic input. The features of the garage door sensors are listed in Table 3.
- *Global Positioning System GPS* tracks the location coordinates such as latitude and longitude of an object remotely, and the GPS features are shown in Table 4.
- *Smart sense motion*, which turn on or off the light based on a pseudo-random generated signal and the features description is presented in Table 5.
- *Modbus* service, which simulates the functionality of the Modbus devices found in many industrial applications as these devices communicate with each other using a master-slave communication to transmit register types such as input, discrete, holding and Coil over serial lines. We extracted the register type features as it is the functionality of the Modbus service. More details can be found in Table 6.
- *Smart thermostat* regulates the temperature of a physical system by controlling a heating/cooling system (e.g., the central heating, Air-conditioning, or water heaters system). The main two features in a smart thermostat sensor are the current temperature and thermostat status as explained in Table 7.
- *Weather monitoring system* generates data about air pressure, humidity, and temperature. More details are presented in Table 8.

B. NORMAL AND ATTACK SCENARIOS

As mentioned in Section IV-A, seven (7) IoT and IIoT sensors were simulated. The Node-RED tool was used to connect sensors and their corresponding backend cloud server to generate normal data. Various cyber-security incidents (i.e., DDoS and ransomware, XSS - Cross-site Scripting, backdoor and injection) were launched against different IoT and IIoT sensors. The hacking scenarios were launched to exploit either the Node-RED's IP address, public and local MQTT brokers or WiFi connections of the physical IoT sensors. In order to label the data as either normal or attack, we used the timestamp field of each well-known attack that occurred to tag each vector in the dataset. Then, after labelling the attacks and their types (i.e., the attack sub-classes), we added the remaining vectors as either normal operations (that occurred via publishing and subscribing through public) or local MQTT brokers. As mentioned in Section III-B, the attack scenarios were carried out by the Offensive Kali systems that include ten Kali Linux VMs with static IP addresses (i.e., 192.168.1.30-39) and various bash and python scripts of attacks scenarios that exploit vulnerable systems in the IIoT network and launch the attacks scenarios against IoT/IIoT services (e.g., IoT/IIoT devices and public MQTT brokers). More details of the hacking scenarios and different scripts used to launch such attacks in the dataset are made available in [51]. A brief description of these cyber-attacks is provided below.

- **Scanning** [52], [53] is considered to be the first step where an attacker gathers information about a target system, such as opening ports and available services about a victim device or sensor, before launching the actual attack. The attacker usually uses scanning tools such as Nmap [54] or Nessus [55] to perform port scanning. We used both Nmap and Nessus tools from the offensive Kali systems to perform scanning attacks against the victim IoT/IIoT devices in the subnet (192.168.1.0/24) and the Public MQTT broker. The attack IP addresses of the Offensive Kali systems used to perform the scanning attack are as follows: *192.168.1.30, 192.168.1.31, 192.168.1.32, 192.168.1.33 and 192.168.1.38*.

TABLE 2. IoT fridge features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
fridge_temperature	Temperature measurement of a fridge sensor
temp_condition	Temperature conditions of a fridge sensor, where temperature is high or low based on a predefined threshold value
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 3. IoT garage door features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
door_state	State of a door sensor where the door is closed or open
sphone_signal	State of receiving the door signal on a phone where the signal is true or false
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 4. IoT GPS tracker features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
latitude	Latitude value of GPS tracker sensor
longitude	longitude value of GPS tracker sensor
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 5. IoT motion light features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
motion_status	Status of a motion sensor is either (on or off) where 1 indicates "on", and 0 indicates "off"
light_status	Status of a light sensor is either on or off
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 6. IoT IoT modbus features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging Modbus register's data
time	Time of logging Modbus register's data
FC1_Read_Input_Register	Modbus function code that is responsible for reading an input register
FC2_Read_Discrete_Value	Modbus function code that is in charge of reading a discrete value
FC3_Read_Holding_Register	Modbus function code that is responsible for reading a holding register
FC4_Read_Coil	Modbus function code that is responsible for reading a coil
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 7. IoT thermostat features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
current_temperature	Current Temperature reading of a thermostat sensor
thermostat_status	Status of a thermostat sensor is either on or off
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

TABLE 8. IoT weather features and description.

Feature	Description
ts	Timestamp of sensor reading data
date	Date of logging sensor's telemetry data
time	Time of logging sensor's telemetry data
temperature	Temperature measurements of a weather sensor
pressure	Pressure readings of a weather sensor
humidity	Humidity readings of a weather sensor
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks
type	A tag with normal or attack sub-classes, such as DoS, DDoS and backdoor attacks

- **Denial of Service (DoS)** [52], [53] is a well-known flooding attack where an attacker typically launches sequences of malicious attempts against a legitimate user to disrupt access to services. Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks are included in the IIoT dataset. DDoS is usually launched by a large number of compromised devices known as bots or botnets [18], [52]. This attack can be done by flooding target IIoT devices with a numerous number of connections so to exhaust the device resources (e.g., CPU and memory). As mentioned in Section I, IIoT devices have a limited computation power and storage capacity which makes them easily vulnerable to DoS attacks [2]. To launch such attacks against vulnerable IoT/IIoT devices in our scenarios, a python script (e.g., dos.py) was developed to perform DoS attacks using the Scapy package [56] and different python scripts (e.g, ddos.py, ddos2.py and ddos2_broker.py) were written to launch DDoS attacks using the UFONet toolkit [57]. The IP addresses of the offensive Kali systems used to perform DoS attacks are: *192.168.1.30, 192.168.1.31, 192.168.1.39* where DDoS attacks were launched by *192.168.1.30, 192.168.1.31 and 192.168.1.{34-38}*.
- **Ransomware** [58] is a sophisticated type of malware that denies a legitimate user access to a system or services by encrypting them and tries to sell the decryption key that allows the user to get back access to the system. An IoT ransomware is similar, however it denies access to IoT devices. IIoT devices and applications are potential victims of IoT ransomware since they often carry out mission-critical tasks where denied access or locked down to these applications could lead to catastrophic consequences such as financial losses to organisations [58], [59]. We used a Metasploitable3 [45] framework to exploit weaknesses in the target devices and then executed the ransomware attack. The ransomware attack was performed by the offensive Kali systems with IP addresses: *192.168.1.33* and *192.168.1.37*.
- **Backdoor** [16] is a passive attack that allows an adversary to gain unauthorised remote access to the infected IIoT devices by a backdoor malware. The adversary uses the backdoor to control infected IIoT devices and makes it a part of botnets to attempt DDoS attack [16], [60]. We utilised the Metasploitable3 [45] framework to perform backdoor attacks, which were launched by two offensive Kali systems with IP addresses as follows: *192.168.1.33* and *192.168.1.37*.
- **Injection Attack** [16], [60] often tries to execute malicious codes or inject malicious data into the IIoT applications. Also, the injection attack can manipulate telemetry data and the control commands in the IIoT system and disrupt the normal operation. To perform injection attacks two bash scripts (e.g., injection-1.sh and injection-2.sh) were written to inject data inputs against web applications (e.g., DVWA and the vulnerable public PHP) and Security Shepherd VMs and webpages of IoT services. The IP addresses of the offensive Kali systems participated in injection attacks are: *192.168.1.{30,31,33,35,36,38}*.
- **Cross-Site Scripting (XSS)** [60] often attempts to run malicious commands on a Web server in the IIoT applications. The XSS allows an attacker to remotely inject arbitrary Web scripts such as malicious HTTP or JavaScript codes. This attack can compromise the data

and the authentication procedures between IIoT devices and a remote Web server. We developed two bash scripts (e.g., xss1-sh and xss2.sh) using the Cross-Site Scripter (XSSer) toolkit [61] to perform XSS attacks on the Web applications of DVWA, Security Shepherd VMs and webpages of IoT services. The attack IP addresses of the offensive Kali systems used to perform the XSS attacks are: *192.168.1.{32,35,36,39}*.

- **Password Cracking Attack** [62] when an attacker uses password cracking methods such as brute force and dictionary attacks to guess the password of IIoT devices. This can enable the attacker to bypass authentication methods and compromise IIoT devices [16]. We developed two bash scripts: *(e.g., password-1.sh)* using the CeWL toolkit [63] for dictionary attacks and *(password-2.sh)* using the Hydra toolkit [64] for brute force attacks. These scripts were developed to simultaneously launch password attacks scenarios against vulnerable IoT/IIoT devices in the testbed. The password cracking attacks were launched by the following the offensive Kali systems with IP addresses, namely *192.168.1.30, 192.168.1.31, 192.168.1.33, 192.168.1.35 and 192.168.1.38*.
- **Man-In-The-Middle (MITM) attack** [9] is a well-known network attack that can intercept the communication channel between two devices and may manipulate their data. Some of the common MITM attacks are ARP Cache poisoning, ICMP redirect and port-stealing [9]. The offensive Kali VMs with IP addresses 192.168.1.31-34 were utilized to launch various MITM scenarios in our testbed. We used the Ettercap tool [65] to execute ARP Cache poisoning, ICMP redirection and port-stealing attacks.

V. CANDIDATE SUPERVISED ML METHODS

Having explained the various types of attacks included in the proposed datasets, we next present the statistics of normal and attack data records in each Train-Test IIoT dataset (see Figures 2 and 3). Several supervised ML methods along with a Deep Learning model have been applied to evaluate their performance on the proposed Telemetry datasets when such methods are used to train different classifiers for intrusion detection purposes. In particular, a combination of classification methods are implemented on the proposed datasets to evaluate their performance in terms of accuracy and other evaluation metrics (e.g., precision, recall, F-score) along with the required time for training and testing each classifier.

Candidate methods were chosen based on their widely used in the security domain as they have demonstrated a good performance on the design of IDSs, and have shown effectiveness in a variety of areas [11]. In particular, we consider these seven methods: Support Vector Machines (SVM) [66], k-Nearest Neighbour (kNN) [67], Naïve Bayes (NB) [68], decision tree-based methods (i.e., Random Forest (RF) & Classification and Regression Trees (CART) [66]), as well as Logistics Regression (LR) [69], and Linear Discriminant

FIGURE 2. Statistics of the training and testing records in Fridge, GPS, Garage and Thermostat datasets.

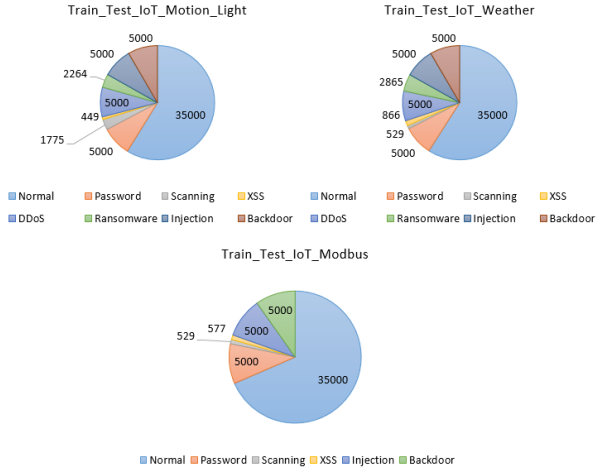


FIGURE 3. Statistics of the training and testing records in Motion_light, Weather and Modbus datasets.

Analysis (LDA) [70]. The authors in [11], [67] also showed that SVM, kNN, NB and CART are the most popular methods used for IDS development. Additionally, Resende and Drummond [71] conducted a survey that shows the effectiveness of the Random Forest (RF) [66] algorithm for IDSs since this can provide both a classification and embedded feature selection. Therefore, Random Forest is chosen here as the ensemble algorithm. LR and LDA are considered in this paper as they have low computation overheads [72]. Finally, Long Short-Term Memory (LSTM) [73] has been chosen as the deep learning method based on the reviewer's suggestion; moreover, it achieves state-of-the-art performance in dealing with sensor data and learning long-term dependencies from observations [74].

Below is a brief description of these methods:

- **Logistic Regression (LR)** [69]: even though it has the name 'regression', LR is commonly used for classification problems such as intrusion detection and spam

filtering, as it can estimate the probability that an observation belongs to a particular class [72]. For instance, if the estimated probability is greater than 50%, then the model will predict that the observation belongs to attack class since it exceeds the threshold; otherwise, it will predict it as a normal class. LR estimates the probability based on the following equation:

$$h_\theta(\mathrm{x}) = \sigma(\theta^T * \mathrm{x}) \tag{1}$$

where h_θ is the hypothesis function which outputs the estimated probability, x is the observation's feature vector, θ is the model's parameter, θ^T is the transpose of θ, and $\sigma(.)$ is a *sigmoid* (i.e., logistic) function which defines the threshold, and the equation of $\sigma(.)$ is defined as:

$$\sigma(\mathrm{z}) = \frac{1}{1 + e^{-\mathrm{z}}} \tag{2}$$

where z is the term $(\theta^T * \mathrm{x})$ in Eq. (1). Basically, the *sigmoid* function outputs a number between 0 and 1, where the value closest to 0 indicates that an observation is normal and value closest to 1 indicates that an observation is attack. The model's parameter θ is estimated during the training phase based on k-fold cross validation as explained in Section VI-A2.

- **Linear Discriminant Analysis (LDA)** [75] is a well-known linear algorithm commonly implemented as a dimensionality reduction method during the pre-processing step. In this work, however, LDA is not employed as a dimensionality reduction, rather it is applied as a classification method to build an intrusion detection model (i.e., a classifier) [72]. Initially, LDA estimates the *means*, and *covariance* matrix for multivariate features from the training data for each class based on the assumption that the data is normally distributed. Then, LDA uses Bayes' Theorem to estimate the probability of the output class (k) (whether it is a normal/attack class) given the observation (x) using the probability of each class. Bayes' Theorem is explained further in Naïve Bayes below. All the means, covariance matrix, and the estimated probability are estimated from the training data, and used in the LDA equation (i.e., a discriminate function). Finally, LDA uses the discriminate function to make the final prediction as the class with a highest probability is the output classification. The discriminate function is defined as:

$$f_k(\mathrm{x}) = \mathrm{x} * \frac{\mu_k}{\Sigma} - \frac{\mu_k^2}{2 * \Sigma} + \log(P_k) \tag{3}$$

where $f_k(\mathrm{x})$ is the discriminate function for class k given observation x, μ is the mean, Σ is the covariance matrix, and P is the estimated probability.

- **k-Nearest Neighbour (kNN)** [67]: unlike LDA, kNN is a non-parametric method that does not make any assumption about the underlying data distribution. This is also considered as a simple method that classifies an incoming observation from a test sample to the nearest sample in the training set based on specific metrics. In particular, kNN attempts to find a group of k observations in the training set that is closest to the test observation, and assigns a label based on the most common class among its k nearest neighbours. The kNN method has two main parameters [66]: a distance or similarity metric (i.e., to compute the distance between observations), and the value of k (i.e., the number of nearest neighbours). In this work, the value of k was set to 5 as it is considered as the default value, and Euclidean Distance is chosen as it is a widely-used distance metric [66], [70]. The equation of Euclidean Distance is defined as:

$$d(\mathrm{x}, \mathrm{y}) = \sqrt{\sum_{i=1}^{n} (\mathrm{x}_i - \mathrm{y}_i)^2} \tag{4}$$

where $d(\mathrm{x}, \mathrm{y})$ is a function that calculates the Euclidean Distance between two observations, x_i is the first observation, y_i is the second observation of data, n is the total number of observations and i is the index to a specific column as we sum across all columns.

- **Classification and Regression Trees (CART)** [66], [67] is a decision tree-based algorithm that constructs a binary tree structure from the training set. Each root node, also known as a non-leaf node, indicates a single input variable (x) and a split point on that variable whereas each leaf node (i.e., it does not have any children nodes) corresponds to an output of variable (y) used to make a prediction. The CART algorithm provides a foundation for other important tree-based methods (e.g., RF - Random Forest) which will be explained next. In this work, Gini impurity is used based on the recommendation in the current literature [67] as a split criterion to decide which feature to split at each step of the tree building process, as shown in Eq 5:

$$G(D) = \sum_{i=1}^{c} (P(i)) * (1 - P(i)) = 1 - \sum_{i=1}^{c} P(i)^2 \tag{5}$$

where D is the dataset, C is a set of classes, and $p(i)$ is the fraction of the number of samples with the class label, i in c. Gini impurity has 0 when there is only one class in c and reaches the maximum value when all classes are potentially equal.

- **Random Forest (RF)** [66], [71] is an ensemble learning that combines multiple decision trees that use randomly picked data points as their input. For classification task, RF can be used to classify an observation based on the results of a collection of decision trees. The final classification result can be decided by majority voting or weighted voting. In this work, the weighted voting is used as the voting technique, and the number of trees in the forest was set to 10 as it is the popular initial value to start with [70]. The Gini impurity (Eq. 5) is also used as a split criterion.

- **Naïve Bayes (NB)** [68] is a probabilistic-based approach which applies Bayes' theorem to perform the classification approach. Like LDA, NB assumes the data is normally distributed, and it calculates the conditional probability of a class label given a dataset. Moreover, the Bayes theorem provides a principled approach for computing this conditional probability with independence assumptions between the features, such as each input feature is independent of all other input features. Bayes' theorem is stated in Eq 6:

$$P(C|\mathrm{x}) = \frac{P(C) * P(\mathrm{x}|C)}{P(\mathrm{x})} \tag{6}$$

where $P(C|\mathrm{x})$ is the posterior probability of class C given the provided observation x, $P(C)$ is the prior probability, $P(\mathrm{x}|C)$ is the likelihood, and $P(\mathrm{x})$ is the evidence. These parameters are estimated from the training set. During the training phase, the Maximum A Posteriori (MAP) estimation is used to estimate the $P(C)$ and $P(\mathrm{x}|C)$, as the goal of NB is to classify a given observation x to the class with the highest posterior probability.
- **Support Vector Machine (SVM)** [66] is a discriminative classifier used for both linear and nonlinear data, and it is mainly defined by a separating hyperplane. SVM is intended to derive a hyperplane that maximizes the separating margin between the normal and attack classes. Different *kernel* functions exist for expressing the hyperplane, ranging from a linear kernel that attempts to find a simple linear separation between the data, to a non-linear one (e.g., Gaussian Radial Basis Function (RBF) *kernel*) [66]. In the experiment, we use the widely adopted Gaussian RBF kernel function of SVM [66], [70], and the kernel coefficient gamma is set to 'auto' as suggested in [70]. RBF is used as a nonlinear mapping to transform the original training data to a higher dimension in which it searches for the linear optimal separating hyperplane. The equation of the hyperplane is defined as:

$$g(\mathrm{x}) = \sum_{i=1}^{n} Wi * \mathrm{x}i + b0 \tag{7}$$

where $g(\mathrm{x})$ is the hyperplane function, W is a weight vector, n is the number of features in n-dimensional space and b is a bias.
- **Long Short-Term Memory (LSTM) [73]** is a variant of Recurrent Neural Network (RNN) architecture used in the field of deep learning, which is primarily designed to accurately model temporal sequences and their long-term dependencies by introducing a collection of memory units in the recurrent hidden layer. LSTM usually consists of memory cells and gates. The use of different gate units in LSTM can address the issue of gradient vanishing or explosion caused by memory loss for long-term sequences, which can be encountered when training RNN [73]. An LSTM model can be formulated as a classification problem in a supervised manner to be used for attack detection by computing a mapping function from an input observation $\mathrm{x} = (x_1, x_2, \ldots, x_N)$ to an output label y within the [0,1] set, by calculating the activation function of the network units. A typical LSTM memory cell is configured with an input vector $\mathrm{x}^{<t>}$, hidden input vector $h^{<t-1>}$ from previous timestep, and output vector $h^{<t>}$. The implementation of the memory cell can be established using the following equations in iterative manner [76].

$$i^{<t>} = \sigma(W_i \mathrm{x}^{<t>} + W_i h^{<t-1>} + b_i) \tag{8}$$
$$f^{<t>} = \sigma(W_f \mathrm{x}^{<t>} + W_f h^{<t-1>} + b_f) \tag{9}$$
$$o^{<t>} = \sigma(W_o \mathrm{x}^{<t>} + W_o h^{<t-1>} + b_o) \tag{10}$$
$$u^{<t>} = \tanh(W_u \mathrm{x}^{<t>} + W_u h^{<t-1>} + b_u) \tag{11}$$
$$c^{<t>} = i^t \odot u^{<t>} + f^{<t>} \odot c^{<t-1>} \tag{12}$$
$$h^{<t>} = o^{<t>} \odot \tanh(c^{<t>}) \tag{13}$$
$$y^{<t>} = \phi(W_y . h^{<t>} + b_y) \tag{14}$$

where $\mathrm{x}^{<t>}$ is the input at the current time, σ is the logistic sigmoid function and $\odot$ indicates element-wise multiplication. The W terms indicate weight matrices, and the b terms indicate bias vectors. u is the cell input activation. i, f, o, and c are the input gate, forget gate, output gate, and memory cell, respectively. These gates collectively decide how to update the current memory cell $c^{<t>}$ and the current hidden state $h^{<t>}$. The input gate manages the flow of input activation into the memory cell. The forget gate is used to rest the memory cells when their contents are outdated. The output gate controls the output flow of internal memory state to the rest of the network [76]. Lastly, y is the network output, and ϕ is the network activation function. In this work, the sigmoid function was used as ϕ for binary classification, whereas the softmax was used for multi-class classification task which maps the last hidden layer vector into a vector whose length is equal to the number of class labels.

The LSTM's hyperparameters were selected based on a grid search method [77], which designs an automated search to test different network configurations. The grid search was performed on the Fridge dataset as it has a combination of both discrete and continuous values; then these configurations were used for all the other datasets. The final hyperparameters used to configure the LSTM network are shown in Table 9. The number of epochs is set to 35 and the batch size is set to 64. In general, an epoch is the number of complete passes through the entire dataset so that each example is seen once by the model, where examples were separated into randomly selected "batch size" groups [78]. In addition, The LSTM models were configured with one input layer with "the number of units equal to the number of input features" based on the number of features at each dataset, and with three hidden layers with {128, 100, 64} units, respectively. The output layer was configured with a single unit and a sigmoid activation

TABLE 9. The hyperparameters of LSTM network.

Parameters	Value/Function
Hidden Layers	3
Units in hidden layers	1^{st}(128),2^{nd}(100),3^{rd}(64)
Epochs	35
Batch size	64
Hidden layer activation function	tanh
Output activation function	sigmoid (binary), softmax (multi-class)
Loss function	crossentropy
Dropout	0.2
Optimiser	Adam

function for the binary classification. For the multi-class classification, the softmax was used with units equal to the number of classes. As RNN like LSTM generally has the problem of overfitting [78], the dropout layer has been added after each layer to prevent model overfitting. The crossentropy was used as the loss function where ''binary crossentropy'' was used for binary classification and ''sparse_categorical crossentropy'' was used for the multi-class classification. Finally, we used Adam optimiser [79] for stochastic gradient descent to update network weights based in training data.

VI. EXPERIMENTAL RESULTS

This section discusses the performance of the candidates ML methods for intrusion detection purposes using the proposed IIoT datasets. As discussed in Section V, we utilised the parameters recommended in the current literature [66], [70], and adopted the default values as the initialisation of the candidate methods, because the goal of this experiment is to provide a first-hand evaluation of the performance of candidate ML methods applied to the proposed datasets as a baseline for further research. The experiments were carried out in Python version 3.7.3. The data processing and ML evaluation were implemented by extension packages including the packages of NumPy [80], SciPy [81], Pandas [82], and Scikit-learn v0.22.1 [83]. For LSTM implementation, TensorFlow v2.1.0 [84] was used with Keras v2.3.1 [85]. All experiments were executed under Windows 7 enterprise with core i7 3.60 GHz CPU and 8 GB memory. In what follows, we will describe the experimental methodology applied to evaluate the performance of the selected ML methods using the proposed IIoT datasets. The experimental results and discussion are also presented for both the per-device IoT datasets and the *combined_IoT_dataset*.

A. EXPERIMENTAL METHODOLOGY

1) DATA PREPARATION

It is essential to clean and prepare the data before applying any ML method to achieve good accuracy and accelerate the learning process. This is often carried out by removing unnecessary features that may degrade the performance, converting non-numerical features and replacing missing values if they exist. The main two steps applied during the data preparation process are *data pre-processing* and *data normalisation*.

- **Data Pre-processing**: categorical features that have nominal values were converted to numeric values in order to easily apply ML methods. For instance, the *temperature* feature in the Fridge dataset, which has categorical values '*high*' and '*low*', were mapped into '0' and '1'. These categorical values have been converted to consecutive numeric values by applying a label-encoding method [70]. The following features (date, time, and timestamp) were emitted from feature vectors as they may cause some ML methods to overfit the training data. For LSTM, the input data were reshaped into three dimensions (e.g., number of samples, timesteps, and feature numbers) in order to fed it into LSTM network. The timesteps has been set to one as it is the default representation when designing a LSTM network [78].
- **Data Normalisation**: some features have larger values than others, and this can lead to inaccurate results since a model might be biased to the large feature values. Thus, data normalisation plays an important role in preventing features with large values from outweighing the features with smaller values by scaling features within a range between [0.0,1.0] without changing the normality of data behaviour [86], [87]. As shown in Eq. 15, a min-max normalisation method is used to scale feature values within [0.0,1.0].

$$z = \frac{(x - x_{min})}{x_{max} - x_{min}} \tag{15}$$

where x is an original value, z is the normalised value, and x_{max} and x_{min} the minimum and maximum values of the feature, respectively.

2) TRAINING PROCESS

As discussed in Section IV-A, the train-test IIoT datasets are formatted in CSV files where each device has its own CSV file that contains both the training and the testing data. First, we further divided the dataset into two parts (i.e., train and test splits): 80% of the data was used to train/evaluate the selected ML methods, and the 20% of the data was held back for the testing dataset for further evaluation of the models (i.e., trained classifiers) with unseen data. The percentage of 80% for training and 20% for testing are chosen as suggested in [70] since the (80% train - 20% test) split is considered to be the best ratio to avoid the over-fitting problem where a model memorises the data rather than learns from it [88]. Next, a k-fold cross-validation was used for parameter tuning where it is applied to each training dataset by randomly dividing the set of observations into k subsets of equal size. Each time, one of the subsets was treated as a validation set, while the remaining k-1 subsets were used to train the models. Then, the average value of all folds was used as the final result [66], [70]. In this experiment, the k value was set to 4 as we used a different set of values (3,4,7 and 10). We found that 4 gives a better result, while the accuracy slightly decreases with a large k value such as 10 in some models. However,

the k value is a user parameter and should not affect the results as such. The performance of the generated models is evaluated with different evaluation metrics, which will be discussed in Section VI-B1. Figure 4 summarises the above steps involved in evaluating the performance of different candidates of ML methods using the proposed datasets discussed.

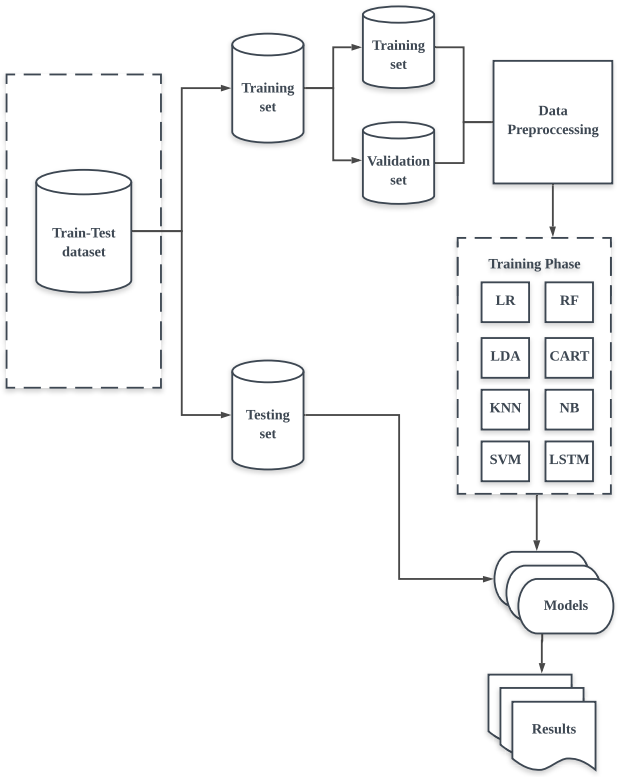


FIGURE 4. Evaluation process on the datasets with candidate ML methods.

B. FURTHER EVALUATION

1) EVALUATION METRICS

several metrics were used to evaluate the effectiveness of ML methods on the proposed IIoT dataset. In particular, accuracy, recall, precision, and F-score were used to quantitatively evaluate the performance of the selected ML methods [8]. The *accuracy* metric shows the overall effectiveness of a model as the fraction of all normal and attack observations that are correctly classified. The *recall* metric shows the number of attacks correctly detected divided by the total number of attack observations in the test dataset [8]. The *precision* metric shows the percentage of correctly detected attack observations over all the detected attacks. The *F-score* calculates the harmonic (equally-weighted) mean of precision and recall [8]. These metrics are defined as follows:

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} \tag{16}$$

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$F - Score = 2 * \frac{(Recall * Precision)}{Recall + Precision} \tag{19}$$

where True Positive (TP) is the # of actual attack records that are correctly detected as attacks, True Negative (TN) is the # of actual normal data that are correctly classified as normal, False Negative (FN) is the # of actual attack instances that are incorrectly classified as normal and False Positive (FP) is the # of actual normal instances that are incorrectly detected as attacks.

In addition, we consider the training time (i.e., the CPU time to build model), and testing time (i.e., the CPU time to test the model) as they are important to evaluate the execution time taken by the model especially during the testing/runtime phase to report the predicted results.

C. PER-DEVICE DATA SET EVALUATION

This section shows the experimental results for per-device datasets. The 4-fold cross validation was applied to all the models. The average value of all the evaluation metrics was computed and shown as the final results. In addition, the training and testing time are presented.

Tables 10 and 11 show the average accuracy, precision, recall and F-score as well as the training and testing time for the candidate methods applied to the seven benchmark datasets. In general, all the candidate methods show significant results for Garage Door dataset with an overall score of (1.00) for all the evaluation metrics. The reason behind this success of all the methods could be due to the fact that the Garage dataset has only discrete values which are easier to deal with, while the other datasets contain continuous values or a combination of both (e.g., discrete and continuous values). In contrast, all these methods perform poorly when applied to both Light sensor and Thermostat datasets with average accuracy and precision of (0.58 and 0.34) for the Light dataset, and (0.66 and 0.44) for the Thermostat dataset. These fluctuations in the performance of these methods could be explained by the heterogeneity of the data sources in IoT datasets. Long Short-Term Memory (LSTM) demonstrates significant results in the Fridge dataset with an overall (1.00) for all the evaluation metrics, followed by the k-Nearest Neighbour (kNN) which shows the second-best results (0.99) for all the evaluation metrics. Moreover, both LSTM and kNN show almost similar result in the GPS dataset with approximate values of (0.88) in all the evaluation metrics. Random Forest (RF), and Classification and Regression Trees (CART) outperform most of the other models in the Modbus dataset where they show accuracy of (0.97) and (0.98), respectively. In the Weather dataset, CART has the best results of (0.88) in precision and (0.87) in the other metrics while RF shows the second-best results of (0.84) in all the evaluation metrics. LSTM and kNN show competitive results in the Weather dataset where LSTM has accuracy and precision of (0.82), and kNN has results of (0.81) in all the metrics.

TABLE 10. The evaluation metrics results for Fridge, Garage Door, GPS and Modbus datasets (training and testing time is in seconds). [The abbreviations are as follows: LR- Logistic Regression, LDA- Linear Discriminant Analysis, *k*NN- *k*-Nearest Neighbour, CART- Classification and Regression Trees, RF- Random Forest, NB- Naïve Bayes, SVM- Support Vector Machine and LSTM- Long Short-Term Memory. These abbreviations also apply to Table (11-13). Note: the best value for each metric is highlighted in bold for each dataset].

Datasets	Models	Accuracy	Precision	Recall	F-score	Train Time	Test Time
Fridge Sensor	LR	0.57	0.34	0.58	0.43	0.017	0.001
	LDA	0.77	0.79	0.77	0.77	0.004	0.001
	*k*NN	0.99	0.99	0.99	0.99	0.147	2.556
	RF	0.97	0.97	0.97	0.97	0.188	0.045
	CART	0.97	0.97	0.97	0.97	0.032	0.005
	NB	0.50	0.53	0.51	0.51	0.011	0.005
	SVM	0.81	0.86	0.82	0.80	44.554	19.148
	LSTM	**1.00**	**1.00**	**1.00**	**1.00**	190.493	3.705
Garage Door	LR	1.00	1.00	1.00	1.00	1.00	0.001
	LDA	1.00	1.00	1.00	1.00	0.031	0.000
	*k*NN	1.00	1.00	1.00	1.00	0.625	0.969
	RF	1.00	1.00	1.00	1.00	0.062	0.000
	CART	1.00	1.00	1.00	1.00	0.016	0.000
	NB	1.00	1.00	1.00	1.00	0.010	0.002
	SVM	1.00	1.00	1.00	1.00	30.558	6.180
	LSTM	1.00	1.00	1.00	1.00	402.57	4.511
GPS Sensor	LR	0.86	0.88	0.86	0.87	0.066	0.001
	LDA	0.86	0.88	0.86	0.87	0.013	0.001
	*k*NN	**0.88**	**0.89**	**0.88**	**0.88**	0.08	1.508
	RF	0.85	0.85	0.85	0.85	0.833	0.099
	CART	0.84	0.85	0.85	0.85	0.277	0.015
	NB	0.84	0.86	0.85	0.86	0.009	0.007
	SVM	0.86	0.88	0.87	0.87	25.127	9.483
	LSTM	0.87	0.89	0.88	0.88	151.832	6.784
Modbus	LR	0.67	0.46	0.68	0.55	0.029	0.001
	LDA	0.67	0.46	0.68	0.55	0.041	0.001
	*k*NN	0.77	0.77	0.78	0.77	0.060	0.116
	RF	0.97	0.98	0.98	0.98	1.587	0.031
	CART	**0.98**	**0.99**	**0.98**	**0.99**	0.367	0.001
	NB	0.67	0.46	0.68	0.55	0.012	0.002
	SVM	0.67	0.46	0.68	0.55	19.286	3.558
	LSTM	0.68	0.47	0.68	0.55	317.959	1.651

For most of the benchmark dataset, both RF and CART obtain very similar results which could be due to the tree-based structure of their design. In addition, Logistics Regression (LR) and Linear Discriminant Analysis (LDA) achieve relatively similar results for most of the datasets and this could be due to the fact that they are both a linear-based algorithm, so they produce similar results. Naïve Bayes (NB) shows a good result only for the GPS dataset with an accuracy of (0.86), and it achieves average results for the other datasets. The results produced by the Support Vector Machines (SVM) average between (0.63) and (0.67) for most of the datasets. In terms of the training and testing time, LSTM has the longest training and testing times compared with those of other models, followed by SVM which has the second-longest times. Both LR and LDA have the lowest training and testing times.

Further evaluation is carried out in this section to assess the performance of the candidates ML methods using the proposed datasets by combing all the per-device datasets into a single dataset, *combined_IoT_dataset*. This can reflect some real-time scenarios since most real-time systems keep their data in a central database where all the data gathered from the system is stored for maintenance, historical, auditing and analysis purposes. Furthermore, the selected ML methods were evaluated for both binary and multi-class classification problems using the *combined_IoT_dataset*.

1) COMBINED_IoT_DATASET

Each IoT dataset was combined into one CSV file *combined_IoT_dataset*. A python script was implemented to automatically combine all IoT dataset into a single CSV file with a total of 22 features. Then, a *median* value for each column was used as an imputation to fill missing values in such column. The use of *median* is recommended as it is less susceptible to outlier errors compared to *mean* imputation [70]. The class distribution of the *combined_IoT_dataset* is shown in Figure 5.

2) BINARY CLASSIFICATION ON *combined_IoT_dataset*

The *combined_IoT_dataset* was similarly used to evaluate the ML methods as the per-device IoT dataset. Accuracy, recall, precision, and F-score were used to quantitatively evaluate the performance of the candidate ML methods on the *combined_IoT_dataset*. Additionally, the training

TABLE 11. The evaluation metrics results for Light_Motion, Thermostat and Weather datasets (training and testing time is in seconds).

Datasets	Models	Accuracy	Precision	Recall	F-score	Train Time	Test Time
Light_Motion	LR	0.58	0.34	0.59	0.43	0.028	0.001
	LDA	0.58	0.34	0.59	0.43	0.036	0.001
	kNN	0.54	0.34	0.59	0.43	3.157	6.409
	RF	0.58	0.34	0.59	0.43	0.1	0.008
	CART	0.58	0.34	0.59	0.43	0.008	0.001
	NB	0.58	0.34	0.59	0.43	0.011	0.002
	SVM	0.58	0.34	0.59	0.43	30.686	6.259
	LSTM	0.59	0.35	0.59	0.44	63.132	3.73
Thermostat	LR	0.66	0.44	0.66	0.53	0.03	0.001
	LDA	0.66	0.44	0.66	0.53	0.035	0.001
	kNN	0.60	0.56	0.61	0.57	0.064	0.088
	RF	0.66	0.59	0.66	0.53	1.044	0.023
	CART	0.59	0.56	0.59	0.57	0.009	0.001
	NB	0.66	0.44	0.66	0.53	0.009	0.002
	SVM	0.66	0.44	0.66	0.53	69.34	10.246
	LSTM	0.66	0.45	0.67	0.54	69.073	1.82
Weather	LR	0.58	0.60	0.59	0.53	0.083	0.001
	LDA	0.60	0.59	0.60	0.53	0.041	0.0
	kNN	0.81	0.81	0.81	0.81	0.066	0.414
	RF	0.84	0.84	0.84	0.84	0.789	0.008
	CART	**0.87**	**0.88**	**0.87**	**0.87**	0.258	0.03
	NB	0.69	0.72	0.69	0.67	0.011	0.002
	SVM	0.63	0.68	0.63	0.55	65.469	8.029
	LSTM	0.82	0.82	0.81	0.80	318.242	3.79

TABLE 12. Evaluation of binary classification models using combined_IoT_dataset (training and testing time is in seconds) [Note: the best value for each metric is highlighted in bold].

Models	Accuracy	Precision	Recall	F-score	Train Time	Test Time
LR	0.61	0.37	0.61	0.46	3.023	0.005
LDA	0.68	0.74	0.68	0.62	1.041	0.004
kNN	0.84	0.85	0.84	0.84	58.018	109.361
RF	0.85	0.87	0.85	0.85	10.884	0.164
CART	**0.88**	**0.90**	**0.88**	**0.88**	6.308	0.022
NB	0.62	0.63	0.62	0.51	0.21	0.069
SVM	0.61	0.37	0.61	0.46	3525.052	558.663
LSTM	0.81	0.83	0.81	0.80	1596.809	9.023

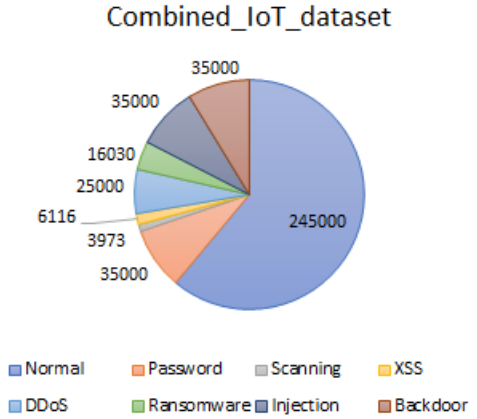


FIGURE 5. Statistics of *combined_IoT_dataset*.

and testing time for each model were calculated. Table 12 shows a summary of the results: CART achieves the highest score of 0.88 in (accuracy, recall and F-score), and a sore of 0.90 for precision with a test time of (0.022 second). RF and kNN score the second good results. RF has a score of 0.85 for (accuracy, recall and F-score), and a score of 0.87 in the precision while kNN shows a score of 0.84 for (accuracy, recall & F-score), and a score of 0.85 in the precision. In terms of the time, RF requires 0.164 seconds which is less testing time than kNN which requires a significant amount of testing time (109.361 seconds). This can be explained as kNN is a lazy leaner which uses the training phase to store the data, and then it uses the data during the test phase to make a prediction which makes the testing phase slower. LSTM shows a precision of (0.83) and a score of (0.81) for both accuracy and recall. LDA has an accuracy of (0.68) and a precision of (0.74). LR, NB and SVM have an overall accuracy around (0.61) but LR and SVM have a precision of (0.37) which is lower than all other models. The low precision of LR and SVM may indicate that most of the predicted labels are incorrect (high false positives). Finally, SVM has by far the longest training and testing time at (3525.052) and (558.663), seconds respectively.

TABLE 13. Evaluation of multi-class classification models on a combined_IoT_dataset (training and testing time is in seconds) [Note: the best value for each metric is highlighted in bold].

Models	Accuracy	Precision	Recall	F-score	Train Time	Test Time
LR	0.61	0.38	0.62	0.47	30.778	0.014
LDA	0.62	0.46	0.63	0.51	1.947	0.017
*k*NN	0.72	0.71	0.73	0.70	18.262	79.02
RF	0.71	0.69	0.72	0.67	8.233	0.17
CART	**0.77**	**0.77**	**0.77**	**0.75**	5.048	0.023
NB	0.54	0.59	0.51	0.52	0.738	0.093
SVM	0.60	0.37	0.61	0.46	10526.148	965.971
LSTM	0.68	0.64	0.68	0.63	1375.305	5.928

3) MULTI-CLASS CLASSIFICATION ON *combined_IoT_dataset*

As mentioned earlier in Section IV-A, the proposed datasets have a type feature that indicates the attack sub-classes for the multi-classification problems. In this sub-section, we further test the candidates ML methods to evaluate their performance on multi-classification problems. The evaluation of candidates ML methods for a multi-class classification problem requires some considerations. To begin, Logistic Regression (LR) is usually used for a binary classification and it cannot be directly applied to solve a mutli-class problem. Therefore, LR is implemented with the one-vs-rest (OvR) scheme to be used for multi-class classification. The one-vs-rest (OvR) scheme involves training a single LR classifier per class, with the training samples of that class as positive samples and all other samples as negatives. Each LR classifier predicts the probability of a particular class, and the class with the highest probability is selected [70]. For LSTM, the softmax was used as the network output activation function and ''sparse_categorical crossentropy'' was used as the loss function as mentioned in Section V.

Lastly, a weighted average is calculated by means of evaluation metrics as a final result for each model. The weighted average can be computed by multiplying each class with a weight factor (i.e., the # instances with that target class) and then calculating the sum for other classes [70]. The evaluation metrics used to compare all models are accuracy, precision, recall and F-score. Table 13 shows a summary of the multi-class classification results. CART achieves good results compared to the other methods with a score of (0.77) for all the metrics, and F-score of (0.75). *k*NN and RF achieve the second best results where *k*NN scores accuracy of 0.72 and recall of (0.73), whereas the RF shows accuracy of (0.71) and recall of (0.72). LSTM has a score of (0.68) in both the accuracy and recall metrics. LR and LDA show almost similar results for accuracy of (LR = 0.61 and LDA = 0.62) and recall of (LR = 0.62 and LDA = 0.63). Both SVM and LR have the worst precision score, (0.37) and (0.38) respectively. NB has the lowest accuracy score of (0.54). The SVM model shows an overall score of 0.60 for accuracy and 0.61 for recall. In terms of the execution time, SVM requires the longest training and testing time. Surprisingly, the LR has a training time of (30.778 seconds) which is the third-longest training time after the LSTM which has a training time of (1375.305 seconds).

VII. CONCLUSION

This paper proposed new IIoT-based datasets, called TON_IoT, which incorporate both normal sensor measurement data as well as various types of attacks targeting IIoT applications. These datasets were designed on a realistic representation of an IoT network testbed and were given a 'label' column to indicate normal and attack instances, and a 'type' column to indicate attack sub-classes for possible multi-class classification purposes. In addition, the datasets were combined into a single dataset, named *combined_IoT_dataset*, to represent real-world scenarios. Various evaluation metrics (i.e., accuracy, precision, recall and F-score) were used to evaluate the performance of seven supervised ML methods along with LSTM as a deep learning method for the purpose of intrusion detection using the proposed datasets. The results of the evaluation have indicated that the proposed datasets can be efficiently utilised to implement and train various ML methods for anomaly-based detection research. The main finding of the evaluation was that RF and CART achieved the highest score in all metrics on both per-device datasets and the combined one. This finding indicated an inherent advantage of both methods in distinguishing normal class and different attack classes. The results have also shown both LSTM and *k*NN had the second-best performance compared to the other methods. Overall, these results may pave the way for the future design and development of robust detection models for the specific datasets. In addition, we aspire that the findings provided in this work, and the contribution of the proposed datasets, will significantly benefit the research community working on IDSs for IoT/IIoT applications. As a future direction, more work could be done to improve the performance of the baseline methods on the proposed datasets. Advanced parameter optimisation methods (e.g., Bayesian optimisation and genetic algorithm) can be utilised to optimise the model's hyperparameters and achieve better results.

REFERENCES

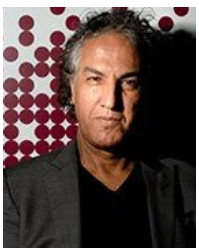
[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, ''Internet of Things (IoT): A vision, architectural elements, and future directions,'' *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.
[3] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.
[4] G. Falco, C. Caldera, and H. Shrobe, "IIoT cybersecurity risk modeling for SCADA systems," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4486–4495, Dec. 2018.
[5] M. Antonakakis, "Understanding the Mirai Botnet," in *Proc. 26th Secur. Symp.*, 2017, pp. 1093–1110.
[6] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
[7] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[8] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.
[9] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.
[10] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 3rd Quart., 2018.
[11] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[13] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[14] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[15] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *Proc. 6th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, Dec. 2010, pp. 269–274.
[16] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st Quart., 2016.
[17] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.
[18] 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 Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[19] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on loT devices via SDN-based monitoring of MUD activity," in *Proc. ACM Symp. SDN Res.*, Apr. 2019, pp. 36–48.
[20] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 10–28, Jun. 2017.
[21] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674–1683, Jan. 2018.
[22] T. Shinohara, T. Namerikawa, and Z. Qu, "Resilient reinforcement in secure state estimation against sensor attacks with a Priori information," *IEEE Trans. Autom. Control*, vol. 64, no. 12, pp. 5024–5038, Dec. 2019.
[23] N. Moustafa. (2020). *TON-IoT Dataset*. [Online]. Available: https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i
[24] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K.-R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 2, pp. 314–323, Apr. 2019.
[25] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in Internet of Things," in *Proc. IEEE 15th Intl Conf Dependable, Autonomic Secure Comput.*, Nov. 2017, pp. 836–843.
[26] S. A. Aljawarneh and R. Vangipuram, "GARUDA: Gaussian dissimilarity measure for feature representation and anomaly detection in Internet of Things," *J. Supercomput.*, pp. 1–38, 2018.
[27] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in Fog-to-Things computing," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 169–175, Feb. 2018.
[28] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.
[29] *KDD Cup 1999 Data*. Accessed: Dec. 14, 2019. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[30] SecureLink. *Ever Thought Virtualizing Your Network*. Accessed: Feb. 3, 2020. [Online]. Available: https://securelink.net/en-be/insights/nsx-vmware/
[31] *Cloud NFV*. Accessed: Feb. 3, 2020. [Online]. Available: https://www.vmware.com/au/products/network-functions-virtualization.htm
[32] *The VMware NSX vCloud NFV platform*. Accessed Feb. 3, 2020. [Online]. Available: https://lenovopress.com/lp0661.pdf
[33] *VMware vSphere Hypervisor*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.vmware.com/au/products/vsphere-hypervisor.html
[34] *Kali Linux*. Accessed: Feb. 5, 2020. [Online]. Available: https://www.kali.org/
[35] *Node-RED*. Accessed: Dec. 14, 2019. [Online]. Available: https://nodered.org/
[36] *HiveMQ*. Accessed: Jan. 13, 2020. [Online]. Available: https://www.hivemq.com/
[37] *Vulnerable Public PHP Website*. Accessed: Jan. 24, 2020. [Online]. Available: http://testphp.vulnweb.com/
[38] *Microsoft Azure IoT Hub*. Accessed: Jan. 27, 2020. [Online]. Available: https://azure.microsoft.com/en-au/services/iot-hub/
[39] *Amazon Web Services (AWS) Lambda*. Accessed: Jan. 27, 2020. [Online]. Available: https://aws.amazon.com/lambda/
[40] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 1–8.
[41] *NSX-Vmware*. Accessed: Feb. 3,2020. [Online]. Available: https://www.vmware.com/au/products/nsx.html
[42] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 3rd Quart., 2015.
[43] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A new threat intelligence scheme for safeguarding industry 4.0 systems," *IEEE Access*, vol. 6, pp. 32910–32924, 2018.
[44] *DVWA*. Accessed: Feb. 2, 2020. [Online]. Available: http://www.dvwa.co.uk/
[45] *Metasploitable3*. Accessed: Feb. 2, 2020. [Online]. Available: https://github.com/rapid7/metasploitable3
[46] *OWASP Security*. Accessed: Feb. 2, 2020. [Online]. Available: https://owasp.org/www-project-security-shepherd/
[47] *Secur. Onion*. Accessed: Feb. 2, 2020. [Online]. Available: https://securityonion.net/
[48] *Node-RED-Contrib-Modbus*. Accessed: Dec. 14, 2019. [Online]. Available: https://flows.nodered.org/node/node-red-contrib-modbus
[49] *ThingPulse ESP8266 Weather Station*. Accessed: Jan. 22, 2020. [Online]. Available: https://github.com/ThingPulse/esp8266-weather-station
[50] A. Alsaedi and N. Moustafa. (2020). *T-IIoT Dataset Evaluation*. [Online]. Available: https://github.com/A-Alsaedi/T-IIoT-Dataset-Evaluation-
[51] *ToN_IoT Attacks Scenarios*. Accessed: Jan. 26, 2020. https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i
[52] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 2nd Quart., 2013.
[53] J. Krupp, M. Backes, and C. Rossow, "Identifying the scan and attack infrastructures behind amplification DDoS attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1426–1437.
[54] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Rothwell, IN, USA: Insecure, 2009.

[55] *Nessus: A Securce Vulnerability Scanning Tool*. Accessed: Jan. 21, 2020. [Online]. Available: https://www.cs.cmu.edu/ dwendlan/personal/nessus.html
[56] *DoS & DDoS Attack using Scapy*. Accessed: Feb. 2, 2020. [Online]. Available: https://www.tutorialspoint.com/python_penetration_testing/python_penetr%ation_testing_dos_and_ddos_attack.htm
[57] *UFONet Toolkit*. Accessed: Feb. 9, 2020. [Online]. Available: https://ufonet.03c8.net/
[58] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Comput. Secur.*, vol. 74, pp. 144–166, May 2018.
[59] M. Al-Hawawreh, F. D. Hartog, and E. Sitnikova, "Targeted ransomware: A new cyber threat to edge system of brownfield industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7137–7151, Aug. 2019.
[60] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6822–6834, Aug. 2019.
[61] *Cross-Site-Script Toolkit*. Accessed: Feb. 9, 2020. [Online]. Available: https://xsser.03c8.net/
[62] T. Nelso and M. Chaffin, "Common Cybersecurity Vulnerabilities in Industrial Control Systems," *Control Syst. Secur. Program. Washington DC: Dept. Homeland Secur. (DHS), Nat. Cyber Secur. Division*, 2011.
[63] *CeWL Package*. Accessed: Jan. 27, 2020. [Online]. Available: https://tools.kali.org/password-attacks/cewl
[64] *Hydra Package*. Accessed: Jan. 27, 2020. [Online]. Available: https://tools.kali.org/password-attacks/hydra
[65] *Ettercap*. Accessed: Feb. 10, 2020. [Online]. Available: https://www.ettercap-project.org/
[66] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts Techninqs*. Amsterdam, The Netherlands: Elsevier, 2011.
[67] X. Wu, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
[68] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," *Procedia Technol.*, vol. 4, pp. 119–128, Oct. 2012.
[69] M. S. Mok, S. Y. Sohn, and Y. H. Ju, "Random effects logistic regression model for anomaly detection," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 7162–7166, Oct. 2010.
[70] A. Géron, *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Newton, MA, USA: O'Reilly Media, 2017.
[71] P. A. A. Resende and A. C. Drummond, "A survey of random forest-based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, p. 48, 2018.
[72] B. Subba, S. Biswas, and S. Karmakar, "Intrusion detection systems using linear discriminant analysis and logistic regression," in *Proc. Annu. IEEE India Conf. (INDICON)*, Dec. 2015, pp. 1–6.
[73] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[74] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short-term memory networks for anomaly detection in time series," *Universitaires de Louvain*, vol. 89, pp. 89–94, Oct. 2015.
[75] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistics Learning*, vol. 112. Cham, Switzerland: Springer, 2013.
[76] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1–5.
[77] *Grid Search*. Accessed: Aug. 1, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html
[78] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[80] *NumPy*. Accessed: Jan. 27, 2020. [Online]. Available: https://docs.scipy.org/doc/numpy/reference/
[81] *Scipy*. Accessed: Jan. 27, 2020. [Online]. Available: https://docs.scipy.org/doc/
[82] *Pandas*. Accessed: Jan. 27, 2020. [Online]. Available: https://pandas.pydata.org/
[83] *Scikit-Learning*. Accessed: Jan. 27, 2020. [Online]. Available: https://scikit-learn.org/stable/
[84] *Tensorflow*. Accessed: Aug. 1, 2020. [Online]. Available: https://www.tensorflow.org/
[85] *Keras*. Accessed: Aug. 1, 2020. [Online]. Available: https://keras.io/
[86] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.
[87] S. Geller. *Normalization vs Standardization Quantitative Analysis*. Accessed: Jan. 27, 2020. [Online]. Available: https://towardsdatascience.com/normalization-vs-standardization-quantit%ative-analysis-a91e8a79cebf
[88] I. Guyon, "A Scaling Law for the Validation-set Training-set Size Ratio," *AT&T Bell Lab.*, vol. 15, pp. 1–11, Oct. 1997.

ABDULLAH ALSAEDI received the master's degree in network systems from Swinburne University, Australia, in 2018. He is currently pursuing the Ph.D. degree in cybersecurity and distributed systems with RMIT University. His research interests include intrusion detection, cybersecurity of cyber-physical systems, and machine learning and deep learning applied to cybersecurity.

NOUR MOUSTAFA (Senior Member, IEEE) received the bachelor's and master's degrees in computer science from the Faculty of Computer and Artificial Intelligence, Helwan University, Egypt, in 2009 and 2014, respectively, and the Ph.D. degree in cybersecurity from UNSW, Canberra, ACT, Australia, in 2017. He was a Postdoctoral Fellow with UNSW Canberra from June 2017 to December 2018. He is currently a Lecturer with the University of New South Wales, Canberra, and Helwan University, Egypt. His areas of interests include cybersecurity, in particular, network security, intrusion detection systems, statistics, deep learning, and machine learning techniques. He holds several research grants with totaling over 1.2 million. He is also an ACM member and CSCRC Fellowship. He was awarded the 2020 prestigious Australian Spitfire Memorial Defence Fellowship Award. He has also served over seven conferences in leadership roles, involving Vice-Chair, Session Chair, TPC member, and proceedings chair, including the 2020 IEEE TrustCom and 2020 Australasian Joint Conference on Artificial Intelligence. He has published many research outputs in top-tier computing and security journals and conferences, such as the IEEE TRANSACTIONS ON FORENSICS AND SECURITY, IEEE IoT, and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He has served his academic community, as the Guest Associate Editor of the IEEE transactions journals, including the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE IoT Journal, as well as the journals of IEEE ACCESS, *Future Internet and Information Security Journal: A Global Perspective*.



ZAHIR TARI received the bachelor's degree in mathematics from the University of Algiers (USTHB), Algeria, in 1984, the M.Sc. degree in operational research from the University of Grenoble, France, in 1985, and the Ph.D. degree in computer science from the University of Grenoble, in 1989. His expertise is in the areas of system's performance, i.e., P2P, Cloud, Edge, IoT, as well as system's security, i.e., SCADA, Smart Grid, Cloud, IoT. He is currently a Full Professor in distributed systems with RMIT University, Australia. He is the coauthor of several books (John Wiley, Springer) and has edited over 25 conference proceedings. He is also a recipient of over 10M in funding from the Australian Research Council (ARC) and lately a part of the successful 7th Framework AU2EU (Australia to European) bid on Authorization and Authentication for Entrusted Unions. Finally, he is an Associate Editor of *ACM Computing Surveys* and was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS (TC), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), and the *IEEE Magazine on Cloud Computing*.

ABDUN MAHMOOD (Senior Member, IEEE) received the B.Sc. degree in applied physics and electronics, and the M.Sc. (research) degree in computer science from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively, and the Ph.D. degree from the University of Melbourne, Australia, in 2008. He worked at UNSW, RMIT, Melbourne University, and the University of Dhaka. He has been working as an Academic in computer science since 1999. In 2011, he joined UNSW as a Lecturer in computer science, until he joined La Trobe University, Melbourne, in 2017, where he is currently working as an Associate Professor in cybersecurity. He is currently with the Department of Computer Science, School of Engineering and Mathematical Sciences, La Trobe University. His research interests include data mining techniques for scalable network traffic analysis, anomaly detection, and industrial SCADA security. He has published his work in various IEEE Transactions and A-tier international journals and conferences.

ADNAN ANWAR received the master's by research and Ph.D. degrees from UNSW. He has worked as a Data Scientist at Flow Power. He has over eight years of research, and teaching experience in universities and research labs, including NICTA, La Trobe University, and University of New South Wales. He is currently a Lecturer in cybersecurity with the School of Information Technology. He is broadly interested in the security research for critical infrastructures, including smart energy grid, SCADA system, and application of machine learning and optimization techniques to solve cybersecurity issues for industrial systems. He has been a recipient of several awards, including the UPA scholarship, UNSW TFR scholarship, best paper award, and several travel grants, including ACM and Postgraduate Research Student Support (PRSS) travel grants. He has authored over 30 articles including journals (mostly in Q1), conference articles, and book chapters in prestigious venues.