



## Article

# A Hadoop Based Framework Integrating Machine Learning Classifiers for Anomaly Detection in the Internet of Things

Ikram Sumaiya Thaseen <sup>1</sup>, Vanitha Mohanraj <sup>1</sup>, Sakthivel Ramachandran <sup>2</sup>, Kishore Sanapala <sup>3</sup>   
and Sang-Soo Yeo <sup>4,\*</sup> 

- <sup>1</sup> School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India; isumaiyathaseen@vit.ac.in (I.S.T.); mvanitha@vit.ac.in (V.M.)  
<sup>2</sup> School of Electronics Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India; rsakthivel@vit.ac.in  
<sup>3</sup> Department of ECE, Marri Laxman Reddy Institute of Technology and Management, Hyderabad 500043, Telangana, India; Kishore.technova@gmail.com  
<sup>4</sup> Department of Convergence Computer & Media, Mokwon University, Daejeon 35349, Korea  
\* Correspondence: sangsooyeo@gmail.com; Tel.: +82-42-829-7636

**Abstract:** In recent years, different variants of the botnet are targeting government, private organizations and there is a crucial need to develop a robust framework for securing the IoT (Internet of Things) network. In this paper, a Hadoop based framework is proposed to identify the malicious IoT traffic using a modified Tomek-link under-sampling integrated with automated Hyper-parameter tuning of machine learning classifiers. The novelty of this paper is to utilize a big data platform for benchmark IoT datasets to minimize computational time. The IoT benchmark datasets are loaded in the Hadoop Distributed File System (HDFS) environment. Three machine learning approaches namely naive Bayes (NB), K-nearest neighbor (KNN), and support vector machine (SVM) are used for categorizing IoT traffic. Artificial immune network optimization is deployed during cross-validation to obtain the best classifier parameters. Experimental analysis is performed on the Hadoop platform. The average accuracy of 99% and 90% is obtained for BoT\_IoT and ToN\_IoT datasets. The accuracy difference in ToN-IoT dataset is due to the huge number of data samples captured at the edge layer and fog layer. However, in BoT-IoT dataset only 5% of the training and test samples from the complete dataset are considered for experimental analysis as released by the dataset developers. The overall accuracy is improved by 19% in comparison with state-of-the-art techniques. The computational times for the huge datasets are reduced by 3–4 hours through Map Reduce in HDFS.

**Keywords:** Hadoop; Internet of Things; anomaly detection; artificial immune network; hyperparameters; K-nearest neighbor; naïve Bayes; support vector machine



check for updates

**Citation:** Thaseen, I.S.; Mohanraj, V.; Ramachandran, S.; Sanapala, K.; Yeo, S.-S. A Hadoop Based Framework Integrating Machine Learning Classifiers for Anomaly Detection in the Internet of Things. *Electronics* **2021**, *10*, 1955. <https://doi.org/10.3390/electronics10161955>

Academic Editor: Mehdi Sookhak

Received: 28 June 2021

Accepted: 10 August 2021

Published: 13 August 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Internets of Things provide internet enabled platform to link heterogeneous devices. Each of these devices are enabled to act smart with the addition of network connection and processing power to them [1]. With the rise of IoT technology, smart home technology has entered the market for controlling the doors, lights, and other appliances in wireless mode. Due to the absence of an administrator [2], various home devices are prone to attacks. Smart home devices were also targeted with information destruction, illegal physical access and privacy violation attacks. These attacks decrease the security level of IoT devices. Hence, it is necessary to develop a robust framework for identifying the IoT attack in an IoT ecosystem in an efficient manner using machine learning models. Traditional classifiers may get biased towards the majority class samples, and this can cause misclassification of the minor but significant class instances. Information loss can be reduced by eliminating such instances using a modified Tomek-linked under-sampling which is detailed in Section 3.

### 1.1. Attacks in the IoT Network

The various botnet attacks namely Mirai, Bashlite, and Hajime pose huge challenges for IoT security. The different categories of botnet attacks are DDoS, identity theft, leakage of information, keylogging, and pushing [3]. The botmasters perform network mapping to gather the OS information by fingerprinting and port scanning to find vulnerabilities and infect the IoT devices. The major purpose to launch a DDoS attack by a botnet is to render the service inaccessible to authentic users [4]. These attacks are targeted in the network and application layer of the IoT environment. Besides, the application layer also has to handle the security issues. The violations of privacy and data disclosure to the users were the challenges addressed in the research study [5]. Amongst the various fields of study, the most prominent and ever progressing one has been the study of network traffic for attack detection and mitigation. The advancements in the technologies have proportionately increased the network attacks in various environments like IoT, IIoT, etc., and therefore novel enhanced approaches are required to cope with the latest updates. Distributed machine learning with the development of new tools and frameworks like Hadoop provides a colossal scope of development in this direction [6]. Hence, in this paper a distributed machine learning based approach is deployed to identify the presence of malicious behavior in IoT traffic.

The contribution of this proposed work is as follows:

- Hadoop based framework to distribute the processing of huge IoT traffic datasets for minimizing the computational time.
- Class imbalance of IoT datasets is mitigated by deploying a modified Tomek-linked under-sampling technique.
- Tuning the individual machine learning models like SVM, KNN, and NB using artificial immune network for better cross-validation accuracy.
- The proposed model is evaluated with benchmark IoT datasets to show the superiority of the technique over existing baseline approaches.

Superior performance of the modified Tomek-linked under-sampling (MTL) in comparison to the baseline sampling approaches namely, random under-sampling (RUS), condensed nearest neighbor (CNN), and Tomek-linked under-sampling has motivated to adapt in our proposed approach. A hyperparameter tuning using AiNet improves the cross-validation accuracy before predicting the accuracy on BoT-IoT and ToN-IoT datasets. The performance of the MTL under-sampling with various classifiers has been validated in the literature with 10 real life data sets [7]. Few of the datasets which were highly unbalanced, resulted in enhanced performance on various metrics like precision, recall, specificity, fall out, F-measure, Mathews correlation coefficient (MCC), and area under the ROC curve (AUC). Hence, this technique is chosen for performing an under sampling in our proposed model for improved performance.

The remaining of the paper is summarized in the following manner: Section 2 briefs the extensive literature of IoT security models developed in the recent few years. Section 3 discusses the techniques utilized to implement the proposed approach. Section 4 details the proposed Hadoop based framework for botnet traffic identification. Section 5 outlines the experimental analysis. Performance analysis is detailed in Section 6. The conclusion is given in Section 7.

### 1.2. Intrusion Detection Systems (IDS)

The security threats on the internet and consumer risks are increasing with the invention of IoT. An example is the DoS attack on Dyn [8], which is a DNS provider in the year 2016 and also Mirai botnet attacks on devices by misusing default credentials [9,10]. The major reason for the effectiveness of Mirai Botnet is because of the easy installation of IoT devices, which were developed with minimal or no security concern and reduced cost [11]. A vulnerable IoT device is very risky, irrespective of its level of security [12]. There were nearly 10,263 botnets introduced in 2018 in various IoT devices [13]. There were also 13,000 IoT devices which launched a powerful DDoS attack on IoT devices discovered in

2017 [14]. IoT security would assist measurement platforms to implement innovative user interfaces for home networks to discover and filter botnet traffic [15]. Besides, users can be notified about malicious activities. There is a challenge associated with traffic classification in the IoT domain because there are very few platforms that analyzes this issue [16]. A limited research is done on IDS using ML deployed on IoT networks [17,18]. The primary reasons are the lack of datasets and also real hardware deployed for all datasets consisting of simulated IoT devices [19].

Figure 1 shows typical IDS for IoT environment. The IoT components are monitored by the detection system and the response mechanisms act against the assumed attacks as soon as it is known that the traffic is not a legitimate one. A machine learning technique deploys a learning classifier to train the detection model. Misuse based techniques categorize the incoming traffic by deploying pattern matching algorithms. Traditional IDS cannot be implemented on ordinary IoT devices. Hence, network-based IDS is appropriate to build an IDS for an IoT environment. Huge challenges for Anomaly Detection Systems (ADS) are high false positives but they are effective for novel attacks. The wide diversity of IoT devices is also a challenge for implementing ADS. The storage capacity and computation power require the IDS for IoT to be lightweight.

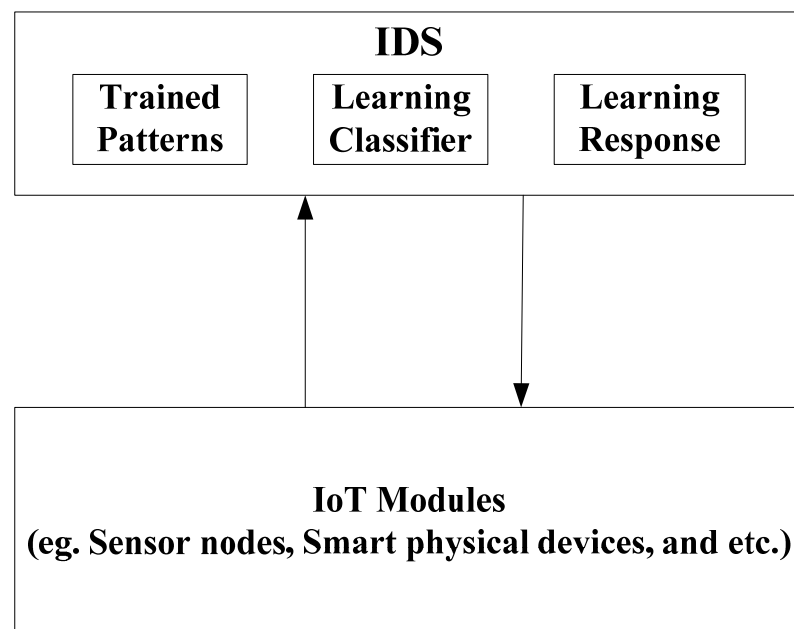


Figure 1. IDS for IoT environment.

## 2. Related Work

An important step in product development is the security testing of IoT devices before they are introduced in the market. In this section, a comprehensive literature review is provided which analyzes various studies of IoT device susceptibilities, testbeds developed for each IoT device, and various machine learning-based IDS. Security breaches and vulnerabilities of IoT devices have been studied by various researchers [1,13,20,21], for implementing security mechanisms. The major challenges to test IoT devices are their features and limitations [22]. A comprehensive IoT security test bed was developed [23]. Security weakness of IoT devices was evaluated by launching attacks [14,24,25]. The research on the network and firmware analyses, identified the private address of the user from IoT device, monitored traffic and utilized proxy TLS traffic for data extraction. The security of the network protocols due to the DoS attack on VoIP [26] is studied and enhanced. Constrained IoT devices which implement Constrained Application Protocol (CoAP) were examined [27]. Table 1 specifies the analysis of various IoT attacks, tools deployed to initiate the attack, and their effects. It is inferred that different attacks were

targeted on IoT devices and there is a need to develop robust frameworks to protect from these attacks.

**Table 1.** Analysis of various IoT attacks and results.

Attack	Products Tested	Tools	Attack Approach	Results
Devices exposed to peripheral IP address to crash the server by creating a Trojan horse. [28]	Nest Thermostat	-	Backdoor and Hardware access by physical tampering	Modifiable firmware and checksum
Channel interception or attacks on running apps in [29].	Smart TV	Binwalk	Firmware extraction by compromising devices in public network ADSL. Web browser vulnerable to XSS attacks.	Attack on firmware modification as it is updated in an insecure channel.
Kiddie Scripts utilized to exploit non-IT devices [24]	Nest Thermostat	Wireshark Kiddie scripts Ettercap Forensic Toolkit (FTK) Autopsy	Gaining credentials by physical access Packet analysis	Unable to gain root access.
Different attacks on Haier home systems [20]	Haier Smart Home Automation	Wireshark, UART	Brute force approach to access password. Gaining root access Network analysis	Root shell access by exposing telnet credentials. Updates on firmware sent in clear text.
Smart bulb security issues [25]	Limitless LED Philips Lux	Own receiver introduced	API used to extract secret information Control packets have eavesdropped	MITM attack caused the private data to be exposed.
Injected malware in an iOS mobile application. Peripheral IP devices were attacked [30]	Camera Netgear Nightawk WeMO plug	iOS app Scan results obtained from cloud-hosted server	Devices are located by nearby LANs Devices exposed to public IP address	Server enabled the attack on exposed devices. Device responses are collected by SSDP to check for IoT devices.
Smart socket reversed communication [31]	Socket Edimax plug	Python scripts	Brute Force Spoofing scanning Firmware	Device authentication absent. Insecure communication protocols Feeble password policy.
Communication protocols analyzed with Edimax IP camera architecture and vulnerabilities [32]	Camera	-	Determine all MAC combinations to check the status of online devices. Brute Force credentials.	Connection status exposed by the camera. Brute force attack. Authentication information is obtained by spoofing attack which impersonates real cameras.
IP Camera traffic is analyzed [33]	IP Cameras	Nmap Wireshark	MITM and network analysis Video streams obtained by Brute force port RTSP	Commands/credentials obtained in clear text. Video streams not obtained Real-time streams are exposed in RSTP port.
Insecam website sued to retrieve live streams of open cameras [34]	IP Cameras	Domain scans of angry IP	Check open-access devices	Passwords not set for many IP cameras.

Table 1. Cont.

Attack	Products Tested	Tools	Attack Approach	Results
Attacks and vulnerabilities on Fitbit analyzed [35]	Fitbit	Gattool APKtool	Modify protocols by analyzing firmware. Information leakage Access cloud by modifying mobile app of Fitbit	Leakage of information Firmware compromised Gain access to cloud by modifying app.
Smartwatches tested for exploiting BLE protocol [36]	Fitbit Keyboard LG watch	Wireshark	Sniffing of packets	Unable to read personal data by Ubertooth. Packet formats not recognizable due to unawareness.

### 2.1. Vulnerability Mitigation Approaches and Security Testbeds

Penetration testing of IoT is performed and architecture was developed [37]. Testing of various IoT devices such as smart home, wireless sensor networks (WSN), and smart wearables is done to identify the security holes. An integrated AI security framework which exploits machine learning approaches to detect new kind of cyber-attacks in IoT is implemented [38]. Tempo-spatial correlation between different sensor data is analyzed for threat identification. In this, knowledge-based and anomaly-based IDS were integrated. Different IDS architectures are developed [39] for IoT devices. A semi-distributed technique results in an accuracy of 99.97% and long CPU time of 186.26 s. However, in a distributed approach, the accuracy obtained is only 97.8% but CPU time is relatively low of about 73.52 s. The constraint of the work is updated datasets on IoT are not available. A cost-effective IDS for IoT was proposed [40], by cooperating between individual sensors and the edge routers. DoS and botnet attacks are effectively identified by correlating events from multiple IoT devices and thereby improve the detection rate and minimizing performance overhead. An IoT dataset for dealing with DDoS attacks is generated [41], and used to build models to prevent DDoS attacks. IDS datasets are limited for evaluation and testing, therefore it is critical to confirm maximum interoperability through various developing technologies. An IDS is developed to detect and minimize ping of death attacks and filter out packets that are more than the required length [42]. A search strategy is used to solve the problem.

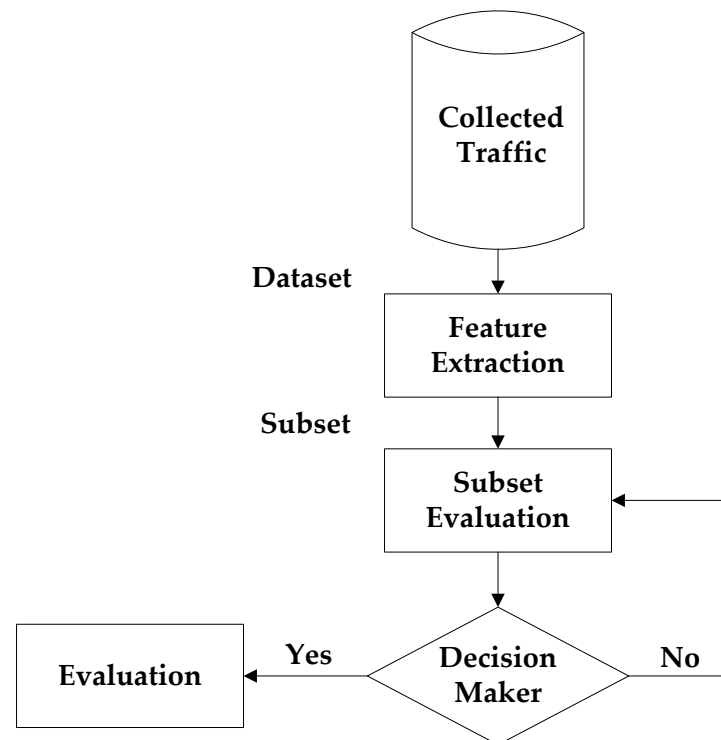
### 2.2. IDS for IoT Using Machine Learning

IDS in the IoT environment have fascinated many researchers and developers due to increase in the real time applicability of IoT devices. Machine learning techniques are implemented for IDS in an IoT environment in many current studies.

An IDS based on LDA and single hidden layer neural network ELM is developed for IoT [43]. A good generalization and detection accuracy of 92.35% is obtained which is superior in comparison to other approaches. An anomaly-based intelligent IDS named Passban is developed to analyze information collected from various IoT sources and identify cyber-attacks which have different flow pattern in comparison to normal events [44]. A real time IoT testbed is deployed on a resource-constrained home automation environment. Malicious traffic is detected with low false positives and high accuracy. Local Outlier Factor (LOF) and isolation Forest are deployed for class predictions which are one-class classification approaches. A supervised three-layer IDS is developed to identify cyber-attacks in IoT networks [45]. The proposed IDS can automatically differentiate between normal and benign network activity with a maximum f-measure of 96.2%.

A detection system was proposed for sinkhole attacks which targeted the routing devices [46]. In another analysis [47], battery exhaustion attacks were prevented on a low energy Bluetooth network. A novel IDS was built using a rule-based architecture for IoT environment [48]. The effort was to predict abnormal activity and identification

of malicious IoT nodes. Naive Bayes was used as the classifier and open-source tool Weka was used for performance evaluation. Another study proposed a machine learning-based forensic mechanism for IoT botnets and Weka was used to determine the detection accuracy [49]. A lightweight IDS on Raspberry Pi is implemented using machine learning to protect against attacks in the IoT environment. A feature selection technique was integrated to develop a lightweight system [50]. Ahmad et al. [51], built a machine learning framework with CFS to identify normal behavior in Command-and-Control Communication channels and malicious behavior. Real-world data sets were used for evaluation and various cost-sensitive approaches integrated with various feature combinations to yield a superior result. A Trust aware Collaborative Learning Automata IDS (T-CLAIDS) is developed for VANETs [14]. The abnormal events are detected better using the proposed approach. The scheme outperforms other approaches in terms of false alarm ratio, detection ratio, and overhead. A standard cryptographic technique is used to develop intrusion detection in the distributed vehicular cloud [52]. A 10% of detection rate in improvement is obtained when compared with existing techniques. Figure 2 shows the traffic detection process of BoT-IoT attacks. Lightweight IDS are necessary for IoT whereas existing IDS only target high accuracy with a low false alarm.



**Figure 2.** Traffic detection process of BoT-IoT attacks.

### 2.3. Enhancements for Upcoming IoT Applications

A standard and a well-built framework for an IoT application are not available. Hence, substantial improvements in the current IoT application are needed to render it trustworthy, robust, and secure. Therefore,

- Severe penetration testing of IoT devices is required to analyze the risk levels in device installation for various applications. A priority list can be made according to the risk involved and appropriately the devices can be deployed.
- Different layers of IoT and protocols are using encryption techniques. There are different phases in the application such as encrypt, decrypt, and re-encrypt which makes the system susceptible to threats. Hence, a viable solution suggested for preventing various threats is end-to-end encryption.



- An authentication service must be implemented for interaction between devices. Digital certificates can provide seamless authentication that is integrated with cryptographic protocols could be a promising solution
- Encryption approaches like RSA and hashing techniques like SHA256 or hash chains must be deployed for securing the user and environment.
- Cloud services are used by most of the applications for storage and retrieval of data and hence cloud storage risks must be analyzed. Data security can be enhanced by storing the encrypted data in the cloud and the provider cannot decrypt any ciphertext.
- IoT devices can be secured by the use of artificial-intelligence-based approaches.

#### 2.4. Class Imbalance Problem in IoT Datasets

There are two major categories to remove class imbalance problem in datasets.

- Data level solutions
- Algorithmic level

Data level solutions result in a balanced data distribution among the class labels by redistributing the samples in the data space. Sampling is an elementary technique to achieve class distribution balance within a dataset, either by adding examples into the minority class called as oversampling or by removing examples from the majority class, called as under-sampling. Tomek-linked under-sampling is widely preferred to eliminate boundary samples [7]. Algorithmic level approaches result in an optimal solution for class imbalance problem by modifying the classifiers. A cost function is defined against misclassification in cost sensitive approaches and is one of the efficient algorithmic based approaches. Various techniques for improving the class imbalance problem are discussed in detail in the next section. In our proposed approach, the IoT traffic is highly imbalanced—i.e., Minority IoT attack samples—are very less in comparison to normal IoT traffic. Therefore, an algorithmic level under-sampling technique is adopted and deployed in the framework.

### 3. Methods and Framework Adopted for Proposed Approach

This section discusses the approaches primarily used to implement the proposed scheme in detail.

#### 3.1. Identification of Outlier

A revision is performed on the concept of minority samples related to each Tomek-linked majority samples. An element  $x_i$  is considered as an outlier or boundary sample according to the number of minority instances to which a majority sample,  $x_i$  is associated by a Tomek-link pair. The majority element  $x_i$  is reclined to the data distribution of the minority region and is responsible for deprivation in performance.

#### 3.2. Identification of Redundant and Noisy Samples

The similarity measure between samples is termed as redundancy. The proposed scheme employs three different similarity measures, namely, Euclidean Distance (ED), Cosine Similarity (CS), and City-block Distance (CD). Each of the individual measures is given in equations below:

- City-block distance ( $d_a$ ) is represented in Equation (1).

$$d_a = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{im} - x_{jm}| \quad (1)$$

and

$$\text{sim}(x_i, x_j) = \frac{1}{d_1} \quad (2)$$

(ii). Euclidean distance ( $d_b$ ) for two samples  $x_i$  and  $x_j$  is represented in Equation (3).

$$d_b = \sqrt{\{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{im} - x_{jm})^2\}} \tag{3}$$

and

$$\text{sim}(x_i, x_j) = \frac{1}{d_b} \tag{4}$$

(iii). Cosine similarity is a similarity degree which determines the cosine angle among two samples,  $x_i$  and  $x_j$  where the result is within the series  $[0, 1]$ . Cosine similarity is given in Equation (5).

$$\text{sim}(x_i, x_j) = \cos \theta \tag{5}$$

(iv). Euclidean dot product is defined by

$$\vec{x}_i \cdot \vec{x}_j = \vec{x}_i \times \vec{x}_j \cos \theta \tag{6}$$

(v). By substituting the value of  $\cos \theta$  in (6)

$$\text{sim}(x_i, x_j) = \frac{\sum_{i,j} (x_i, x_j)}{\sqrt{\sum_i x_i^2} \times \sqrt{\sum_j x_j^2}} \tag{7}$$

The majority pair samples which contribute to the maximum similarity score are removed.

### 3.3. Contribution Element, ( $Contr_x$ )

This element specifies the probability mass function  $f(\cdot)$  which specifies a fixed group of distributions, represented as  $\{f(\cdot|\theta), \theta \in x\}$ , where  $\theta$  represents model parameter elements and the entire observation set is represented as  $x$ .

An estimator for predicting the parameter true value,  $\theta$  is to be determined by the function  $f(\cdot)$ . Thus, Equation (8) represents joint density function belonging to a specific category.

$$f(x_1, x_2, \dots, x_n|C_l) = f(x_1|C_l) \times f(x_2|C_l) \times \dots \times f(x_n|C_l) \tag{8}$$

In Equation (8),  $C_l$  represents the majority class label and  $(x_1, x_2, \dots, x_n) \in S'_{maj}$  where  $S'_{maj}$  represents the redundant and noisy bulk samples.

The likelihood calculation of each sample,  $x_1 (1 \leq i \leq n, n = \text{total number of samples})$  is the objective of  $f(x_1|C_l)$ . This specifies the  $C_l$  possibility to be arranged in the discrete class label set (0 or 1); if the class label of  $C_l$  available in the dataset is true. The likelihood is represented in Equation (9) below

$$L(C_l|x_i) = f(x_i|C_l) = \prod_{i=1}^n f(x_i|C_l) \tag{9}$$

The disadvantage of maximum likelihood of zero estimation is overcome by the log-likelihood function while determining the class label is  $C_l$ . A modified log-likelihood function is a combination of each attribute value,  $x_{ik} (1 \leq k \leq a, \text{ where } a \text{ is the number of attributes})$  with the joint density function as given below

$$\ln L(C_l|x_{ik}) = \ln \prod_{i=1}^n \prod_{k \in 1,f} f(x_{ik}|C_l) = \sum_{i=1}^n \sum_{k=1}^a \ln f(x_{ik}|C_l) \tag{10}$$



Thus, the contribution term is represented as

$$Contr_x = \frac{1}{N} \times \frac{\ln L(C_l|x_{ik}) \times P(C_l)}{P(x_1, x_2, \dots, x_n)} = \frac{1}{N} \times \sum_{i=1}^n \sum_{k=1}^m \ln f(x_{ik}|C_l) \tag{11}$$

where, the prior probability of class is represented by  $P(C_l)$  and the log-likelihood function is specified as  $\ln f(x_{ik}|C_l)$ .

### 3.4. Hadoop Architecture Overview

Big Data platforms are flexible in terms of data acquisition as it supports structured and unstructured data. In addition, big data platforms like Hadoop are also used for data integration, aggregation, representation, and query processing. Multiple tools are deployed in the Hadoop environment thereby the computational complexity is minimized and readily programmable. Huge volume of IoT network traffic records need to be processed and Hadoop server can send the files to the server and obtain the desirable result. The Hadoop ecosystem utilizes the pig tool to aggregate, MapReduce, join and filter the dataset tuples to produce an optimal result. Figure 3 shows a high-level application architecture design of Hadoop. A master slave approach is followed by Hadoop for storage and data processing. The master node in HDFS is called as the NameNode and the slave nodes manage data storage and complex computations. DataNode is responsible for CPU intensive processes like statistics, machine learning tasks, language, and semantic analysis. In addition, I/O intensive activities like data import, export, clustering, search, decompression, and indexing are also executed by the DataNode. Thus, in our proposed model, Hadoop architecture is implemented for faster and efficient processing of data.

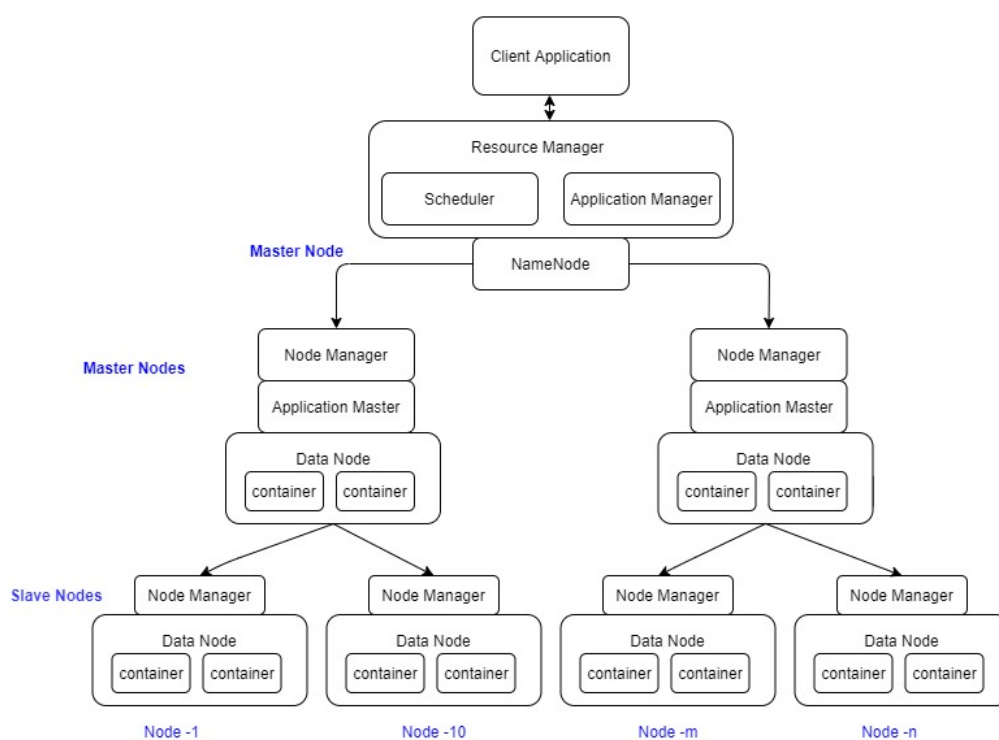


Figure 3. High level design of Hadoop application architecture.

## 4. Proposed IoT Intrusion Detection Model

### 4.1. Machine Learning to Predict IoT Threats

Different IoT attacks listed below can be effectively identified by machine learning techniques.

- DoS Attacks: A serious concern is DoS attacks that originate from IoT devices. Multi-Layer Perceptron (MLP) is one approach that secures networks against such attack. MLP trained by integrating particle swarm optimization with backpropagation technique [53] was proposed which can enhance the security of wireless networks. The detection accuracy will increase by using ML techniques and IoT devices will be secured from the DoS attacks.
- Eavesdropping: ML techniques are used for protecting from eavesdrop on messages during data transmission. ML techniques namely non-parametric Bayesian techniques [54] and Q-learning based offloading strategy [55] can be used.
- Spoofing: Attacks can be avoided using Dyna-Q, Support Vector Machines (SVM) [10], distributed FrankWolfe (DFW) [9], and Deep Neural Network [10], techniques. Classification accuracy increases using these techniques and the false alarm rate and average error rate gets reduced.
- Privacy Leakage: IoT application trust [56] is developed by commodity integrity detection algorithm (CIDA) developed from Chinese remainder theorem (CRT).
- Digital Fingerprinting: IoT systems can be secured by digital fingerprinting and thereby end users will be confident to utilize applications. Smartphones, payments, unlocking car and home doors are implementing fingerprinting. Nontraditional solutions are developed using various machine learning algorithms which are given below.

Machine learning detects undesirable events in IoT devices by training algorithms for data loss prevention and other security concerns. SVM algorithms are widely used in digital fingerprinting and compared their approach with traditional models. SVM is trained using the feature vector obtained based on the fingerprint pixel values. Artificial Neural Network (ANN) provides several benefits such as generalization, adaptive learning, and fault tolerance. A framework for identifying fingerprints digitally by using ANN has been developed [56]. Hence, ML is capable of securing the IoT environment. Liang et al. used machine learning techniques for analyzing the security models for IoT [57]. The authors presented many review techniques using ML for IoT security.

#### 4.2. Secure Model for IoT Traffic

The proposed model is developed in two phases as represented in Figure 4. A multi-level architecture is implemented in the proposed model. In the initial stage, the benchmarks IoT datasets are uploaded into the HDFS. Yarn, which is the resource manager of Hadoop manages the name node, data node and secondary name node of the cluster. The pig script is then executed in the Map Reduce node to implement the Map Reduce functionality. The processed data is saved in the HDFS for further analysis. In the next level, a data pre-processing is performed by standard scaling, label encoder, and under-sampling of majority labeled samples. In the under-sampling technique, the samples with majority class records concerning noise, contribution factor, and redundancy is performed. The final stage is the prediction of the IoT network traffic by deploying three classifiers KNN, SVM, and NB to predict the class labels. This approach improves the accuracy and minimizes the false alarm rate. A hyperparameter optimization is performed using artificial immune network. The desired accuracy is obtained in minimal amount of time because the datasets are processed in Hadoop environment which is discussed in the experimental section. The parameters of SVM, KNN, and naïve Bayes are tuned for optimal accuracy as shown in Tables 2 and 3. The pseudocode of the various classifier predictions given in Appendix A.

**Table 2.** Optimal aiNet parameters.

Parameters	Default Values	Optimized Values
Threshold suppression	0.1	0.1
Number of clones generated	30	50
Number of clones multiplier (N)	20	50
Maximum number of generations	100	50

Table 3. Tuned hyper classifier parameters.

Machine Learning Classifiers	Parameter Name	Default Parameter Value	Optimized Parameter Range
SVM	Sigma ( $\sigma$ )	0.7	(0.1,0.9)
	Cost (C)	1	(0.25,4)
KNN	Number of Neighbors (k)	3	(1,17)
	Exponent (E)	3	(0.5,5)
NB	Distribution	0	Normal (0,1)

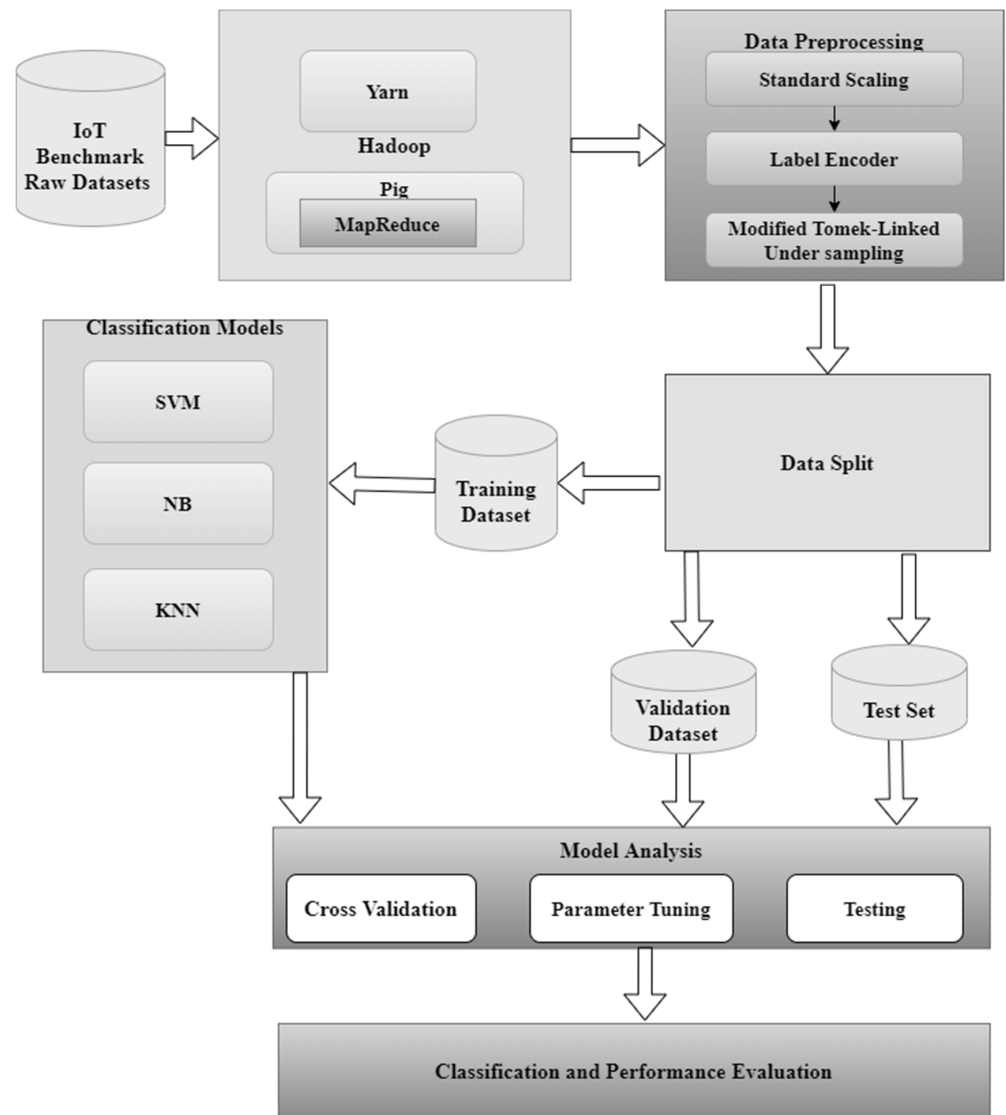


Figure 4. Proposed IoT intrusion detection model.

4.3. Data Pre-Processing

4.3.1. Identification of Outlier and Tomek-Link Pairs

Input:

- Dataset,  $D$  using ‘ $s$ ’ samples and the feature space consisting of ‘ $a$ ’ number of attributes.
- $S_{min}$ ,  $S_{maj}$ , and threshold ( $t$ ) to calculate the majority index samples,  $x_j$ .

Every sample is represented as  $(\vec{x}_m y_n)$ , where  $(\vec{x}_m)$  is the input vector for  $m$ -th sample, denoted as  $\vec{x}_m = (x_{m1}, x_{m2}, \dots, x_{mh})$ , where  $h$  is the total feature set,  $1 \leq m \leq s$ , and  $y_n = (y_{n1}, y_{n2}, \dots, y_{nc})$  is the selected class labels of the samples with a maximum of  $c$ -classes.

Notation:

- $S_{s \times h}$ —An imbalanced dataset, with ‘s’ number of samples and ‘a’ represents the number of attributes
- $S_{min}$ —Minority class set samples,  $S_{min} = \{x_i | i = 1, 2, \dots, p\}$ , ‘p’ is the sum of minority class samples.
- $S_{maj}$ —Majority class set samples,  $S_{maj} = \{x_j | j = 1, 2, \dots, q\}$ , ‘q’ is the sum of majority class samples.
- $Tk$ —Set of Tomek-linked samples.
- $Index_{x_j}$ —Total minority samples, to which majority sample,  $x_j$  is connected as Tomek-linked pair.
- $Tk'$ —Improved subset of  $Tk$ , after removing  $x_j$  samples, identified to be outliers.

Begin:

1. Split the complete dataset,  $S_{s \times l}$  into two subsets,  $S_{maj}$  and  $S_{min}$ .
2. For every  $x_i \in S_{min}$ , calculate the k-nearest neighbor using Euclidean measure,  $x_j, x_j \in S_{maj}$  for  $k = 1$ .
3. The nearest neighbor  $(x_i, x_j)$  pairs are included in the subset,  $Tk = \{(x_i, x_j) | x_i \in S_{min}, x_j \in S_{maj}\}$ .
4. For each  $x_j \in S_{maj}$ , determine index  $x_j$ .
5. If index  $x_j > h$ , where h represents threshold;  $x_j$  is identified as ‘outlier’, existing in the minority area of the subset,  $REM = \{x_j | \forall x_j, index\ x_j > h\}$  and are removed.
6. Updating the subset  $Tk$  to  $Tk'$  as  $Tk' = Tk - \{x_j\}$
7. Updating the subset  $S_{maj}$  to  $S'_{maj}$  as  $S'_{maj} = S_{maj} - \{REM\}$ .

Samples present in the set REM represent the outliers. There is a greater chance of samples creating Tomek-link pair with multiple minority samples to be placed on the incorrect side of the decision boundary. Samples belonging to  $Tk'$  are represented as Tomek-link paired samples which were sent to the subsequent phase for checking redundancy.

#### 4.3.2. Estimation of Redundancy among Majority Samples

The majority class instances which exist close to the decision margin are identified by redundancy and noise factors and are maximum similar to other majority samples. The similarity measures which has been considered are CB, CS, and ED as detailed in Section 3.2.

Input:

- $S'_{maj}, Tk', r$  = number of utmost redundant samples which helps to create,  $Redn$ . Set  $r = 1$ .

Begin:

1. For every  $x_m \in S'_{maj}$ , deploy the various distance measures like ED, CB and CS to calculate the redundant pairs,  $S'_{maj}$  specified in Equations (1)–(7) and transferred to the subset,  $Redn$  as  $Redn = \{(x_m, x_r) | \forall x_m, x_r \in S'_{maj} \text{ and } sim(x_m, x_r) = \text{maximum}\}$ .
2. A revised subset,  $T_1\_Redn$  is generated by eliminating the majority samples contributing to redundancy and noise. Thus, the intersection of  $T_1'$  and  $Redn$ , is given as:  $Tk\_Redn = \{\cap x_m \in S'_{maj} | x_m \in Tk, x_m \in Redn\}$ .

#### 4.3.3. Estimation of the Contribution Factor

The contribution element,  $Contr_e$  is deployed to the majority category samples within the set,  $Tk\_Redn$ . The steps to calculate  $Contr_e$  is detailed in Section 3.3.

#### 4.3.4. Under-Sampling of Majority Samples

The outliers  $x_j$  are the initial ones to be removed from the majority samples. These samples are determined based on their redundancy factor and contribution term. Noisy samples which contribute to maximum redundancy and minimum contribution will be

identified for removal. Hence, the objective function that can exclude a majority sample, from a pair which is redundant,  $(x_u, x_v)$  where  $(1 < u, v \leq j \leq q)$  is represented by Equation (12) as

$$\text{objective}(\text{elimination}_{x_u \text{ or } x_v}) \leftarrow \max\left(\frac{\text{sim}(x_u, x_v)}{\text{cont}_{x_u}}, \frac{\text{sim}(x_u, x_v)}{\text{cont}_{x_v}}\right) \quad (12)$$

$x_u, x_v \in Tk\_Redn$ , i.e., it is to be noted that the sample with the minimum contribution term will be removed if two noise samples are detected.

#### 4.3.5. Algorithm

The pseudo code of the proposed pre-processing phase is summarized in Algorithm 1 below:

---

##### Algorithm 1. Tomek-linked and redundancy-based under-sampling

---

###### Input:

A dataset,  $S$  with “ $n$ ” samples and the feature space using “ $h$ ” features. Each sample is represented by  $(\vec{x}_a, y_b)$  where  $\vec{x}_a$  is the vector input for  $a^{\text{th}}$  th-sample, identified as  $\vec{x}_a = (x_{a1}, x_{a2}, \dots, x_{ah})$  with a sum of  $h$ -features and  $y_b = (y_1, f_2, \dots, y_c)$  are the chosen category labels from the samples containing the sum of  $c$ -classes; where  $k$  = total of nearest neighbors with set  $k = 1$  to identify Tomek-link pair.

###### Begin:

Step 1: The two subsets of the data are,  $S_{min}$  and  $S_{maj}$ ;  $S_{min} : \{x_i | i = 1, 2, \dots, p\}$  and

$$S_{maj} : \{x_j | j = 1, 2, \dots, q\}$$

Step 2: Identification of Tomek-link pair and outlier:

2.1: For  $\forall x_i \in S_{min}$ , create a subset,  $Tk$  with its  $k$ -NN from  $S_{maj}$ , as

$$Tk = \{(x_i, x_j) | x_i \in S_{min}, x_j \in S_{maj}\}.$$

2.2: Determine  $\text{index}_{x_j} \leftarrow x_i$  to which  $x_j$  is linked,  $\forall x_j \in Tk$ .

2.3: Hold  $x_j$  in  $Tk$ , if  $\text{index}_{x_j} < h$ ; where  $h$  = threshold, otherwise, generate a subset,  $REM$  as  $REM = \{x_j | \forall x_j, \text{index}_{x_j} > h\}$ , and is selected for direct removal.  $Tk$  is updated to  $Tk'$  as  $Tk' = Tk - \{x_j\}$ .

2.4:  $S_{maj}$  is updated to  $S'_{maj}$  as  $S'_{maj} = S_{maj} - \{REM\}$ .

Step 3: Redundant pair identification:

3.1:  $\forall x_m \in S'_{maj} (1 < m \leq j)$ , construct a set,  $Redn$  by using similarity methods given in Equations (1)–(7) as  $Redn = \{(x_m, x_n) | \forall x_m \in S'_{maj}\}$ .

3.2: A modified subset,  $Tk\_Redn$  is built as the intersection of  $Tk'$  and  $Redn$  as  $Tk\_Redn = \{\cap x_m \in S'_{maj} | x_m \in Tk, x_m \in Redn\}$ .

3.3: For each  $(x_u, x_v) \in Tk\_Redn$ , determine contribution term for individual  $x_u$  or  $x_v$ , according to Equation (11).

3.4: Delete  $x_u$  or  $x_v (x_u, x_v \in Tk\_Redn)$ , if either  $x_u$  or  $x_v$  fulfils the objective of removal, specified in Equation (12).

3.5: Modify  $S'_{maj}$  to  $S''_{maj}$  as  $S''_{maj} \leftarrow S'_{maj} - Tk\_Redn$ .

3.6: Integrate the updated  $S''_{maj}$  and  $S_{min}$  to build a balanced dataset,  $S_B$  as  $S_B \leftarrow S_{min} \cup S''_{maj}$

**Output:** New dataset,  $S_B$  containing a balanced distribution of category labels

---

#### 4.3.6. Naive Bayes (NB)

NB belongs to the category of probabilistic algorithm wherein the probability of all the attributes and their outcome. This approach determines the event probability with regard to earlier events occurrence which is called as the posterior probability.

#### 4.3.7. KNN

KNN is the simplest of all the existing models because training will be done only while we classify the data [58,59]. Related data points will be grouped and also it examines the closest data point having the K value. It is most widely used in the intrusion detection. The KNN is used to examine the normal data and attack data.

#### 4.3.8. SVM

Support vector machine (SVM) is a supervised learning approach that practices a hyperplane to classify future predictions and to separate the training data. It splits a dataset into two categories and the decision boundaries which helps to classify the data points. SVM is widely deployed for intrusion detection as observable by the tremendous volume of studies done over the years [57].

The detailed pseudocode of each of the three classifiers for prediction is given in the Algorithm A1 of Appendix A section.

#### 4.4. Classification

The model deploys three classifiers. Each classifier in this stage is trained with the newly updated training set obtained after under-sampling. The prediction is then computed using the test set. The three classifiers used in the proposed approach are:

1. Naive Bayes;
2. KNN; and
3. SVM

#### 4.5. Artificial Immune Network (aiNet) for Hyperparameter Optimization

An optimization version (Opt-aiNet) which can be used in optimization problems is available in aiNet [60]. A population is grown in Opt-aiNet which consist of a network of antibodies and the population size can be dynamically adjustable.

The optimal hyper-parameters of machine learning techniques are determined by Opt-aiNet as shown in Figure 5. SVM consists of the hyper-parameters  $\sigma$  and C. The misclassification cost is represented by 'C' which represents the distance from the error and margin. A balance of 'C' and  $\sigma$  is very crucial because the model gets overfitted if there is a high value of C and  $\sigma$ . The hyperparameters of KNN are K which represents the number of neighbors and E is Minkowski distance. NB has the hyperparameter namely distribution type which can be normal or kernel density estimation. Table 2 shows the optimal aiNet parameters. Table 3 represents the optimal parameter range of different machine learning techniques obtained from aiNet.

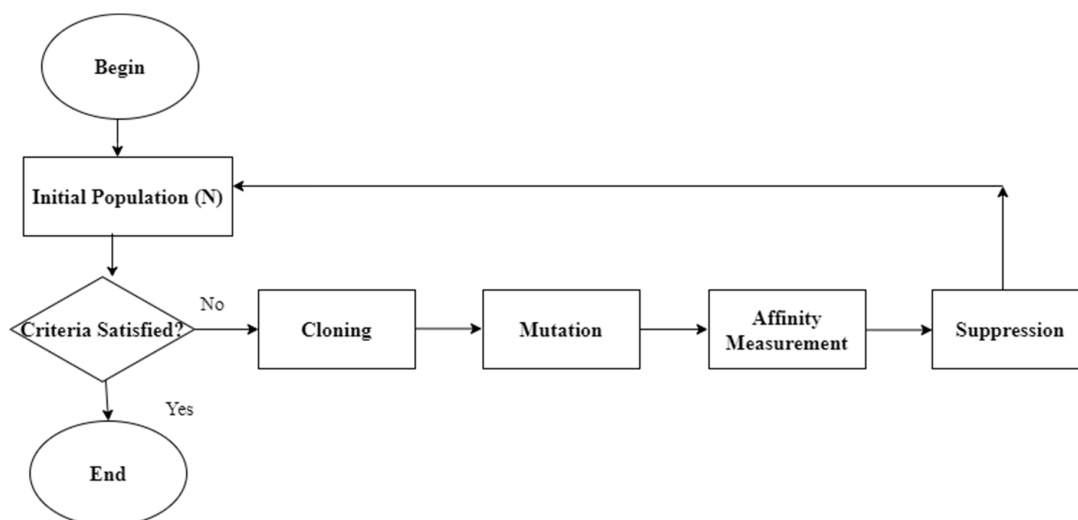


Figure 5. Flow diagram of opt-aiNet.

An initial population is created randomly with a size of 50. The antigen and antibody values are determined and the similarity between antibody and antigen ( $A_g$ ) is also determined. The similarity function is calculated as

$$\text{Similarity}(Ab_a, Ab_b) = \frac{1}{d(Ab_a, Ab_b)}, \quad (13)$$

where Similarity ( $Ab_x, Ab_y$ ) is the association among antibodies and  $d$  represents the euclidean distance between two antibodies  $Ab_a$  and  $Ab_b$ . The performance obtained by every classifier is determined and the optimization process is iterated till the specified number of generations perform satisfactorily. The similarity mutation is given as

$$C' = c + \gamma \cdot F(0, 1) \quad (14)$$

$$\alpha = (1/\beta) \cdot e^{-I^*} \quad (15)$$

where  $C$  and  $C'$  represent cell and mutated cell, the Gaussian random value of mean zero and standard deviation  $\sigma = 1$  is specified by  $F(0, 1)$ , the control parameter is given by  $\beta$  for determining the mutation range and the inverse exponential function decay,  $\gamma$  is the similarity proportional and  $i^*$  is the individual fitness. A mutation is accepted only when the value of ' $c$ ' lies in the domain range. In the final step, new antibodies replace the old ones. The fitness function process ends when the stopping criteria are reached. Thus, the hyper-parameters are optimized and the optimal accuracy is achieved.

## 5. Results and Analysis

The model is implemented in the Hadoop and Jupyter platform. The performance of the proposed scheme is evaluated with benchmark IoT datasets.

### 5.1. Datasets

#### 5.1.1. ToN-IoT Dataset

Data is collected from different heterogeneous data sources such as IIoT sensors, Operating Systems datasets of Windows and Linux from the UNSW Cyber Range and IoT labs. These are the recent new generations of IoT and IIoT datasets for evaluating the models built using machine learning approaches. The attributes of the dataset are different for the edge, fog, and cloud layer captured data samples which are detailed in [7]. Different attacks such as DDoS, DoS, and ransomware were implemented against IoT gateways, web applications and computer systems within the IIoT network. In the proposed approach, only edge layer data samples are used for analysis.

#### 5.1.2. BoT-IoT Dataset

A dataset generated from the UNSW Canberra Cyber Range lab is deployed for the analysis. The dataset encompasses both normal and malicious traffic. The different attacks included in the dataset are Keylogging, DDoS, DoS, OS, service scan, and data exfiltration attacks. The BoT-IoT dataset [6] is extracted using MYSQL by the researchers of that lab and they have utilized only 5% of the original samples as the original dataset is huge. The extracted 5% of data is approximately 1.07 GB of the total size, which is about 3 million records. This is the motivation for adopting Hadoop framework in our proposed work as the dataset is huge and can be processed only in a distributed cluster-based environment. The different attributes in the data are detailed in [61]. Table 4 shows the brief statistics of the various dataset files of BoT-IoT and ToN-IoT utilized for analysis. In this table, the number of samples, features, number of minority samples, majority samples, and imbalance ratio for every dataset is specified. Majority samples are the instances in the dataset containing larger amount of class labels. Minority samples are the instances in the dataset containing minimum amount of class labels. Tomek-linked under-sampling is widely preferred to eliminate boundary samples [7]. For example from Table 4, for the dataset Train\_Test\_Windows7, the following are Majority samples: Normal—



10,000 samples and Minority samples: Backdoor-1780 samples, DDoS-2135, Injection-999, Password-758, Ransomware-83, Scanning-227 and XSS-5. Imbalance ratio is calculated to determine which datasets will require an under sampling based on the minimum difference between minority and majority class labels. Thus, a modified Tomek-linked sampling is deployed to perform under sampling. Similarly, the same procedure is followed for all the datasets specified in Table 4. Imbalance ratio is the number of majority samples to minority samples. Low IR specifies the minimum difference between minority and majority class labels regarding sample count and hence those datasets will be preferred for under-sampling.

**Table 4.** Statistics of dataset.

Dataset	Dataset_Files	Samples	Features	Minority Samples	Majority Samples	Imbalance Ratio (IR)
ToN-IoT dataset	Train_Test_IoT_Fridge	59,944	5	24,944	35,000	1.40
	Train_Test_Iot_garage_door	59,587	5	24,587	35,000	1.42
	Train_Test_Iot_gps_tracker	58,960	5	23,960	35,000	1.46
	Train_test_iot_modbus	51,106	7	16,106	35,000	2.17
	Train_test_iot_thermostat	52,775	5	17,774	35,000	1.96
	Train_test_iot_weather	59,260	6	24,260	35,000	1.44
	Train_test_iot_motion_light	59,488	5	24,488	35,000	1.42
	Train_test_Network	461,043	43	161,043	300,000	1.86
	Train_test_Windows7	15,980	132	5980	10,000	1.67
	Train_test_Windows10	21,104	124	10,000	11,104	1.11
BoT-IoT dataset	Final_10_Best_testing	733,705	16	348,395	385,310	1.10

### 5.2. Comparison with Baseline Approaches

The proposed approach is compared with baseline models for performance. The various baseline approaches for comparative analysis are:

1. Random under-dampling (RUS): This is one of the basic techniques for undersampling to eliminate majority samples randomly.
2. Condensed nearest neighbor (CNN): This approach eliminates samples randomly concerning nearest neighbor rule.
3. Tomek-link: This is a modification of CNN which eliminates interior samples that are in the closeness of the decision boundary.

### 5.3. Setting Appropriate Values for Parameter ‘*t*’

In this approach, a parameter ‘*t*’ is deployed to identify an outlier from the samples of Tomek-link pairs. The main objective is to identify the outliers from the Tomek-associated samples. It is therefore required to select the value of ‘*t*’ in a medium scale. If the value of ‘*t*’ is very high, then it can incorrectly identify outlier as potential samples. Similarly, a low value of ‘*t*’ can delete important majority class samples. Selecting the ‘*t*’ parameter is influenced by the classifier and similarity type measure for a particular category. The impact of changing ‘*t*’ values over this proposed approach is evaluated for sensitivity analysis. It is observed that ‘*t*’ value ranging from 10–15 are appropriate. The ‘*t*’ value that results in the maximum accuracy for an explicit dataset is identified as the ideal ‘*t*’ value for a specific dataset. The investigations are accomplished on three classifiers and three distance measures as discussed in Sections 3 and 4.3. The sensitivity analysis results of BoT-IoT and ToN-IoT dataset is presented in Tables 5 and 6 respectively. The best train accuracy obtained is highlighted in tables for changing ‘*t*’ values. Table 7 presents the statistics of various IoT datasets on which under-sampling is executed.

**Table 5.** Sensitivity analysis of BoT-IoT dataset.

Distance Measure	Classifier	Average Accuracy with Changing Values of 't'					
		t = 10	t = 11	t = 12	t = 13	t = 14	t = 15
Euclidean distance	KNN	98.52%	99.26%	97.08%	95.04%	96.69%	97.23%
	SVM	96.24%	96.37%	97.37%	95.57%	94.37%	93.37%
	Naïve Bayes	96.23%	96.02%	95.76%	95.59%	95.76%	96.49%
City-block Distance	KNN	97.22%	95.22%	97.08%	96.89%	96.64%	96.64%
	SVM	97.08%	97.23%	96.37%	97.37%	96.89%	97.08%
	Naïve Bayes	95.91%	95.91%	95.47%	95.47%	95.47%	96.05%
Cosine similarity	KNN	97.22%	97.22%	98.22%	96.93%	96.78%	96.78%
	SVM	96.67%	96.79%	96.79%	96.85%	96.85%	97.08%
	Naïve Bayes	95.76%	96.05%	96.05%	95.85%	97.61%	95.61%

**Table 6.** Sensitivity analysis of ToN-IoT dataset.

Distance Measure	Classifiers	Average Accuracy with 't'					
		t = 10	t = 11	t = 12	t = 13	t = 14	t = 15
Euclidean distance	KNN	92.26%	92.37%	92.67%	90.89%	92.37%	91.37%
	SVM	91.29%	91.37%	91.37%	91.37%	91.37%	92.54%
	Naïve Bayes	94.45%	92.20%	90.59%	90.59%	92.34%	93.34%
City-block distance	KNN	93.67%	91.08%	91.89%	92.89%	90.79%	91.79%
	SVM	91.90%	92.23%	90.23%	93.23%	91.37%	92.22%
	Naïve Bayes	92.59%	90.76%	91.22%	92.76%	92.05%	91.05%
Cosine similarity	KNN	91.69%	91.37%	91.37%	91.69%	90.64%	92.12%
	SVM	93.02%	90.37%	92.37%	90.37%	91.37%	90.89%
	Naïve Bayes	92.64%	91.64%	90.64%	91.23%	92.61%	93.61%

**Table 7.** Imbalance ratio statistics of IoT datasets.

Dataset	Dataset_Files	IR (Prior Undersampling)	IR (Post Undersampling)
ToN-IoT dataset	Train_Test_IoT_Fridge	1.40	0.41
	Train_Test_Iot_garage_door	1.42	0.44
	Train_Test_Iot_gps_tracker	1.46	0.48
	Train_test_iot_modbus	2.17	1.12
	Train_test_iot_thermostat	1.96	0.94
	Train_test_iot_weather	1.44	0.47
	Train_test_iot_motion_light	1.42	0.44
	Train_test_Network	1.86	0.91
	Train_test_Windows7	1.67	0.90
	Train_test_Windows10	1.11	0.25
BoT-IoT dataset	Final_10_Best_testing	1.10	0.23

#### 5.4. Hyper Tuning Parameters Using AiNet

Tables 8 and 9 presents the different parameter values tuned by aiNet during cross-validation for SVM, KNN, and NB on BoT-IoT and ToN-IoT datasets, respectively. The parameter value which produces minimum training errors and optimum cross-validation accuracy are selected. The parameter setting performed by aiNet with varying parameter values results in the highest average accuracy for further experimental analysis. The proposed approach and the baseline techniques are analyzed by five-fold cross-validation.

**Table 8.** Hyper parameter tuning cross-validation results on BoT-IoT dataset.

Model Name	Precision	Recall	F1-Score	Accuracy	MCC Score
SVM	<b>92.05%</b>	91.62%	<b>92.22%</b>	90.07%	91.71%
NB	90.56%	91.12%	90.14%	<b>91.57%</b>	90.03%
KNN	91.86%	<b>91.93%</b>	91.07%	90.73%	<b>93.00%</b>

**Table 9.** Hyper parameter tuning cross-validation results on ToN-IoT dataset.

Model Name	Precision	Recall	F1-Score	Accuracy	MCC Score
SVM	97.15%	95.58%	94.26%	95.11%	95.71%
NB	<b>98.45%</b>	96.11%	96.15%	<b>98.59%</b>	98.03%
KNN	96.77%	<b>96.89%</b>	<b>98.10%</b>	97.73%	<b>99.05%</b>

### 5.5. Results

Tables 10–12 shows the graphical illustration of the average test accuracy achieved by the classifiers for all 11 datasets as described in Table 4. It can be observed from Tables 10–12, there is a substantial improvement of the proposed work in case of seven datasets which have imbalance ratio less than 1.5. There are four datasets which have imbalance ration more than 1.5. Low IR specifies the minimum difference between minority and majority class labels concerning sample count. The implementation of under-sampling is a good option in these cases because the elimination of a specific amount of potential weak samples creates a balanced dataset. The outliers, noisy, and redundant samples are removed from the common group. The removal of likely weak samples modifies the decision boundary near the minority region, thereby creating a satisfactory state for training the three classifiers, KNN, NB, and SVM.

**Table 10.** KNN classifier prediction on all datasets.

Datasets	Existing Approaches			Proposed Approach		
	RUS	CNN	TOMEK-LINK	Euclidean	City-Block	Cosine Similarity
Train_Test_IoT_Fridge	91.90%	92.70%	91.10%	91.90%	93.00%	94.90%
Train_Test_Iot_garage_door	90.80%	91.90%	89.80%	96.67%	92.81%	93.37%
Train_Test_Iot_gps_tracker	91.37%	92.17%	93.56%	94.68%	92.67%	93.82%
<b>Train_test_Iot_modbus</b>	<b>91.60%</b>	<b>92.80%</b>	<b>89.80%</b>	<b>91.00%</b>	<b>92.00%</b>	<b>91.40%</b>
<b>Train_test_Iot_thermostat</b>	<b>87.00%</b>	<b>86.00%</b>	<b>89.00%</b>	<b>90.00%</b>	<b>89.90%</b>	<b>88.78%</b>
Train_test_Iot_weather	90.30%	91.70%	91.50%	92.38%	92.78%	92.80%
Train_test_Iot_motion_light	90.00%	90.22%	90.30%	93.00%	92.00%	92.80%
<b>Train_test_Network</b>	<b>82.70%</b>	<b>80.20%</b>	<b>80.20%</b>	<b>81.80%</b>	<b>82.05%</b>	<b>80.00%</b>
<b>Train_test_Windows7</b>	<b>81.48%</b>	<b>80.60%</b>	<b>82.30%</b>	<b>81.70%</b>	<b>80.40%</b>	<b>80.00%</b>
Train_test_Windows10	92.80%	91.30%	92.50%	93.00%	93.40%	92.75%
Final_10_Best_testing	96.01%	97.21%	95.10%	98.10%	97.50%	98.40%

The datasets highlighted in bold in Tables 10–12 represent the low IR datasets wherein the undersampling is performed and resulted in improved performance.

**Table 11.** NB classifier prediction on all datasets.

Datasets	Existing Approaches			Proposed Approach		
	RUS	CNN	TOMEK-LINK	Euclidean Distance	City-Block Distance	Cosine Similarity
Train_Test_IoT_Fridge	89.10%	88.10%	89.60%	92.80%	91.10%	92.10%
Train_Test_Iot_garage_door	87.94%	88.37%	89.80%	90.80%	91.80%	92.80%
Train_Test_Iot_gps_tracker	89.30%	90.30%	91.24%	93.65%	94.10%	92.70%
<b>Train_test_Iot_modbus</b>	<b>87.50%</b>	<b>88.00%</b>	<b>86.90%</b>	<b>89.00%</b>	<b>89.30%</b>	<b>89.70%</b>
<b>Train_test_Iot_thermostat</b>	<b>86.40%</b>	<b>87.60%</b>	<b>88.70%</b>	<b>89.00%</b>	<b>90.00%</b>	<b>90.40%</b>
Train_test_Iot_weather	89.70%	88.50%	89.00%	90.14%	92.10%	93.30%

Table 11. Cont.

Datasets	Existing Approaches			Proposed Approach		
	RUS	CNN	TOMEK-LINK	Euclidean Distance	City-Block Distance	Cosine Similarity
Train_test_Iot_motion_light	92.30%	93.40%	91.30%	93.70%	94.00%	95.51%
<b>Train_test_Network</b>	<b>87.00%</b>	<b>88.73%</b>	<b>86.50%</b>	<b>89.70%</b>	<b>89.00%</b>	<b>88.40%</b>
<b>Train_test_Windows7</b>	<b>86.26%</b>	<b>88.00%</b>	<b>89.60%</b>	<b>90.10%</b>	<b>90.56%</b>	<b>91.00%</b>
Train_test_Windows10	89.90%	87.75%	89.40%	90.00%	91.30%	92.43%
Final_10_Best_testing	96.30%	97.30%	96.75%	98.40%	99.10%	99.30%

Table 12. SVM classifier prediction on all datasets.

Datasets	Existing Approaches			Proposed Approach		
	RUS	CNN	TOMEK-LINK	Euclidean	City-Block	Cosine Similarity
Train_Test_IoT_Fridge	91.10%	90.10%	92.50%	94.50%	93.10%	94.10%
Train_Test_Iot_garage_door	91.50%	92.10%	90.80%	92.30%	94.00%	95.00%
Train_Test_Iot_gps_tracker	92.94%	94.60%	93.70%	96.00%	94.10%	94.70%
Train_test_iot_modbus	88.53%	87.37%	90.00%	94.80%	95.00%	93.00%
<b>Train_test_iot_thermostat</b>	<b>89.40%</b>	<b>87.23%</b>	<b>82.30%</b>	<b>88.50%</b>	<b>88.34%</b>	<b>89.56%</b>
<b>Train_test_iot_weather</b>	<b>86.90%</b>	<b>84.70%</b>	<b>88.50%</b>	<b>83.70%</b>	<b>93.65%</b>	<b>91.40%</b>
Train_test_iot_motion_light	94.30%	92.75%	91.40%	94.70%	96.00%	96.61%
<b>Train_test_Network</b>	<b>89.80%</b>	<b>89.60%</b>	<b>85.90%</b>	<b>89.30%</b>	<b>90.60%</b>	<b>91.10%</b>
<b>Train_test_Windows7</b>	<b>82.30%</b>	<b>83.50%</b>	<b>87.00%</b>	<b>88.50%</b>	<b>89.00%</b>	<b>90.10%</b>
Train_test_Windows10	91.37%	93.90%	94.50%	95.80%	98.10%	96.61%
Final_10_Best_testing	89.90%	88.24%	87.50%	92.50%	93.80%	93.25%

The results are better for the following datasets namely, IoT\_Fridge, IoT\_garage\_door, gps\_tracker, IoT\_modbus, IoT\_Weather, IoT\_motion\_light, IoT\_Windows10, and Final\_10\_Best\_Testing of BoT-IoT dataset. The KNN classifier accomplishes minimum for unbalanced datasets because the majority voting inclines the decision of the classifier. This limitation of KNN is overcome in the implemented proposed work as shown in Table 10. The execution of the proposed scheme with NB and SVM is also illustrated in the results obtained in Tables 11 and 12. It is observed from the tables that the proposed model achieves good average accuracy in comparison to the Tomek-link technique, for seven datasets which have low imbalance ratio. In the case of naive Bayes, the approach aims to maintain the prior probability of classes and eliminating boundary samples rather than outliers. The dataset characteristics are represented by imbalance ratio (IR) which is as a statistical parameter. Both prior under-sampling and post oversampling cases are evaluated in context to the scenario and the statistics are shown in Table 7.

The proposed approach comes with an inbuilt under-sampling technique inclusive of three similarity measures to estimate similarity among the samples namely, Euclidean, city-block, and Cosine. It is observed that every dataset performance is influenced by IR. Besides, it is also observed that Euclidean distance produces reasonable results in a substantial number of datasets with all three classifiers. Therefore, Euclidean is considered as the superior similarity measure in the proposed approach. Bold highlighted values in Tables 10–12 show the high imbalance datasets resulting in low accuracy and the proposed approach producing better results for all other seven datasets.

## 6. Performance Analysis

The performance of the proposed model is determined based on the performance measures given in the next subsection.

### 6.1. Confusion Matrix

The actual and real classification performed by a learner is obtained in the confusion matrix. The various data entries present in a confusion matrix for a binary classifier are following:

- True Positive (TP)—represents the total ‘positive’ samples characterized as positive
- False Positive (FP)—represents the total ‘negative’ samples characterized as positive
- False Negative (FN)—represents the total ‘positive’ samples characterized as negative
- True Negative (TN)—represents the total ‘negative’ samples characterized as negative

The various performance measures for binary categorization retrieved from the confusion matrix values are shown in Table 13. The area-under-curve among two samples is determined by calculating an integral among two separate data samples. An AUC test can be confirmed according to the values given in Table 14.

**Table 13.** Performance measure assessment.

Assessment	Formulae
Precision (P)	$TP/(TP + FP)$
Recall (TPR/Sensitivity/R)	$TP/(TP + FN)$
Specificity	$TN/(TN + FP)$
F-measure	$2 \times P \times R/(P + R)$
AUC	$\frac{1}{2}((TP/(TP+FN)) + TN/(TN + FP))$

**Table 14.** Test validation using AUC score.

Range of AUC	Test Result
0.5–0.6	Fail
0.6–0.7	Poor
0.7–0.8	Fair
0.8–0.9	Good
0.9–1.0	Excellent

The advantage of the proposed scheme is shown for derived performance measures. The outliers, noisy and redundant sample of around 5% are removed from the majority group. Thus, only a minor amount of potential weak sample is removed from the dataset to create a balanced one. Tables 15 and 16 signify the performance measure assessment of edge layer of ToN\_IoT dataset and Final\_10\_Best\_Testing of BoT\_IoT dataset. Similarly, performance is calculated for remaining datasets also. However, the outcomes of other datasets are not represented because of space restrictions. The bold highlighted values of Tables 15 and 16 shows the paramount results.

**Table 15.** Performance metric values for edge layer of ToN-IoT dataset.

Type of Classifier	Distance Function	Precision	Recall	Specificity	F-Measure	AUC	
Classifier: KNN	RUS	-	92.36%	90.24%	91.36%	91.67%	0.9231
	CNN	-	92.36%	91.15%	91.29%	90.08%	0.9131
	Tomek-link	Euclidean	92.36%	90.24%	90.36%	92.67%	0.9231
	Proposed scheme	Euclidean	<b>93.00%</b>	92.91%	92.14%	<b>93.50%</b>	0.9459
		City-block	92.37%	<b>93.50%</b>	91.49%	92.87%	0.9377
		Cosine Similarity	91.56%	92.90%	<b>93.00%</b>	90.14%	<b>0.9495</b>

Table 15. Cont.

Type of Classifier	Distance Function	Precision	Recall	Specificity	F-Measure	AUC	
Classifier: SVM	RUS	-	91.59%	89.83%	89.12%	92.07%	0.9023
	CNN	-	92.45%	90.82%	93.31%	91.11%	0.9103
	Tomek-link	Euclidean	90.45%	90.82%	92.31%	93.11%	0.9204
	Proposed scheme	Euclidean	92.00%	<b>92.75%</b>	92.22%	91.48%	0.9075
		City-block Cosine Similarity	91.00%	92.25%	91.59%	90.99%	0.8907
		90.00%	91.73%	91.22%	89.48%	0.8857	
Classifier: Naive Bayes	RUS	-	89.45%	88.13%	88.45%	88.67%	0.8873
	CNN	-	89.27%	89.11%	89.01%	87.18%	0.8864
	Tomek-link	-	90.45%	90.00%	89.45%	89.67%	0.8973
	Proposed scheme	Euclidean	89.23%	<b>90.00%</b>	<b>90.59%</b>	<b>91.99%</b>	<b>0.9112</b>
		City-block Cosine Similarity	88.00%	89.00%	89.11%	90.48%	0.8934
		89.56%	89.36%	88.59%	90.99%	0.9056	

The bold values highlighted in Tables 15 and 16 represent the best results obtained for the respective undersampling technique integrated with the classifier.

Table 16. Performance metric values for Final\_10\_Best\_Testing of BoT-IoT dataset.

Type of Classifier	Distance Function	Precision	Recall	Specificity	F-Measure	AUC	
Classifier: KNN	RUS	-	95.61%	90.00%	94.00%	90.40%	0.9189
	CNN	-	90.25%	100.00%	93.33%	97.27%	0.9596
	Tomek-link	-	97.91%	96.76%	92.96%	97.94%	0.8987
	Proposed scheme	Euclidean	98.12%	95.00%	96.67%	<b>99.18%</b>	0.9424
		City-block Cosine Similarity	97.45%	<b>99.59%</b>	96.96%	91.82%	0.9088
		<b>98.79%</b>	98.60%	<b>97.78%</b>	98.00%	<b>0.9626</b>	
Classifier: SVM	RUS	-	97.37%	96.19%	93.14%	93.00%	0.9231
	CNN	-	95.54%	91.58%	94.55%	92.06%	0.9693
	Tomek-link	-	96.34%	95.32%	91.81%	97.55%	0.9525
	Proposed scheme	Euclidean	96.79%	<b>98.93%</b>	97.74%	<b>100.00%</b>	0.9754
		City-block Cosine Similarity	97.22%	97.04%	95.29%	98.47%	0.9683
		<b>98.05%</b>	98.33%	<b>99.71%</b>	98.34%	<b>0.9895</b>	
Classifier: Naive Bayes	RUS	-	85.85%	87.98%	92.52%	91.73%	0.9071
	CNN	-	88.89%	93.94%	91.33%	90.73%	0.9374
	Tomek-link	Euclidean	92.82%	87.84%	90.76%	90.78%	0.9104
	Proposed scheme	Euclidean	96.70%	95.86%	96.41%	92.13%	0.9163
		City-block Cosine Similarity	97.90%	<b>97.87%</b>	95.41%	92.69%	0.9214
		<b>98.77%</b>	94.85%	<b>97.96%</b>	<b>98.76%</b>	<b>0.9591</b>	

## 6.2. Intrusion Detection in Hadoop

Different sub packages of the org.apache are utilized for various functionality of execution and storage in the Hadoop environment. The various packages used in the Hadoop environment is executed and shown in Figure 6. It is inferred from Figure 6 that the



job subclass package present in the org.apache.hadoop is used to allow the user to configure the job, submit, control the execution and query the state. In general, the user creates the application, describes various aspects of the job and submits the job and monitors the progress. The JobSubmitter class can specify access control lists for viewing or modifying a job via the configuration properties mapreduce.job.acl-view-job and mapreduce.job.acl-modify-job respectively. By default, nobody is given access in these properties. In addition, MapReduceLauncher is a main class that launches pig for Map Reduce. The other class namely RMPProxy which is a subclass of org.apache.hadoop.yarn.client is utilized for connecting to the Resource Manager and used by the NodeManagers only. This class creates a proxy for the specified protocol. An Inter-process communication is established between client and server by utilizing the following class of org.apache.hadoop.ipc.client. The ClientServiceDelegate class is responsible for monitoring the application state of the process running in various namenode clusters. It returns the FinalApplicationStatus as succeeded and redirects to job history.

```

2021-05-24 14:39:10,734 [JobControl] INFO org.apache.hadoop.mapreduce.JobSubmitter - number of splits:1
2021-05-24 14:39:10,994 [JobControl] INFO org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2021-05-24 14:39:11,744 [Thread-22] INFO org.apache.hadoop.hdfs.protocol.datatransfer.sasl.SaslDataTransferClient - SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-24 14:39:11,788 [JobControl] INFO org.apache.hadoop.mapreduce.JobSubmitter - Submitting tokens for job: job_1621846769726_0001
2021-05-24 14:39:11,788 [JobControl] INFO org.apache.hadoop.mapreduce.JobSubmitter - Executing with tokens: []
2021-05-24 14:39:12,415 [JobControl] INFO org.apache.hadoop.mapred.YARNRunner - Job jar is not present. Not adding any jar to the list of resources.
2021-05-24 14:39:12,605 [JobControl] INFO org.apache.hadoop.conf.Configuration - resource-types.xml not found
2021-05-24 14:39:12,605 [JobControl] INFO org.apache.hadoop.yarn.util.resource.ResourceUtils - Unable to find 'resource-types.xml'.
2021-05-24 14:39:16,580 [JobControl] INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl - Submitted application application_1621846769726_0001
2021-05-24 14:39:16,790 [JobControl] INFO org.apache.hadoop.mapreduce.Job - The url to track the job: http://vedant:8088/proxy/application_1621846769726_0001/
2021-05-24 14:39:16,790 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - HadoopJobId: job_1621846769726_0001
2021-05-24 14:39:16,790 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - Processing aliases data
2021-05-24 14:42:48,734 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - detailed locations: M: data[1,7],data[-1,-1] C: R:
2021-05-24 14:39:16,810 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - 0% complete
2021-05-24 14:39:16,811 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - Running jobs are [job_1621846769726_0001]
2021-05-24 14:41:33,412 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - 7% complete
2021-05-24 14:41:33,412 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - Running jobs are [job_1621846769726_0001]
2021-05-24 14:42:48,734 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce_layer.MapReduceLauncher - 20% complete
2021-05-24 14:42:48,759 [main] INFO org.apache.hadoop.yarn.client.RMPProxy - Connecting to ResourceManager at /0.0.0.0:8032
2021-05-24 14:42:48,775 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-05-24 14:42:50,178 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:51,181 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 1 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:52,183 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 2 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:53,184 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 3 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:54,185 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 4 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:55,186 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:56,188 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:57,189 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:58,190 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:59,193 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:42:59,341 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-05-24 14:43:00,348 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:43:01,375 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 1 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:43:02,377 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 2 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2021-05-24 14:43:03,382 [main] INFO org.apache.hadoop.tpc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 3 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

```

Figure 6. HDFS dataset partitioning—initial level.

Figure 6 shows the HDFS partitioning of the dataset files at the initial level which will be executed in three different slave nodes. Various commands are executed such as \$start-dfs.sh for invoking namenode, datanode, and secondary namenode. The node manager and resource manager are initialized with the command \$start-yarn.sh. This is required for distributing the resources and data in HDFS. Thus, from the figure, it is evident that three files are executed in map reduce and shows the percentage of complete as 0%, 7%, and 20% in yellow color in the figure. Figure 7 shows the shows the HDFS partitioning at final level where in the computation is 100% complete after partitioning and executing the computation in a distributed manner. Our input data is the rawdataset.csv which is in HDFS. After executing the store statement, the following output will be obtained. A directory is created with the specified name and the data will be stored in it. The statistics inferred from Figure 7 is there are five Job Ids which have various feature outputs to perform mapping, sampling, ordering, querying, and mapping to the final job status. The complete BoT-IoT dataset is executed in 24 min 39 s in comparison to 6 h in a normal stand-alone environment.



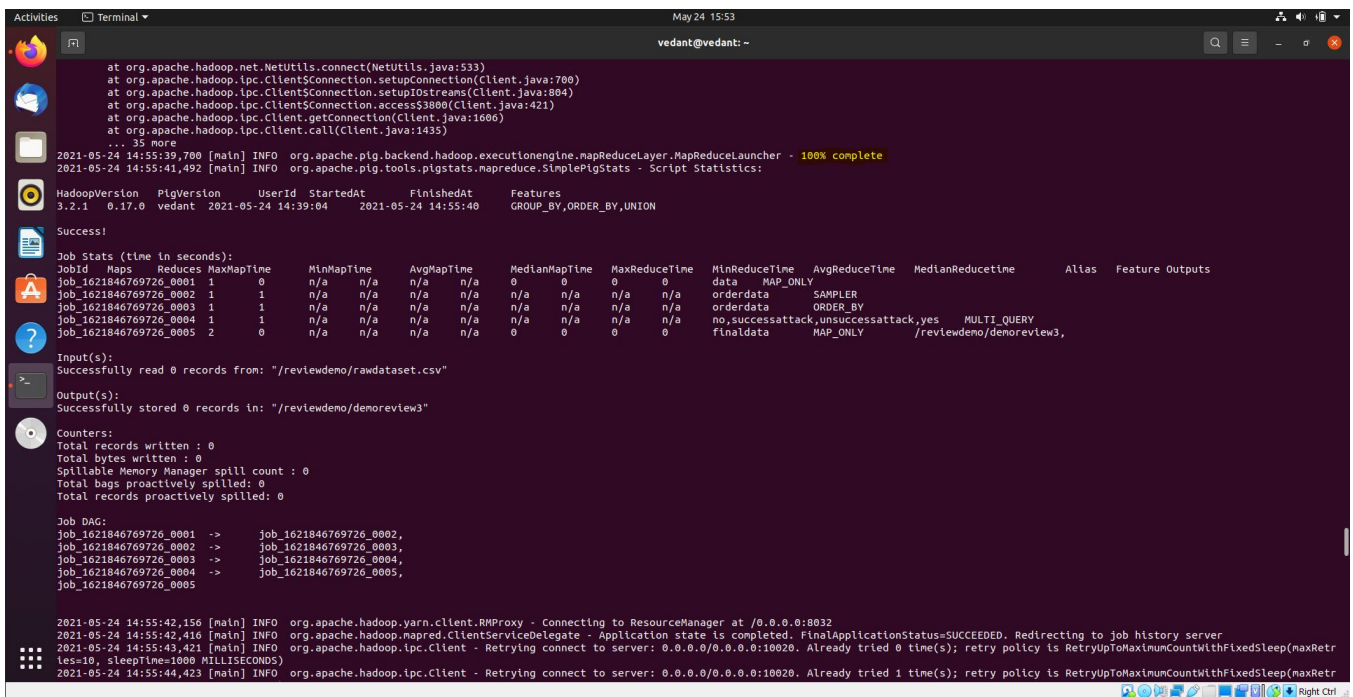


Figure 7. HDFS execution-completion status.

Table 17 shows the computational time of the benchmark datasets in Hadoop environment. The datasets having less samples do not contribute much in the computational time in a Big Data platform. However, samples of Train\_test\_network containing 461,043 records and final\_10\_best\_testing samples containing 733,705 records have been executed in less than 30 min whereas in standalone environment, execution time was more than 3 to 4 h.

### 6.3. Discussion

The proposed model implements a Hadoop based framework, effective pre-processing using under-sampling technique to remove outlier, noisy and redundant samples from the majority group. The threshold, for identifying the outlier is considered between 10 and 15. The sensitivity analysis result indicates the impact of the parameter, ‘t’ on the effectiveness of the model and the similarity measure choice is critical for the model. The deployment of three classifiers such as KNN, naïve Bayes, and SVM is used to test the superiority of the baseline schemes with different parameter settings as shown in Table 3. A hyper parameter tuning using AiNet also improves the cross-validation accuracy before prediction on BoT-IoT and ToN-IoT datasets.

Table 17. Computational time of the benchmark datasets in Hadoop environment.

Datasets	No. of Samples	Computational Time
Train_Test_IoT_Fridge	59,944	
Train_Test_Iot_garage_door	59,587	
Train_Test_Iot_gps_tracker	58,960	
Train_test_iot_modbus	51,106	1 min 15 s
Train_test_iot_thermostat	52,775	
Train_test_iot_weather	59,260	
Train_test_iot_motion_light	59,488	
<b>Train_test_Network</b>	<b>461,043</b>	<b>12 min 14 s</b>
Train_test_Windows7	15,980	
Train_test_Windows10	21,104	40 s
Final_10_Best_testing	733,705	24 min 39 s

The various performance measure results of the proposed approach are shown in Section 6.1 and baseline techniques using three classifiers for all dataset samples of BoT-IoT and ToN-IoT. The classification accuracy increases by detecting outliers for the minority samples, which results in increased recall and precision in maximum cases for the proposed approach. Concerning specificity, the proposed approach outdoes the baseline techniques for both datasets. Table 7 presents a statistics of IR on datasets which demonstrates the efficiency of the proposed approach as it focused on a balanced IoT traffic.

Figure 8 demonstrates the performance of various machine learning approaches [62] on the Bot-IoT dataset. It is evident from the results that the integration of the two phases, namely, modified Tomek link under-sampling and machine learning parameter tuning provides the best performance with an accuracy of 99% for BoT-IoT dataset. The proposed model is also compared with a recent literature [17] which has been verified on BoT-IoT dataset. The proposed model has an overall 19% improvement in accuracy in comparison with the state of the art techniques. Figure 9 signifies the performance comparison of different machine learning approaches on the edge layer of ToN-IoT dataset with the proposed approach [57,60–64]. The proposed model integrates modified Tomek link under-sampling and hyper parameter tuning using AiNet approach for the supervised machine learning classifiers which enhances the overall performance on both the datasets. Conflicting with prevailing works, our analysis shows a broad assessment for the real attack and simulated attack data which were generated by simulating a realistic network at the University of New South Wales where real attacks on IoT networks were recorded.

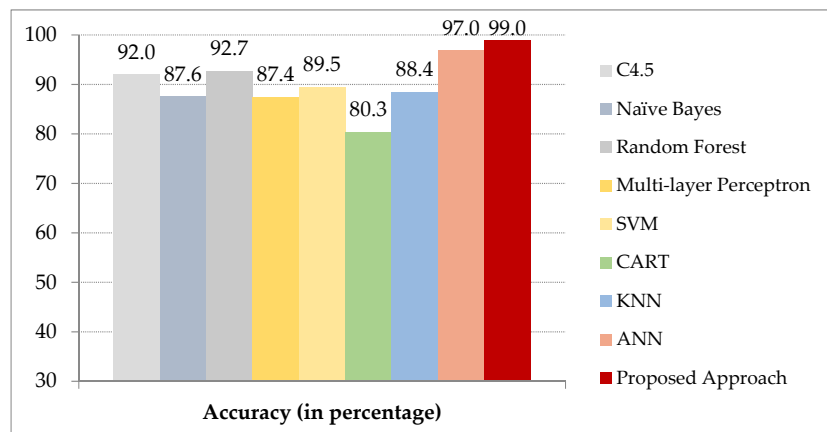


Figure 8. Comparison of the proposed scheme with existing approaches on BoT-IoT dataset.

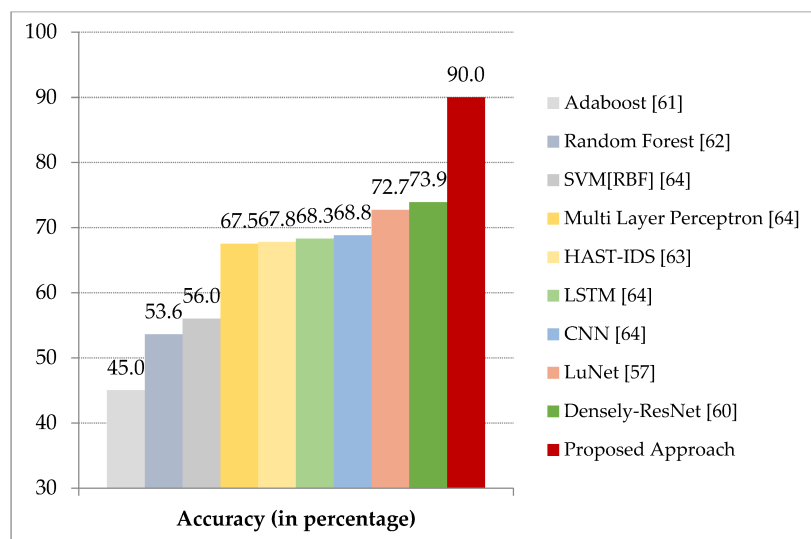


Figure 9. Comparison of the proposed scheme with existing approaches on ToN-IoT dataset.

## 7. Conclusions

Autonomous malware attacks are increasing day by day in IoT. This malware affects cross-platform applications and spread between platforms thereby resulting in a devastating effect on connected devices. Numerous researchers have proposed different machine learning techniques for developing a secure framework for IoT. The major contribution of this paper is the usage of a Hadoop based framework integrating an enhanced Tomek-link under-sampling technique for preprocessing and classification using three different supervised classifiers namely NB, SVM, and KNN. The huge volume of IoT traffic datasets increases the computational time in a stand-alone environment. Hence, a HDFS is deployed and all the datasets are loaded inside the Hadoop repository for faster computation. The datasets are hugely imbalanced and hence there is a need to perform under-sampling of majority samples during preprocessing and the classification is performed by machine learning models. A hyper-parameter tuning is performed using Opt-AiNet for tuning the parameters of NB, SVM, and KNN for improved cross-validation accuracy. The performance of BoT-IoT and ToN-IoT datasets on the proposed approach was experimented with various performance metrics like precision, recall, specificity, f-measure, and AUC. The results are compared with traditional approaches to illustrate the supremacy. This is because big data platform is deployed to minimize the computational time and an increase in accuracy is obtained due to the effective data pre-processing and model tuning techniques. There has been a significant increase in average accuracy in BoT-IoT and ToN-IoT datasets in comparison with the contemporary IDS techniques for IoT using machine learning approaches. The computational time is also significantly reduced for the datasets in an HDFS environment.

**Author Contributions:** Conceptualization, methodology, software, and validation, I.S.T., V.M. and S.R.; Formal analysis, investigation, data curation, and structure of the paper, K.S. and S.-S.Y.; Writing—original draft preparation, I.S.T., V.M., S.R. and K.S.; Writing—review and editing, and supervision, S.-S.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** There is no external funding for this research work.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

CFS	Correlation based Feature Selection
CST-GR	Correlated-set thresholding on gain-ratio
DNS	Domain Name System
ELM	Extreme Learning Machine
FAR	False Alarm Rate
IDS	Intrusion Detection System
IoMT	Internet of Medical Things
IoT	Internet of Things
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LMT	Logistic Model Tree
NB	Naïve Bayes
PCA-GWO	Principal Component Analysis-Grey Wolf Optimization
RF	Random Forest
SVM	Support Vector Machine
VFDT	Very Fast Decision Tree
VoIP	Voice Over Internet Protocol
WSN	Wireless Sensor Network

## Appendix A

Pseudocode of machine learning classifiers of the proposed approach:

**Algorithm A1. Proposed Approach****Given:** Classifier M1(SVM), M2(NB), M3(KNN)<sub>i</sub>**Input:** Optimal featured dataset D**Output:** Class label**Step 1:** Set all weights in  $W_i = 1/n$ , where  $n$  represents the total sample count.**Step 2:** For every sample data  $d_i$ 

- Fit the SVM classifier to  $(X_t, Y_t)$  using weights  $W_i$
- For each class label  $k = 1 \dots K$  obtain the hypothesis
- $x_s \leftarrow \operatorname{argmin}(\lambda |f(x_j)| + (1 - \lambda) \max \frac{|k(x_i, x_j)|}{\sqrt{k(x_i, x_j)k(x_j, x_i)}}$
- $D \leftarrow D \cup \{x_s\}$
- Label ( $D$ )
- $L \leftarrow L \cup \{S\}$

**Step 3:** For every sample data  $d_i$ 

- Fit the NB classifier to  $(X_t, Y_t)$ , using weights  $W_i$
- For each class label  $k = 1 \dots K$  obtain the prior and posterior probabilities
- Calculate the posterior probability
- $P(x|c_j) = \sum_{i=1}^n P(X_i = x_i|c_j)$
- And the class label is assigned as follows:  

$$L^* = h_{NB}(x) = \operatorname{argmax} P(c_j) \sum_{i=1}^n P(X_i = x_i|c_j)$$

**Step 4:** if  $Label(M1) \neq Label(M2)$ **Step 5:** Fit the KNN to  $(X_t, Y_t)$  using weights  $W_i$ 

- Choose the value of  $K$
- Calculate the Euclidean distance for all the training samples.
- Store the Euclidean distances in an array to sort it.
- Determine the first  $k$  points.
- Assign a class to the test sample by estimating the majority of classes identified in the chosen points
- Perform hyper parameter tuning of SVM, NB and KNN using Artificial Immune Network

**Step 6:** Predict the test data label from steps (2), (3) and (4) using a majority voting**Step 7:** Determine the performance metrics.**References**

1. Demiris, G.; Hensel, B.K. Technologies for an aging society: A systematic review of “smart home” applications. *Yearb. Med. Inform.* **2008**, *3*, 33–40.
2. Denning, T.; Tadayoshi, K. Empowering consumer electronic security and privacy choices: Navigating the modern home. In Proceedings of the Symposium on Usable Privacy and Security (SOUPS), Newcastle, UK, 24–26 July 2013.
3. Amini, P.; Araghizadeh, M.A.; Azmi, R. A survey on Botnet: Classification, detection and defense. In Proceedings of the 2015 International Electronics Symposium (IES), Surabaya, Indonesia, 29–30 September 2015; pp. 233–238.
4. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2009**, *100*, 779–796. [CrossRef]
5. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
6. Jain, M.; Kaur, G. Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data. *Clust. Comput.* **2021**, *2021*, 1–16.
7. Devi, D.; Purkayastha, B. Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance. *Pattern Recognit. Lett.* **2017**, *93*, 3–12. [CrossRef]
8. Hilton, S. Dyn Analysis Summary of Friday October 21 Attack. Dyn, 2016. Dyn Blog. Available online: <https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack> (accessed on 2 June 2021).
9. Xiao, L.; Xiaoyue, W.; Zhu, H. PHY-layer authentication with multiple landmarks with reduced overhead. *IEEE Trans. Wirel. Commun.* **2017**, *17*, 1676–1687. [CrossRef]
10. Cong, S.; Liu, J.; Liu, H.; Chen, Y. Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT. In Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Chennai, India, 10–14 July 2017; pp. 1–10.

11. Jerkins, J.A. Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code. In Proceedings of the 2017 IEEE 7th annual computing and communication workshop and conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017; pp. 1–5.
12. Badve, O.; Gupta, B.B.; Gupta, S. Reviewing the security features in contemporary security policies and models for multiple platforms. In *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*; IGI Global: Hershey, PA, USA, 2016; pp. 479–504.
13. Starr, M. Fridge Caught Sending Spam Emails in Botnet Attack. 2014. Available online: <https://www.cnet.com/home/kitchen-and-household/fridge-caught-sending-spam-emails-in-botnet-attack/> (accessed on 19 January 2014).
14. Kumar, N.; Naveen, C. Collaborative trust aware intelligent intrusion detection in VANETs. *Comput. Electr. Eng.* **2014**, *40*, 1981–1996. [[CrossRef](#)]
15. Bailey, M.; Evan, C.; Farnam, J.; Yunjing, X.; Manish, K. A survey of botnet technology and defenses. In Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security, Institute of Electrical and Electronics Engineers (IEEE), Washington, DC, USA, 3–4 March 2009; pp. 299–304.
16. Tahaei, H.; Afifi, F.; Asemi, A.; Zaki, F.; Anuar, N.B. The rise of traffic classification in IoT networks: A survey. *J. Netw. Comput. Appl.* **2020**, *154*, 25–38. [[CrossRef](#)]
17. Churcher, A.; Ullah, R.; Ahmad, J.; Rehman, S.U.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W. An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. *Sensors* **2021**, *21*, 446. [[CrossRef](#)]
18. Hussain, F.; Hussain, R.; Hassan, S.A.; Hossain, E. Machine Learning in IoT Security: Current Solutions and Future Challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1686–1721. [[CrossRef](#)]
19. Alsamiri, J.; AlSubhi, K. Internet of Things Cyber Attacks Detection using Machine Learning. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *2019*, 10. [[CrossRef](#)]
20. Wurm, J.; Hoang, K.; Arias, O.; Sadeghi, A.; Jin, Y. Security analysis on consumer and industrial IoT devices. In Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macau, China, 25–28 January 2016; pp. 519–524.
21. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks. *Electronics* **2019**, *8*, 1210. [[CrossRef](#)]
22. Murad, G.; Badarneh, A.; Quscf, A.; Almasalha, F. Software Testing Techniques in IoT. In Proceedings of the 8th International Conference on Computer Science and Information Technology (CSIT), Amman, Jordan, 11–12 July 2018; pp. 17–21.
23. Siboni, S.; Shabtai, A.; Elovici, Y. Leaking data from enterprise networks using a compromised smartwatch device. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, Pau, France, 9–13 April 2018; pp. 741–750.
24. Moody, M.; Hunter, A. Exploiting known vulnerabilities of a smart thermostat. In Proceedings of the 14th IEEE Annual Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 12–14 December 2016; pp. 50–53.
25. Ronen, E.; Shamir, A. Ronen, Eyal; Adi Shamir. Extended functionality attacks on IoT devices: The case of smart lights. In Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrücken, Germany, 21–24 March 2016; pp. 3–12.
26. Fernandez, E.; Pelaez, J.; Larrondo-Petrie, M. Attack patterns: A new forensic and design tool. In Proceedings of the IFIP International Conference on Digital Forensics, Orlando, FL, USA, 28–31 January 2021; Springer: New York, NY, USA, 2007; pp. 345–357.
27. Alghamdi, T.A.; Lasebae, A.; Aiash, M. Security analysis of the constrained application protocol in the Internet of Things. In Proceedings of the Second International Conference on Future Generation Communication Technology, London, UK, 12–14 November 2013; pp. 163–168.
28. Arias, O.; Wurm, J.; Hoang, K.; Jin, Y. Privacy and Security in Internet of Things and Wearable Devices. *IEEE Trans. Multi-Scale Comput. Syst.* **2015**, *1*, 99–109. [[CrossRef](#)]
29. Bachy, Y.; Basse, F.; Nicomette, V.; Alata, E.; Kaaniche, M.; Courcege, J.-C.; Lukjanenko, P. Smart-TV Security Analysis: Practical Experiments. In Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil, 22–25 June 2015; pp. 497–504.
30. Sivaraman, V.; Chan, D.; Earl, D.; Boreli, R. Smart-Phones Attacking Smart-Homes. In Proceedings of the 9th ACM Conference on Creativity & Cognition, Sydney, Australia, 17–20 June 2013; pp. 195–200.
31. Ling, Z.; Liu, K.; Xu, Y.; Jin, Y.; Fu, X. An end-to-end view of iot security and privacy. In Proceedings of the GLOBECOM 2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–7.
32. Ling, Z.; Luo, J.; Xu, Y.; Gao, C.; Wu, K.; Fu, X. Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System. *IEEE Internet Things J.* **2017**, *4*, 1899–1909. [[CrossRef](#)]
33. Seralathan, Y.; Oh, T.T.; Jadhav, S.; Myers, J.; Jeong, J.P.; Kim, Y.H.; Kim, J.N. IoT security vulnerability: A case study of a Web camera. In Proceedings of the 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si, Korea, 11–14 February 2018; pp. 172–177.
34. Xu, H.; Xu, F.; Chen, B. Internet protocol cameras with no password protection: An empirical investigation. In Proceedings of the International Conference on Passive and Active Network Measurement, Berlin, Germany, 27–28 March 2018; pp. 47–59.
35. Classen, J.; Wegemer, D.; Patras, P.; Spink, T.; Hollick, M. Anatomy of a vulnerable fitness tracking system: Dissecting the fitbit cloud, app, and firmware. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2018**, *2*, 1–24. [[CrossRef](#)]



36. Willingham, T.; Henderson, C.; Kiel, B.; Haque, S.; Atkison, T. Testing vulnerabilities in bluetooth low energy. In Proceedings of the ACMSE 2018 Conference, Richmond, KY, USA, 29–31 March 2018; pp. 1–7.
37. Sachidananda, V.; Siboni, S.; Shabtai, A.; Toh, J.; Bhairav, S.; Elovici, Y. Let the cat out of the bag: A holistic approach towards security analysis of the internet of things. In Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, Abu Dhabi, United Arab Emirates, 2 April 2017; pp. 3–10.
38. Bagaa, M.; Taleb, T.; Bernabe, J.B.; Skarmeta, A. A Machine Learning Security Framework for Iot Systems. *IEEE Access* **2020**, *8*, 114066–114077. [[CrossRef](#)]
39. Rahman, A.; Asyharina, A.T.; Leong, L.S.; Satrya, G.B.; Tao, M.H.; Zolklipli, M.F. Scalable Machine Learning-Based Intrusion Detection System for IoT-Enabled Smart Cities. *Sustain. Cities Soc.* **2020**, *61*, 10–23. [[CrossRef](#)]
40. Arshad, J.; Azad, M.A.; Abdeltaif, M.M.; Salah, K. An intrusion detection framework for energy constrained IoT devices. *Mech. Syst. Signal Process.* **2020**, *136*, 106436. [[CrossRef](#)]
41. Al-Hadhrami, Y.; Hussain, F.K. Real time dataset generation framework for intrusion detection systems in IoT. *Future Gener. Comput. Syst.* **2020**, *108*, 414–423. [[CrossRef](#)]
42. Abdollahi, A.; Fathi, M. An Intrusion Detection System on Ping of Death Attacks in IoT Networks. *Wirel. Pers. Commun.* **2020**, *112*, 2057–2070. [[CrossRef](#)]
43. Zheng, D.; Hong, Z.; Wang, N.; Chen, P. An Improved LDA-Based ELM Classification for Intrusion Detection Algorithm in IoT Application. *Sensors* **2020**, *20*, 1706. [[CrossRef](#)]
44. Eskandari, M.; Janjua, Z.H.; Vecchio, M.; Antonelli, F. Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices. *IEEE Internet Things J.* **2020**, *7*, 6882–6897. [[CrossRef](#)]
45. Anthi, E.; Williams, L.; Slowinska, M.; Theodorakopoulos, G.; Burnap, P. A Supervised Intrusion Detection System for Smart Home IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 9042–9053. [[CrossRef](#)]
46. Cervantes, C.; Poplade, D.; Nogueira, M.; Santos, A. Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 606–611.
47. Anthi, E.; Williams, L.; Burnap, P. Pulse: An adaptive intrusion detection for the internet of things. *Living Internet Things Cybersecur. IoT* **2018**, *2018*, 35–40.
48. Nobakht, M.; Sivaraman, V.; Boreli, R. A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. In Proceedings of the 11th International conference on availability, reliability and security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 147–156.
49. Fu, Y.; Yan, Z.; Cao, J.; Koné, O.; Cao, X. An Automata Based Intrusion Detection Method for Internet of Things. *Mob. Inf. Syst.* **2017**, *2017*, 1–13. [[CrossRef](#)]
50. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features. *Electronics* **2020**, *9*, 144. [[CrossRef](#)]
51. Azab, A.; Alazab, M.; Aiash, M. Machine Learning Based Botnet Identification Traffic. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 1788–1794.
52. Kumar, N.; Singh, J.P.; Bali, R.S.; Misra, S.; Ullah, S. An intelligent clustering scheme for distributed intrusion detection in vehicular cloud computing. *Clust. Comput.* **2015**, *18*, 1263–1283. [[CrossRef](#)]
53. Xiao, L.; Li, Y.; Han, G.; Liu, G.; Zhuang, W. PHY-layer spoofing detection with reinforcement learning in wireless networks. *IEEE Trans. Veh. Technol.* **2016**, *16*, 10037–10047. [[CrossRef](#)]
54. Xiao, L.; Yan, Q.; Lou, W.; Chen, G.; Hou, Y.T. Proximity-based security techniques for mobile users in wireless networks. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 2089–2100. [[CrossRef](#)]
55. Xiao, L.; Xie, C.; Chen, T.; Dai, H.; Poor, H.V. A mobile offloading game against smart attacks. *IEEE Access* **2016**, *4*, 2281–2291. [[CrossRef](#)]
56. Oulhiq, R.; Ibntahir, S.; Sebgui, M.; Guennoun, Z. A fingerprint recognition framework using Artificial Neural Network. In Proceedings of the 10th International Conference on Intelligent Systems: Theories and Applications (SITA), Rabat, Morocco, 20–21 October 2015; pp. 1–6.
57. Ly, K.; Jin, Y. Security studies on wearable fitness trackers. In Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Orlando, FL, USA, 16–20 August 2016.
58. Ali, N.; Neagu, D.; Trundle, P. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Appl. Sci.* **2019**, *1*, 1559. [[CrossRef](#)]
59. Li, C.; Wang, G. A light-weight commodity integrity detection algorithm based on Chinese remainder theorem. In Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012; pp. 1018–1023.
60. Khan, F.; Kanwal, S.; Alamri, S.; Mumtaz, B. Hyper-Parameter Optimization of Classifiers, Using an Artificial Immune Network and Its Application to Software Bug Prediction. *IEEE Access* **2020**, *8*, 20954–20964. [[CrossRef](#)]
61. UNSW Canberra Website. Available online: [https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot\\_ietf.php](https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_ietf.php) (accessed on 2 June 2021).

- 
62. Foley, J.; Moradpoor, N.; Ochenyi, H. Employing a Machine Learning Approach to Detect Combined Internet of Things Attacks against Two Objective Functions Using a Novel Dataset. *Secur. Commun. Netw.* **2020**, *2020*, 1–17. [[CrossRef](#)]
  63. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **2017**, *6*, 1792–1806. [[CrossRef](#)]
  64. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON\_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* **2020**, *8*, 165130–165150. [[CrossRef](#)]