International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I9P213

Volume 71 Issue 9, 138-156, September 2023

Original Article

Ryu Controller-Based Attack Detection and Mitigation Method in Software Defined Internet of Things

Pinkey Chauhan[1], Mithilesh Atulkar[2]

[1,2]*Department of Computer Applications, NIT Raipur, India*

[1]*Corresponding Author : pchauhan.phd2018.mca@nitrr.ac.in*

Received: 29 June 2023 Revised: 28 July 2023 Accepted: 25 August 2023 Published: 03 September 2023

Abstract *- The innovative areas of software-defined networks and the Internet of Things are currently receiving significant attention in the IT industry and academic circles. As a result of their popularity, they have become a target for numerous attacks in the realm of SD-IoT (Software Defined Internet of Things). The attackers may aim to either pilfer or obstruct users' data, in addition to depleting network resources through futility, thereby frustrating legitimate user demands. The category of attacks includes a form known as Distributed Denial of Service (DDoS). In this work, a centralized attack detection and mitigation approach has been proposed. For getting the most efficient attack detection and prevention method, a number of classifiers, namely Random Forest (RF), XGB, Light Gradient Boosting Machine (LGBM), ET, GB, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), NB, SVM(Linear), LR, and SVM(Poly) have been trained and tested on two controller-based datasets. Their performance has been evaluated under precision, F1, Cohen's Kappa Coefficient (CKC), recall, accuracy, False Alarm Rate (FAR), Testing Time, and AUC value. In both datasets, it is discovered that LGBM outperforms all other classifiers, but here, the performance of LGBM on the second dataset is better than that of the first dataset, so finally, LGBM trained with the second dataset is deployed in the controller of SDN where it detects and mitigates the attack from the live traffic in SD-IoT.*

Keywords *- Distributed Denial of Service Attack, Ryu, Mininet, Software-defined Internet of Things, hping3, D-ITG.*

1. Introduction

The use of IoT is becoming more widespread day by day due to its usefulness in different areas of IT. Some major fields where it is being used are healthcare, agriculture, disaster, economics, meteorology, etc. [1, 2].

SDN has several advantages over conventional network architecture due to separating the control plane and data plane. Virtualization, big data, scalability, WAN, programmability, security, cloud and monitoring are some fields where SDN is gaining popularity [3, 4]. SD-IoT is the field where IoT has incorporated the advantages of SDN and is hence being used in many areas as it has the advantages of both fields. Picture 1 depicts the SD-IoT's working diagram, which divides the network into three layers: the infrastructure, control, and application layers [5]. In this case, the SDN's essential component is the controller, which regulates all network activity. The data plane devices get all the forwarding rules, and these devices operate according to the supplied rules. These sent rules are stored in the form of flow entries in flow tables. There might be more than one flow table in a switch, which might contain many flow entries. By using this approach, a centralized working environment is created in SDN, which lets the controller manage all the activities happening in the network [4, 6–8]. Due to this feature, the controller of SDN is called the brain of the network [9, 10].

To depict the SD-IoT architecture, two groups of devices in the data plane are presented in fig. 1. In the first group, a gateway has been used to connect the IoT devices to the data plane devices. The user devices in the second group are directly linked to the data plane devices [11]. The device to whom the IoT gateway is connected determines how to deal with the packets coming from the IoT devices. Whenever a new request from the gateway comes to the data plane device, it sends the packet_in request to the controller. The controller replies with a packet_out that represents the response. This response is stored in the form of a flow entry in the flow table of the device. For this, the OpenFlow protocol is stored to communicate between the data plane and the control plane.

For creating hindrances in the smooth functioning of SD-IoT, some attacks are generated from the data plane devices of SD-IoT, where IoT devices are also part of this. By sending floods of requests, attackers overwhelm the controller to respond back, and sometimes, they fill the spaces available in the data plane devices of the SD-IoT.

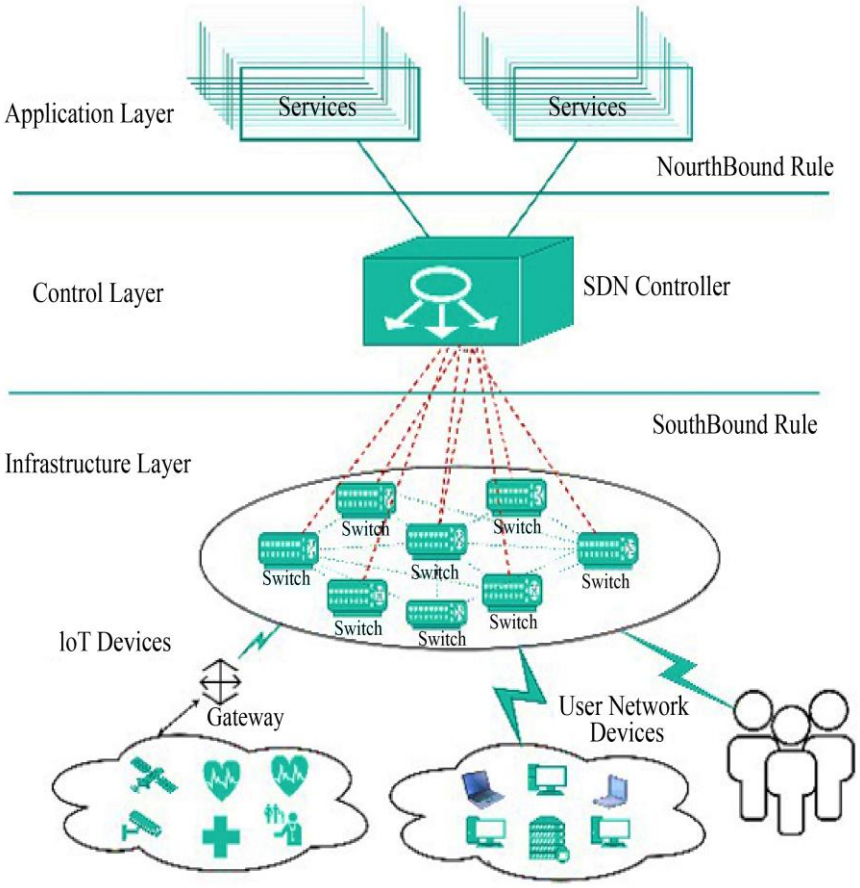


Fig. 1 SDN Based IoT

It creates the need to develop an IDS for SD-IoT, which can efficiently detect and mitigate the attack. In the SD-IoT context, there are two ways to find and stop attacks: first, in the control plane, and second, in the data plane. Additionally, two different approaches, known as Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS), can be used to find and stop attacks [12]. In the case of AIDS, the model is trained using typical traffic behaviour. Any divergence from this flow is seen as an attack. In the event of a SIDS, the database contains the attack's signature. Whenever a request comes to it, it matches the database, and if any match is found, the traffic is treated as attack traffic.

In this work, a centralized controller-based AIDS has been proposed, which can centrally detect and mitigate the attack from IoT devices connected to the data plane device. For this purpose, two controller-based datasets have been used to find the best-performing model. The first dataset, which is being named dataset(A), was created by the author of this work, and the second dataset, named dataset(B), was created by [13]. The details about these datasets are given in section 6.1. A few classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been trained and tested on these datasets and the best-performing classifier is selected for the deployment in the SD-IoT controller. The names of the metrics under which performance is evaluated are precision, F1, CKC, recall, accuracy, FAR, Prediction Time, and AUC value. Since LGBM with the dataset (B) is found to perform best among all classifiers, it is installed in the SD-IoT controller, where it can identify attacks. Further, it is deployed in such a way that it can also mitigate the attack by stopping the attacking source.

The work is structured as follows: Related work is shown in Section 2, motivation is discussed in Section 3, methodology is discussed in Section 4, implementation environment is discussed in Section 5, experiments and discussion are presented in Section 6, the work's contribution is presented in Section 7, and conclusions and future work are discussed in Section 8.

2. Related Work

To detect and prevent DDoS attacks on IoT using SDN, [14-16] designed a semi-supervised deep Extreme Learning Machine-based approach. Details and the names of the features they are using in their work have not been mentioned by them. However, they are claiming to detect the attack in a

distributed manner. They used their own internal dataset in addition to publicly accessible UNB-ISCX datasets to assess performance. As our study focuses on feature extraction, dataset generation, and classifier training, a close look at the work of the authors of [14] has been taken. The researchers stated that they extracted 155 features using the SPAN component of the Cisco Nexus Switch, although they did not describe how they did it. The data plane is the sole place where the aforementioned features can be extracted because they are primarily targeted at data packets. Furthermore, they have exclusively used UDP packets to demonstrate their research. They were able to achieve a precision of 97.2 percent, an accuracy of 97.9 percent, an F1 of 97.2 percent, and a recall of 97.6 percent.

Despite making the claim that the controller should record all traffic, [17] constructed an SDN using Mininet and Floodlight and applied a dataset made for the data plane for the purpose of spotting attacks. The datasets used included CTU-10, CTU-11, ISCX-IDS- 2012, and ISCX-SlowDDoS-2016. They claim to detect the attack in the controller but use the dataset made for data plane devices. This is the main issue with their approach. They are limiting their productivity since they are not putting in the necessary work in the controller to detect the threats. A trigger-based approach has been given by [18], where on detecting the attack in the data plane, one alert is sent to the controller to check whether this alert is right or wrong. For this, they are using K-Means and KNN. On receiving the alert, the controller extracts five vector tuples from the traffic and predicts. The right countermeasures can be implemented as soon as malicious traffic is found. They are utilising Mininet and ONOS, respectively, for their work. The XGBoost classifier was suggested by [19-22] to recognise the attack. With the help of the Mininet and POX, they developed the network. The traffic was gathered using TCPdump, and the Hyenae tool created the attack. [23] used a deep learning-based sparse autoencoder for their attack prevention system. For traffic collecting and extraction, the TCFI module of the controller was used. A dual-tiered security architecture was suggested by [24] in their study. In their work, they found that DNN is performing better than SVM. All the aforementioned tasks were performed using the Mininet emulator and the Ryu controller.

The TCP SYN attack, a type of DDoS attack, was suggested to be early detected and countered by [25] in their article. The entropy method was used to determine the flow data's unpredictable nature. They have integrated their logic into a floodlight controller and a network emulator called Mininet for various situations.

A lot of effort has been put on the SDN controller's attack defence. The approach proposed by [3] focuses on detecting DDoS attacks in SDN using the pox controller. They produced a dataset from which they extracted 12 UDP, TCP, and ICMP traffic characteristics under normal and attack conditions. They compared the performance measuring metrics of the NB, SVM, KNN, and ANN classifiers after training them on both all features and the six chosen features. They discovered that the KNN classifier outperformed the others with an accuracy of 98.3 percent, recall of 97.72 percent, precision of 97.70 percent, and F1-Value of 97.70 percent with the six chosen features. Finally, they asserted that the outcomes after feature selection are superior to the ones before.

Were able to determine the type of attack and its scope using entropy (Giotis 2014) [26]. They utilised the NOX controller for traffic sampling through SDN and sFlow. The NOX controller's three components are data collecting, periodic entropy-based calculations, and flow table modification, which is used to manage the various tasks of attack detection and mitigation.

A method based on using several controllers was suggested by [27] to detect an attack in the SDN control plane. A data module and a control module are both present. Detecting unusual network traffic is the data plane module's job. This module enhances the performance of the Back Propagation Neural Network by employing four vector tuples taken from incoming input. The forward controller must be remapped, and this module must provide access control. Usually, only the master controller is in operation, while the slave controller is inactive. While the slave controller is activated when an attack is detected, the master controller communicates the ACL to the data plane routers. They used a Mininet emulator and a Ryu controller to demonstrate their abilities.

The control plane and data plane engaged in interaction in [28]. This method aims to reduce the burden on the controller and southbound traffic by leveraging the data plane switches' idle processing capacity. Entropy is used by the edge switch to do preliminary calculations while also keeping track of the network's health and alerting the controller to any issues. In order to identify attacks at the micro-level, a set of classifiers called random forest was used in the admin panel. In order to determine if the sent traffic represents an attack, the classifier analyses the five vector tuples. If it does, it issues a discard instruction to the switch. They used Ryu as an SDN controller, Mininet as an emulator, and scapy to produce ICMP and SYN floods to test their strategy.

In order to generate UDP, TCP, and ICMP attacks and regular traffic, [13] used Mininet and Ryu controllers. From this traffic, they were able to extract 23 features. From a total of 23, they extracted 8 features. For accuracy, recall, FAR, precision, and F1 respectively, they scored 98.8 percent, 97.9 percent, 0.02 percent, 98.27 percent, and 97.65 percent.

Mininet and Floodlight controllers were used by [29] to create the SDN. They simultaneously generated constant-

packet-size normal and attack traffic. They created a dataset using the flow entries from the switch in order to extract the features. They created a dataset to train the model using 6 characteristics as input. The average accuracy and FAR for the SVM classifier were 95.24 percent and 1.26, respectively.

An approach for detecting and preventing DDoS attacks at the edge of a network is given by [30]. In the detection stage, it is expected that (i) successful connections are more likely than unsuccessful ones, and (ii) requests for connections from compromised hosts are significantly more frequent than those from healthy hosts. A host's ability to establish connections successfully or unsuccessfully over a period of time can be utilised to identify whether or not the host is infected. New flow rules are then sent from the SDN controller to the switch, blocking traffic from the malicious host.

Built a Support Vector Machine (SVM) using a Genetic Algorithm (GA) and Kernel Principal Component Analysis (KPCA) [31]. They were able to adjust the SVM's settings using GA and KPCA, respectively, to reduce the feature vector's dimension. An enhanced kernel function was applied to reduce the background noise caused by the feature variations significantly.

The SDN was developed by [32] using Miniedit and the OpenDayLight controller. They produced their dataset by generating DDoS attacks, SYN flooding DDoS attacks, ordinary traffic, UDP flooding, and five attribute-rich DDoS attack traffic. They examined one linear kernel with multiple decision functions using Advanced SVM to evaluate performance. It was possible to achieve a 97 percent accuracy, a 96 percent detection rate (recall), and a FAR of 0.02 percent.

To protect SDN controllers from DDoS attacks in the IoT paradigm, [33] presents a solution based on user trust levels. Based on their history, the controller assigns a trust value to each user in the network; a higher trust value means the controller would prioritize the request, whilst a lower trust level can cause the user's packets to be rejected. In addition, the controller has a queue for enquiries that, if it fills up, deletes the least trustworthy request to create room for a more reliable one. The authors of the study [34] strengthened the SDN architecture. They increased their resistance to DDoS attacks by enhancing the security of the SDN by applying a joint entropy-based security strategy (JESS).

3. Motivation

The primary driving force behind this effort is to create a controller-based attack detection method that effectively identifies attacks in an SD-IoT environment. The rationale behind doing this is that it preserves the core characteristic of SD-IoT: centralized management. Another method can still be applied to SD-IoT threat detection and mitigation. This strategy is based on the SD-IoT data plane. [18], [27] are just two examples of the numerous studies that have already been done to identify attacks in the SD-IoT data plane, but they do not support the fundamental aspect of SDN; therefore, an effective centralised controller-based attack detection approach has been devised in this method.

4. Methodology

In this work, two datasets have been used with several well-known and used classifiers to find the best-performing classifier. Both datasets are controller-based datasets, i.e., they have been generated from the logs created by southbound traffic, i.e., by OpenFlow protocol. The complete methodology is given in Fig. 2. In the first phase, preprocessing of the dataset is performed. It is then broken into two parts: train and test. In the second phase, the train part is used to train all the classifiers, and then the test part is used to check the performance of all the classifiers under some metrics. Following a performance review, the top-performing classifier is then chosen for implementation in the SD-IoT controller. In phase 3, this classifier is installed in the SD-IoT controller, determining whether the incoming communication is legitimate or malicious. By restricting the port from which the attack traffic originates, it lessens the impact of attacks.

4.1. Dataset Preprocessing

All the datasets' features already have numerical values, so there was no need to perform any preprocessing operations other than scaling, as there was a difference between the values of the features. For this, MinMax scaling was performed because it is easier and quicker to implement [35, 36]. The formula for performing MinMax scaling, equation 1, has been used.

$$x_i = \frac{(X - X_{min})}{(X_{max} - X_{min})} \quad (1)$$

Here, X is the single feature vector, X_{min} and X_{max} are its Minimum and Maximum values, and X_i is its scaled value.

4.2. Classifiers Used

In order to obtain the most effective classifier from the 6 and 8 features that have been extracted and selected, a literature survey was conducted on various machine learning algorithms known for their application in attack detection and mitigation. The surveyed algorithms included RF, XGB, LGBM, ET, GB, KNN, NB, SVM Linear, LR, and SVM Poly. Following dataset preparation, a thorough evaluation of various classifiers is performed. This assessment involves measuring the classifiers' performance utilizing several metrics, including precision, F1, CKC, recall, accuracy, FAR, Prediction Time, and AUC value. The most effective classifier is chosen for implementation within the controller of the SD-IoT infrastructure. A succinct exposition of classifiers is presented herewith:

4.2.1. Random Forest (RF)

The DT method and the Random Forest (RF) algorithm are similar. RF is made up of many different DTs. The bootstrap aggregation approach of" bagging" may be employed for unbalanced datasets. Bagging and RF, as the names suggest, use a random approach to generate output. In RF, a tree with all features is not necessarily constructed, but decision trees provide a fixed-size subset of all characteristics. As a result, the computational cost was considerably reduced. Each tree determines the contingent aspects of an RF on its own. When a sample reaches the root, it is disseminated to all child nodes in the tree structure. Each subtree's node displays the expected categorization for a particular sample as a label. That sample is then assigned to the highest category possible.

4.2.2. Extreme Gradient Boosting (XGB)

A research initiative at the University of Washington created the scalable ensemble-based ML technique known as Extreme Gradient Boosting (XGBoost). It quickly and accurately solves various ML issues with structured or tabular data [37]. The decision trees were constructed by XGBoost sequentially and with substantial weights. Weights are assigned to each independent variable and subsequently used to feed the DT that will predict the results. Variables that the tree incorrectly categorised are fed into the second DT if their weight increases. The individual classifiers are then combined to produce a more potent model. Some well-known applications of XGBoost include classification, regression, ranking, and custom prediction.

4.2.3. Light Gradient Boosting Machine (LGBM)

Light Gradient Boosting Machine (LGBM) is the second technique applied in this paper. This tree-based learning algorithm-based gradient boosting framework. Due to the homogeneous ensemble, [38] uses the same type of numerous weak classifiers for prediction. According to [39], the foundation of LGBM is Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

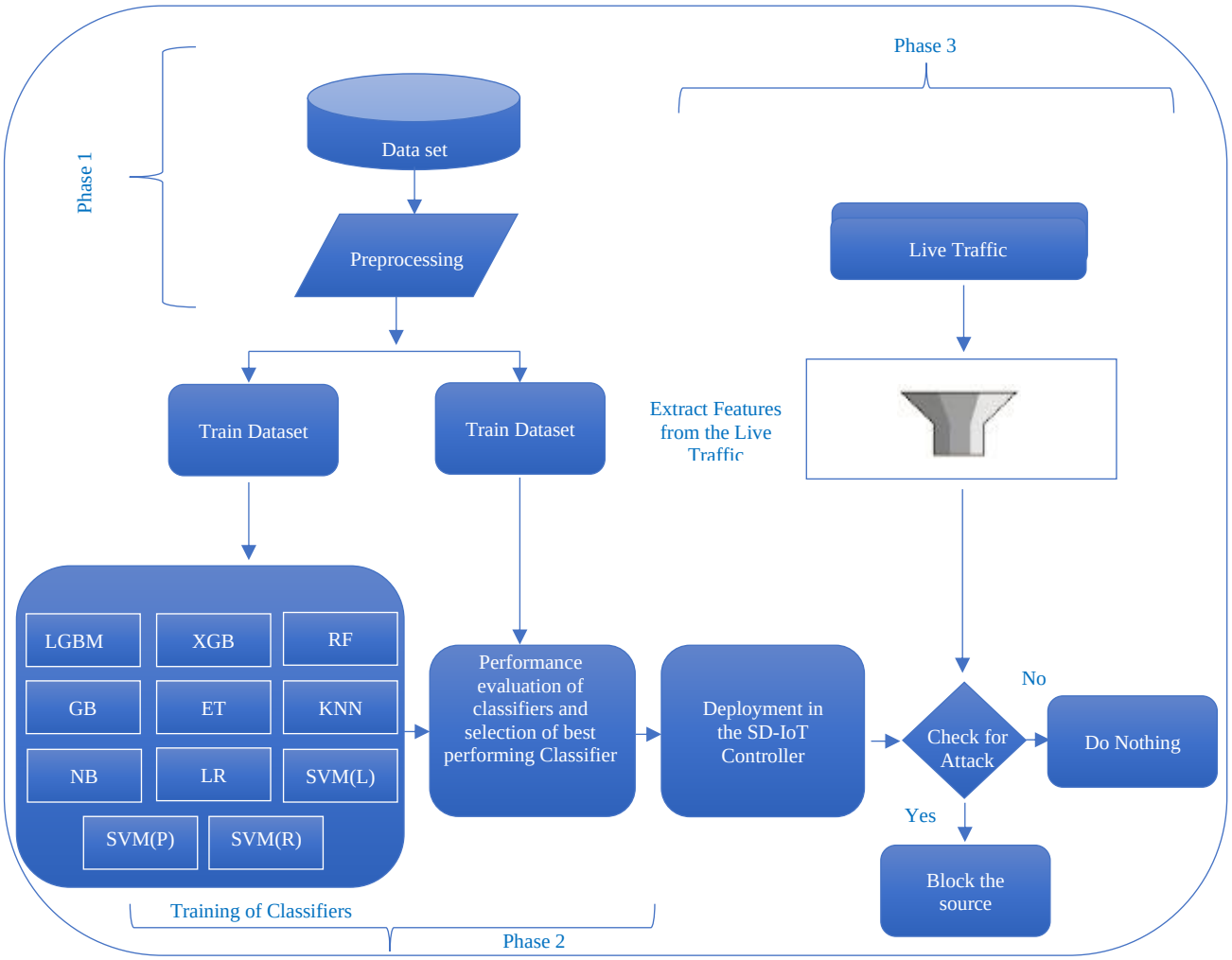


Fig. 2 Methodology of the proposed work

The GOSS recommends including cases with higher gradients and excluding instances with lower gradients since the information gained from instances with higher gradients is more than that gained from instances with lower gradients. Instances from the dataset are fewer as a result, and prediction time is faster without sacrificing efficiency. EFB makes an effort to group features that infrequently store nonzero values, such as the numerous features that are produced during a single hot encoding. Both GOSS and EFB work to reduce dataset size and increase efficiency; they both offer quicker training speeds and consume little memory. It supports parallel learning and has greater accuracy than any other boosting technique. According to[38, 39], the training pace of Light GBM is 20 times faster than that of traditional Gradient Boosting Decision Trees (GBDT) without sacrificing prediction accuracy. Because it is a boosting strategy, it lessens DT bias.

4.2.4. Support Vector Machine (SVM)

According to studies [3], [26], a support vector machine can be used as a classifier or a regressor. The foundation for its operation is the hyperplane and the vectors. The points that are closest to the hyperplane are referred to as vectors, and the plane that lies between those points is called the hyperplane. SVM may utilise a variety of kernels, including linear, polynomial, Gaussian, etc., depending on the dynamics of the goal value.

4.2.5. KNN

KNN is a supervised technique that can be used to solve classification and regression issues. It needs to calculate the distance between the object in question and the k nearest points, making it a slow learner, but it can be used to resolve the categorization issue. In contrast, the KNN Classifier, which is noise resistant and produces the maximum precision, efficiently identifies DDoS attacks. The distance between two points can be calculated using the Manhattan, Minkowski, and Euclidean distance functions.

Some of the attack detection works [3, 40–42] are using it, so it has also been used for the same objective in the SD-IoT context.

4.2.6. NB(NB)

According to the independent variable comparison concept, NB determines the link between these independent variables [43]. The Bayes theorem, which stipulates that attributes must be true, is the foundation of this machine-learning approach. Due to the algorithm's lack of parameter evaluation, it is easy to design. It can now handle very huge datasets thanks to this. The class variable y and input vector values are related in the way that the Bayes theorem specifies as x1 through xn:

$$P(y \mid x1, \ldots, xn) = \frac{P(y)P(x1, \ldots, xn \mid y)}{P(x1, \ldots, xn)} \quad (2)$$

4.2.7. LR(LR))

Logistic regression is a common Machine Learning method that belongs to the Supervised Learning approach. It is used to forecast the categorical dependent variable from a group of independent factors.

4.3. Classifier Training and Deployment

After performing preprocessing, the next phase is to train all the classifiers and test their performance. For this, k-fold cross-validation has been performed. The value of K has been set to 10 based on the works done in [43-45].

5. Implementation Environment

For implementing the complete things, Python programming has been used. In many places, built-in libraries developed in Python have been used, like Scikitlearn, Numpy, Pandas, etc. For creating the SDN environment, Mininet and Ryu have been used as remote controllers. Ryu has been stated to perform better than other controllers [46], so it has been used in this work. Ubuntu 18 has been used as an Operating System. The system has 8GB RAM and an AMD PRO R7 processor.

6. Experiments and Discussion

In this section, details about performing different operations are given.

6.1. Traffic Generation, Feature Extraction and Dataset Creation

All the Mininet emulators have been used to create the SDN topologies, and the Ryu controller has been used in remote mode as a controller. For creating dataset (A), mgen and hping3 tools have generated normal and attack traffic, respectively. Similarly, for creating dataset (B), D-ITG [47] and hping3 tools have been used to generate normal and attack traffic, respectively. The complete details about the creation of dataset (A) and dataset (B) is given in [13], respectively. The names of the features of these datasets are given in Table 1 and Table 3.

6.2. Analysis of the Features

To show the relationship among the features, both datasets' heat matrices are given in Fig. 3 and Fig. 4, respectively. From the heat matrix, it can be clearly observed how informative these features are.

Table 1. Name of the features of dataset(A)

S.No	Name of Feature
1	Count of Source IP (SIP Count)
2	Port Count (Port Count)
3	Flow Pair to Flow Count Ratio (FPFCR)
4	Packet In count Difference (PIN Diff)
5	Lookup Count Difference (Look Up Count)
6	Packet Type (UDP/TCP/ICMP) (Packet Type)

Table 2. Details about the samples collected for different traffic in dataset(A)

Traffic	Type	Number of Samples
ICMP	Attack	235
	Normal	245
TCP	Attack	220
	Normal	255
UDP	Attack	230
	Normal	240

6.3. Dataset Preprocessing

No action other than feature scaling has been done because all the features in both datasets already have numerical values.

Table 3. Name of the features of dataset(B)

S.No	Features Used
1	Average Packet count per flow (APPF)
2	Average Byte count per flow (ABPF)
3	Total number of flows in a switch (Total Flow)
4	Protocol
5	Duration
6	Number of Packet in messages (PIN Count)
7	Packet rate (Packet Rate)
8	Port Bandwidth (Port Bandwidth)

Table 4. Traffic category of each traffic instance

Traffic class	Benign	Malicious
ICMP	24957	16364
TCP	18897	10539
UDP	22772	10816

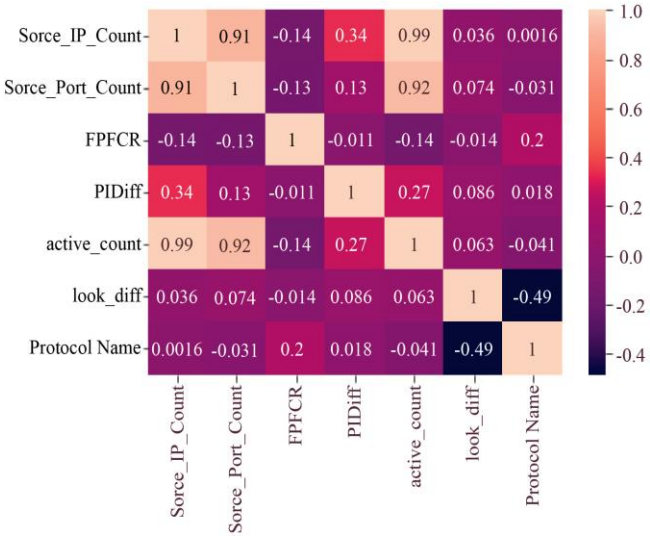


Fig. 3 Heat matrix of the features of dataset A

6.4. Evaluation of the Models

The performance of all the classifiers has been evaluated under Dataset (A) and Dataset (B). All the classifiers have been trained and verified using 10-fold cross-validation to get accurate results. The names of the metrics under which performance has been evaluated are precision, F1, CKC, recall, accuracy, FAR, Prediction Time, and AUC value.

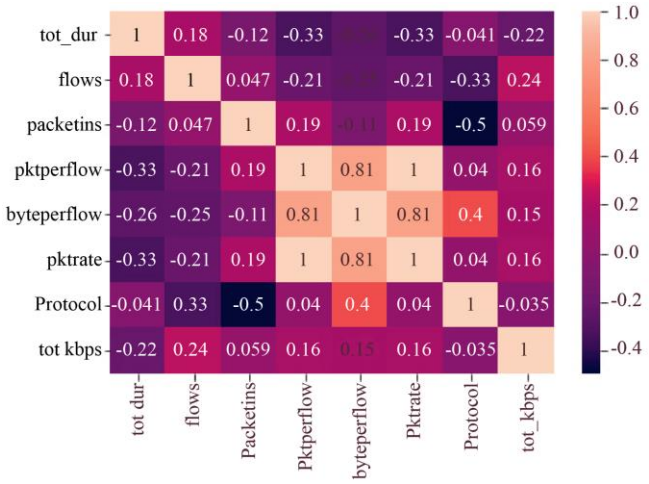


Fig. 4 Heat matrix of the features of dataset B

Equations 3 to 8 show the formulas used for calculating these metrics. They have also been described in detail in [48].

6.4.1. Accuracy

The percentage of correctly predicted test outcomes is referred to as accuracy.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \text{ X } 100 \qquad (3)$$

6.4.2. Precision

Precision is defined as the percentage of relevant examples (true positives) among all examples predicted to belong in a given class. Details are given in [50].

$$\text{Precision} = \frac{TP}{TP + FP} \text{ X } 100 \qquad (4)$$

6.4.3. Recall

The proportion of examples expected to belong to a class compared to all examples that actually belong in the class is known as recall.

$$\text{Recall} = \frac{TP}{TP + FN} \text{ X } 100 \qquad (5)$$

6.4.4. F1-Value

Instead of focusing on overall performance like accuracy does, the F1 score is an alternative machine learning evaluation statistic that assesses a model's prediction skills by focusing on its class-wise performance. The F1 score combines two conflicting metrics of a model's precision and recall scores, which has led to its extensive adoption in recent literature.

$$\text{F1-Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \quad \text{X} \quad 100 \qquad (6)$$

6.4.5. False Alarm Rate (FAR)

The FAR measures the percentage of negative data cases that are incorrectly labelled as positive cases.

$$\text{False Alarm Rate} = \frac{\text{F P}}{\text{F P} + \text{T N}} \text{ X } 100 \qquad (7)$$

6.4.6. Cohen's Kappa Coefficient

For both multiclass and imbalanced class issues, Cohen's Kappa statistic is a great tool. Other measures are unable to express the full picture in situations involving several classes and unbalanced classes, but Kappa can.

$$k = \frac{\text{po - pe}}{\text{1 - pe}} = 1 - \frac{\text{1- po}}{\text{1 - pe}} \text{X } 100 \qquad (8)$$

The value of this coefficient might be less than or equal to 1. Values near 0 or less than 0 indicate that the classifier is not useful, while a value close to 1 shows better performance.

Testing Time

The classifier's testing period is the time needed to forecast test data. Testing time is more significant than training time since it demonstrates how long it takes to observe an attack in a practical setting.

AUC-ROC Curve

This curve is used to measure problems with binary classification. The true positive and false positive rates are on this probability curve. The AUC gauges how well a classifier can distinguish between classes.

6.5. Performance Evaluation of the Classifiers and Discussions

The performance of all the classifiers has been evaluated under Dataset (A) and Dataset (B). All the classifiers have been trained.

6.5.1. Under the Dataset(A)

The performance of all the classifiers has been evaluated under the above metrics to find the best-performing classifier in dataset A. The performance under these metrics has been shown in Fig. 5 and Table 5.

Performance evaluation in terms of accuracy of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), has been shown in Fig. 5a. The values of the accuracy of all the classifiers are 98.47%, 98.09%, 98.09%, 98.41%, 98.79%, 96.76%, 98.22%, 94.41%, 94.41%, 93.52%, 95.10%, respectively. It is found that GB is performing better than other classifiers in terms of accuracy.

Performance evaluation in terms of Precision of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been shown in Fig. 5b. The values of accuracy of all the classifiers are 98.54%, 98.19%, 98.18%, 98.48%, 98.83%, 96.97%, 98.31%, 94.45%, 94.56%, 93.73%, 95.61%, respectively. It is found that GB performs better than other classifiers in terms of precision.

Performance evaluation regarding the recall of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been shown in Fig. 5c. The values of the accuracy of all the classifiers are 98.47%, 98.09%, 98.09%, 98.41%, 98.79%, 96.76%, 98.22%, 94.41%,94.41%, 93.52% and 95.10% respectively. It is found that GB is performing better than other classifiers regarding recall.

Performance evaluation in terms of the f1 score of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, SVM(Poly), have been shown in Fig. 5b.The values of f1 score of all the classifiers are 98.47%, 98.09%, 98.09%, 98.40%, 98.79%, 96.75%, 98.22%, 94.40%, 94.40%, 93.50% and 95.08%. It is found that GB is performing better than other classifiers regarding the f1 score. Performance evaluation in terms of prediction time of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), has been shown in Fig. 5f. The values of prediction time of all the classifiers are 1.04%,0.49%, 0.42%, 0.76%, 0.80%, 0.27%, 0.09%, 0.06%, 0.21%, 0.11%, 0.28%, It is found that GB is performing better than other classifiers in terms of prediction time.

Performance evaluation in terms of FAR of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), has been shown in Fig. 5e. The values of prediction time of all the classifiers are 0.00%, 0.00%, 0.00%, 0.00%, 0.00%, 0.00%, 0.00%, 4.45%, 3.14%, 3.07%, 0.00%. It is found that GB performs better than other classifiers in terms of FAR.

Performance evaluation in terms of CKC of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, SVM(Poly), have been shown in Fig. 5g. The values of prediction time of all the classifiers are 96.93%, 96.15%, 96.15%, 96.77%, 97.56%, 93.45%, 96.42%, 88.72%, 88.78%, 86.82%, 90.14%. It is found that GB performs better than other classifiers in CKC.

6.5.2. Under the Dataset(B)

The performance of all the classifiers has been evaluated under the above metrics to find the best-performing classifier in dataset B. The performance under these metrics is shown in Fig. 7 and Table 6.

Performance evaluation in terms of accuracy of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been shown in Fig. 7a. The values of prediction time of all the classifiers are

99.558%, 99.604%, 99.704%, 99.567% 98.274% 92.902% 98.463% 61.628% 61.624% 68.003% 86.566%. It is found that LGBM is performing better than other classifiers in terms of accuracy.

Performance evaluation in terms of precision of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM (Poly), have been shown in Fig. 7b. The values of precision of all the classifiers are 99.558%, 99.605%, 99.705%, 99.567%, 98.282%, 92.894%, 98.463%, 61.293%, 59.707%, 67.532%, 86.754%. It is found that LGBM is performing better than other classifiers in terms of precision.

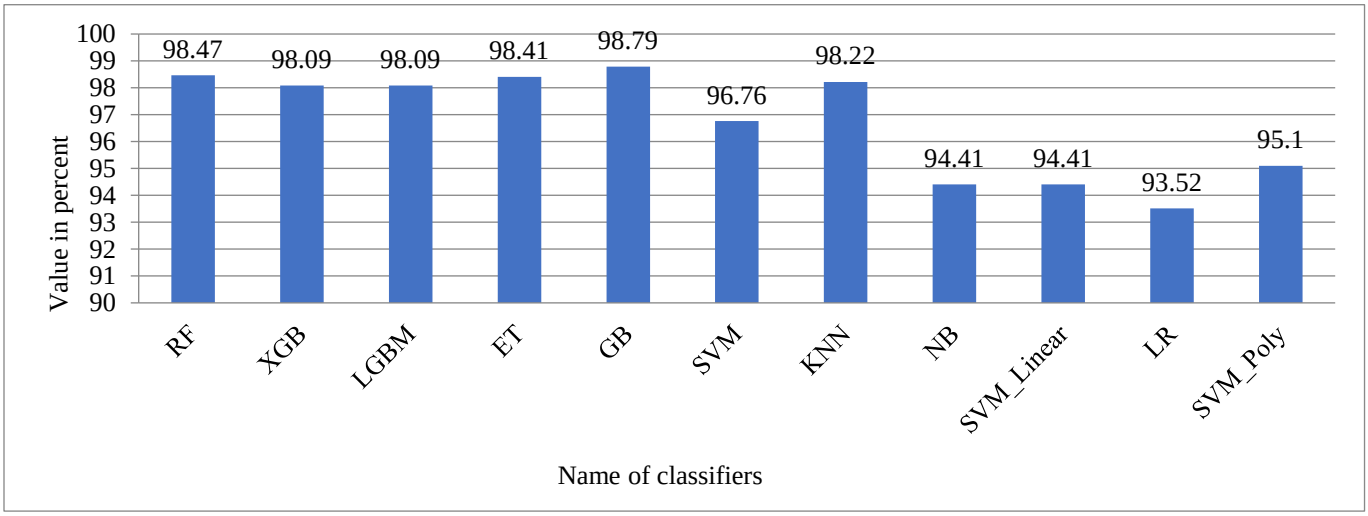
Performance evaluation in terms of recall of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been shown in Fig. 7c. The values of recall of all the classifiers are 99.558%, 99.604%, 99.704%, 99.567%, 98.274%, 92.902%, 98.463%, 61.628%, 61.624%, 68.003%, 86.566%. It is found that LGBM is performing better than other classifiers in terms of recall.

Performance evaluation in terms of F1 of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, SVM(Poly), have been shown in Fig. 7d. The values of F1 of all the classifiers are 99.558%, 99.604%, 99.704%, 99.567%, 98.276%, 92.895%, 98.463%, 61.430%, 50.962%, 65.372%, 86.326%. It was found that LGBM performs better than other classifiers in terms of F1.
Performance evaluation in terms of FAR of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly), have been shown in Fig. 7e. The values of FAR of all the classifiers are 0.425%, 0.389%, 0.380%, 0.382%, 1.759%, 5.487%, 1.280%, 29.829%, 3.576%, 11.938%, 6.162%. It was found that LGBM is performing better than other classifiers regarding FAR.

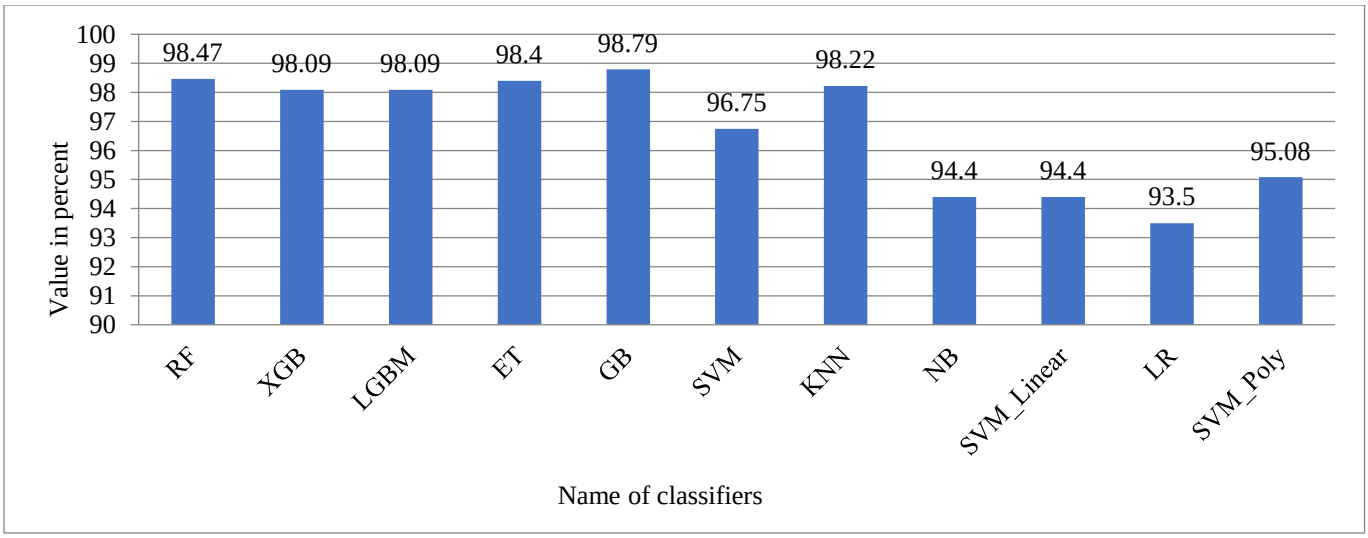
Performance evaluation in terms of prediction time of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, SVM(Poly), have been shown in Fig. 7f. The values of prediction time of all the classifiers are 32.467%, 7.710%, 2.900%, 20.401%, 67.854%, 4633.667%, 8.437%, 0.661%, 8336.396%, 2.250%, 4489.978%. It is found that LGBM is performing better than other classifiers regarding prediction time.

Performance evaluation in terms of CKC of all the classifiers, namely RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, SVM(Poly), have been shown in Fig. 7g. The values of CKC of all the classifiers are 99.069%, 99.169%, 99.378%, 99.089%, 96.378%, 85.026%, 96.774%, 18.651%, 4.257%, 26.597%, 70.989%. It was found that LGBM is performing better than other classifiers regarding CKC.

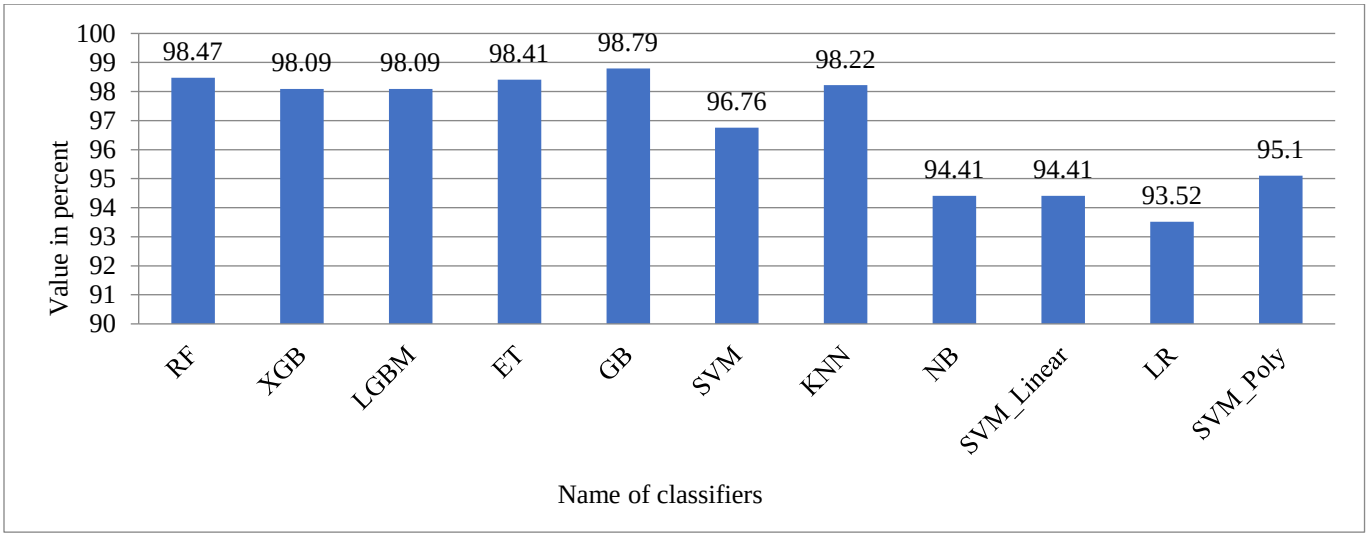
The reason for the better performance of LGBM is that the foundation of LGBM uses Gradient-based OneSide Sampling (GOSS) and Exclusive Feature Bundling (EFB). The GOSS recommends including cases with higher gradients and excluding instances with lower gradients since the information gained from instances with higher gradients is more than that gained from instances with lower gradients. Instances from the dataset are fewer as a result, and prediction time is faster without sacrificing efficiency. EFB tries to group features that infrequently store nonzero values, such as the numerous features produced during a single hot encoding. Both GOSS and EFB work to reduce dataset size and increase efficiency; they both offer quicker training speeds and consume little memory. It supports parallel learning and has greater accuracy than any other boosting technique. According to [33], [34], the training pace of LightGBM is 20 times faster than that of traditional Gradient Boosting Decision Trees (GBDT) without sacrificing prediction accuracy. Because it is a boosting strategy, it lessens DT bias.



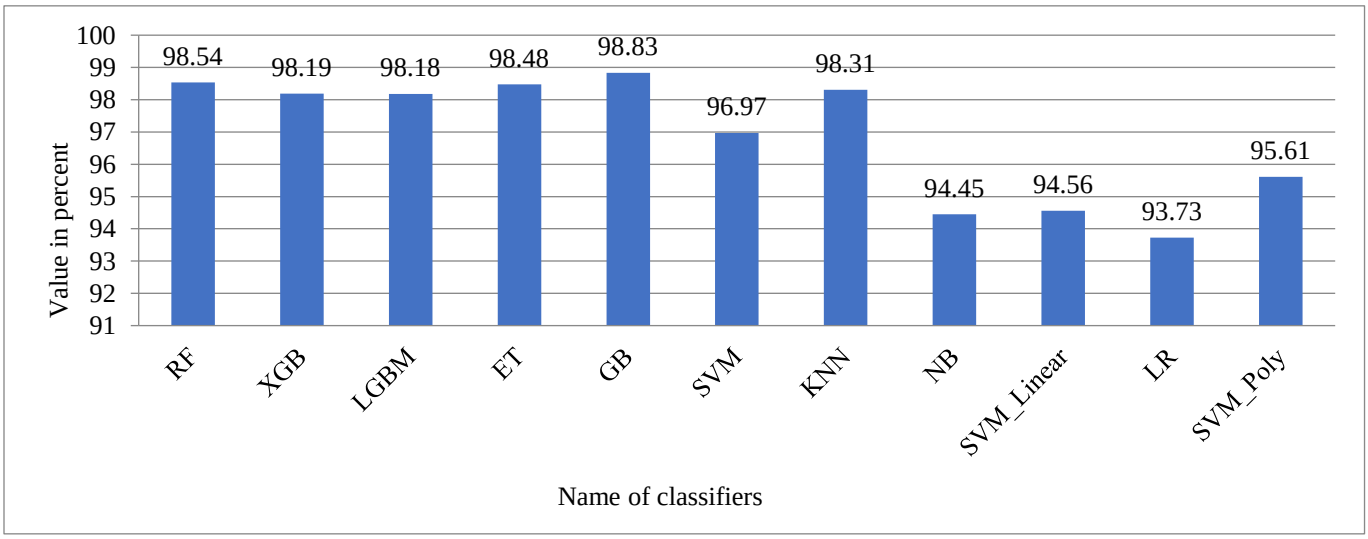
(a)


Value in percent
100
99
98
97
96
95
94
93
92
91
90
98.47
98.09
98.09
98.4
98.79
96.75
98.22
94.4
94.4
93.5
95.08
RF
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

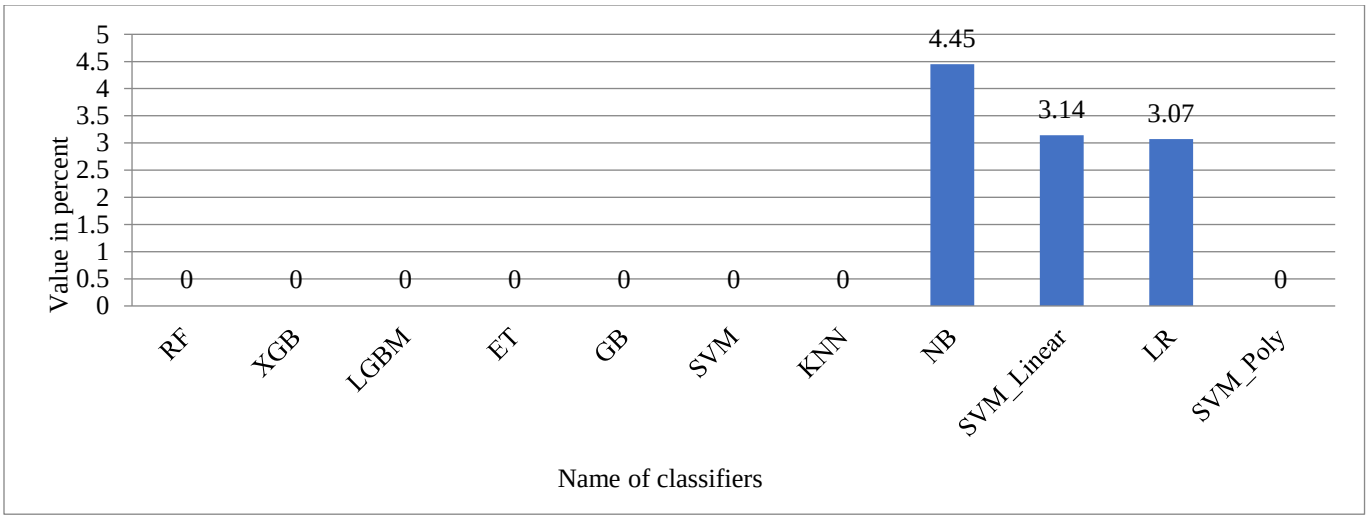
(b)


Value in percent
100
99
98
97
96
95
94
93
92
91
90
98.47
98.09
98.09
98.41
98.79
96.76
98.22
94.41
94.41
93.52
95.1
RF
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

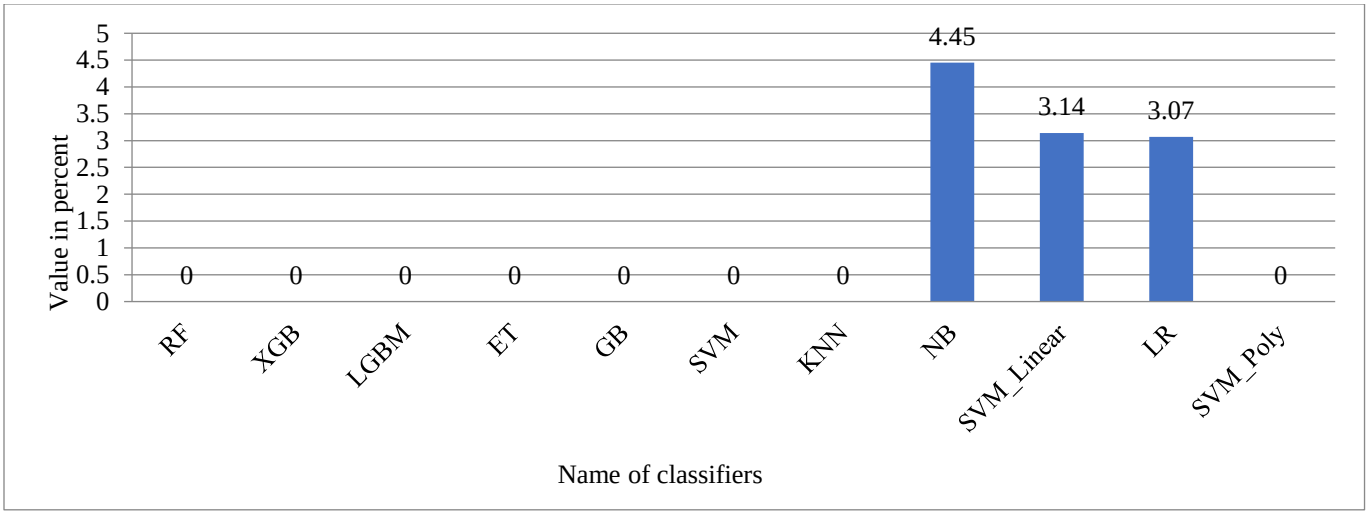
(c)


Value in percent
100
99
98
97
96
95
94
93
92
91
98.54
98.19
98.18
98.48
98.83
96.97
98.31
94.45
94.56
93.73
95.61
RF
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

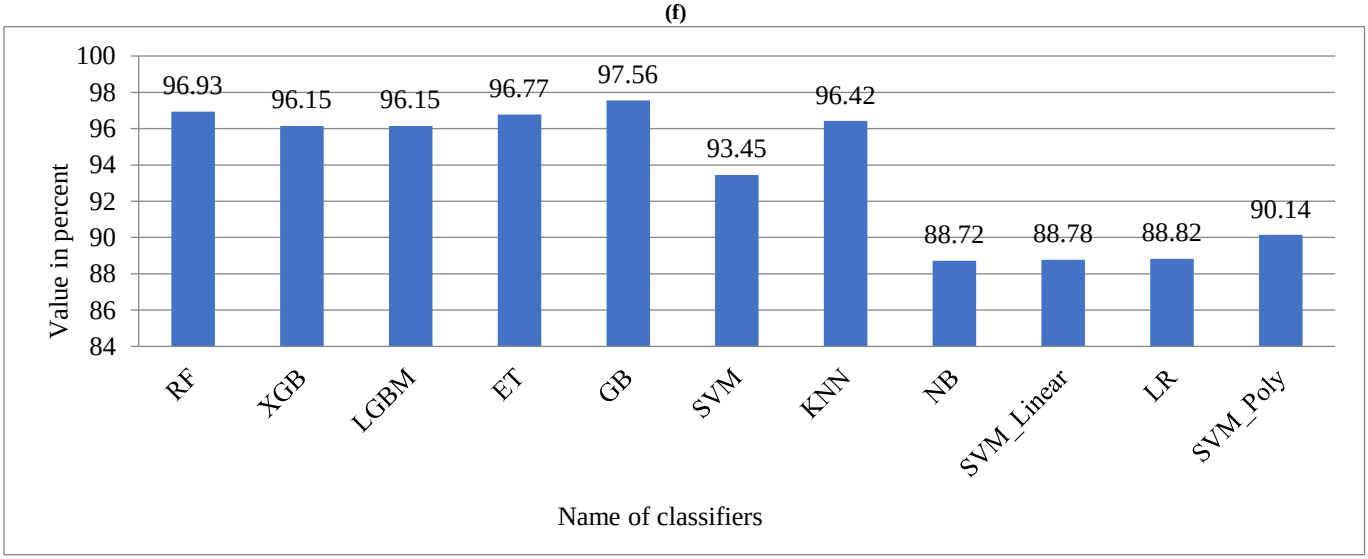
(d)



(e)



(f)



(g)

Fig. 5 Performance comparison of all the classifiers in the dataset(A) under metrics (a). Accuracy (b). Precision (c). Recall (d). F1 (e). FAR (f). Prediction Time (g). CKC

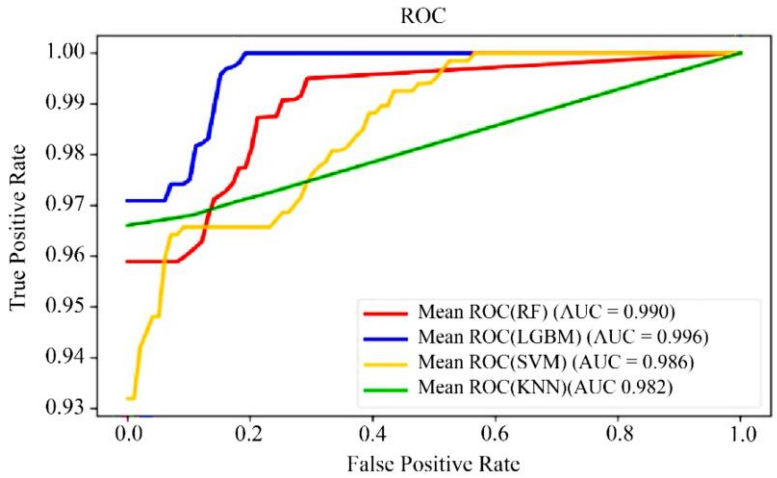
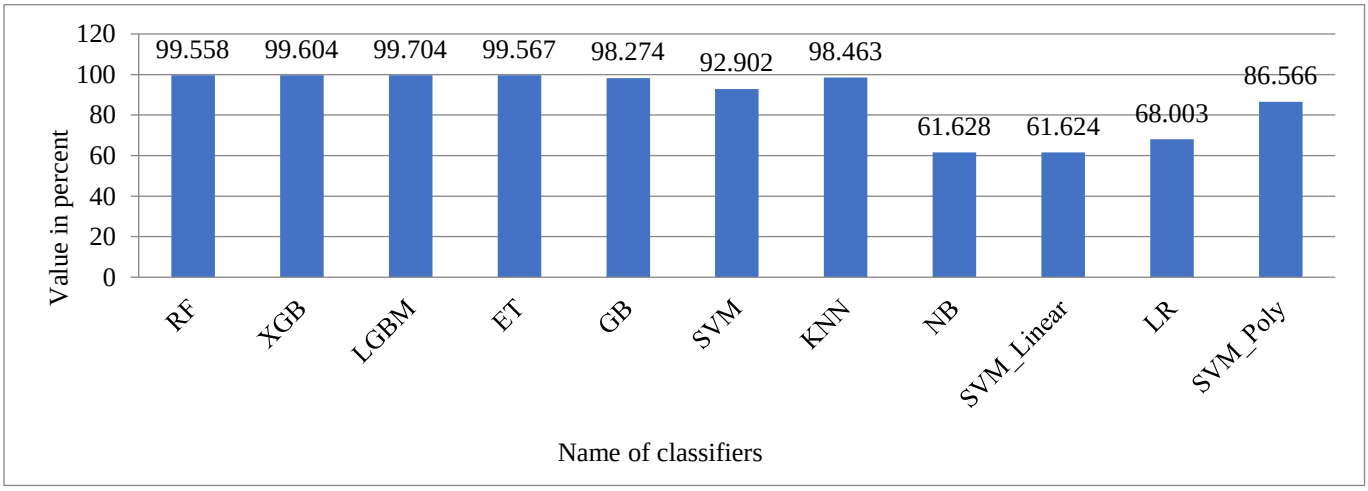


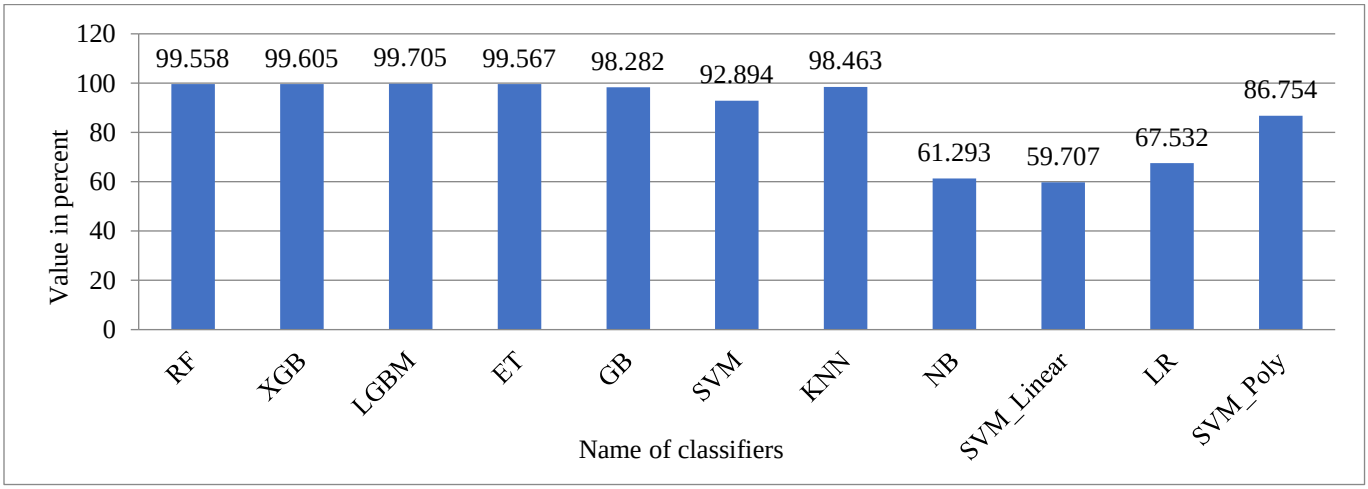
Fig. 6 10-Fold mean AUC-ROC curve of all the classifiers under dataset(A)

6.6. Model Deployment in the Controllers

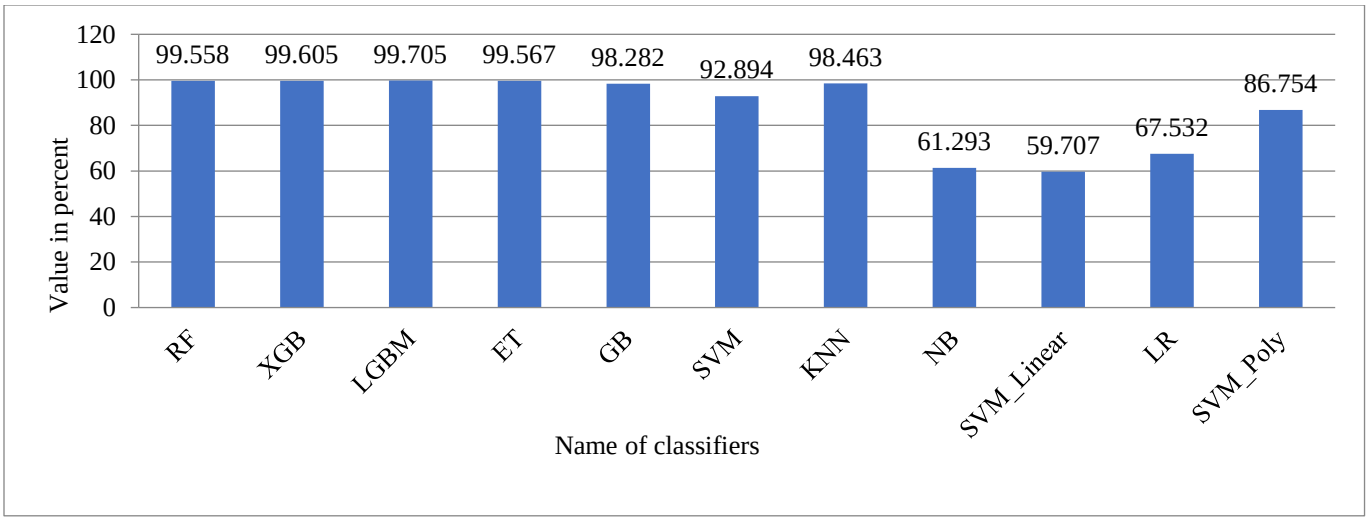
From the above results and discussion, it is found that LGBM is the best-performing classifier among others that can be deployed in the controller of SD-IoT. Once deployed, it can detect and mitigate the attack from the live traffic. The algorithm to detect the attack is given in algorithm 1, and the algorithm to mitigate the attack is given in algorithm 2.



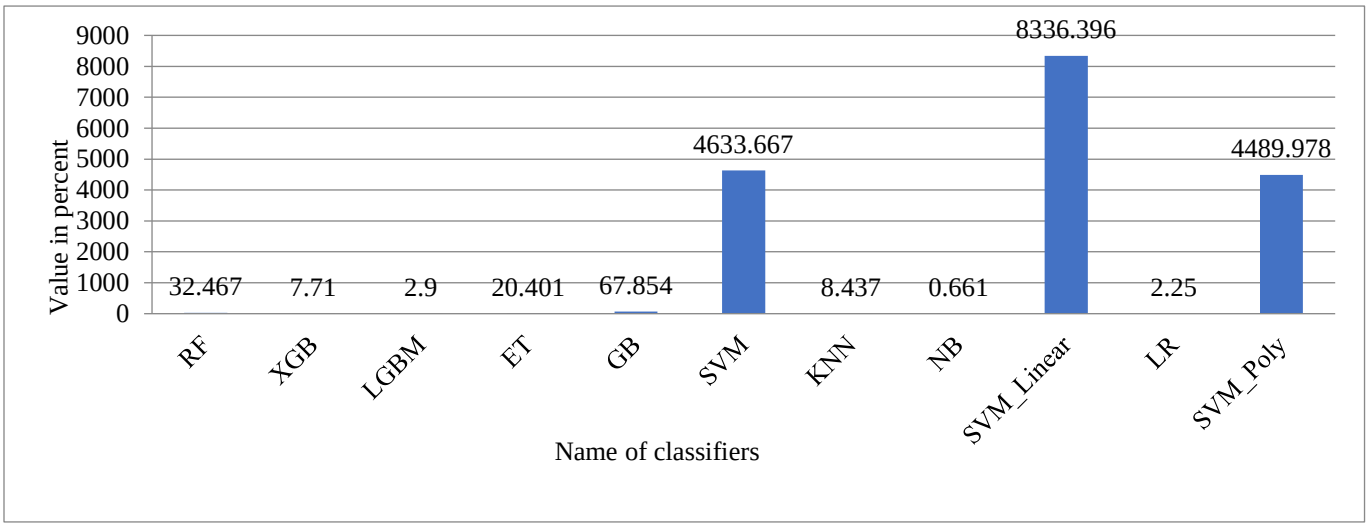
(a)



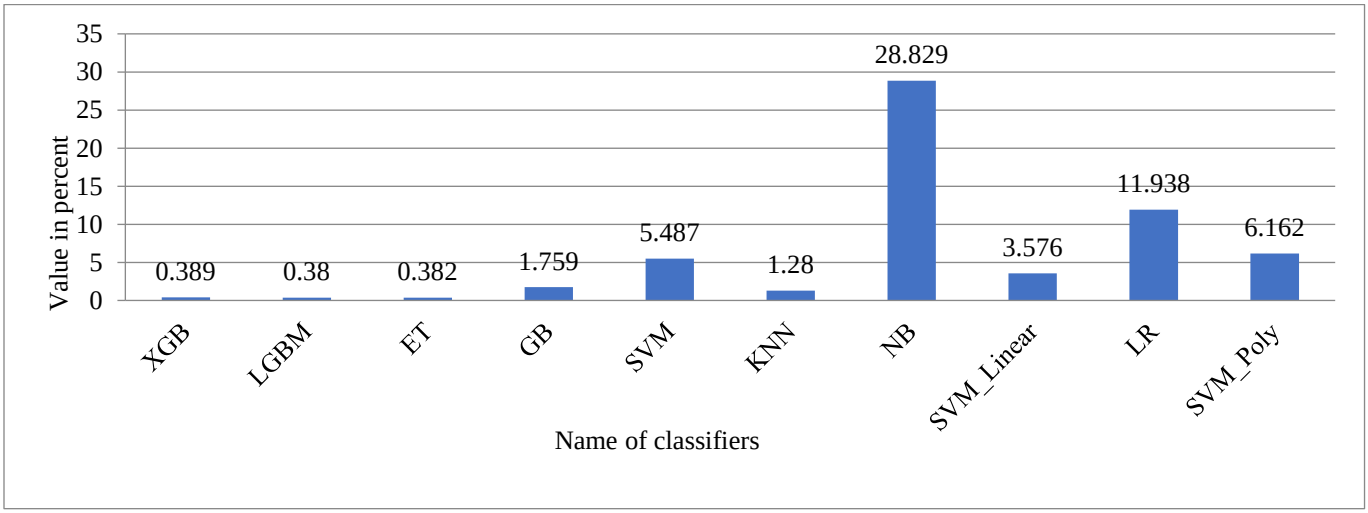
(b)


120
100
80
60
40
20
0
Value in percent
99.558
99.605
99.705
99.567
98.282
92.894
98.463
61.293
59.707
67.532
86.754
RF
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

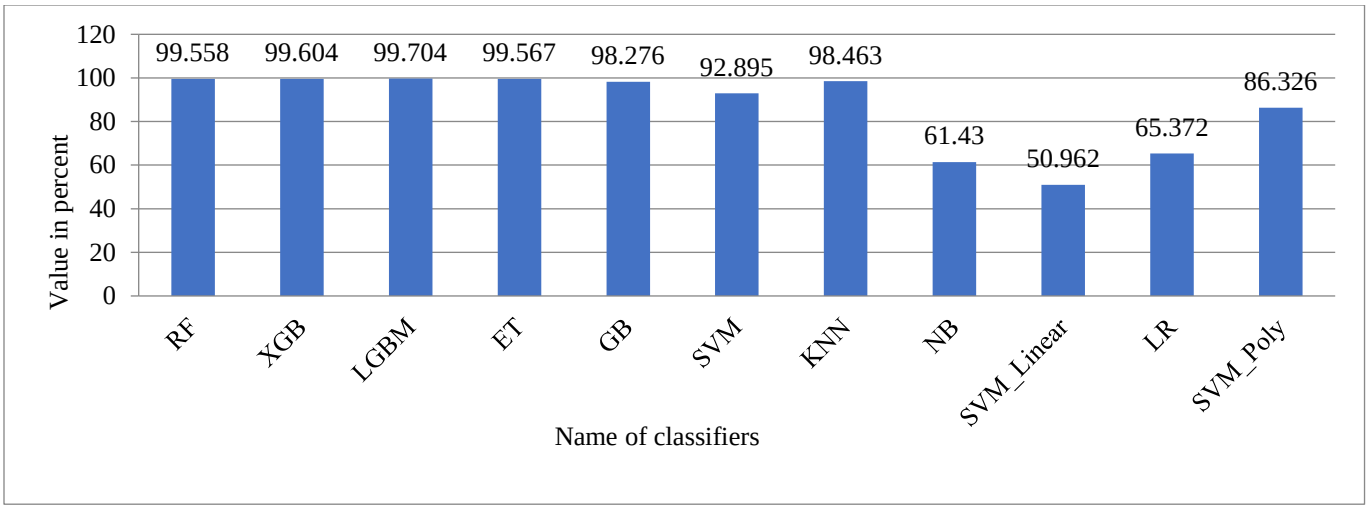
(c)


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Value in percent
32.467
7.71
2.9
20.401
67.854
4633.667
8.437
0.661
8336.396
2.25
4489.978
RF
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

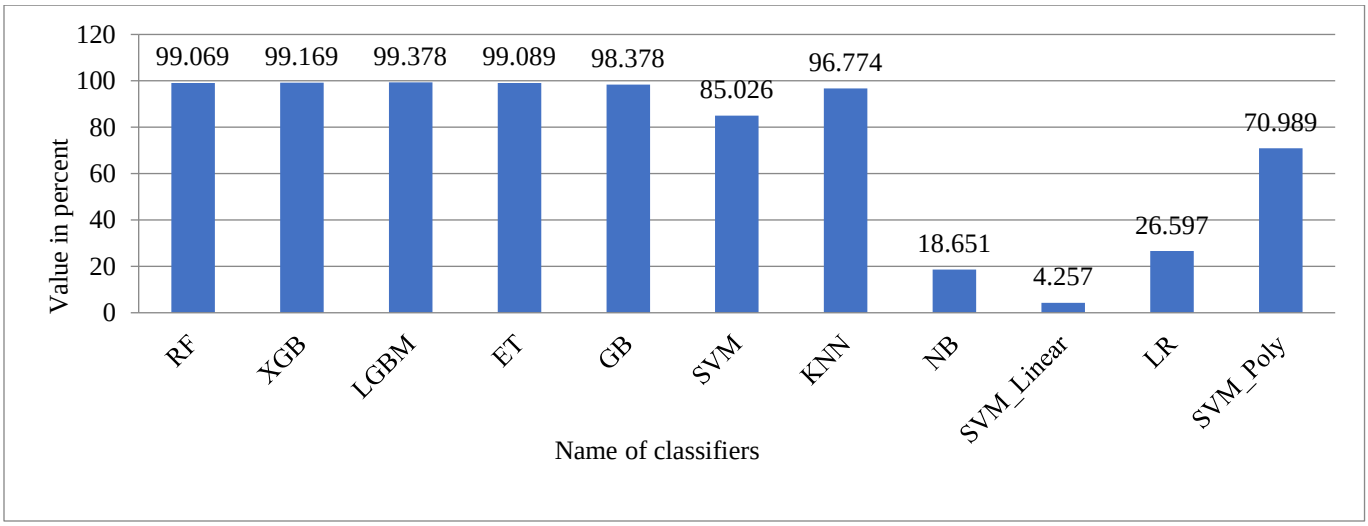
(d)


35
30
25
20
15
10
5
0
Value in percent
0.389
0.38
0.382
1.759
5.487
1.28
28.829
3.576
11.938
6.162
XGB
LGBM
ET
GB
SVM
KNN
NB
SVM_Linear
LR
SVM_Poly
Name of classifiers

(e)



(f)



(g)

Fig. 7 Performance comparison of all the classifiers in the dataset(B) under metrics (a). Accuracy (b). Precision (c). Recall (d). F1 (e). FAR (f). Prediction Time (g). CKC

Table 5. Performance of different classifiers under dataset(A)

Classifiers	RF	XGB	LGBM	ET	GB	SVM (RBF)	KNN	NB	SVM (Linear)	LR	SVM (Poly)
Accuracy	98.47 %	98.09%	98.09%	98.41%	98.79%	96.76%	98.22%	94.41%	94.41%	93.52%	95.10%
Precision	98.54	98.19	98.18	98.48	98.83	96.97	98.31	94.45	94.56	93.73	95.61
Recall	98.47	98.09	98.09	98.41	98.79	96.76	98.22	94.41	94.41	93.52	95.10
F1	98.47	98.09	98.09	98.40	98.79	96.75	98.22	94.40	94.40	93.50	95.08
Prediction Time	1.04	0.49	0.42	0.76	0.80	0.27	0.09	0.06	0.21	0.11	0.28
FAR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.45	3.14	3.07	0.00
CKC	96.93	96.15	96.15	96.77	97.56	93.45	96.42	88.72	88.78	86.82	90.14

Table 6. Performance of different classifiers under dataset(B)

Classifiers	RF	XGB	LGBM	ET	GB	SVM	KNN	NB	SVM (Linear)	LR	SVM (Poly)
Accuracy	99.558	99.604	99.704	99.567	98.274	92.902	98.463	61.628	61.624	68.003	86.566
Precision	99.558	99.605	99.705	99.567	98.282	92.894	98.463	61.293	59.707	67.532	86.754
Recall	99.558	99.604	99.704	99.567	98.274	92.902	98.463	61.628	61.624	68.003	86.566
F1	99.558	99.604	99.704	99.567	98.276	92.895	98.463	61.430	50.962	65.372	86.326
Prediction Time	32.467	7.710	2.900	20.401	67.854	4633.667	8.437	0.661	8336.396	2.250	4489.978
FAR	0.425	0.389	0.380	0.382	1.759	5.487	1.280	29.829	3.576	11.938	6.162
CKC	99.069	99.169	99.378	99.089	96.378	85.026	96.774	18.651	4.257	26.597	70.989

Table 7. Comparison with other similar works

Ref.& Year	Accuracy	Recall	Precision	F1 Score	FAR	Prediction Time	CKC
[3], 2020	92.11%	88.71%	91.42%	89.91%	NA	NA	NA
[29], 2018	95.24%	NA	NA	NA	1.26%	NA	NA
[14], 2020	97.9%	97.6 %	97.2 %	97.2%	NA	NA	NA
[13], 2021	98.8%	97.91%	98.27%	97.65%	0.02%	NA	NA
[32], 2019	97%	96%	NA	NA	0.02%	NA	NA
[50], 2020	99.9%	NA	NA	NA	NA	NA	NA
[Proposed work]	99.704%	99.705%	99.704%	99.704%	2.900%	0.380%	97.29%

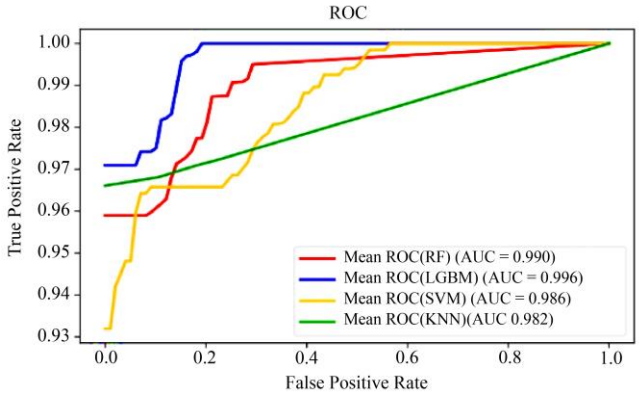


Fig. 8 10-Fold mean AUC-ROC curve of all the classifiers under dataset(B)

For the deployment, the pickle file of the trained LGBM is selected and installed as a module in the controller of SDN. The current work targets the Ryu controller; that is why the pickle file is being taken, which is basically a Python file. Apart from the LGBM's pickle file, the scaler function's pickle file is also required for scaling the features extracted from the live traffic.

The working of the SD-Controller after deploying the attack detection and mitigation module is shown in Fig. 9. In this fig., the feature extractor module installed in the controller will extract all the required features and send them to the attack detection module. This attack detection module will check whether it is normal traffic or attack traffic. Once attack traffic is detected, the mitigation module will send the necessary steps to the data plane device to immediately stop the attack for a fixed interval.

After this mentioned interval, the blocked source is unblocked, and again, the feature extractor module will extract the feature and the above process is repeated.

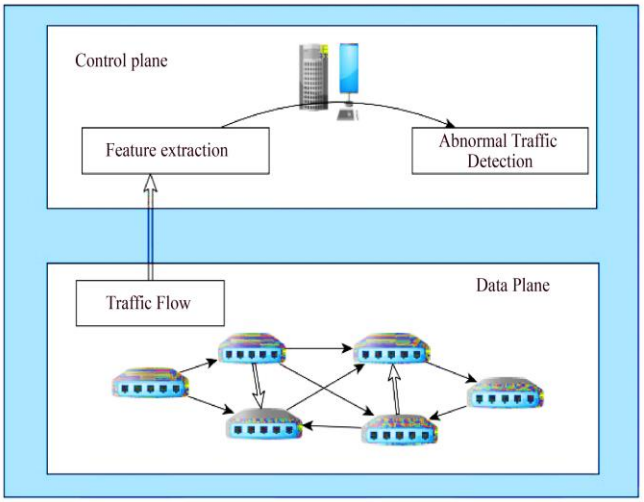


Fig. 9 Deployment of the trained classifier in SDN controller

Algorithm 1: Algorithm for Attack Detection in the SD-IoT Controller

Input: trained model: Model which has been trained for attack detection,
trained MinMaxScaler: Scaler which has been trained for scaling the features.
Output: The traffic status is either Attack or Normal.
/* To read the statistics send an OFPFlowStats request to the switch;
Load model and scaler and Goto the EventOFPFlowStatsReply handler */

1. model ← load(trained model)
2. scaler ← load(trained M inMaxScaler)
   /* By extracting features from real-time traffic, a tuple of new observations can be created */
3. ***if*** dataset = A then
   /* Extract the required features from dataset(A) */
4. new observation= $\Pi_{SIP\_Count,Port\_Count,FPFCR,PIN\_Diff,Look\_Up\_Count,PacketType}$ (Traffic)
5. ***Else***
6. /* Extract the required feature from dataset(B) */
   new observation =$\Pi_{APPF,ABPF,Total\_Flow,P\ rotocol,Duration,PIN\_Count}$(Traffic)
7. *scaled_observation*=scaler.transform(*new_observation*)
8. result = *model.predict(scaled_observation)*
9. Return result

Algorithm 2: Algorithm to prevent the attacks in SD-IoT.

Input: T: Pause Time,
Idle Timeout Value: Flow entry lifetime
Trained Model: Trained Model,
Scaler: Scaler function.
Output: Block the attacking port

/* Set the pause time */
1. INTERVAL=T
2. sleep(INTERVAL)
   /* Get the datapath of all the switches and then send the OFPFlowStateRequest to all the switches for reading the data */
3. **foreach** dp in {datapath.values} **do**
   OFPFlowStateRequest(dp)
   /* Get the responses from all the switches and read statistics. */
4. **foreach** Stat in {en.msg.body} **do**
   /* Check attack using algorithm 1. */
5. attack = Detect_Attack(Trained_Model, Scaler)
6. if attack = 1 then
7. mac adr ← stat.src ether
   /*Get the physical address */
8. to do action ← []
   /* Set No action to drop the traffic */
9. get_priority ← highets_priority_flow()
   match←flow_entry.match[eth_src=mac _adr]
   /* Find the flow entries having this physical address */
10. add_flow(datapath, get_priority, to_do_action, match, idle_timeout = N)
    /* Add the flow entry to the block source for N seconds. */
11. **else**
12. Let the normal operations go on.

6.7. Comparison and Discussion with Other Similar Works

The works similar to this study, i.e., those that detect and/or mitigate the attacks centrally in an SDN environment, have been selected, compared and shown in Table 7.

7. Contribution

In this work, a centralized controller-based attack detection and mitigation approach has been proposed. With the objective of finding the best attack detection model, a number of classifiers have been trained and tested with two datasets created in the controller of the SDN. It is found that LGBM outperforms other classifiers, so it has been deployed as a module in the controller so that it can detect the attack. Further, an attack mitigation approach for the Ryu controller has been given to extend the work, which can centrally block the attacking source immediately after it detects it.

8. Conclusion and Future Work

This work proposes a controller-based DDoS attack detection and mitigation approach in the SD-IoT architecture.

Two datasets created in the controller of SDN have been used with the objective of finding the best-performing classifier with them.

The names of the classifiers used for performance evaluation are RF, XGB, LGBM, ET, GB, SVM, KNN, NB, SVM(Linear), LR, and SVM(Poly).

It is found that LGBM outperforms other classifiers under dataset(A) in terms of precision, F1, Cohen's Kappa Coefficient (CKC), recall, accuracy, False Alarm Rate (FAR), Testing Time, and AUC value, and giving performance of 98.47%, 98.09%, 98.09%, 98.41%, 98.79%, 96.76%, 98.22%, 94.41%, 94.41%, 93.52% and 95.10%, respectively.

Under dataset(B), LGBM again outperforms other classifiers and gives the performance of 99.558%, 99.604%, 99.704%, 99.567%, 98.274%,92.902%, 98.463%, 61.628%, 61.624%, 68.003% and 86.566%, respectively under the above-mentioned metrics.

The top performing classifier, i.e., LGBM, is deployed in the Ryu controller, where it can detect and mitigate the attacks in the live traffic. In future, the same thing can be done in the multicontroller environment. The second work might be to do the same thing with the cooperative approach, where the preliminary attacks are detected in the data plane, and the controller decides on the final attack detection.

References

[1] Tariq Hussain et al., "Improving Source Location Privacy in Social Internet of Things using a Hybrid Phantom Routing Technique," *Computers and Security*, vol. 123, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Preetinder Singh Brar et al., "Using Modified Technology Acceptance Model to Evaluate the Adoption of a Proposed IoT-Based Indoor Disaster Management Software Tool by Rescue Workers," *Sensors*, vol. 22, no. 5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Huseyin Polat, Onur Polat, and Aydin Cetin, "Detecting Ddos Attacks in Software-Defined Networks through Feature Selection Methods and Machine Learning Models," *Sustainability*, vol. 12, no. 3, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jian Su et al., "Redundant Rule Detection for Software-Defined Networking," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 6, pp. 2735–2751, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Jagmeet Kaur et al., "Packet Optimization of Software Defined Network using Lion Optimization," *Computers, Materials and Continua*, vol. 69, no. 2, pp. 2617–2633, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Alexandru L. Stancu et al., "A Comparison between Several Software Defined Networking Controllers," *12th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, TELSIKS*, pp. 223–226, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[7] Lusani Mamushiane, Albert Lysko, and Sabelo Dlamini, "A Comparative Evaluation of the Performance of Popular SDN Controllers," *IFIP Wireless Days*, pp. 54–59, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] K. Kaur, S. Kaur, and V. Gupta, "Performance Analysis of Python-based Openflow Controllers," *IET Conference Publications*, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[9] Safaa Mahrach, and Abdelkrim Haqiq, "DDoS Flooding Attack Mitigation in Software Defined Networks," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 693–700, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Nisharani Meti, D.G. Narayan, and V.P. Baligar, "Detection of Distributed Denial of Service Attacks using Machine Learning Algorithms in Software Defined Networks," *International Conference on Advances in Computing, Communications and Informatics*, pp. 1366–1371, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] Mimi M Cherian, and Satishkumar L. Varma, "Mitigation of DDOS and MiTM Attacks using Belief Based Secure Correlation Approach in SDN-Based IoT Networks," *International Journal of Computer Network and Information Security*, vol. 14, no. 1, pp. 52–68, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ranjit Panigrahi et al., "Performance Assessment of Supervised Classifiers for Designing Intrusion Detection Systems: A Comprehensive Review and Recommendations for Future Research," *Mathematics*, vol. 9, no. 6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Nisha Ahuja et al., "Automated DDoS Attack Detection in Software Defined Networking," *Journal of Network and Computer Applications*, vol. 187, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Nagarathna Ravi, and S. Mercy Shalinie, "Learning-Driven Detection and Mitigation of DDoS Attacks in IoT via SDN-Cloud Architecture," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3559–3570, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] I.Lakshmi, "Security Analysis in Internet of Things using DDoS Mechanisms," *SSRG International Journal of Mobile Computing and Application*, vol. 6, no. 1, pp. 19-24, 2019. [CrossRef] [Publisher Link]

[16] Richakunal Sharma, and Nalini Kant Joshi, "Security and Privacy Problems in Cloud Computing," *International Journal of Computer and Organization Trends*, vol. 9, no. 4, pp. 30-39, 2019. [Publisher Link]

[17] Afsaneh Banitalebi Dehkordi, and Mohammadreza Soltanaghaei, "A Novel Distributed Denial of Service (DDoS) Detection Method in Software Defined Networks," *IEEE Transactions on Industry Applications*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Liang Tan et al., "A New Framework for DDoS Attack Detection and Defense in SDN Environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] M.V Sangeetha, and J Bhavithra, "Applying Packet Score Technique in SDN for DDoS Attack Detection," *SSRG International Journal of Computer Science and Engineering*, vol. 5, no. 6, pp. 20-24, 2018. [CrossRef] [Publisher Link]

[20] Animesh Kumar, Sandip Dutta, and Prashant Pranav, "A Comparative Study of DDoS Attack in Cloud Computing Environment," *SSRG International Journal of Electronics and Communication Engineering*, vol. 10, no. 7, pp. 87-96, 2023. [CrossRef] [Publisher Link]

[21] Raveendranadh Bokka, and Tamilselvan Sadasivam, "Securing IoT Networks: RPL Attack Detection with Deep Learning GRU Networks," *International Journal of Recent Engineering Science*, vol. 10, no. 2, pp. 13-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Zhuo Chen et al., "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud," *IEEE International Conference on Big Data and Smart Computing, BigComp*, pp. 251–256, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[23] Quamar Niyaz, Weiqing Sun, and Ahmad Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," *ICST Transactions on Security and Safety*, vol. 4, no. 12, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[24] B.V. Karan, D.G. Narayan, and P.S. Hiremath, "Detection of DDoS Attacks in Software Defined Networks," *Proceedings 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS*, pp. 265–270, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[25] Prashant Kumar et al., "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[26] K. Giotis et al., "Combining Openflow and Sflow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments," *Computer Networks*, vol. 62, pp. 122–136, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[27] Yang Wang et al., "SGS: Safe-Guard Scheme for Protecting Control Plane Against DDoS Attacks in Software-Defined Networking," *IEEE Access*, vol. 7, pp. 34699–34710, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[28] Shanshan Yu et al., "A Cooperative DDoS Attack Detection Scheme based on Entropy and Ensemble Learning in SDN," *Eurasip Journal on Wireless Communications and Networking*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[29] Jin Ye et al., "A DDoS Attack Detection Method Based on SVM in Software Defined Network," *Security and Communication Networks*, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[30] Mert Özçelik, Niaz Chalabianloo, and Gürkan Gür, "Software-Defined Edge Defense Against IoT-Based DDoS," *IEEE CIT 17th IEEE International Conference on Computer and Information Technology*, pp. 308–313, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[31] Kshira Sagar Sahoo et al., "An Evolutionary SVM Model for DDoS Attack Detection in Software Defined Networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[32] Myo Myint Oo et al., "Advanced Support Vector Machine-(ASVM-) based Detection for Distributed Denial of Service (DDoS) attack on Software Defined Networking (SDN)," *Journal of Computer Networks and Communications*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[33] Muhammad Arslan Sarwar et al., "FlowJustifier: An optimized Trust-Based Request Prioritization Approach for Mitigation of SDN Controller DDoS Attacks in the IoT Paradigm," *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, pp. 1-9, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[34] Kübra Kalkan et al., "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[35] Manal Abdullah et al., "Enhanced Intrusion Detection System using Feature Selection Method and Ensemble Learning Algorithms," *International Journal of Computer Science and Information Security*, vol. 16, no. 2, pp. 48–55, 2018. [Google Scholar] [Publisher Link]

[36] Meng Wang, Yiqin Lu, and Jiancheng Qin, "A Dynamic MLP-based DDoS Attack Detection Method using Feature Selection and Feedback," *Computers and Security*, vol. 88, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[37] Irfan Ullah Khan et al., "A Proactive Attack Detection for Heating, Ventilation, and Air Conditioning (HVAC) System Using Explainable Extreme Gradient Boosting Model (XGBoost)," *Sensors*, vol. 22, no. 23, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[38] S.R. Khonde, and V. Ulagamuthalvi, "Ensemble and Feature Selection-based Intrusion Detection System for Multi-Attack Environment," *5th International Conference on Computing, Communication and Security (ICCCS)*, pp. 1-8, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[39] Guolin Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems*, pp. 3147–3155, 2017. [Google Scholar] [Publisher Link]

[40] Shi Dong, and Mudar Sarem, "DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks," *IEEE Access*, vol. 8, pp. 5039–5048, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[41] Gholamreza Farahani, "Black Hole Attack Detection Using K-Nearest Neighbor Algorithm and Reputation Calculation in Mobile Ad Hoc Networks," *Security and Communication Networks*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[42] Amit V Kachavimath, Shubhangeni Vijay Nazare, and Sheetal S Akki, "Distributed Denial of Service Attack Detection using Naïve Bayes and K-Nearest Neighbor for Network Forensics," *2nd International Conference on Innovative Mechanisms for Industry Applications*, pp. 711–717, 2020. [CrossRef] [Google Scholar] [Publisher Link].

[43] Taqwa Ahmed Alhaj et al., "Feature Selection using Information Gain for Improved Structural-Based Alert Correlation," *PLoS ONE*, vol. 11, no. 11, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[44] Pullagura Indira Priyadarsini, "ABC-BSRF: Artificial Bee Colony and Borderline-SMOTE RF Algorithm for Intrusion Detection System on Data Imbalanced Problem," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 56, pp. 15–29, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[45] Zhe Wang, Chenjie Cao, and Yujin Zhu, "Entropy and Confidence-Based Undersampling Boosting Random Forests for Imbalanced Problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5178–5191, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[46] Ravindra Kumar Chouhan, Mithilesh Atulkar, and Naresh Kumar Nagwani, "Performance Comparison of Ryu and Floodlight Controllers in Different SDN Topologies," *1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing and Communication Engineering, ICATIECE*, pp. 188–191, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[47] Alessio Botta, Alberto Dainotti, and Antonio Pescapé, "A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[48] Salma Elhag et al., "A Multi-Objective Evolutionary Fuzzy System to Obtain a Broad and Accurate Set of Solutions in Intrusion Detection Systems," *Soft Computing*, vol. 23, pp. 1321–1336, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[49] Faisal Hussain et al., "IoT DoS and DDoS Attack Detection using ResNet," *IEEE 23rd International Multitopic Conference*, pp. 1-6, 2020. [CrossRef] [Google Scholar] [Publisher Link]