

Anomaly-Based Intrusion Detection System in the Internet of Things Using Machine Learning

Laboni Sarkar¹, Dr. Ranjan Kumar Mondal²

¹Research Scholar, CSE, Swami Vivekananda University,
Barrackpore, WB, India

²Assistant Professor, CSE, Swami Vivekananda University,
Barrackpore, WB, India

labonisarkar.sona@gmail.com, ranjankm@svu.ac.in

Abstract

The Internet of Things (IoT) refers to a new category of computing that links the Internet primarily with tangible items. Belonging to different domains With IoT involvement increasing in modern technological advancements, attacks and anomaly detection are also concerns in the IoT domain. This survey provides a structured and comprehensive anomaly detection algorithm evaluation, a testament to our commitment to resolving IoT security issues. In this paper, we recommend assessing the workings of several machine learning models used in this study to predict attacks and adequately capture anomalies in IoT systems. We choose datasets from the Distributed Smart Space Orchestration System (DS2OS). We chose thirteen well-established algorithms from the sklearn library to build the final Models and compare results. The statistical evaluation metric compares the model's performance and detects anomalies in the IoT system. The final result of the proposed work is an evaluation of the performance and accuracy of the machine learning models, demonstrating the thoroughness of our research.

1. Introduction

The Internet of Things (IoT) is an expanding network of intelligent items. It alludes to the idea that physical objects may communicate with one another through the exchange

of information. Authentication and encryption are necessary to handle the difficulty of providing secure communication in the IoT network. Although authentication and encryption help safeguard the IoT network, they are not impervious to cyberattacks.

One tool used to keep an eye out for harmful activity on a specific node and network is the Intrusion Detection System (IDS). IDS can examine and look into user and machine behavior, find malicious network activity, and find signatures of known attacks. IDS monitors networks and nodes, finds different network intrusions, and notifies users when an intrusion is discovered. By raising an alarm before the attackers launch their attack, the IDS functions as a network observer or alarm, protecting the systems from harm. Internal attacks are malevolent acts on a system from within the organization or from an insider. The other is an attack against one or many compromised nodes that are part of the network, while the external network initiated parties by an external network [1]; for the previous few years, research in the field of intrusion detection systems has increased rapidly. Several new papers have been published on developing the intrusion detection algorithm. The detection algorithm can be classified into three categories: supervised, semi-supervised, and unsupervised learning by their training methods. A full or partial role is played in the learning process for supervised and semi-supervised learning methods. Still, in the case of unsupervised learning, the machine is trained by a total algorithm without any human intervention. The type of this detection algorithm can be divided further by the detection method. In anomaly detection, the main goal of this review is to detect the abnormal behavior of the system with a previously trained machine learning algorithm. In contrast, in the case of the signature-based method, the previously known attack signature is used to detect and train the algorithm. The systems are then provided with an anomaly detection method to find the outliers. We can divide the outliers into three main categories: 1. Global Outliers, 2. Contextual Outliers 3. Collective Outliers were discussed in the review. We also collected various datasets used to train the algorithms, which are extensively reviewed in this paper. In the future, we will

train some algorithms in those datasets to compare intrusion detection and discover the best algorithm for detecting and preventing attacks in IoT.

Generic IoT Architecture:

The Internet of Things An abstraction consisting of several hierarchical levels is a helpful way to perceive architectures. They are formed of three layers: application, network, and perception, which are at the primary levels of abstraction. Although devices inside a layer may employ diverse technologies, each layer has its own unique set of technologies. Because of this, managing them is a challenging and intricate task. To overcome this difficulty, a middleware layer that hides the underlying implementation details is occasionally introduced to handle various kinds of service.

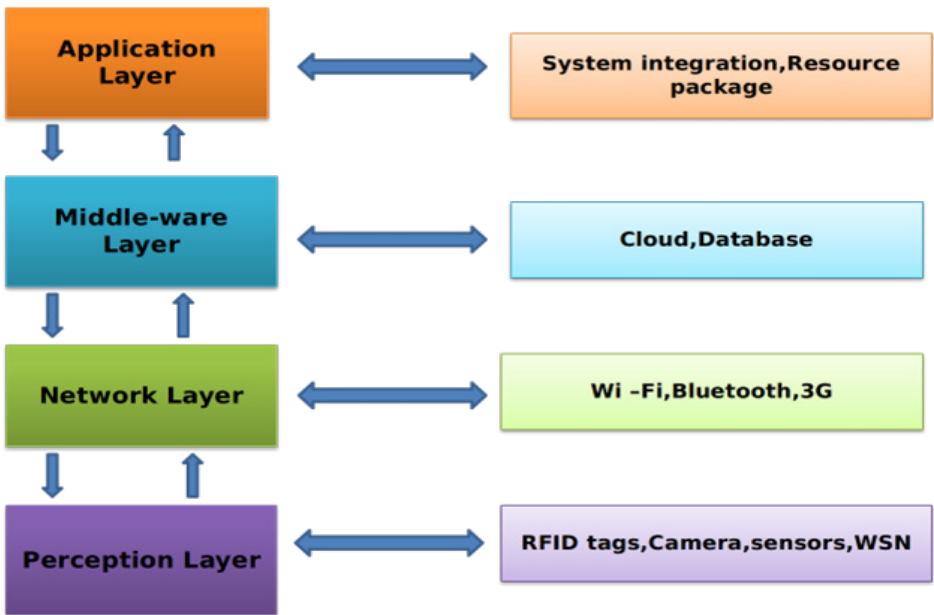


Fig 1. Figure of generic IOT architecture[1].

Layer name and work:

Table 1. Layer-wise name and work:

Application Layer	Middleware Layer	Network Layer	Perception Layer
The Application Layer is a vital component of the Internet of Things, as it combines the social aspect of IoT with industrial requirements to achieve widespread intellectualization. This high-quality support is provided to the end users and manages and processes data from the middleware layer. However, the application Layer also presents unique security challenges, particularly when handling sensitive data. Unauthorized access, malicious data alteration, and authorization lifetime are among the primary issues that need to be addressed to ensure the security of the IoT network.	The middleware layer processes and stores data, connects systems to the cloud and databases, and collects data from the Network layer. The primary concerns at the middleware layer that impact the application layer's quality of service are database security and cloud security[5-6].	This layer handles the connection to the IoT infrastructure. It also gathers and sends data from the perception layer to the top layer. The primary technologies include Bluetooth, WiFi, 3G, ZigBee, etc. The transmission method can be wired or wireless[6-7].	The perception layer converts target data into digital signals while attempting to detect things and gather target information. This layer's primary technologies include wireless sensor networks (WSN), cameras, RFID tags, and sensors – energy and processing power impact perception layer technology[8].

2. Literature Survey:

1. Use of Anomaly Detection in IOT:

- Identify a possible attack before it happens. The anomaly detection algorithm contentiously checks the outliers of the data point to identify a potential attack on the server or network and prevent damage.
- Identify possible unauthorized access to the network in IOT devices through various algorithms and network models so that only users with inclusive authorization can access the network.
- Identify an intruder using a machine learning model and apply various intrusion prevention methods to prevent the system from sustaining unwanted damage.

- In the healthcare industry, an Anomaly detection system can be used to monitor patient health conditions like heart rate, BMR, and temperature so that the system can identify any anomaly behavior in real-time.

- The anomaly detection algorithm is used in all outlier detection in a wide range of datasets, not only network attack datasets. Anomaly detection can be used in time series data, stock exchange stock price prediction, and cancer cell detection using pathology images by convolution neural network algorithm.

3. Attacks in IOT system:

Table 2. Layer-wise attack name and their description:

Layer Name	Attack Name	Description	References
Application Layer	Code Injection	An attack involves injecting malicious programming code into the target machine and exploiting software glitches. It can be used for various purposes, such as stealing data, gaining systematic control, and propagating worms.	[7,8]
	Buffer Overflow	Attack means a breach of code or data buffer bounds by exploiting program vulnerabilities. To illustrate, in a single write operation, the Attacker writes hundreds of millions of data symbols to a particular region, enhancing the series beyond its intensity – a predetermined area of residence.	[7,8,9]
	Phishing Attack	It involves an attacker creating and becoming somebody else with a legitimate credential to acquire the user's sensitive data, such as passwords, credit card information, etc.	[7,10]
Middleware Layer	SQL Injection Attack	A poorly built application might be open to these assaults since it incorporates SQL statements into the input data. Attackers use these SQL statements to perform reading, writing, and deleting activities. In addition to obtaining the user's personal information, this type of assault poses a risk to the database system as a whole. SQL injection attacks on Web applications result in pages displaying data that differs from what is there.	[11,12]
	Signature Wrapping Attack	The attacker changes the eavesdropped messages he sends when the Cloud does not invalidate the signature. An XML signature ensures the integrity of the service. An attacker can also execute arbitrary commands and operations as Legitimate users.	[13,14]
	Flooding Attack in Cloud	In particular, one type of denial of service is the cloud. So, constantly attacking. Depleting requests to a service in the cloud affects the service quality due to the resources.	[15-17]
	Cloud Malware Injection	Attackers can infiltrate a hostile service instance or virtual machine into the cloud, alter data, take control, and run malicious code.	[15-17]

Network Layer	DOS Attack	A denial-of-service (DOS) attack on a network is carried out by overloading the target with requests, which causes a significant volume of network traffic. Furthermore, a distributed denial-of-service (DDOS) assault can be launched against one or more targets by combining many computers as an attack platform.	[7,18,19]
	Sybil Attack	Redundancy is defeated when a system node gives a victim node several identities, enabling the victim node to operate more than once. Because the attacker in a wireless sensor network (WSN) assumes numerous identities, information from the victim node may be sent through a compromised node, resulting in a longer routing distance.	[7,18,20]
	Sinkhole Attack	When an attacker uses a compromised node to draw data flow from neighboring nodes, the system is tricked into believing the data has arrived at its intended location. An attacker may also use a rogue node in a WSN to draw in network traffic, after which the sensor data may be used however they choose.	[7,18]
	Man-in-the-Middle Attack	This kind of assault happens in real time between two victim nodes that are in communication with one another. The attackers also engage in data manipulation. In WSN or wireless sensor networks, once the attacker gets an opportunity, they will definitely. If a person wants to join the network, one can launch an attack or use the network without paying for anything.	[21,22,7]
Perception Layer	Unauthorized Access to the Tags	Unauthorized attackers can readily access tags and alter data because RFID systems lack efficient authentication methods. Once they can access the network, attackers can use it or conduct assaults without paying.	[23]
	Tag Cloning	RFID tag cloning is one of the effective attacks. For this, the attacker has to get the relevant. Information directly from its deployment environment through reverse engineering. In this case, the controller can't identify the original RFID vs Cloned RFID	[7]
	RF Jamming	The RF signal sent from the attacker interferes with the communication between the legitimate tag and the reader. In this way, an RFID tag is used to interfere with all signals in the range and prevent communication with all tags.	[24]

4. Intrusion Detection Systems (Ids):

An intrusion is any disapproved or unauthorized activity in any network or system. A Breakthrough Detection System (IDS) is a collection of resources, tools, and algorithms that make it possible to find, assess, and report intrusions. Rather than being a stand-alone security solution, intrusion detection is typically a component of a broader protective [24] system placed around a system or device. Anomaly, abuse, signature-based detection, and specification-based detection are the three functional categories into which IDS are divided.

- **Anomaly-based detection:** This depends on the models of statistical behavior. The agents' normal operations are identified, and a specific deviation from normal behavior is recognized as an anomaly. The disadvantage of this type of discovery is that regular profiles must be updated periodically because network behavior can change quickly. This can increase the load on the sensor nodes with limited resources.
- **Misuse-based (signature-based or rule-based) detection:** Signatures (profiles) for pre-known attacks are created and used as a reference for detecting future attacks. For example, a typical example of signing would be a brute force password attack where the attacker uses a login attempt by assuming the password and failing three attempts of password generation. The advantage of this type of discovery is that it can detect known attacks accurately and proficiently; therefore, they have a low false positive rate.
- **Specification-based detection:** A set of specifications and constraints describing the accurate operation of a program or protocol is defined. Then, the program's execution concerning the specified requirements and constraints is monitored.

4.1. Definition of anomaly detection:

The data mining process called anomaly detection, also known as outlier analysis, finds observations, occurrences, or data points that don't match the typical patterns in a dataset. Unusual data may point to important events, like a technological malfunction, or possible openings, like a shift in customer behavior. Automating anomaly detection with machine learning is becoming increasingly common.

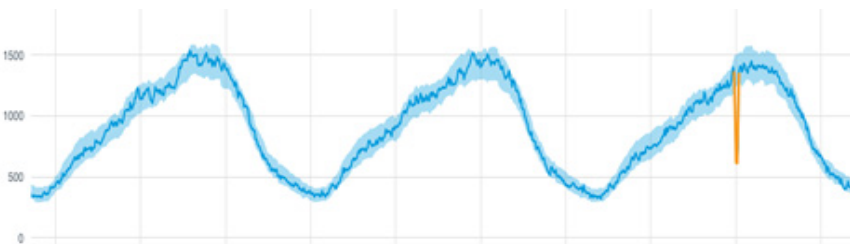


Fig 2. Figure of Anomaly detection.[24]

This graph shows the anomalous drop in time series data. The yellow part is the anomaly. The usual range for this metric is on the line, which extends far below the blue-colored area.

4.2. The three different types of time series anomalies:

To maximize the value of created insights, one must comprehend the kinds of outliers that an anomaly detection system may discover. When your anomaly detection system alerts you to a problem or opportunity, you risk making incorrect choices if you don't know what the opposition is like.

Global, contextual, and collective outliers are the three basic categories into which abnormalities in your company data often fall.

4.2.1. Global outliers:

Also known as point anomalies, these outliers exist far outside the entirety of a data set.

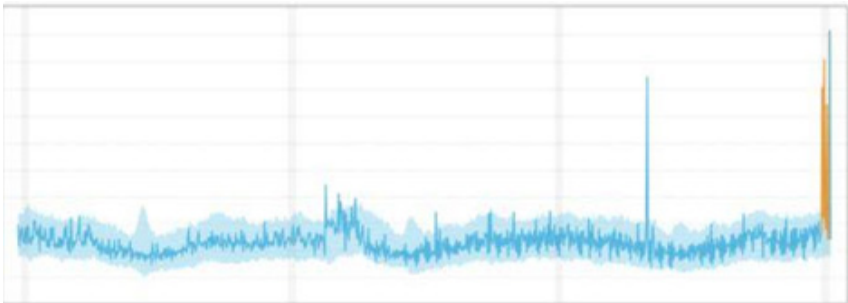


Fig 3. Figure of Global outliers[24]

4.2.2. Contextual outliers:

These anomalies are also called conditional outliers, with values that deviate significantly from other data points in the same context. Anomalies in the context of one data set may not be anomalies in another. These extreme values are expected in time series data because these data sets record specific quantities in a given period. The value is in global forecasts but may appear abnormal in a particular seasonal data pattern.

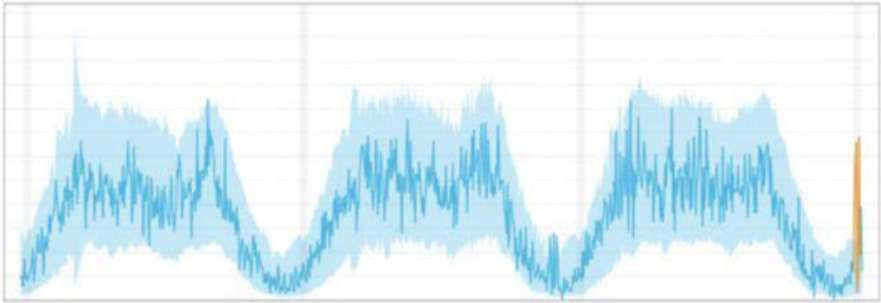


Fig 4. Figure of Contextual outliers.[24]

4.2.3. Collective outliers:

Since it is a subset of data points, it resides inside an abnormal set of the entire data set with values called collective outliers. In this category, individual values are not uncommon worldwide or contextual. You start to see these types of outliers when looking at different time series together. Individual behavior may not deviate from the normal range in a specific time series data set. However, when combined with another set of time series data, the most critical anomalies become evident.

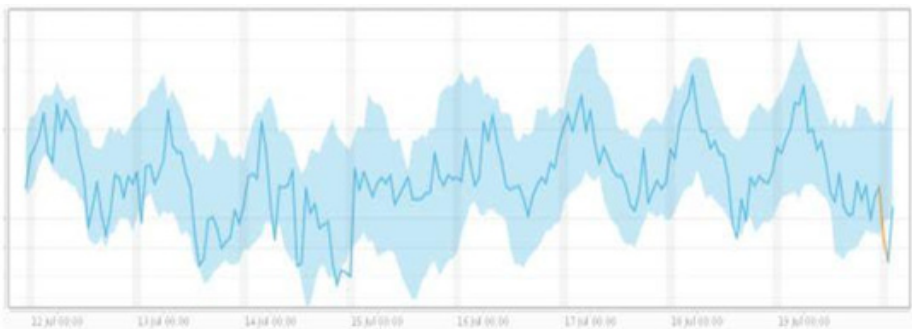


Fig 5. Figure of Collective Outliers [24]

4.3. Categorization of Anomaly Detection:

Several configurations are conceivable when discussing anomaly detection. In essence, the configuration to be employed is determined by the labels in the dataset; as shown, three primary categories may be distinguished.

4.3.1. Supervised Anomaly Detection:

Explains the configuration in which the training and test data sets are fully labeled. It is possible to learn and then apply a standard classifier. The main difference between this situation and standard pattern recognition is that classes are usually highly imbalanced. Therefore, not every classification method is ideal for this purpose. Decision trees such as C4.5 [25] are not well suited to handle imbalanced data; on the other hand, Support Vector Machines (SVM) [26] and Artificial Neural Networks (ANN) [27] ought to function better in this regard. However, this arrangement is not meaningful because anomalies are assumed to be recognized and appropriately tagged. Anomalies for many applications could arise as novelties on their own during the testing process, or they might be unknown beforehand.

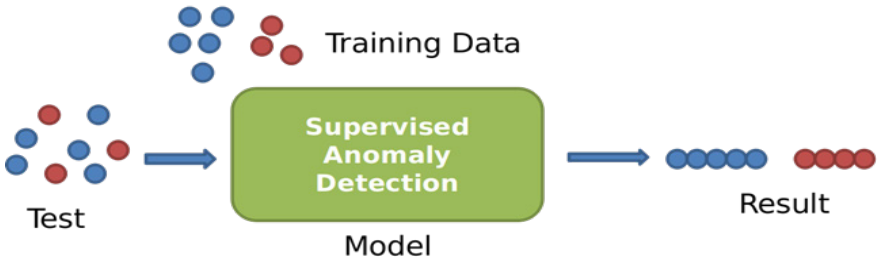


Fig 6. Figure of Supervised anomaly detection [25].

4.3.2. Semi-supervised Anomaly Detection:

Test and training datasets are also used; training data comprises average, anomaly-free data. The fundamental principle is that after learning a model of the typical class, anomalies may be found by departing from that model. "One-class" categorization is another term for this concept [28]. Well-known methods include One-class SVMs [29] and auto encoders [30]. The probability density function of the regular classes can be modeled using any density estimation technique, such as Kernel Density Estimation [32] or Gaussian Mixture Models [31].

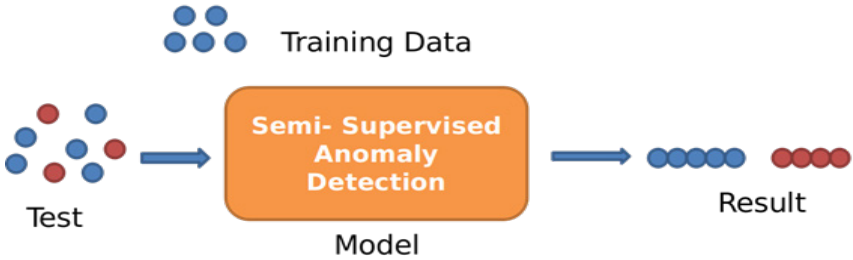


Fig 7. Figure of Semi-supervised anomaly detection [25].

4.3.3. Unsupervised Anomaly Detection:

This configuration, which is the most versatile, does not need labeling. Moreover, a training dataset and a test dataset are identical. An unsupervised anomaly detection system is supposed to score the data only based on the intrinsic qualities of the dataset. Densities or distances are typically utilized to determine what is expected and what is an outlier. This article covers only one unsupervised anomaly detection setup [33].

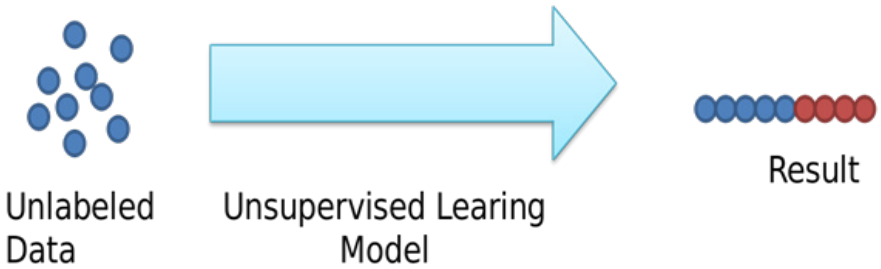


Fig 8. Figure of Un-supervised anomaly detection [25]

5. Algorithm used in Anomaly Detection:

In this paper, we have used thirteen well-established algorithms to detect anomalies. All the thirteen algorithms used from the sklearn python library are used for this study to evaluate the comparison between various groups of attacks described below:

5.1. K Nearest Neighbor (K neighbors Classifier): The Neighbors-based categorization relies on the instance-based learning or non-generalizing learning principle, which states

that instances of the training data are merely stored rather than an attempt made to build a general internal model. In this classification technique, a majority vote is done to decide the neighbors of all the points: a query point is assigned to the data class with the highest more than a reasonable number of representatives in the nearest neighbors of the end. The K Neighbors Classifier uses an integer k that the user specifies to execute learning based on each query point's k nearest neighbors [35].

5.2. Gaussian Naive Bayes (GaussianNB): One classifier model exists. In addition to the Gaussian Naive Bayes, there are two more naïve bayes models: Multinomial and Bernoulli. However, the Gaussian Naive Bayes model is the most often used and least complicated. Gaussian NB implements the Gaussian Naive Bayes technique for classification. It is assumed that the features have a Gaussian probability [36].

5.3. Bernoulli Naive Bayes (BernoulliNB): A probabilistic model built on the fundamental ideas of the probability theory's Bayes theorem is called a naïve Bayes classifier. For data distributed according to multivariate Bernoulli distributions—when there may be many features, but each is considered a binary-valued (Bernoulli, boolean) variable—BernoulliNB implements the naïve Bayes training and classification methods. Because of this, samples submitted to this class must be expressed as binary-valued feature vectors; otherwise, By using the `binarize` parameter, a BernoulliNB instance may binarize its input.[36][41].

5.4. Extra Tree Classifier: Decision trees serve as the foundation for the ensemble learning technique known as the Extra Trees Classifier. Like Random Forest, Extra Trees Classifier uses randomization to reduce overfitting and overlearning from the data by assigning random numbers to particular decisions and subsets of data. The two main differences between Extra Trees and Random Forest are that Trees create multiple trees and split nodes with randomly selected subsets of features. At the same time, the latter splits nodes into random splits rather than optimal splits[37][38]. The former does not bootstrap observations, i.e., samples without replacement.

5.5. Decision Tree: In a decision tree, each node represents a feature (attribute), a decision (rule) is defined by each link (branch), and a result (categorical or continuous value) is represented by each leaf. Often, a DT is started from one node and then splits off into many other complex options) [39]. All of these results produced more nodes that split off into other instances. From then, it took on the form of a tree, or more specifically, a structure akin to a flowchart. Consider a binary tree in which a parent node splits into a left child and a right child [39].

5.6. Random Forest Classifier: The random forest classifier is built upon a random sample of the entire training data set to produce the set of decision trees. The ultimate class of the test item is subsequently determined by adding the votes from several decision trees. A random forest is created when several decision trees combine to form an ensemble, and it makes predictions by averaging the forecasts of its trees. It often offers substantially higher prediction accuracy than a single decision tree. A forest generally appears sturdy with more trees [39][44].

5.7. Logistic Regression: This is frequently used to calculate the likelihood that an instance belongs to a specific class (Logit Regression). When examining categorical response variables, discriminant analysis and logistic regression compete [40]. Assume that the two results of a binary variable have the numerical values 0 and 1. A negative answer is often represented by a zero, and a positive response by a one. The percentage of affirmative answers will equal the variable's mean. The probability of a result of 0 is $1-p$, where p is the fraction of observations with an outcome of 1. The chances are defined as the ratio $p/(1-p)$, and the logit, or log odds, is the logarithm of the odds. Mathematically, the Logit transformation is written as :

$$l = \text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

5.8. SGD Classifier: This estimator applies stochastic gradient descent (SGD) learning to regularized linear models. Precisely, each sample's gradient of the loss is calculated, and the model is updated using a decreasing strength schedule, also known as the learning rate. Minibatch (online/out-of-core) learning using the partial_fit approach is made possible by SGD. The data should have zero mean and unit variance to get the best results using the default learning rate schedule. A straightforward stochastic gradient descent learning procedure that supports various classification penalties and loss functions is implemented by the class SGD Classifier. The SGDClassifier's decision boundary, which is comparable to a linear SVM and was trained using the hinge loss, is located below this [42].

5.9. Perceptron: A perceptron is a type of single-layer neural network. In contrast, neural networks are multi-layer perceptrons. (A perceptron is a single-layer neural network, and neural networks exist in many layers). Are multi-layer perceptrons. A binary linear classifier is the perceptron. It is employed in supervised learning as well. Classifying the provided input data is helpful. Like a single processing unit, a perceptron may create a linear border for categorization or decision-making between several classes. Finding a line in the shape is the responsibility of perception: $WX + b = 0$, where W is the coefficient vector corresponding to each X (e.g., $W = [w_1, w_2]$), b is a scalar bias value applied to the line equation [43], and X is the dimension size vector that creates the variables (e.g., $X = [x_1, x_2]$).

5.10. Multilayer Perceptron (MLP Classifier): Multilayer Perceptron generates weights for input-output pairs and then learns how to represent these inputs and outputs as related (or dependent). Training entails modifying the model's parameters, weights, and biases to reduce error. They are made about error via backpropagation, changing the weight and bias. The error can be assessed in several ways, including root mean squared error. (RMSE)

Thus, one way to conceptualize an MLP would be as a deep artificial neural network. It is made up of many perceptrons. The actual computational engine of the MLP is an arbitrary

number of hidden layers interposed between a layer that receives the inputs. It consists of a signal and an output layer that determines, based on the input, whether it should be judged or predicted [43]

5.11. Gradient Boosting Classifier: Gradient Boosting ensures that prior mistakes are fixed by gradually adding the predictors that were underfitted forecasts to the ensemble. Gradient Boosting is an extension of Tree Boosting or Gradient Boosted Decision Trees (GBDT). One of the loss functions, let's say it, is any differentiable loss function. Web search ranking and ecology are only two domains where GBDT is a versatile and off-the-shelf approach that is accurate for both Challenges in regression and classification. Three components are included in gradient boosting: A loss function that has to be refined, Not very good at making predictions, and A weak learner additive model to reduce the loss function [45].

5.12. Ridge Classification: The linear regression method was adapted for use in this approach. In ridge regression, a penalty equal to the square of the coefficient magnitudes is added to the cost function to change it. The coefficients (w) are constrained using ridge regression. The optimization function is punished if the coefficients assume high values since the penalty term (λ) regularizes the coefficients. Thus, ridge regression reduces multi-co linearity and model complexity by shrinking the coefficients. Therefore, the model will resemble a linear regression model when the restriction on the features is lowered (low λ) [46].

5.13. Linear SVC: A support vector machine constructs a hyper-plane or a set of hyper-planes in an infinite or huge feature space for applications such as regression and classification. It makes the closest training data determine that the hyperplane has the most significant distance to the nearest training points. Each class known as the functional margin—would accomplish a successful separation since, generally speaking, a more substantial margin corresponds to a minor classifier generalization error. The decision function for a linearly separable issue with three samples, or “support vectors,” on the margin borders is depicted in the figure below. Another similar multi-class approach, known as the

“multi-class SVM,” was developed by Crammer and Singer and is likewise implemented by linear SVC [47].

- **Evaluation criteria:**

We have chosen evaluation metrics like the Confusion matrix to assess the models’ performance. All learning algorithms are tested on these evaluation metrics.

Confusion matrix: A confusion matrix is a table often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the actual values are known. It is a table for describing the performance of a classification model on a set of test data with actual values known. It allows for easy discrimination and confusion among classes. It can be used to summarize the performance of some simulation maneuvers. We can use this classification model to compare the actual and predicted values. Let C_a be any class out of the thirteen classes

◇ **True positives (TP):** In these cases, we predicted yes (they have the attack). And they do have the attack.

◇ **True negatives (TN):** We predicted a normal/different attack, and they don’t have the attack.

◇ **False positives (FP):** We predicted yes, but they don’t have the attack. (Also known as a “Type I error.”)

◇ **False negatives (FN):** We predicted no, but they have the attack. (Also known as a “Type II error.”)

Accuracy: Accuracy is also used to measure how well a classification model test correctly identifies.

- **Accuracy = $(TP + TN)/(TP + TN + FP + FN)$**

Precision: Precision means two or more values of the measurements are close to each other. The value of precision differs because of the observational error. Precision is used to find the consistency or reproducibility of the measurement.

- **Precision: $TP/(TP + FP)$**

Recall: Recall is the actual positive rate, the number of positives in the model claims relative to total positives in the data.

- **Recall = $TP/(TP + FN)$**

F1 Score: A model's performance can also be measured by the F1 score. It is weighted the precision and recall averaged over a model.

- **F1 Score = $2*TP/(2*TP+FP+FN)$**

ROC Curve: Another standard tool for binary classifiers is the receiver operating characteristic (ROC) curve. It resembles the precision/recall curve quite a bit. However, the ROC curve compares the actual positive rate, which is another term for recall, against the false positive rate (FPR) rather than accuracy vs recall. The fraction of negative cases mistakenly categorized as positive is known as the FPR. It is equal to 1 - the ratio of negative situations that are accurately classified as hazardous or the actual negative rate (TNR). Another name for the TNR is specificity. As a result, the ROC curve plots recall (sensitivity) against specificity.

● **FPR = Number of False Positive Samples / Total Sample**

● **TPR = Number of True Positive Samples / Total Sample**

- **Dataset Collection:**

The data set used in the analysis is an open-source dataset obtained from Kaggle [33] and provided by Pahl et al. [34]. They have simulated an IoT context using the Distributed Smart Space Orchestration System. The latter is generated through Dataset Synthesis to Other Sources (DS2OS). The dataset has samples of 357,952 and features of 13. It has 347,935 Normal and 10,017 anomalous data and eight classified classes. Features "Accessed There are 148 rows missing data in the "Node Type" column, and in the "Value," there are 2050 missing data.

Table 3: The Features of data are described in the table:

Serial No.	Features	Data types
1.	Source ID	Nominal

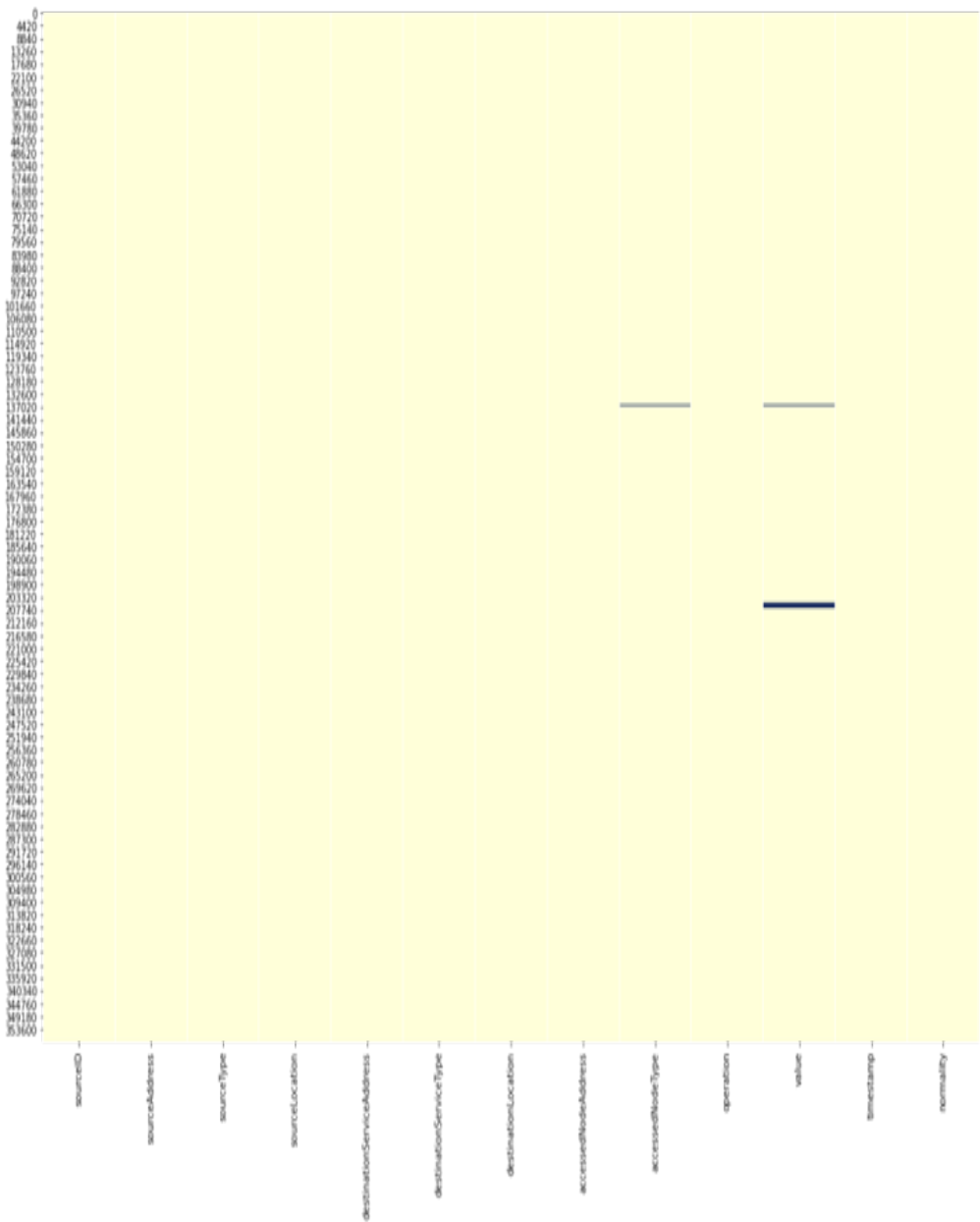
2.	Source Address	Nominal
3.	Source Location	Nominal
4.	Destination Service Address	Nominal
5.	Destination Service Type	Nominal
6.	Destination Location	Nominal
7.	Accessed Node Address	Nominal
8.	Accessed Node Type	Nominal
9.	Operation	Nominal
10.	Value	Continuous
11.	Timestamp	Nominal
12.	Normality	Discrete

Table 4 provides a clear picture of the distribution of several attacks and anomalous behaviors throughout the whole data:

Attacks	Frequency Count	Percentage of Total Data	Percentage of Anomalies
Denial of Service	5780	01.61 %	57.70%
Data Type Probing	342	00.09%	03.41%
Malicious Control	889	00.24%	08.87%
Malicious Operation	805	00.22%	08.03%
Scan	1547	00.43%	15.44%
Spying	532	00.14%	05.31%
Wrong Setup	122	00.03%	01.21%

● **Data Preprocessing:**

Our first goal in this study is exploratory data analysis and understanding of data sets. We used Python code to visualize the missing values of the datasets using a heat map. The black spot in the yellow background shows column-wise all feature columns missing values. Most sklearn classifiers cannot handle the missing data, so our next goal is to remove missing values from the data. We have used a feed-forward fill method to fill the missing values. After that, we again visualize the data using a heat map of disappeared values; the heat map shows no missing values on the datasets. The results are visualized in Figure 9.



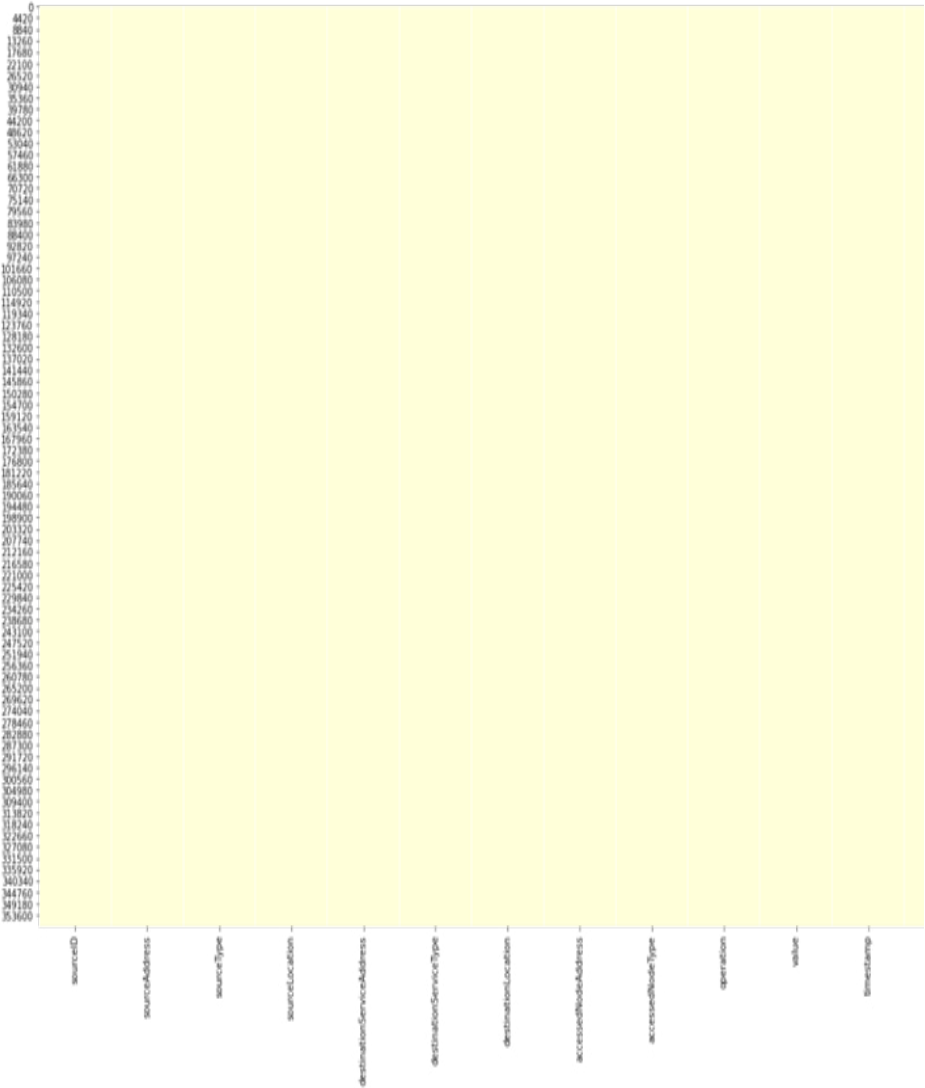


Fig 9: Missing values in the Data
Unexpected values are either removed or converted meaningfully according to previous data values. For example,

False", "True," "Twenty," and "none" in the "Value" feature are replaced by meaningful values "0.0", "1.0", "20.0," and "0.0", respectively. To understand the number of attacks in the system, as an exploratory data analysis, all the attacked data are visualized as a bar plot, as shown in Figure 10 and summarized in Table 5.

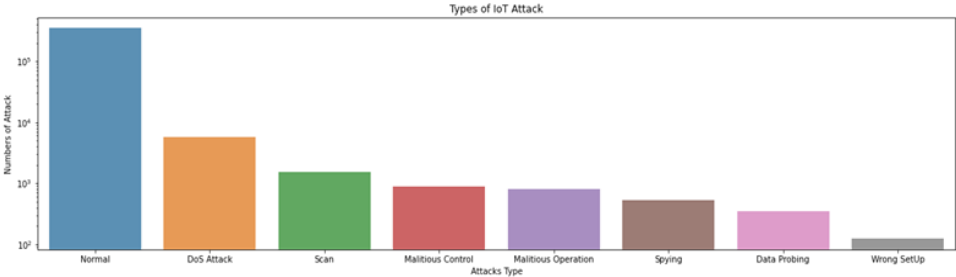
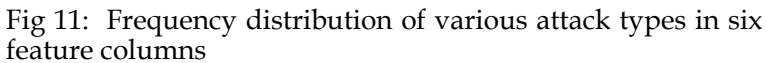


Fig 10: Frequency distribution of all the IoT attacks.

Table 5: Summarization of frequency distribution:

Attack Type	Normal	DoS Attack	Scan	Malicious Control	Malicious Operation	Spying	Data Probing	Wrong Setup
Frequency	347935	5780	1547	889	805	532	342	122

Our next goal is to understand the frequency distribution of various attack types in six feature columns. Figure 11 shows the distribution of the seven attack types in all six feature columns.



The results illustrate that the Dos attack has the highest number but comes from a very selective source and destination type. In contrast, the scan attack has the highest source and destination distributions. For the sake of complexity, we have removed the 'Timestamp' columns from this study. The following important task in any machine learning study is feature generation. As the IoT attack datasets contain vast features and columns, we have used Label Encoder from the sklearn library for feature generation. Then, we labeled binarize and used it to make multiclass labels, and Standard Scalar was used to normalize the data. Finally, data is split into 80-20, 70-30, and 60-40 ratios for further analysis using the thirteen models described above.

- **Results Analysis:**

As in the evaluation criteria section, we discuss several metrics for this study; as mentioned above, we used two training test split ratios to evaluate all thirteen models described in the algorithm sections. For the sake of the simplicity of the study, we have used only the build algorithm model from the sklearn library without any hypertuning modification. The final results of the test data accuracy score for the twelve models are shown in Figure 12. The result shows other than Perceptron and Multilayer Perceptron, all models show robust accuracy scores for all three types of train test ratio splits, and the results are summarized In Table 6:

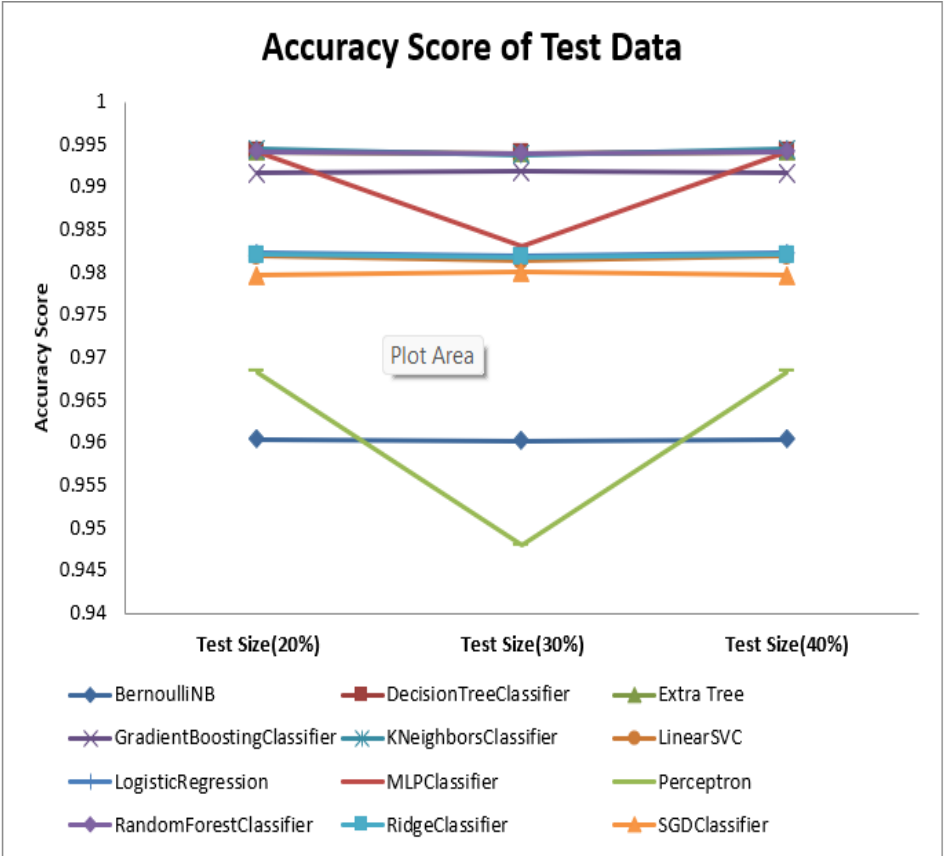


Fig 12: Comparing Model results according to training-test size ratio

Table 6.Data score of test data:

	Train(80) - Test(20) Split			
Model	Accuracy Score	Precision Score	Recall Score	F1 Score
Bernoulli NB	0.9604	0.9464	0.975	0.9599
Decision Tree Classifier	0.9942	0.994	0.9942	0.9936
Extra Tree	0.9942	0.994	0.9942	0.9936
Gaussian NB	0.0004	0.9579	0.9823	0.9586

Gradient Boosting Classifier	0.9916	0.9919	0.9937	0.9923
K Neighbors Classifier	0.9944	0.9944	0.9944	0.9938
Linear SVC	0.9819	0.9812	0.9869	0.982
Logistic Regression	0.9823	0.9819	0.9879	0.9836
MLP Classifier	0.9942	0.9941	0.9942	0.9936
Perceptron	0.9683	0.9751	0.992	0.9824
Random Forest Classifier	0.9942	0.994	0.9942	0.9936
Ridge Classifier	0.9821	0.9705	0.9823	0.9759
SGD Classifier	0.9797	0.9806	0.9854	0.9819

Our next goal is to compare the models according to their ROC curves. Figure 13 shows all twelve models' respective ROC curves (we removed one for an abysmal accuracy score).

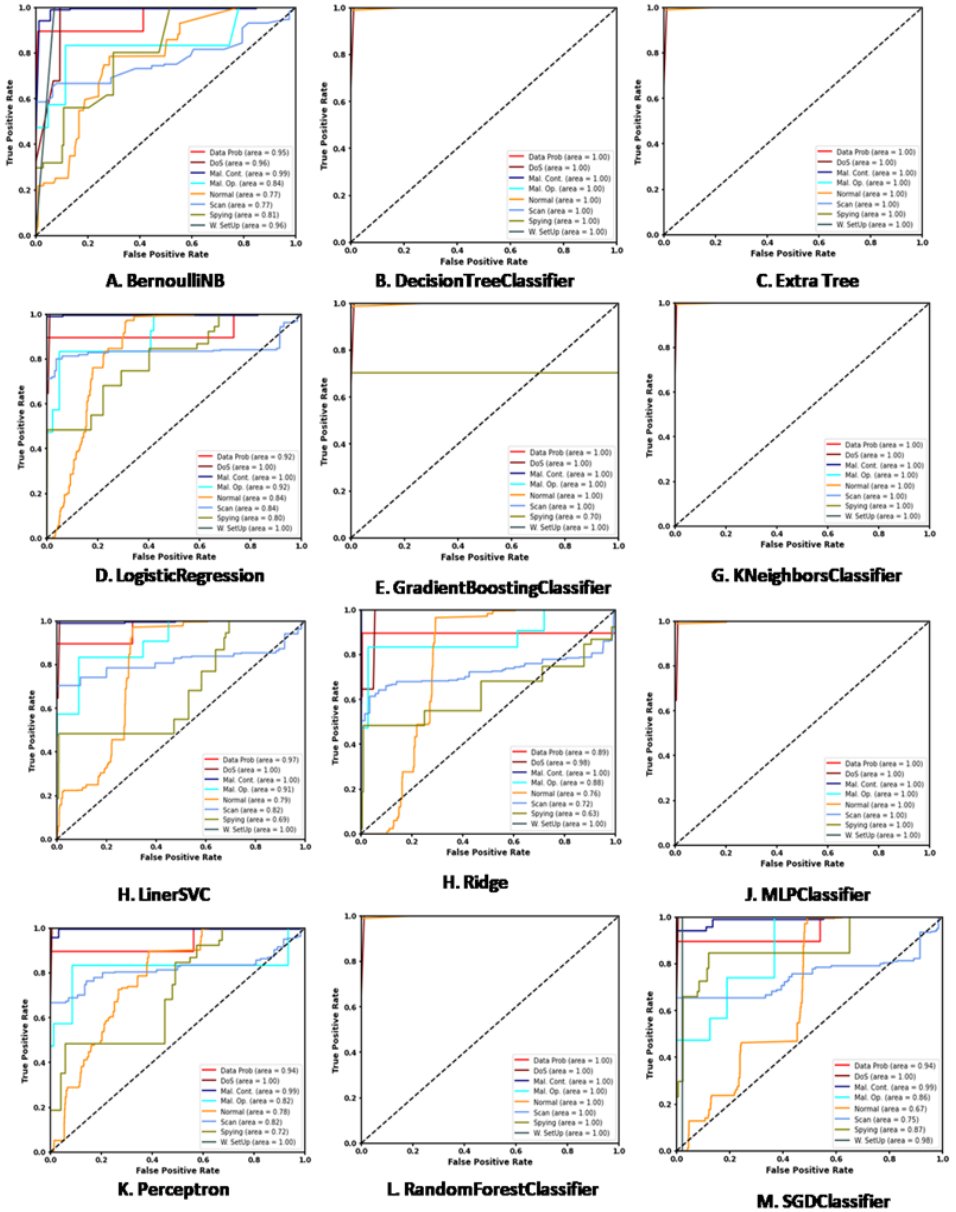
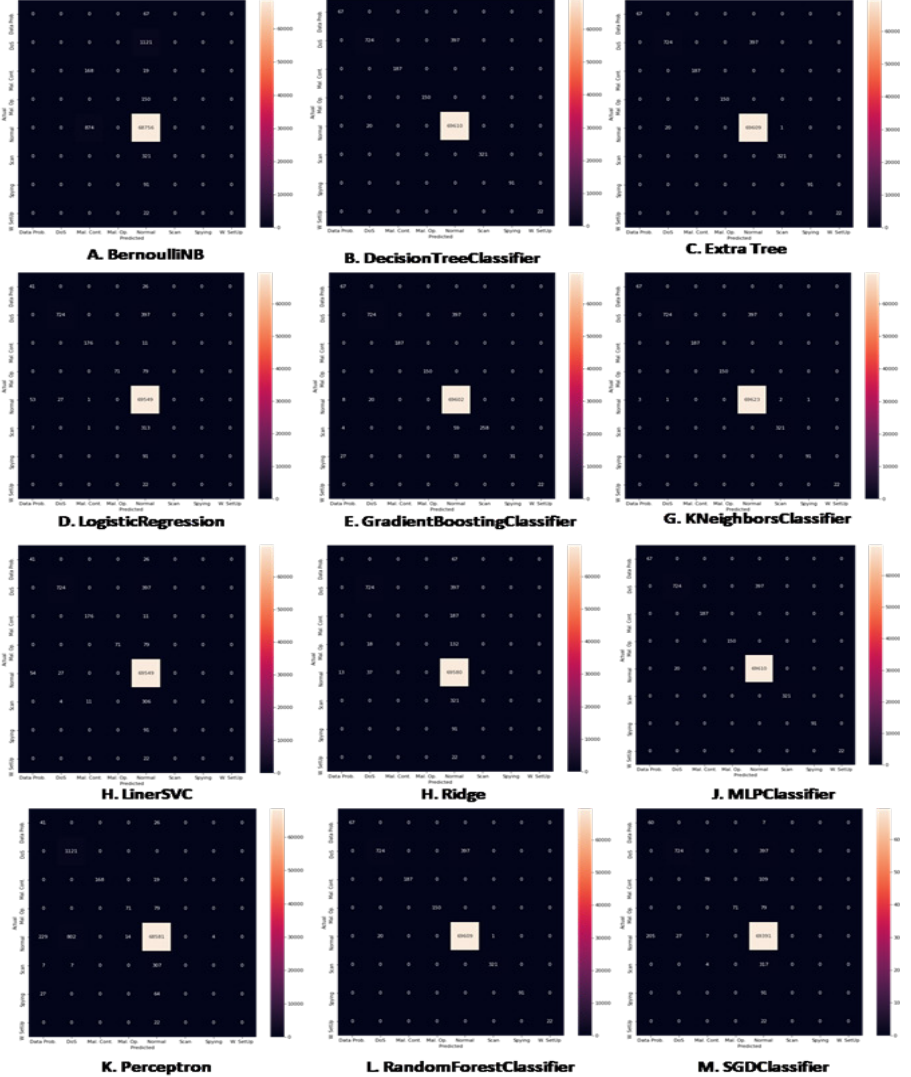


Figure 13: ROC curve of all algorithms model

The ROC curve of the tasted model illustrated in Figure 11 for all seven anomaly class IOT attacks and standard class is represented by the change in true and false favorable rates. According to the results, the best-performing model is the K nearest Neighbors algorithm, which performed very well in



identifying all seven attacks and one regular class. However, the running time is longer than that of the other algorithm. The three-tree algorithm, e.g., Decision Tree, Extra Tree, and Random Forest, shows similar kinds of ROC curves and a nearly 99% accuracy rate for all the classes. Single-layer neural network algorithm perception is excellent in predicting DoS attacks but fails to identify standard IoT signals. However, the multilayer neural network MLP classifier shows comparably better results for all eight class identifications with a 99% accuracy rate. Gradient boosting classifiers perform very well in identifying all seven classes but poorly identifying scanning attacks. Other models like Bernoulli naïve Bayes, logistic regression, Linear SVC, and SGD classifiers show a 96-88% accuracy rate of identification of different attack classes.

In the next part of this study, we analyzed the confusion matrix of our data for all twelve algorithm models. The confusion matrix is illustrated for all twelve models, as shown in Figure 12. The three-tree algorithm (i.e., Extra Tree, Random Forest, decision tree) shows a relatively similar confusion matrix: Dos attack 397 Dos attack misclassified as usual, and 20 average misclassified as DoS attack. K neighbors algorithm models also misclassified 397 DOS attacks as usual, three regular classes as Data Probing, and 1 DoS attack. Multi-layer Perceptron and Gradient Boosting Classifier Dos attack 397 Dos attack misclassified as usual, and 20 normal misclassified as DoS attack showing similar pattern like three tree algorithms. Single-layer Perceptron did not misclassify a single DoS attack but misclassified all other classes. Other algorithm algorithms have high misclassification rates, as shown in Figure 12.

Conclusion

Based on this IoT attack study, we analyzed twelve algorithm models using machine learning methods and evaluated their performance on various evolution metrics. Our findings of this study suggest that tree-based algorithms like decision trees, extra trees, and random forests show comparably similar results with a 99.42% accuracy score. In contrast, K's nearest neighbor shows a 99.44% accuracy score. The gradient boosting algorithm shows a 99.16% accuracy score

on test datasets. Although all these classifiers have issues with misclassification in DoS attacks, as usual, further study needs to resolve these problems. Consequently, this study identified five prospective models with the highest performance as an intrusion detection system.

References

Sherasiya, Tariqahmad, and Hardik Upadhyay. "Intrusion detection system for Internet of things." *Int. J. Adv. Res. Innov. Ideas Educ. (IJARIIE)* 2.3 (2016).

Muhammad, MU Farooq, Waseem Anjum, and Khairi Sadia Mazhar. "A Critical Analysis on the Security Concerns of Internet of Things (IoT)." *International Journal of Computer Applications* (0975 8887) 111.7 (2015).

Khan, Rafiullah, et al. "Future Internet: the Internet of Things architecture, possible applications, and key Challenges." 2012 10th International Conference on Frontiers of Information Technology. IEEE, 2012.

Wu, Miao, et al. "Research on the Internet of Things architecture." *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*. Vol. 5. IEEE, 2010.

Zhang, Weizhe, and Baosheng Qu. "Security architecture of the Internet of Things oriented to perceptual layer." *International Journal on Computer, Consumer, and Control (IJ3C)* 2.2 (2013): 37-45.

Cui, Ang, and Salvatore J. Stolfo. "A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan." *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010.

Mahmoud, Rwan, et al. "Internet of Things (IoT) security: Current status, challenges, and prospective measures." *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015.

Zhu, Bonnie, Anthony Joseph, and Shankar Sastry. "A taxonomy of cyber attacks on SCADA systems." *2011 International conference on the internet of things and 4th international conference on cyber, physical, and social computing*. IEEE, 2011.

Simmons, Chris, et al. "AVOIDIT: A cyber attack taxonomy." *9th Annual Symposium on Information Assurance (ASIA'14)*. 2014.

Thakur, Bhupendra Singh, and Sapna Chaudhary. "Content sniffing attack detection in client and server side: A survey." *International Journal of Advanced Computer Research* 3.2 (2013): 7.

Lucking Reiley, David. "Auctions on the Internet: What's being auctioned, and how?." *The journal of industrial economics* 48.3 (2000): 227-252.

Dorai, Ramakanth, and Vinod Kannan. "SQL injection-database attack revolution and prevention." *J. Int'l Com. L. & Tech.* 6 (2011): 224.

Gruschka, Nils, and Meiko Jensen. "Attack surfaces: A taxonomy for attacks on cloud services." *2010 IEEE 3rd international conference on cloud computing*. IEEE, 2010.

Jensen, Meiko, et al. "On technical security issues in cloud computing." *2009 IEEE International Conference on Cloud Computing*. Ieee, 2009.

Gruschka, Nils, and Meiko Jensen. "Attack surfaces: A taxonomy for attacks on cloud services." *2010 IEEE 3rd international conference on cloud computing*. IEEE, 2010.

Jensen, Meiko, et al. "On technical security issues in cloud computing." *2009 IEEE International Conference on Cloud Computing*. Ieee, 2009.

Padhy, Rabi Prasad, Manas Ranjan Patra, and Suresh Chandra Satapathy. "Cloud computing: security issues and research challenges." *International Journal of Computer Science and Information Technology & Security (IJCSITS)* 1.2 (2011): 136-146.

Pathan, Al-Sakib Khan, Hyung-Woo Lee, and Choong Seon Hong. "Security in wireless sensor networks: issues and challenges." *2006 8th International Conference Advanced Communication Technology*. Vol. 2. IEEE, 2006.

Babar, Sachin, et al. "Proposed embedded security framework for Internet of things (IoT)." *2011 2nd International Conference on Wireless Communication, Vehicular Technology,*

Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE). IEEE, 2011.

Ahmed, Nadeem, Salil S. Kanhere, and Sanjay Jha. "The holes problem in wireless sensor networks: a survey." *ACM SIGMOBILE Mobile Computing and Communications Review* 9.2 (2005): 4-18.

Cho, Jung-Sik, Sang-Soo Yeo, and Sung Kwon Kim. "Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value." *Computer Communications* 34.3 (2011): 391-397.

Hossain, Md Mahmud, Maziar Fotouhi, and Ragib Hasan. "Towards an analysis of security issues, challenges, and open problems in the Internet of Things." *2015 IEEE World Congress on Services*. IEEE, 2015.

Welch, Donald, and Scott Lathrop. "Wireless security threat taxonomy." *IEEE Systems, Man and Cybernetics Society/Information Assurance Workshop*, 2003. IEEE, 2003.

Mitrokotsa, Aikaterini, Melanie R. Rieback, and Andrew S. Tanenbaum. "Classification of RFID attacks." *Gen* 15693 (2010): 14443.

Quinlan, J. Ross. *C4. 5: programs for machine learning*. Elsevier, 2014.

Scholkopf, Bernhard, and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

Mehrotra, Kishan, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of artificial neural networks*. MIT Press, 1997.

Moya, Mary M., and Don R. Hush. "Network constraints and multi-objective optimization for one-class classification." *Neural Networks* 9.3 (1996): 463-474.

Schölkopf, Bernhard, et al. "Estimating the support of a high-dimensional distribution." *Neural Computation* 13.7 (2001): 1443-1471.

[30]. Hawkins, Simon, et al. "Outlier detection using replicator neural networks." *International Conference on Data Warehousing and Knowledge Discovery*. Springer, Berlin, Heidelberg, 2002.

Lindsay, Bruce G. "Mixture models: theory, geometry, and applications." *NSF-CBMS regional conference series in probability and statistics*. Institute of Mathematical Statistics and the American Statistical Association, 1995.

Rosenblatt, Murray. "Remarks on some nonparametric estimates of a density function." *The Annals of Mathematical Statistics* (1956): 832-837.

Pahl, Marc-Oliver, and François-Xavier Aubet. "All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection." *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018.

Pahl, Marc-Oliver, and François-Xavier Aubet. "All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection." *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018.

S B Imandoust et al. *Int. Journal of Engineering Research and Applications* www.ijera.com Vol. 3, Issue 5, Sep-Oct 2013, pp.605-610

A. McCallum and K. Nigam (1998). A comparison of event models for Naive Bayes text classification. *Proc. AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48.

P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees", *Machine Learning*, 63(1), 3-42, 2006.

Ho T.K. *Proceedings of the Third International Conference on Document Analysis and Recognition*, 1995, 1, IEEE (1995), pp. 278-282

L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees," Wadsworth, Belmont, CA, 1984.

Tolles, Juliana; Meurer, William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes." *JAMA*. 316

A. McCallum and K. Nigam (1998). A comparison of event models for Naive Bayes text classification. *Proc. AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48.

Xu, Wei. (2011). *Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent*.

Hinton, Geoffrey E. "Connectionist learning procedures." *Artificial intelligence* 40.1 (1989): 185-234.

Ho T.K. *Proceedings of the Third International Conference on Document Analysis and Recognition*, 1995, 1, IEEE (1995), pp. 278-282

J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics*, Vol. 29, No. 5, 2001.

He Jinrong et al., "Kernel ridge regression classification," *Neural Networks (IJCNN) 2014 International Joint Conference on*, 2014.

Crammer and Singer *On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines*, JMLR 2001.