
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 13–28, DOI 10.12694:/scpe.v22i1.1818

MITIGATING MALICIOUS INSIDER ATTACKS IN THE INTERNET OF THINGS USING SUPERVISED MACHINE LEARNING TECHNIQUES

MIR SHAHNAWAZ AHMAD* AND SHAHID MEHRAJ SHAH†

Abstract. The interconnection of large number of smart devices and sensors for critical information gathering and analysis over the internet has given rise to the Internet of Things (IoT) network. In recent times, IoT has emerged as a prime field for solving diverse real-life problems by providing a smart and affordable solutions. The IoT network has various constraints like: limited computational capacity of sensors, heterogeneity of devices, limited energy resource and bandwidth etc. These constraints restrict the use of high-end security mechanisms, thus making these type of networks more vulnerable to various security attacks including malicious insider attacks. Also, it is very difficult to detect such malicious insiders in the network due to their unpredictable behaviour and the ubiquitous nature of IoT network makes the task more difficult. To solve such problems machine learning techniques can be used as they have the ability to learn the behaviour of the system and predict the particular anomaly in the system. So, in this paper we have discussed various security requirements and challenges in the IoT network. We have also applied various supervised machine learning techniques on available IoT dataset to deduce which among them is best suited to detect the malicious insider attacks in the IoT network.

Key words: Internet of Things, attack detection, security, malicious insider, supervised machine learning.

AMS subject classifications. 68M14

1. Introduction. The recent advancement in various technological fields has led to the interconnection of enormous devices over the Internet, thus giving rise to the Internet of Things (IoT) network. With the help of IoT network even the ordinary devices (e.g. wearables, smart meters, smart water meters etc.) used by human beings in day-to-day living can be used to gather the information from surroundings using sensors/actuators, which can be used to solve various real life problems. IoT has transformed the classical field into smart field like: smart healthcare, smart transport, smart grid, smart home, smart waste management and many more [1]. However, an IoT network has some constraints like, limited computational capability of sensors, heterogeneity of devices, limited energy resource and bandwidth etc. These constraints restrict the use of high-end security mechanisms in IoT network and thereby makes such networks more vulnerable to malicious insider attacks[2].

A malicious insider can be employee or an ex-employee or a business partner of a company, who has or once had an authorized access to the company's data or network. Malicious insider can exploit that access to launch various malicious attacks in the company's network [3]. Non-authorized attackers are comparatively easy to detect since they have to break various authentication and authorization mechanisms employed by a company. Whereas, an insider is already known to company's security mechanisms and gets an easy access to the company's network and crucial data, thus making it hard for the already implemented security mechanisms to detect such attackers in the network. With the increasing use of IoT devices the insider attacks may also increase drastically because we are surrounded by a large number of IoT devices. Since IoT devices are usually low-end devices with limited resources hence it becomes easy for the insider attackers to launch malicious attacks in the network.

To solve such problem of detecting insider attackers in an IoT network, machine learning techniques can be used as they have the ability to learn the behaviour of the system and predict a particular anomaly in the system[4]. In order to use various machine learning techniques to detect malicious insiders in the IoT network,

*Communication Control & Learning Lab, Department of Electronics & Communication Engineering, National Institute of Technology, Srinagar, J&K, India. (mirshahnawaz888@gmail.com).

†Communication Control & Learning Lab, Department of Electronics & Communication Engineering, National Institute of Technology, Srinagar, J&K, India.

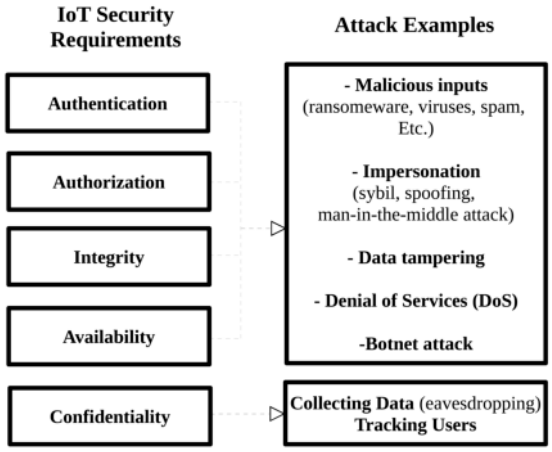

Fig. 2.1: Attacks that can affect the security requirements of IoT devices.

we need some dataset to train the algorithm. One such dataset is Network Security Laboratory–Knowledge Discovery in Databases (NSL-KDD). NSL-KDD is the latest version of KDD99 dataset [5]. The KDD99 was generated in 1999 as a result of international competition for Knowledge Discovery in Databases (KDD) and using DARPA98 network traffic. The various attacks included in NSL-KDD dataset include: DoS (Denial of Services) attack, User-to-Root attack, Remote-to-local attack and Probes. In the literature, this dataset has been widely used to assess the performance of anomaly-based attack detection systems for IoT network [6, 7, 8].

In this paper we apply various supervised machine learning algorithms on NSL-KDD dataset and analyse the performance of each algorithm. Based on various performance metrics we identify the best machine learning technique for malicious insider attack detection.

The rest of the paper is organized as: Section 2 describes various IoT security requirements and challenges; Section 3 highlights the characteristics of a malicious insider, its types and various malicious activities carried out by such attacker in an IoT network; Section 4 describes various supervised machine learning techniques used in our study; Section 5 outlines the attributes of NSL-KDD dataset and describes various performance parameters to measure the performance of each machine learning classifier for detecting a malicious insider; Finally, section 6 concludes the paper.

2. IoT security requirements and challenges. IoT is the integration of physical objects/things over an internet in order to create a smart living environment. The IoT devices are usually deployed in diverse environment to sense or gather the desired data, which can be used to accomplish various tasks. However, to successfully accomplish a task using IoT network, it must meet various security requirements [9]. The heterogeneous nature of IoT network makes it susceptible for various attacks. Due to the limited computational resources of IoT devices, it becomes computationally difficult to use complex security mechanisms for attack detection. The IoT devices are usually connected over a wireless medium, this makes them vulnerable to malicious intruders. Figure 2.1 shows some of the attacks that can affect various security requirements of IoT network, which are summarized as:

- *Authentication:* The identity of IoT network user should be first verified and only then they should be allows to access the network. However, due to limited resources of IoT devices the implementation of robust authentication mechanism is a challenging task. The authentication mechanism should have a trade-off between robustness, flexibility and IoT device constraints [10].
- *Authorisation:* It includes those security procedures which grant permission to legitimate users so that they can access various IoT devices in a network [2].
- *Integrity:* The sensed data by IoT devices is usually transferred/ shared via wireless channels, due to which it can be easily accessed by illegitimate users that can then modify it. So, one must ensure

integrity of IoT network data and should be flexible enough to deal with wide variety of IoT devices [2].

- *Availability:* The services provided by various IoT devices should be readily available to all the authorized users, but attacks like Denial of Services (DoS) and jamming attacks may create a hindrance by overwhelming the IoT devices. So, we need to implement such mechanisms which can detect such attacks in IoT network, but these attacks are difficult to detect and the heterogeneity & resource constraints of IoT network make it much more challenging task.
- *Confidentiality:* Like other networks, the security of gathered data in IoT network is very important. The confidentiality of data can be done by implementing various encryption algorithms, but keeping in view the resource constraints of IoT devices, these algorithms should be lightweight [11].

In this paper we deal with attacks related to availability requirement for the IoT network.

3. Malicious Insider Attacks in IoT. In this section we will be discussing in detail about the malicious insider attacks, their types and how they can affect the IoT networks.

3.1. Malicious Insider. Greitzer et al. [12] defines a malicious insider as a user of an organization who once had or has currently an authorized access to the organization's confidential data, resources or network, and uses these authorized privileges to launch various attacks in the network. Since the malicious insider has an authorized access to the network, so the authentication/ authorization mechanism implemented at the organizational level is unable to detect such attackers in the network. Due to this characteristic of an insider attacker node, it becomes very challenging for a security mechanism to detect such attackers in the network. Also, if these attackers are left undetected in an IoT network, it makes IoT nodes vulnerable to a malicious node. In order to further understand the details of insider attackers, we will be discussing various types of malicious insiders in the following section.

3.2. Types of Malicious Insiders. The malicious insiders can be categorized mainly into three categories: traitor, masquerader and unintended insiders. Among all these insiders, traitor is most threatening to a system and is responsible for launching around 92% of total attacks out of all the malicious insider attacks [13]. The traitors are those attackers who have authorized access to company's resources and use these privileges to launch various attacks in the company's network. They can be more severe since they already know the vulnerabilities of a network and are independent of any time constraint for launching attacks. Another category of insider attackers is a masquerader [14] who does not belong to an organization but somehow gets an access to the organization's networks (e.g. by guessing/ stealing the password of a legitimate user). The unintended insiders can also be categorized as a malicious insider, who threatens an organization by accidentally breaking the security mechanism [15].

3.3. Classification of Malicious Insider activities. Various activities that a malicious insider can perform in order to harm an organization mainly includes: IT Sabotage, Theft of Intellectual Property, Fraud and Espionage [9]. In IT sabotage, a malicious insider makes use of privileged access to network data and uses it to directly vandalise an organization or an individual by lunching attacks like DoS, sybil, man-in-the-middle, botnet and various other attacks. Since a malicious insider can access the organization's resources, so it can steal various innovative ideas, schemes and other essential artefacts which constitute various intellectual properties of an organization. A malicious insider can also perform fraudulent transactions that can affect an organization. The espionage includes all those activities with which a malicious insider can overhear the network data and sell the organization's critical information to other rival companies.

All the above discussed malicious activities by an insider can critically harm an organization and when an organization uses IoT network, then these attacks become more catastrophic due to low computational abilities of IoT devices. So, one needs to use a security procedure that will not only detect such anomalies in the system but also puts less load on IoT devices. The characteristics of machine learning techniques make them suitable for detecting such malicious attacks in the IoT network.

4. Supervised machine learning algorithms for detecting malicious insiders in IoT network. In this section we will discuss various supervised machine learning techniques that are used in our study.

4.1. Ridge Regression. If the data is given as (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, then a linear regression model can be written as:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} \tag{4.1}$$

where, $\beta_1, \beta_2, \cdots, \beta_p$ represents the regression coefficients for different x_i. The aim of linear regression is to choose those β_i for which the residual sum of squares minimize. Hence the optimization problem is framed as:

$$\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] \tag{4.2}$$

But, this model does not generalize well for the new data (i.e., the data with high variance) and hence result into overfitting. To overcome this problem, the ridge regression is used, which continuously narrows down the regression coefficients and hence producing a stable model [16]. The ridge regression overcomes this problem by adding a constraint for optimization problem of 4.2. Hence the optimization problem corresponding to a ridge regression can be written as:

$$\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right], s.t. \sum_j |\beta_j|^2 \leq 1 \tag{4.3}$$

Now to solve this optimization problem, we introduce Lagrange multiplier α, also known as regularization constant. Hence, Ridge estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \alpha \sum_j |\beta_j|^2 \tag{4.4}$$

subjected to the condition that for each x_{ij}, $\sum_i x_{ij}/N = 0$ and $\sum_i {x_{ij}}^2/N = 1$. Where, $\sum_j |\beta_j|^2$ is the regularization term and α is constant which controls the amount of regularization applied. In our study, we have set $\alpha = 1$ and allowed maximum iterations to 10.

4.2. Lasso Regression. In a model where a set of features are highly correlated the ridge regression will distribute weights equally to all the correlated features. Also, ridge regression will shrink the coefficients of less important predictors, but never makes them zero. This hinders the process of feature selection in a model. To overcome these limitation Lasso (Least Absolute Shrinkage and Selection Operator) regression can be used, which narrows down some of the model coefficients and sets others to zero [17]. It combines the best features of ridge regression and subset selection. For a given data (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, the Lasso estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \alpha \sum_j |\beta_j| \tag{4.5}$$

subjected to the condition that for each x_{ij}, $\sum_i x_{ij}/N = 0$ and $\sum_i {x_{ij}}^2/N = 1$. where α is the constant which controls the amount of regularization applied. The penalty term $\sum_j |\beta_j|$ has such effect that it forces some of the coefficients to exactly zero for large value of α. Thus, helping in variable selection and yielding sparse models. In our study, we have set $\alpha = 0.019$ and allowed maximum iterations to 150.

4.3. Elastic Net. The Lasso only selects one variable among a group of variables having high correlation and due to convex optimization it selects a limited number of features. These properties of Lasso affects its performance. To overcome this issue, elastic net regularization technique was proposed [18]. The elastic net can continuously shrink the model parameters and can also select a group of variables which have high correlation.

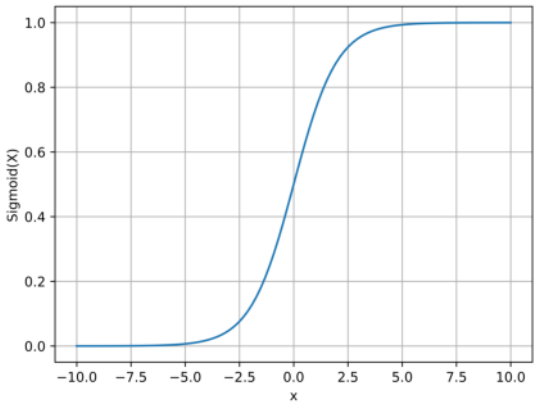

Fig. 4.1: Sigmoid Function

If the data is given as (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, then the elastic net estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \lambda_2 \sum_j |\beta_j|^2 + \lambda_1 \sum_j |\beta_j| \tag{4.6}$$

under the condition that: $\sum_{i=1}^{n} y_i = 0$, $\sum_{i=1}^{n} x_{ij} = 0$, and $\sum_{i=1}^{n} {x_{ij}}^2 = 1$, for $j = 1, 2, \cdots, p$. where λ_1 and λ_2 are non-negative constants. In our study, we have set $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$ with allowed maximum iteration to 150.

4.4. Lasso with Lars. Lars is a least-angle regression technique which helps to fit regression model to a high dimensional data. In our study we have used the combination of Lasso with Lars which resembles forward step-wise regression process, but at each step it selects those estimated coefficients which are in the direction equiangular with the square of residual error [19]. In its implementation for detecting malicious traffic in the IoT dataset, we have set α (constant which controls the amount of regularization applied) = 0.02 and maximum iteration to 50.

4.5. Logistic Regression (LR). Logistic regression describes a statistical method of predicting a binomial event. Like other Machine Learning classifiers, the input data can consist of one or multiple features. The outcome for the binary logistic regression is 0 or 1 that performs a differential positive class classification from the negative class. To give out a probabilistic value, Sigmoid curve shown in figure 4.1 is used in logistic regression [20].

The hypothesis function used in logistic regression in shown in equation below:

$$h(x) = S(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \tag{4.7}$$

where $x_1, x_2, \ldots, x_n$ denotes the features, $w_0, w_1, w_2, \ldots, w_n$ denotes the model weights and $S()$ represents the sigmoid function identified by:

$$S(Z) = \frac{1}{1 + e^{-Z}} \tag{4.8}$$

The range of output of $S()$ is from 0 to 1. All the values below 0.5 refer to negative class while as values from 0.5 to 1 indicate a positive class. While implementing logistic regression on the dataset we have set maximum iteration of 100 and used $\mathcal{L}^2$ norm for regularization, also we have used BFGS (Broyden Fletcher Goldfarb Shanno) optimization algorithm [21].

4.6. K-Nearest Neighbors (KNN). In KNN algorithm, the aim is to classify any new, unknown data point by analyzing the data points (most similar to the input or feature space) given in the training set [22]. The algorithm works by finding the K-nearest neighbours of the new data point by usually using one of the three distance measures for continuous variable viz. Euclidean distance (EuD), Manhattan distance (MaD) or the Minkowski distance (MiD) represented below:

$$EuD = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.9}$$

$$MaD = \sum_{i=1}^{n}|x_i - y_i| \tag{4.10}$$

$$MiD = \left[\sum_{i=1}^{n}(x_i - y_i)^q\right]^{1/q} \tag{4.11}$$

where, $x = (x_1, x_2, \ldots, x_n)^T$ & $y = (y_1, y_2, \ldots, y_n)^T$ represent the data points.

We have implemented the KNN classifier for various values of K on the given dataset. To choose the optimal value of K we observed that the accuracy of KNN algorithm increases with K but for $K > 10$ the accuracy saturates, hence choose the value of K as 10. We have used Euclidean distance to find K-nearest neighbours for a new data point.

4.7. Support Vector Machine (SVM). It is a Machine Learning classifier that is currently receiving the most significant attention due to its high performance [23] and the ability to alleviate the problem of classifying non-linear data. Typically SVMs are the non-probabilistic, binary classifiers aimed at discovering a hyperplane that divides the two classes of the training set with the highest margin.

If the data is given as $(\vec{x_i}, y_i)$ where, $\vec{x_i}$ represent the input vector and y_i is the outcome which indicates to which a particular $\vec{x_i}$ belongs. The SVM tries to find a maximum-margin hyperplane which divides $\vec{x_i}$ into different classes such that the distance between the hyperplane and the nearest point $\vec{x_i}$ from either classes is maximized. The equation of a hyperplane is given as:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{4.12}$$

where, $\vec{x}$ represent the input vector, $\vec{w}$ is the vector normal to hyperplane and b is the intercept (bias term). The objective of SVM is to minimize:

$$\left[\frac{1}{p}\sum_{i=1}^{p}\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))\right] + \alpha\|\vec{w}\|^2 \tag{4.13}$$

where, $\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))$ represent the hinge loss function, which return a value proportional to distance from margin to $\vec{x_i}$, if $\vec{x_i}$ is misclassified, otherwise returns zero. $\|\vec{w}\|^2$ represents the penalty for incorrect classification, which is controlled by constant α.

Apart from linear classification, SVMs perform non-linear classification of data by implicitly mapping an input variable into high dimensional attribute spaces using kernel trick [24].

As such SVMs tried in our work are classified into three categories: Linear SVM, SVM with Radial Basis Function (RBF) kernel and SVM with the polynomial kernel. The information about their kernels is given below.

Radial Basis Function (RBF) is the common kernel choice that is implemented in SVM. Given x and y as the input feature vectors, then the RBF kernel is given as:

$$K(x, y) = \exp(-\gamma||x - y||^2) \tag{4.14}$$

where, γ defines the influence of single training example in a model, i.e., increased value of γ results into over-fitting and decreased value results into under-fitting of model. $\gamma > 0$, and is often parametrized by $\gamma = \frac{1}{2\sigma^2}$, where σ is a free independent parameter. Therefore, the RBF kernel becomes:

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \tag{4.15}$$

To define the similarity among the input samples, Polynomial kernel not only takes into account the given features, but also the combinations of samples. A polynomial kernel is given as:

$$K(x, y) = (x^T \cdot y + t)^c \tag{4.16}$$

where x, y denote input feature vectors, t is a complexity parameter which trades-off between higher-order and lower-order terms in the polynomial ($t \geq 0$), and c is the integer exponent. If $t = 0$, then the kernel is homogeneous.

In our study, for SVM, we have used $\mathcal{L}^2$ norm for regularization in all kernel types. For Linear SVM, maximum iteration was set to 10000 and kernel coefficient was $1/N$, where N is the total features in dataset. For polynomial kernel type the degree was chosen to be 3.

4.8. LR and SVM with Stochastic Gradient decent (SGD). SGD is an extended version of Gradient Descent algorithm that decreases the number of computation per iteration by incorporating randomness in the learning process. It repeatedly considers a single or batch training example and evaluates the gradient of the loss function to reduce the model's risk, and hence updates the parameters used by the model. Whether it be a strictly convex problem or a non-convex one, SGD performs better and supports robustness and scalability [25]. The main objective of SGD algorithm is to minimize the regularized training error, which is the combination of empirical risk and the regularization term. The empirical risk term measures the performance of training set using a loss function $L(\hat{y}, y)$ (that gives the cost of predicting $\hat{y}$, given y as the actual output). The regularization term decides the penalty for the model's complexity, whose impact is decided by the term α (non- negative hyperparameter) [26].

We have studied the performance of SGD using SVM and logistic regression. SVM uses Max-margin classification, where correct category score should be larger (by a predefined margin) than the collective wrong category scores. Log loss (or cross-entropy loss or the logistic loss) decreases as the probability of correct prediction increases. During simulation, maximum number of iterations was set to 10000 with $\alpha = 0.1$, where α is a constant which is multiplied to regularization term to control the degree of regularization. Here again we have used $\mathcal{L}^2$ for regularization.

4.9. Random Forest (RF) Classifier. Random Forest Classifier is an Ensemble method of classification, which combines the outputs of various decision trees to generate more accurate results. Thus, it is based on the simple philosophy that the collection of classifiers will always perform better than a single classifier. This classifier works well for large datasets, gives an estimate of essential features in the dataset and possesses robustness for noise and outliers [27]. It also decreases the generalization error by randomly selecting subset of features during the splitting of a node (which also decreases the degree of correlation between various generated trees). Random Forest uses the Gini Index (GI) as an approximation to choose the best set of features during tree formation.

$$GI = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{4.17}$$

where, p_i is the probability with which an element is classified to a particular class. Due to the use of multiple trees, some of the features in the dataset may be used more than once for the training purpose. This increases the classifier stability because a slight change in input may not cause any change in the output of classifier, hence making it more robust and accurate [27].

In our study we have chosen the best split at each node in a decision tree using gini index with a minimum allowed splits at each node as 2. The classifier was iteratively run on the training data to find the optimal

number of trees in the forest and maximum training samples to be used to train each base estimator. We observed that if we increase the number of trees in the forest beyond 10 and maximum training samples to be used for base estimator training beyond 1000, then there was no change in classifier's accuracy. So, we selected number of trees in the forest as 10 and maximum training samples to be used for base estimator training as 1000.

4.10. AdaBoost (AB). AdaBoost is an iterative machine learning technique which attempts to fit weak classifiers iteratively [28]. The process starts with un-weighted training sample, using which a weak classifier is trained and then at each iteration the weights are boosted and the classifier is trained again. At each iteration the weights of those examples are increased for which the classifier at previous iteration predicted wrong label and the weights for those examples are decreased for which the classifier at previous iteration classified correctly. This process continues until all the data entries are correctly classified or it becomes difficult for a classifier to predict a label for unclassified/misclassified data. In this way the final classifier acts as the linear combination of various classifiers used at each iteration.

In our study, we have used Decision tree classifier as the weak classifier at each step and the maximum size of estimator (at which it terminates) was set to 200. The learning rate was set to 1 and we have used SAMME (Stage-wise Additive Modelling using a Multi-class Exponential loss function) algorithm for implementing the boosting process [29].

4.11. Gradient Boosting (GB). Gradient boosting is also a form of ensemble method which uses gradient descent to minimize the model's loss function by incorporating weak classifiers in an iterative manner [30]. At each iteration the gradient descent algorithm selects those classifiers which minimize the loss function. In this technique each weak classifier is trained on the residue of strong classifier. The input to the algorithm is the training data, a loss function (which should be differentiable) and number of iterations. In each iteration, it computes the residual error from strong classifiers, uses it to fit the weak classifiers, chooses that classifier which minimizes the loss function and finally updates the model.

We have used Decision tree classifier as the weak classifier at each step and log loss function with learning rate of 0.1 in our study. Number of boosting stages was chosen to be 200 and the quality of node split in decision tree was done using mean squared error (with an improved score by Friedman).

4.12. Artificial Neural Network (ANN) Classifier. For ANN based classification we have used multi-layer perceptron (MLP) classifier. MLP is a multi-layer feed-forward neural network that trains using back-propagation algorithm [31]. The first layer of neurons acts as the input layer, which is responsible for taking inputs (feature vector), and the final layer is responsible for generating the classified output. In between these two layers, there can be multiple layers of neurons, known as hidden layer, which processes the input vector and helps the output layer to generate output. In hidden layers each connection from one layer to another is associated with a weight vector. Figure 4.2 shows an example of Multi-layer perceptron with only one hidden layer of neurons and a single neuron at the output layer.

Each neuron is associated with an activation function, which generates an output depending on the values of previous layer neurons and the associated weights. The training algorithm (SGD) tries to fit a perfect combination of weight vector values to reduce the loss.

For implementing MLP, we have used a single hidden layer of neurons (with number of neuron = 90 and the activation function as a rectified linear unit function) and a single neuron at the output layer to classify the input data into malicious (abnormal) or non-malicious (normal). We train the neural network using SGD algorithm with a learning rate = 0.001 and $\mathcal{L}^2$ norm for regularization.

5. Numerical results. To compare the efficiency and applicability of various Machine Learning algorithms for detecting malicious insider attacks in IoT, we have used a popular Intrusion detection dataset – NSL-KDD, the details of which are given in next sub-section.

5.1. Introduction to NSL-KDD dataset and data pre-processing. NSL-KDD is the latest version of KDD99 dataset and has been widely used by researchers to evaluate the performance of the attack detection system in IoT [6, 7, 8]. In NSL-KDD dataset, the training data is available in "KDDTrain+" file, consisting of 125973 data entries, out of which 67343 entries represent non-malicious and 58630 entries represent malicious.

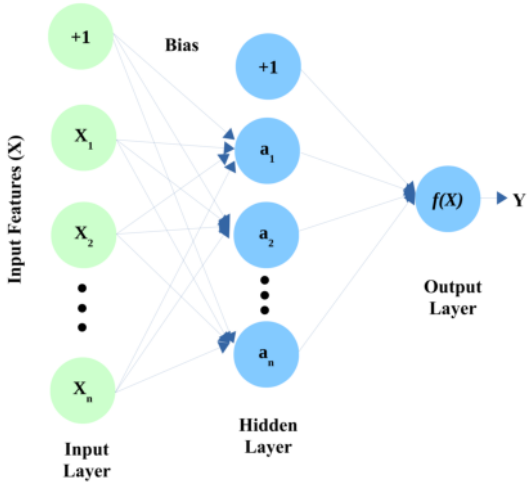

Fig. 4.2: Multi-layer Perceptron with single hidden layer and only one neuron in output layer

The testing data is available in "KDDTest+" file, consisting of 22543 entries, out of which 9710 represent non-malicious and 12833 entries represent malicious. Each dataset has 41 features and a single class label (as shown in figure 5.1). After analysing the feature of dataset, we have categorized them into four categories. Category – I represents those features which reveal the properties of TCP connection between source and destination node. Category – II highlights the basic features of the data that is being shared over a single connection between source and destination nodes. Category – III represents those features which depict the status of various connections in the network with a time frame of 2 seconds. Attributes of destination nodes in various connections in a network which help to analyse an attack that lasted for more than 2 seconds are represented by category – IV. The last feature represents the class label for each data entry (malicious or non-malicious) in the dataset. Among all the features, some of them represent various sub-features/attributes, they include: 'Protocol type', 'Service' and 'flag', and for the computational purposes we have converted them into numeric data. In 'protocol type' we have used 1, 2 and 3 for representing ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols respectively. The 'service' attribute contained 70 different web services used by the destination network. Various attributes of 'flag' feature are shown in figure 5.2.

Since the available data in the dataset is multidimensional and to enhance the performance of Machine Learning algorithm, we have normalized the data using standard scalar:

$$\hat{X}_i = \frac{X_i - \mu}{\sigma} \tag{5.1}$$

where, $\hat{X}_i$ is the normalized value of a data sample X_i with mean μ and standard deviation σ. It is evident from the above equation that the Standard Scaler normalizes the data by removing the mean of each feature and scaling the variance to 1.

5.2. Performance Parameters. To evaluate the performance of each trained machine learning model, confusion matrix was used, that is shown in Table 5.1.

Using the values obtained from the confusion matrix for each classifier, the following performance parameters were used to analyse the performance:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{5.2}$$

Category I	duration (0–to–54451)	protocol type (1, 2, 3)	service (1–to–70)	src_bytes (0–to–1379963888)	dst_bytes (0–to–309937401)
	Flag (1–to–11)	Land (0, 1)	wrong_fragment (0, 1, 3)	Urgent (0–to–3)	
Category II	Hot (0–to–101)	num_failed_logins (0–to–4)	logged_in (0,1)	num_compromised (0–to–7479)	root_shell (0, 1)
	su_attempted (0, 1)	num_root (0–to–7468)	num_file_creations (0–to–100)	num_shells (0–2)	num_access_files (0–9)
	is_guest_login (0, 1)	is_hot_logins (0, 1)	num_outbound_cmds (0)		
Category III	Count (0–511)	srv_count (0–511)	serror_rate (0–to–1)	srv_serror_rate (0–to–1)	rerror_rate (0–to–1)
	srv_rerror_rate (0–to–1)	same_srv_rate (0–to–1)	diff_srv_rate (0–to–1)	srv_diff_host_rate (0–to–1)	
Category IV	dst_host_count (0–to–255)	dst_host_srv_count (0–to–255)	dst_host_same_srv_rate (0–to–1)	dst_host_diff_srv_rate (0–to–1)	dst_host_same_src_port_rate (0–to–1)
	dst_host_srv_diff_host_rate (0–to–1)	dst_host_serror_rate (0–to–1)	dst_host_srv_serror_rate (0–to–1)	dst_host_rerror_rate (0–to–1)	dst_host_srv_rerror_rate (0–to–1)
	class_label (0–to–1)				

Fig. 5.1: Various features of NSL–KDD dataset with corresponding values in the dataset

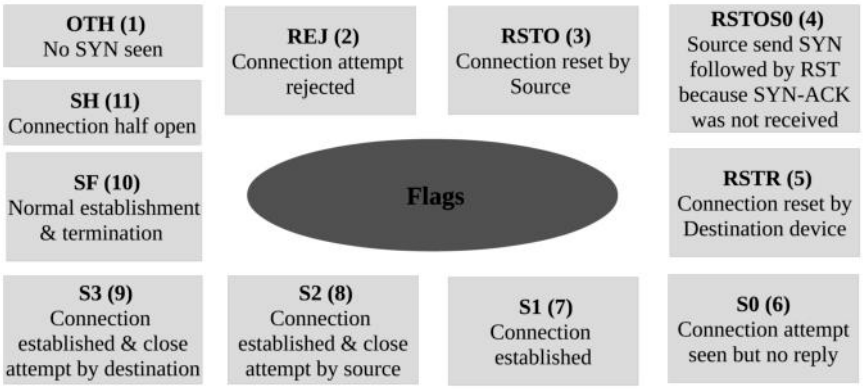

Fig. 5.2: Attributes of Flag feature in NSL–KDD Dataset

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

Table 5.1: Confusion Matrix

Actual Value	**Predicted Value**	
	Malicious	**Non-malicious**
Malicious	True Positive (TP)	False Negative (FN)
Non-malicious	False Positive (FP)	True Negative (TN)

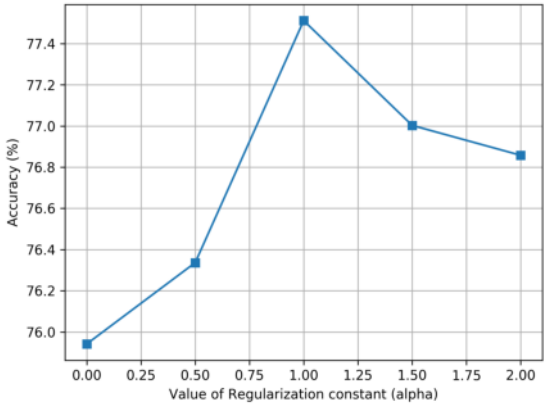

Fig. 5.3: Accuracy of Ridge regression for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of α

$$F1 - score = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.5}$$

The accuracy give the percentage of samples that are correctly classified out of all the tested samples, precision depicts the classifier's ability to detect only relevant data, recall gives the efficiency of a classifier to detect all interested data points in the dataset (i.e total positives detected by the system to that of actual positives throughout the system) and F1-score is the harmonic mean of both precision and recall.

We have also used Area Under the Curve (AUC) of a Receiver Operating Characteristic (ROC) curve as a performance matrix, which gives the overall performance of each classifier used in classifying the malicious and non-malicious traffic in KDDTest+ dataset. The value of AUC lies between [0, 1] and the performance of a particular classifier is directly proportional to the value of AUC obtained, i.e. a classification model is considered efficient if its AUC value lies close to 1 and inefficient if its value lies close to 0. Besides these performance parameters, we have also noted the time a particular classifier takes to train on KDDTrain+ dataset (Training Time) and the time it takes to predict a label for data in KDDTest+ dataset (Test Time). Evidently, more the training time of a particular classifier, more computation resources are needed for achieving classification, and lesser the test time indicates that the classifier will have quick response.

5.3. Performance evaluation of various machine learning algorithms for detecting insider attacks in IoT. In order to analyse the performance of various supervised machine learning algorithms for detecting insider attackers in IoT network, they were first trained on KDDTrain+ dataset and then their classification performance was evaluated using KDDTest+ dataset. For the implementation of these models, we have used intel core i3-5005U CPU @ 2.00GHz * 4, RAM: 4GB and operating system: ubuntu 14.04 LTS (64bit), using Scikit-learn and pandas package.

To choose the optimal value of various critical parameters in various classifiers, we have simulated the models for different values as shown in figure 5.3 to 5.8.

Figure 5.3 plots the accuracy of ridge regression to detect malicious IoT network traffic against various

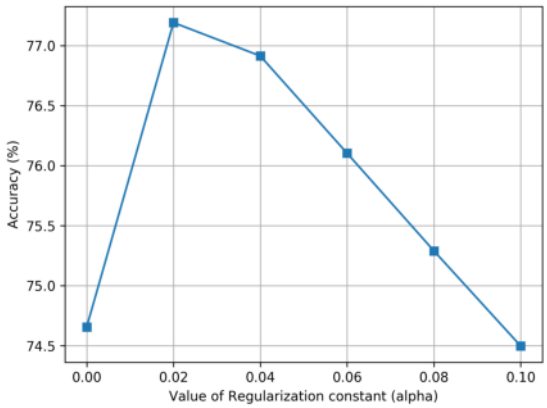

Fig. 5.4: Accuracy of Lasso regression for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of α

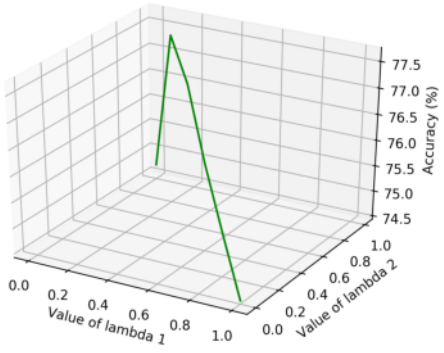

Fig. 5.5: Accuracy of Elastic Net for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of λ_1 and λ_2

values of α, and it is evident from the figure that maximum detection accuracy is achieved for $\alpha = 1$. Also, figure 5.4 proves that maximum detection accuracy is achieved at $\alpha = 0.02$ for Lasso regression. We have tested the Elastic Net for varying values of λ_1 and λ_2 on KDDTest+ dataset, as shown in figure 5.5. The figure shows that maximum accuracy is achieved for $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$.

The value of K in K-Nearest Neighbor classifier was chosen to be 10, since the detection accuracy saturates after $K = 10$ (as shown in figure 5.6). Figure 5.7 gives the optimal number of iterations for various classifiers by analysing the detection accuracy. Optimal values for number of training samples used to train base estimator for Random forest, AdaBoost and Gradient Boosting respectively are shown in figure 5.8.

Thus, after getting the optimal values for various parameters, the detailed results for detecting malicious insider attacks in the IoT network by various supervised machine learning algorithms are shown in Table 5.2, figure 5.9 and 5.10.

After thorough analysis of performance parameters in table 5.2 and figure 5.9 & 5.10 for various machine learning algorithms, we conclude that ensemble machine learning methods (Random forest, AdaBoost and

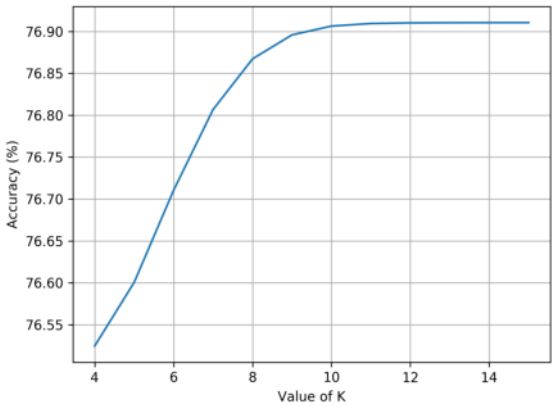

Fig. 5.6: Accuracy of K-Nearest Neighbor classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of K

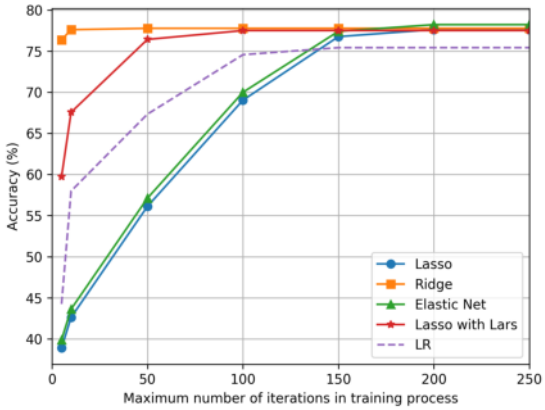

Fig. 5.7: Accuracy of various classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying number of iterations during training process

Gradient Boosting) perform better than the other studied machine learning classifiers for classifying the IoT network traffic into malicious and non-malicious. Among them the Gradient Boosting outperforms with an accuracy of 81.29% and AUC value of 0.96. Although the training time of Gradient Boosting algorithm is slightly more than some of the discussed algorithms, the testing time is nominal.

6. Conclusion. In this paper, we have discussed various characteristics and security requirements of an IoT network. We have also discussed how a malicious insider can be a major threat to IoT network and analysed various supervised machine learning algorithms which make them favourable for detecting such attacks in the IoT network. In order to demonstrate that the machine learning techniques can be used to detect malicious insiders in the IoT network, we trained various supervised machine learning algorithms on NSL-KDD dataset. The results show that Gradient Boosting technique outperforms the other discussed techniques. As part of future work, new and updated optimization techniques can be used to further enhance the attack detection accuracy of Gradient Boosting algorithm.

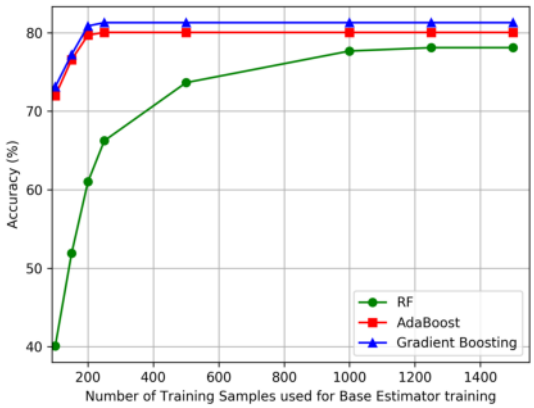

Fig. 5.8: Accuracy of various classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying number of training samples used to train base estimator

Table 5.2: Performance Analysis of various Machine Learning Classifiers on KDDTest+ dataset

Machine Learning Classifiers	**TP**	**FP**	**TN**	**FN**	**Accuracy (%)**	**Precision**	**Recall**	**F1-score**	**Training Time (Sec)**	**Test Time (Sec)**	**AUC**
Lasso	9391	4725	8108	319	77.62	0.83	0.78	0.77	0.93	0.0021	0.80
Ridge	9484	4786	8047	226	77.77	0.84	0.78	0.77	0.22	0.0021	0.80
Elastic Net	9404	4604	8229	306	78.22	0.84	0.78	0.78	0.62	0.0021	0.80
Lasso with Lars	9389	4752	8081	321	77.5	0.83	0.77	0.77	0.08	0.002	0.80
LR	9070	4899	7934	640	75.42	0.81	0.75	0.75	11.38	0.0021	0.87
K-NN	9482	4977	7856	228	76.91	0.84	0.77	0.77	74.39	184.52	0.85
Linear SVM	9071	4953	7880	639	75.19	0.81	0.75	0.75	410.21	0.0022	0.87
SVM with RBF Kernel	9508	4699	8134	202	78.25	0.84	0.78	0.78	117.93	9.68	0.94
SVM with Poly. Kernel	9500	5092	7741	210	76.48	0.83	0.76	0.76	100.81	7.85	0.89
LR with SGD	9342	4781	8052	368	77.16	0.83	0.77	0.77	0.50	0.0021	0.92
SVM with SGD	9515	4931	7902	195	77.26	0.84	0.77	0.77	0.36	0.0021	0.90
Random Forest	9430	4655	8178	280	78.10	0.84	0.78	0.78	14.05	0.32	0.96
AdaBoost	9409	4196	8637	301	80.05	0.85	0.80	0.80	34.18	0.79	0.95
Gradient Boosting	9424	3932	8901	286	81.29	0.86	0.81	0.81	73.47	0.11	0.96
MLP	9466	4581	8252	244	78.59	0.84	0.79	0.78	406.99	0.14	0.96

Acknowledgments. We would like to thank TEQIP-III and MITS, Gwalior for supporting this research.

REFERENCES

[1] F. Javed, M. K. Afzal, M. Sharif, and B.-S. Kim, *Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review*, in IEEE Communications Surveys & Tutorials, 20, No. 3 (2018), pp. 2062–2100.

[2] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, *Internet of things security: A survey*, in Journal of Network

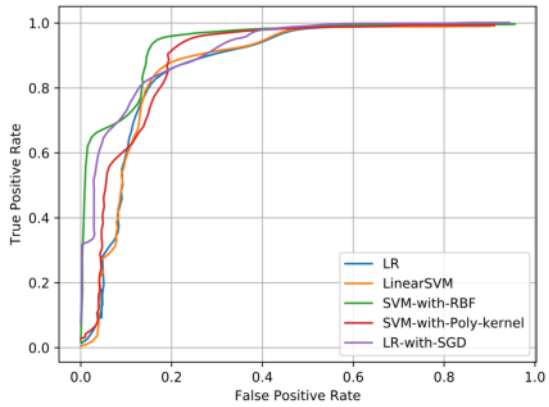

Fig. 5.9: ROC curves for various classifiers (having AUC > 0.85) after classifying IoT network traffic in KDDTest+ dataset (Part 1)

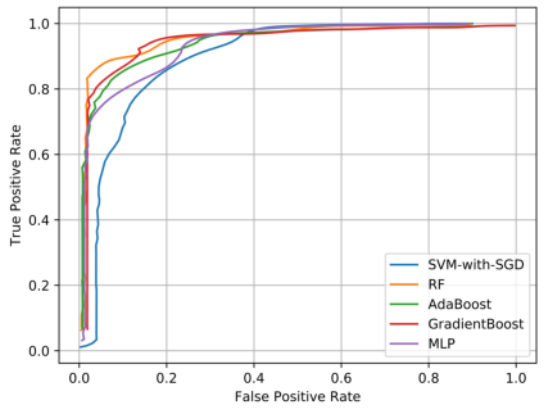

Fig. 5.10: ROC curves for various classifiers (having AUC > 0.85) after classifying IoT network traffic in KDDTest+ dataset (Part 2)

and Computer Applications, 88 (2017), pp. 10–28.

[3] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, *A survey on data leakage prevention systems*, in Journal of Network and Computer Applications, 62 (2016), pp. 137–152.

[4] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, *Iot security techniques based on machine learning: How do IoT devices use AI to enhance security?*, in IEEE Signal Processing Magazine, 35, no. 5 (2018), pp. 41–49.

[5] Nsl-kdd dataset, *https://www.unb.ca/cic/datasets/nsl.html*, (accessed: 02.08.2020).

[6] R. K. Gunupudi, M. Nimmala, N. Gugulothu, and S. R. Gali, *Clapp: A self constructing feature clustering approach for anomaly detection*, in Future Generation Computer Systems, 74 (2017), pp. 417–429.

[7] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, *Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset*, in IEEE Access, 8 (2020), pp. 29,575–29,585.

[8] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral, and G. dos Santos Vieira, *Hybrid approach to intrusion detection in fog-based IoT environments*, in Computer Networks, 180 (2020), pp. 107417.

[9] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, *Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures*, in ACM Computing Surveys (CSUR), 52, no. 2 (2019), pp. 1–40.

[10] J. Lopez, R. Roman, and C. Alcaraz, *Analysis of security threats, requirements, technologies and standards in wireless*

sensor networks, in Foundations of Security Analysis and Design V. Springer (2009), pp. 289–338.
[11] X. Yao, Z. Chen, and Y. Tian, *A lightweight attribute-based encryption scheme for the internet of things*, in Future Generation Computer Systems, 49 (2015), pp. 104–112.
[12] F. L. Greitzer and D. A. Frincke, *Combining traditional cyber security audit data with psychosocial data: towards predictive modeling for insider threat mitigation*, in Insider threats in cyber security. Springer (2010), pp. 85–113.
[13] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow et al., *Detecting insider threats in a real corporate database of computer usage activity*, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (2013), pp. 1393–1401.
[14] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, *Detecting and preventing cyber insider threats: A survey*, in IEEE Communications Surveys & Tutorials, vol. 20, no. 2 (2018), pp. 1397–1417.
[15] F. L. Greitzer, J. Strozer, S. Cohen, J. Bergey, J. Cowley, A. Moore, and D. Mundie, *Unintentional insider threat: contributing factors, observables, and mitigation strategies*, in 2014 47th Hawaii International Conference on System Sciences. IEEE (2014), pp. 2025–2034.
[16] D. W. Marquardt and R. D. Snee, *Ridge regression in practice*, in The American Statistician, 29, no. 1 (1975), pp. 3–20.
[17] R. Tibshirani, *Regression shrinkage and selection via the lasso*, in Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1 (1996), pp. 267–288.
[18] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, in Journal of the royal statistical society: series B (statistical methodology), vol. 67, no. 2 (2005), pp. 301–320.
[19] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani et al., *Least angle regression*, in The Annals of statistics, vol. 32, no. 2 (2004), pp. 407–499.
[20] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, in John Wiley & Sons (2013), vol. 398.
[21] R. Battiti and F. Masulli, *Bfgs optimization for faster and automated supervised learning*, in International neural network conference. Springer (1990), pp. 757–760.
[22] T. Cover and P. Hart, *Nearest neighbor pattern classification*, in IEEE transactions on information theory, vol. 13, no. 1 (1967), pp. 21–27.
[23] C. Cortes and V. Vapnik, *Support-vector networks*, in Machine learning, vol. 20, no. 3 (1995), pp. 273–297.
[24] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, *Machine learning for internet of things data analysis: A survey*, in Digital Communications and Networks, vol. 4, no. 3 (2018), pp. 161–175.
[25] G. Xu, Z. Cao, B.-G. Hu, and J. C. Principe, *Robust support vector machines based on the rescaled hinge loss function*, in Pattern Recognition, 63 (2017), pp. 139–148.
[26] S. Mehrkanoon, X. Huang, and J. A. Suykens, *Non-parallel support vector classifiers with different loss functions*, in Neurocomputing, 143 (2014), pp. 294–301.
[27] T. G. Dietterich, *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization*, in Machine learning, 40, no. 2 (2000), pp. 139–157.
[28] T. Hastie, S. Rosset, J. Zhu, and H. Zou, *Multi-class adaboost*, in Statistics and its Interface, vol. 2, no. 3 (2009), pp. 349–360.
[29] X. Zhu, P. Zhang, X. Lin, and Y. Shi, *Active learning from stream data using optimal weight classifier ensemble*, in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 40, no. 6 (2010), pp. 1607–1621.
[30] J. H. Friedman, *Stochastic gradient boosting*, in Computational statistics & data analysis, vol. 38, no. 4 (2002), pp. 367–378.
[31] A. Guezzaz, A. Asimi, A. Mourade, Z. Tbatou, and Y. Asimi, *A multilayer perceptron classifier for monitoring network traffic*, in International Conference on Big Data and Networks Technologies. Springer (2019), pp. 262–270.

Edited by: Dana Petcu
Received: Oct 15, 2020
Accepted: Jan 20, 2021