Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.024424
Article

Tech Science Press

An Efficient Unsupervised Learning Approach for Detecting Anomaly in Cloud

P. Sherubha[1,*], S. P. Sasirekha[2], A. Dinesh Kumar Anguraj[3], J. Vakula Rani[4], Raju Anitha[3], S. Phani Praveen[5,6] and R. Hariharan Krishnan[5,6]

[1]Department of Information Technology, Karpagam College of Engineering, Coimbatore, Tamilnadu, India
[2]Department of Computer Science and Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamilnadu, India
[3]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, India
[4]Department of MCA, CMR Institute of Technology, Bengaluru, Karnataka, India
[5]Department of Computer Science and Engineering, Prasad V. Potluri Siddhartha Institute of Technology, Andhra Pradesh, India
[6]Department of Computer Science and Engineering, Residency College, Chennai, India
*Corresponding Author: P. Sherubha. Email: sherubha0106@gmail.com
Received: 16 October 2021; Accepted: 30 December 2021

Abstract: The Cloud system shows its growing functionalities in various industrial applications. The safety towards data transfer seems to be a threat where Network Intrusion Detection System (NIDS) is measured as an essential element to fulfill security. Recently, Machine Learning (ML) approaches have been used for the construction of intellectual IDS. Most IDS are based on ML techniques either as unsupervised or supervised. In supervised learning, NIDS is based on labeled data where it reduces the efficiency of the reduced model to identify attack patterns. Similarly, the unsupervised model fails to provide a satisfactory outcome. Hence, to boost the functionality of unsupervised learning, an effectual auto-encoder is applied for feature selection to select good features. Finally, the Naïve Bayes classifier is used for classification purposes. This approach exposes the finest generalization ability to train the data. The unlabelled data is also used for adoption towards data analysis. Here, redundant and noisy samples over the dataset are eliminated. To validate the robustness and efficiency of NIDS, the anticipated model is tested over the NSL-KDD dataset. The experimental outcomes demonstrate that the anticipated approach attains superior accuracy with 93%, which is higher compared to J48, AB tree, Random Forest (RF), Regression Tree (RT), Multi-Layer Perceptrons (MLP), Support Vector Machine (SVM), and Fuzzy. Similarly, False Alarm Rate (FAR) and True Positive Rate (TPR) of Naive Bayes (NB) is 0.3 and 0.99, respectively. When compared to prevailing techniques, the anticipated approach also delivers promising outcomes.

Keywords: Network intrusion detection system; feature selection; auto-encoder; support vector machine (SVM); anomaly

1 Introduction

Cloud Computing (CC) is a rising Internet model that offers complete services and provides them with complex software, hardware and protocol stack functionality [1]. Users make use of these services for productive computing. Even though service usage is an essence for CC, it will not provide ineffective information. Moreover, it abuses and attacks the network architecture [2]. For instance, a spiteful user resides over Virtual Machine (VM), intrudes on other VMs over the cloud, and utilizes other machines to spread malicious software or Denial of Services (DoS) attacks. However, the user's character will produce huge network traffic specifically to traffic users' accessing services from other networks and shows traffic among VM in the virtual network [3]. Cisco industries' research reports that global network traffic accounts for 95% of the total traffic network traffic by 2022. Specifically, traffic among VMs in a cloud environment may account for 85%. This traffic may be consistent and increased drastically and inevitably encounters malicious attacks [4]. A network attack results in huge damage to the cloud; however, it causes users to lose confidence in CC, affecting CC's sustainable and healthy development [5]. Intrusion detection is a technology for protecting the CC from various malicious attacks. Hence, cloud-based intrusion detection identifies and examines network traffic, specifically identifying malicious attack characteristics and eliminating damage [6]. Therefore, guaranteeing reliability and safety towards CC.

Also, cloud infrastructure is modeled with virtualization technology that renders virtual flow among VMs uncontrollable and invisible by conventional IDS (See Fig. 1). It behaves as a global view, centralized control and programmability. Therefore, it is extensively utilized in CC. In diverse investigations like [7], the author anticipated using SDN technologies to redirect traffic to grunt IDS to identify attacks. Here, snort is used for the signature detection model, cannot identify unidentified attacks, and adopts traffic. Anomaly detection is used for distinguishing anomalous traffic from normal traffic [8]. It is extremely suited for unknown attack detection; however, it shows a false alarm rate. Various kinds of Machine Learning approaches are used extensively in IDS Neural Network, Decision Tree, Random Forest and Support Vector Machine. Moreover, these techniques may offer detection accuracy and unsatisfactory classification. Intrusion detection-based outcomes depend not only on classifier performance [9]; however on the performance of input data quality. Usually, network traffic includes feature reduction and high dimensionality that causes feature dimensionality disaster [10]. Hence, feature detection is specifically essential for effectual enhancement in supervised classifier performance. It comprises two kinds of approaches: feature extraction and feature selection. Feature subset selection functions effectually by eliminating redundant or relevant features; feature subset selection may offer the finest performance based on a certain objective function. Various investigations have reported that feature selection approaches eliminate the 'dimensionality curse and attain superior detection performance over NIDS. Feature extraction-based mapping provides original higher dimensionality features to lower dimensionality features and produces new non-linear and linear combinations of original features. Recently, diverse investigators have demonstrated deep learning technology with resourceful IDS for feature extraction. It uses ML approaches to automatically extract effectual features from raw data and input them into the classifier for attack identification. The significant contributions of the research work are:

1. Initially, the dataset should be acquired from the online accessible resource. This work includes the NSL-KDD dataset for threat prediction.
2. Next, the most influencing features need to be analyzed from the dataset as this threat may show a huge impact on the application. Here, an auto-encoder is used for feature selection.
3. Finally, the prediction accuracy needs to be computed with the classifier model. Thus, the Naïve Bayes classifier is used for predicting the classifier's accuracy. The simulation is conducted in a MATLAB environment where metrics like accuracy, FAR, TPR are evaluated.

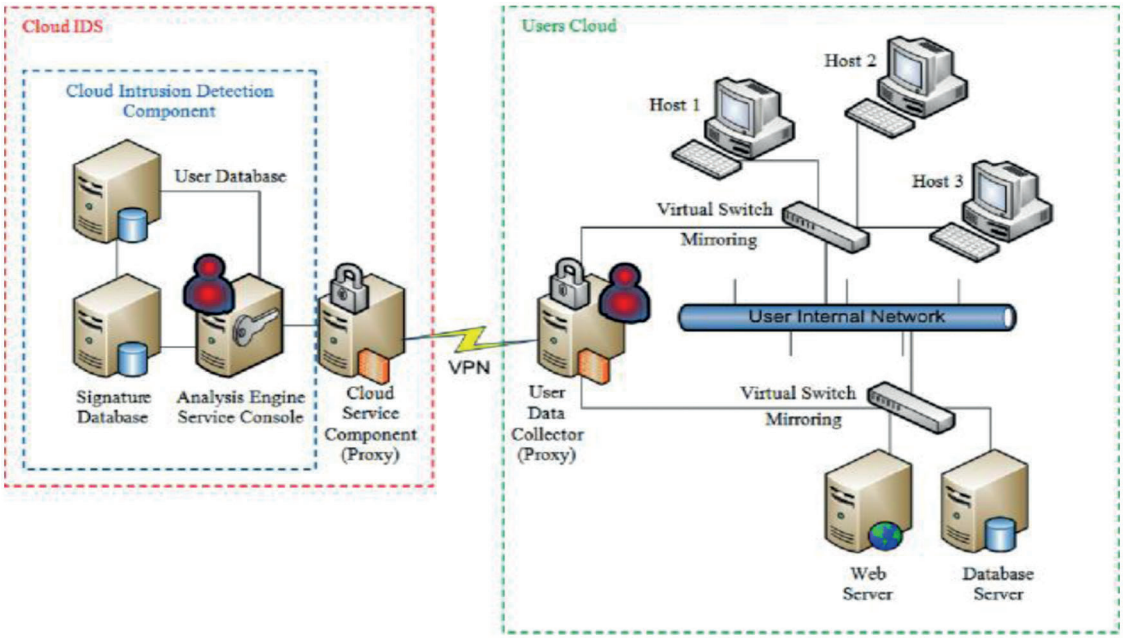

Figure 1: Generic IDS model

The remainder of the work is organized as. Section 2 depicts background studies of feature selection and classification approaches. Section 3 demonstrates anticipated auto-encoder-based feature selection and a Naïve Bayes classifier for modeling IDS. Section 4 illustrates numerical analysis and results of the anticipated model, and Section 5 gives results and directions for future research.

2 Related Works

This section discusses intrusion detection approaches that various other authors perform. An extensive study is done with IDS is discussed below. The author in [11] anticipated an anomaly detection approach that functions over the VM layer. These entries are stored in key-value format. Here, it specifies system cal, and values specify the list of system call. It executes program-based detection, cloud administrator and matches with database for identifying mismatch. It fails in longer traces, and attackers can evade those systems when VM's supervision process is done. The author in [12] depicts a frequency-based approach for safeguarding VM over cloud environment. Frequency counts of all system calls are maintained. Therefore, the system call order is lost. Approaches like the Markov model are utilized to generate more appropriate characteristics of malware-dependent probabilistic computation. The training time is extremely higher and shows substantial growth for huge databases. It includes experimental tasks and consumes time. It is diverse from other models as it converts frequency values to weights for performing testing/training with (k-Nearest Neighbor) k-NN classifiers.

The Virtual Machine (VM) model uses text mining techniques to generate feature vectors to enhance n-grams discriminative powers. It is utilized for feature selection by demonstrating the rarity and frequency of n-gram. Some models do not utilize virtualization features as introspection and hence lack predicting attacks that thwart IDS at VM. The author in [12] depicts VM based security framework for deployment over VM in virtualization. Policy engines are a core detection element that describes security for malicious activities by examining VM events. Signature-based approaches are utilized for detecting unknown and variant attacks. The author in [13] anticipated an IDS framework for modeling Operating System hooks. This hook is deployed over kernel modules to intercept OS events. Security analyzer examines the malicious activities of events. Also, attackers with kernel models may transmit false information. The author in [14]

anticipates a VM-malicious process for detection. It uses a process for identification approach regarding the tracking process to handle address space. It is based on address space. Active IDS specifies some new processes. It resourceful executes the implementation of Lycosid based on ID implementation. Similarly, the classifier model is used for predicting anomaly ineffectual manner. Self-training is to train both unlabelled and labeled data. Initially, it produces classifier fitting for sample labeling. Then, the classifier allocates labels with unlabelled samples. It is retrained with both unlabelled and labeled data for prediction. This process is iterated recursively till it satisfies termination criteria. Hence, various efforts are used to develop unsupervised learning [15–17]. But it suffers from certain disadvantages where confident instances are self-labeled from decision boundaries. Training is trapped due to outliers. There are some spaces for self-training approaches. It fails to enhance the performance of random splitting and reduce data redundancy [18–20].

3 Methodology

This work uses CC to construct NIDS for decoupling conventional cloud infrastructure. This anomaly detection approach is utilized to acquire three essential functionalities. 1) Pre-processing where network-based traffic is standardized; 2) feature selection is performed with auto-encoders; and 3) classifier model. Finally, attack recognition is utilized to identify intrusion over network traffic and testing datasets. This section explains two modules and is discussed as given below.

3.1 Data Collection

It is considered to be a preliminary step for predicting intrusion. Here, a network collector is utilized for traffic monitoring from a pre-defined platform. The physical machine comprises domain name and non-privileged domain. It is connected with a conventional switch and VM for connection with a virtual switch. VM communicates with successive VM or over different PM through virtual switches. It is utilized for forwarding network flow, while the collector is accountable for network flow and routing control. It is attained to handle anomaly detection with intrusion detection. This pre-processing model comprises of transformation. Former is utilized to change nominal to numerical values. For instance, in the NSL-KDD dataset, three features are considered: Transmission Control Protocol (TCP) status flag, service type, and protocol type. It is determined to be more nominal. It transfers attack type to numerical values, i.e., 0–4 is normal, DoS, probe, Root to Local (R2L) and User to Root (U2R) specifically [20–22]. The description of the NSL-KDD dataset is given in Section 4. To eliminate feature bias with superior values and the huge amount of sparse features, 0 should be standardized for scaling every feature value to a proportioned range [23–25]. Here, Z-score is utilized for standardization.

3.2 Basic Auto-Encoder (AE)

Auto-encoders are considered a popular method in Neural networks towards the unsupervised fashion. This method is anticipated and examined. The ultimate target of this process is to learn the dense representation of inputs during the maintenance of some essential information. The basic autoencoder is generally a Back Propagation (BP) method based on original data at the input, which are reconstructed at the output while passing through the encoding layer by diminishing the number of hidden layers. AE learns compact and deep features with reduced hidden nodes. Therefore, this specification is considered to be entirely reconstructed with some original data. Fig. 2 depicts the flow diagram of the anticipated model.

Consider, the training samples $x(x = [x_1, \ldots, x_m], x_k \in R^m, k = (1, \ldots, M)$ are the learning objective of an AE is to reduce the reconstruction error as specified in Eq. (1):

$$minimize: \sum_{k}^{M} ||x_k - \widehat{x_k}||^2 \tag{1}$$

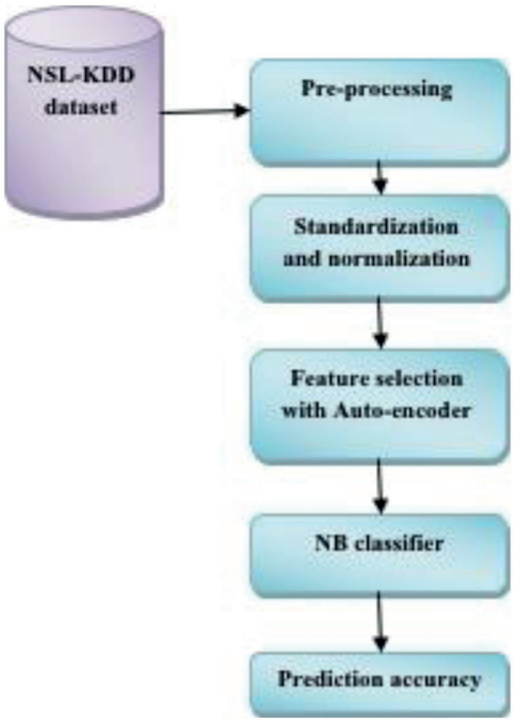

Figure 2: Flow diagram of the proposed model

Here, x_k and $\widehat{x_k}$ are input and reconstructed outputs respectively. Mathematically, the hidden layer relies for encoding and decoding process as expressed in Eq. (2):

$$\begin{cases} H_f = G(a_f,\ b_f,\ x) \\ \hat{x} = G(a_n,\ b_n,\ H_f) \end{cases} \tag{2}$$

Here, a_f and a specify the weight for encoding and decoding layers, b_f and b_n are biased, and $G(a_i, b_i, x)$ specify hidden neurons. The learning process is considered a preliminary auto-encoder that includes the iterative update of weights/coefficient s' a' and $'b$,' where the error among the inputs and reconstructed network outputs are gradually reduced until given below the threshold value. Generally, the learning processes of the auto-encoders are based on BP fine-tuning and layer-wise initialization. The anticipated AE structure is constructed in a multi-layer fashion. Initially, data transformation is performed to transfer input towards random feature space, followed by encoding layers to update the replacement learning process. At last, the parameters in encoding layers are updated, and input data is mapped into very low-dimensional features [26–28]. Mathematically, the two-layer network $'f'$ based output data is expressed as in Eq. (3):

$$f = G(a_n,\ \ b_n,\ \ G(a_f,\ b_f,\ x)) \tag{3}$$

Moreover, with multilayer network model, the output data for all hidden layers are expressed as in Eq. (4):

$$H_f^i = G(a_f^i,\ \ b_{pf}^i,\ \ H_f^{i-1}) \tag{4}$$

Here, H^i is i^{th} layers output data, and H^{i-1} is $(the\ i-1)^{\ th}$ layer. There are diverse discriminations among the multi-layer auto-encoders. The multi-layer AE is completely different from prevailing BP-based AE,

where the layers are merged to form a system. The system parameter needs to be iteratively retrained. The training speed of the anticipated model is multiple times faster than the conventional approaches. Similarly, the hidden nodes of the encoding layers are generated randomly. However, it is unreasonable to generate all the layers randomly. It is because the random layers can destroy the essential features. Therefore, the anticipated model with hidden nodes is evaluated and relatively nearer to the input data. However, the network layer needs a weighted output layer to absorb system errors as the hidden nodes are randomly generated. This model is naturally asymmetric as output layers' weight is not required during the encoding process but prevails during the decoding process. The unnecessary mapping can eliminate the necessity of low-dimensionality features [29–32]. The anticipated model can remove the learning system naturally. The features are optimized based on the preliminary findings of optimal parameters to reduce the output error with top-priority AE.

3.3 Learning Rate

This section explains the training process of the network architecture. Consider, $'M'$ training points $\{(x_k,\ y_k\}\ _{k=1}^{M},\ \ x_k\ \varepsilon\ R^m$ from the continuous system. The AE pretends to reconstruct the original input by $x = y,\ y \rightarrow (0,\ 1)$. The initial parameters of encoding layers are attained with the random process as expressed in Eq. (5):

$$H_f = G(a_f,\ \ b_f,\ \ x)$$

$$(a_f)^T.a_f = I$$

$$(b_f)^T.b_f = 1 \tag{5}$$

Here, $a_f \varepsilon\ R^{D*m}$ is random weights and bias of encoding layers. Initially, $b_f \varepsilon\ R^D$, then b_f belongs to $'R.'$ H_f is feature data. The hidden neuron $'G'$ is based on linear function, RBF function, sigmoid function, wavelet function, threshold function, and applied in decoding layer. Then, the invertible function (sine or sigmoid) can attain parameters from the encoding layer and be expressed as (6):

$$G^{-1}(.) = \begin{cases} arcsiin\ (.) & if\ G(.) = \sin(.) \\ -\log\left(\dfrac{1}{(.)} - 1\right) & if\ G(.) = 1/(1 + e^{-(.)}) \end{cases} \tag{6}$$

Here, Moore inverse matrix H is used. With Moore's matrix and invertible function, the solution possesses the least norm between the least-square solutions. The projection techniques are efficiently utilized for evaluating the inverse matrix; $H^{\dagger} = (H^T H)^{-1} H^T$; if $H^T H$ is non-singular; then $H^{\dagger} = H^T (H^T H)^{-1}$ if HH^T is singular. Based on regression theory, it is recommended to have a positive value and added to the diagonal of HH^T. It enhanced the stability and is expressed as in Eqs. (7) and (8):

$$H^{\dagger} = H^T \left(\frac{C}{I} + HH^T\right)^{-1} \tag{7}$$

$$H^{\dagger} = \left(\frac{C}{I} + HH^T\right)^{-1} H^T \tag{8}$$

Then, update a_f, b_f and expressed as in Eqs. (9) and (10):

$$a_f = (a_n)^T; \tag{9}$$

$$b_f = b_n; \qquad b_f\ \varepsilon\ R \tag{10}$$

The feature data is updated with $H_f = G(a_f, b_f, x)$. the above-given process is repeated concurrently. The parameters a_n, b_n, a_f, *and* b_f are attained along with the feature data H_f. Unlike BP, the conventional AE uses network weight and establishes it by pulling hidden nodes' input and output data. Moreover, when the parameters of encoding layers are randomly generated, the random encoding process can eliminate the essential information. There the input weights of the methods are evaluated and extremely correlated with input/output data. The anticipated model shows some benefits as the weight of the encoding layers does not compute and replace the weights of the decoding layer. Therefore, the computational workloads are eliminated. Based on this analysis, training error is reduced, and the testing process is increased gradually. Thus, the testing accuracy improves with the promising solution and diminishes enormous computational workloads. It is given in Algorithm 1:

Algorithm 1: Auto-encoder for feature selection

Step 1: Given a training set $\{(x_k, y_k\}_{k=1}^{M}$, $x_k \varepsilon R^m$, which is an invertible activation function, $G(.)$ is maximum loop number, and j = 1.

Step 2: Learning process;

Step 3: Random generation with the initial node a_f *and* b_f based on Eq. ()

Step 4: while $j < L$ do

Step 5: attain hidden nodes, a_n and b_n in decoding layer

Step 6: obtain hidden nodes from the encoding layer

Step 7: end while

Step 8: Attain feature data H_f.

3.4 Auto-Encoder for Feature Selection

Auto-encoder is an unsupervised dimensionality reduction approach that comprises of encoder and decoder [25]. It includes input, hidden and output layers. The encoder is cast-off for reducing dimensionality, and the decoder is utilized for reconstruction. The functionality is a vice-versa process. The training dataset has various' n'; $D = \{x^i, y^i | x^i \varepsilon R^d, y(i) \varepsilon R\}_{i=1}^{n}$, with feature reduction vector and class labels. Encoder functionality is mapped as input to hidden nodes, and decoder functionality is specified as the hidden specification for reconstruction [26]. When hidden layer neurons are smaller than several input/output neurons, the compression vector is attained and realizes dimensionality reduction. This process is depicted as in Eqs. (11) and (12):

$$h = f(wx + b) \tag{11}$$

$$z = g(w'h + b') \tag{12}$$

Here, $'f'$ and $'g'$ are non-linear activation functions. $'\omega'$ and $'\omega'$ are weighted matrices, *and* $'b'$ and' b' are biased values. Here, $\theta = \{w, b\}$ and $\theta' = \{w', be\}$ where θ' and θ specifies encoder and decoder parameters. The ultimate objective is to reduce reconstruction error among input and output by handling these factors. The objective functions are depicted as in Eqs. (13) and (14):

$$J_{auto-encoder} = \sum_{x \varepsilon D_n} L(x, z) \tag{13}$$

$$= \sum_{i=1}^{n} L\ (x^i, \ g_\theta, \ (f_\theta(x^{(i)})) \tag{14}$$

Here, $'L'$ is reconstruction error is MSE and cross-entropy loss. It is formulated as in Eqs. (15) and (16):

$$L(x,\ z) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \ z^{(i)})\hat{2} \tag{15}$$

$$L(x,\ z) = -\frac{1}{n}\sum_{i=1}^{n}[x^{(i)}\log Z^{(i)} - (1 - x^{(i)})\log(1 - z^{(i)}) \tag{16}$$

The smallest reconstruction error is depicted with output to input that implies effectual lower dimensionality feature specification. Here, reconstruction leads to an identical output problem to input. Thus, an auto-encoder is efficient in selecting features. To resolve this crisis, this work uses constraint specification or input corruption with the addition of noise. Unlike traditional AE, auto-encoder-based de-noising attempts to recognize effectual and appropriate feature specifications from noisy input data (See Fig. 3). It corrupts input and transmits that corrupted data to the anticipated mode for de-noising. Finally, it reconstructs *the* $'x'$ format. It gives objective function as in Eq. (17):

$$J = \sum_{i=1}^{n} E\_(\tilde{x}^{(i)} - \ q\ (\tilde{x}^{(i)}|x^{(i)})L(x^{(i)},\ \ g_\theta(f_\theta(\hat{x}^{(I)}))) \tag{17}$$

The noisy inputs are attained from the corruption process. Some approaches include Gaussian noise and masking noise for setting random input features to zero.

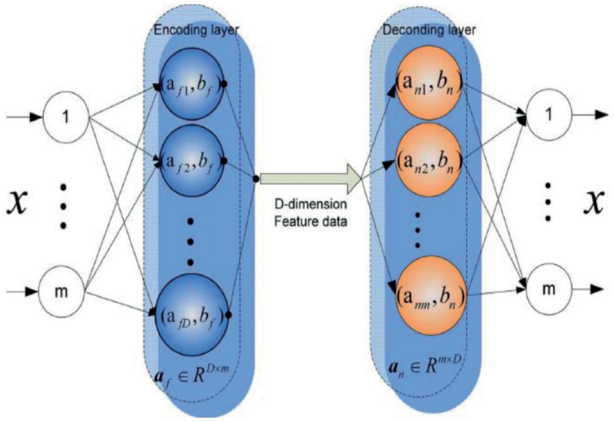

Figure 3: Auto-encoder for selecting appropriate threat features

3.5 Naïve Bayes Classifier

This model is a traditional approach with the Bayesian probabilistic model. It works on a stronger independence assumption towards probability measures. The probability measure does not influence the probability of other models. With a series of attributes $2n!$ gives independent assumptions. The outcomes of the NB classifier are generally more appropriate. The functionality of the NB classifier works effectually. It is depicted based on error factors like training noisy data, bias and variance. Training data noise is reduced by selecting appropriate training data. It is partitioned into various groups. Bias is known as an error owing to the training data group, which is extremely larger. Variance is depicted as an error owing to a grouping of smaller values. NB is a better classifier model when compared to existing approaches. It is because the essential characteristics of NB are extremely stronger assumptions towards

independence from every event or condition. Next, it is easier for execution. It is executed for larger datasets. The basics of the NB theorem are expressed as in Eq. (18):

$$P(C|X) = \frac{P(C)|P(X|C)}{P(X)} \tag{18}$$

where X is attributes; $'C'$ is class; $P(C|X)$ is even probability with X which has been occurred; $P(X|C)$ is even probability C which has been occurred; $P(C)$ is probability of $'C'$ event; $P(X)$ is probability of $'X'$ event. Similarly, X is expressed as in Eq. (19):

$$X = (x_1,\ x_2,\ \ldots,\ x_n) \tag{19}$$

The connectivity among X and C are expressed as in Fig. 4. Then, substitute 'X', bayes formula is expressed as in Eq. (20):

$$P(C|x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(C)P(x_1,\ x_2,\ \ldots,\ x_n|C)}{P(x_1,\ x_2,\ \ldots,\ x_n)} \tag{20}$$

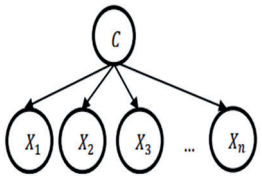

Figure 4: Naïve Bayes classifier

Moreover, $P(C|x_1, x_2, \ldots, x_n)$ as in Eq. (21):

$$\begin{aligned} P(C|x_1,\ \ldots x_n) &= P(C)P(x_1,\ x_2,\ \ldots,\ x_n|C) \\ &= P(C)P(x_1|C)(x_2,\ \ldots,\ x_n|C,\ x_1) \\ &= P(C)P(x_1|C)P(x_2|C,\ x_1)(x_3 \ldots x_n|C,\ x_1,\ x_2) \\ &= P(C)P(x_1|C)P(x_2|C,\ x_1)P(x_3|C,\ x_1,\ x_2)\ldots \\ &P(x_n|C,\ \ x_1,\ x_2,\ \ldots,\ x_{n-1}) \end{aligned} \tag{21}$$

From the above Eq. (21), it is known that there are some complex probability factors. It is extremely complex for analysis. Based on Eq. (21), there should be some assumptions that are free from one another. Therefore, the equation is expressed as in Eq. (22):

$$P(C|x_1,\ \ x_2,\ \ldots,\ x_n) = P(C)\prod_{i=1}^{n} P(x_i|C) \tag{22}$$

With NB classifier, the maximal probability of every class is expressed using hypothesis of maximum posteriori as in Eqs. (23) and (24):

$$H_{map} = \arg\max P(C|x_1,\ x_2,\ \ldots,\ x_n) \tag{23}$$

$$H_{map} = \arg\max P(C)\prod_{i=1}^{n} P(x_i|C) \tag{24}$$

In the NB classifier, the prediction class needs to be determined. However, if 'X' attributes are quantitative, the probability value is extremely lesser than $P(X|C)$, which cannot predict the H_{map} values. Thus, other approaches like Gaussian distribution have to be used as in Eqs. (25) and (26):

$$P = (X_i = x_i | C = c_j) \tag{25}$$

$$P = \frac{1}{\sqrt{2\pi\sigma_{ij}}} \exp\left(- \frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \tag{26}$$

Here, $'P'$ is the opportunity; X_i is *the* i^{th} attribute; x_i is the attribute value; $'C'$ is the value; C_i is the sub-class; μ is the attribute mean value; σ is the standard deviation.

Algorithm 2: NB classifier

Input: Unlabelled sample

1. Use encoder for dataset based dimensionality reduction
2. Allocate labels and datasets are re-written due to labeling
3. Use standardization and normalization to compute pre-processing
4. Perform feature selection
5. Initialize classifier model
6. Generate sampling model to normalize
7. Train NB classifier
8. End

Output: accurate anomaly detection over network model.

4 Numerical Results

This work considers a MATLAB environment with 6 GB RAM and a 2.2 GHz Intel Core i7 processor. Here, the NSL-KDD dataset is used for dataset validation (See Tabs. 4 and 5). The metrics like accuracy are used for predicting attacks (See Tabs. 6 and 8). Here, attributes like duration, protocol_type, service, flag, source bytes, destination bytes, land, wrong fragments, and urgent are considered samples. UDP and TCP are the two protocol types considered. The classifier test is given for class predictions that are loaded from the input file. The robustness of this model is measured for training and testing. The outcomes are validated with improved feature selection and classifier while comparison with another model. Accuracy detection is attained with diverse experimentation and constant valuation. Generally, a confusion matrix is utilized to determine the performance of the classification algorithm, as shown in Tab. 2. The terminology is extremely confusing; however, it is simple and gives a better understanding (see Tab. 1).

Here, true and false specify whether the class is predicted correctly or not. Similarly, positive and negative specify class prediction to show whether the class is correct or not. With the confusion matrix, accuracy, precision, F1-measure and recall are predicted. Here, accuracy, false alarm rate and true positive rate are predicted. The formulas for these metrics are expressed below in Eqs. (27) to (29):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{27}$$

$$TPR = \frac{TP}{TP + FN} \tag{28}$$

$$FPR = \frac{FP}{FP + TN} \tag{29}$$

Table 1: Challenges encountered in cloud IDS

S. no	Characteristics	Descriptions
1	Attribute identification	To deal with the complexity in predicting the appropriate quantitative data instance locality over high-dimensional space.
2	Distance measurement	Deals with data sparsity. The data points are equidistance in high dimensional space based on distance measurements
3	Sub-space prediction	Handles potential subspace features with increased input data dimensionality that outcomes in search space exponentially
4	Hubness	To deal with high dimensional behavior, which comprises data instances that concurrently appear in the nearest neighborhood, termed as hubs.
5	Uncertainty	The uncertain data is due to external events from vulnerable sources like attribute measures, vagueness, imprecision, ambiguity, and inconsistency. The data that does not deal with complete certainty is termed as uncertain.
6	Performance	The performance is measured in terms of memory and time essential during anomalies detection over high-dimensional data.
7	Scalability	The system ability is based on data size and increasing dimensions.

Table 2: Confusion matrix

	Actual	
Predicted	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Tab. 3 depicts accuracy prediction with the NSL-KDD dataset. Figs. 5 and 6 depict graph representation of accurate prediction. Tab. 7 depicts the TPR and FPR comparison.

Table 3: Accuracy prediction

Accuracy	
Naïve Bayes	92
J48	91
AD tree	89
SVM	88
Random forest	74
Bayes model	78
Decision stump	80

Table 4: NSL-KDD attributes and their descriptions

Att_no.	Att_name	Explanation	Sample data
1	Duration	Time duration length of the connection	0
2	Protocol_type	Protocol utilized in establishing a connection	TCP
3	Service	Utilization of destination network service	FTP_data
4	Flag	Connection status–normal	SF
5	Src_bytes	Sum of data bytes from S to D	491
6	Dst_bytes	Sum of data bytes from D to S	0
7	Land	If both IP addresses/port numbers are similar, the variable considers 1/0	0
8	Wrong_fragment	The total amount of incorrect fragments	0
9	Urgent	Sum of urgent packets. It is packets with urgent bit activated	0

Table 5: Attack identification attributes

Protocol type	src_bytes	dst_bytes	Logged in	Num root	srv_count	serror_rate	srv_error rate
‘udp'	146	0	0	0	1	0	0
‘tcp'	287	2251	1	0	7	0	0
‘tcp'	232	8153	1	0	5	0.2	0.2
‘tcp'	343	1178	1	0	10	0	0
‘tcp'	300	13788	1	0	9	0	0.11
‘tcp'	253	11905	1	0	10	0	0
‘udp'	147	105	0	0	1	0	0
‘tcp'	437	14421	1	0	1	0	0
‘tcp'	227	6588	1	0	22	0	0
‘tcp'	215	10499	1	0	14	0	0
‘tcp'	303	555	1	0	9	0	0
‘udp'	45	45	0	0	181	0	0
‘udp'	105	147	0	0	2	0	0

Table 6: Accuracy prediction with NSL-KDD dataset

Accuracy	NSL-KDD
J48	81
AB tree	76
RF	82
RT	80
MLP	81
SVM	77
Fuzzy	69
NB	93

Table 7: FAR and TPR comparison

Classifiers	FAR	TPR
SVM	0.9	0.94
k-NN	0.27	0.98
J48	0.10	0.97
AB tree	0.65	0.93
RF	0.73	0.95
RT	0.54	0.89
MLP	0.48	0.85
Fuzzy	0.26	0.93
NB	0.3	0.99

Table 8: Training accuracy

Training %	Accuracy
10	91.02
20	90.50
30	90.50
40	91.30
50	90.56
60	92.86
70	93
80	91.25
90	90.48

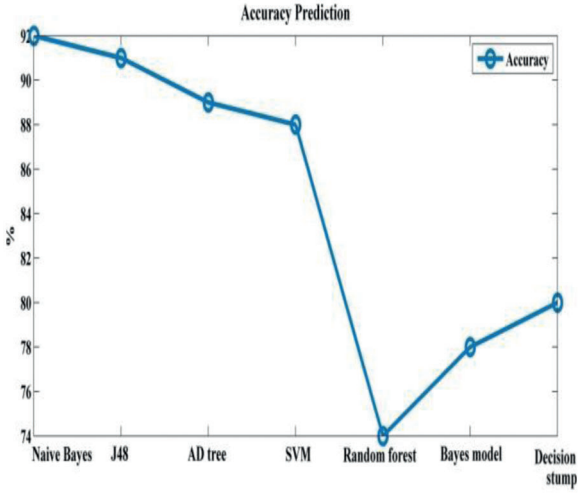

Figure 5: Accuracy prediction

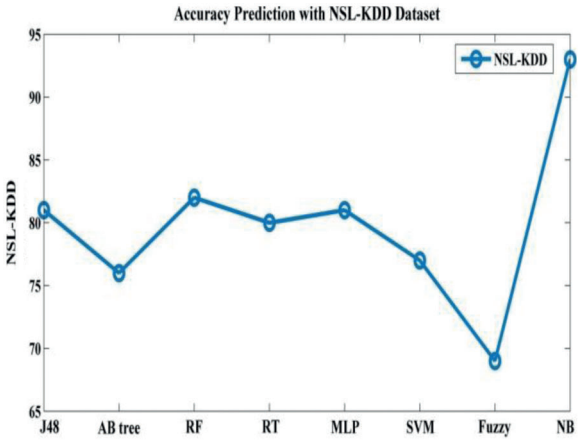

Figure 6: Accuracy prediction with NSL-KDD

This work compares the detection accuracy of anticipated NB with other models like J48, AD tree, SVM, random forest, Bayes model and decision stump, respectively. The anticipated model acquires 93% accuracy for classifier performance. The other model attains 91%, 89%, 88%, 74%, 78% and 80% respectively.

From the Table mentioned above, accuracy is computed based on evaluation with the NSL-KDD dataset. The prediction accuracy of NB is higher compared to other approaches. The detection accuracy of the anticipated NB approach is given in the above Table. The maximal prediction accuracy is 93%, superior to fuzzy models, RF, RT, SVM, MLP, etc. It is known that the anticipated model is more appropriate for classifying intrusion detection. The NB model shows the finest and productive performance compared to other models using testing and training data. Intrusion detection is done more effectually and suited as a basis for classification for anomaly detection. False Alarm Rate (FAR) and True Positive Rate (TPR) are the metrics used for evaluation (See Figs. 7 and 8). The comparison is made among existing approaches like SVM, k-NN, J48, AB tree, RF, RT, MLP, and Fuzzy approaches. The FAR of the proposed NB is 0.3, which is lesser when compared to other approaches. Similarly, the TPR of NB is 0.99 which is 0.5%, 0.1%, 0.2%, 0.6%, 0.4%, 0.10%, 0.14%, 0.6% higher than SVM, k-NN, j48, AB tree, RF, RT, MLP, and

Fuzzy respectively. The training of dataset samples are done with various percentages, i.e., 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% respectively with an accuracy rate of 91.02%, 90.50%, 90.50%, 91.30%, 90.56%, 92.86%, 93%, 91.25%, and 90.48% respectively. This work uses a 70:30 ratio for training and testing data partitioning. The auto-encoding-based feature selection can be identified for the NSL-KDD dataset. The anticipated model shows the finest algorithm within IDS with maximal intrusion detection accuracy. From Tab. 7, the FAR computation is measured to be 0.3 with 99% TRP. It is shown in Fig. 9, where the theoretical and experimental outcomes are determined. With this, it is known that the anticipated model works well in predicting the anomaly over the network.

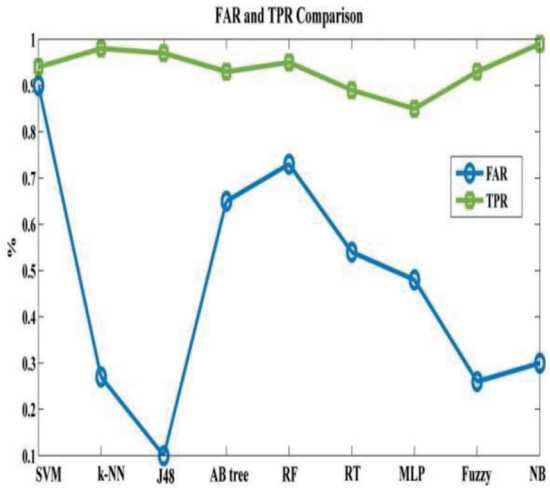

Figure 7: FAR and TPR computation

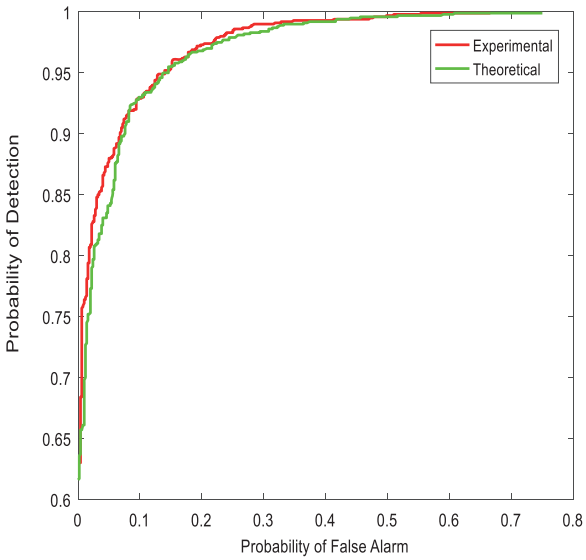

Figure 8: Anomaly detection probability

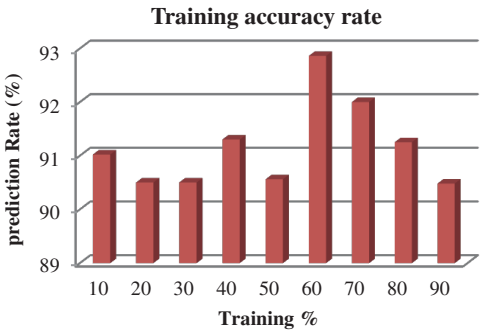

Figure 9: Training accuracy rate

5 Conclusion

The summary is based on the proposed model while performing various experimentations using the NB algorithm for anomaly detection. In recent days, eliminating security breaches encountered with current technologies has been extremely unrealistic. Therefore, intrusion detection is essential for measuring features in network security. Moreover, detection techniques are not capable of identifying unknown attacks. Thus, anomaly detection has to be utilized for predicting attacks. In contrast, the Anomaly detection approach is used for enhancing intrusion detection accuracy. This work develops an improved NB model for anomaly and intrusion detection. This method is extremely effective for identifying various attacks and showed superior detection accuracy compared to other models. This model could classify data as abnormal or normal. It is utilized to predict accuracy and improve performance; IDS-based accuracy and performances of the anticipated model have been enhanced, and it is executed over real-time network environments. The experimental outcomes demonstrate that the anticipated approach attains superior accuracy with 93%, higher than J48, AB tree, RF, RT, MLP, SVM, and Fuzzy. Similarly, FAR and TPR of NB is 0.3 and 0.99, respectively. The limitations of this work are the systematic analysis of the constant dataset, but the research lacks real-time investigation of the threat. Similarly, zero-day attacks should also be analyzed as it shows a huge impact on real-time applications. In the future, a deep learning classifier or ensemble classifier will be used for enhancing prediction accuracy.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] N. Pereira and R. M. de Moraes, "Comparative analysis of AODV route recovery mechanisms in wireless Ad hoc networks," *IEEE Latin America Transactions*, vol. 8, no. 4, pp. 385–393, 2010.

[2] S. Seo, S. Park and J. Kim, "Improvement of network intrusion detection accuracy by using restricted boltzmann machine," in *Proc. of 8th Int. Conf. on Computational Intelligence and Communication Networks (CICN)*, Tehri, India, pp. 413–417, 2016.

[3] K. Haseeb, N. Islam, A. Almgren and I. U. Din, "Intrusion prevention framework for secure routing in WSN-based mobile internet of things," *IEEE Access*, vol. 7, pp. 185496–185505, 2019.

[4] F. N. Nur, S. Sharmin, M. A. Habib, M. A. Razzaque and M. S. Islam, "Collaborative neighbour discovery in directional wireless sensor networks," in *Proc. of IEEE Region Conf. (TENCON)*, Singapore, pp. 1097–1100, 2017.

[5] F. N. Nur, S. Sharmin, M. A. Habib, M. A. Razzaque, M. S. Islam *et al.,* "Collaborative neighbor discovery in directional wireless sensor networks: Algorithm and analysis," *EURASIP Journal on Wireless Communications and Networking*, vol. 1, pp. 1–15, 2017.

[6] J. Yan, M. Zhou and Z. Ding, "Recent advances in energy-efficient routing protocols for wireless sensor networks: A review," *IEEE Access*, vol. 4, pp. 5673–5686, 2016.

[7] V. V. Mandhare, V. R. School and R. R. Manthalkar, "QoS routing enhancement using metaheuristic approach in a mobile ad-hoc network," *Computer Networks*, vol. 110, pp. 180–191, 2016.

[8] H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao *et al.,* "Hypergraph-induced convolutional networks for visual classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 2963–2972, 2019.

[9] P. Sherubha, S. P. Sasirekha, V. Manikandan, K. Gowsic and N. Mohanasundaram, "Graph based event measurement for analyzing distributed anomalies in sensor networks," *Springer Sådhanå*, vol. 1, no. 45, pp. 1–5, 2020.

[10] P. Sherubha and N. Mohanasundaram, "An efficient network threat detection and classification method using ANP-MVPS algorithm in wireless sensor networks," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 1597–1606, 2019.

[11] P. Sherubha and N. Mohanasundaram, "An efficient intrusion detection and authentication mechanism for detecting clone attack in wireless sensor networks, "*Journal of Advanced Research in Dynamical and Control Systems*, vol. 11, no. 5, pp. 55–68, 2019.

[12] K. Piotrowski, P. Langendoerfer and S. Peter, "How public key cryptography incense wireless sensor node lifetime," in *Proc. of Fourth ACM Workshop on Security of Ad-hoc and Sensor Networks*, New York, USA, pp. 169–176, 2006.

[13] F. B. Ian, G. Seroussi and N. P. Smart, "Advances in elliptic curve cryptography," in *London Mathematical Society Lecture Note Series*, 2nd edition. , vol. 317, Kindle Edition: Cambridge University Press, pp. 298–305, 2005.

[14] S. Zhu, S. Setia and S. Jajodia, "LEAP + efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 500–528, 2006.

[15] B. Panja, S. K. Madria and B. Bhargava, "Energy and communication efficient group key management protocol for hierarchical sensor networks," in *Proc of. IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan, pp. 384–93, 2006.

[16] X. Zhang and J. Wang, "An efficient key management scheme in hierarchical wireless sensor networks," in *Proc. of Int. Conf. on Computing, Communication and Security*, Washington, DC, USA, pp. 1–7, 2015.

[17] Q. Yuan, C. Ma, X. Zhong, G. Du and J. Yao, "Optimization of key pre-distribution protocol based on supernetworks theory in heterogeneous WSN," *Tsinghua Science and Technology*, vol. 21, no. 3, pp. 333–343, 2016.

[18] F. Gandino, R. Ferrero, and M. Rebaudengo, "A key distribution scheme for mobile wireless sensor networks: q–s–composite," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 34–47, 2017.

[19] S. Rahmani, A. Bilami and D. E. Boubiche, "EDAK: An efficient dynamic authentication and key management mechanism for heterogeneous WSNs," *Future Generation Computer Systems*, vol. 92, pp. 789–799, 2019.

[20] H. Fakhri, M. Johnston, F. Angelini and R. Tiwari, "The optimum design of location-dependent key management protocol for a multiple sink WSN using a randomly selected cell reporter," *IEEE Sensors Journal*, vol. 18, no. 24, pp. 10163–10173, 2018.

[21] B. Sun, Q. Li and B. Tian, "Local dynamic key management scheme based on layer-cluster topology in WSN," *Wireless Personal Communications*, vol. 103, no. 1, pp. 699–714, 2018.

[22] M. Omar, I. Belalouache, S. Amrane and B. Abbache, "Efficient and energy-aware key management framework for dynamic sensor networks," *Computers & Electrical Engineering*, vol. 72, pp. 990–1005, 2018.

[23] Y. Liu and Y. Wu, "A key pre-distribution scheme based on sub-regions for multi-hop wireless sensor networks," *Wireless Personal Communications*, vol. 109, no. 2, pp. 1161–1180, 2019.

[24] S. I. The Chu, Y. J. Huang and W. C. Lin, "Authentication protocol design and low-cost key encryption function implementation for wireless sensor networks," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2718–2725, 2015.

[25] K. A. Shim, "Basis: A practical multi-user broadcast authentication scheme in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1545–1554, 2017.

[26] R. Amin, S. K. H. Islam, G. P. Biswas and M. S. Obaidat, "A robust mutual authentication protocol for WSN with multiple base stations," *Ad Hoc Networks*, vol. 75, pp. 1–18, 2018.

[27] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2018.

[28] D. Wang, W. Li and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4081–4092, 2018.

[29] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symp. on Computational Intelligence for Security and Defence Applications*, Ottawa, ON, Canada, pp. 1–6, 2019.

[30] F. Farahnakian and J. Heikkinen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. of 20th Int. Conf. on Advanced Communication Technology*, Chuncheon, South Korea, pp. 178–183, 2018.

[31] R. Sathiyasheelan, "A survey on cloud computing for information storing," *Journal of Computational Science and Intelligent Technologies*, vol. 1, no. 2, pp. 9–14, 2020.

[32] M. Alqdah, "Intrusion detection attacks classification using machine learning techniques," *Journal of Computational Science and Intelligent Technologies*, vol. 2, no. 2, pp. 1–6, 2021.