Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Internet of things and intrusion detection fog computing architectures using machine learning techniques

Maha Helal[a]*, Tariq Kashmeery[b]*, Mohammed Zakariah[c] and Wesam Shishah[a]

[a]*College of Computing and Informatics, Saudi Electronic University, Riyadh 11673, Saudi Arabia*
[b]*College of Computing, Umm Al-Qura University, Makkah, Saudi Arabia*
[c]*College of Computers and Information Systems, King Saud University, Riyadh, Saudi Arabia*

C H R O N I C L E

Article history:
Received: April 7, 2024
Received in the revised format: July 25, 2024
Accepted: September 12, 2024
Available online: September 12, 2024

Keywords:
Machine Learning (ML)
Internet of Things
Anomaly detection
Intrusion Detection System (IDS)
Anomaly detection in IoT
Fog Computing
UNSW-NB15 dataset

A B S T R A C T

The exponential expansion of the Internet of Things (IoT) has fundamentally transformed the way people, machines, and gadgets communicate, resulting in unparalleled levels of interconnectedness. Nevertheless, the growth of IoT has also brought up notable security obstacles, requiring the creation of strong intrusion detection systems to safeguard IoT networks against hostile actions. This study investigates the utilization of fog computing architectures in conjunction with machine learning approaches to improve the security of the IoT. The UNSW-NB15 dataset, containing an extensive range of network traffic characteristics, is used as the basis for training and assessing the machine learning models. The study specifically applies and evaluates the performance of various models, including linear regression, Ridge classifier, SGD classifier, and ensemble learning. Furthermore, the findings indicate that these models are capable of accurately identifying intrusions, with success rates of 94%, 97%, 96.60%, and 96.50%, respectively. The Ridge Classifier demonstrates exceptional accuracy, highlighting its potential for effective implementation in IoT security frameworks. The results emphasize the efficacy of combining machine learning with fog computing to tackle the distinct security obstacles faced by IoT networks. In the future, our work will prioritize optimizing these models for real-time applications, improving their scalability, and investigating more advanced ensemble strategies to enhance detection accuracy. The project intends to enhance these areas to create a comprehensive and scalable intrusion detection system that can offer strong security solutions for the growing IoT environment. This will guarantee the integrity and dependability of linked devices and systems.

© 2024 by the authors; licensee Growing Science, Canada.

1. Introduction

Security breaches have considerably grown over the past ten years because of a thriving black market for cybercrime (Dhirani et al., 2023). The methods and tools employed in such security breaches have likewise significantly advanced. Such sophisticated assaults cannot be stopped or detected by traditional cybersecurity breach prevention measures like firewalls and anti-virus software (Alneyadi et al., 2016; Kochhar et al., 2023). The IoT has enabled various applications, including smart health, smart parking, supply chain, smart traffic control, power distribution, and logistics (Lu & Da Xu, 2018). All of these IoT-enabled applications share two tasks: "monitoring" (frequently verifying the condition of the sensors) and "actuating" (acting based on the data received during monitoring). While some of the data acquired were unexpected, some of the facts that were seen were crucial. The emergence of random data might be caused by environmental changes, deliberate action, incorrect operation, coincidence, or any of these factors. Recognizing notable or unanticipated network incidences requires anomaly detection (Prasad et al., 2009). A model must be found in most regular data to detect anomalies in a dataset. The data vectors that deviate from the usual model may subsequently be recognized as anomalies. A significant

* Corresponding author.
E-mail address takashmeery@uqu.edu.sa (T. Kashmeery)

© 2024 by the authors; licensee Growing Science, Canada.
doi: 10.5267/dsl.2024.9.003

problem is detecting irregularities in the network (Nimrah & Saifullah, 2022) with little overhead and the best detection accuracy. In addition, many Internet of Things applications are prone to delays. To protect the integrity of the executions, detecting anomalies as soon as possible, fixing anomalous causes, and adapting to environmental factors are critical. Anomaly detection is possible in the cloud (Garg et al., 2019). Nevertheless, using the cloud to provide the input data for anomaly detection will require a very high bandwidth due to the extremely high number of sensors—possibly millions in specific usage—and the enormous data they generate (Xin et al., 2023). Moreover, the functionality would not be available to edge systems. However, many devices do not have constant connectivity to the cloud. Therefore, to reduce the time, it takes to find anomalies, they should be found in the fog or edge layer (Jaiswal et al., 2020). This system facilitates information filtering, analysis, monitoring, allocation, and accumulation, which saves time and computational resources when big data analytics and cyber security applications are implemented and executed. Switches, routers, access points, and setup boxes for network edge design must have large amounts of GPU and CPU processors and storage. These systems, also known as fog nodes, can provide customers with computing resources as services (Manimurugan, 2021). The incapacity of a centralized cloud (Gupta & Namasudra, 2022) to satisfy IoT needs, such as resource allocation and scalability, is a significant factor in the present detection techniques for abnormalities. With IoT, processes span several devices, and massive volumes of data are created exponentially (Bustamante-Bello, 2022). The cloud is crucial to the IoT since it allows users to access Internet-based services (IoT). However, while performing costly computations, it cannot handle IoT devices due to its centralized architecture. A long detection time is also a result of the wide separation between IoT devices and the centralized anomaly detection scheme. Anomaly detection in IoT varies from presently employed approaches because the centralized cloud infrastructure can support IoT's service needs (Diro & Chilamkurti, 2018). The gap is closed using "computation," a brand-new distributed intelligence technology. The fog communicates by handing out data close to the data foundations or IoT gadgets.

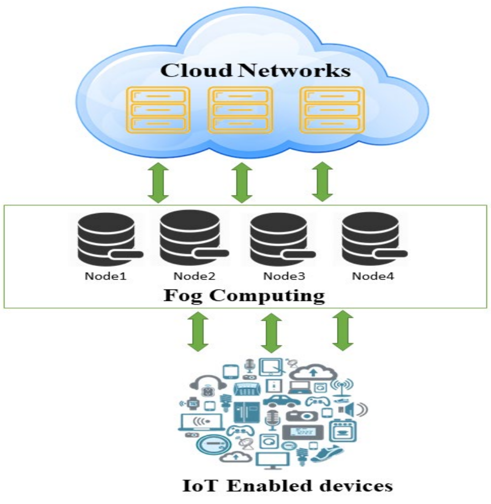


Fig. 1. Implementation of a system for anomaly detection in IoT systems

Fig. 1 depicts the IoT system's implementation of the anomaly detection system. The diagram suggests that IoT-enabled devices transmit the signal to the cloud network and the fog computing nodes. IoT is a network of interconnected devices that creates and consumes data; necessitating data flow across various devices. When it comes to the IoT, one of the most critical challenges is making sure that the data is secure. Malicious packets are frequently found and removed using a network IDS before they have an opportunity to penetrate a network. Because IoT devices are inherently limited energy and do not need many processing resources, this has to be accomplished at the fog node. Using characteristics from the UNSW-NB15 dataset, we provide a new IDS that could be deployed on fog nodes to detect unwanted traffic heading near IoT devices. This paper provided a conceptual framework and a mixed approach to address the practical issues of anomalies and intrusion of IoT network activity in a fog computing environment from various ML approaches. The following are a few of this study's main contributions:

- Machine learning methods aim to create Anomaly Detection in Fog Computing Architectures.
- Most IDS research is based on obsolete KDD cup99 or NSL-KDD data sets. In this analysis, we used the most current data set (UNSW-NB15), which covers the latest assaults, instead of the KDD cup99 dataset.
- An adapted Tab transformer model is suggested. This novel technique is being applied for the first time to identify abnormalities in fog nodes.
- We employed four of the most reliable machine learning techniques, such as linear regression, ridge classifier, SGD classifier, and ensemble learning, to evaluate the anticipated architecture for the dataset.

This paper is organized as follows. Section 2 discusses the analysis of the prior work's literature. Section 3 discusses the proposed strategy, as well as data collection. The methodology is discussed in section 4, and the results are discussed in section 5. Finally, section 6 is for discussion, while Section 7 outlines the conclusion.

2. Literature Review

This section covers the selection of ML techniques for detecting abnormalities and intrusions in IoT network traffic, relevant research, and extensive background data. In addition, we go through a few methods that are currently in use for fog architecture anomaly detection. One of the more severe attacks on the IoTs is the denial-of-service (DOS). Verma and Ranga (2020) research the potential of utilizing ML classification methods for protecting IoT from DOS attacks concerning this. The classifiers that can promote the growth of anomaly-based IDSs are thoroughly studied. Classifier effectiveness is evaluated using well-known statistics and authentication techniques. The CIDDS-001 (Daoud et al., 2023), UNSW-NB15 (Janarthanan & Zargari, 2017), and NSL-KDD (Dhanabal & Shantharajah, 2015) popular datasets are utilized to benchmark classifiers. The statistical analysis of the significant differences among classifiers uses the Friedman and Nemenyi tests. Raspberry Pi is also utilized to assess the classifiers' reaction times on devices designed for the Internet of Things (Bezerra et al., 2019). We also go through a way to choose the optimal classifier based on the application's demands. The primary aims of this paper are to encourage IoT security researchers to build IDSs utilizing ensemble learning and to provide suitable approaches for statistical classifiers measuring performance (Verma & Ranga, 2020).

IDS on an IoT network might use cognitive fog computing, according to the (Prabavathy et al., 2018). Instead of using a centralized cloud-based architecture, the recommended technique might identify harmful activity in adjacent fog nodes. The NSL-KDD dataset is utilized to evaluate the suggested framework, and the online sequential extreme learning machine (OSELM) method is used to detect anomalies. This approach has a 97.36% accuracy rate and a 0.37% FAR (Prabavathy et al., 2018). The authors of (Pacheco et al., 2020) created an Anomaly Behaviour Assessment Method based on ANNs (Ahmad, 2022) and ensemble strategy (de Souza et al., 2022) to incorporate an adaptive IDS that can identify after determining that a Fog node has been hacked, take the required steps to ensure that communication is still accessible. The IoT testbed was used to create the training dataset. The strategy has a 97.51% accuracy rate. One of the fields requiring a high level of knowledge to provide good outcomes is anomaly detection. However, Kayan et al. (2021) proposes AnoML, an end-to-end data science pipeline that enables the amalgamation of several anomaly detection methods, wireless communication conventions, and placement to the fog, edge, and cloud platforms and does so with a minimum of user intervention. By lowering the hurdles created by the diversity of an IoT ecosystem, they assist in expanding IoT anomaly detection methods. Training, data intake, strategic placing, model maintenance, and model inference are the four primary steps supported by the suggested pipeline. They assess the pipeline using two datasets for anomaly detection while contrasting the effectiveness of numerous ML methods at various nodes (Kayan et al., 2021).

Because of the vast quantity of data generated by IoT devices, ML is commonly employed for attack detection. The issue is that fog devices could not have the assets, such as memory and processing speed, to quickly identify threats. Tomer and Sharma (2022) provide a method for offloading the instantaneous forecast duty to the fog nodes and the ML framework selection task in the cloud. An ensemble ML framework is developed in the cloud using the suggested technique based on historical data, and then real-time attack detection on fog nodes is performed (Tomer & Sharma, 2022). According to the literature review, despite multiple research ideas on different identification models for effectively identifying IoT harmful traffic, a study has yet to show that ML procedure effectively detects IoT risky traffic. Most academics do tests to gauge the effectiveness of ML algorithms, and then they select the most effective strategy based on the findings. Nevertheless, it is essential to study and identify the most effective ML approach for detecting anomalies and intrusion in IoT network data by examining widely referenced and well-researched literature evaluation. A list of earlier reference studies is provided in Table 1, along with methodology and findings.

Table 1
List of Past Paper References with Methodology Used and Results.

Ref	Dataset	Methodology	Results
(Prabavathy et al., 2018)	NSL-KDD benchmark dataset	Online Sequential Extreme Learning Machine	The accuracy of the suggested strategy is 97.36%, with a lower false alarming rate of 0.37%.
(Diro and Chilamkurti, 2018)	NSL-KDD	Deep learning (DL) method	The results of the studies indicate that our distributed attack detection system is superior to deep learning-based centralized detection methods, with an accuracy of 96.5%.
(Bemessahel et al., 2019)	NSL-KDD and UNSW-NB15,	Feed-forward neural network (FNN), locust swarm optimization (LSO),	With an accuracy of 94.04%, the findings demonstrated the potential application of ENN (FNN-LSO) as a robust candidate solution for creating useful IDSs.
(Zhou et al., 2020)	UNSW-NB15 IBD	Variational Long short-term memory (VLSTM)	Based on an effective training method, the model's findings reveal that it dramatically improves feature extraction, lowers the false-positive rate, and increases detection accuracy.
(Pacheco et al., 2020)	IoT testbed Dataset	Artificial Neural Network (ANN)	The results demonstrate that the method is efficient in identifying both known and undiscovered assaults with high detection rates (over 90%) and low false-positive alerts (under 3.3%), as well as having little execution time, memory, and CPU use overhead.
(Aliyu et al., 2022)	UNSW-NB15 IBD	Long Short Term Memory (LSTM)	The results show that the model has an accuracy of up to 98.8%. We also observed a 10% reduction in the fog node's energy consumption as compared to before we installed a neural network on it.
(Labiod et al., 2022)	UNSW-NB15 data set	Variational Auto Encoder and Multilayer Perceptron (VAE-MP)	The results demonstrate that the proposed system can detect various attack types, such as DDoS attacks, with high detection rates (99.98%) and low false alarm rates (below 0.01%). It can also accurately describe typical activity inside fog nodes.

3. Data Collection

3.1. Exploratory Data Analysis EDA

This section will discuss several machine learning principles that may be used for anomaly Detection in Fog Computing and conduct exploratory data analysis. Data Scientists frequently utilize exploratory data analysis (EDA) (Lohani et al., 2022) while examining and researching data sets, summarizing the significant properties of data to the visualization technique. It assists the Data Scientist in discovering Data Patterns, detecting anomalies, hypothesis testing, and making assumptions. So, it is a strategy that helps the Data Scientist determine the best ways to change the supplied data source to obtain the desired result. Analysis and observations, we make future judgments about what features to bring into the dataset when training or what important characteristics to remove to improve the model's accuracy.

3.2. Data Description

The UNSW-NB15 (Moustafa and Slay, 2015) dataset was used in the study. Ne work packet capture (pcap), a CSV data from the UNSW2015 dataset, is correctly categorized as either normal or attack-oriented. UNSW-NB15 is a network traffic dataset created by the Australian National University with separate categories for routine activity and harmful assaults. Indeed, traffic is divided into nine categories, many assaults, and a large selection of genuine normal activities. The total number of rows in the dataset is 257,673 records, each with 49 attributes and a class label. In this research, we will be limited to the binary class of this dataset. The dataset contains 45 attributes that aid in adequately categorizing the targets (Moustafa et al., 2019). The four CSV files containing UNSW-NB151.csv, UNSW-NB152.csv, UNSW-NB153.csv, and UNSW-NB154.csv contain a total of two million and 540,044 records as listed in Table 2.

Table 2
Dataset distribution.

CSV files	Descriptions
UNSW_NB15_features.csv	49 features with the class label
UNSW-NB15GT.csv	The name of the ground truth table
UNSW-NB15LIST EVENTS.csv	Name of the event list
UNSWNB15train/test.csv	Divide the data into 20% and 80% for training and testing.

3.3. Data Preprocessing

The initial stage in ML for model construction is data preprocessing. D a preprocessing is how ML algorithms may use raw data. In the actual world, data is irregular; preprocessing organizes disorganized information for the data visualization stage. D a preparation comprises data cleaning to prepare the data using a machine learning model. These are the conclusions we reached after employing preprocessing data procedures. Before starting this exploratory investigation, import libraries and define functions for charting the data with matplotlib. The data will determine which plots are and are not possible.

3.4. Data Cleaning

Data cleaning includes filling in missing values or removing rows with missing values, filtering noisy data, and addressing data discrepancies. For example, in this dataset, we check the null value shown in Fig. 2.

Train.isnull().sum()	
id	Ø
dur	Ø
proto	Ø
service	Ø
state	Ø
spkts	Ø
dbkts	Ø
sbytes	Ø
dbytes	Ø
rate	Ø
stt1	Ø
dtt1	Ø
sload	Ø

Fig. 2. Null value check

After going through each feature's characterization, we observed that the data needs standardization and normalization. Normalization ensures that no data is duplicated, that it is all kept in one location, and that any dependencies are logical.

3.5. Data Reduction

When the amount of information is significant, datasets become slower, more expensive to acquire, and more difficult to maintain effectively. Da a reduction aims to give a simplified visualization of the information in a data store. Before proceeding to the data visualization stage, we eliminated outliers from the set to better understand how the properties are represented, even though we included all samples of the selected 49 characteristics in the final training. Fig. 3 shows details of features.

3.6. Data Visualization

3.6.1. Univariate Analysis

The UNSWNB15train/test.csv dataset's normal and Attack sample counts were displayed using subplots. We can assess how many attacks there were compared to the average using Fig. 3. The left-side graph displays the training data distribution, while the right-side chart displays the distribution of the testing data. The graph below shows data separated into common, generic, exploits, fuzzes, DoS, Backdoors, Shellcode, Reconnaissance, and worms. In contrast, the chart above depicts the data distribution, i.e., normal vs. Attack.

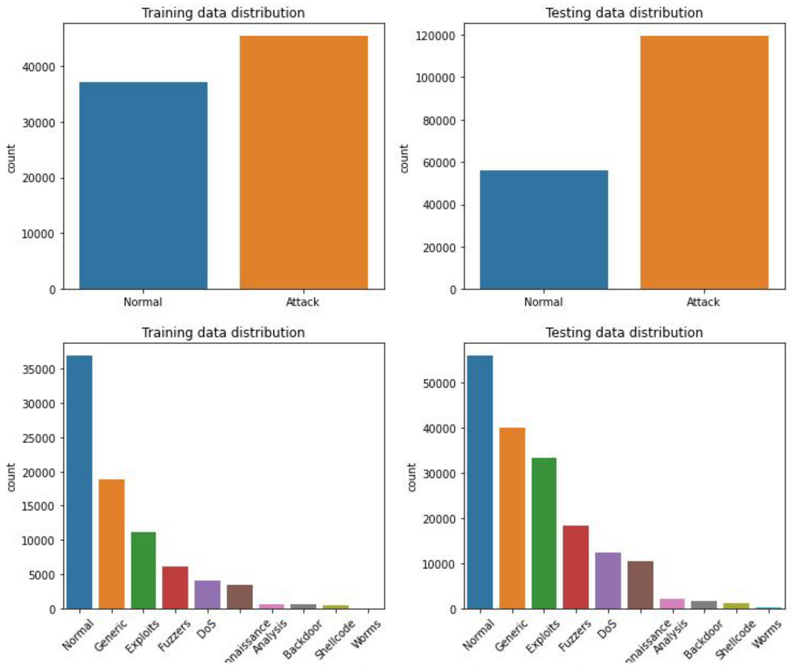


Fig. 3. Plot for Normal vs. Attack.

3.6.2. Bivariate Analysis

We simultaneously compare multiple variables using bivariate analysis, another practical method for understanding UNSW NB15 testing testing.csv data.

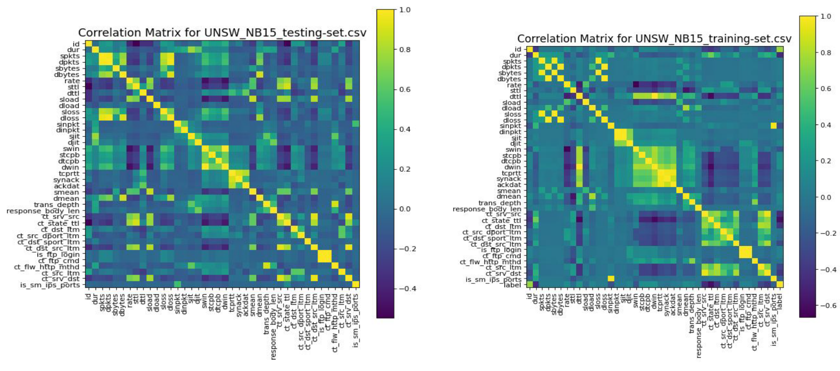


Fig. 4. Confusion matrix for testing and training set

Fig. 4 shows the heat map we created for testing and training the dataset (Moustafa and Slay, 2019). The dataset's numerical features span a wide range. It was crucial to normalize the values as a result. Except for the “id” and “label” columns, the “MinMaxScaler” has been used to normalize the numerical feature columns. The binary categorization of the characteristics into “normal” and “abnormal” classes as “0” for the normal class and “1” for the abnormal class was encoded into the “labels” column using LabelEncoder (). The binary dataset is composed of 61 columns once more. The attribute's nine subcategories for multi-class classification were Analysis, Exploits, Fuzzers, Backdoors, DOS, “Normal”, “Reconnaissance”, and “Worms”, as well as Generic. Fig. 5 shows a scatter plot of two numerical variables in the test and train cases. Although scatter plots can be used to visualize any data, they are beneficial for unevenly distributed data. The data points are plotted on a coordinate grid, and a line is drawn to link the dots, resulting in a scatter plot. The term "correlation" refers to the connection between two variables. The two variables have a high correlation if the markers are very close to forming a straight line in the scatter plot. The flyout of visualization kinds does not include the density plot because it is derived from a scatter plot visualization. A re-configured density map, however, is one of the visualizations in the Data in Analysis flyout if you choose two numerical columns.

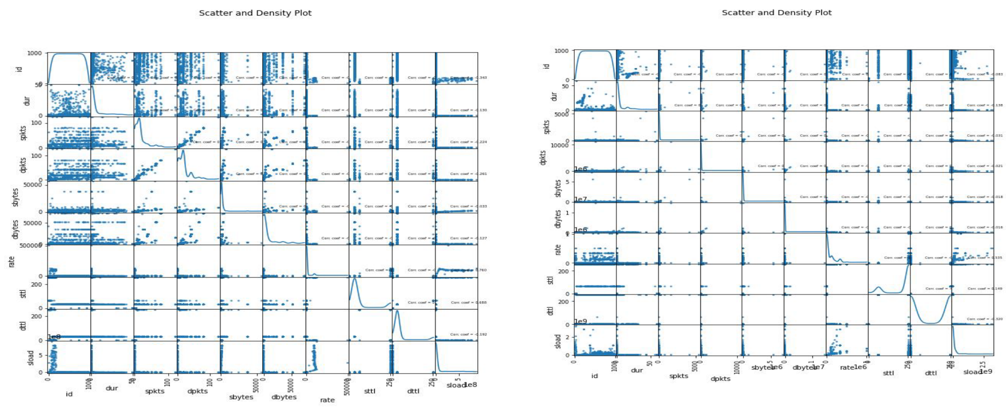
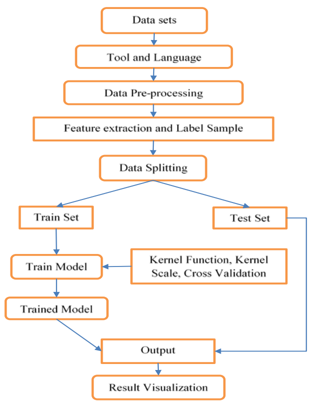


Fig. 5. Scatter and Density Plot.

4. Methodology

Fig. 6 shows the general organization of the machine learning architecture, with data collected from the data set as the first stage. The dataset is then put through a series of steps to collect information for preparation. First, the data are subjected to feature extraction and label sampling before being divided. Next, two sets must be created to split data: the trainset and the test set. The train set is then given to the model to train, and a trained model is produced after the model is trained using cross-validation, kernel scale, and kernel function. The data from the training model are immediately output together with the test set. The outcomes are then represented visually.



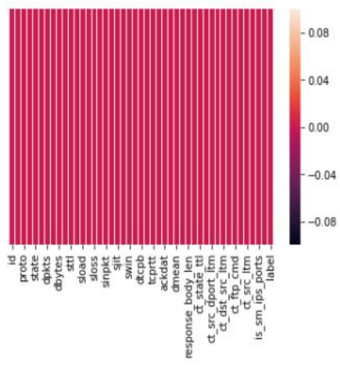


Fig. 6. Proposed Machine Learning Architecture

Fig. 7. Missing values Heat map

4.1. *Preprocessing*

In this stage, we use statistical approaches to clean the data and substitute values irrelevant to our experimental research. This is a requirement for all data analysis during the initial evaluation phase. We will then be able to translate information into a trustworthy format. Fig. 7 depicts the effects of missing values graphically. The findings reveal that no extraneous matters need to be removed. We discovered & recognized that our datasets are nearly clean throughout the preprocessing stage.

Now we have labelled the data set. Fig. 8 shows the Repartition of services, protocols, and attack types, as histograms use Pyplot and Seaborn libraries.

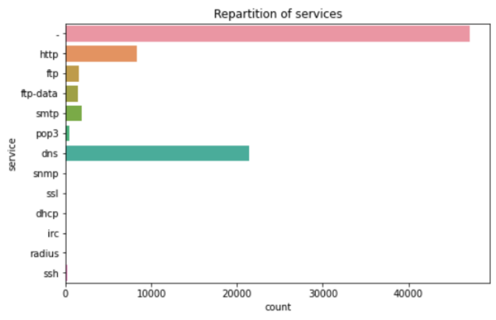


Fig. 8. Repartition of services

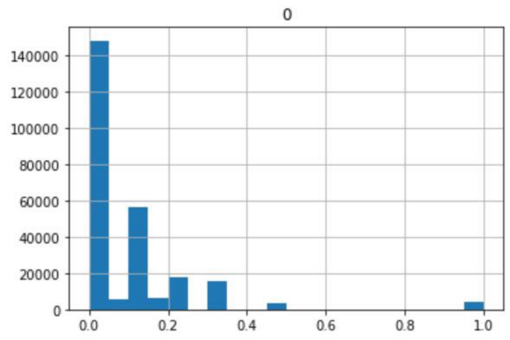


Fig. 9. Data normalization

4.1.1. Data Splitting

We need two separate sets of data to train/test the data. The test set serves as a fresh data collection for the model to predict the outcome in this Sklearn model evaluation package used.

4.1.2. Feature Engineering

In this input, UNSWNB-15 characteristics & properties are represented as structural columns. All algorithms require data characteristics with specific qualities to function correctly. Therefore, feature engineering aims to create an input data set that meets the requirements of ML models. In conclusion, we begin by converting all categorized traits into similar numerical labels.

4.1.3. Data Normalization

Normalization is used during data preparation to adjust the values of quantitative columns in a UNSWNB-15 to use a similar scale when the attributes in the data have varying ranges. A massive gap among various sizes of characteristic data within the dataset might cause issues such as sluggish model training and minimal accuracy increase; so, to tackle this issue, the MinMaxScaler was used to shift the dataset into the band of (0,1) as follows:

$$x' = x - xmin \,/\, xmax - xmin \tag{1}$$

After normalization, we got the data shown in Fig. 9.

4.2. ML Model

The advancement of the human experience led to the creation of many technologies. These devices simplify human existence by facilitating travel, industry, and computing, among other necessities. It is used to train computers to use more effective data management techniques. Numerous mathematicians and programmers use several strategies to discover the answer to this challenge using enormous datasets. The discipline of computer science that allows computers to learn from examples and experience is known as ML. Anomaly detection systems based on ML are the construction of a ground-breaking new anomaly detection model to identify irregular network traffic based on learning algorithms that may signal an intrusion attempt. Several ML algorithms for creating a mathematical model based on sample data provide the capacity for computers to make decisions without being explicitly programmed. There are three kinds of machine learning algorithms depending on the form of training supervision. There are four types, which are mentioned below in Table 3.

Table 3
Machine Learning Types

ML type	Description
Supervised learning	Refers to techniques for learning that use the support of a supervisor. It comprises labelled sample data that facilitates the algorithm's transition from input to output and its capacity for learning and prediction.
Unsupervised learning	Refers to the process of examining data without labels. Al o referred to as clustering. It resembles an independent learning process.
Semi-supervised learning	An approach is a learning technique used in ML that combines a significant amount of unlabelled data with a small amount of labelled data.
Reinforcement learning	The discipline of machine learning is dependent on agents, actions, states, rewards, and the surrounding environment.

In this research, we use four machine learning algorithms: linear regression, Ridge classifier, SGD, and ensemble learning techniques to the dataset and measure their accuracy, precision score, recall, and f1-score. These are the categorization models. These rating criteria are derived from the confusion matrix: genuine positives, true negatives, and false positives. Calculated false negatives for a classification algorithm compared to a validation data set display a confusion matrix representation.

4.3. Linear Regression

The dependent variable, which we are attempting to forecast or predict, or the independent variable(s), which serve as the input variable(s) for the prediction, are assumed to have a linear association in a predictive model known as linear regression (Shipe et al., 2019). Therefore, only one independent/ input variable might be utilized in simple linear regression to forecast the dependent variable. It is organized as follows in Eq. (2):

$$A = A1 + M \times X \tag{2}$$

where the above variables stand as:

- A = Dependent variable
- A1 = Constant
- M = Slope of the regression line
- X = Independent variable

There could be more than one independent variable and a dependent variable. Therefore, we may use multiple linear regressions for these sorts of models with the following procedure, as shown in Eq. (3):

$$A = A1 + M \times X1 + M * X2 + \cdots \tag{3}$$

4.4. Ridge Classifier

An approach for analysing linear discriminant models using ML is called ridge classification. It is a regularization where overfitting is avoided by penalizing model coefficients. Overfitting is a common issue in ML, where a model grows overly complex and picks up noise in the data instead of the underlying signal. When dealing with new data, this could lead to subpar generalization performance. Ridge classification addresses this problem by including a complexity-discouraging penalty element in the cost function. As a result, the model is more adaptable to new data (Tertytchny et al., 2020). The cost-function penalty term can be used to implement the ridge approach by discouraging complexity. The squared coefficients of the model's parts often make up the penalty term. As a result, overfitting is avoided, and the coefficients remain modest. To alter the degree of regularization, adjust the penalty term. Move regularization and lower coefficient values result from a more significant penalty. When there are training data, this can be helpful. Eq. (4) shows the ridge classifier:

$$\mathrm{Y} = \mathrm{XB} + \mathrm{e} \tag{4}$$

In contrast to Logistic Regression, the Ridge classifier's loss function is not a cross-entropy loss. Instead, mean square loss with L2 penalty serves as the loss function. The following methods for solving binary classification issues employ the Ridge regression technique:

- First, change the stated variable as necessary between +1 and -1.
- Train a Ridge model using an L2 regularization (ridge) penalty term and a mean square loss function.
- The anticipated class label is -1 if the expected value is less than zero; otherwise, the scheduled class label is +1.

4.5. Ensemble Learning

4.5.1. Ensembles and Boosting

Individual ML models or ensembles of models can be adapted to data. An ensemble is a group of basic unique models that, when combined, form a new solid system (Al-Hashedi et al., 2022).

4.5.2. Boosting Algorithm

Boosting algorithms, in contrast to specific other ML techniques, work to improve predicting accuracy by creating a succession of flawed model types, each of which fixes the flaws of the one before it. The boosting method is seen in action in Fig. 10. There are many models with various weights.

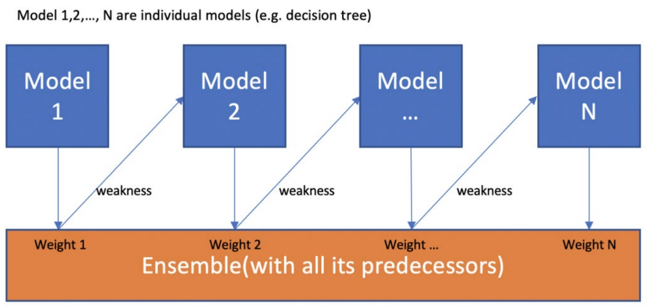

Model 1,2,..., N are individual models (e.g. decision tree)
Model 1
Model 2
Model ...
Model N
weakness
weakness
weakness
Weight 1
Weight 2
Weight ...
Weight N
Ensemble(with all its predecessors)

Fig. 10. Working of Boosting algorithm

4.5.3. AdaBoost

Adaptive Boosting, or AdaBoost, was the first realization of boosting that gained widespread use. Ad Boost's learning methods are id3 with a singular split, sometimes known as judgment stumps due to their shorter length. Ad Boost functions by weighing the observed, giving more weight to cases that are difficult to identify and less to those already well-classified. New weak trainees are introduced one at a time to concentrate their instruction on the increasingly challenging patterns. Forecasts are based on a democratic majority of the inadequate learners' predictions, graded based on personal accuracy. Ad Boost was the most effective version of the AdaBoost method for binary classifier tasks. The given Eq. (5) is for AdaBoost:

$$H(x) = P_i,\ \text{if} \sum_{i=1}^{t} h_i^j(x) > \frac{1}{2}\sum_{l=1}^{m}\sum_{i=1}^{t} h_i^l(x)\,. \tag{5}$$

AdaBoost finds missing-classified data sets in each iteration, raising their weights (and, in a sense, decreasing the consequences of valid points), so the following classification will pay more attention to getting them right. The graph below shows how weights improve the output of a simple decision problem (a tree with depth 1). The problem has been identified; the next goal is to determine how to integrate the models to make the ensemble more significant over time.

4.5.4. Boosting Stochastic Gradients

Subsampling the training data & training individuals on randomized samples obtained by this subsampling is what stochastic gradient boosting is all about. It decreases the correlation across individualized learning scores, and combining low-correlation results yields a more substantial overall effect. Several types of stochastic boosting may be used: Before making any tree, sample the rows and columns.

4.6. SGD Classifier

During the training of ML models, the cost function is optimized using the optimization approach SGD. However, the most often used algorithm for large datasets, such as gradient descent, takes a while to converge. This is where SGD, a gradient descent, comes into play. Us ng the Perceptron ML method, ideas of SGD are illustrated in (Ketu & Mishra, 2020). It is an iterative approach, meaning it runs over the training data repeatedly and modifies the model parameters slightly each time to reduce error. Below, equations 6 and 7 show the equation for the linear model and general hypothesis.

$$\min_{w}\left\{\frac{1}{N}\sum_{n=1}^{N} l\left(w' x_n, y_n\right)\right\} := f(w) \qquad \text{(linear model)} \tag{6}$$

$$\min_{w}\left\{\frac{1}{N}\sum_{n=1}^{N} l\left(h_w(x_n), y_n\right)\right\} := f(w) \qquad \text{(general hypothesis)} \tag{7}$$

To achieve anomaly Detection in Fog Computing Architectures using IoT, linear regression is adopted to classify to detect anomaly detection. The training and testing of all the classifiers, including Linear regression, the proposed ensemble model, and the boosting approaches (SGD), are performed.

4.7. Model Evaluation

We employed four of the most dependable machine-learning techniques to classify the data. First, we measured each classification model's accuracy, precision score, recall, and f1 score. These evaluation parameters are created using the confusion matrix, a measure of true positives, true negatives, false positives, and false negatives (fn) for a binary classifier computed over a validation data set. The models with the lowest validation loss were selected and then used to classify the data in the test set to assess the classification skills of our models. A confusion matrix in Fig. 11 is used to show and evaluate the classification results for each model.

		Actual	
		Positive	Negative
Predicted	Positive	**True Positive**	**False Positive**
	Negative	**False Negative**	**True Negative**

Fig. 11. True vs. False negative/positive.

By constructing confusion metrics that gauge the effectiveness of anomaly detection per class using the values of the matrix, we further assess the optimal model for each neural network. Accuracy is the percentage of accurate predictions across all samples (O'Reilly et al., 2014). Precision is the proportion of genuine positives to all positives; Recall is the percentage of precise predictions made over all pertinent pieces. The harmonic mean of recall and accuracy is the F1 score.

Accuracy: It measures how frequently the classifier makes accurate predictions by dividing the number of correct predictions by the total number of guesses.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{8}$$

Here, Eq. (8) relates to an equation for accuracy, which expresses the proportion of correctly classified data instances to all other data instances. If the dataset is unbalanced, accuracy might not be a good metric (both negative and positive classes have a different number of data instances).

Precision: Proportion of anticipated positives that are positive.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

The precision model is shown in Eq. (9). A good classifier's precision should preferably be 1 (high). On y when the numerator and denominator are equal, or when TP = TP + FP, precision becomes 1, which also implies that FP is zero. The accuracy value drops as FP rises because the denominator value exceeds the numerator.

Recall: The fraction of true positives successfully identified.

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

The recall equation is shown in Eq. (10), where recall on a classifier should ideally be 1 (high). When the numerator and denominator are the same, as in TP = TP + FN, recall becomes one, implying that FN is also zero. As FN increases, the denominator value exceeds the numerator, and the recall value decreases. Conversely, as FN increases, the denominator value rises above the numerator, and the recall value falls.

F1 score: The harmonic means of recall and precision.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

The F1 Score formula is presented in equation 11. The F1 score is 1 when precision and recall are both 1. The F1 score can only increase when precision and memory are both excellent. The F1 score, the harmonic mean of recall and precision, is a more helpful indicator than accuracy.

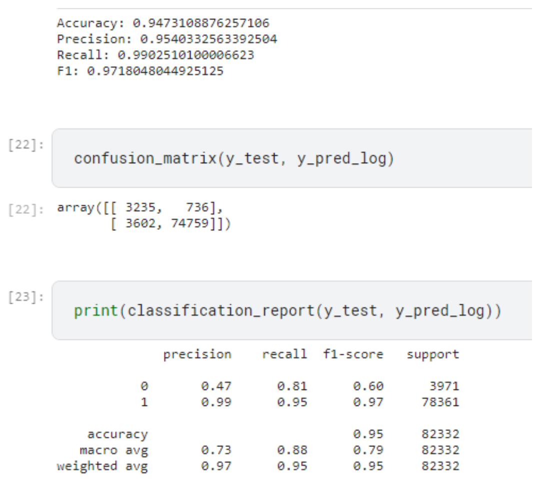
5. Results

5.1. Linear Regression

Using the Scikit-learn library, the training set was used to build a linear regression classification model for Normal or Attack behaviour. The model performance accuracy against the testing set was 94%, the confusion matrix shown below in Fig. 12.

5.2 Ridge Classifier

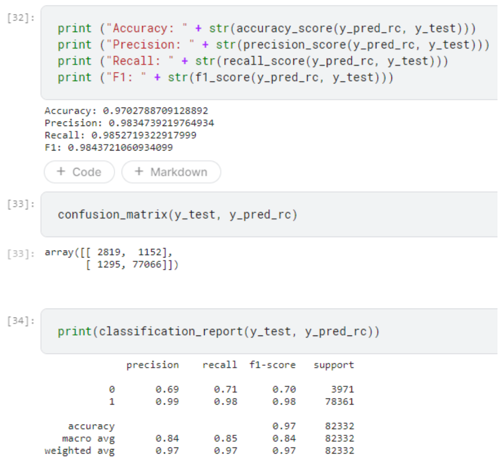
Using the Scikit-learn library, the training set was used to build a ridge classifier model for Normal or Attack behaviour. The model performance accuracy against the testing set was 97%, the confusion matrix shown below in Fig. 13.



```
Accuracy: 0.9473108876257106
Precision: 0.9540332563392504
Recall: 0.9902510100006623
F1: 0.9718048044925125
```

```
[22]: confusion_matrix(y_test, y_pred_log)

[22]: array([[ 3235,   736],
             [ 3602, 74759]])

[23]: print(classification_report(y_test, y_pred_log))

              precision    recall  f1-score   support

           0       0.47      0.81      0.60      3971
           1       0.99      0.95      0.97     78361

    accuracy                           0.95     82332
   macro avg       0.73      0.88      0.79     82332
weighted avg       0.97      0.95      0.95     82332
```

Fig. 12. Confusion Matrix of Linear Regression

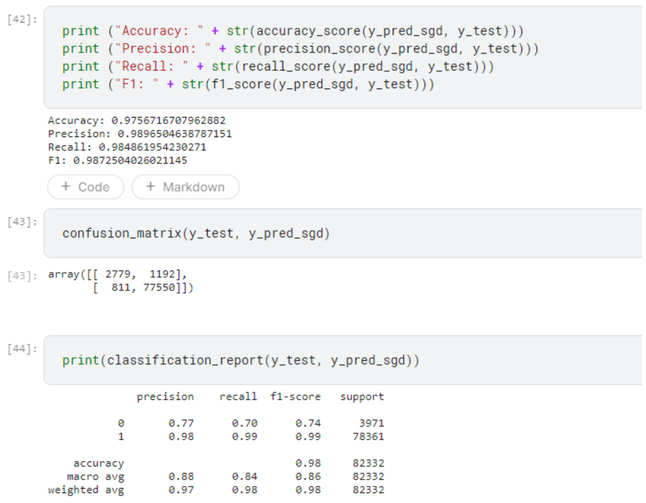


```
[32]: print ("Accuracy: " + str(accuracy_score(y_pred_rc, y_test)))
      print ("Precision: " + str(precision_score(y_pred_rc, y_test)))
      print ("Recall: " + str(recall_score(y_pred_rc, y_test)))
      print ("F1: " + str(f1_score(y_pred_rc, y_test)))

Accuracy: 0.9702788709128892
Precision: 0.9834739219764934
Recall: 0.9852719322917999
F1: 0.9843721060934099

[33]: confusion_matrix(y_test, y_pred_rc)

[33]: array([[ 2819,  1152],
             [ 1295, 77066]])

[34]: print(classification_report(y_test, y_pred_rc))

              precision    recall  f1-score   support

           0       0.69      0.71      0.70      3971
           1       0.99      0.98      0.98     78361

    accuracy                           0.97     82332
   macro avg       0.84      0.85      0.84     82332
weighted avg       0.97      0.97      0.97     82332
```

Fig. 13. Confusion Matrix of Ridge Classifier

5.3 *SGD Classifier*

Using SGDClassifier () libraries, the training set was used to build an SGD classification model for Normal or Attack behavior. The model performance accuracy against the testing set was 97% of the confusion matrix shown below in Fig. 14 and Fig. 15:



```
[42]: print ("Accuracy: " + str(accuracy_score(y_pred_sgd, y_test)))
      print ("Precision: " + str(precision_score(y_pred_sgd, y_test)))
      print ("Recall: " + str(recall_score(y_pred_sgd, y_test)))
      print ("F1: " + str(f1_score(y_pred_sgd, y_test)))

Accuracy: 0.9756716707962882
Precision: 0.9896504638787151
Recall: 0.984861954230271
F1: 0.9872504026021145

[43]: confusion_matrix(y_test, y_pred_sgd)

[43]: array([[ 2779,  1192],
             [  811, 77550]])

[44]: print(classification_report(y_test, y_pred_sgd))

              precision    recall  f1-score   support

           0       0.77      0.70      0.74      3971
           1       0.98      0.99      0.99     78361

    accuracy                           0.98     82332
   macro avg       0.88      0.84      0.86     82332
weighted avg       0.97      0.98      0.98     82332
```

Fig. 14. Confusion matrix for SGD

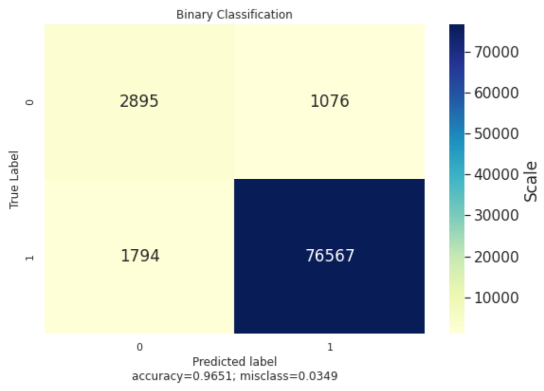


Fig. 15. Confusion Matrix for Ensemble Learning.

5.4 *Ensemble Learning*

To produce a master model, ensemble learning connects smaller models. Each base model forecasts the input tuple for every new piece of information.

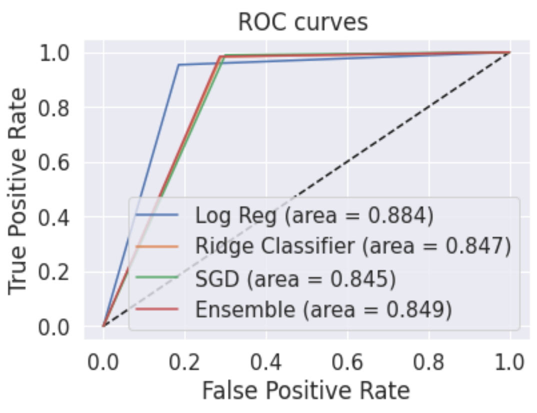


Fig. 16. ROC Curved

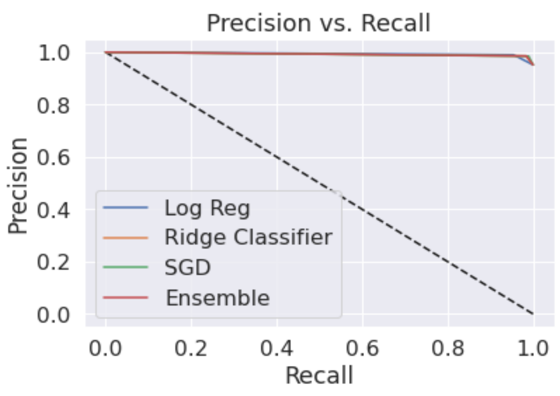


Fig. 17. Precision vs. Recall

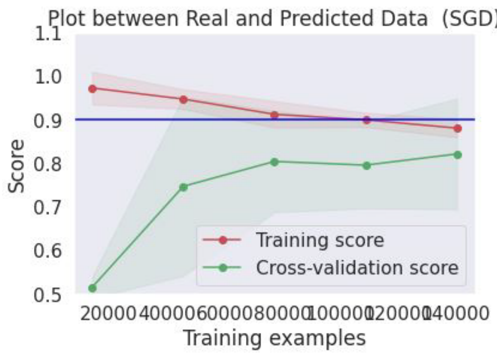


Fig. 18. Plot between Real and Predicted Data (SGD).

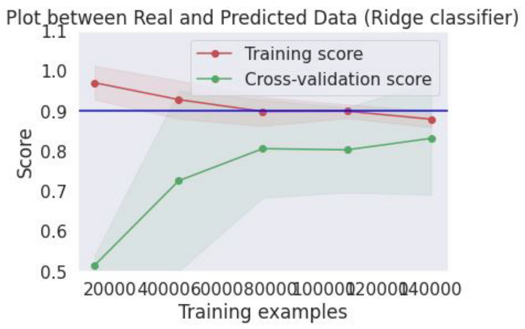


Fig. 19. Plot Between Real and Predicted Data (Ridge Classifier).

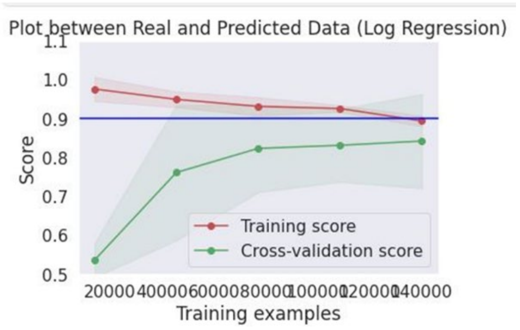


Fig. 20. Plot Between Real and Predicted Data Log Regression, SGD, and Ridge Classifier.

The class that receives the most significant number of votes wins in the master or aggregated model. Ba e models are capable of producing suggestions from any classification. The confusion matrix for ensemble learning is shown in Fig. 15. The projected label is 0.9651 percent accurate. Precision vs. Recall is also shown in Fig. 17. In contrast, the false positive rate for the log reg area classification, ridge classifier, SGD area, and ensemble area are all 0.888, 0.840, and 0.840, respectively. The learning curves for the logistic regression, ridge classifier, and SGD approaches are shown in Fig. 18, 19, and Fig. 20. The learning curve can be computed using samples from a training dataset and estimating the error rate over a validation dataset. The data points are perfectly fitted by the best-fit method. The model results for linear regression, Ridget classifier, SGD classifier, and ensemble learning are displayed in Table 4 below.

Table 4
Model results

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Linear regression	94%	95%	99%	97%
Ridget classifier	97%	98.30%	98.50%	98.40%
SGD classifier	96.60%	98.90%	98.40%	98.70%
Ensemble learning	96.50%	98.40%	98.50%	98.40%

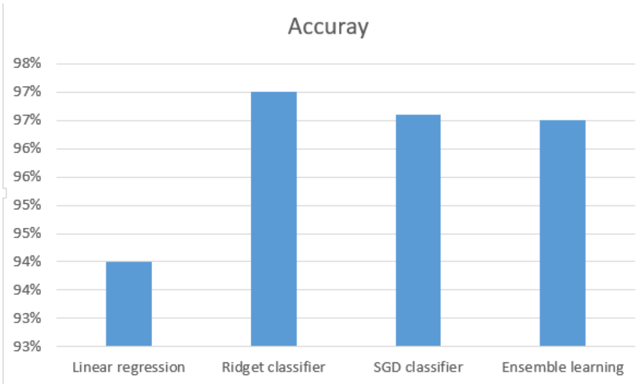


Fig. 21. Comparison of results.

Fig. 21 depicts the classification of data that used a binary classifier to determine the overall accuracy, sensitivity, error rate, specificity, and precision. The confusion matrix for ensemble learning, as shown in Fig. 15, which is a two-by-two matrix composed of these results, is shown in the analysis. The binary classifier generates results with labels like 0/1 and Yes/No. In addition, the classifier predicts all test data's true positive, true negative, false positive, and false negative cases. The leading indicators are the accuracy and error rate derived from the matrix. The accuracy, in this case, is 0.9651, and the error rate is 0.0349; the confusion matrix is constructed between the most real label and predicted label, using labels like 0/1, and it has a data scale. Fig. 16 displays the areas under the ROC curve for the logistic regression, ridge classifier, SGD, and ensemble algorithms to find plots showing TPR and FPR, considering the threshold value. This is done by finding the probability curve to distinguish the signal from the noise. The method performs better when the area under the curve has a more significant value. The Precision-Recall curve for our detection model is shown in Fig. 17. While identifying the classes, all curves completely or nearly fill the area. The learning curves for the SGD, ridge classifier, and logistic regression approaches are shown in Fig. 18, 19, and 20. One may calculate the learning curve by using the sample from a training dataset and estimating the error rate across a validation dataset. To fit the data points, the best-fit method has a zero-error rate. The size of the training instance has an impact on the model's error rate. The curve displays how well the training instance's modification affects the error, cross-validation score, and training score.

6. Discussion

The current work proposes an IoT network anomaly detection system based on fog. The use of fog nodes might effectively decentralize an IoT-based network with cloud architecture, according to anomaly detection implementation. The UNSW-NB15 dataset's design was used to construct the proposed model, which was then used to detect unexpected traffic in IoT networks. The most effective strategy would be ML-based because it enables computers to learn from and enhance previous data. Most earlier studies on detecting anomalies at the fog node employed conventional ML algorithms and got the best results. The research-developed tab transformer model outperforms currently used ML methods for investigating anomaly detection in the UNSW-NB15 dataset. This technology has proven superior to current deep neural networks for tabular data. Thus, we use it to address data imbalance concerns more intelligently. Given that our study combines binary and multi-classification and that the available dataset has an uneven distribution of items, accuracy might not be a helpful performance evaluation metric. Precision, recall, and F1 scores must be considered while comparing various ML algorithms. The outcomes of the binary classification (normal or abnormal) demonstrated that the Tab transformer model had successfully resolved the data imbalance issue. Although the data imbalance impacts the multi-class categorization of anomalies, the recommended model performs better than many previous attempts in the domain using the same dataset. Our work's decision tree model had 96.5% accuracy using correlation-based feature selection. SVM was created to discover binary and multi-class abnormalities in the dataset. However, the model needed to be corrected. Due to the uneven distribution of classes in the original data set, the result suggests that accuracy may not be the most significant performance measure for evaluating the model's performance. When used with the UNSW-NB15 dataset, the accuracy findings showed excellent performance for determining whether connected clients in an IoT network were carrying out ordinary tasks or launching an assault. Regression, the Ridge classifier, SVM, and ensemble learning were all tested, and it was found that the Ridge classifier, which has a 97% accuracy rate, is the best. The training set was utilized to create a linear regression classification model for normal or attacking behaviour using the Scikit-Learn toolkit. The model's performance versus the testing set in the confusion matrix was 94%. In ensemble learning, smaller models are linked to produce a master model. Additionally, the confusion matrix is built using labels like 0/1 and includes a data scale between the true label and the predicted label. The binary classifier produces results with labels like Yes or No. The matrix derives the accuracy and error rate as the primary indicators. Fig. 16 also shows the areas under the ROC curve for the ensemble, SGD, ridge classifier, logistic regression, and TPR and FPR plots while considering the threshold value. Fig. 16 shows the logistic regression, ridge classifier, SGD, and ensemble algorithms' regions under the ROC curve. To separate the signal from the noise, one finds the probability curve. The curve shows how the alteration of the training instance impacts the error, cross-validation score, and training score.

In conclusion, the suggested model more closely matches the data for detecting anomalies at the fog node. While performing multi-class classification procedures, a few issues need to be swiftly handled, such as the issue of data imbalance. In addition, the fact that the model must be implemented at the fog node necessitates a full investigation of the compute, power, and time requirements for anomaly prediction. We used the Tab transformer technique in our research to detect undesired traffic toward IoT devices. We provided a unique intrusion detection model that can be implemented at fog nodes using features from the UNSW-NB15 dataset. Table 5 below compares the many algorithms the authors have worked with, including XGBoost, Random Forest, MLP, CNN, BAT, BAT-MC, Linear Regression, Ridge Classifier, SGD, and Ensemble learning. Then, for each algorithm introduced in 2022 by Mohmand et al. (2022), accuracy, recalls, f1-score and precision are coming in at 90%, whereas for MLP, introduced in (Yin et al., 2022), precision coming in at 84.24%, recalls are coming in at 83%, accuracy is coming in at 82.25%, and f1-score is coming in at 83.3%. Tran et al. (2021) worked on the CNN, BAT-MC, and BAT algorithms, where the classification accuracy is below 80%. After everything was said and done, our model—which consists of Linear Regression, Ridge classifier, SDG, and ensemble learning—gave us the best results out of all the earlier algorithms employed by various authors.

Table 5
Performance Comparison

Algorithms	Accuracy	Precision	Recalls	F1-score
XGBOOST classifier, Random forest,	90%	90%	90%	90%
Multilayer perceptron (MLP)	82.25%	84.24%	83%	83.3%
CNN, BAT-MC, BAT	79%	79.3%	79%	79.2%
Linear regression, Ridge classifier, Sgd, and Ensemble learning	**97%**	**98.3%**	**98.5%**	**98.4%**

6 Conclusion

A fog-based anomaly detection approach for IoT networks is put out in an ongoing endeavour. The capacity to spot anomalies demonstrated how fog nodes might be effectively used to diversify a cloud-based IoT network. The presented model was created using the architecture of the UNSW-NB15 dataset, which was used to detect unusual traffic in IoT networks. Using correlation-based feature selection, the suggested detection method decreases the number of features for binary and multi-class datasets. However, there is still an imbalance in the test dataset. However, the intended Tab and ML transformers both produced good results. For multi-class detection tasks, our Tab transformer architecture outperformed conventional ML models by 97.22%, and for binary classification, it outperformed them by 98.35%. (A normal traffic vs. Normal). The importance of the correlation-based feature selection technique has also been made evident by comparing the performance of the suggested model to that of earlier models developed using the same dataset. We might build a lightweight anomaly detection model by combining the best possible combinations of IoT devices' varied memory sizes, network speeds, and battery life limits. Future computes complexity and time investigations will evaluate the performance of the proposed model using additional balanced IoT-based data sets.

The suggested methodology for finding anomalies still has several limitations despite performing better than others. It is possible to increase the computational complexity by employing novel approaches for data augmentation. By using customized models, the parameters can also be reduced. To increase precision even more, a few additional components can be included. Hu an-immune and lightweight methods are used. In addition to presenting a novel intrusion detection model that may be implemented at fog nodes to detect unauthorized traffic destined for IoT devices, we also offer the Tab transformer technique, which was used in our work. The UNSW-NB15 dataset's features are used in this model.

References

Ahmad, F. (2022). Deep image retrieval using artificial neural network interpolation and indexing based on similarity measurement. *CAAI Transactions on Intelligence Technology*, *7*(2), 200-218.

Al-Hashedi, A., Al-Fuhaidi, B., Mohsen, A. M., Ali, Y., Gamal Al-Kaf, H. A., Al-Sorori, W., & Maqtary, N. (2022). Ensemble Classifiers for Arabic Sentiment Analysis of Social Network (Twitter Data) towards COVID-19-Related Conspiracy Theories. *Applied Computational Intelligence and Soft Computing*, *2022*(1), 6614730.

Aliyu, F., Sheltami, T., Deriche, M., & Nasser, N. (2022). Human immune-based intrusion detection and prevention system for fog computing. *Journal of Network and Systems Management*, *30*(1), 11.

Alneyadi, S., Sithirasenan, E., & Muthukkumarasamy, V. (2016). A survey on data leakage prevention systems. *Journal of Network and Computer Applications*, *62*, 137-152.

Benmessahel, I., Xie, K., Chellal, M., & Semong, T. (2019). A new evolutionary neural networks based on intrusion detection systems using locust swarm optimization. *Evolutionary Intelligence*, *12*, 131-146.

Bezerra, V. H., da Costa, V. G. T., Barbon Junior, S., Miani, R. S., & Zarpelão, B. B. (2019). IoTDS: A one-class classification approach to detect botnets in Internet of Things devices. *Sensors*, *19*(14), 3188.

Bustamante-Bello, R., García-Barba, A., Arce-Saenz, L. A., Curiel-Ramirez, L. A., Izquierdo-Reyes, J., & Ramirez-Mendoza, R. A. (2022). Visualizing street pavement anomalies through fog computing v2i networks and machine learning. *Sensors*, *22*(2), 456.

Daoud, M., Dahmani, Y., Bendaoud, M., Ouared, A., & Ahmed, H. (2023). Convolutional neural network-based high-precision and speed detection system on CIDDS-001. *Data & Knowledge Engineering*, *144*, 102130.

De Souza, C. A., Westphall, C. B., & Machado, R. B. (2022). Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments. *Computers & Electrical Engineering*, *98*, 107694.

Dhanabal, L., & Shantharajah, S. P. (2015). A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, *4*(6), 446-452.

Dhirani, L. L., Mukhtiar, N., Chowdhry, B. S., & Newe, T. (2023). Ethical dilemmas and privacy issues in emerging technologies: A review. *Sensors*, *23*(3), 1151.

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, *82*, 761-768.

Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A. Y., & Ranjan, R. (2019). A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. *IEEE Transactions on Network and Service Management*, *16*(3), 924-935.

Gupta, A., & Namasudra, S. (2022). A novel technique for accelerating live migration in cloud computing. *Automated Software Engineering*, *29*(1), 34.

Jaiswal, R., Chakravorty, A., & Rong, C. (2020, August). Distributed fog computing architecture for real-time anomaly detection in smart meter data. In *2020 IEEE sixth international conference on big data computing service and applications (BigDataService)* (pp. 1-8). IEEE.

Janarthanan, T., & Zargari, S. (2017, June). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)* (pp. 1881-1886). IEEE.

Kayan, H., Majib, Y., Alsafery, W., Barhamgi, M., & Perera, C. (2021). AnoML-IoT: An end to end re-configurable multi-protocol anomaly detection pipeline for Internet of Things. *Internet of Things*, *16*, 100437. https://doi.org/10.1016/j.iot.2021.100437.

Ketu, S., & Mishra, P. K. (2020). Performance analysis of machine learning algorithms for IoT-based human activity recognition. In *Advances in Electrical and Computer Technologies: Select Proceedings of ICAECT 2019* (pp. 579-591). Springer Singapore.

Kochhar, S. K., Bhatia, A., & Tomer, N. (2023). Using Deep Learning and Big Data Analytics for Managing Cyber-Attacks. In *New Approaches to Data Analytics and Internet of Things Through Digital Twin* (pp. 146-178). IGI Global.

Labiod, Y., Amara Korba, A., & Ghoualmi, N. (2022). Fog computing-based intrusion detection architecture to protect iot networks. *Wireless Personal Communications*, *125*(1), 231-259.

Lohani, K., Bhardwaj, P., & Tomar, R. (2022). Fog Computing and Machine Learning. In *Fog Computing* (pp. 133-151). Chapman and Hall/CRC.

Lu, Y., & Da Xu, L. (2018). Internet of Things (IoT) cybersecurity research: A review of current research topics. *IEEE Internet of Things Journal*, *6*(2), 2103-2115.

Manimurugan, S. (2021). IoT-Fog-Cloud model for anomaly detection using improved Naïve Bayes and principal component analysis. *Journal of Ambient Intelligence and Humanized Computing*, 1-10.

Mohmand, M. I., Hussain, H., Khan, A. A., Ullah, U., Zakarya, M., Ahmed, A., ... & Haleem, M. (2022). A machine learning-based classification and prediction technique for DDoS attacks. *IEEE Access*, *10*, 21443-21454.

Moustafa, N., & Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)* (pp. 1-6). IEEE.

Moustafa, N., Hu, J., & Slay, J. (2019). A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, *128*, 33-55.

O'Reilly, C., Gluhak, A., Imran, M. A., & Rajasegarar, S. (2014). Anomaly detection in wireless sensor networks in a non-stationary environment. *IEEE Communications Surveys & Tutorials*, *16*(3), 1413-1432.

Pacheco, J., Benitez, V. H., Felix-Herran, L. C., & Satam, P. (2020). Artificial neural networks-based intrusion detection system for internet of things fog nodes. *IEEE Access*, *8*, 73907-73918.

Prabavathy, S., Sundarakantham, K., & Shalinie, S. M. (2018). Design of cognitive fog computing for intrusion detection in Internet of Things. *Journal of Communications and Networks*, *20*(3), 291-298.

Prasad, N. R., Almanza-Garcia, S., & Lu, T.T. (2009). Anomaly Detection. *ACM Computing Surveys, 14*(1), 1–22. https://doi.org/10.1145/1541880.1541882.

Shakeel, N., & Shakeel, S. (2022). Context-free word importance scores for attacking neural networks. *Journal of Computational and Cognitive Engineering*, *1*(4), 187-192.

Shipe, M. E., Deppen, S. A., Farjah, F., & Grogan, E. L. (2019). Developing prediction models for clinical use using logistic regression: an overview. *Journal of thoracic disease*, *11*(Suppl 4), S574.

Tertytchny, G., Nicolaou, N., & Michael, M. K. (2020). Classifying network abnormalities into faults and attacks in IoT-based cyber physical systems using machine learning. *Microprocessors and Microsystems*, *77*, 103121.

Tomer, V., & Sharma, S. (2022). Detecting IoT attacks using an ensemble machine learning model. *Future Internet*, *14*(4), 102. https://doi.org/10.3390/fi14040102.

Tran, N., Chen, H., Jiang, J., Bhuyan, J., & Ding, J. (2021). Effect of class imbalance on the performance of machine learning-based network intrusion detection. *International Journal of Performability Engineering*, *17*(9), 741.

Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications*, *111*(4), 2287-2310.

Xin, R., Liu, H., Chen, P., & Zhao, Z. (2023). Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework. *Journal of Cloud Computing*, *12*(1), 7.

Yin, Y., Jang-Jaccard, J., Xu, W., Singh, A., Zhu, J., Sabrina, F., & Kwak, J. (2023). IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. *Journal of Big Data*, *10*(1), 15.

Zhou, X., Hu, Y., Liang, W., Ma, J., & Jin, Q. (2020). Variational LSTM enhanced anomaly detection for industrial big data. *IEEE Transactions on Industrial Informatics*, *17*(5), 3469-3477.

© 2024 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).