International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Parallel Rank Based Multi-Class Ensemble Classification Framework on ISOT Cyber Threat Detection

Lakshmi Prasanna B. [1*], M. Saidi Reddy [2]

Submitted: 23/10/2023 **Revised:** 17/12/2023 **Accepted:** 24/12/2023

Abstract:A group of internet-connected compromised devices forming a network is called a Botnet. This network can consist of personal computers, servers, IoT devices, and mobile devices. Botnets are one of the most common network security threats, used for malicious activities such as data theft, spamming, collecting personal information from users, and launching Distributed Denial of Service (DDoS) attacks. The growing popularity of IoT and mobile devices has made them an attractive target for attackers, as they often have unpatched security vulnerabilities. In today's world, computer networks play a crucial role in the information and communication technology era, connecting heterogeneous devices for data communication and sharing. However, the large number of Internet-connected devices makes them vulnerable to massive security attacks. Most widely-used IoT devices lack security design, making them vulnerable to recent attacks that exploit these weaknesses and recruit the devices to cause severe harm. Parallel multi-class classification refers to the process of performing classification tasks simultaneously on multiple classes or categories of data using parallel computing techniques. In traditional multi-class classification, a model is trained to classify data into one of several mutually exclusive classes. However, in some scenarios, it may be advantageous to perform these classifications in parallel, especially when dealing with a large number of classes or when speed and efficiency are crucial. Parallel multi-class classification can be implemented using parallel processing or distributed computing frameworks to train and evaluate multiple classifiers concurrently.

Keywords: *Multi-class classification, data filtering, outlier detection, cyber-attack detection.*

1. Introduction

Intrusion represents an abnormal activity that poses a threat to computer or network systems. Such intrusions can be categorized as either internal or external. Internal intrusions originate from within the target network and involve acquiring unauthorized access to harm the system. On the other hand, external intrusions come from outside the network, where individuals gain illicit access to the network's information. Malicious and compromised nodes typically initiate internal attacks, while third parties initiate external ones.IoT, short for the Internet of Things, comprises a network of interconnected devices, objects, and systems via the internet. It interacts with its environment both internally and externally, detecting and responding to environmental changes[1]. IoT introduces innovative approaches to various aspects of life, ultimately enhancing the quality of human living standards. It enables real-time or remote communication between objects. With IoT's integration, the environment becomes smarter and can connect to virtually any device at any time. IoT collects and interprets data from a variety of sensors and devices, transmitting it wirelessly to smartphones or computers. Its applications are diverse, impacting areas such as household appliances, energy efficiency, environmental monitoring, and commercial enterprises, all contributing to a more comfortable living environment. IoT finds utility in distribution, logistics, automation, robotics, and surveillance systems, enhancing their modernization.IoT plays a pivotal role in enhancing people's lives by facilitating the exchange of data among a network of devices, thus creating a conducive environment for application development while anticipating market trends in advance. In today's fast-paced world, IoT has the capability to meet people's needs and desires.IoT technology also supports Industrial Internet of Things (IIoT), enhancing manufacturing operations and industrial processes[2]. It captures, shares, and interprets data across a distributed system interconnected by communication networks, enabling quicker decision-making. For instance, IIoT can predict and rectify equipment flaws in industrial settings before they lead to breakdowns, thereby saving time, resources, and costs. These small devices are commonly employed within conventional Industrial Control Systems (ICS).The Internet of Medical Things (IoMT) offers numerous advantages for both patients and healthcare professionals. Many IoMT devices are designed to meticulously monitor critical patient parameters, enabling physicians to make more accurate diagnoses and develop personalized treatment plans based on real-time data. IoMT facilitates remote healthcare and guidance, allowing devices to collect data from patients' homes and securely transmit it to healthcare providers, eliminating the need for frequent clinic visits. Patients can now monitor their health in real-time, rather than relying solely on annual check-ups.While IoMT devices provide numerous benefits such as convenience, enhanced patient care, and cost reduction, they also bring certain risks, making security a critical concern[3-5]. One of the most significant threats is the potential theft of sensitive medical information, which, if exposed, could be embarrassing or harmful to

[1] *Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad, Telangana, India*
Assistant Professor, Department of IT, AnuragUniversity, Hyderabad, Telangana, India.
lakshmiprasanna.byrapuneni@gmail.com
[2] *Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad, Telangana, India.*

individuals. Hackers are not solely interested in consumer health data.In the context of data analysis, there are various strategies for determining the best features to split instructional datasets, including the Gini index and Information Gain. Decision trees are commonly used to classify unknown samples based on distinctive features. These trees are constructed by creating a path from the root node to a leaf node, following a top-down recursive division and conquer strategy. Decision trees offer advantages such as intuitive knowledge representation, accurate classification, and ease of implementation[6]. However, they have limitations, including a bias toward features with higher values when dealing with continuous attributes with varying strata. Support Vector Machine (SVM): SVM is a technique used to classify data by identifying the optimal line that separates data points into different categories. This separation is achieved by maximizing the margins between the categories, resulting in equal spacing between the lines. When provided with labeled training data, the SVM algorithm constructs an ideal hyperplane that can be used to label new data points. Random Forest: The Random Forest system consists of multiple trees that are constructed randomly and then collectively used to vote for a category. The category with the most votes is selected as the final classification. Although Random Forest methods are derived from decision trees, they differ significantly. In traditional Decision Trees, a series of rules is generated and applied to classify new data when a test dataset is fed into the system. On the other hand, Random Forest employs decision trees to create a set of rules and then selects a category, effectively addressing overfitting issues. K-Nearest Neighbors (KNN): KNN represents a method of categorizing samples based on their proximity in an n-dimensional space. When classifying an unknown sample, the algorithm identifies k training samples in the pattern space that are nearest to the unknown sample and assigns the category based on the majority among its k closest neighbors. KNN is considered a slow learner because it retains all training data and constructs a classification system only when needed. This method is suitable for use with labeled, noise-free, and small datasets.Naïve Bayes: Naïve Bayes Classification estimates the likelihood of an event based on prior knowledge of that event, making it particularly relevant in scenarios like DoS threat detection using network traffic data. It relies on Bayes' theorem and strict independence assumptions, meaning the presence or absence of one attribute does not influence the presence or absence of another. Bayesian classifiers perform well in large databases in terms of execution time and accuracy, especially when used in conjunction with big data[7].

Logistic Regresion: Logistic regression is a mathematical technique used to analyze data, where the outcome is determined by one or more independent factors. It aims to find the best-fit line between dependent and independent variables. Logistic regression can be used for projecting values, such as predicting income based on various factors. It is also used for forecasting trends when time is considered an independent factor. Machine learning technologies offer valuable insights into safety across various practical challenges and scenarios. Automatic machine learning methods provide a systematic and adaptable approach to addressing the complexities of emerging production connectivity platforms, ensuring stability and reliability[].For instance, SCADA platforms often exhibit repetitive communication patterns, with a limited set of commands repeated frequently. Such patterns can be used by ML techniques to build models for detecting anomalies and improving cybersecurity. However, the integration of IoT, cloud technology, and other technologies in SCADA networks can pose significant security challenges, as discussed in [9].Another approach involves a hazard analysis technique prioritizing confidentiality, security, adaptability, and consistency in the context of IT and IIoT sectors [10]. This method assesses the likelihood of a hazard exploiting vulnerabilities and its impact on various stakeholders. Additionally, [11] presents a comprehensive examination of ML algorithms and their application to IoT data, emphasizing the need for suitable information prototypes and the challenges of scale and real-time data processing.Cybersecurity in IIoT is a critical concern, [11] discusses effective cybersecurity options for IIoT design, considering data breaches and security measures.[12] focuses on deep learning methods for intrusion detection, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), highlighting the importance of accurate detection of intrusions in wideband communication systems.This type of Intrusion Detection System (IDS) overcomes the limitations of Signature-Based Intrusion Detection Systems (SIDS) by focusing on network behavior. Anomaly Intrusion Detection System (AIDS) incorporates machine learning, knowledge-based techniques, and statistical-based techniques to create a robust intrusion detection system. AIDS identifies anomalies, intrusions, or unknown threats in a network by detecting even slight deviations from the typical network behavior, triggering alerts when danger is detected. The development process of AIDS involves both a training phase and a testing phase. During training, AIDS learns the network's normal behavior, and during testing, it uses a dataset with unseen intrusions to detect and classify threats. AIDS is particularly advantageous as it can detect zero-day attacks, making it a valuable addition to network security. Additionally, it has the capability to identify internal threats. In a Distributed Intrusion Detection System (DIDS), multiple Intrusion Detection Systems (IDS) are employed to monitor an extensive network. Each node in the network is equipped with an IDS agent, and these IDS agents collaborate with each other or a centralized server to observe network activities. The IDS agents perform event scrutiny, intrusion detection, and take appropriate actions when necessary. DIDS offers the benefit of avoiding a single point of failure and is highly scalable. It can be implemented individually, where each node monitors the behavior of others and alerts a reporting agent if unusual behavior is detected. Alternatively, in a cooperative approach, nodes collaborate to determine whether a monitored node is under attack.

2. Related Works

Smart metering applications encompass a variety of use cases, including monitoring the stock of goods in the retail sector, controlling electricity usage in smart grids, and monitoring water, oil, and gas levels. Smart meters play a crucial role in improving the efficiency of solar energy plants by adjusting the angles of solar panels. However, these applications face both physical and cyber threats, with analog meters being more vulnerable to physical attacks. In the context of smart homes, electronic devices are interconnected with smart meters in the Smart Home Area Network, enabling cost and load management. Ensuring the security of data collected from these devices is crucial, as attackers may attempt to tamper with data, resulting in financial losses[13].Smart medical care systems rely on three essential components: hospital organization, lab administration, and clinical findings. By embedding sensors and actuators in medical devices and instruments accessible via the internet, smart

hospitals are created. This connectivity facilitates the communication of information between doctors, patients, test results, and medical devices. In the retail sector, IoT applications are developed to monitor and manage goods in warehouses and enhance the shopping experience. However, retail companies face security challenges, with adversaries attempting to manipulate product information to boost sales or gain unauthorized access to customer data, such as credit card details.Smart homes leverage IoT technology to remotely control and regulate electronic appliances, enhance security through sensors on doors and windows, and monitor energy and water supply. Logic-based security techniques, as proposed by [14], can enhance safety by comparing user activities with standard actions to detect intrusions.IoT applications in the smart environment domain cover various areas, including fire detection, snow monitoring, landslide prevention, earthquake detection, and pollution monitoring. The data gathered by these applications is crucial for government organizations. However, false negatives and false positives in smart environment applications can have disastrous consequences, emphasizing the need for robust data integrity and security measures.Booters, a form of Distributed-Denial-of-Service (DDoS) threat, are employed by malicious users to disrupt network speeds. The dataset related to booter threats comprises nine different types of threats executed against IP addresses, each consisting of data packets exceeding 250 GB. The dataset is openly accessible, although individual packets are not labeled. Instead, various booter threats are grouped into files.The Botnet dataset combines existing datasets through overlay techniques, including ISOT, ISCX 2012, and CTU-13, capturing both normal and botnet behavior. It offers a testing dataset of 8.5 GB and a training dataset of 5.3 GB, both in packet form.The Canadian Institute for Cybersecurity provides the CIC Dos dataset, focusing on denial-of-service attacks at the application layer[16]. This dataset includes eight distinct DoS threats and captures network traffic data over a 24-hour period in packet form.The CIDDS-002 port scan dataset is derived from the base CIDDS-001 dataset, encompassing network flow data observed over two weeks. It includes port scan attacks and normal network behavior, with metadata available in a technical report.Ambusaidi et al. (2016) employ the Mutual Information (MI) feature selection technique to select optimal features for an intrusion detection system (IDS). They combine multiple classifiers, including the Least Square-Support Vector Machine (LS-SVM), to differentiate between various attack classes based on computed relevance scores between class labels and instances.[17] propose an anomaly-based IDS using Recurrent Neural Networks (RNN), a deep learning technique. They evaluate the RNN-based IDS model using the NSL-KDD intrusion detection dataset, conducting experiments for multiclass and binary classification scenarios. The IDS utilizes RNN's ability to provide feedback from past data to make real-time predictions.[18] developed an Anomaly Traffic Detection method utilizing Support Vector Machine (SVM), a supervised machine learning classification technique. This approach introduces a novel algorithm for estimating data instance entropy. It identifies deviations from the normal network behavior by setting a threshold value, with anomalies occurring when this threshold is exceeded. SVM serves as the classifier, and its effectiveness is enhanced through the application of Particle Swarm Optimization (PSO). The evaluation of this Anomaly Traffic Detection method employs datasets such as KDD CUP 99 and DARPA to classify various types of attacks. Feature selection and voting criteria are employed to select the most relevant attributes for classification, adhering to the min-max principle. This method excels in achieving high accuracy while maintaining a low error rate in identifying different attack varieties.[19] propose a unique intrusion detection system tailored for the IoT environment, leveraging deep learning technology. They highlight the challenges posed by numerous protocols in IoT platforms, which often encounter zero-day threats that deviate slightly from known cyber risks. Their approach involves a network intrusion detection scheme based on Conditional Variational Autoencoder (CVAE), which consolidates intrusion labels within the decoder. A key advantage of this model is feature reconstruction, making it suitable for IoT networks' network intrusion detection. It streamlines computational resources by training the technique in a single step. [20] introduce an anomaly-based intrusion detection system to mitigate cloud threats. They employ Binary-based Particle Swarm Optimization (BPSO) to select the most relevant instances and classify them using Support Vector Machine (SVM), fine-tuning SVM control parameters using Standard-based Particle Swarm Optimization (SPSO).[21] address security issues in the virtual network layer of cloud computing. They design a security scheme based on snort and various classifiers, such as decision trees, associative models, and Bayesian models. The intrusion detection system is deployed in each host, enabling both offline and real-time analysis.[22] develop the Online Intrusion Detection System Cloud System (OIDCS) to detect zero-day threats in online mode. They introduce the NeuCube spiking neural network architecture to OIDCS and employ the TBR algorithm, resulting in high accuracy.[23] create a packet scrutinization algorithm and normalized K-means with recurrent neural network (NK-RNN) to enhance trust authority, cloudlet controller, and virtual machine security. They introduce a one-time signature scheme for cloud data access, effectively detecting port scans and flooding attacks.[24] address security challenges in Unmanned Aerial Vehicle Networks (UAVs) using an agent-based self-protective method (ASP-UAVN) based on the Human Immune System (HIS). This method distinguishes secure routes from attack-prone UAVs, resulting in improved Packet Delivery Rate (PDR) and reduced False Positive Rate, False Negative Rate, and Packet Loss Rate (PLR) compared to other techniques.
[25] deploys an Adaptive Intrusion Detection System based on deep learning in drones to detect intruders. Self-Taught Learning (STL) maintains the True Positive Rate, using a multi-class support vector machine and the Deep-Q network algorithm for self-remediation.[26] introduce a Spline function-based intrusion detection system to mitigate host system threats. This Client-Server model employs various spline functions to enhance IDS performance, with B-Splines providing particularly strong results.[27] propose a Big data-based Network Intrusion Detection framework to address security issues in Vehicular Ad-hoc Networks (VANETs). The framework consists of network traffic detection and collection components, using Micro-batch data processing and Random Forest classification. Data normalization is achieved using [27] enhance security in Vehicular Ad-hoc Networks (VANETs) with a Spline-Based IDS merged with clustering, forming Knot Flow Classification (KFC) for improved attack detection. [28] introduce the Trust-Based Collaborative Intrusion Detection System (TBICDS), enabling vehicles to manage score tables of nearby vehicles and enhance security.

3. Proposed Framework

In this framework, a hybrid framework is developed on the

heterogeneous data in three phases as shown in figure 1. Cloud IoT Bot dataset is an important tool for detecting network intrusions in the Internet of Things (IoT) network. The dataset is a collection of data that represents the behavior of different devices connected to the network. The data includes information about the devices, such as their type, behavior, and communication patterns. This data is collected and stored in the cloud, where it can be analyzed to identify potential intrusions.The first step in analyzing the Cloud IoT Bot dataset is statistical outlier detection. Outliers are data points that are significantly different from the other data points in the dataset. Input Data:Input data refers to the raw information or observations that are provided to a system or algorithm for processing, analysis, or any other purpose. In the context of machine learning and data analysis, input data typically consists of features or variables that are used to make predictions, classifications, or gain insights. Input data can be structured (e.g., in a tabular format), unstructured (e.g., text, images, audio), or semi-structured (e.g., JSON). The quality and relevance of input data are critical factors in determining the performance of machine learning models.Data filtering is the process of selecting a subset of data from a larger dataset based on specific criteria or conditions. This is often done to reduce the size of the dataset or to focus on relevant information. Removing duplicates: Eliminating duplicate records from the dataset.Applying conditions: Selecting data points that meet certain criteria or conditions (e.g., selecting customers who made a purchase in the last month).Removing outliers: Excluding data points that are significantly different from the majority of the data, which can distort analysis or model training.Data filtering helps in cleaning and preparing data for further analysis or modeling by reducing noise and improving the quality of the dataset.Ranking involves ordering items or data points in a dataset based on a specific attribute or set of attributes. It assigns a numerical or ordinal position to each item, indicating its relative importance, value, or relevance within the dataset. Parallel multi-class classification refers to the process of performing classification tasks simultaneously on multiple classes or categories of data using parallel computing techniques. In traditional multi-class classification, a model is trained to classify data into one of several mutually exclusive classes. However, in some scenarios, it may be advantageous to perform these classifications in parallel, especially when dealing with a large number of classes or when speed and efficiency are crucial. Parallel multi-class classification can be implemented using parallel processing or distributed computing frameworks to train and evaluate multiple classifiers concurrently. Each classifier is responsible for classifying data into one of the classes, and the results are combined to make the final prediction as shown in figure 1.

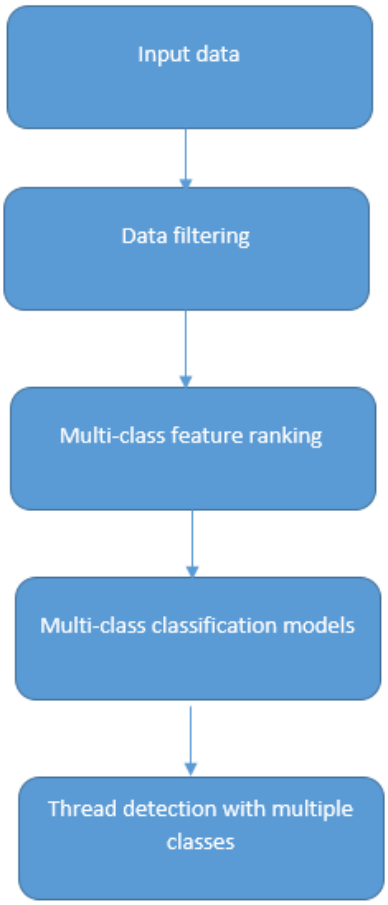

Fig 1: Multi-class parallel ranked based Classification Framework

Algorithm 1: Filling missing values and Data transformation approach

Filling missing values and Data transformation approach
1. Read BoT-IoT training dataset.
2. To each feature F in dataset D.
3. if(F[i]==Null)
4. then
5. end if
6. end loop
7. To each feature F in dataset D.
8. Compute non-linear data transformation as

$$NLG = F[i].log(F[i]).\sum_{r=0}^{N}\frac{1}{\sqrt{2\pi\, log(F[i])}}.e^{-|Max\{F[r]\}-F[i]|/2}$$

The given steps outline the process for handling missing values and transforming data in the BoT-IoT training dataset. Step 1 involves reading the BoT-IoT training dataset. This is a crucial step as it sets the foundation for the subsequent steps.Step 2 to 6 focuses on handling missing values. For each feature F in the dataset D, the process checks if the value at index i is null (Step 3). If it is null, the missing value is filled in (Step 4). The process then ends the if loop (Step 5) and repeats this process for each feature in the dataset (Step 6).Step 7 to 8 focuses on data transformation. For each feature F in the dataset D, a non-linear data transformation is applied (Step 8). This step is important as it helps in making the data more meaningful and useful for analysis.

Multi-class Fast Parallel Decision tree:

This parallel and fast decision tree is designed for regression

tasks, where the goal is to predict a continuous target variable based on the values of other attributes or features. The code defines several functions, including covariance, variance, and pearsons_correlation_coefficient, which are essential for calculating correlation and variance between variables. The heart of the code is the FPCCTree class, which represents the decision tree model. It allows you to fit the model to a dataset using the fit method, build the tree structure with the build_tree method, and make predictions for new instances using the predict method. The tree construction process relies on finding attribute-value pairs with the highest Pearson's correlation coefficient with the target variable, and it continues recursively until specific stopping criteria are met. This decision tree can be a useful tool for regression tasks where understanding the linear relationship between attributes and the target variable is crucial for making accurate predictions.

```
FUNCTION covariance(x, y):
  RETURN mean((x - mean(x)) * (y - mean(y)))
FUNCTION variance(x):
  RETURN mean((x - mean(x))^2)
FUNCTION pearsons_correlation_coefficient(attr, decision_attr):
  cov = covariance(attr, decision_attr)
  RETURN cov / sqrt(variance(attr) * variance(decision_attr))
CLASS PCCTree:
  INIT(epsilon = 0.05):
    self.epsilon = epsilon
  FUNCTION fit(data, target_column):
    self.target = target_column
    self.tree = build_tree(data)
  FUNCTION build_tree(data):
    IF length(unique(data[self.target])) == 1:
      RETURN { "type": "leaf", "class": data[self.target].first_element() }
    max_pcc = -infinity
    best_attr = None
    best_value = None
    FOR attr IN data.columns:
      IF attr == self.target:
        CONTINUE
      FOR value IN unique(data[attr]):
        subset = filter data where data[attr] == value
        IF length(subset) == 0:
          CONTINUE
        pcc = pearsons_correlation_coefficient(subset[attr], subset[self.target])
        IF pcc > max_pcc:
          max_pcc = pcc
          best_attr = attr
          best_value = value
    IF best_attr is None:
      RETURN { "type": "leaf", "class": majority_class(data[self.target]) }
    left_subset = filter data where data[best_attr] <= best_value
    right_subset = filter data where data[best_attr] > best_value
    IF length(left_subset) == 0 OR length(right_subset) == 0:
      RETURN { "type": "leaf", "class": majority_class(data[self.target]) }
    RETURN {
      "type": "node",
      "attribute": best_attr,
      "value": best_value,
      "left": build_tree(left_subset),
      "right": build_tree(right_subset)
    }
  FUNCTION predict_instance(instance, node):
    IF node["type"] == "leaf":
      RETURN node["class"]
    IF instance[node["attribute"]] <= node["value"]:
      RETURN predict_instance(instance, node["left"])
    ELSE:
      RETURN predict_instance(instance, node["right"])
  FUNCTION predict(data):
    RETURN apply predict_instance on each row of data
```

The main function to train or construct the tree.
It sets the target column (the one we want to predict) and starts the recursive tree-building process using the build_tree function.
build_tree(data):
If all data points in the current dataset belong to the same class, it returns a leaf node with that class label.
It then initializes variables to track the best attribute and value to split on, as well as the maximum Pearson's correlation coefficient.
For each attribute in the data (excluding the target):
For each unique value in that attribute:
It computes the subset of data where the attribute equals that value.
Computes the Pearson's correlation coefficient for that subset against the target column.
If this coefficient is the highest seen so far, it updates the best attribute, best value, and max coefficient variables.
If no suitable attribute is found to split on (i.e., all have the same value), it returns a leaf node with the majority class label.
Otherwise, it splits the data into two subsets based on the best attribute and value found.
If either subset is empty (which shouldn't happen with the given logic but is a safeguard), it returns a leaf node with the majority class label.
It then recursively builds the tree for both left and right subsets.
predict_instance(instance, node):
Makes a prediction for a single instance.
If the current node is a leaf, it returns the class label of that leaf.
Otherwise, it checks the value of the instance's attribute that matches the current node's splitting attribute. Based on this value, it either moves to the left or right child node and continues the prediction process recursively.
predict(data):
Uses the predict_instance function to predict the class for each row in the data and returns a list of predictions.
This algorithm constructs a decision tree based on maximizing the Pearson's correlation coefficient at each split. As a result, the tree divides the dataset in a way that each split is most correlated with the target variable. This is different from more traditional tree algorithms like the ID3 or C4.5, which use entropy or the Gini impurity as the criterion for making splits.

4. Multi-Class Feature rank and Gaussian Kernel based classification

Step 1: Feature Ranking

In the feature ranking step, the significance or importance of each feature in the dataset is determined with respect to the target variable. A specific method, such as mutual information, correlation coefficient, or feature importance from a model like a decision tree, is used to rank each feature. The outcome is a rank

or score that indicates the relevance or importance of each feature with respect to the target variable. These ranks are then stored in a vector for further processing in the subsequent steps.

Step 2: Feature Selection

Once the features have been ranked, the feature selection step aims to select a subset of the most important or relevant features based on predefined criteria. The ranking vector is first sorted in descending order, ensuring that features with the highest ranks or scores are considered first. Two criteria are provided for feature selection:

a. The sum of ranks of the first i features is compared against a fraction (defined by C1) of the total rank. If it's less than or equal to this fraction, the feature is considered important and added to the selected list.

b. Each feature's rank is compared to a threshold, which is the maximum rank multiplied by C2. If a feature's rank is above this threshold, it's considered important and added to the selected list.

Step 3: Classification using the Anisotropic Gaussian Kernel

With a selected subset of features, the classification step uses a non-parametric method based on an anisotropic Gaussian kernel. For each instance in the dataset, the method calculates a score indicating the likelihood of the instance belonging to each class. This is done using the Gaussian kernel, which computes the similarity between the instance and all training samples. The kernel is weighted by the rank of each feature, making it anisotropic. The outcome is a score that provides the likelihood of the instance belonging to a particular class. The class that produces the maximum score for a given instance is assigned as its predicted label.

Algorithm Steps:

Input:

Training data with features and target labels

C1, C2: Threshold values for feature subset selection (can be determined by cross-validation)

Output:

Model for classification using selected features

Steps:

1. Feature Ranking:
   - Calculate the rank (σ_j) of each feature in the training data using a feature ranking method (e.g., mutual information).
   - Store the ranks in vector R.
2. Feature Selection:
   a. Sort vector R in descending order to get a sorted ranking vector S.
   b. Select features based on the following criteria:
      i. Calculate the total rank of all features.
      ii. Initialize an empty list called "selected_features".
      iii. For each feature f_i in the dataset:
         1. If the sum of ranks of features from 1 to i (inclusive) divided by the total rank is less than or equal to C1:
            - Add f_i to "selected_features".
         2. If the rank of f_i is greater than or equal to (maximum rank * C2):
            - Add f_i to "selected_features".
3. Classification using the Anisotropic Gaussian Kernel:
   a. For each instance x in the dataset:
      i. For each class i:
         1. Calculate g_i(x) using the formula:
            g_i(x) = (1 / (n * product of σ_j from 1 to k)) * sum from t=1 to n of [exp(sum from j=1 to k of [(x_j - x_j^t)^2 / (2 * σ_j^2)]) * r_i^t]
      ii. Assign the class with the highest g_i(x) value to instance x.

End of Algorithm

5. Experimental results

The dataset is composed of multiple parts, with a baseline dataset capturing normal activities gathered during a 10-minute simulation. Additionally, six different attack scenarios were executed independently against the baseline architecture, each involving RT0 as the rogue terminal. These attacks ranged from basic denial-of-service (DOS) attacks to fake data injection and logic attacks, each with varying numbers of messages and durations.

The dataset is provided in the form of separate CSV files, and it includes various fields such as message ID, timestamps, error indicators, mode codes, channel information, and much more. These fields provide detailed information about the messages exchanged within the MIL-STD-1553 databus during both normal operations and the simulated attacks. The dataset is a valuable resource for studying the behavior of this databus under different conditions and assessing the effectiveness of intrusion detection and security measures.

Table 1: Sample thread cloud dataset

Features	Skewness	Kurtosis	Mode	Range	Variance
msgId	0.189494	-1.30816	1	22999	53529192
timestamp	3.310311	10.78406	67.47067	5266.15	1056135
Error					
modeCode					
Channel	2.78255	5.743005	0	1	0.085169
connType	0.554083	-1.17511	0	2	0.617999
Sa	9.670794	161.3402	1	31	2.758459
Ssa	1.624146	1.227912	5	30	75.58434
Da	9.936773	110.0526	3	30	6.181331

Features	Skewness	Kurtosis	Mode	Range	Variance
Dsa	2.534756	4.540211	0	20	36.35458
Wc	2.379626	4.722685	2	31	67.95668
modeCodeVal	-24.3663	591.7619	17	13	0.282783
txRsp	-8.69845	73.66835	8.5	8.5	0.907073
txSts	-0.58344	-1.00942	6	6	3.289346
rxRsp	0	0	8.5	0	0
rxSts	-0.00579	-1.59076	5	5	2.786449
dw0	1.552994	1.336516	0	47	157.6272
dw1	-0.759	-1.42304	11	11	26.41443
dw2	-1.09264	-0.62502	19	19	52.16003
dw3	-1.19516	-0.3292	26	26	94.19917
dw4	-2.01009	2.580657	20	20	38.30193
dw5	-1.15126	-0.38477	12	12	18.67084
dw6	-1.3204	0.019284	20	20	53.94966
dw7	-1.57751	1.229838	13	13	17.25815
dw8	-0.96041	-1.07694	11	11	24.56234
dw9	-2.78109	5.734853	1	1	0.085229
dw10	-2.78109	5.734853	1	1	0.085229
dw11	-2.78109	5.734853	1	1	0.085229
dw12	-2.78109	5.734853	1	1	0.085229
dw13	-2.78109	5.734853	1	1	0.085229
dw14	-2.78109	5.734853	1	1	0.085229
dw15	-2.78109	5.734853	1	1	0.085229
dw16	-2.78109	5.734853	1	1	0.085229
dw17	-2.78109	5.734853	1	1	0.085229
dw18	-2.78109	5.734853	1	1	0.085229
dw19	-2.78109	5.734853	1	1	0.085229
dw20	-3.35466	9.254449	1	1	0.065565
dw21	-3.35466	9.254449	1	1	0.065565
dw22	-3.35466	9.254449	1	1	0.065565
dw23	-3.35466	9.254449	1	1	0.065565
dw24	-3.35466	9.254449	1	1	0.065565
dw25	-3.35466	9.254449	1	1	0.065565
dw26	-3.35466	9.254449	1	1	0.065565
dw27	-3.35466	9.254449	1	1	0.065565
dw28	-3.35466	9.254449	1	1	0.065565
dw29	-3.35466	9.254449	1	1	0.065565
dw30	-3.35466	9.254449	1	1	0.065565
dw31	-3.35466	9.254449	1	1	0.065565
Gap	2.542136	4.462845	14	249968	5.89E+09
msgTime	2.049213	3.352466	65	667	29916.96
Class	-6.83715	44.74991	1	1	0.019708
attack_type	-2.69207	6.426248	6	6	1.389317

```
Correctly Classified Instances         674               96.4235 %
Incorrectly Classified Instances        25                3.5765 %
Kappa statistic                          0.9211
Mean absolute error                      0.0591
Root mean squared error                  0.1663
Relative absolute error                 13.0727 %
Root relative squared error             34.9901 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.969     0.046     0.976     0.969     0.973     0.921     0.992     0.996     1
                 0.954     0.031     0.943     0.954     0.948     0.921     0.992     0.982     2
Weighted Avg.    0.964     0.040     0.964     0.964     0.964     0.921     0.992     0.991
```

```
Correctly Classified Instances         687               98.2833 %
Incorrectly Classified Instances        12                1.7167 %
Kappa statistic                          0.9642
Mean absolute error                      0.0301
Root mean squared error                  0.1115
Relative absolute error                  6.276  %
Root relative squared error             22.7536 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.971     0.010     0.986     0.971     0.978     0.964     0.999     0.998     cluster1
                 0.990     0.029     0.981     0.990     0.986     0.964     0.999     0.999     cluster2
Weighted Avg.    0.983     0.021     0.983     0.983     0.983     0.964     0.999     0.999
```

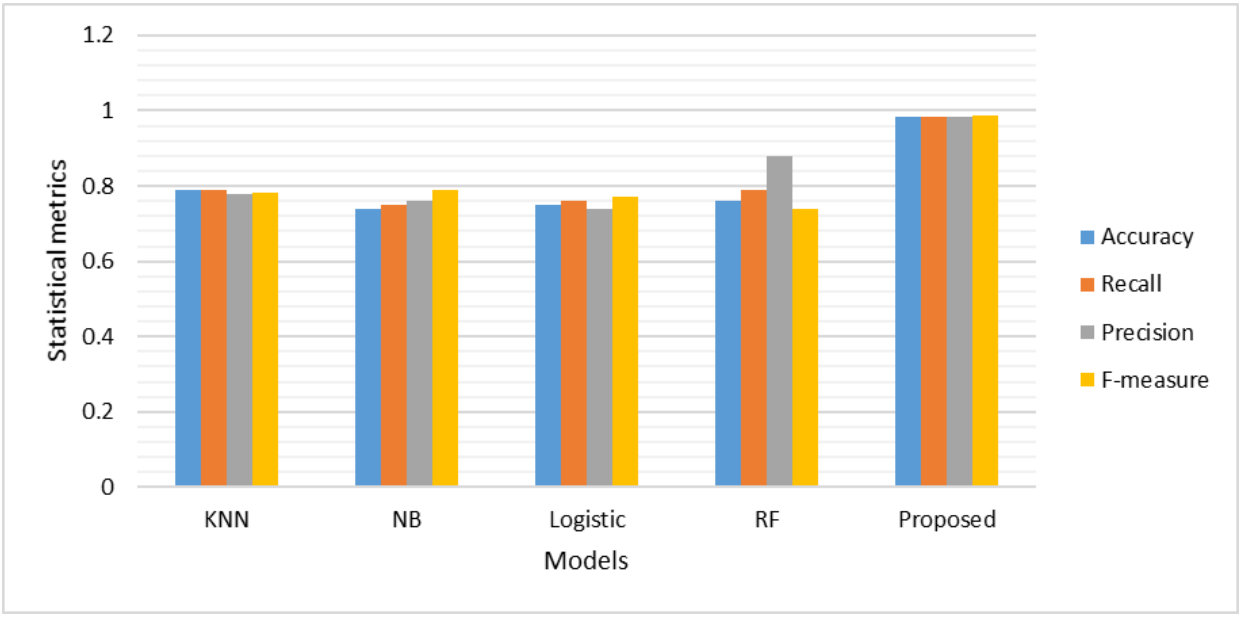

Fig.2: Statistical performance metrics and its analysis on maternity with 100000 test records.

The graph illustrates the evaluation outcomes of various machine learning models applied to a maternity dataset. The evaluation metrics employed include Accuracy, Recall, Precision, and F-measure. Among the models tested, the proposed model demonstrated the highest overall performance, achieving an Accuracy of 0.985, as well as impressive scores for Recall, Precision, and F-measure. These results indicate that the proposed model exhibits strong predictive capabilities, effectively identifying patterns and making accurate predictions on the dataset.

In contrast, the KNN, NB, and Logistic models exhibited lower performance, with accuracy ranging from 0.74 to 0.79. The RF model attained a relatively higher Accuracy of 0.76, but its Precision score was low at 0.74.

```
Correctly Classified Instances        5338               97.5333 %
Incorrectly Classified Instances       135                2.4667 %
Kappa statistic                          0.8676
Mean absolute error                      0.0159
Root mean squared error                  0.0893
Relative absolute error                 20.893  %
Root relative squared error             45.8453 %
Total Number of Instances             5473

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.990    0.129    0.985      0.990    0.988      0.877    0.991     0.999     1
                 0.900    0.005    0.914      0.900    0.907      0.901    0.989     0.935     2
                 0.857    0.001    0.857      0.857    0.857      0.856    0.999     0.884     3
                 0.875    0.002    0.856      0.875    0.865      0.863    0.987     0.855     4
                 0.678    0.003    0.813      0.678    0.739      0.737    0.985     0.812     5
Weighted Avg.    0.975    0.116    0.975      0.975    0.975      0.875    0.991     0.988


Correctly Classified Instances         687               98.2833 %
Incorrectly Classified Instances        12                1.7167 %
Kappa statistic                          0.9642
Mean absolute error                      0.0301
Root mean squared error                  0.1115
Relative absolute error                  6.276  %
Root relative squared error             22.7536 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.971    0.010    0.986      0.971    0.978      0.964    0.999     0.998     cluster1
                 0.990    0.029    0.981      0.990    0.986      0.964    0.999     0.999     cluster2
Weighted Avg.    0.983    0.021    0.983      0.983    0.983      0.964    0.999     0.999
```

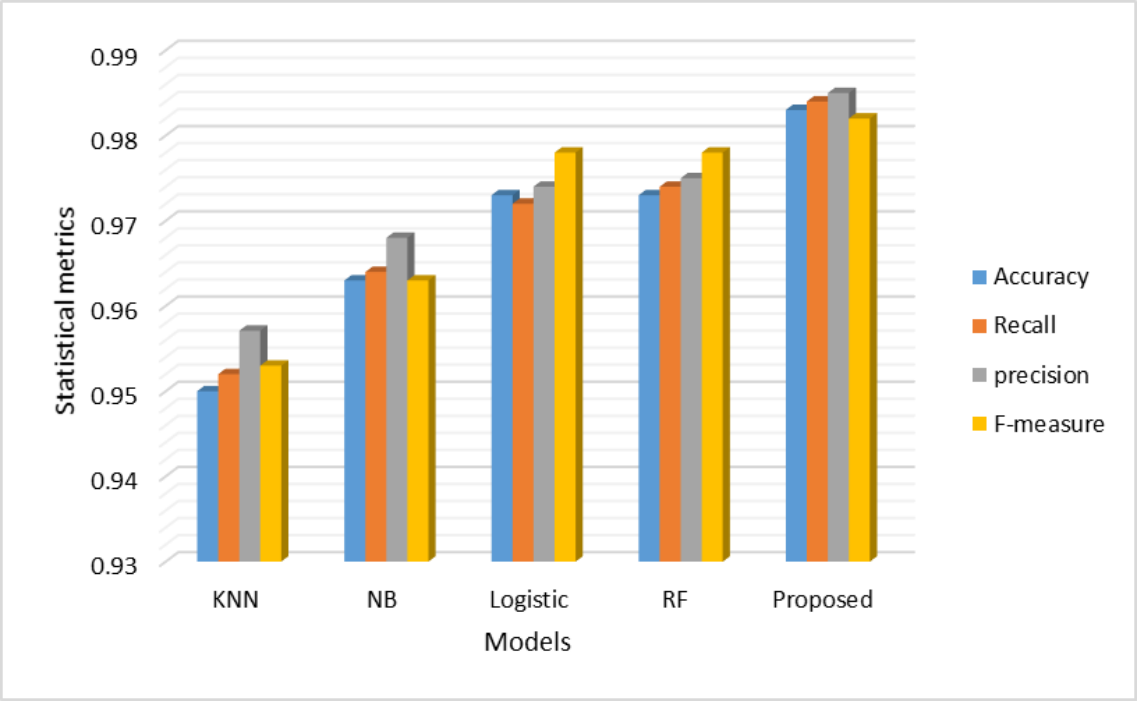

Fig. 3: Statistical performance metrics and its analysis on maternity dataset with 200000 records.

Each model's accuracy is presented as a percentage, representing the proportion of instances correctly classified in the dataset. The K-Nearest Neighbor (KNN) algorithm attained an accuracy of 0.95, while the proposed model outperformed all others with the highest accuracy of 0.983.Recall, also referred to as sensitivity, assesses the proportion of true positives accurately identified by the model. The proposed model demonstrated the highest recall of 0.984, indicating its ability to correctly identify a significant number of true positives in the dataset.

6. Conclusion

IoT devices and networks play a crucial role in the Internet but have security weaknesses and vulnerabilities. Most widely-used IoT devices lack security design, making them vulnerable to

recent attacks that exploit these weaknesses and recruit the devices to cause severe harm. In this work, a cluster based classification approach was proposed for detecting IoT bot cyber attacks. The proposed method achieved good results in terms of accuracy, precision, recall and F1-score, compared to traditional methods. The results demonstrate that the cluster based classification approach is a promising solution for detecting IoT bot cyber attacks in real-time Parallel multi-class classification refers to the process of performing classification tasks simultaneously on multiple classes or categories of data using parallel computing techniques. In traditional multi-class classification, a model is trained to classify data into one of several mutually exclusive classes. However, in some scenarios, it may be advantageous to perform these classifications in parallel, especially when dealing with a large number of classes or when speed and efficiency are crucial. Parallel multi-class classification can be implemented using parallel processing or distributed computing frameworks to train and evaluate multiple classifiers concurrently. Each classifier is responsible for classifying data into one of the classes, and the results are combined to make the final prediction as shown in figure 1.

References

[1] M. Elsisi, M. Amer, A. Dababat, and C.-L. Su, "A comprehensive review of machine learning and IoT solutions for demand side energy management, conservation, and resilient operation," Energy, vol. 281, p. 128256, Oct. 2023, doi: 10.1016/j.energy.2023.128256.

[2] C. Alex, G. Creado, W. Almobaideen, O. A. Alghanam, and M. Saadeh, "A Comprehensive Survey for IoT Security Datasets Taxonomy, Classification and Machine Learning Mechanisms," Computers & Security, vol. 132, p. 103283, Sep. 2023, doi: 10.1016/j.cose.2023.103283.

[3] I. A. Adeleke, N. I. Nwulu, and O. A. Ogbolumani, "A hybrid machine learning and embedded IoT-based water quality monitoring system," Internet of Things, vol. 22, p. 100774, Jul. 2023, doi: 10.1016/j.iot.2023.100774.

[4] M. E. Elaraby, A. A. Ewees, and A. M. Anter, "A robust IoT-based cloud model for COVID-19 prediction using advanced machine learning technique," Biomedical Signal Processing and Control, vol. 87, p. 105542, Jan. 2024, doi: 10.1016/j.bspc.2023.105542.

[5] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," Knowledge-Based Systems, vol. 279, p. 110941, Nov. 2023, doi: 10.1016/j.knosys.2023.110941.

[6] F. Aloraini, A. Javed, O. Rana, and P. Burnap, "Adversarial machine learning in IoT from an insider point of view," Journal of Information Security and Applications, vol. 70, p. 103341, Nov. 2022, doi: 10.1016/j.jisa.2022.103341.

[7] Md. A. Ahmed, Md. S. Hossain, W. Rahman, A. H. Uddin, and Md. T. Islam, "An advanced Bangladeshi local fish classification system based on the combination of deep learning and the internet of things (IoT)," Journal of Agriculture and Food Research, vol. 14, p. 100663, Dec. 2023, doi: 10.1016/j.jafr.2023.100663.

[8] D. Tiwari, B. S. Bhati, B. Nagpal, S. Sankhwar, and F. Al-Turjman, "An enhanced intelligent model: To protect marine IoT sensor environment using ensemble machine learning approach," Ocean Engineering, vol. 242, p. 110180, Dec. 2021, doi: 10.1016/j.oceaneng.2021.110180.

[9] S. Hamdan, S. Almajali, M. Ayyash, H. Bany Salameh, and Y. Jararweh, "An intelligent edge-enabled distributed multi-task learning architecture for large-scale IoT-based cyber–physical systems," Simulation Modelling Practice and Theory, vol. 122, p. 102685, Jan. 2023, doi: 10.1016/j.simpat.2022.102685.

[10] Y. Cao, Z. Wang, H. Ding, J. Zhang, and B. Li, "An intrusion detection system based on stacked ensemble learning for IoT network," Computers and Electrical Engineering, vol. 110, p. 108836, Sep. 2023, doi: 10.1016/j.compeleceng.2023.108836.

[11] H. Huang, L. Zhao, and Y. Wu, "An IoT and machine learning enhanced framework for real-time digital human modeling and motion simulation," Computer Communications, Sep. 2023, doi: 10.1016/j.comcom.2023.09.024.

[12] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Anomaly based network intrusion detection for IoT attacks using deep learning technique," Computers and Electrical Engineering, vol. 107, p. 108626, Apr. 2023, doi: 10.1016/j.compeleceng.2023.108626.

[13] S. Kaddoura, A. El Arid, and A. Al-Dulaimy, "Chapter 2 - Supervised machine learning techniques to protect IoT healthcare environment against cyberattacks," in Intelligent Edge Computing for Cyber Physical Applications, D. J. Hemanth, B. B. Gupta, M. Elhoseny, and S. V. Shinde, Eds., in Intelligent Data-Centric Systems. , Academic Press, 2023, pp. 17–34. doi: 10.1016/B978-0-323-99412-5.00001-0.

[14] S. Kakandwar, B. Bhushan, and A. Kumar, "Chapter 3 - Integrated machine learning techniques for preserving privacy in Internet of Things (IoT) systems," in Blockchain Technology Solutions for the Security of IoT-Based Healthcare Systems, B. Bhushan, S. K. Sharma, M. Saračević, and A. Boulmakoul, Eds., in Cognitive Data Science in Sustainable Computing. , Academic Press, 2023, pp. 45–75. doi: 10.1016/B978-0-323-99199-5.00012-4.

[15] R. K. Muna, M. I. Hossain, Md. G. R. Alam, M. M. Hassan, M. Ianni, and G. Fortino, "Demystifying machine learning models of massive IoT attack detection with Explainable AI for sustainable and secure future smart cities," Internet of Things, vol. 24, p. 100919, Dec. 2023, doi: 10.1016/j.iot.2023.100919.

[16] A. Alatram, L. F. Sikos, M. Johnstone, P. Szewczyk, and J. J. Kang, "DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol," Computer Networks, vol. 231, p. 109809, Jul. 2023, doi: 10.1016/j.comnet.2023.109809.

[17] N. Tekin, A. Acar, A. Aris, A. S. Uluagac, and V. C. Gungor, "Energy consumption of on-device machine learning models for IoT intrusion detection," Internet of Things, vol. 21, p. 100670, Apr. 2023, doi: 10.1016/j.iot.2022.100670.

[18] N. Prazeres, R. L. de C. Costa, L. Santos, and C. Rabadão, "Engineering the application of machine learning in an IDS based on IoT traffic flow," Intelligent Systems with Applications, vol. 17, p. 200189, Feb. 2023, doi: 10.1016/j.iswa.2023.200189.

[19] P. Sanju, "Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks," Journal of Engineering Research, p. 100122, Jun. 2023, doi: 10.1016/j.jer.2023.100122.

[20] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H.

Ali, and J. Ahmad, “Enhancing IoT network security through deep learning-powered Intrusion Detection System,” Internet of Things, vol. 24, p. 100936, Dec. 2023, doi: 10.1016/j.iot.2023.100936.

[21] B. Sharma, L. Sharma, C. Lal, and S. Roy, “Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach,” Expert Systems with Applications, vol. 238, p. 121751, Mar. 2024, doi: 10.1016/j.eswa.2023.121751.

[22] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, “Feature extraction for machine learning-based intrusion detection in IoT networks,” Digital Communications and Networks, Sep. 2022, doi: 10.1016/j.dcan.2022.08.012.

[23] S. Ben Atitallah, M. Driss, and H. Ben Ghezala, “FedMicro-IDA: A federated learning and microservices-based framework for IoT data analytics,” Internet of Things, vol. 23, p. 100845, Oct. 2023, doi: 10.1016/j.iot.2023.100845.

[24] A. Hennebelle, H. Materwala, and L. Ismail, “HealthEdge: A Machine Learning-Based Smart Healthcare Framework for Prediction of Type 2 Diabetes in an Integrated IoT, Edge, and Cloud Computing System,” Procedia Computer Science, vol. 220, pp. 331–338, Jan. 2023, doi: 10.1016/j.procs.2023.03.043.

[25] S. Saif, P. Das, S. Biswas, M. Khari, and V. Shanmuganathan, “HIIDS: Hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in IoT based healthcare,” Microprocessors and Microsystems, p. 104622, Aug. 2022, doi: 10.1016/j.micpro.2022.104622.

[26] A. Sharifi and S. Goli-Bidgoli, “IFogLearn++: A new platform for fog layer’s IoT attack detection in critical infrastructure using machine learning and big data processing,” Computers and Electrical Engineering, vol. 103, p. 108374, Oct. 2022, doi: 10.1016/j.compeleceng.2022.108374.

[27] O. Habibi, M. Chemmakha, and M. Lazaar, “Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection,” Engineering Applications of Artificial Intelligence, vol. 118, p. 105669, Feb. 2023, doi: 10.1016/j.engappai.2022.105669.

[28] T. Gaber, A. El-Ghamry, and A. E. Hassanien, “Injection attack detection using machine learning for smart IoT applications,” Physical Communication, vol. 52, p. 101685, Jun. 2022, doi: 10.1016/j.phycom.2022.101685.