Enhancing IoT Security: A Comparative Analysis of Preprocessing Techniques and Classifier Performance on IoT23 and CIC IoT 2023 Datasets

Zyad ELKHADIR and Mohammed ACHKARI BEGDOURI

***Abstract*—This research delves into the escalating cybersecurity issues brought about by the rapid integration of Internet of Things (IoT) technologies. With the proliferation of IoT systems, the intricate connections between devices are heightened, leading to a surge in data flow and consequently, presenting numerous openings for cyber threats. As a result, the detection and mitigation of cyberattacks aimed at IoT systems have become indispensable priorities within the cybersecurity domain. The objective of this technical assessment is to offer a detailed insight into how employing PCA/KPCA feature extraction algorithms with many machine learning classifiers contribute to classifying cyber-attacks within IoT systems. By analyzing the performance of many classifiers such as Support Vector Machine (SVM), multilayer perceptron (MLP), Decision Tree (DT), Logistic Regression (LR), k-Nearest Neighbors (k-NN), Random Forest (RF), cybersecurity professionals can glean valuable insights to craft resilient protection strategies for the IoT landscape. The outcome indicates that the combination of Principal Component Analysis (PCA) with Decision Tree/Random Forest outperformed the other models in the evaluation. The experiments were conducted on iot23 dataset and CIC IoT 2023 dataset**

Index Terms*— *Internet of Things (IoT), PCA, KPCA, IDS, KNN, Decision Tree (DT), Random Forest (RF)

I. Introduction

The concept of the Internet of Things (IoT) encompasses a framework designed to link diverse computing devices and sensors via the Internet, enhancing a wide array of applications such as smart homes, healthcare, agriculture, and industrial settings. IoT has spurred significant advancements across industries and in our daily lives by interconnecting billions of devices and generating vast volumes of data. Yet, this interconnectedness also brings forth significant security challenges, leaving systems vulnerable to various forms of attacks. Recognizing anomalous behavior within IoT systems is paramount for safeguarding their integrity, as it enables the early detection of potential malicious activities or system malfunctions.

The rapid expansion and increasing complexity of the IoT landscape necessitate the development of robust and flexible intrusion detection systems (IDS). Conventional IDS methods, such as statistical and rule-based approaches, frequently face challenges in adapting to the dynamic nature of IoT environments and the ever-evolving range of cyber threats. Leveraging Machine Learning (ML) has proven to be a highly promising strategy for detecting anomalies in IoT systems.

Numerous recent studies incorporate Support Vector Machines (SVM) [1] as a core component of their approaches. For instance, the article [2] proposes a streamlined, precise, and high-performing IDS for IoT networks by utilizing finely-tuned Linear Support Vector Machines (LSVMs) along with advanced feature selection techniques. This method applies four feature selection strategies: Importance Coefficient, Forward-Sequential, Backward-Sequential, and Correlation Coefficient-based methods, to identify the most critical features from extensive datasets, significantly improving IDS performance. Similarly, the work [3] introduces a hybrid intrusion detection system that combines SVM with Grey Wolf Optimization (GWO), harnessing the strengths of both algorithms. In this framework, SVM is used to train the system and differentiate anomalous records from normal ones, while GWO optimizes the kernel function, selects features, and fine-tunes SVM parameters, thereby enhancing the overall classification effectiveness.

Other works explore the application of Decision Tree-based approaches, for example, the study [4] integrates multiple classifiers rooted in decision tree and rule-based methodologies, including REP Tree, the JRip algorithm, and Forest PA. The first two classifiers utilize dataset features to categorize network traffic as either Attack or Benign, while the third classifier combines the original dataset features with the outputs of the first two classifiers. Experimental evaluations using the CICIDS2017 and BoT-IoT datasets highlight the proposed IDS's superior

Manuscript received August 13, 2024; revised January 28, 2025.

Zyad Elkhadir is an assistant professor attached to SIGL Laboratory, ENSATe of Tetouan, Abdelmalek Essaâdi University, Morocco (e-mail: z.elkhadir@uae.ac.ma)

Mohammed Achkari Begdouri is a professor attached to SIGL Laboratory, ENSATe of Tetouan, Abdelmalek Essaâdi University, Morocco (e-mail: m.achkaribegdouri@uae.ac.ma)

performance compared to existing methods, particularly in terms of accuracy, detection rate, false alarm rate, and time efficiency.

In [5], a combination of three decision trees is utilized for intrusion detection, and the performance of the proposed approach is compared with several other classifiers. Experiments conducted on the NSL-KDD dataset reveal that the proposed method outperforms alternative techniques in intrusion detection effectiveness

In [6], the authors propose an enhanced Intrusion Detection System (IDS) leveraging Gradient Boosting (GB) and Decision Tree (DT) techniques, implemented through the open-source CatBoost framework for IoT security. The improved NSL-KDD, IoT-23, BoT-IoT, and Edge-IIoT datasets were utilized, with GPU support to optimize the experimental setup. Compared to existing IDS methods, the proposed approach demonstrates superior performance metrics, particularly in terms of accuracy.

Logistic Regression [7] has also been utilized as a classifier. In [8], the authors introduced mIDS, a system designed to monitor and detect attacks using a statistical analysis tool based on Binary Logistic Regression (BLR). mIDS operates by analyzing local node parameters for both benign and malicious behavior, constructing a normal behavior model to identify abnormalities within constrained nodes.

K-Nearest Neighbors (KNN) [9] has been widely used in various studies. In [10], the authors proposed a network intrusion detection model for IoT environments that combines the KNN classifier with feature selection techniques. They developed the Network Intrusion Detection System (NIDS) using KNN to improve accuracy (ACC) and detection rate (DR). To enhance data quality and identify the most relevant features, they applied Principal Component Analysis (PCA), univariate statistical tests, and a Genetic Algorithm (GA) for feature selection. The paper [11] introduced Deep Neural Networks (DNN) with kNN algorithm. This method utilizes information gain to select the most important features. The effectiveness of this approach was evaluated using the public NSL-KDD and CICIDS2017 datasets. Recently, the authors of [12] systematically evaluate many PCA variants in combination with KNN classifier to improve the intrusion detection. They conduct extensive experiments on IoT23 and CICIoT 2023. The same authors evaluate KNN algorithm with different distance metrics in the research [13].

The paper [14] demonstrates that Multi-layer Perceptron (MLP) is a viable solution, achieving top performance and outperforming many previous neural network implementations. The study [15] highlights MLP's effectiveness in classifying IoT botnet traffic.

In [16], a multi-level random forest algorithm integrated with a fuzzy inference system was developed for intrusion detection. This approach combines the strengths of filter and wrapper techniques to create an advanced multi-level feature selection method, enhancing network security. In the first stage of feature selection, a filter method using correlation-based feature selection identifies key features based on multicollinearity within the dataset. A genetic search is then applied within this method to identify the optimal features, evaluating each attribute's effectiveness and selecting those with the highest fitness values. Additionally, a rule assessment is used to determine if two feature subsets have identical fitness values, ultimately selecting the subset with the fewest features. The second stage employs a wrapper method with sequential forward selection to refine the top features, optimizing them based on the accuracy of the baseline classifier. The selected top features are then fed into the random forest algorithm for intrusion detection. Finally, fuzzy logic is applied to categorize the intrusions into levels such as normal, low, medium, or high, helping to reduce misclassification.

In other studies, researchers selected the best classifier by comparing it with alternative models. In [17], the authors trained their model using the IoTID20 dataset with features selected by a Genetic Algorithm (GA) and concluded that the Decision Tree and Random Forest classifiers demonstrated the most optimal performance.

The study [18] applied various machine learning techniques to detect anomalies in cyber-attacks targeting IoT systems and evaluated the performance of these methods. The experiments were conducted using the ToN-IoT and Bot-IoT datasets, and the results showed that the neural network outperformed the other models.

In [19], the authors explore various machine learning classifiers. The system achieved a test accuracy of 99.4% for Decision Tree, Random Forest, and Artificial Neural Networks (ANN). While these methods showed similar accuracy levels, additional metrics indicated that Random Forest performed better overall. A key takeaway from the study is that simpler models, such as Decision Tree and Random Forest, can be effectively compared with more complex networks like ANN for anomaly detection. The open-source dataset used in the study was sourced from Kaggle [20].

The methodology presented in our article is similar to the approaches discussed in the previously cited works, it involves the comparison of multiple classifiers to identify the most effective one. What distinguishes our work is the use of newer datasets, such as the IoT23 and CIC IoT 2023 datasets, the application of PCA and KPCA for feature extraction, and the employment of several scalers, including StandardScaler, MinMaxScaler, and RobustScaler, prior to the feature extraction step. The experimental results show that combining PCA with the Decision Tree classifier produces the best accuracy while requiring less CPU time.

This paper is structured as follows: Section II describes the proposed IDS. Section III introduces the datasets and their preprocessing. Section IV explains the characteristics of each scaler. In Section V, we revisit Principal Component Analysis (PCA) and Kernel PCA. Section VI addresses the machine learning classifiers and their mathematical formulations. Section VII covers the experimental setup, analysis results, and comparisons with other classifiers. Finally, Section VIII presents the

conclusion and outlines future directions for the research.

II. THE PROPOSED APPROACH

The Intrusion Detection System (IDS) proposed in this study, as illustrated in Fig. 1, operates in two key parts: the training part and the testing part. During the first part, data is gathered from multiple devices and undergoes preprocessing steps, which include data cleaning and handling incomplete data, as outlined in Section III. Following this, the data is scaled using numerous methods, as described in Section IV, ensuring uniform scaling across all features to prevent any single feature from dominating due to its scale.

Once the data is scaled, dimensionality reduction techniques, such as Principal Component Analysis (PCA) or Kernel PCA (KPCA), are applied to extract principal components, resulting in a dataset with reduced features. Further information on PCA and KPCA is provided in Section IV. In the testing part, data is gathered from the identical devices and subjected to the same preprocessing and normalization procedures as in the training part. This data is then mapped onto the reduced feature space using the principal components. In the end, various classifiers—including K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP)—are employed to classify the testing data as either normal or malicious. Section VI provides an overview and mathematical formulation of each used machine learning technique.

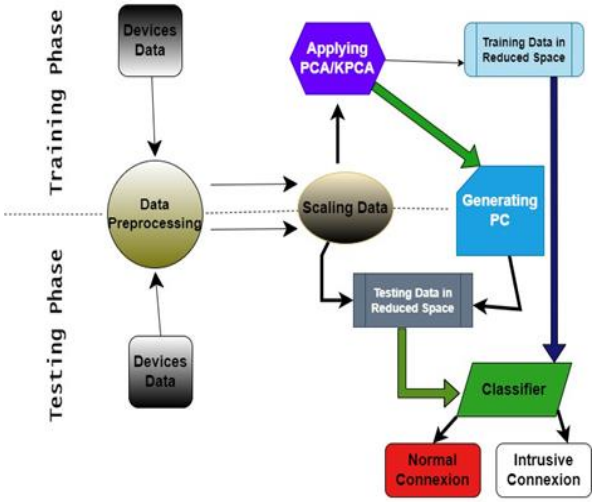

Fig. 1. The proposed IDS model

III. DATASETS

A. Iot-23

The first dataset used in this study is the IoT23 dataset, described in [21] and published in January 2020. It contains network traffic data from three different smart home IoT devices: Amazon Echo, Philips HUE, and Somfy Door Lock. Designed specifically for the development of machine learning algorithms, the dataset includes both real and labeled instances of IoT malware infections, as well as benign traffic. It consists of 23 captures or scenarios, with 20 representing malicious activity and 3 representing benign traffic.

Each capture from infected devices is labeled with the corresponding malware sample executed. The malware labels within the IoT-23 dataset encompass categories such as Attack, C&C, C&C-FileDownload, C&C-HeartBeat, C&C- HeartBeatAttack, C&C-HeartBeat-FileDownload, C&C-Mirai, C&C-Torii, DDoS, FileDownload, Okiru, Okiru-Attack, and PartOfAHorizontalPortScan.

Given the large size of the dataset, we chose to extract a subset of records from each capture and combine them into a new dataset. This approach allows us to manage the computational workload more efficiently while preserving most of the attack types found in the original IoT-23 dataset

B. Data Preprocessing

The Python library Pandas was used to load each of the 23 datasets from the IoT-23 Dataset into individual data frames.

We applied a condition to skip the first 10 rows and read the next one hundred thousand rows from each dataset. These 23 data frames were then consolidated into a single data frame, with missing values replaced by 0.

C. CIC IoT 2023

The second dataset is a recent release from the Canadian Institute for Cybersecurity [22], designed to support the development of security analytics applications for real-world IoT operations. This dataset provides a unique and comprehensive collection of IoT attack data, covering 33 attacks within an IoT network of 105 devices. The attacks are classified into seven categories: DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai.

Detailed information on each attack, along with the corresponding number of packets used for each during the analysis, is provided in Table 1

TABLE I. DESCRIPTION OF CICIOT2023 ATTACKS

Attack Category	Attack Name	Packet Number
DDoS	ACK Fragmentation	285104
	UDP Flood	5412287
	SlowLoris ICMP	23426
	Flood RSTFIN	7200504
	Flood PSHACK	4045285
	Flood HTTP	4094755
	UDP Fragmentation	28790
	ICMP Fragmentation	286925
	TCP Flood	452489
	SYN Flood	4497667
	Synonymous IP Flood	4059190
DoS	TCP Flood	3598138
	HTTP Flood	2,671,445
	SYN Flood	71,864
	UDP Flood	2,028,834
Recon	Ping Sweep	3,318,595
	OS Scan	2262
	Vulnerability Scan	98,259
	Port Scan	37,382
	Host Discovery	82,284
		134,378

Web-Based	Sql Injection Command Injection Backdoor Malware Uploading Attack XSS Browser Hijacking	5245 5409 3218 1252 3846 5859
Brute Force	Dictionary Brute Force	13064
Spoofing	Arp Spoofing DNS Spoofing	307593 178911
Mirai	GREIP Flood Greeth Flood UDPPlain	751682 991866 890576

IV. SCALING TECHNIQUES

Within the realm of data preprocessing for machine learn- ing, scaling methods play a pivotal role in bolstering the resilience and performance of predictive models. Three fre- quently employed scalers include the StandardScaler, Min- MaxScaler, and RobustScaler.

The StandardScaler normalizes features by removing the mean(μ) and adjusting them to unit variance by normalizing with the standard deviation (σ):

$$\text{Standardized Feature} = \frac{\text{Feature} - \mu}{\sigma}$$

This technique proves especially potent when features display varying scales, guaranteeing that each feature carries an equal weight in the model. Consequently, the obtained data boasts a mean of zero and a standard deviation of one.

The MinMaxScaler transforms original features to a defined range, usually between zero and one, using the formula:

$$\text{Scaled Feature} = \frac{\text{Feature} - \min(\text{Feature})}{\max(\text{Feature}) - \min(\text{Feature})}$$

This method is beneficial in scenarios where the data distribution deviates from a Gaussian distribution. It aids in reducing the impact of outliers while maintaining the general structure of the original distribution.

The RobustScaler effectively manages outliers by scaling features using robust statistical measures. It employs the median $\tilde{\mu}$ and interquartile range (IR) for this purpose: specified range, typically between 0 and 1, using the following formula:

$$\text{Robustly Scaled Feature} = \frac{\text{Feature} - \tilde{\mu}}{\text{IR}}$$

This normalization approach is more resistant to extreme values, making it especially appropriate for datasets containing outliers or uneven distributions. The selection of a scaler ultimately relies on the specific traits of the dataset and the objectives of the machine learning task. Each scaler presents unique benefits for handling data patterns, dealing with anomalies, and preserving the essence of the original information

V. THE FEATURE EXTRACTION METHODS

Integrating Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA) into an IoT-based Intrusion Detection System (IDS) can enhance its effectiveness in detecting and mitigating security breaches. These techniques reduce the dimensionality of network traffic data while retaining key features, making it easier to identify abnormal patterns and deviations from normal behavior. Previous studies [23], [24], [25] have demonstrated the effectiveness of PCA and KPCA in improving IDS performance.

KPCA, in particular, excels at capturing nonlinear relationships in data, which is valuable for detecting complex and evolving attack patterns in IoT environments. By leveraging kernel methods, KPCA uncovers subtle anomalies that traditional linear methods may miss, helping the IDS adapt to new and sophisticated threats.

The following sections provide the mathematical foundations of these feature extraction methods.

A. Principal Components Analysis

Principal Component Analysis (PCA) serves the purpose of dimensionality reduction while preserving the maximum possible of variance. This is obtained by considering only the first few Principal Components (PCs), arranged in descending order [26].

Mathematically, let's consider a training set comprising M vectors $w_1, w_2, \ldots, w_M$, each containing n features. To obtain n' principal components of the training set, where $n' < < n$, we follow these steps:

1- Compute the average σ of this set:

$$\sigma = \frac{1}{M}\sum_{i=1}^{M}(w_i) \quad (1)$$

2- Subtract the mean σ from w_i to obtain ρ_i:

$$\rho_i = w_i - \sigma \quad (2)$$

3- Compute the covariance matrix C, where:

$$C_{n\times n} = \frac{1}{M}\sum_{i=1}^{M}(\rho_i \rho_i^T) = AA^T \quad (3)$$

And

$$A_{n\times M} = \frac{1}{\sqrt{M}}\rho_i \quad (4)$$

Let U_k be the k^{th} eigenvector of C corresponding to the λ_k associated eigenvalue. Form a matrix $U_{n\times n'} = [U_1 \ldots U_{n'}]$ consisting of these eigenvectors, such that:

$$CU_k = \lambda_k U_k \quad (5)$$

The first U_k corresponding to the largest eigenvalues λ_k are termed Principal Components (PCs).

B. Kernel Principal Components Analysis

Kernel Principal Component Analysis (Kernel PCA) [27] enhances the capabilities of PCA by using kernel methods to perform nonlinear dimensionality reduction. The method involves projecting the input data into a higher-dimensional feature space to achieve linear separability, and then conducting PCA within this transformed space. It enables Kernel PCA to identify the nonlinear relationships inherent in the original data.

The kernel trick is employed to efficiently calculate the dot product within the higher-dimensional feature space indirectly, without explicitly performing the transformation. Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent input data points, the kernel PCA algorithm can be summarized as follows:

1- Compute the kernel matrix K:

$$K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \quad (6)$$

2- Center the kernel matrix:

$$\mathrm{K}' = \mathrm{K} - 1\mathrm{K} - \mathrm{K}1 + 1\mathrm{K}1 \quad (7)$$ Where 1 represents a matrix filled of ones.

3- Compute the eigenvectors α_i and eigenvalues λ_i of the centered kernel matrix K':

$$\mathrm{K}'\alpha_i = \lambda_i \alpha_i \quad (8)$$

4- Choose the top n′ eigenvectors associated with the highest eigenvalues to construct the principal components. The Radial Basis Function (RBF) Kernel is frequently utilized and is defined as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{2\sigma^2}\right) \quad (9)$$ Where σ represents the kernel bandwidth parameter

VI. CLASSIFIERS

A. KNN

K-Nearest Neighbors (KNN) is a simple yet effective machine learning algorithm applicable to both classification and regression tasks. The fundamental concept behind KNN is to determine the class of a data point based on the majority class among its k-nearest neighbors within the feature space. Given a dataset (x_1, y_1), (x_2, y_2),. . ., (x_n, y_n) where x_i denotes the feature vector and y_i represents the associated class label, the KNN algorithm functions as follows:

1) Select the number of neighbors k.
2) Calculate the distance between the query point and all data points in the training set.
3) Identify the k nearest neighbors based on the calculated distances.
4) For classification, assign the class label by majority vote among the k neighbors. For regression, predict the average of the k neighbors' target values.

B. Decision tree

Decision tree operates by partitioning the feature space into a hierarchy of simple decision rules, represented as a tree- like structure. At each internal node of the tree, a decision is made based on the value of a particular feature, leading to subsequent branches corresponding to different outcomes. The process continues recursively until reaching leaf nodes, where final predictions are made. Decision trees are favored for their ability to handle both numerical and categorical data, their transparency in model interpretation, and their capability to capture nonlinear relationships between features and target variables. Here are the principal steps of Decision tree Algorithm.

1) Initialization:
   - Define the feature space X and the target variable $\mathbf{y}$.
   - Set the root node of the decision tree.
2) Splitting Criteria:
   At each decision node:
   - Iterate over each feature xi in X.
   - Calculate a splitting criterion, such as Gini impurity or entropy, to evaluate the impurity of the data after splitting based on xi.
   - Select the feature x_i and threshold value ϑ that minimizes the impurity or maximizes the information gain.
3) Partitioning:
   - Split the data into two subsets based on the selected feature and threshold:

$$\mathbf{X}_{\text{left}} = \{(\mathbf{x}, \mathbf{y}) \mid x_i \leq \vartheta\}$$
$$\mathbf{X}_{\text{right}} = \{(\mathbf{x}, \mathbf{y}) \mid x_i > \vartheta\}$$

4) Stopping Criteria:
   - Check if stopping criteria are met:
     - Maximum tree depth reached.
     - Minimum number of samples in a node reached
     - All samples in a node belong to the same class
5) Recursive Splitting:
   If stopping criteria are not met:
   - Create child nodes for the current node corresponding to the subsets $\mathbf{X}_{\text{left}}$ and $\mathbf{X}_{\text{right}}$.
   - Recursively apply steps 2-4 to each child node.
6) Leaf Node Prediction:
   - At leaf nodes:
     For classification tasks:

- Assign the majority class label of the samples in the node as the predicted class.

For regression tasks:

- Calculate the mean or median of the target variable of the samples in the node as the predicted value.

7) Tree Building:

Continue splitting and partitioning until all nodes are either leaf nodes or the stopping criteria are met.

C. Random Forest

Random Forest is a powerful ensemble learning algorithm that leverages the wisdom of crowds to make accurate predictions in classification and regression tasks. It operates by constructing multiple decision trees during training and combining their predictions through a process called bootstrapping and aggregation. Let X denote the feature space and y represent the target variable. The Random Forest algorithm consists of the following steps:

1) Bootstrapping:

- Randomly sample with replacement from the original dataset to create multiple bootstrap samples.
- Each bootstrap sample is used to train an individual decision tree.

2) Decision Tree Construction:

- For each bootstrap sample:
- Build a decision tree using a subset of features selected randomly at each node.
- Split the data at each node based on a selected splitting criterion, such as Gini impurity or entropy.
- Continue recursively partitioning the data until reaching leaf nodes.

3) Aggregation:

- Aggregate the predictions of all decision trees to make a final prediction:
  a. For classification tasks, use the majority vote among the predictions of individual trees.
  b. For regression tasks, calculate the mean or median of the predictions.

Random Forest offers several advantages over individual decision trees, including improved generalization, reduced overfitting, and increased robustness to noise in the data. By combining multiple trees trained on different subsets of the data, Random Forest harnesses the collective knowledge of diverse models to produce more accurate and stable predictions.

D. Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. They aim to find the optimal hyperplane that best separates data points of different classes in the feature space.

Given a training dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where x_i represents the feature vector and y_i is the corresponding class label, the SVM algorithm seeks to find the hyper plane with the maximum margin that separates the data points. Mathematically, this can be formulated as the optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad (10)$$

Subject to the constraints:

$$y_i(w \cdot x_i + b) \geq 1 \text{ for } i = 1,2,\ldots,n \quad (11)$$

Where w is the weight vector perpendicular to the hyper-plane, b is the bias term, and $\|w\|$ denotes the Euclidean norm of w. The term $w.x_i + b$ represents the decision function, and y_i denotes the class label of data point x_i. In the case of non- linearly separable data, SVM can be extended using the kernel trick to map the input data into a higher-dimensional feature space where it becomes linearly separable. This allows SVM to learn complex decision boundaries and handle non-linear relationships between features and target variables. Support Vector Machines offer several advantages, including robustness to over fitting, effectiveness in high-dimensional spaces, and versatility in handling both linear and non-linear classification problems. They are widely used in various domains, including image classification, text categorization, and bioinformatics, making them a valuable tool in the field of machine learning

E. Logistic Regression

Logistic Regression is a widely used statistical method for binary classification tasks. Unlike linear regression, which predicts a continuous output, logistic regression predicts the probability of a binary outcome. It is particularly useful when the dependent variable is categorical and represents two possible outcomes, such as success/failure or yes/no.

Given a dataset $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$ where x_i represents the feature vector and y_i is the corresponding class label, the logistic regression model predicts the probability that y_i=1 given xi by applying the logistic function to a linear combination of the input features. Mathematically, this is expressed as:

$$P(y_i = 1|x_i) = \sigma(w \cdot x_i + b) \quad (12)$$

Where $\sigma(z)$ is the logistic (or sigmoid) function defined as

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (13)$$

Here, w is the weight vector, and b is the bias term. The

logistic function maps any real-valued number into the range (0, 1), which can be interpreted as a probability.

The parameters w and b are estimated by maximizing the likelihood of the observed data, which is equivalent to minimizing the logistic loss function (also known as binary cross-entropy loss):

$$L(w,b) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})] \quad (14)$$

Where $\hat{y_i} = \sigma(w \cdot x_i + b)$ is the predicted probability for the i-th instance

Logistic Regression is popular due to its simplicity, efficiency, and interpretability. It is widely used in various applications such as medical diagnosis, credit scoring, and marketing, where binary classification is required. Despite its name, logistic regression is a linear model in the transformed feature space, making it a powerful tool for binary classification problems

F. Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a class of artificial neural networks (ANNs) that consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. MLPs are widely used for both classification and regression tasks due to their ability to model complex non- linear relationships between input features and target variables.

Given a dataset (x_1, y_1), (x_2, y_2)... (x_n, y_n) where x_i represents the feature vector and yi is the corresponding target vector, the MLP model transforms the input through multiple layers using a series of weighted linear combinations followed by non-linear.

The mathematical formulation of MLP can be summarized as follows:

1. Input Layer:

$$h_0 = x_i \quad (15)$$

2. Hidden Layers: for each hidden layer l = 1, 2, ..., L, where L is the number of hidden layers:

$$h_l = f(W_l h_{l-1} + b_l) \quad (16)$$

Here, W_l and b_l are the weight matrix and bias vector for the l-th layer, and f is the non-linear activation function (e.g., ReLU, sigmoid, or tanh).

3. Output Layer:

$$\hat{y_i} = g(W_{L+1} h_L + b_{L+1}) \quad (17)$$

Where W_{L+1} and b_{L+1} are the weight matrix and bias vector for the output layer, and g is the activation function for the output layer (e.g., softmax for classification or identity for regression).

4. Loss Function: The parameters of the MLP (i.e., weights and biases) are learned by minimizing a loss function $L(\hat{y}, y)$ over the training dataset. For classification, the cross-entropy loss is commonly used, while for regression, the mean squared error (MSE) is often employed

$$L(\hat{y}, y) = -\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{m}(y_{ij}\log(\hat{y_{ij}}))) \quad (18)$$

For classification tasks with m classes, or for regression tasks

$$L(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}((\hat{y_i} - y_i)^2) \quad (19)$$

5. Backpropagation: The MLP model uses the backpropagation algorithm to update the weights and biases by computing the gradient of the loss function with respect to each parameter and applying an optimization algorithm such as gradient descent or its variants.

Multilayer Perceptrons are versatile and capable of capturing intricate patterns in data, making them suitable for a wide range of applications, including image and speech recognition, natural language processing, and financial forecasting.

VII. THE EXPERIMENTS

Performance Measures

A critical measure of an Intrusion Detection System (IDS) is its accuracy, which reflects its effectiveness in detecting and categorizing intrusions or abnormal network behavior. Accuracy is defined as the proportion of correctly classified instances (true positives and true negatives) to the total instances (including true positives, true negatives, false positives, and false negatives). The formula for calculating accuracy is:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}}$$

The F1-score is a metric used to evaluate the performance of an intrusion detection system. It is the harmonic mean of precision and recall and is calculated as follows:

$$F1 = 2 \times \frac{(Precision \times Recall)}{Precision + Recall}$$

- Precision = True Positives / (True Positives + False Positives)
- Recall = True Positives / (True Positives + False Negatives)

Experimental Results

In our study, we conducted five distinct experiments on two datasets: IoT23 and CIC IoT 2023. The goal was to assess the performance of various classifiers combined with Principal Component Analysis (PCA), Kernel Principal Component Analysis (KPCA), and different scaling techniques. The evaluated classifiers include K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), and Logistic Regression (LR). We explored several configurations for these experiments:

In Experiment 1, we fixed the size of the training dataset and varied the number of principal components. This allowed us to assess how the number of components affects classifier performance using different scaling methods.

In Experiment 2, we set the number of principal components at three and progressively increased the size of the training dataset. This approach enabled us to observe how classifier performance changes as the amount of training data grows.

Experiment 3 used RobustScaler, a training size of 2,000, and three principal components. This experiment focused on evaluating the accuracy of each classifier in identifying different types of attacks, offering insights into their performance against specific threat categories.

The fourth experiment focuses on calculating CPU time for various classifiers combined with PCA. This experiment examines how computational efficiency is affected by the choice of PCA dimensionality reduction technique and classifier, across different principal components number.

The fifth experiment investigates the F1-score performance of various classifiers combined with PCA focusing on their effectiveness as the number of principal components increases.

For the last experiment, we fixed the number of training size at 2000 and we use RobustScaler.

The following paragraphs will provide a detailed description of each experiment, along with an analysis of the corresponding results.

Experiment 1

In the first experiment, we randomly selected 10,000 normal data points and 1,000 malicious data points for the training phase. For the testing phase, we selected 5,000 normal data points and 1,000 malicious data points. This selection process was repeated 20 times.

This setup enhances the reliability and generalizability of the results by minimizing biases from a single random split and evaluating variability across multiple iterations. By simulating real-world scenarios and assessing system performance with diverse data samples, it provides valuable insights into the system's generalization abilities and stability. Additionally, this approach ensures statistical significance, enabling efficient resource usage despite repeated trials.

Our objective was to analyze how classifier accuracy varies with different numbers of components and to compare the effectiveness of PCA and KPCA.

After scaling the data, the feature extraction methods were applied with a varying number of components, ranging from 2 to 8 and the transformed data was used to train each classifier.

The classifiers were instantiated with specific parameters to ensure optimal performance, for KNN we consider Euclidean distance metric, for MLP two hidden layers (sizes 100 and 50), ReLU activation, and adaptive learning rate were chosen. For SVM we consider RBF kernel.

We examined three distinct scaling techniques: Standard- Scaler, MinMaxScaler, and RobustScaler.

We divide the results into two sections: the first pertains to IoT23 dataset (Fig.2), and the second focuses on the CIC IoT 2023 dataset (Fig.3).

Each row of the figures corresponds to a different data scaling technique. The accuracies of the classifiers are plotted against the number of components, which vary from 2 to 8.

Under StandardScaler normalization (first row of Fig. 2), the Decision Tree classifier, when combined with PCA, shows a rapid increase in accuracy, reaching near-perfect accuracy with just 3 components and maintaining stability as more components are added. This suggests that the Decision Tree is highly effective for this dataset with minimal feature extraction through PCA. Similarly, Random Forest paired with PCA quickly achieves high accuracy, reflecting Decision Tree's performance. Both classifiers demonstrate stability and resilience with StandardScaler, indicating that tree-based models are particularly well-suited for this IoT dataset. For SVM and MLP, accuracy also improves quickly as more components are added, stabilizing around 4-5 components, although they do not reach the peak accuracy of Decision Tree and Random Forest. Logistic Regression with PCA requires more components to achieve similar accuracy, showing a slower performance increase compared to other classifiers. When paired with KPCA (RBF kernel), Decision Tree and Random Forest continue to perform well, closely matching their PCA performance. However, Logistic Regression and SVM with KPCA exhibit greater sensitivity to the number of components, stabilizing at slightly higher component counts than with PCA.

With MinMaxScaler normalization (second row of Fig. 2), the Decision Tree and Random Forest classifiers with PCA quickly achieve high accuracy, reaching near-optimal performance with just 3 components. These tree-based models maintain stability across all tested component numbers, highlighting their effectiveness with minimal dimensionality. For MLP and SVM with PCA, high accuracy is achieved with slightly more variability than the tree-based models, particularly at lower component numbers, though they stabilize around 4 components

Logistic Regression with PCA again requires more components to reach comparable accuracy, showing greater sensitivity in the early component range. When paired with KPCA, Decision Tree and Random Forest perform similarly to PCA, achieving high accuracy early on. However, Logistic Regression and SVM with KPCA exhibit significant variability, indicating sensitivity to both the number of components and the scaling method. This suggests that these classifiers may require additional parameter tuning when used with KPCA on this dataset.

With RobustScaler normalization (as shown in the third row of Fig.2), the Decision Tree and Random Forest classifiers combined with PCA maintain strong performance, achieving high accuracy with only 3-4 components and remaining stable as more components are added. This demonstrates that tree-based models are highly versatile across different scaling methods when paired with PCA. MLP and SVM with PCA show slightly more variation in accuracy under RobustScaler, particularly at lower component numbers, although they stabilize as more components are introduced.

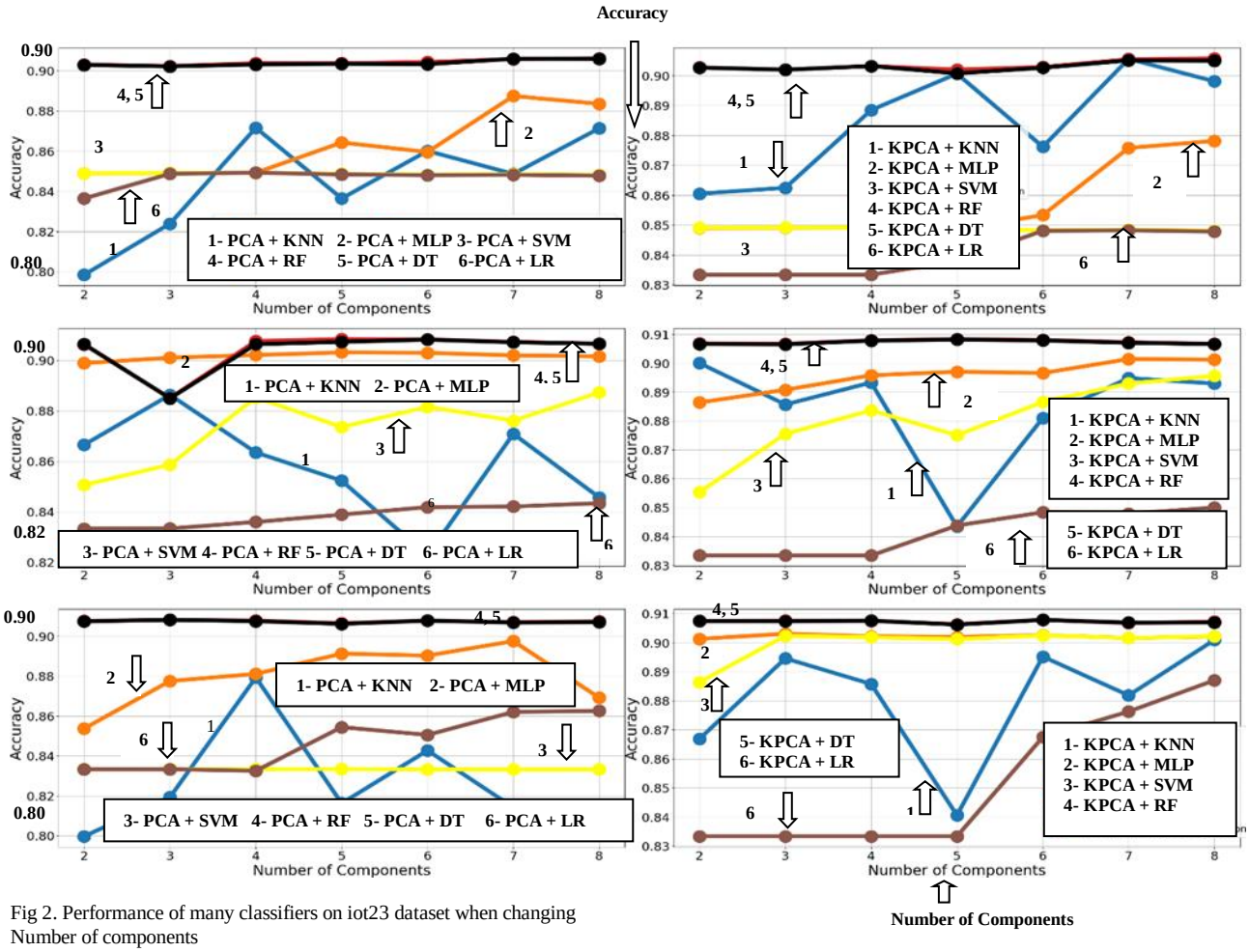

Fig 2. Performance of many classifiers on iot23 dataset when changing Number of components

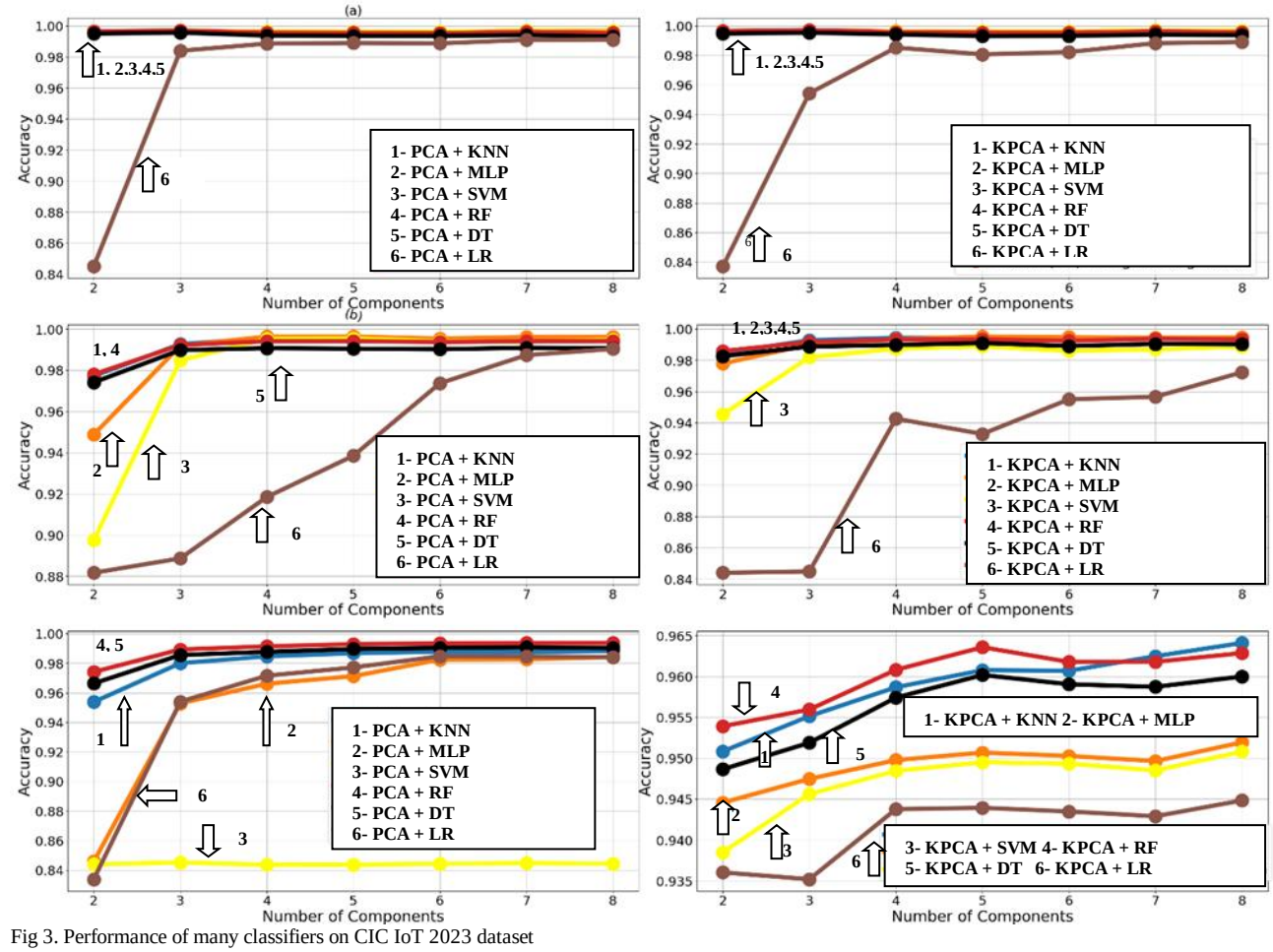

Fig 3. Performance of many classifiers on CIC IoT 2023 dataset

Logistic Regression with PCA displays the most instability under RobustScaler, starting with lower accuracy and improving gradually as additional components are added. When using KPCA, Decision Tree and Random Forest again show high accuracy, demonstrating robustness even with the RobustScaler transformation. However, Logistic Regression and SVM with KPCA exhibit noticeable sensitivity to the number of components, with greater fluctuations in accuracy compared to PCA, especially with fewer components. This suggests that these classifiers may not be as compatible with KPCA when using RobustScaler on the IoT23 dataset.

Fig.3 illustrates the performance of the classifiers on the CIC IoT 2023 dataset under varying normalization methods.

In the first row of Fig.3, where StandardScaler normalization is applied, the Decision Tree classifier combined with PCA quickly achieves high accuracy, reaching near-perfect results with just 3 components and maintaining this level of performance as more components are added. This indicates that the Decision Tree model is well-suited for the CIC IoT 2023 dataset, requiring minimal feature extraction through PCA. Similarly, Random Forest with PCA shows rapid accuracy improvement, stabilizing at 3 components and remaining resilient as more components are added. SVM and MLP also improve in accuracy with PCA under StandardScaler, although they reach peak performance at slightly higher component counts (around 4-5). Logistic Regression, however, requires more components to achieve similar accuracy, with a more gradual improvement compared to the other classifiers. When paired with KPCA (RBF kernel), both Decision Tree and Random Forest continue to perform well, achieving high accuracy quickly and stabilizing across components. On the other hand, Logistic Regression and SVM with KPCA exhibit greater fluctuations in accuracy, particularly at lower component counts, suggesting sensitivity to the number of components in this configuration.

In the second row of Fig.3, where MinMaxScaler normalization is applied, the Decision Tree and Random Forest classifiers combined with PCA continue to show strong performance, achieving high accuracy with as few as 3 components and maintaining stability across the component range. This further confirms their effectiveness for the CIC IoT 2023 dataset under MinMaxScaler. MLP and SVM also reach high accuracy with PCA under MinMaxScaler, although they exhibit slightly more variability, particularly at lower component counts. Logistic Regression with PCA shows a more gradual improvement in accuracy, requiring additional components to achieve similar performance to the other classifiers.

When paired with KPCA, Decision Tree and Random Forest maintain strong performance, achieving high accuracy and stability across the number of components. However, SVM and Logistic Regression with KPCA demonstrate greater fluctuations in accuracy, especially at lower component counts. This suggests that these models may need further tuning when combined with KPCA under MinMaxScaler for this dataset

Under RobustScaler normalization (third row of Fig. 3), Decision Tree and Random Forest with PCA maintain their trend of strong performance, reaching high accuracy by 3-4 components and remaining stable across additional components. This demonstrates the robustness of tree-based models when paired with PCA, regardless of scaling method.

MLP and SVM with PCA under RobustScaler exhibit more variability in accuracy, particularly at lower component counts, though they stabilize with higher components. Logistic Regression with PCA shows the most instability, with a more gradual increase in accuracy across components. For KPCA, Decision Tree and Random Forest remain strong performers, displaying high accuracy with only a few components and remaining stable. However, SVM and Logistic Regression with KPCA show marked sensitivity to the number of components, exhibiting greater variability and requiring additional components to stabilize, which suggests a less compatible fit for these models under RobustScaler with KPCA on this dataset.

Experiment 2

Figures 4 and 5 represent the results of Experiment 2. In this experiment, the training dataset size was gradually increased, while the number of principal components was fixed at three. This setup provides insights into how each classifier adapts to larger training size under different normalization methods

In the first row of Fig.4, we can see that under StandardScaler normalization, PCA+ Decision Tree and PCA+ Random Forest maintain consistently high accuracy as the training data increases. Decision Tree, in particular, demonstrates exceptional stability.

It effectively captures the data structure even with larger training sizes. SVM, MLP, and KNN classifiers deliver moderate performance. Their accuracy is stable but lower compared to tree-based models. Logistic Regression performs poorly throughout. It struggles to handle the complex feature transformations produced by PCA. When KPCA with an RBF kernel is used, Decision Tree and Random Forest still perform well. However, KPCA offers no significant improvement over PCA in this case

In the second row of Fig.4, MinMaxScaler normalization shows that Decision Tree and Random Forest with PCA deliver high accuracy with minimal variation. This highlights the reliability of tree-based models with MinMaxScaler. SVM and MLP combined with KPCA exhibit more fluctuation as the training size increases. This may be due to MinMaxScaler's impact on feature distribution transformations in kernel-based methods. Logistic Regression performs consistently poorly, indicating that linear models may fail to capture the complexity of the transformed data effectively.

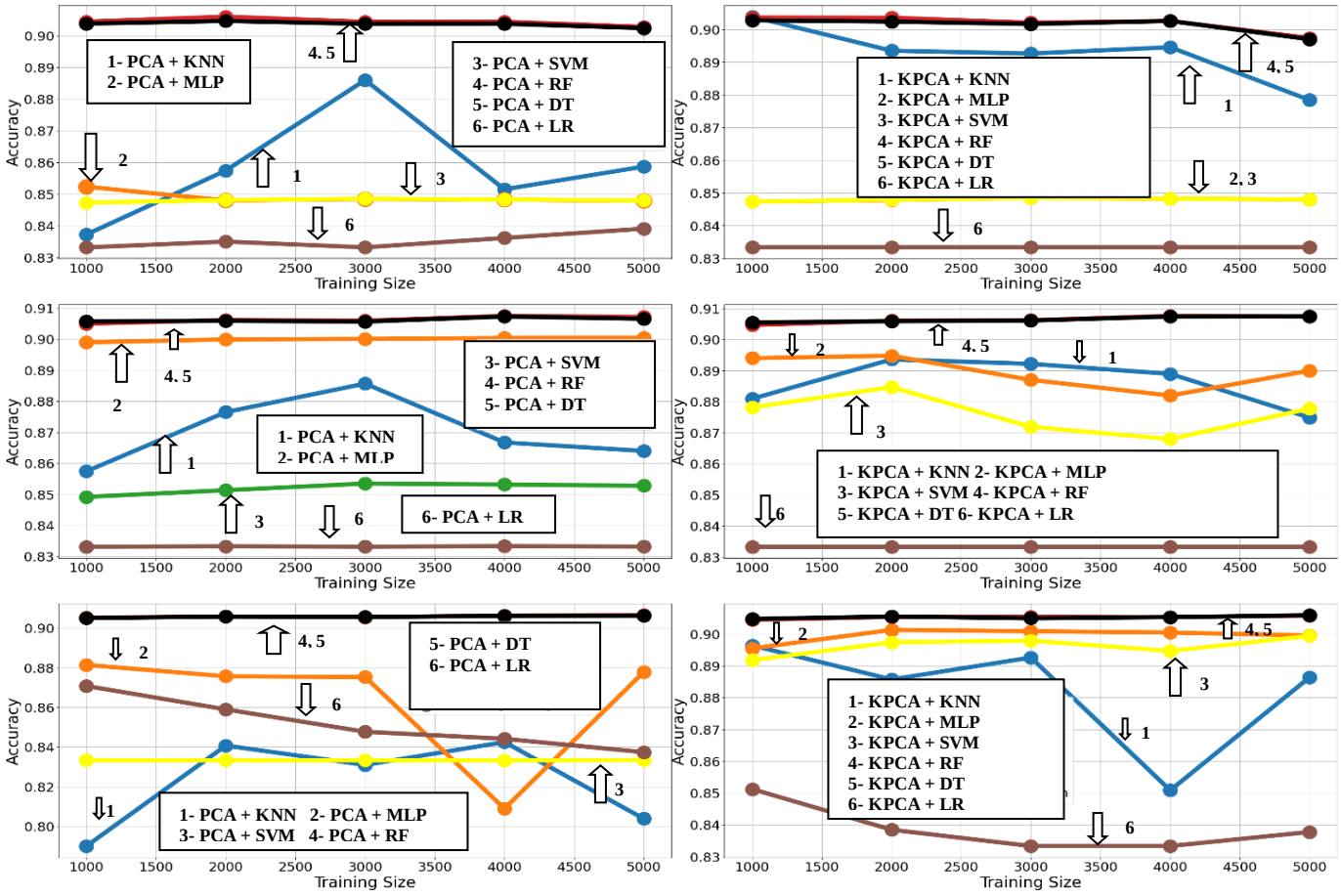

Fig. 4. Performance of many classifiers on iot23 dataset when changing Training size

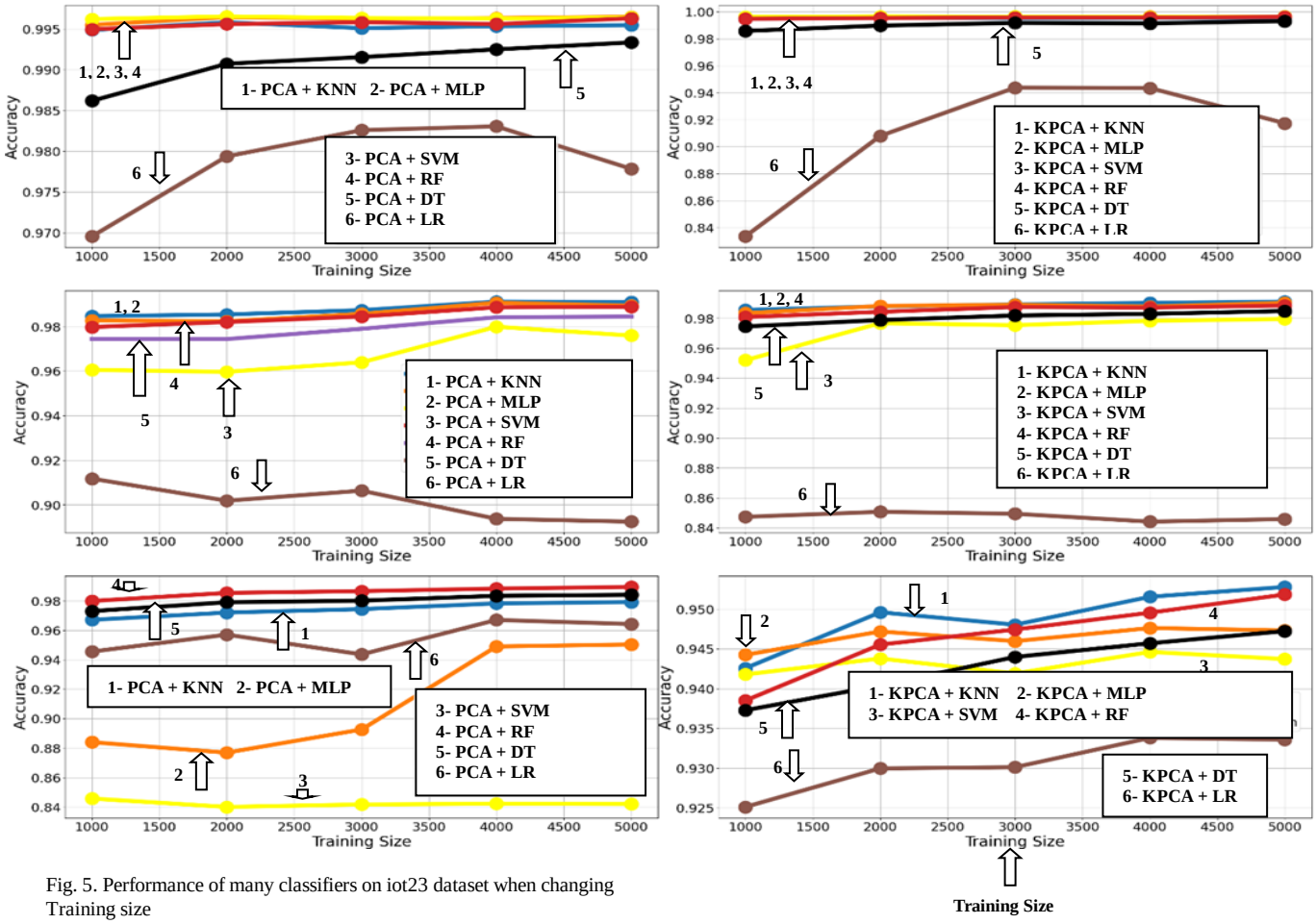

Fig. 5. Performance of many classifiers on iot23 dataset when changing Training size

In the third row of Fig.4, RobustScaler normalization is applied. We can see clearly that Tree-based models such as Decision Tree and Random Forest maintain consistent accuracy and stability as the training size grows. Non-tree classifiers, especially SVM and MLP, exhibit more fluctuation. This is likely due to how RobustScaler handles outliers within the PCA/KPCA-transformed feature space. The variability highlights the difficulties some classifiers encounter with outlier-sensitive scaling. Meanwhile, Decision Tree and Random Forest continue to classify IoT attack data effectively in this setup.

Fig.5 illustrates the results of experiment 2 using CIC IoT 2023 dataset.

Under StandardScaler (first row), Decision Tree and Random Forest with PCA consistently achieve high accuracy, with Decision Tree slightly ahead. Both leverage PCA-transformed features effectively. KPCA models like SVM and MLP show more variability, while Logistic Regression performs poorly, struggling with the transformed data.

With MinMaxScaler (second row), Decision Tree and Random Forest maintain high accuracy with minimal fluctuation, reinforcing their adaptability to MinMaxScaler-transformed features. KPCA-based SVM and MLP show greater variability, and Logistic Regression remains underwhelming.

Under RobustScaler (third row), Decision Tree and Random Forest continue to perform well, unaffected by outlier handling. SVM and MLP experience more variability, reflecting sensitivity to outliers and KPCA transformations, while Logistic Regression again shows low accuracy, unable to manage the dataset's complexities effectively.

Experiment 3

The analysis of Experiment 3 on the IoT23 dataset reveals significant insights into the effectiveness of various classifiers in identifying attack types under different dimensionality reduction methods (PCA and KPCA) with RobustScaler. The Table VI contains every individual attack name with it corresponding number. The Table II highlights distinct patterns in classifier accuracy, with tree-based classifiers demonstrating stable performance across multiple attack categories, whereas other classifiers like MLP and SVM show variability depending on the reduction method and attack type.

The Decision Tree and Random Forest classifiers achieve consistently high accuracy across the majority of attack types, demonstrating robust detection capabilities. Both classifiers perform exceptionally well on attacks such as DDoS and C&C-HeartBeat-FileDownload, achieving close to or even perfect accuracy scores (0.9995 and above). For example, the Decision Tree classifier reaches 100% accuracy for C&C-HeartBeat-FileDownload attacks using both PCA and KPCA, while Random Forest exhibits similar stability across these categories. This indicates that tree-based classifiers are well-suited to capture the underlying structure in PCA/KPCA transformed features, showing resilience in accurately identifying both simpler and more nuanced attack types.

KNN performs reasonably well on certain attack types, particularly with PCA. For instance, it achieves 99.98% and 100% accuracy for FileDownload and C&C-HeartBeat-FileDownload, respectively, suggesting that KNN benefits from PCA's linear transformations, which may better approximate distances in the dataset. However, KNN's accuracy declines substantially when KPCA is applied, particularly for complex attacks like Okiru and PartOfAHorizontalPortScan, where it only reaches accuracies of 0.4930 and 0.5333. This indicates that KNN may struggle with the non-linear transformations introduced by KPCA, impacting its ability to detect more intricate patterns in certain attack types

Both MLP and SVM classifiers exhibit significant variability across attack types, particularly when using PCA. For example, MLP performs poorly in detecting attacks such as DDoS and Okiru, scoring 0.0000 in both cases, while SVM with PCA also fails on DDoS and PartOfAHorizontalPortScan. This suggests that MLP and SVM may have difficulty capturing complex patterns when constrained to a lower-dimensional, linear space. KPCA provides some performance improvements for these classifiers in detecting certain attacks, such as FileDownload for MLP (0.8617) and C&C-Mirai for SVM (0.9970), though both classifiers continue to struggle with attacks requiring more complex decision boundaries, like C&C-HeartBeat.

Logistic Regression shows the least effectiveness among the classifiers, especially when paired with PCA. It fails to detect multiple attacks, including PartOfAHorizontalPortScan, DDoS, and C&C-HeartBeat, where it scores 0. While KPCA provides slight performance improvements for some attacks, such as FileDownload and C&C-Mirai (0.8676 and 0.8222, respectively), its overall performance remains low. This outcome confirms that Logistic Regression's linear nature is inadequate for capturing the complexities within this IoT dataset, especially for non-linear attack types.

Overall, tree-based classifiers, particularly with PCA, are consistently more effective across various attack categories, highlighting their adaptability and reliability in this experimental setup. KPCA, although beneficial for certain classifiers, does not universally enhance detection rates across attack types, particularly for MLP, KNN, and SVM, which exhibit more variable accuracy under this method. The experiment underscores the superior performance of tree-based models with PCA, as well as the limitations of linear models like Logistic Regression for complex, non-linear IoT data classification tasks.

The analysis of individual attack detection accuracy for the CIC IoT 2023 dataset (Tables III, IV and V) reveals significant insights into classifier performance across various attack types.

TABLE II. INDIVIDUAL ATTACK ACCURACY FOR IoT23 DATASET

Classifier	EM	1	2	3	4	5	6	7	8	9	10
KNN	PCA	0.5816	0.3538	0.8094	0.806	0.7255	**0.5604**	**0.9998**	**0.9988**	1.0000	0.9986
KNN	KPCA	0.5333	0.4930	0.5893	0.701	0.7454	0.5243	0.8674	0.9845	1.0000	1.0000
MLP	PCA	0.0203	0.0000	0.0000	0.110	**1.0000**	0.0000	0.8233	0.0844	0.0104	0.0104
MLP	KPCA	0.5726	0.2903	0.7056	0.297	0.0749	0.1047	0.4053	0.8617	0.9082	0.9980
SVM	PCA	0.0000	0.1518	0.0000	0.606	**1.0000**	0.4291	0.9231	0.4594	0.5161	0.5505
SVM	KPCA	0.3865	0.2903	0.6335	0.021	0.0000	0.1027	0.3912	0.8188	0.9416	0.9970
Random Forest	PCA	**0.6124**	0.5719	0.9998	0.809	0.9139	0.4005	0.9998	**0.9988**	1.0000	0.9986
Random Forest	KPCA	**0.6151**	0.6143	0.9976	0.813	0.9095	0.3948	0.8662	0.9890	1.0000	0.9992
Decision Tree	PCA	**0.6111**	**0.5721**	**0.9995**	**0.8147**	0.9195	0.3944	**0.9998**	0.9986	**1.0000**	0.9986
Decision Tree	KPCA	**0.6162**	**0.6143**	**0.9988**	**0.8157**	0.8342	0.3944	0.8402	0.9960	**1.0000**	**1.0000**
Logistic Regression	PCA	0.0000	0.0000	0.0106	0.0000	0.5687	0.0095	0.8363	0.0099	0.0000	0.0530
Logistic Regression	KPCA	0.1046	0.2874	0.4052	0.0000	0.0000	0.0731	0.0813	0.8169	0.8676	0.8222

TABLE III. FIRST PART OF INDIVIDUAL ATTACK ACCURACY FOR CIC IoT 2023 DATASET

Classifier	11	12	13	14	15	16	17	18	19	20	21	22
PCA + KNN	0.7584	0.9090	0.8479	0.4548	0.8024	0.8145	0.8690	0.7553	0.7572	0.7784	0.7666	0.5024
KPCA + KNN	0.8958	0.7740	0.8214	0.4375	0.6500	0.6325	0.6673	0.2895	0.7319	0.8267	0.7563	0.4070
PCA + MLP	0.0133	0.0073	0.0000	0.2871	0.0330	0.0344	0.0000	0.6008	0.0408	0.0135	0.0130	0.5484
KPCA + MLP	0.6486	0.3054	0.5723	0.4438	0.2016	0.4066	0.3807	0.1288	0.5094	0.6779	0.3497	0.1788
PCA + SVM	0.0704	0.0237	0.0105	0.0576	0.0272	0.0805	0.0510	0.2228	0.0089	0.0279	0.0035	0.0125
KPCA + SVM	0.1464	0.0445	0.0484	0.1146	0.0401	0.3066	0.2395	0.0538	0.0165	0.0120	0.0846	0.0866
PCA + Random Forest	**0.9471**	**0.9604**	**0.9637**	0.8215	0.9435	0.9348	0.9406	0.9436	0.9554	0.9535	0.9661	0.8251
KPCA + Random Forest	0.9060	0.8133	0.9299	0.4536	0.7080	0.6775	0.7365	0.3251	0.7524	0.8327	0.7945	0.4270
PCA + Decision Tree	**0.9543**	**0.9677**	**0.9670**	**0.8311**	**0.9604**	**0.9491**	**0.9559**	**0.9526**	**0.9263**	**0.9502**	**0.9511**	**0.8544**
KPCA + Decision Tree	0.9142	0.7158	0.9069	0.4390	0.6283	0.6752	0.7020	0.3125	0.6887	0.8583	0.7284	0.4490
PCA + Logistic Regression	0.0017	0.0114	0.0011	0.1398	0.0076	0.0108	0.0129	0.1096	0.0009	0.0008	0.0023	0.0947
KPCA + Logistic Regression	0.0138	0.0357	0.0966	0.0198	0.0801	0.3616	0.2305	0.0448	0.0453	0.0000	0.0724	0.0867

TABLE IV. SECOND PART OF INDIVIDUAL ATTACK ACCURACY FOR CIC IoT 2023 DATASET

23	24	25	26	27	28	29	30	31	32	33	34
0.3250	0.8041	0.8075	0.7096	0.8041	0.1901	0.2032	0.5433	0.1683	0.3203	0.7388	0.3676
0.0717	0.7529	0.6809	0.3609	0.7186	0.1290	0.0771	0.6611	0.1191	0.2565	0.4237	0.1881
0.3561	0.0300	0.0587	0.0507	0.0305	0.0183	0.1835	0.3756	0.0553	0.1124	0.0317	0.0541
0.0775	0.7136	0.5329	0.3548	0.7053	0.0834	0.0146	0.6424	0.0364	0.0992	0.4530	0.1165
0.2344	0.0094	0.0526	0.0000	0.0382	0.0245	0.1560	0.0012	0.0082	0.0078	0.0103	0.0133
0.0162	0.5084	0.3008	0.3593	0.3963	0.0548	0.0180	0.5559	0.0141	0.0769	0.2568	0.0826
0.4599*	**0.9430***	**0.9630***	**0.8655***	**0.9271***	**0.3314***	**0.3098***	**0.8303***	**0.2592***	**0.3497***	**0.8736***	**0.4668***
0.0563	0.7924	0.7139	0.4028	0.7548	0.1339	0.0752	0.6513	0.1176	0.2660	0.4341	0.1971
0.3890	**0.9406***	**0.9452***	**0.8352***	**0.8939***	**0.3201***	**0.2643***	**0.8070***	**0.2613**	**0.3511***	**0.8413***	**0.4330**
0.0617	0.6842	0.7012	0.3492	0.7548	0.1391	0.0743	0.6353	0.1150	0.2631	0.4138	0.1909
0.1548	0.0013	0.0091	0.0000	0.0166	0.0369	0.0747	0.0000	0.0111	0.0532	0.0067	0.0092
0.0358	0.3804	0.3221	0.2432	0.1148	0.0359	0.0515	0.6196	0.0078	0.0815	0.1636	0.0768

TABLE V. THIRD PART OF INDIVIDUAL ATTACK ACCURACY FOR CIC IoT 2023 DATASET

35	36	37	38	39	40	41	42	43
0.3345	0.6879	0.4783	0.3777	0.3083	0.2462	0.2899	0.4640	0.4820
0.2404	0.3998	0.3354	0.2844	0.2298	0.1551	0.2342	0.3814	0.4597
0.1905	0.5259	0.3522	0.2009	0.0869	0.1208	0.1579	0.4931	0.0911
0.9304	0.5529	0.4001	0.0136	0.0500	0.0719	0.0308	0.0629	0.1524
0.0657	0.0350	0.2211	0.0356	0.0390	0.3066	0.0403	0.0149	0.0173
0.0441	0.4933	0.2091	0.0328	0.0503	0.0275	0.0336	0.1376	0.0789
0.3975*	**0.8772***	**0.7054***	**0.4416***	**0.3816***	**0.3960***	**0.3991***	**0.5412***	**0.4987***
0.2321	0.3007	0.3056	0.2752	0.2053	0.1591	0.2325	0.3955	0.4322
0.3523*	**0.8275***	**0.7018**	**0.4399***	**0.3439***	**0.3978***	**0.3962***	**0.5495***	**0.5132***
0.2191	0.3560	0.2999	0.2778	0.2220	0.1526	0.2212	0.3826	0.4471
0.2212	0.1088	0.0929	0.0607	0.0210	0.0158	0.0492	0.0649	0.0176
0.0238	0.5659	0.2443	0.0300	0.0290	0.0308	0.0567	0.0126	0.0477

Among the tested classifiers, PCA + Random Forest and PCA + Decision Tree consistently achieve high accuracy, particularly for complex Distributed Denial of Service (DDoS) attacks. These two classifiers demonstrate strong performance for attacks like DDoS-ACK Fragmentation, DDoS-ICMP Fragmentation, and DDoS-HTTP Flood, often reaching over 90% accuracy, as marked by the asterisks. This result suggests that PCA-based dimensionality reduction, when combined with ensemble methods like Random Forest and Decision Tree, is highly effective in identifying sophisticated DDoS patterns within IoT data.

Conversely, KPCA-based classifiers, such as KPCA + KNN and KPCA + MLP, generally struggle to achieve comparable accuracy levels across most attacks. For example, KPCA + KNN manages moderate success on certain DDoS attacks, including DDoS-ACK Fragmentation and Mirai-greip Flood, but its performance significantly declines for more complex or subtle attacks, like MITM-ArpSpoofing and Recon-HostDiscovery. KPCA + MLP, in particular, has difficulty with various attacks, frequently falling below 40% accuracy. This disparity suggests that KPCA may not be as suitable as PCA for dimensionality reduction on this IoT dataset, likely due to PCA's more straightforward projection onto principal components, which might better capture the underlying patterns in network traffic data.

The analysis also highlights certain attack types that pose challenges across all classifiers. Reconnaissance and spoofing attacks, such as Recon-OSScan, DNS Spoofing, and MITM-ArpSpoofing, consistently exhibit lower detection rates, regardless of the classifier and dimensionality reduction technique used. These attacks likely involve subtle patterns or lower signal-to-noise ratios that make them harder to detect with the current feature set and models. Even high-performing classifiers like PCA + Random Forest and PCA + Decision Tree struggle with these attack types, achieving moderate accuracy at best. This indicates a need for potential refinement, such as additional feature engineering or the exploration of alternative algorithms specifically tailored to detect subtle anomalies.

Certain attack types demonstrate mixed detection rates, suggesting a nuanced response by the classifiers. For example, XSS and Recon-PortScan attacks yield varied results, with PCA + Random Forest generally outperforming other classifiers but still not reaching high accuracy. This inconsistency might be attributed to the attack behavior within IoT environments, where reconnaissance activities or cross-site scripting could exhibit unique traffic patterns that are difficult to generalize.

Experiment 4

This experiment focuses on measuring the required CPU time of various classifiers combined with PCA using the RobustScaler preprocessing method. The goal is to compare the computational efficiency of these methods.

For the IoT23 dataset, the results are illustrated in Figure 6. Among the combinations of PCA and the classifiers, the PCA+MLP exhibited the highest CPU time, in contrast, the combination of the feature extraction method and KNN, Random Forest, Decision Tree, and Logistic Regression, maintained consistently low CPU times regardless of the number of components

The results for the CICIoT 2023 dataset are shown in the Figure 7. The trends for PCA are similar to those seen in the IoT23 dataset. MLP required the highest CPU time, especially as the number of components increased. Other classifiers, like Logistic Regression, Decision Tree, and KNN, had stable and low CPU times across all configurations.

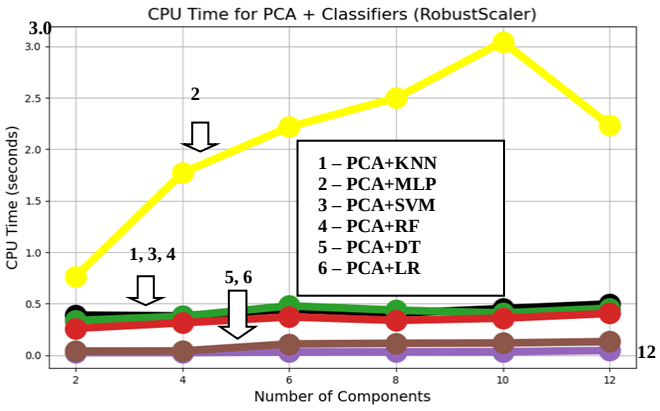

Fig.6 CPU Times of PCA +classifiers vs. Number of Components for iot23 dataset

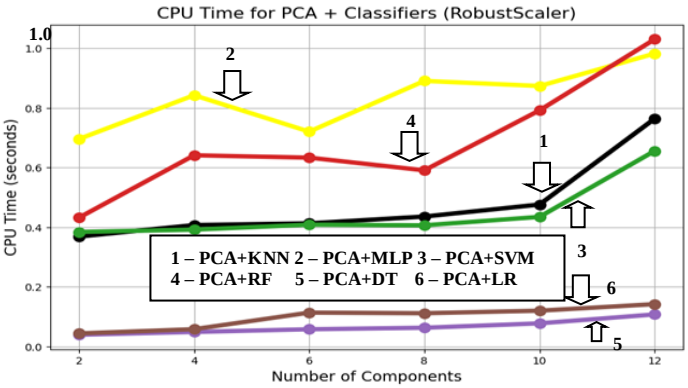

Fig.7 CPU Times of PCA +classifiers vs. Number of Components for CIC IoT23

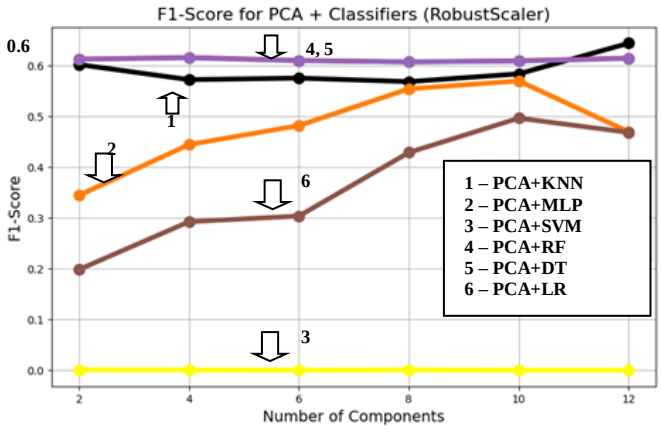

Fig.8. F1-Score of PCA +classifiers vs. Number of Components for IoT23

Experiment 5

The figure 8 represents the F1-score trends for classifiers integrated with PCA on the IoT23 dataset. Among the classifiers, Decision Tree and Random Forest emerge as the most effective, achieving F1-scores consistently around 0.6. This consistency highlights the robustness of tree-based models when handling IoT datasets, even after dimensionality reduction. KNN also performs competitively, showing stable and moderate F1-scores across all PCA components, suggesting its suitability for this dataset.

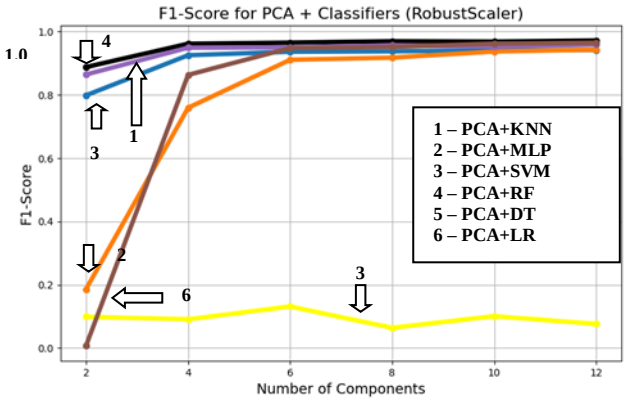

Fig.9. F1-Score of PCA +classifiers vs. Number of Components for CIC IoT2023

The MLP classifier, although improving slightly with additional components, maintains a lower F1-score compared to the top-performing models, reflecting its limited adaptability or the dataset's potential challenges for deep learning. Similarly, Logistic Regression steadily increases but does not exceed an F1-score of 0.4, indicating that linear classifiers are less effective in capturing the complexities of this dataset. The SVM classifier, in contrast, performs poorly, remaining near an F1-score of zero throughout, which could result from insufficient kernel customization or scalability issues with PCA-transformed data.

These observations indicate that the IoT23 dataset benefits most from tree-based methods, which can handle complex patterns efficiently, while simpler or computationally intensive models struggle.

The figure 9 highlights the performance of the same classifiers on the CICIoT23 dataset. Here, Decision Tree and Random Forest achieve near-perfect F1-scores (close to 1.0) as the number of PCA components increases. These results suggest that the CICIoT23 dataset contains well-defined patterns, allowing these classifiers to excel in intrusion detection. KNN and Logistic Regression also show significant improvement, stabilizing at high F1-scores (above 0.8) with fewer components. This indicates that simpler classifiers can also achieve good performance on the CICIoT23 dataset when supported by effective dimensionality reduction.

While MLP improves slightly compared to its performance on the IoT23 dataset, it still struggles to maintain consistency, reflecting its sensitivity to dataset characteristics. The rapid performance gains across classifiers, particularly with fewer PCA components, suggest that the CIC23 dataset is more separable and easier to classify than IoT23. The strong performance of Decision Tree and Random Forest on both datasets highlights their versatility and suitability for intrusion detection tasks in IoT networks.

TABLE VI. EVERY ATTACK NAME WITH IT CORRESPONDING NUMBER

N°	Attack's Name	N°	Attack's Name
1	PartOfAHorizontalPortScan	22	DDoS-UDP_Flood
2	Okiru	23	MITM-ArpSpoofing
3	DDoS	24	DDoS-ACK_Fragmentation
4	C&C-HeartBeat	25	Mirai-greip_flood
5	C&C-Torii	26	DoS-HTTP_Flood
6	C&C	27	DDoS-ICMP_Fragmentation
7	C&C-FileDownload	28	Recon-PortScan
8	FileDownload	29	DNS_Spoofing
9	C&C-HeartBeat FileDownload	30	DDoS-UDP_Fragmentation
10	C&C-Mirai	31	Recon-OSScan
11	DDoS-RSTFINFlood	32	XSS
12	DoS-TCP_Flood	33	DDoS-HTTP_Flood
13	DDoS-ICMP_Flood	34	Recon-HostDiscovery
14	DoS-UDP_Flood	35	CommandInjection
15	DoS-SYN_Flood	36	VulnerabilityScan
16	Mirai-greeth_flood	37	DDoS-SlowLoris
17	DDoS-SynonymousIP_Flood	38	Backdoor_Malware
18	Mirai-udpplain	39	BrowserHijacking
19	DDoS-SYN_Flood	40	DictionaryBruteForce
20	DDoS-PSHACK_Flood	41	SqlInjection
21	DDoS-TCP_Flood	42,43	Recon-PingSweep, Uploading_Attack

VIII. CONCLUSION

This study evaluates preprocessing techniques and many machine learning classifiers for enhancing IoT network security through intrusion detection. Integrating PCA with Decision Tree and Random Forest classifiers significantly improves accuracy while managing high-dimensional IoT data. A comparative analysis of scalers (StandardScaler, MinMaxScaler, and RobustScaler) highlights the importance of tailoring preprocessing to dataset characteristics. Furthermore, the results suggest that this combination is particularly effective in identifying sophisticated DDoS patterns within IoT data.

Future work should focus on optimizing the trade-off between computational cost and detection accuracy by exploring lightweight models, hybrid dimensionality reduction techniques, and scalable architectures.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, pp. 273–297, 1995.

[2] J. Azimjonov and T. Kim, "Designing accurate lightweight intrusion detection systems for iot networks using fine-tuned linear svm and feature selectors," Computers & Security, vol. 137, p. 103598, 2024.

[3] H. . Ghasemi and S. Babaie, "A new intrusion detection system based on svm–gwo algorithms for internet of things," Wireless Networks, pp. 1–13, 2024.

[4] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks," Future internet, vol. 12, no. 3, p. 44, 2020.
[5] S. M. Taghavinejad, M. Taghavinejad, L. Shahmiri, M. Zavvar, and M. H. Zavvar, "Intrusion detection in iot-based smart grid using hybrid decision tree," in 2020 6th International Conference on Web Research (ICWR). IEEE, 2020, pp. 152–156.
[6] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for iot security using decision tree and gradient boosting," The Journal of Supercomputing, vol. 79, no. 3, pp. 3392–3411, 2023.
[7] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, Logistic regression. Springer, 2002.
[8] C. Ioannou and V. Vassiliou, "An intrusion detection system for constrained wsn and iot nodes based on binary logistic regression," in Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2018, pp. 259–263.
[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE transactions on information theory, vol. 13, no. 1, pp. 21–27, 1967.
[10] M. Mohy-Eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for iot security using k- nn classifier and feature selection," Multimedia Tools and Applications, vol. 82, no. 15, pp. 23 615–23 633, 2023.
[11] C. A. De Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral, and G. dos Santos Vieira, "Hybrid approach to intrusion detection in fog-based iot environments," Computer Networks, vol. 180, p. 107417, 2020.
[12] E. Zyad and A. B. Mohammed, "Evaluation of PCA Variants for Intrusion Detection in IoT Networks," in Sixth International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, 2024, pp. 1-5
[13] Z. Elkhadir and M. Achkari Begdouri, "Enhancing internet of things attack detection using principal component analysis and kernel principal component analysis with cosine distance and sigmoid kernel," International Journal of Electrical & Computer Engineering, vol. 15, no 1, pp. 1099–1108, 2025.
[14] A. Rosay, F. Carlier, and P. Leroux, "Mlp4nids: An efficient mlp- based network intrusion detection for cicids2017 dataset," in Machine Learning for Networking: Second IFIP TC 6 International Conference, MLN 2019, Paris, France, December 3–5, 2019, Revised Selected Papers 2. Springer, 2020, pp. 240–254.h
[15] Y. Javed and N. Rajabi, "Multi-layer perceptron artificial neural network based iot botnet traffic classification," in Proceedings of the Future Technologies Conference (FTC) 2019: Volume 1. Springer, 2020, pp.973–984.
[16] J. B. Awotunde, F. E. Ayo, R. Panigrahi, A. Garg, A. K. Bhoi, and P. Barsocchi, "A multi-level random forest model-based intrusion detection using fuzzy inference system for internet of things networks," International Journal of Computational Intelligence Systems, vol. 16, no. 1, p. 31, 2023.
[17] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly detection ids for detecting dos attacks in iot networks based on machine learning algorithms," Sensors, vol. 24, no. 2, p. 713, 2024.
[18] M. M. Inuwa and R. Das, "A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on iot networks," Internet of Things, vol. 26, p. 101162, 2024.
[19] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," Internet of Things, vol. 7, p. 100059, 2019.
[20] M. O. Pahl, and F. X. Aubet, "DS2OS traffic traces," 2018.
[21] S. Garcia, A. Parmisano, and M.J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Stratosphere Lab., Praha, Czech Republic, Tech. Rep, 2020.
[22] E.C.P. Neto, S. Dadkhah, R. ferreira, and al., "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensors, vol. 23, no 13, p. 5941, 2023.
[23] Z. Elkhadir, K. Chougdali, and M. Benattou, "Intrusion detection system using pca and kernel pca methods," in Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015. Springer, 2016, pp. 489–497.
[24] T. M. Pattewar and H. A. Sonawane, "Neural network based intrusion detection using bayesian with pca and kpca feature extraction," in 2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS). IEEE, 2015, pp. 83–88.
[25] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid kpca and svm with ga model for intrusion detection," Applied Soft Computing, vol. 18, pp. 178–184, 2014.
[26] I. Jolliffe, Principal component analysis. Wiley Online Library, 2002.
[27] B. Schölkopf, A. Smola, and K.-R. Mu¨ller, "Kernel principal component analysis," in International conference on artificial neural networks. Springer, 1997, pp. 583–588.