Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 6 (2024) 788 - 798 e-ISSN: 2580-0760

IoT Security: Botnet Detection Using Self-Organizing Feature Map and Machine Learning

Susanto[1]*, Deris Stiawan[2], Budi Santoso[3], Alex Onesimus Sidabutar[4], M. Agus Syamsul A[5], Mohd. Yazid Idris[6], Rahmat Budiarto[7],
[1,3,4]Department of Informatica, Faculty of Engineering Science, Universitas Bina Insan, Lubuklinggau, Indonesia
[2]Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya, Palembang, Indoensia
[5]Department of Computer Engineering, Faculty of Engineering Science, Universitas Bina Insan, Lubuklinggau, Indonesia
[6]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia
[7]College of Computer Science and IT, Al-Baha University, Alaqiq, Saudi Arabia
[1]susanto@univbinainsan.ac.id, [2]deris@unsri.ac.id, [3]budisantoso@univbinainsan.ac.id, [4]alextkj12@gmail.com,
[5]mas.arifin@univbinainsan.ac.id, [6]yazid@utm.my, [7]rahmat@bu.edu.sa

Abstract

The rapid advancement of Internet of Things (IoT) technology has created potential for progress in various aspects of life. However, the increasing number of IoT devices also raises the risk of cyberattacks, particularly IoT botnets often exploited by attackers. This is largely due to the limitations of IoT devices, such as constraints in capacity, power, and memory, necessitating an efficient detection system. This study aims to develop a resource-efficient botnet detection system by using the Self-Organizing Feature Map (SOFM) dimensionality reduction method in combination with machine learning algorithms. The proposed method includes a feature engineering process using SOFM to address high-dimensional data, followed by classification with various machine learning algorithms. The experiments evaluate performance based on accuracy, sensitivity, specificity, False Positive Rate (FPR), and False Negative Rate (FNR). Results show that the Decision Tree algorithm achieved the highest accuracy rate of 97.24%, with a sensitivity of 0.9523, specificity of 0.9932, and a fast execution time of 100.66 seconds. The use of SOFM successfully reduced memory consumption from 3.08 GB to 923MB. Experimental results indicate that this approach is effective for enhancing IoT security in resource-constrained devices.

Keywords: *Botnet; IoT; Feature Engineering; SOFM; Machine Learning*

How to Cite: Susanto, "IoT Security: Botnet Detection Using Self-Organizing Feature Map and Machine Learning", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 6, pp. 788 - 798, Dec. 2024.
DOI: https://doi.org/10.29207/resti.v8i6.5871

1. Introduction

The swift progress and adoption of intelligent and Internet of Things (IoT)-based technologies have facilitated numerous potential advancements across various facets of life [1]. This has become a new paradigm that transforms traditional lifestyles into high-tech living. The IoT has brought about transformations in the form of smart transportation, smart cities pollution control, energy savings, smart homes, and smart industries [2]. On the other hand, the rapid expansion of IoT devices has resulted in a rise in cyberattacks aimed at these devices [3]. Attacks utilizing IoT-based botnets are becoming increasingly common and favored by cybercriminals [4]. One of the reasons is the inherent limitations of IoT devices, such as restricted range, power, and memory, as well as the absence of compatible security solutions for these devices and their applications [5]. These limitations in IoT systems have become a focal point in the research domain for developing reliable detection systems that align with the architecture of IoT devices [6].

Recently, Intrusion Detection Systems (IDS) have become crucial components in network security infrastructure to ensure strong protection against cyberattacks [7]. The primary challenge in developing IDS for IoT networks is managing high-dimensional data, as the vast amounts of data generated can lead to increased storage usage, longer detection times, and reduced IDS efficiency [8]. In this context, feature engineering has proven to be a valuable solution for overcoming the limitations of IoT devices, particularly when managing high-dimensional data [9]. One feature engineering model that can be used to address high-

Received: 16-06-2024 | Accepted: 23-11-2024 | Published Online: 28-12-2024

dimensional data issues is the dimensionality reduction method [10].

The effectiveness of machine learning algorithms relies on various factors, such as the quality of the dataset and the choice of optimal feature [11]. This research contributes to maximizing the performance of machine learning algorithms by considering the use of feature quantities through dimensionality reduction using the Self-Organizing Feature Map (SOFM) method. Subsequently, performance evaluation is reviewed based on False Negative Rate (FNR), execution time, accuracy, False Positive Rate (FPR), sensitivity, and specificity.

The authors have organized this paper into six sections. Section 2 examines related work on data dimensionality reduction in IDS, as investigated in prior research. Section 3 explains the datasets used in the experiment, SOFM, classification algorithms, performance evaluation, experiment setup, and analysis tools. Section 4 explains the experimental results and discusses their implications, while Section 5 delves into the conclusions drawn and outlines further research directions concerning IDS.

The implications of this study's findings are significant for both the academic and industrial sectors. First, the demonstrated effectiveness of SOFM in feature engineering for botnet detection suggests a promising approach for improving IoT security. The high accuracy rate indicates that SOFM can reliably identify threats, making it a viable solution for real-world applications. Furthermore, the considerable reduction in memory usage highlights the method's practicality for deployment in resource-constrained IoT environments. This opens the door for broader adoption of advanced security measures in IoT devices, even those with limited computational resources. Industries relying on IoT technology can implement this method to enhance their security infrastructure without the need for substantial hardware upgrades.

To address the issue of high-dimensional data in IoT botnet detection, many researchers have conducted feature engineering. Bahsi et al. [12] and Alqahtani et al. [13] employed the Fisher score method to tackle the high-dimensional data problem in the N-BaIoT dataset, with both studies achieving high accuracy rates. Additionally, Alshamkhany et al. [14] applied Principal Component Analysis (PCA) to tackle the high-dimensional data challenge in the UNSW-NB15 dataset, showcasing significant efficacy in detecting IoT botnets.

Furthermore, Pokhrel et al. [15] evaluated the effectiveness of the chi-square method for feature engineering in addressing the high-dimensional data challenge within the BoT-IoT dataset. The experiment successfully addressed the problem and achieved high accuracy rates. Susanto et al. [16]–[18] used methods such as Fast Independent Component Analysis (Fast-ICA), random projection, and Linear Discriminant Analysis (LDA) to resolve high-dimensional data issues. Their results showed high effectiveness in detecting IoT botnets using the N-BaIoT dataset.

Deris et al. [19] used an autoencoder to address high-dimensional data issues in the medBIoT dataset, achieving very high accuracy in IoT botnet detection. Nomm and Bahsi [20] utilized feature engineering to detect IoT botnets using methods such as Entropy, Variance, and Hopkins. Their experiment, conducted on the N-BaIoT dataset, resolved high-dimensional data problems with relatively high accuracy.

Moreover, Duan et al. [21] employed autoencoder neural networks for feature engineering to resolve high-dimensional data problems in IoT botnet detection. Using the Information Security Centre of Excellence (ISCX)-botnet dataset, their experiment demonstrated high accuracy rates. Haq and Khan [22] utilized PCA to reduce high-dimensional data in the N-BaIoT dataset, and their feature engineering results showed reasonable accuracy in detecting IoT botnets. A summary of feature engineering research addressing high-dimensional data issues in IoT botnet detection is presented in Table 1.

Table 1. Summary of Research on Feature Engineering in IoT Botnet Detection

Authors, Year	Feature Engineering Methods	Classification Methods
[12], 2018	Fisher score	Decision tree nd k- Nearest Neighbor (k-NN),
[13], 2020	Fisher score	Extreme gradient boosting + Genetica algorithm
[14], 2020	PCA	Decision tree, Support Vector Machine, k-NN, Naïve bayes
[15], 2021	Chi-square	k-NN, Multi-Layer perception, Gausian naïve bayes,
[16], 2023	Fast ICA	Random Forest, k-NN, Decision tree, Gradient Boosting, Adaboost,
[17], 2021	Random Projection	Gradient Boosting, Random Forest, k-NN, Decision tree, Adaboost,
[18], 2024	LDA	Gradient Boosting, Random Forest, k-NN, Decision tree, Adaboost,
[20], 2019	Entropy, Variance, Hopkins	Support vector machine, Isolation forests
[19], 2023	Autoencoder	Artificial Neural Networks (ANN)
[21], 2022	Autoencoder Neural Network	Decision tree, Gradient boosting,
[22], 2022	PCA	Deep Neural Network
[12], 2018	Fisher score	k-NN, Decision tree

2. Research Methods

This research is an experimental study aimed at developing and testing a botnet detection system for resource-constrained IoT devices by combining Self-Organizing Feature Map (SOFM) techniques for feature engineering with various machine learning algorithms.

This experimental study involves hypothesis testing using a series of controlled experiments to evaluate the effectiveness of the proposed method.

2.1 System Workflow

The proposed workflow of the IoT security system is illustrated in Figure 1. The IoT botnet detection system uses SOFM for feature engineering, followed by a multi-classifier classification process. In this process, dimensionality reduction is performed, reducing the data from 100 features to 10 features. The dimensions or features used for reduction can be selected randomly, as stated by [23], [24]. The final step involves validation using k-Fold Cross Validation. This technique ensures that the experimental error closely approximates the actual prediction error on the IDS [25]. It is also used to detect overfitting issues [26].

The experiment conducted in this research adopts existing methods and combines them into a new model. Based on our literature review, the proposed model has not been previously explored by other researchers.

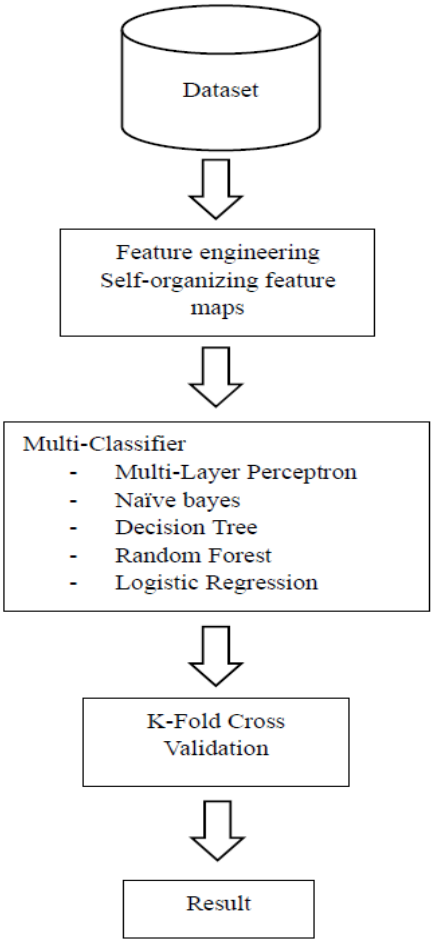

Figure 1. Proposed experimental workflow

2.2 Dataset

This study uses the MedBIoT dataset [27]. The dataset extraction process employs the incremental statistics method [28], with a total of 100 features, indicating that the dataset has a high-dimensional nature. Furthermore, the dataset is generated from three physical devices and 80 virtual devices. It comprises four types of traffic: benign traffic and attack traffic (bashlite, torii, and mirai), totaling 17.845.567 traffic data points. The MedBIoT dataset has high dimensionality, necessitating feature engineering to optimize the detection process. This is evident from several studies. For instance, Kalakoti et al. [29] employed feature engineering methods such as Pearson's correlation-based technique, Fisher score, Analysis of Variance (ANOVA) F-test, mutual information, recursive feature elimination, sequential forward selection, and sequential backward selection to reduce the dimensionality of the MedBIoT data. Deris et al. [19] used Autoencoder for feature engineering on the MedBIoT dataset. Additionally, Manzanares et al. [30] applied PCA to reduce the dimensionality of the MedBIoT data. Furthermore, Malik et al. [31] utilized feature selection methods such as Univariate Filter, Multivariate Filter, Forward Feature Selection, and Backward Feature Elimination for feature engineering on the MedBIoT data.

This study utilizes approximately 15.29% of the total traffic data. The distribution of the MedBIoT dataset is presented in Table 2.

Table 2. MedBIoT Dataset Distribution

New Label	Traffic Type	Amount of data	The amount of power used
Attack	BashLite	4143276	841843
	Mirai	842674	733856
	Torii	319139	64755
Benign	Benign	12540478	1087832

2.3. Self-Organizing Feature Map

The Self-Organizing Feature Map (SOFM) is a type of a type of Artificial Neural Network (ANN) introduced by [32] and subsequently reviewed in studies [33]–[35]. Self-Organizing Maps (SOFMs) are a type of neural network architecture where processing units are organized on a grid with n dimensions [36], which can be used for dimensionality reduction [37]. Essentially, SOFM is employed to recognize the spatial distribution of a dataset using a network of neurons. This outlines the steps involved in training a basic version of a neural network with a grid-based architecture [38].

Network Initialization: The initial network can be structured in a hexagonal or rectangular shape, characterized by longitudinal and latitudinal lines. The training input vector for SOFM is represented as Equation 1.

$$x = [x_i]_{n\ x\ 2^{i=1\ldots n}} \quad (1)$$

Each user event has a location represented by a vector denoted by x_i. Here, i-th indicates the specific user event (i = 1 to N), and N represents the total number of user events. The Self-Organizing Map (SOFM) is initialized as a $p\ x\ q$ grid, where each node (also called a neuron) has a weight vector. These weight vectors are

initially distributed uniformly within the minimum bounding box of the city map. The neuron vectors (or weight vectors) are denoted by Equation 2.

$$W = \left[W_{ij}\right]_{p\ x\ q\ x\ 2^{i=1,\ldots\ldots,p,j=1,\ldots\ldots,q}} \quad (2)$$

Identifying the Winning Neuron: During training, the algorithm finds the most suitable neuron for a given input data point. This neuron, known as the Best Matching Unit (BMU) or Best Matching Cell [34], has the weight vector closest to the input vector based on a distance metric like Euclidean distance

Adjusting Neuron Weights: During each training step, the algorithm: Selects an input: It picks a data point (denoted by x_k)from the training data. Identifies the winning neuron (BMU): It finds the neuron with the weight vector $(w_{(k)})$ closest to x_k, typically using a distance metric like Euclidean distance. Updates neighboring neurons: The weights of neurons within a specific neighborhood surrounding the BMU are adjusted to become more similar to the input data point (x_k). The size of this neighborhood typically decreases over time during training. The process of updating a neuron can be formulated as Equation 3.

$$\forall w_i \in \aleph\,(w_k)\; w_i^{(t+1)} = w_i^t - \alpha_{i,k}^t\left(x_k^{(t)} - w_i^t\right) \quad (3)$$

$\varkappa\,(w_k)$ is a neighboring set of neurons from $w_{(k)}$ and $\alpha_{i,k}^t$ is a scalar value ranging between 0 and 1. The research in [35] suggests a feasible choice for $\alpha_{i,k}^t$, which is Equation 4.

$$a_{i,k}^t = c(t)\exp\left\{\frac{dist^2\,(w_i,w_k)}{2\sigma^t(t)}\right\} \quad (4)$$

$dist\,(w_i.w_k)$ represents the Euclidean distance between w_i and $w_{(k)}$, $c(t)$ and $\sigma(t)$ are two monotonically decreasing functions of t, and the initial value of $\sigma(t)$ is sufficiently large.

In this study, the use of SOFM was chosen because [39]: It can be applied to non-normally distributed data; It can reveal non-linear relationships between variables; It is relatively robust against missing data; The output visualization produces a projection of high-dimensional data in a two-dimensional space while maintaining the topological structure of the input data, thus mapping similar samples together; SOFM provides an easy way to identify potential outliers. Outliers can be isolated in a single node, indicating a large distance from neighboring nodes, or scattered across the map.

Moreover, SOFM has better capabilities compared to other feature engineering methods, such as:

SOFM features allow for the extraction of meaningful information from large and complex data. In comparison, when analyzing large datasets using techniques like PCA, the graphical representation of the results must be optimized using density scatter plots to enable visual exploration [40].

SOFM has the advantage of visualization features and the ability to maintain data topology and handle noisy and missing data, compared to PCA [41].

T-distributed stochastic Neighbor Embedding (T-SNE) can be computationally expensive for large datasets, whereas SOFM can be more scalable with appropriate optimizations [42].

Methods like T-SNE and clustering are unsupervised, but SOFM provides a structured representation of the clusters on a 2D map, offering more interpretability [43].

2.4 Classification algorithm

Machine learning is emerging as a powerful tool for effectively identifying cyberattacks on diverse platforms Machine learning is emerging as a powerful tool for effectively identifying cyberattacks on diverse platforms [44]. Kiran et al. [45] have utilized machine learning in developing an IDS model to detect attacks on IoT networks. Additionally, Nugroho et al. [46] state that addressing vulnerabilities in IDS involves employing machine learning techniques. This serves as a reference for researchers in developing IDS for IoT, particularly for botnet detection.

In this experiment, several machine learning algorithms are employed, namely Multi-Layer Perceptron (MLP), Random Forest, Naïve Bayes, Logistic Regression (LR), and Decision Tree (DT). Different algorithms are used to ensure a comprehensive evaluation [47]. Additionally, different algorithms may perform differently on various datasets and tasks. Comparing them can help identify the best-performing algorithm [48]. Some algorithms may be more robust to noise or different data distributions [49].

MLP was chosen because it can capture complex non-linear relationships in data due to its neural network architecture and is widely used in many applications, providing a solid foundation for comparison [50]. Random Forest was used in this experiment because it can combine multiple decision trees to enhance accuracy and control overfitting. It also performs well on various datasets with different characteristics [51]. Naïve Bayes was selected for this research analysis because it is often used as a baseline due to its simplicity and effectiveness in many classification problems [52]. Additionally, Logistic regression was used in this study because it can provide clear insights into the relationships between features and the target variable [53]. Finally, Decision Tree was employed in this experiment because it is easy to understand and interpret, as decisions are based on simple if-then rules [54].

2.5 Performance Evaluation

Table 3 illustrates how a confusion matrix can be used to assess the effectiveness of feature engineering within a machine-learning model [55].

Table 3. Confusion Matrix

		Predicted	
		Possitive (P)	Negative (N)
Actual	True (T)	TP	FP
	False (F)	FN	TN

Based on Table 3, in detecting IoT botnet, the process is described as follows:

True Positive (TP): In machine learning classification for binary problems (positive vs negative), TP refers to the number of actual positive data points that were correctly predicted as positive. So instead of "benign," it should be "positive" for a general classification task.

False Positive (FP): It represents the number of actual negative data points that were incorrectly predicted as positive (attack).

False Negative (FN): This refers to the number of actual positive data points (attacks) that were incorrectly predicted as negative (benign).

True Negative (TN): This indicates the number of actual negative data points that were correctly predicted as negative (benign).

Thus, the classification matrix calculation is obtained to evaluate the performance of using SOFM in machine learning, including Equations 5-9.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (6)$$

$$Specificity = \frac{TN}{TN+FP} \quad (7)$$

$$False\ Possitive\ Rate = \frac{FP}{TN+FP} \quad (8)$$

$$False\ Negative\ Rate\ \frac{FN}{FN+TP} \quad (9)$$

The Receiver Operating Characteristic (ROC) curve is a visual tool that plots the model's sensitivity (True Positive Rate) on the y-axis against the False Positive Rate (FPR) on the x-axis. In an ideal scenario, the ROC curve would reach the top-left corner, indicating perfect classification with 100% sensitivity and 0% FPR.

2.6 Analysis Tools

The simulation in this research was conducted using a notebook with the following specifications: Intel Core i7 9th gen processor, memory Solid-State Drive (SSD) 512GB, Video Graphics Array (VGA) NVIDIA Giga Texel Shader eXtreme (GTX) 1660 Ti Graphic Processing Unit (GPU), and Random Access Memory (RAM) 16GB. The operating system used was Windows 11, and Python was utilized for the analysis.

3. Results and Discussions

3.1 Results

This research evaluates the performance of five machine learning classification algorithms through experimentation. Performance measurement was conducted using six evaluation matrices, namely confusion matrix, accuracy, sensitivity, FPR, specificity, and FNR. Additionally, execution time was also measured during the botnet detection process.

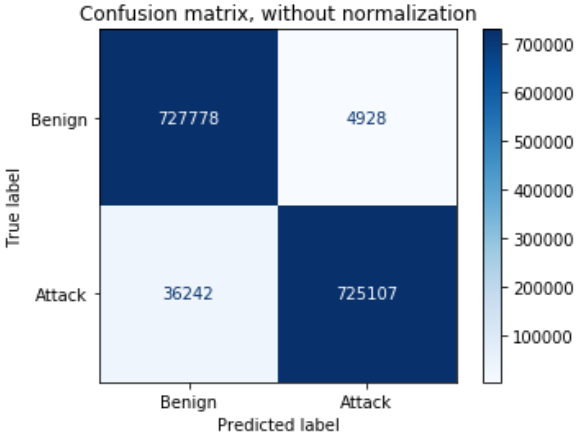

Figure 2. Confusion matrix Decision tree

Figure 2 shows the results confusion matrix to evaluate the performance of the Decision Tree classification model derived from the SOFM method for botnet detection. This matrix summarizes the number of correctly and incorrectly classified benign and malicious traffic events. The experiment results indicate that the utilization of feature engineering with SOFM and Decision Tree classification correctly classified 1,458,885 data points, accounting for 97.24%, and misclassified 41,170 data points, representing 2.76%.

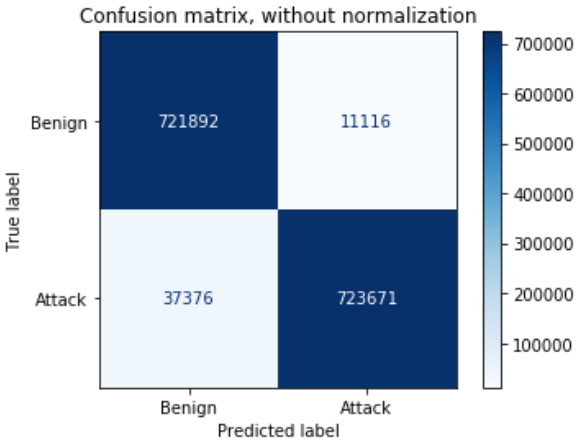

Figure 3. Confusion matrix Logistic regression

Meanwhile, Figure 3 shows the results confusionshows the results confusion matrix to evaluate the performance of the Logistic regression classification model derived from the SOFM method for botnet detection. This matrix summarizes the number of correctly and incorrectly classified benign and malicious traffic events. The experiment results reveal that the application of feature engineering with SOFM and Logistic Regression classification correctly classified 1,445,563 data points, amounting to 96.75%, and misclassified 48,492 data points, representing 3.25%.

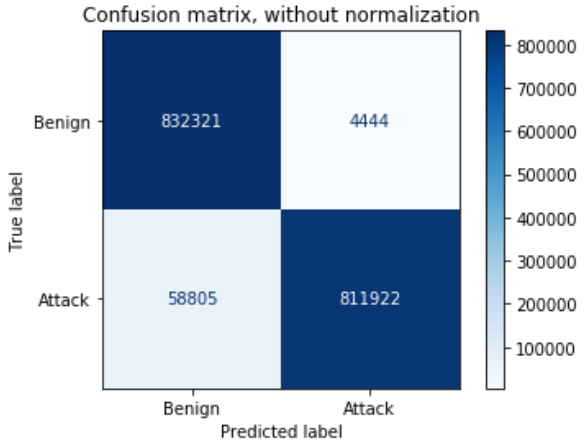

Figure 4. Confusion matrix MLP

Figure 4 shows the results confusionshows the results confusion matrix to evaluate the performance of the MLP classification model derived from the SOFM method for botnet detection. This matrix summarizes the number of correctly and incorrectly classified benign and malicious traffic events. The experiment results indicate that the utilization of feature engineering with SOFM and MLP classification correctly classified 1,644,243 data points, accounting for 96.30%, and misclassified 63,249 data points, representing 3.70%.

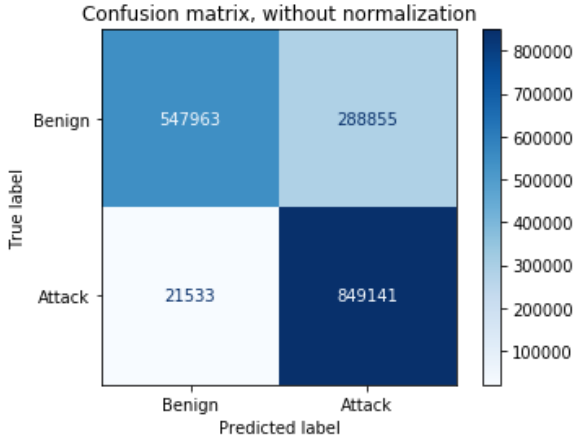

Figure 5. Confusion matrix naïve bayes

Figure 5 shows the results confusionshows the results confusion matrix to evaluate the performance of the Naïve bayes classification model derived from the SOFM method for botnet detection. This matrix summarizes the number of correctly and incorrectly classified benign and malicious traffic events. The experiment results indicate that the utilization of feature engineering with SOFM and Naïve Bayes classification correctly classified 1,397,104 data points, accounting for 81.82%, and misclassified 310,388 data points, representing 18.18%.

Figure 6 shows the results confusionshows the results confusion matrix to evaluate the performance of the Random Forest classification model derived from the SOFM method for botnet detection. This matrix summarizes the number of correctly and incorrectly classified benign and malicious traffic events. The experiment results indicate that the utilization of feature engineering with SOFM and Random Forest classification correctly classified 164,975 data points, accounting for 96.63%, and misclassified 57,617 data points, representing 3.37%.

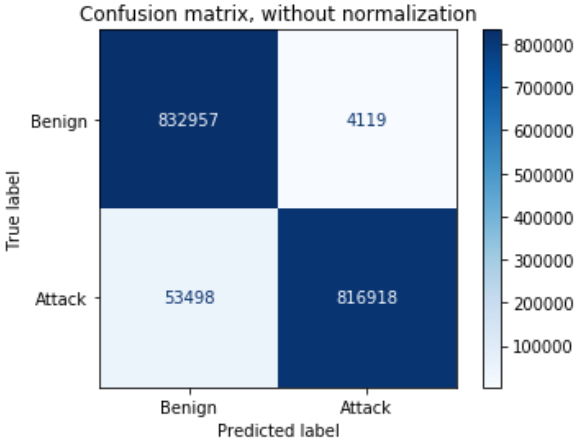

Figure 6. Confusion matrix random forest

Based on the experiment results presented through the confusion matrices from Figure 2 to Figure 6, the accuracy, sensitivity, specificity, FPR, and FNR values are calculated and presented in Table 4. The experimental results of applying feature engineering using SOFM in the IoT botnet classification process show that Decision Tree has the highest accuracy rate, reaching 97.24%. Furthermore, Naïve Bayes has the highest sensitivity rate with a value of 0.9753. Additionally, MLP has the highest specificity value with a value of 0.9947. On the other hand, the lowest FPR and FNR values are owned by Random Forest and Naïve Bayes, with values of 0.0049 and 0.0247, respectively. Naive Bayes stands out as the most efficient algorithm for botnet detection in this experiment, completing the task in just 74.64 seconds. Despite Naïve Bayes having the fastest detection time, its accuracy rate still significantly lags behind Decision Tree.

Table 4. Comparison of Evaluation Results

Classification	Accuracy	Sensitivity	Specificity	FPR	FNR	Execution time (s)
Decision tree	97.24%	0.9523	0.9932	0.0067	0.0476	100.66
MLP	96.30%	0.9325	0.9947	0.0053	0.0675	308.76
Naïve bayes	81.82%	0.9753	0.6548	0.3452	0.0247	74.64
Random forest	96.63%	0.9385	0.9951	0.0049	0.0615	613.63
LR	96.75%	0.9509	0.9848	0.0151	0.0491	101.52

The next performance evaluation process is based on the ROC curve. According to Hogan and Adams [56], ROC curves offer a valuable tool for evaluating classification models across different classification

thresholds. They visualize the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) as the threshold is adjusted. The Area Under the Curve (AUC) summarizes this performance by capturing the total area beneath the ROC curve. A higher AUC translates to a stronger ability of the model to distinguish between classes. In ideal scenarios, an AUC of 1 signifies perfect classification, while an AUC of 0.5 represents random guessing. For instance, a high AUC in attack traffic detection indicates the model's proficiency in separating normal traffic from attacks.

Figure 7 show results the performance of the Decision Tree classification model derived from the SOFM method for botnet detection by analyzing its ROC AUC value. This metric summarizes the model's ability to differentiate between malicious and benign traffic events. The experiment results indicate that the utilization of feature engineering with SOFM and Decision Tree classification can effectively differentiate between attack and normal traffic, with AUC values of 1 for both.

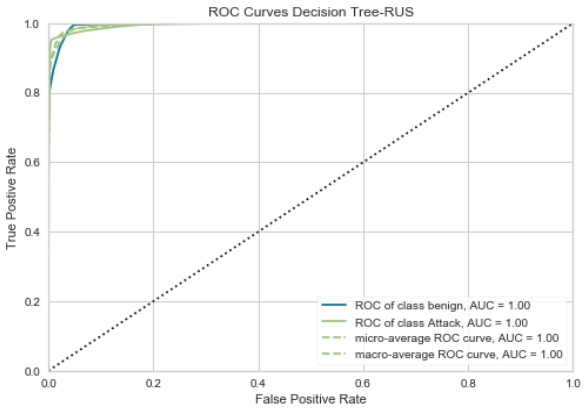

Figure 7. ROC AUC Decision tree

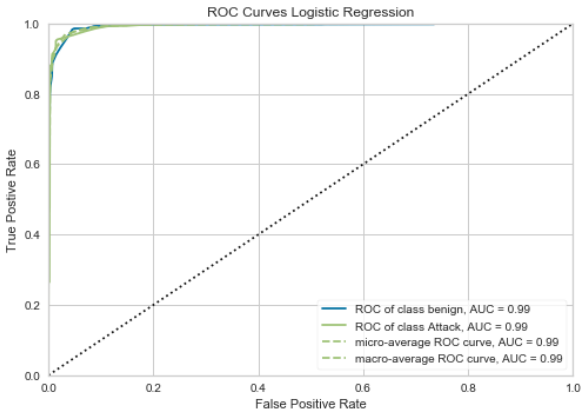

Figure 8. ROC AUC Logistic regression

Figure 8 depicts the ROC AUC values for botnet detection using the Logistic Regression classification from the SOFM method. The experiment results reveal that the utilization of feature engineering with SOFM and Logistic Regression classification effectively distinguishes between attack and normal traffic, with AUC values of 0.99 for both.

Figure 9 illustrates the ROC AUC values for botnet detection using the MLP classification from the SOFM method. The experiment results show that the utilization of feature engineering with SOFM and MLP classification effectively distinguishes between attack and normal traffic, with AUC values of 1 for both.

Figure 10 displays the ROC AUC values for botnet detection using the Naïve Bayes classification from the SOFM method. The experiment results indicate that the utilization of feature engineering with SOFM and Naïve Bayes classification effectively distinguishes between attack and normal traffic, with AUC values of 0.98 for attack traffic and 0.85 for normal traffic.

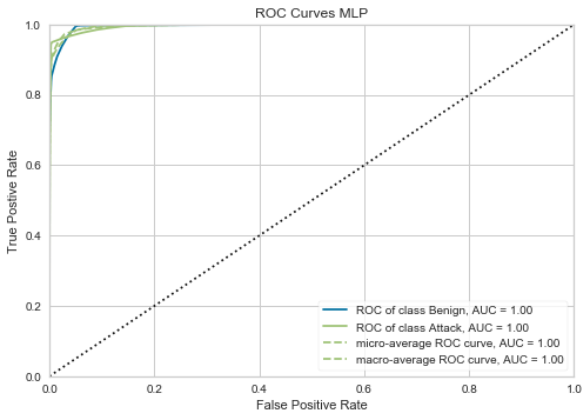

Figure 9. ROC AUC MLP

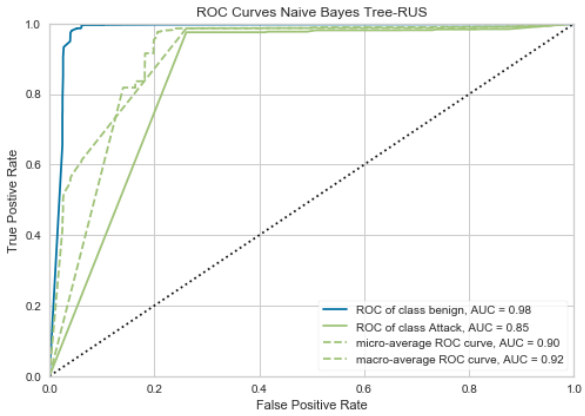

Figure 10. ROC AUC Naïve bayes

Figure 11 presents the ROC AUC values for botnet detection using the Random Forest classification from the SOFM method. The experiment results demonstrate that the utilization of feature engineering with SOFM and Random Forest classification effectively distinguishes between attack and normal traffic, with AUC values of 0.99 for both.

Table 4 presents the validation results of the IoT botnet detection process using feature engineering with SOFM and machine learning. Upon comparison with the accuracy results in Table 5 and the validation results with k-fold cross-validation, it is evident that the

differences are not significant. According to statements by Li et al. [57] and Vabalas et al. [58], if the accuracy results do not significantly differ from those of k-fold cross-validation, it can be concluded that there is no overfitting.

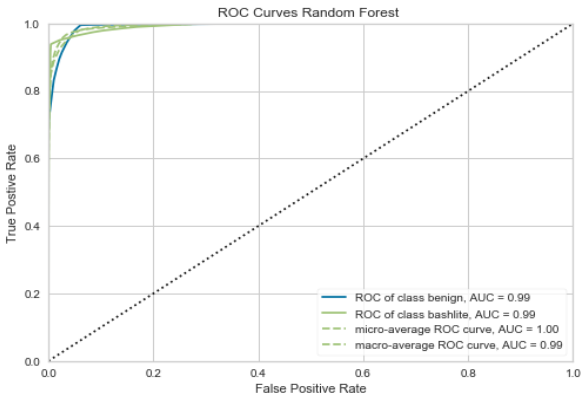

Figure 11. ROC AUC Random forest

Table 5. Classification Validation Results Using Cross Validation

Iteration	DT	MLP	Naïve bayes	Random forest	LR
1	97.24	96.48	81.68	97.19	96.83
2	97.29	96.42	80.64	97.22	96.76
3	97.29	96.38	81.82	97.20	96.78
4	97.21	96.43	81.99	97.25	96.79
5	96.24	96.47	81.77	97.20	96.75
6	97.23	96.43	81.97	97.16	96.82
7	97.27	96.44	81.68	97.22	96.76
8	97.28	96.43	81.84	97.22	96.76
9	97.15	96.41	80.66	97.23	96.67
10	97.24	96.43	81.77	97.15	96.72
Average	97.14	96.43	81.58	97.20	96.76

Furthermore, the dimensionality reduction process using feature engineering can reduce the dataset dimensions from the original 100 columns to 10 columns. As a result, the size of the initial dataset, which was 3.08 GB, was reduced to 923 MB after undergoing the feature engineering process with dimensionality reduction. This clearly demonstrates that employing feature engineering can effectively decrease memory usage during the botnet detection process.

3.2 Comparison with previous research

Table 6 shows the comparison results of the proposed methods with previous research (state of the art).

Table 6. Comparison with previous research in IoT Botnet Detection

Authors, Year	Feature Engineering Methods	Classification Methods	Accuracy
[14], 2020	PCA	k-NN	90.40
		Naïve Bayes	96.40
[15], 2021	Chi-square	k-NN	99.60
		Multi-Layer perception	87.40
		Gausian naïve bayes	99.40
[22], 2022	PCA	Deep Neural Network	91.44
[59], 2022	Recursive feature elimination	XGBoost	94.00
[60], 2023	Information Gain	Recurrent Neural Network	97.00
[61], 2024	Feature significance	k-NN	78.00
This work	SOFM	Decision tree	97.24

3.3 Discussions

Based on the experiment results presented, several observations can be made regarding the effectiveness of feature engineering with SOFM and machine learning algorithms in IoT botnet detection.

Firstly, the accuracy results obtained from the experiments, as shown in Table 4, demonstrate promising performance across all classification algorithms, with Decision Tree achieving the highest accuracy of 97.24%. This indicates the efficacy of Decision Tree in accurately classifying benign and attack instances.

Moreover, the sensitivity values, which represent the ability to correctly identify attack instances, are notably high across all algorithms. Naïve Bayes, in particular, exhibited the highest sensitivity of 0.9753, indicating its proficiency in minimizing false negatives and ensuring fewer attacks go undetected.

Additionally, the specificity values, indicating the ability to accurately classify benign instances, were also commendable. MLP recorded the highest specificity of 0.9947, indicating its capability to reduce false positives and minimize misclassifications of benign data as attacks.

Based on the evaluation results provided in Table 4, it can be argued that the decision tree algorithm offers the best overall performance compared to the other four algorithms (MLP, Naïve Bayes, Random Forest, and Logistic Regression) for the given features. The decision tree has the highest accuracy, indicating that it correctly classifies the most instances. While Naïve Bayes has the highest sensitivity, the decision tree still performs very well, demonstrating its effectiveness in identifying positive instances. The decision tree has a high specificity, meaning it correctly identifies negative instances, although Random Forest and MLP have slightly higher specificity. The decision tree has a low FPR, indicating fewer incorrect positive classifications, though Random Forest and MLP have slightly lower FPR.

Furthermore, the decision tree has a low FNR, meaning fewer missed positive instances. Only Naïve Bayes has a lower FNR, but its overall accuracy and specificity are much worse. The decision tree has a reasonable execution time, much lower than MLP and Random Forest, and slightly lower than Logistic Regression, making it efficient. The decision tree provides the best balance of high accuracy, good sensitivity, high

specificity, low FPR, low FNR, and reasonable execution time. While other algorithms might excel in one or two metrics, the decision tree consistently performs well across all evaluation criteria, making it the best choice for the given features.

Furthermore, the ROC AUC values, as illustrated in Figures 7 to 11, consistently demonstrated excellent performance in distinguishing between attack and normal traffic for all classification algorithms, with AUC values predominantly close to 1. This indicates the robustness of the models in effectively discriminating between the two classes.

In addition, the validation results presented in Table V, when compared with the accuracy results from Table IV, exhibit only minor differences, suggesting that the experiment outcomes do not indicate overfitting. This finding aligns with the conclusions drawn by Li et al. [57] and Vabalas et al. [58], indicating that the model can generalize well to unseen data.

Lastly, the dimensionality reduction process through feature engineering effectively reduced the dataset's size from 3.08 GB to 923 MB, demonstrating the practical benefits of feature engineering in reducing memory usage during the botnet detection process.

Feature engineering is a critical step in the data preprocessing phase that significantly impacts the performance of machine learning models. The methods compared with previous research (state of the art) in this discussion include Principal Component Analysis (PCA), Chi-square, Recursive feature elimination, Information Gain, Feature significance, and Self-Organizing Feature Maps (SOFM).

The PCA was utilized in multiple studies. PCA combined with k-NN by Alshamkhany et al. [14] combined PCA with k-NN to achieve 90.40% accuracy, whereas with Naïve Bayes, it reached 96.40%. Another study by Haq and Khan [22] used PCA with a Deep Neural Network (DNN) to achieve 91.44% accuracy. Furthermore, Recursive feature elimination, used by Alissa et al. [59], yielded high accuracy with a Extreme Gradient Boosting **(**XGBoost) (94.00%).

Chi-square, as reported by Pokhrel et al. [15], achieved a remarkable 99.60% accuracy when paired with k-NN. Other combinations included Multi-Layer Perception (MLP) with 87.40% and Gaussian Naïve Bayes with 99.40%. In the research by Bojarajulu et al. [60] Information Gain was paired with a Recurrent Neural Network, showing the result accuracy achieved 97.00%. On the other hand, research by Sharma et al. using the Feature significance method produced an accuracy level of 78.00%. Meanwhile, the proposed method achieved a high detection accuracy of 97.24%.

The experiments highlight the effectiveness of feature engineering with SOFM and machine learning algorithms in IoT botnet detection, showcasing high accuracy, sensitivity, specificity, and robustness in distinguishing between attack and normal traffic while mitigating the risk of overfitting and reducing memory usage.

4. Conclusions

The IoT botnet detection system leverages a combination of feature engineering with the SOFM method and machine learning. Experiment results indicate that the application of SOFM can reduce the dataset size to a relatively smaller scale compared to the original data, thus conserving memory usage. Moreover, the utilization of the Decision Tree classification proves to be quite effective in detecting IoT botnets compared to Logistic Regression, Naïve Bayes, MLP, and Random Forest. Through the Decision Tree algorithm, the classification results yield an accuracy rate of 97.24%, sensitivity of 0.9523, specificity of 0.9932, FPR of 0.0067, and FNR of 0.0476%, with an execution time of 100.66 s. For future research, the authors intend to explore the utilization of artificial intelligence techniques to autonomously identify, detect, and mitigate various forms of cyber attacks on IoT networ.

Acknowledgements

To ensure transparency, the authors affirm no conflicting interests in this research. The study was independently funded by the researchers and did not involve grants from public, commercial, or non-profit funding agencies.

References

[1] S. Nižetić, P. Šolić, D. López-de-Ipiña González-de-Artaza, and L. Patrono, “Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future,” *J. Clean. Prod.*, vol. 274, 2020, doi: 10.1016/j.jclepro.2020.122877.

[2] S. Kumar, P. Tiwari, and M. Zymbler, “Internet of Things is a revolutionary approach for future technology enhancement: a review,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0268-2.

[3] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, “Machine learning-based IoT-botnet attack detection with sequential architecture,” *Sensors (Switzerland)*, vol. 20, no. 16, pp. 1–15, 2020, doi: 10.3390/s20164372.

[4] I. Ali *et al.*, “Systematic Literature Review on IoT-Based Botnet Attack,” *IEEE Access*, vol. 8, pp. 212220–212232, 2020, doi: 10.1109/ACCESS.2020.3039985.

[5] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, “A survey on security in internet of things with a focus on the impact of emerging technologies,” *Internet of Things (Netherlands)*, vol. 19, p. 100564, 2022, doi: 10.1016/j.iot.2022.100564.

[6] G. Eric and A. Jurcut, “Intrusion Detection in Internet of Things Systems : A Review on Design Approaches Leveraging Multi-Access Edge,” *Sensors*, vol. 22, pp. 1–33, 2022.

[7] A. Binbusayyis and T. Vaiyapuri, “Comprehensive analysis and recommendation of feature evaluation measures for intrusion detection,” *Heliyon*, vol. 6, no. 7, p. e04262, 2020, doi: 10.1016/j.heliyon.2020.e04262.

[8] A. Adnan, A. Muhammed, A. A. A. Ghani, A. Abdullah, and F. Hakim, “An intrusion detection system for the internet of things based on machine learning: Review and challenges,” *Symmetry (Basel).*, vol. 13, no. 6, pp. 1–13, 2021, doi:

10.3390/sym13061011.
[9] Z. Azam, M. M. Islam, and M. N. Huda, "Comparative Analysis of Intrusion Detection Systems and Machine Learning-Based Model Analysis Through Decision Tree," *IEEE Access*, vol. 11, no. July, pp. 80348–80391, 2023, doi: 10.1109/ACCESS.2023.3296444.
[10] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, "A Review of Dimensionality Reduction Techniques for Efficient Computation," in *Procedia Computer Science*, 2019, vol. 165, pp. 104–111.
[11] M. Farhan and M. G., "Efficient Botnet Detection using Feature Ranking and Hyperparameter Tuning," *Int. J. Comput. Appl.*, vol. 182, no. 48, pp. 55–60, 2019, doi: 10.5120/ijca2019918739.
[12] H. Bahsi, S. Nomm, and F. B. La Torre, "Dimensionality Reduction for Machine Learning Based IoT Botnet Detection," in *Proc. 2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV*, 2018, pp. 1857–1862.
[13] M. Alqahtani, H. Mathkour, and M. M. Ben Ismail, "IoT botnet attack detection based on optimized extreme gradient boosting and feature selection," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–21, 2020, doi: 10.3390/s20216336.
[14] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, "Botnet Attack Detection using Machine Learning," in *Proc. 14th International Conference on Innovations in Information Technology, IIT*, 2020, no. November, pp. 203–208.
[15] S. Pokhrel, R. Abbas, and B. Aryal, "IoT Security: Botnet detection in IoT using Machine learning," *arXiv*, pp. 1–11, 2021.
[16] Susanto *et al*., "Dimensional Reduction With Fast ICA for IoT Botnet Detection," *J. Appl. Secur. Res.*, vol. 18, no. 4, pp. 665–688, 2023, doi: 10.1080/19361610.2022.2079906.
[17] Susanto, D. Stiawan, M. A. S. Arifin, J. Rejito, M. Y. Idris, and R. Budiarto, "A Dimensionality Reduction Approach for Machine Learning Based IoT Botnet Detection," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2021–Octob, no. October, pp. 26–30, 2021, doi: 10.23919/EECSI53397.2021.9624299.
[18] Susanto, D. Stiawan, M. Agus Syamsul Arifin, M. Y. Idris, and R. Budiarto, "Effective and efficient approach in IoT Botnet detection," *Sinergi*, vol. 28, no. 1, pp. 31–42, 2024, doi: 10.22441/sinergi.2024.1.004.
[19] D. Stiawan, Susanto, A. Bimantara, M. Y. Idris, and R. Budiarto, "IoT botnet attack detection using deep autoencoder and artificial neural networks," *KSII Trans. Internet Inf. Syst.*, vol. 17, no. 5, pp. 1310–1338, 2023, doi: 10.3837/tiis.2023.05.001.
[20] S. Nomm and H. Bahsi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," in *Proc.- 17th IEEE International Conference on Machine Learning and Applications, ICMLA*, 2019, pp. 1048–1053.
[21] L. Duan, J. Zhou, Y. Wu, and W. Xu, "A novel and highly efficient botnet detection algorithm based on network traffic analysis of smart systems," *Int. J. Distrib. Sens. Networks*, vol. 18, no. 3, 2022, doi: 10.1177/15501477211049910.
[22] M. A. Haq and M. A. R. Khan, "Dnnbot: Deep neural network-based botnet detection and classification," *Comput. Mater. Contin.*, vol. 71, no. 1, pp. 1729–1750, 2022, doi: 10.32604/cmc.2022.020938.
[23] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, "A Review of Dimensionality Reduction Techniques for Efficient Computation," *Procedia Comput. Sci.*, vol. 165, pp. 104–111, 2019, doi: 10.1016/j.procs.2020.01.079.
[24] L. H. Nguyen and S. Holmes, "Ten quick tips for effective dimensionality reduction," *PLoS Comput. Biol.*, vol. 15, no. 6, pp. 1–19, 2019, doi: 10.1371/journal.pcbi.1006907.
[25] A. McCarthy, E. Ghadafi, P. Andriotis, and P. Legg, "Functionality-Preserving Adversarial Machine Learning for Robust Classification in Cybersecurity and Intrusion Detection Domains: A Survey," *J. Cybersecurity Priv.*, vol. 2, no. 1, pp. 154–190, 2022, doi: 10.3390/jcp2010010.
[26] H. Shafique, A. A. Shah, M. A. Qureshi, and M. K. Ehsan, "Machine Learning Empowered Efficient Intrusion Detection Framework," *VFAST Trans. Softw. Eng.*, vol. 10, no. 2, pp. 27–35, 2022.
[27] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," *ICISSP 2020 - Proc. 6th Int. Conf. Inf. Syst. Secur. Priv.*, pp. 207–218, 2020, doi: 10.5220/0009187802070218.
[28] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *arXiv*, no. February, pp. 18–21, 2018.
[29] R. Kalakoti, S. Nomm, and H. Bahsi, "In-Depth Feature Selection for the Statistical Machine Learning-Based Botnet Detection in IoT Networks," *IEEE Access*, vol. 10, no. July, pp. 94518–94535, 2022, doi: 10.1109/ACCESS.2022.3204001.
[30] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Using MedBIoT Dataset to Build Effective Machine Learning-Based IoT Botnet Detection Systems," *Commun. Comput. Inf. Sci.*, vol. 1545 CCIS, pp. 222–243, 2022, doi: 10.1007/978-3-030-94900-6_11.
[31] K. Malik, F. Rehman, T. Maqsood, S. Mustafa, O. Khalid, and A. Akhunzada, "Lightweight Internet of Things Botnet Detection Using One-Class Classification," *Sensors*, vol. 22, no. 10, pp. 1–17, 2022, doi: 10.3390/s22103646.
[32] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982, doi: 10.1007/BF00337288.
[33] J. A. Kangas, T. Kohonen, and J. T. Laaksonen, "Variants of Self-Organizing Maps," *IEEE Trans. Neural Netw.*, vol. I, no. I, pp. 93–99, 1990, doi: 10.1007/978-3-642-97966-8_5.
[34] T. Kohonen, "The Self-Organizing Map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990, doi: 10.1109/5.58325.
[35] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013, doi: 10.1016/j.neunet.2012.09.018.
[36] D. Miljkovic, "Brief review of self-organizing maps," *2017 40th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2017 - Proc.*, no. May, pp. 1061–1066, 2017, doi: 10.23919/MIPRO.2017.7973581.
[37] A. Saraswati, V. T. Nguyen, M. Hagenbuchner, and A. C. Tsoi, "High-resolution Self-Organizing Maps for advanced visualization and dimension reduction," *Neural Networks*, vol. 105, pp. 166–184, 2018, doi: 10.1016/j.neunet.2018.04.011.
[38] X. Chen, M. Simsek, and B. Kantarci, "Locally reconfigurable Self Organizing Feature Map for high impact malicious tasks submission in Mobile Crowdsensing," *Internet of Things*, no. January, pp. 1–14, 2020.
[39] S. Licen, A. Astel, and S. Tsakovski, "Self-organizing map algorithm for assessing spatial and temporal patterns of pollutants in environmental compartments: A review," *Sci. Total Environ.*, vol. 878, no. March, p. 163084, 2023, doi: 10.1016/j.scitotenv.2023.163084.
[40] S. Licen, S. Cozzutto, and P. Barbieri, "Assessment and comparison of multi-annual size profiles of particulate matter monitored at an urban-industrial site by an optical particle counter with a chemometric approach," *Aerosol Air Qual. Res.*, vol. 20, no. 4, pp. 800–809, 2020, doi: 10.4209/aaqr.2019.08.0414.
[41] J. Xiao, L. Wang, N. Chai, T. Liu, Z. Jin, and J. Rinklebe, "Groundwater hydrochemistry, source identification and pollution assessment in intensive industrial areas, eastern Chinese loess plateau," *Environ. Pollut.*, vol. 278, 2021, doi: 10.1016/j.envpol.2021.116930.
[42] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nat. Commun.*, vol. 10, no. 1, pp. 1–12, 2019, doi: 10.1038/s41467-019-13055-y.
[43] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Som-Vae: Interpretable discrete representation learning on time series," *7th Int. Conf. Learn. Represent. ICLR 2019*, pp. 1–18, 2019.
[44] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and

F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, pp. 1–29, Oct. 2020.

[45] K. V. V. N. L. Sai Kiran, R. N. K. Devisetty, N. P. Kalyan, K. Mukundini, and R. Karthi, "Building a Intrusion Detection System for IoT Environment using Machine Learning Techniques," *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 2372–2379, 2020, doi: 10.1016/j.procs.2020.04.257.

[46] E. P. Nugroho, T. Djatna, I. S. Sitanggang, A. Buono, and I. Hermadi, "A Review of Intrusion Detection System in IoT with Machine Learning Approach: Current and Future Research," *2020 6th Int. Conf. Sci. Inf. Technol. Embrac. Ind. 4.0 Towar. Innov. Disaster Manag. ICSITech 2020*, pp. 138–143, 2020, doi: 10.1109/ICSITech49800.2020.9392075.

[47] A. Geron, *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. 2019.

[48] M. A. Lones, "How to avoid machine learning pitfalls: a guide for academic researchers," *arXiv*, pp. 1–28, 2021.

[49] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13–17–Augu, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.

[50] F. Chollet, *Deep learning with Python*. 2017.

[51] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: The lasso and generalizations*. 2015.

[52] V. M. S Raschka, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*, vol. 69, no. 4. 2019.

[53] L. L. Kupper, D. W. Hosmer, and S. Lemeshow, *Applied Logistic Regression.*, vol. 85, no. 411. 2013.

[54] G. James;, D. Witten;, T. Hastie;, R. Tibshirani;, and J. Taylor, *An Introduction to Statistical Learning*, vol. 102. 2023.

[55] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informatics*, vol. 17, no. 1, pp. 168–192, 2021, doi: 10.1016/j.aci.2018.08.003.

[56] J. Hogan and N. M. Adams, "On Averaging ROC Curves," *Trans. Mach. Learn. Res.*, pp. 1–12, 2023.

[57] H. Li, G. K. Rajbahadur, D. Lin, C.-P. Bezemer, Z. Ming, and Jiang, "Keeping Deep Learning Models in Check: A History-Based Approach to Mitigate Overfitting," *arXiv*, 2024.

[58] A. Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, "Machine learning algorithm validation with a limited sample size," *PLoS One*, vol. 14, no. 11, pp. 1–20, 2019, doi: 10.1371/journal.pone.0224365.

[59] K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, and S. Sakib, "Botnet Attack Detection in IoT Using Machine Learning," *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/4515642.

[60] B. Bojarajulu, S. Tanwar, and T. P. Singh, "Intelligent IoT-BOTNET attack detection model with optimized hybrid classification model," *Comput. Secur.*, vol. 126, p. 103064, 2023, doi: https://doi.org/10.1016/j.cose.2022.103064.

[61] R. Sharma, S. M. ud din, N. Sharma, and A. Kumar, "Enhancing IoT Botnet Detection through Machine Learning-based Feature Selection and Ensemble Models," *EAI Endorsed Trans. Scalable Inf. Syst.*, vol. 11, no. 2, pp. 1–6, 2024, doi: 10.4108/eetsis.3971.

