JCTA Journal of Computing Theories and Applications ISSN:3024-9104

Research Article

IoT Security Using Machine Learning Methods with Features Correlation

Chaw Su Htwe [1,*], Zin Thu Thu Myint [2], and Yee Mon Thant [1]

[1] Cyber Security Research Lab, University of Computer Studies, Yangon, Myanmar; e-mail: chawsuhtwe@ucsy.edu.mm; yeemonthant@ucsy.edu.mm
[2] Faculty of Information Science, University of Computer Studies, Yangon, Myanmar; e-mail: zinthuthumyint@ucsy.edu.mm
* Corresponding Author: Chaw Su Htwe

Abstract: The Internet of Things (IoT) is an innovative technology that makes our environment smarter, with IoT devices as an integral part of home automation. Smart home systems are becoming increasingly popular as an IoT service in the home that connects via a network. Due to the security weakness of many devices, the malware is targeting IoT devices. After being infected with malicious attacks on smart devices, they act like bots that the intruders can control. Machine learning methods can assist in improving the attack detection process for these devices. However, the irrelevant features raise the computation time as well as affect the detection accuracy in the processing with many features. We proposed a machine learning-based IoT security framework using feature correlation. The feature extraction scheme, one-hot feature encoding, correlation feature selection, and attack detection implement an active detection mechanism. The results show that the implemented framework is not only for effective detection but also for lightweight performance. The proposed system outperforms the results with the selected features, which have almost 100% detection accuracy. It is also approved that the proposed system using CART is more suitable in terms of processing time and detection accuracy.

Keywords: Botnet; DDoS; IoT; Feature extraction; Feature selection; Machine learning; Malware.

1. Introduction

Many intrusions have been targeting IoT devices in recent years due to their popularity and challenges, such as limited resources, using default passwords by users, etc. Therefore, developing a good detection system for these devices is necessary. Many studies have been conducted in the past, but the emergence of different types of malwares is not enough to effectively detect different types of attacks. Among the attack detection systems, intruders can circumvent the signatures, but these kinds of methodologies can be found in the public detection systems. Another possible approach is based on the anomaly-based system. The machine learning-based system is also a type of anomaly-based system. Machine learning-based methods can support the system with better results but cannot be embedded into the IoT devices to perform as a host-based system. Therefore, a machine learning-based system can be implemented as a network-based detection system on a middleware device or a cloud server.

Today, mobile and IoT infrastructure are growing, and attackers are targeting these devices, so security issues are more challenging on these devices. Cybercriminals are intensifying their attempts to create malware attacking IoT devices, reached up three times as in 2017. About 121,588 malware samples were collected in 2018 by Kaspersky Lab [1]. The number of malicious challenges increased nearly ten-fold to nearly 249 million malicious attempts in December 2019. Trend Micro recorded almost 194 million brute force logins from the botnet attacks challenge from 2019 to 2020 by capturing the attacks [2]. IoT devices can be attacked once connected to the Internet. Figure 1 shows a general increase in the percentage of organizations experiencing successful cyberattacks that were experienced in the years 2019 through 2023. The data suggests that cyber threats are becoming more sophisticated and frequent.

Received: July, 16th 2024
Revised: August, 12th 2024
Accepted: August, 16th 2024
Published: August, 18th 2024

The consistently high or increasing frequency of successful attacks underscores the necessity for robust cybersecurity strategies[3].

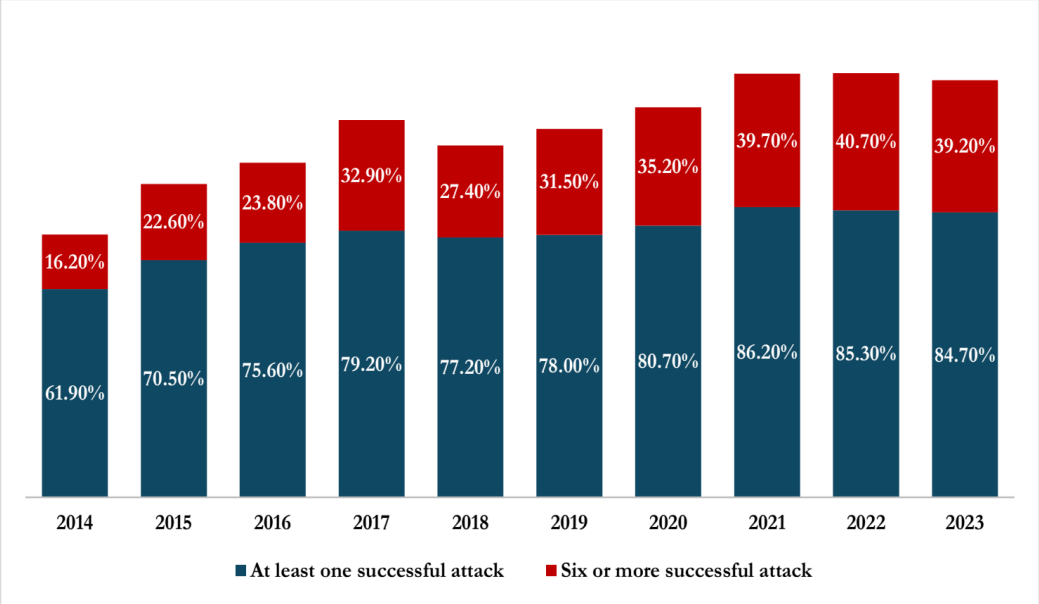

Figure 1. Percentage of successful attacks (2014 – 2023)[3]

Due to the immoderate encounter rates of malware attacks in the ASEAN region, the cyber-security IVO research is implemented to monitor the network, find the vulnerabilities, and improve the detection algorithms[4]. The machine learning approach is a reliable mechanism for malicious attacks on IoT devices because of the capability of the detection attacks' essential differences. Many attack detection works were used in well-known datasets, such as KDDCUP99[5] and KDD-NSL[6], but these are not suitable for attack detection in IoT networks because they have no IoT malware attack patterns. IoT-23 is applied in this study, and it is a state-of-the-art dataset[7] that was captured by using real IoT devices. It includes malicious patterns, different types of attacks, and benign records, capturing the most challenging attacks from the Mirai botnet.

The proposed attack detection framework has 4 phases: feature extraction, feature selection, feature encoding, and machine-learning based attack detection. The feature extraction mechanism, which is shown in Algorithm1, is proposed for performing real-time attack detection capability. Many public datasets, such as KDD99, NSL-KDD, and BoT-IoT, can be available for attack detection. However, KDD99 and NSL-KDD datasets are outdated and less relevant for IoT attack detection. BoT-IoT dataset is a modern dataset, but its records are based on the simulation of IoT data. The IoT-23 dataset is also modern, providing the data in a realistic environment with real IoT devices. Therefore, the IoT-23 dataset is applied to attack building for the machine learning-based attack detection model. It is suitable for modern attack detection, but the challenge is the size and complexity of its dataset. It contains large volumes of data with extensive network traffic captures. There are also many features included in the original dataset. This will require a high resource demand and may reduce the detection accuracy. The correlation measurement can evaluate the features' dependency on the target variable, and Chi-Squared can evaluate the correlation of categorial features[8]. However, the IoT-23 dataset includes continuous variables. Therefore, a feature selection mechanism called the Pearson correlation method, is applied to reduce unimportant features in the system. It will support the system in achieving maximum detection performance, simplify data input, and produce more accurate results. A one-hot encoding mechanism is used to smooth machine learning classification using the nominal features in the original dataset. The CART algorithm is applied in the proposed system as an attack detection. Generally, it is a popular machine learning method to get high detection rates for the cyber-attack protection system. Naive Bayes is also applied to compare with the CART results, and it is the most straightforward and fast classification algorithm.

The hypothesis of the proposed system is focused on the efficacy of feature selection in improving the performance of machine learning algorithms for detecting cyber-attacks in IoT environments. Using the Person correlation-based feature selection method will reduce the number of useful features for attack detection. In addition, it will reduce the computational complexity of performing machine learning processes.

The experiment results indicate that the detection results with selected features are improved, especially in terms of the processing time and detection rate in Naïve Bayes[9]. Besides, the results with the CART algorithm outperformed the result of Naïve Bayes. Therefore, it can be observed that the CART algorithm is suitable for the proposed system, and the unnecessary features may cause a high processing demand without supporting high detection accuracy.

The contributions of the proposed system are:

1. The feature extraction mechanism was capable of real-time attack detection in an IoT environment.
2. The feature selection process is faster processing and better detection accuracy by reducing the number of features.
3. The implementation of the CART algorithm on better performance in both the detection accuracy and the response time.

The paper is organized into five sections as follows. The current challenges of cyber-attacks and detection systems are introduced in Section 1. The related work of the cyber-attack detection approaches and the methodologies are discussed in Sections 2 and 3. Section 4 presents the proposed system and the experimental results, and the paper concludes in Section 5.

2. Literature Review

Many attack detection proposals were widely based on well-known network datasets, like KDDCUP99[10], and its development (KDD-NSL). The various machine learning algorithms were used to analyze the NSL-KDD dataset in [11] using the classification algorithm and are available in the WEKA tool. But, these datasets are for conventional networks and not the IoT environment. So, some researchers have used it for recent attack detection based on the modern dataset, Bot-IoT [12]. Deployed using the random forest regressor algorithm to extract new features from the dataset.

The proposals [13], [14] were implemented with a signature-based detection system, showing that formal snort rules are insufficient for the detection system. Their proposals were focused on a conventional network; the signature-based system cannot detect unknown attacks because these approaches are based on pre-defined attack signatures.

The system [15] proposed the cut-off value of correlation to select some of the best features for the Intrusion Detection System. The Pearson correlation of the feature selection method is used to reduce the features of the NSL-KDD dataset and implement classification methods, namely SVM, KNN, and RF. Their results showed that selected features slightly reduced the computation time, but the detection accuracy also degraded and was not as good as using all original features with these three algorithms.

The application of machine learning to intrusion detection in IoT environments has seen significant advancements, moving beyond traditional datasets like KDDCUP99 and NSL-KDD to more relevant datasets such as Bot-IoT and IoT-23. The work of N. Abdal-Gawad et al. [16] illustrates the use of deep learning models on the IoT-23 dataset to identify and mitigate complex IoT-based attacks. Their research employed Adversarial Autoencoders (AAE) and Bidirectional Generative Adversarial Networks (BiGAN) to achieve high detection accuracy, demonstrating the potential of advanced neural networks in handling IoT traffic anomalies. Similarly, N. Saini et al. [17] utilized the UNSW-NB15 dataset to develop a hybrid ensemble learning model, combining random forest and XGBoost for enhanced intrusion detection, reporting improvements in detection accuracy compared to conventional approaches.

A significant trend in recent literature is the integration of explainable AI (XAI) to enhance the interpretability of intrusion detection systems. For instance, M. Keshk et al. [18] applied SHAP (SHapley Additive exPlanations) values to a machine learning-based detection system using the TON_IoT dataset. This approach provided insights into feature importance and decision-making processes, thus addressing the "black box" nature of traditional machine

learning models. Furthermore, Baahmed et al.[19] leveraged XAI techniques to interpret the decisions made by graph neural networks (GNNs), which were trained on the IoT dataset, offering a transparent and robust framework for IoT security.

Another emerging focus is optimizing feature selection methods for improving detection efficiency. Y. N. Kunang et al. [20] introduced a novel autoencoder-based feature selection technique applied to the BoT-IoT dataset, effectively reducing the feature set while maintaining high detection performance. Moreover, K. Ren et al. [21] developed a reinforcement learning-based feature selection framework, tested on the CSE-CIC-IDS2018 dataset, to select relevant features during training. Their approach led to a reduction in computational overhead and an improvement in the detection capabilities. These advancements highlight the ongoing efforts to adapt intrusion detection methodologies to IoT networks' specific challenges, enhancing accuracy and efficiency.

3. Methodology

The proposed malware attack detection system has two main parts, shown in Figure 2.

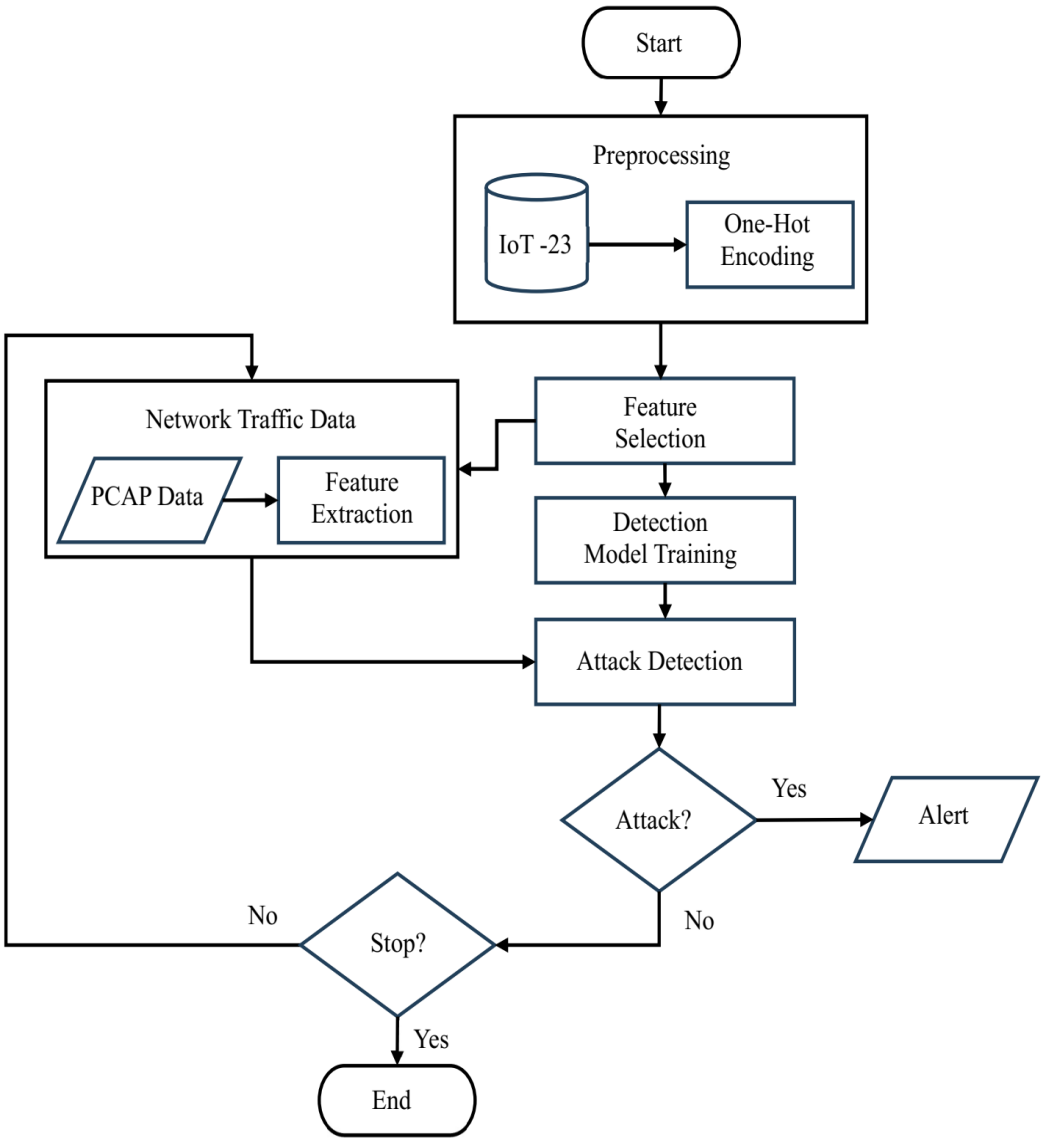

Figure 2. The proposed attack detection model

The first is the model training part, and the second is the attack detection part. The dataset is loaded in the model training phase to process one-hot encoding to prepare categorical data for numerical records. After that, the Pearson correlation algorithm selects the feature. Once the important features are selected, the attack detection model is trained using the CART algorithm. Moreover, it informs the selected features of the network traffic detection mechanism to perform the attack detection

mechanism effectively. In the attack detection phase, raw network traffic data is captured to get the pcap files. After getting the raw data, feature extraction processing is performed, and a few features are extracted by matching them with the category of the selected features. Then, attack detection is performed using the CART algorithm. If the attack is detected, the system will generate an alert. If any malicious pattern is not detected, the system will continue the attack detection is performed as the background process.

3.1. Feature Extraction

There are five phases in the proposed feature extraction mechanism (see Algorithm 1). The first one is for capturing the network traffic data using TShark[22] to get the raw data packets from the network and prepare them into packet data. Another phase is packet data manipulation to extract basic features using Python libraries, such as Scapy and Pandas. The next phase is for the Python script preparing for automation of the feature-preparing process. The next step is to extract the sophisticated features by CICFlowMeter [23] and Zeek [24]. Then, the feature aggregation process is done on the features set, and the feature cleaning mechanism is done by handling missing values and One-Hot encoding. It is readily the feature for attack detection with a machine learning algorithm. The final process is necessary to match the final extracted features with the selected features by the feature selection process.

Algorithm 1. Feature Extraction

```
INPUT: pcap_data, which TShark captures
OUTPUT: features_df
1:  Initialize Data Structures
        a. packets ← ReadPcap(pcap_data)
        b. features ← ∅
2:  Extract Basic Features
        for each packet in packets, do
            time ← ExtractTime(packet)
            src_ip ← ExtractSrcIP(packet)
            dst_ip ← ExtractDstIP(packet)
            src_port ← ExtractSrcPort(packet)
            dst_port ← ExtractDstPort(packet)
            protocol ← ExtractProtocol(packet)
            length ← ExtractLength(packet)
            feature ← {time, src_ip, dst_ip, src_port, dst_port, protocol, length}
            features ← features ∪ {feature}
        end for
3:  Convert Extracted Features to DataFrame
        features_df ← ConvertToDataFrame(features)
4:  Advanced Feature Extraction
        if UseCICFlowMeter then
            cic_df ← ExtractUsingCICFlowMeter(pcap_data)
        end if
        if UseZeek then
            zeek_df ← ExtractUsingZeek(pcap_data)
        end if
5:  Aggregate and Preprocess Features
        if cic_df exists then
            features_df ← Merge(features_df, cic_df)
        end if
        if zeek_df exists then
            features_df ← Merge(features_df, zeek_df)
        end if
        features_df ← HandleMissingValues(features_df)
        features_df ← OneHotEncode(features_df)
6:  Return Final DataFrame
        return features_df
```

There are 21 features, including the class label, extracted from network traffic data collected from the original dataset. These are shown in Table 1.

Table 1. Original features from the dataset

No.	Features	Description
1	fields-ts	Flow start time
2	uid	Unique ID
3	id.orig-h	Source IP address
4	id.orig-p	Source port
5	id.resp-h	Destination IP address
6	id.resp-p	Destination port
7	proto	Transaction protocol
8	service	Http, ftp, smtp, ssh, dns, etc.
9	duration	Record total duration
10	orig-bytes	Source to destination transaction bytes
11	resp-bytes	Destination to source transaction bytes
12	conn-state	Connection state
13	local-orig	Source local address
14	local-resp	Destination local address
15	missed-bytes	Missing bytes during transaction
16	history-orig-pkts	History of source packets
17	orig-ip-bytes	Flow of source bytes
18	resp-pkts	Destination packets
19	resp-ip-bytes	Flow of destination bytes
20	tunnel-parents	Traffic tunnel
21	label	Attacks or benign

3.2. One-Hot Encoding

One-hot encoding is a crucial data preprocessing technique in machine learning, particularly when dealing with categorical data. This method transforms categorical variables into a binary (0 or 1) vector representation, allowing algorithms to process categorical data more effectively [25]. It converts categorical data into a binary matrix with a unique vector of zeros and ones representing each category. This technique ensures that categorical variables are transformed into a format suitable for machine learning algorithms that require numerical input. Each unique category in a categorical variable is assigned a new binary column. If a categorical feature has k distinct values, one-hot encoding converts it into k binary columns. For each instance, the column corresponding to the categorical value is set to 1, and all other columns are set to 0.

3.3. Feature Selection

The Pearson correlation algorithm selects the most relevant features from the dataset. It is a well-known correlation-based algorithm, and its similarity measures between variables or features of a dataset. Pearson correlation, commonly used in coefficient (ϱ), which means the Linear correlation coefficient, is a measurement of two random variables' dependence[26]. Its linear correlation is higher, and the correlation coefficient's absolute value is stronger if the relationship between the two features is closer to some linear function. It is based on the equation (1), which is for the coefficient ϱ, on the values x_i on variables X and the values y_i on the variables Y.

$$\rho \frac{cov(X,Y)}{\sqrt{\sigma^2(X)\sigma^2(Y)}} \tag{1}$$

Therefore, this measurement can investigate the redundancies for which feature strongly correlates to some features. There are 21 features extracted from network packet data. The

Pearson correlation method calculates the features ranking and arranges their values by descending order. The threshold value higher than 0.5 decides the most important feature set. After selecting the most important features, there are only six features designated. The selected features are shown in Table 2.

Table 2. The selected features from the dataset

No.	Features	Description
1	id.resp_p	Destination port
2	orig_bytes	Source to destination transaction bytes
3	history_org_pkts	History of source packets
4	id.orig_h	Source IP address
5	resp_ip_bytes	The flow of destination bytes
6	resp_bytes	Destination to source transaction bytes

3.4. CART

It is a classification and regression tree algorithm for building prediction models from data. The data is partitioned recursively, and a simple prediction model is fitted within each partition to obtain the detection model. This partitioning can be presented graphically as a decision tree. These trees are planned for subordinate factors that take a predetermined number of unordered qualities, with expectation mistake estimated regarding misclassification cost, and furthermore for subordinate factors that review nonstop or requested discrete qualities, with forecast blunder normally estimated by the squared distinction between the noticed and anticipated qualities [27]. CART uses a generalization of the binomial variance called the Gini index which is shown in equation (2). It supplies the sum of the squared probabilities of each class, and i is from 1 to the possible number of classes.

$$Gini = 1 - \sum (Pi)^2 \tag{2}$$

This calculation can handle the missing values. It will be ignored in the analysis if the dependent variable of a case is missing. It will also be ignored if all predictor variables of a case are missing. The case is disregarded if the case weight is missing, zero, or negative. The case is disregarded if the recurrence weight is missing, zero, or negative.

3.5. Naïve Bayes(NB)

The NB is famous for its simple principle and widely used supervised algorithm for predictive modeling. Its classifier belongs to categorize traffic as normal or abnormal for attack detection, using the Bayesian theorem. It requires strong independent assumptions between the features. The main process of Naïve Bayes is based on equation (3). $P(A|B)$ represents the posterior probability, $P(B|A)$ represents the likelihood, $P(A)$ represents the marginal likelihood and $P(B)$ represents the prior probability. It has many attributes make NB user-friendly, and the training process is fast and easy.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

4. Experimental Results

The dataset applied to the proposed system is called IoT-23 [28]. Stratosphere Laboratory, CTU University, and Avast develop it. The dataset contains IoT devices' labeled network traffic captures (PCAP files), including benign and malicious traffic. Traffic is categorized into attacks and normal traffic behaviors. It includes traffic from real IoT devices under different conditions, such as normal operation and DDoS attacks, captured by the Mirai attack.

4.1. Performance Evaluation

CART is a learning algorithm that is supervised for both classification and regression. It also can be used for predictive modeling problems. Python libraries are used to implement

this algorithm in the system, especially sci-kit-learn. The selected datasets are initially uploaded to a learning program implemented in the Python Programming language. There are five folds categorizations with attack and benign data for the performance evaluation. Among them, 66% of the dataset is used for system training, and the remaining one-third is used for testing. Both classification and feature selection are implemented in Python. The evaluation results are computed by equation (4), based on the confusion matrix.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

In the notations of the above equation, Acc represents the accuracy measurement for the attack classification, TP represents the number of attack classes correctly classified, TN denotes the number of benign classes correctly classified, FP signifies the number of attack classes incorrectly classified, and FN is for the number of the benign class is incorrectly classified.

The distribution attack and benign data in each fold are shown in Figure 3. In IoT attack datasets, benign records comprise only a small percentage of the data. This is because normal traffic is relatively rare compared to the substantial volume of traffic generated during attack conditions. In the experiments, no over-sampling or under-sampling methods were applied. This decision ensures that the results are derived from the original data, providing a more realistic evaluation of the dataset's inherent characteristics.

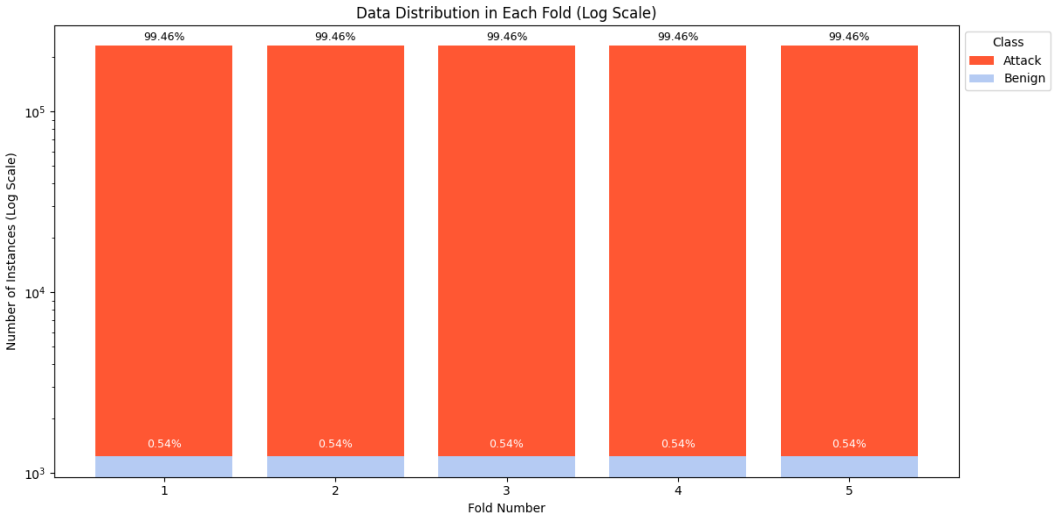

Figure 3. The distribution of attack and benign records

4.2. Result Discussion

4.2.1 Attack Detection Accuracy

The attack detection accuracy by CART is shown in Table 3. It compares the performance metrics for both of using all features versus selected features across five folds. Although there are measurement criteria, the accuracy, precision, true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) are considered for evaluation of the attack detection mechanism because these are more comprehensive and suitable for scenarios where the balance between different types of errors is critical. They provide a detailed understanding of the model's performance, especially in attack detection, where the consequences of misclassification can be significant. TPR focuses on capturing actual positive cases and can also refer to recall or sensitivity. Precision is extremely high for all features, with selected features showing perfect precision in all folds. Thus, many classification and prediction scenarios evaluate precision, recall, and accuracy [29], [30]. These measurements are important for evaluation in attack detection. The higher precision indicates a lower false positive rate, and recall is also important because missing an attack will result in severe consequences. According to the results, the proposed model achieves near-perfect accuracy with all features, but the selected features consistently result in perfect accuracy. The TPR is perfect for both all and selected features across all folds. TNR is slightly lower for all

features compared to selected features, which consistently achieve a perfect TNR. The FPR is very low for all features but is completely eliminated with selected features. The FNR is zero for both all features and selected features. Using selected features results in slightly better performance across all metrics, with perfect scores in every fold, indicating that feature selection improves the model's overall effectiveness.

Table 3. The attack detection accuracy by CART

Fold	Features	Accuracy	Precision	TPR	TNR	FPR	FNR
1	All features	99.99	99.92	100.00	99.99	< 0.001	0.000
	Selected features	100.00	100.00	100.00	100.00	0.000	0.000
2	All features	99.99	99.99	100.00	99.92	0.001	0.000
	Selected features	100.00	100.00	100.00	100.00	0.000	0.000
3	All features	99.99	99.99	100.00	99.92	0.001	0.000
	Selected features	100.00	100.00	100.00	100.00	0.000	0.000
4	All features	99.99	99.99	100.00	99.92	0.001	0.000
	Selected features	100.00	100.00	100.00	100.00	0.000	0.000
5	All features	99.99	99.99	100.00	99.92	0.001	0.000
	Selected features	100.00	100.00	100.00	100.00	0.000	0.000

The attack detection accuracy by Naïve Bayes is shown in Table 4. It evaluates the Naive Bayes's performance using all features versus selected features across five folds. It shows high accuracy with both all and selected features, with a slight decrease for selected features in some folds. Precision is very high for both feature sets, but there's a notable drop in Fold 1 with selected features. The TPR is consistently high for both all and selected features, though it drops slightly for selected features in some folds. TNR is perfect or near perfect for both feature sets. The FPR is extremely low or zero for both feature sets. The FNR is low, with slight increases for selected features in some folds.

Table 4. The attack detection accuracy by Naïve Bayes (NB)

Fold	Features	Accuracy	Precision	TPR	TNR	FPR	FNR
1	All features	99.99	99.84	99.45	99.99	< 0.001	0.543
	Selected features	99.89	84.05	99.37	99.90	0.104	0.638
2	All features	99.99	100.00	99.99	100.00	0.000	< 0.001
	Selected features	99.92	100.00	99.92	100.00	0.000	0.076
3	All features	100.00	100.00	100.00	100.00	0.000	0.000
	Selected features	99.93	100.00	99.93	100.00	0.000	0.070
4	All features	99.99	100.00	99.99	100.00	0.000	< 0.001
	Selected features	99.88	100.00	99.88	100.00	0.000	0.124
5	All features	99.99	100.00	99.99	100.00	0.000	< 0.001
	Selected features	99.86	100.00	99.86	100.00	0.000	0.140

Tables 3 and 4 show high accuracy, but the selected features with CART achieve perfect accuracy across all folds, while Table 4 shows a slight decrease in some folds with selected features. CART shows perfect precision with selected features in all folds, while Naive Bayes shows a drop in Fold 1 with selected features. The detection accuracy with CART achieves a perfect TPR with selected features, whereas the accuracy with Naive Bayes has minor decreases in some folds. Both results show high TNR, but the CART achieves perfection with selected features, while Naive Bayes shows minor variations. Both classifiers maintain very low or zero FPR, with the CART achieving zero across all folds with selected features. The CART shows zero FNR with selected features, while the Naive Bayes shows slight increases in FNR for selected features in some folds.

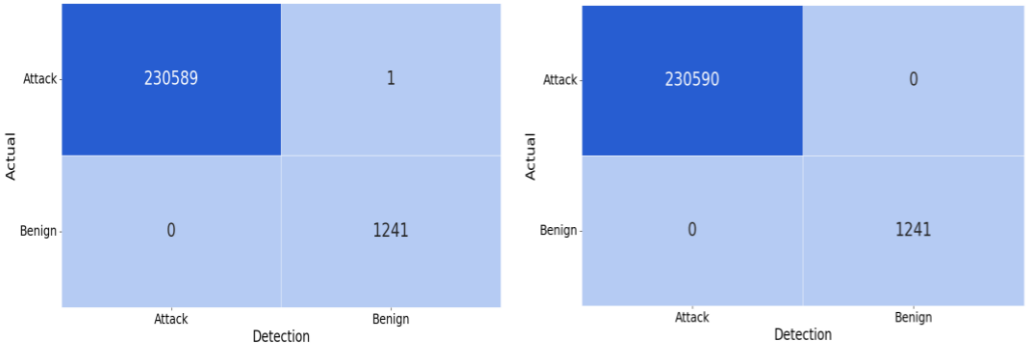

Figure 4. Confusion matrix for attack detection with CART (**a**) All features; (**b**) Selected features.

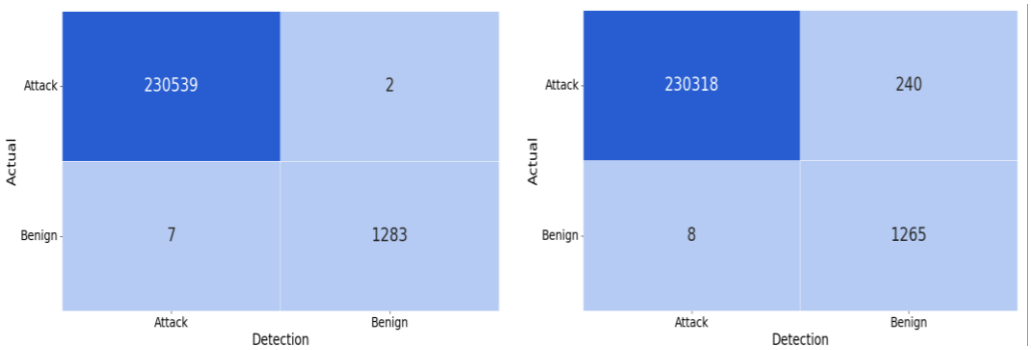

Figure 5. Confusion matrix for attack detection with Naïve Bayes(**a**) All features; (**b**) Selected features.

The detection accuracy with the confusion matrix by CART is shown in Figure 4. The accuracy of using all original features is shown in Figure 4 (a), and the result of applying the selected features is shown in Figure 4 (b). The result comparisons are also presented with Figure 5, a classification by Naïve Bayes, including using all features in Figure 5 (a) and selected features in Figure 5 (b). The detection accuracy with CART showed slightly better overall performance with selected features, achieving perfect scores across all metrics and folds. However, the performance with Naive Bayes, while still showing high performance, indicates minor drops in accuracy, precision, TPR, and increases in FNR for selected features in some folds. The results with Naive Bayes suggest that while feature selection improves computational efficiency, it may lead to a slight decrease in some performance metrics compared to the previous results. On the other hand, the performance with CART is not only better in detection accuracy but also improves computational efficiency.

4.2.2 Procession Time Comparison

Table 5 shows CART's training and testing times across five different folds, comparing the performance when using all features versus selected features. In fold 1, the training time is 3.3 seconds, while the testing time is 0.06 seconds using all features. When using selected features, the training time is significantly reduced to 0.16 seconds, with a testing time of 0.02 seconds. In fold 2, the training time is 0.6 seconds, and the testing time is 0.03 seconds while using all features. When applying only the selected features, the training time drops to 0.09 seconds, and the testing time is 0.02 seconds. In fold 3, the training time is 0.71 seconds, with a testing time of 0.04 seconds for all features. When using the selected features, the training time is reduced to 0.11 seconds, with the testing time at 0.02 seconds. In fold 4, the training time is 0.9 seconds, and the testing time is 0.06 seconds if all features are applied. When using only the selected features, the training time decreases to 0.07 seconds, and the testing time is 0.02 seconds. In the final fold, the training time is 1.09 seconds, with a testing time of 0.05 seconds for all features. When applying the selected features, the training time reduces to 0.09

seconds, and the testing time is 0.02 seconds. Across all folds, the training and testing times are consistently lower when using selected features compared to using all features. The reduction in training time is particularly notable, suggesting that feature selection significantly improves the efficiency of the training process without compromising the testing performance.

Table 5. Processing Time Comparison by CART

Fold	Features	Training (seconds)	Testing (seconds)
1	All features	3.3	0.06
	Selected features	0.16	0.02
2	All features	0.6	0.03
	Selected features	0.09	0.02
3	All features	0.71	0.04
	Selected features	0.11	0.02
4	All features	0.9	0.06
	Selected features	0.07	0.02
5	All features	1.09	0.05
	Selected features	0.09	0.02

Table 6 also compares training and testing times for Naive Bayes's across five folds, considering all features and selected features. For most folds, using selected features significantly reduces both the training and testing times compared to using all features. This indicates that feature selection improves the efficiency of the training process and results in faster testing. As an all comparison, training time is longer with all features, and the testing time also benefits from reduction. Overall, feature selection appears to be advantageous for improving computational efficiency.

Table 6. Processing Time Comparison by Naïve Bayes

Fold	Features	Training (seconds)	Testing (seconds)
1	All features	3.78	0.2
	Selected features	0.67	0.07
2	All features	2.44	2.02
	Selected features	0.64	0.07
3	All features	2.37	0.24
	Selected features	1.58	0.12
4	All features	2.91	0.23
	Selected features	1.09	0.11
5	All features	2.88	0.27
	Selected features	0.65	0.1

CART consistently shows faster training and testing times compared to Naive Bayes, especially when using selected features. Feature Selection greatly improves the efficiency of both algorithms, but the impact is more pronounced in CART. CART appears to be the better choice for computational efficiency based on the provided data.

4.2.3 Results Comparison

A comparison of the results is also done by the related research, which uses the same dataset. Y. Liang [31] focused on the attack detection model using machine learning methods to detect the attacks. They evaluated the performance by combining the dataset with several attacks. The study [32] proposed a machine learning model for anomaly detection in IoT networks. They evaluated the performance with machine learning algorithms, including Naïve Bayes and a decision tree algorithm (Random Forest).

Table 7. Results Comparison

Model	Decision Tree			Naïve Bayes		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Y. Liang et al. [31]	0.73	0.77	0.73	0.30	0.85	0.30
N. A. Stoian [32]	1.00	1.00	1.00	0.23	0.76	0.23
Ours	1.00	1.00	1.00	0.99	0.97	0.99

Table 7 shows the results of comparing the proposed system with other related systems. The results in the study [31] are conducted by Random Forest (a decision tree algorithm), and Naïve Bayes. Decision Tree and Naïve Bayes conduct the results in the study [32]. The proposed model is conducted by CART (a decision tree), and Naïve Bayes. The results of the proposed system are based on the average values of the tested in each fold with selected features, which are shown in Tables 3 and 4. According to the overall results in these studies, the decision tree-based model consistently performs better, particularly in the proposed system, because it achieves perfect results with selected features. The study [32] also provides perfect accuracy, but these are conducted using the Random Forest algorithm. Basically, CART is computationally simpler and faster than Random Forest. Besides, it is more suitable for fewer features. In the Naïve Bayes model, the proposed system provides higher accuracy than other studies. Therefore, the proposed system enhances the effectiveness of the other models.

5. Conclusions

More attacks focus on IoT devices because of their computational resources for deploying formal protection systems. The proposed system is for detecting attacks with a lightweight framework. Thus, the proposed system adopted the Pearson correlation feature selection method for reducing irrelevant features. The correlation coefficient was investigated between the features on the IoT-23 dataset. For real-time attack detection purposes, the feature-extracting mechanism is implemented. In addition, the CART algorithm is applied for building the attack detection model and compared with the performance of Naïve Bayes. The experiment results indicated that CART could detect the attacks with 100% accuracy, even reducing many features. Naïve Bayes performance also achieved up to 99.93% with the selected features while reducing the number of features from 21 to 6. The experiment results indicated that the feature selection process can provide the best accuracy by using CART and acceptable by using Naïve Bayes algorithms. Overall, the CART system provides higher accuracy and faster detection response than the others. In future work, the feature sets can be investigated to classify the other attack classes more appropriately.

Author Contributions: Conceptualization: C.S.H. and Z.T.T.M.; methodology, C.S.H.; software: C.S.H.; validation: C.S.H. and Y.M.T..; formal analysis: C.S.H.; investigation: Z.T.T.M.; writing—original draft preparation: C.S.H.; writing—review and editing: Z.T.T.M.; visualization: C.S.H. and Y.M.T.; supervision: Z.T.T.M.;

Funding: There is no funding.

Conflicts of Interest: There are no conflicts of interest.

References

[1] M. Kuzin, Y. Shmelev, and V. Kuskov, "New trends in the world of IoT threats," *Securelist by Kaspersky*, 2018. https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/ (accessed Jul. 01, 2024).

[2] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022," 2022. [Online]. Available: https://cloud.report/Resources/Whitepapers/eea79d9b-9fe3-4018-86c6-3d1df813d3b8_white-paper-c11-741490.pdf

[3] CyberEdge Group, "2023 Cyberthreat Defense Report," 2023. [Online]. Available: https://www.cyberedgegroup.com/wp-content/uploads/2023/04/CyberEdge-2023-CDR-Report-v1.0.pdf

[4] ASEAN IVO, "ASEAN-Wide Cyber-Security Research Testbed." https://www.nict.go.jp/en/asean_ivo/ASEAN_IVO_2020_Project03.html (accessed Jul. 02, 2024).

[5] N. Anjum and M. R. Chowdhury, "International Journal of Advanced Research in Computer and Communication Engineering," *SSRN Electron. J.*, 2024, doi: 10.2139/ssrn.4847308.

[6] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The UCI KDD archive of large data sets for data mining research and experimentation," *ACM SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 81–85, Dec. 2000, doi: 10.1145/380995.381030.
[7] R. Kumar and D. Sharma, "HyINT: Signature-Anomaly Intrusion Detection System," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2018, pp. 1–7. doi: 10.1109/ICCCNT.2018.8494088.
[8] M. I. Akazue, I. A. Debekeme, A. E. Edje, C. Asuai, and U. J. Osame, "UNMASKING FRAUDSTERS: Ensemble Features Selection to Enhance Random Forest Fraud Detection," *J. Comput. Theor. Appl.*, vol. 1, no. 2, pp. 201–211, Dec. 2023, doi: 10.33633/jcta.v1i2.9462.
[9] S. Taheri and M. Mammadov, "Learning the naive Bayes classifier with optimization models," *Int. J. Appl. Math. Comput. Sci.*, vol. 23, no. 4, pp. 787–795, Dec. 2013, doi: 10.2478/amcs-2013-0059.
[10] H. Bennadi, K. Ibrahimi, and A. Benslimane, "Improving the Intrusion Detection System for NSL-KDD Dataset based on PCA-Fuzzy Clustering-KNN," in *2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Oct. 2018, pp. 1–6. doi: 10.1109/WINCOM.2018.8629718.
[11] G. Meena and R. R. Choudhary, "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA," in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, Jul. 2017, pp. 553–558. doi: 10.1109/COMPTELIX.2017.8004032.
[12] J. Alsamiri and K. Alsubhi, "Internet of Things Cyber Attacks Detection using Machine Learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 12, 2019, doi: 10.14569/IJACSA.2019.0101280.
[13] A. Ganesan, P. Parameshwarappa, A. Peshave, Z. Chen, and T. Oates, "Extending Signature-based Intrusion Detection Systems WithBayesian Abductive Reasoning," *ArXiv*. Mar. 28, 2019. [Online]. Available: http://arxiv.org/abs/1903.12101
[14] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to Snort system," *Futur. Gener. Comput. Syst.*, vol. 80, pp. 157–170, Mar. 2018, doi: 10.1016/j.future.2017.10.016.
[15] Y. Sugianela and T. Ahmad, "Pearson Correlation Attribute Evaluation-based Feature Selection for Intrusion Detection System," in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, Feb. 2020, pp. 1–5. doi: 10.1109/ICoSTA48221.2020.1570613717.
[16] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul, "Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset," *IEEE Access*, vol. 10, pp. 6430–6441, 2022, doi: 10.1109/ACCESS.2021.3140015.
[17] N. Saini, V. Bhat Kasaragod, K. Prakasha, and A. K. Das, "A hybrid ensemble machine learning model for detecting APT attacks based on network behavior anomaly detection," *Concurr. Comput. Pract. Exp.*, vol. 35, no. 28, Dec. 2023, doi: 10.1002/cpe.7865.
[18] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in IoT networks," *Inf. Sci. (Ny).*, vol. 639, p. 119000, Aug. 2023, doi: 10.1016/j.ins.2023.119000.
[19] A. R. E.-M. Baahmed, G. Andresini, C. Robardet, and A. Appice, "Using Graph Neural Networks for the Detection and Explanation of Network Intrusions," in *ECML PKDD International Workshop on eXplainable Knowledge Discovery in Data Mining*, 2023. [Online]. Available: http://xkdd2023.isti.cnr.it/papers/422.pdf
[20] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "An end-to-end intrusion detection system with IoT dataset using deep learning with unsupervised feature extraction," *Int. J. Inf. Secur.*, vol. 23, no. 3, pp. 1619–1648, Jun. 2024, doi: 10.1007/s10207-023-00807-7.
[21] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model," *Sci. Rep.*, vol. 12, no. 1, p. 15370, Sep. 2022, doi: 10.1038/s41598-022-19366-3.
[22] Wireshark Foundation, "tshark(1) Manual Page." https://www.wireshark.org/docs/man-pages/tshark.html (accessed Jul. 03, 2024).
[23] Canadian Institute for Cybersecurity, "CICFlowMeter (formerly ISCXFlowMeter)," *University of New Brunswick*. https://www.unb.ca/cic/research/applications.html.
[24] The Zeek Project, "Zeek: Network Security Monitoring," *Zeek Project*. https://zeek.org
[25] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011, Accessed: Feb. 17, 2020. [Online]. Available: http://scikit-learn.sourceforge.net.
[26] D. Freedman, R. Pisani, and R. Purves, *Statistics (International Student Edition)*, 4th ed. University of California, Berkeley; Boulder, Colorado: WW Norton, 2007.
[27] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Routledge, 2017. doi: 10.1201/9781315139470.
[28] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," *Stratosphere Lab*, 2020. https://www.stratosphereips.org/datasets-iot23
[29] D. R. I. M. Setiadi, S. Widiono, A. N. Safriandono, and S. Budi, "Phishing Website Detection Using Bidirectional Gated Recurrent Unit Model and Feature Selection," *J. Futur. Artif. Intell. Technol.*, vol. 2, no. 1, pp. 75–83, 2024, doi: 10.62411/faith.2024-15.
[30] F. Omoruwou, A. A. Ojugo, and S. E. Ilodigwe, "Strategic Feature Selection for Enhanced Scorch Prediction in Flexible Polyurethane Form Manufacturing," *J. Comput. Theor. Appl.*, vol. 1, no. 3, pp. 346–357, Feb. 2024, doi: 10.62411/jcta.9539.
[31] L. Yue, "Machine Learning and Deep Learning Methods for Better Anomaly Detection in IoT-23 Dataset Cybersecurity," *GitHub repository*, 2021. https://github.com/yliang725/Anomaly-Detection-IoT23
[32] N. A. Stoian, "Machine Learning for anomaly detection in IoT networks : Malware analysis," University of Twente, 2020. [Online]. Available: https://purl.utwente.nl/essays/81979