

Enhancing the performance of IoT network intrusion detection models using NF-ToN-IoT-V2 and IoTID20 datasets with chi-square feature selection

Nadia Thereza¹, Pardomuan Raja Harahap²

¹ Department of Electrical Engineering, University of Indonesia, Jl. Lingkar Pondok Cina, Depok 16424, Indonesia

² Department of Information Technology, PT. Rajawali Nusantara Indonesia, Jl. MT Haryono No.12, Jakarta Timur 13330, Indonesia

ARTICLE INFO

Article history:

Received
Revised
Published

Keywords:

Intrusion detection system;
Internet of things;
NF-ToN-IoT-V2;
IoTID20;
Machine learning;
Feature selection

ABSTRACT (10 PT)

The Internet of Things (IoT) expansion increases the number and size of networks and the volume of sensitive and private data. Consequently, IoT networks have become vulnerable to various threats and attacks. Researchers have recently devised intrusion detection systems (IDSs) to detect threats and attacks on IoT networks. However, in developing IDS for IoT networks, previous studies predominantly used more limited datasets to depict the actual IoT network characteristics. Thus, this research used datasets containing network flow records from real IoT networks, namely the NF-ToN-IoT-V2 and IoTID20. Various machine-learning algorithms, such as random forest, decision tree, naïve Bayes, AdaBoost, and XGBoost, were employed to train and evaluate the datasets for developing intrusion detection models. We investigated the model's performance based on accuracy, precision, recall, F1-score, false positive rate, training, and testing time utilization. We used the chi-square algorithm for feature selection to select the most relevant and valuable features. The findings indicate that implementing feature selection using chi-square improves the performance of the detection system models. By applying the Chi-Square algorithm, the RF model that outperforms in terms of accuracy performance increases its accuracy up to 0.42% on the NF-ToN-IoT-V2 testing and 0.16% on the IoTID20 testing. The DT model with the fastest training and testing time reduces its time utilization by 6.98% on the NF-ToN-IoT-V2 testing and 29.63% on the IoTID20 testing through feature selection.

This work is licensed under a [Creative Commons Attribution-Share Alike 4.0](https://creativecommons.org/licenses/by-sa/4.0/)



Corresponding Author:

Nadia Thereza, Department of Electrical Engineering, University of Indonesia, Jl. Lingkar Pondok Cina, Depok 16424, Indonesia
Email: nadia.thereza@ui.ac.id

1. INTRODUCTION

A fast-developing technical advancement, the Internet of Things (IoT) allows gadgets to identify their surroundings, interact online, and share data with little human assistance [1]. By 2030, IoT will generate vast traffic [2, 3], requiring the integration of various devices and equipment. However, the enormous amount of sensitive and private data generated by IoT devices makes it vulnerable to security attacks, making safeguarding it a crucial task. Low-security measures can be compromised by network malfunctions, data theft, accidents, intruders, malicious software, and viruses [4, 5].

Intrusion detection systems (IDS) are crucial in securing networks by detecting anomalous conditions and malicious activity, preventing unauthorized access, and detecting real-time data packets, making them popular for safeguarding IoT devices [6-8].

Cybersecurity solutions often rely on artificial intelligence, particularly in intrusion detection systems. Machine learning (ML) and deep learning (DL) approaches are used to develop these systems, requiring large datasets for accurate evaluation. ML enables computers to learn automatically and manage actions with minimal human involvement [9, 10], while DL improves systems' ability to recognize anomalous conditions. DL-based IDS have low false positive rates and greater accuracy, but their high complexity and size make implementation challenging, especially in IoT networks with more storage and processing capabilities [11-15].

Machine-learning-trained data and current network traffic datasets are necessary to develop an effective intrusion detection system (IDS) [16]. Antiquated network traffic is insufficient for capturing the authentic performance of intrusion detection systems in the current network environment [17]. UNSW_NB15 and NSL-KDD datasets have been extensively used in previous research but are not current with current network traffic. More IoT sensor data is needed to understand network system dynamics. IoTID20 and NetToN-IoT-V2 are IoT infrastructure datasets that depict genuine network models. The IoTID20 dataset, developed in 2020, collects data from IoT devices and connected object networks [18]. The NF-ToN-IoT-V2 dataset improves the IDS machine-learning algorithm's F1 score and detection precision [19], offering better intrusion detection results than the ToN-IoT dataset, which is the old version of NF-ToN-IoT-V2. Therefore, datasets such as the NF-ToN-IoT-V2 and IoTID20 that are capable of implying real IoT networks should be utilized to experimentally build intrusion detection models for IoT networks.

The accuracy of network intrusion detection systems relies on removing irrelevant features and incorporating only pertinent information [20]. Researchers have explored effective feature-selection techniques to improve classification results. The challenge lies in selecting essential characteristics without compromising classification accuracy, as different characteristics contribute differently to attack detection. The feature space significantly affects the accuracy of ML and DL-based approaches [21-23].

In this study, we investigated the performance of machine learning-based intrusion detection models for IoT networks using real dataset of IoT networks. It uses NF-ToN-IoT-V2 and IoTID20 datasets to be trained and tested using various algorithms. The Chi-square algorithm is used for feature selection, reducing irrelevant features and improving model performance. The study also investigates training and testing time, focusing on accuracy, precision, recall, F1 score, false positive rate, training, and testing time utilization. The results are evaluated to determine the best results for IoT networks.

2. METHODS

Through training and testing on the datasets using machine learning algorithms, this research aimed to build intrusion detection models that could effectively find attacks on the real IoT networks. A conceptual overview of the approach we suggest is depicted in Fig. 1. The IDS model was linked to IoT network nodes or devices in the actual IoT network implementation. Through this link, the IDS network acquired raw packets from the IoT network, which the IDS model subsequently extracted and evaluated.

The ML-based intrusion detection models were developed and evaluated by applying the *Sci-kit-learn* library in Python 3.9 and 64-bit ARM CPU architecture. The model was constructed on the cloud layer and utilized datasets for training and testing. At the onset of intrusion detection model development, feature selection was performed to reduce imbalanced data features in the dataset. In addition, five ML algorithms were deployed to train and evaluate datasets. After the study, the test results were analyzed using evaluation metrics, and the algorithm with the best performance was determined.

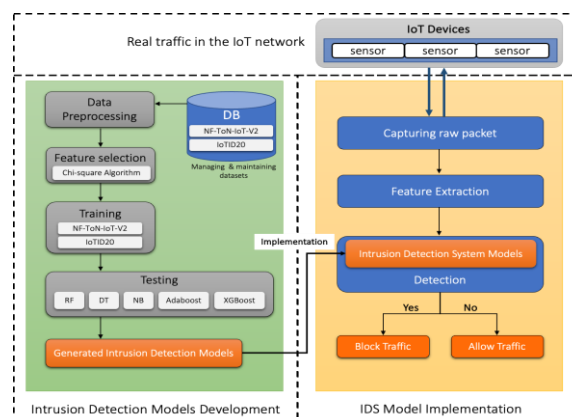


Fig. 1. The conceptual framework of the proposed system

2.1. Data Preparation and Processing

2.1.1. Dataset

This study utilized NF-ToN-IoT-V2 and leveraged IoTID20 as a benchmark dataset. The NF-ToN-IoT-V2 dataset contains network flow records from an IoT network. It is the latest version of NF-ToN-IoT and one of the datasets released by The University of Queensland, Australia. The ToN-IoT dataset's currently accessible packet capture files (pcaps) produce NetFlow records. This dataset has ten classes, including binary normal and anomaly [19]. The overall data flow is 16,940,496, with 63.99% attacks and 36.01% benign. Fig. 2 depicts the class distribution of the NF-ToN-IoT-V2 dataset.

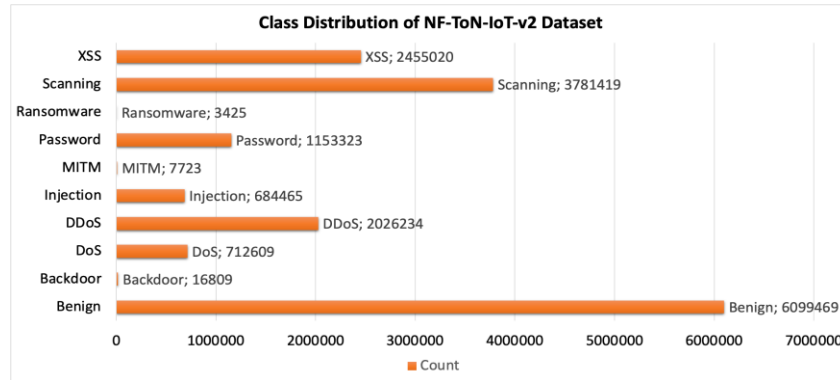


Fig. 2. Class distribution of NF-ToN-IoT-V2 dataset

Ullah and Mahmoud [24] adopted Kang's study [25] on various network attacks in the IoT environment and named it the IoTID20 dataset. This dataset represents current IoT network communication trends. The complete IoTID20 includes 83 network-specific features and 3 label features. Fig. 3 shows the graph of how the category labels are spread out in the IoTID20 dataset, which has 625,415 events.

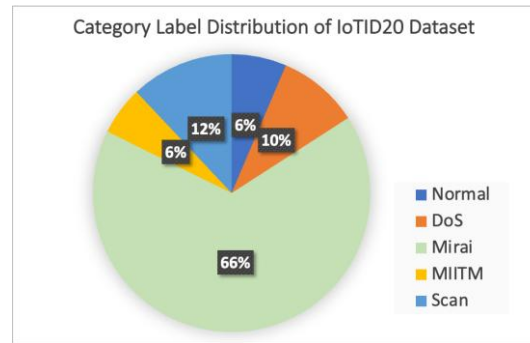


Fig. 3. Category distribution of IoTID20 dataset

2.1.2. Feature Selection

Implementation of the feature selection algorithm is a crucial stage in determining the optimal set of features that can be employed to distinguish benign from malicious traffic [26]. This process improves the time required for computation [21]. The Chi-square algorithm is considered when determining the optimal set of features. This algorithm determines the independence between a characteristic and its respective class label. It takes into account the feature dimension values and labels. The symbols in the calculation consist of p , q , r , and s , as shown in Equation (1). " p " symbolizes the frequency with which t and c co-occur; " q " symbolizes the frequency with which t occurs without c ; " r " symbolizes the frequency with which c occurs without t ; " s " denotes the frequency with which neither t nor c occurs; N shows the total number of instances recorded in the dataset; t and c represent the examined feature dimension and label [26-28]. The Chi-square formula is shown in the following equation:

$$x^2(t, c) = \frac{N(ps - qr)^2}{(p + r)(q + s)(p + q)(r + s)} \quad (1)$$

The dataset was preprocessed before the features were optimized using feature selection techniques. Table 1 presents the pseudocode algorithm for feature selection employing the *sci-kit-learn* library.

Table 1. Pseudocode for feature selection

Algorithm 1: Feature selection procedures

Input

data_set : dataset

k_count_of_selected_feature : total feature defined

drop_feature_in_data: list of non-calculated

→ will be discarded

Output

feature_selected : the selected features

Step 1. load the input *data_set*

Step 2. load the input *k_count_of_selected_feature*

Step 3. load the input *drop_feature_in_data*

Step 4. drop data containing features or field headers referred to the list *drop_feature_in_data*

→ set it to *data_all_clean*.

Step 5. obtain the outcome data from *data_all_clean*, referred to field name from *field_target_outcome*

→ put it to *data_out_come*

Step 6. run *run_chi_square* with the input parameters

data_all_clean

k_count_of_selected_feature

data_out_come

→ insert to *feature_selected*

We determined the number of selected features denoted by *k*. For NF-ToN-IoT-V2, which has 45 feature columns, the value of *k* was determined to vary between 10, 25, and 35. Fig. 4 shows the feature selection results for the NF-ToN-IoT-V2 using Chi-square selector.

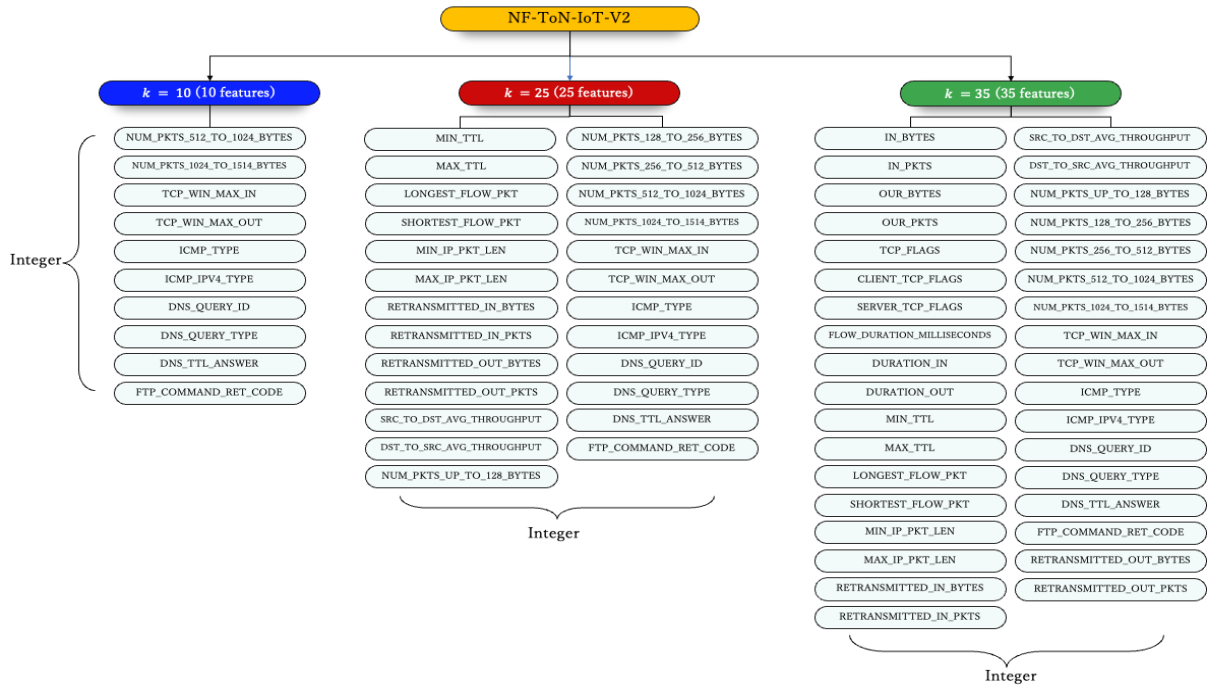


Fig. 4. Feature selection results for NF-ToN-IoT-V2 dataset

We decided that the number of selected features for the IoTID20 dataset should range from 20 to 45 and 59. Fig. 5 depicts the feature selection results for the IoTID20 using the Chi-square selector.

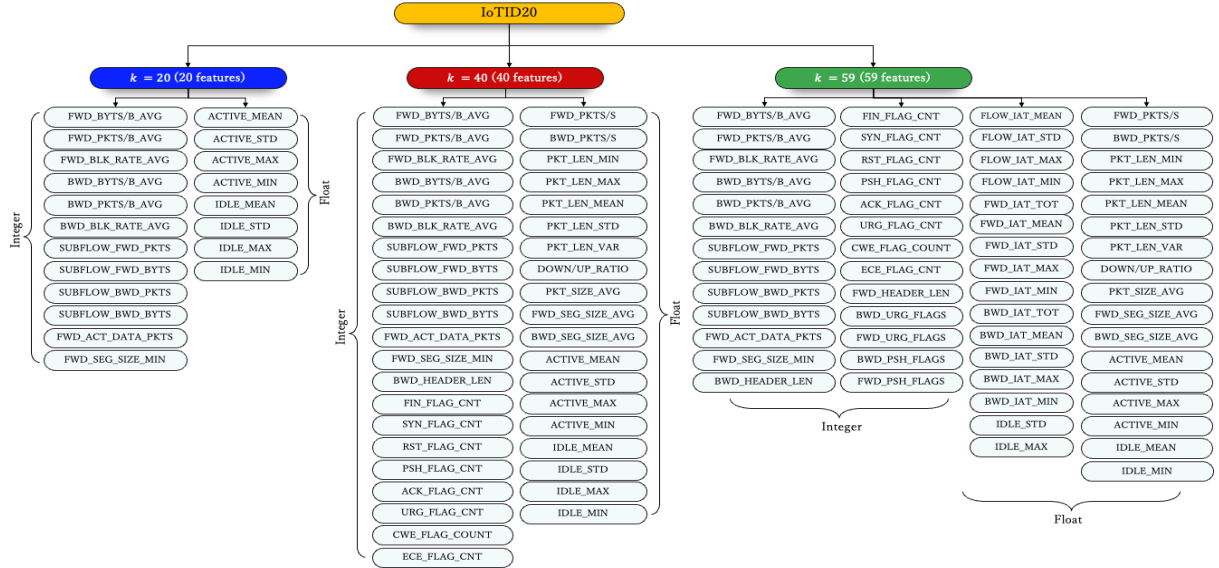


Fig. 5. Feature selection results for the IoTID20 dataset

2.2. Training and Testing

The training and testing stages are essential in the development of an IDS. In this study, we used ML techniques that estimate the value of a new sample on the basis of patterns discovered in a large dataset. Multiple ML algorithms were implemented, including Random Forest, Decision Tree, Naïve Bayes, AdaBoost, and XGBoost. The intrusion detection model in this study was developed using the Python programming language and modules from the Scikit-learn library. One of the hyperparameters set was $n_estimators$ in the random forest algorithm. The $n_estimators$ is the number of trees that must be implemented. The number of trees in this research was determined at 50 ($n_estimators = 50$). Thus, a random forest will create 50 decision trees, each trained with a random subset of the data and features, and then combine them to make the final decision. Apart from the $n_estimators$ in the random forest algorithm, the hyperparameters in the other algorithms are applied following the default settings of the Scikit-learn library. Some of the hyperparameters that follow the default settings are as follows: $max_depth = None(int)$, $random_state = None(int)$ and $splitter = best$ for Decision Tree; $var_smoothing = 1e - 9(float)$ and $priors = None$ for Naïve Bayes; $n_estimators = 50(int)$, $learning_rate = 1.0(float)$, and $estimator = None$ for AdaBoost classifier; and $learning_rate = 0.1(float)$ and $subsample = 1.0(float)$ for Extreme Gradient Boost Classifier (XGBoost).

2.3. Performance Evaluation Metrics

Multiple metrics are utilized to investigate the effectiveness of an ML-based IDS. Four parameters are needed to calculate system performance: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

a) Accuracy

Accuracy score (Acc) is the most often used metric to evaluate model performance in binary classification conditions. It is calculable as

$$Acc = (TP + TN) / (TP + TN + FP + FN) \quad (4)$$

b) Precision

Precision (Prec) is defined as the classification capability of the model system to precisely identify attacks out of the total number of correct predictions. It is calculable as:

$$Prec = \frac{TP}{TP + FP} \quad (5)$$

c) Recall

Recall or detection rate (DR) is a metric of how well the model IDS detects true positive samples, which means it correctly distinguishes attacks from actual attacks. It is referred to as the *true positive rate* or *sensitivity*. It is calculable as:

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

d) F1 Score

The F1 score is the compromise between precision and recall that considers false positive and false negative. It is calculable as:

$$F1\ Score = 2 \times \left(\frac{Recall \times Precision}{Recall + Precision} \right) = \frac{2TP}{2TP + FP + FN} \quad (7)$$

e) False Positive Rate

The false positive rate (FPR) is determined as the ratio of the error when normal samples are predicted as positive attacks to the total quantity of normal samples. It implies an excessive estimation that incorrectly necessitates human intervention. When detecting intrusions, less FN is required because it is more dangerous than FP. The FPR is calculable as

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

3. RESULTS AND DISCUSSION

The intrusion detection models were built using several ML algorithms: RF, DT, NB, AdaBoost, and XGBoost. We used several groups of NF-ToN-IoT-V2 and IoTID20 datasets for training and testing, which were previously selected using the Chi-square algorithm. The dataset groups were divided on the basis of the value of k (number of selected features), namely k groups 10, 25, and 35 for the NF-ToN-IoT-V2 and k 20, 40, and 59 for the IoTID20. The test-size parameter was used to define the size of the testing data divided into three groups, namely, 20%, 50%, and 80% of the entire NF-ToN-IoT-V2 and IoTID20.

Table 2 lists the results of evaluating the NF-ToN-IoT-V2 with $k = 10$ and $k = 25$, and Table 3 lists the results of assessing with $k = 35$. The RF model produced the highest accuracy of 96.58% with $k = 35$. The DT model achieved an accuracy value of 96.05% with $k = 35$. The RF model achieved the highest precision value of 99.54%, and the DT model gained 98.91%. The best recall and F1 scores were 91.73% and 95.48%, both achieved by the RF algorithm at $k = 35$. From Table 2 and Table 3, we see that k could affect the performance value of the system model. The resulting model performance value was better when applying a dataset with a more significant number of selection features (indicated by a larger k value).

However, the differentiation value is evident when we talk about the FPR. There is no expectation of a high value from the FPR measurement results. Unlike other performances, the expected value of the FPR metric measurement was the lowest. This finding indicated that the smaller the measurement value when normal samples are misclassified as positive attacks compared to the total number of truly normal samples, the better the performance. In applying attack detection systems in IoT networks, suppressing the number of false positives is essential to maintain the smooth operation, efficiency, and reliability of the detection system applied to services that utilize high-risk IoT. The development of detection models must consider the trade-off between sensitivity and precision so that the system remains responsive without causing unnecessary interference. Health services, industry, and plantation land monitoring are examples of high-risk IoT applications. As in health services, false detections can cause systems to block legitimate devices or services automatically, halt production processes, or cut off critical communications. Similar to industrial operations, FPR generates false alarms that require manual action. Such behavior increases the workload for security teams and engineers and removes resources that should be focused on real threats. Frequent false alarms lead operators to disregard warnings, heightening the danger that genuine threats will be overlooked. In IoT systems vulnerable to delays or disruptions, such as medical equipment or smart grids, false positives might result in diminished efficiency, postponed responses, or even safety risks.

In the results of the IoTID20 dataset testing shown in Table 4 and Table 5, the accuracy values achieved by the RF and DT models outperformed the other models. RF produced 98.89% accuracy, and DT achieved 98.53% accuracy. The RF algorithm obtained the best precision, F1 score, and FPR with percentages of 87.62,

92.2%, and 0.987%, respectively. The XGBoost model achieved the best recall with a 100% rate. However, the precision value obtained by the XGBoost model was relatively low, at 38-57%.

Table 2. NF-ToN-IoT-V2 dataset evaluation results of the model with $k = 10$ and $k = 25$

Algorithm	Performance Metrics (%)										
	Test size (%)	$k = 10$					$k = 25$				
		Acc	Prec	Recall	F1 score	FPR	Acc	Prec	Recall	F1 score	FPR
Random Forest	20	91.29	94.75	84.62	89.4	3.59	95.48	99.27	89.5	94.13	0.45
	50	91.28	94.75	84.6	89.38	3.6	95.47	99.27	89.48	94.12	0.45
	80	91.27	94.74	84.58	89.37	3.6	95.46	99.27	89.46	94.11	0.45
Decision Tree	20	90.42	93.99	83.65	88.52	4.26	94.93	98.66	88.92	93.54	0.85
	50	90.27	93.82	83.47	88.36	4.35	94.89	98.62	88.87	93.49	0.87
	80	90.05	93.66	83.18	88.11	4.48	94.83	98.56	88.79	93.42	0.91
Naïve Bayes	20	47.89	1.86	78.29	3.67	52.62	54.01	2.26	87.8	4.42	46.46
	50	20.83	1.87	77.96	3.67	80.37	54.02	2.26	87.7	4.42	46.45
	80	14.11	1.91	77.66	3.75	87.35	53.97	2.42	82.62	4.72	46.51
Adaboost	20	85.74	92.04	78.59	84.79	6.95	58.4	87.57	47.88	61.91	16.33
	50	78.68	89.01	68.21	77.24	9.5	69.58	83.38	58.93	69.06	15.96
	80	76.81	91.17	64.01	76.86	3.85	68.83	81.94	59.54	68.96	18.25
XGBoost	20	86.05	88.31	80.8	84.39	9.36	88.89	85.18	86.76	85.96	9.74
	50	86.02	88.3	80.77	84.37	9.37	88.88	85.18	86.74	85.96	9.74
	80	85.69	88.38	80.14	84.06	9.37	88.98	85.23	86.85	86.03	9.66

Table 3. NF-ToN-IoT-V2 dataset evaluation results of the model with $k = 35$

Algorithm	Performance Metrics (%)					
	$k = 35$					
	Test size (%)	Acc	Prec	Recall	F1 score	FPR
Random Forest	20	96.58	99.54	91.73	95.48	0.27
	50	96.57	99.54	91.71	95.47	0.27
	80	96.57	99.54	91.7	95.46	0.27
Decision Tree	20	96.05	98.91	91.13	94.86	0.67
	50	96	98.87	91.08	94.82	0.69
	80	95.95	98.8	91.01	94.75	0.74
Naïve Bayes	20	50.61	2.07	88.74	4.06	49.91
	50	50.35	2.08	88.52	4.07	50.17
	80	50.38	2.18	85.28	4.27	50.15
Adaboost	20	59.65	9.71	50.28	16.27	39.56
	50	61.54	9.86	59.43	16.92	38.31
	80	55.53	10.46	43.25	16.85	43.04
XGBoost	20	88.68	84.87	86.57	85.71	9.96
	50	88.67	84.88	86.55	85.71	9.95
	80	88.78	84.97	86.66	85.81	9.85

Table 4. IoTID20 dataset evaluation results of the model with $k = 20$ and $k = 40$

Algorithm	Performance Metrics (%)										
	$k = 20$						$k = 40$				
	Test size (%)	Acc	Prec	Recall	F1 score	FPR	Acc	Prec	Recall	F1 score	FPR
Random Forest	20	98.78	86.46	97.17	91.5	1.1	98.89	87.54	97.28	92.15	0.99
	50	98.75	85.92	97.13	91.18	1.13	98.84	86.77	97.22	91.7	1.05
	80	98.77	85.91	97.25	91.23	1.17	98.84	86.71	97.3	91.7	1.04
Decision Tree	20	98.46	83.18	96.11	89.18	1.37	98.53	83.99	95.97	89.58	1.29
	50	98.43	83.37	95.17	88.88	1.33	98.54	84.6	95.19	89.58	1.22
	80	98.24	82.65	93.35	87.68	1.41	98.35	84.41	92.84	88.43	1.25
Naïve Bayes	20	52.28	82.21	16.75	27.83	4.42	70.69	79.02	22.71	35.29	3.27
	50	52.57	82.72	16.65	27.72	4.19	71.04	79.93	22.71	35.37	3.06
	80	52.71	82.69	16.79	27.91	4.21	71.55	80.01	22.8	35.48	2.98
Adaboost	20	61.34	61.88	12.27	20.49	5.15	91.21	14.99	34.75	20.95	6.83
	50	60.03	65.65	12.29	20.7	4.74	94.4	52.21	67.52	58.89	3.9
	80	91.74	23.22	50.72	31.86	6.64	56.66	70.25	11.65	19.99	4.28
XGBoost	20	95.14	38.58	100	55.68	5.01	96.61	57.74	97.89	72.64	3.45
	50	95.2	38.37	100	55.46	4.95	96.68	57.75	98.13	72.71	3.39
	80	95.16	38.02	100	55.09	4.99	96.66	57.46	98.46	72.57	3.42

From the test results using the IoTID20 dataset, the Naïve Bayes model produced an accuracy of 52–86%, whereas when testing is carried out with the NF-ToN-IoT-V2, the Naïve Bayes model could only produce an

accuracy of 14–54%. The size of the dataset was one of the factors that led to this result. When trained and tested using the large NF-ToN-IoT-V2 with 16,940,496 instances, NB does not produce effective system performance. Also, the research results from the experiment using the NF-ToN-IoT-V2 with $k = 35$ and a test size of 20% showed that the number of detected FPs was 1,195,145, while the numbers of TP and TN were 25,306 and 1,199,676, respectively. This condition indicated a reasonably high FP value compared with the TP value.

Table 5. IoTID20 dataset results of evaluating the model with $k = 59$

Algorithm	Performance Metrics (%)					
	$k = 59$					
	Test size (%)	Acc	Prec	Recall	F1 score	FPR
Random Forest	20	98.89	87.62	97.28	92.2	0.987
	50	98.84	86.82	97.25	91.74	1.043
	80	98.85	86.76	97.3	91.73	1.04
Decision Tree	20	98.53	84.02	95.97	89.6	1.29
	50	98.54	84.42	95.36	89.56	1.24
	80	98.33	84.19	92.76	88.27	1.26
Naïve Bayes	20	85.96	73.29	41.05	52.62	3.51
	50	86.2	73.64	41.06	52.73	3.19
	80	86.2	73.61	41.15	52.79	3.4
Adaboost	20	91.58	28.24	44.48	34.55	5.95
	50	93.74	34.17	71.58	46.25	5.39
	80	43.26	79.84	59.94	37.9	1.85
XGBoost	20	96.01	54.82	89.83	68.1	3.69
	50	96.65	57.18	98.58	72.38	3.43
	80	96.71	57.98	98.59	73.02	3.38

We also conducted training and testing without feature selection stages to see the significance of performance between model developments with (w/ FS) and without feature selection (w/o FS). We summarize the highest success rate that the machine learning model achieved when tested using feature selection and contrast it with the outcomes of experiments conducted without feature selection. The comparison of the results shown in Fig. 6 and Fig. 7 indicated that the system built with data processing through feature selection produces better performances than those without feature selection. The feature selection process eliminates features considered to not provide important information for the model training and testing process. These irrelevant features can be regarded as noise, so that they can impact the algorithm when learning more efficiently and accurately. Irrelevant features make it difficult for the model to recognize correct patterns in the data, because the signal (important information) is mixed with noise (unimportant or misleading information). Although the increase in accuracy was not so significant in this study, it still shows that feature selection has quite an impact on the accuracy of the attack detection system.

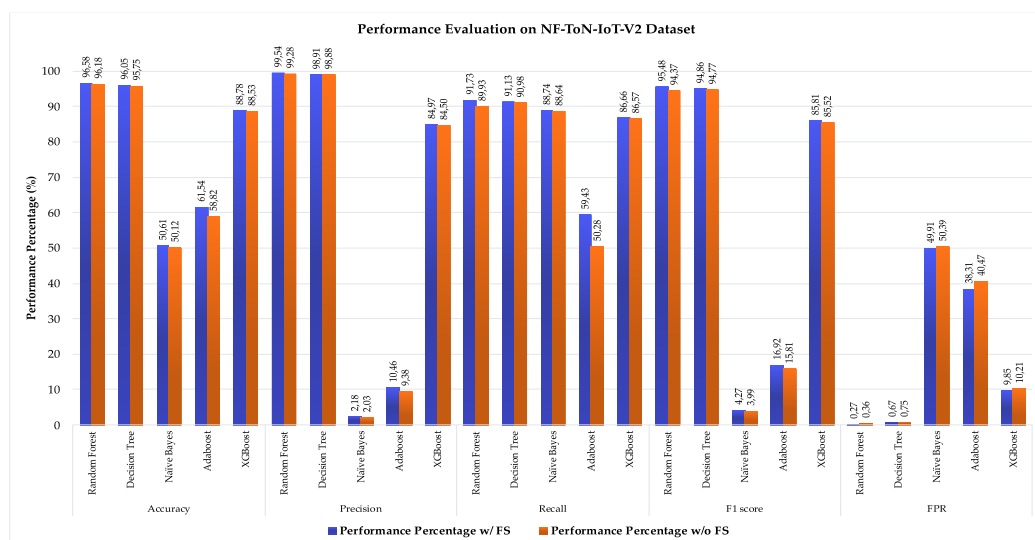


Fig. 6. Performance results with and without feature selection for NF-ToN-IoT-V2 dataset

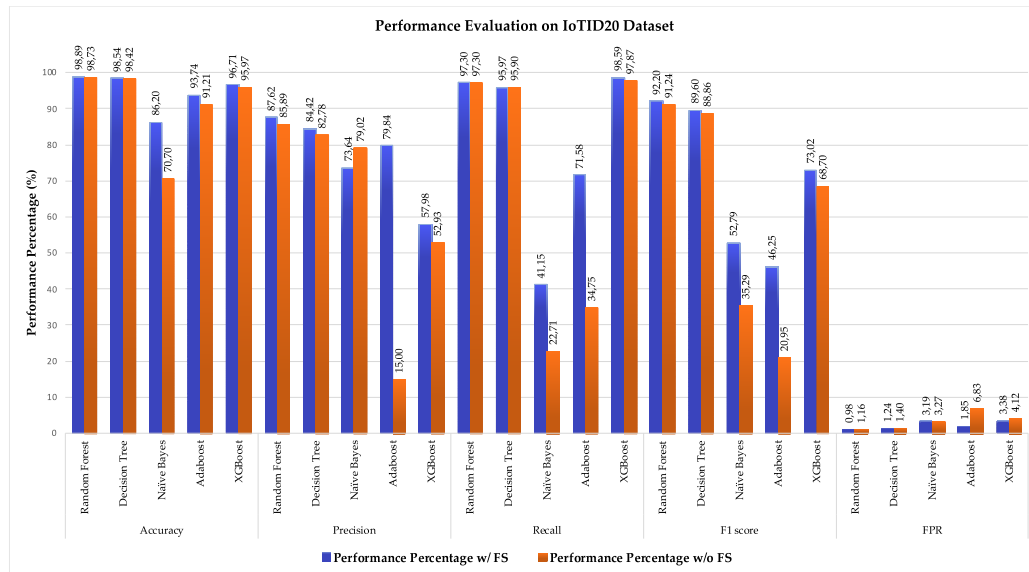


Fig. 7. IoTID20 dataset performance results with and without feature selection

We conducted a comparative analysis of the training and testing duration of the model with and without the standard feature selection techniques. We determined the test size in each experiment to compare the results of measuring the training and testing times. The selected test size was 80% of the dataset, so for NF-ToN-IoT-V2, the total sample tested was 13,552,397, while for IoTID20 dataset, it was 500,332 instances. The results of training and testing time measurements are presented in Table 6 and Table 7. Fig. 8 provide a more explicit comparison of the measurement results.

Table 6. The testing results for the NF-ToN-IoT-V2 dataset with and without feature selection

Feature Selection	Training time (second), test size 80%					Testing time (second), test size 80%				
	RF	DT	NB	Adaboost	XGBoost	RF	DT	NB	Adaboost	XGBoost
with	935.32	20.71	6.33	363.57	259.9	189.26	4.53	50.46	183.57	8.8
without	1007.29	20.97	6.51	373.02	273.67	196.35	4.87	51.37	198.44	11.41

Table 7. The testing results for the IoTID20 dataset with and without feature selection

Feature Selection	Training time (second), test size 80%					Testing time (second), test size 80%				
	RF	DT	NB	Adaboost	XGBoost	RF	DT	NB	Adaboost	XGBoost
with	24.17	0.87	0.19	12.49	5.24	2.54	0.19	0.59	2.78	0.28
without	33.2	1.22	0.29	16.06	7.43	3.08	0.27	1.14	3.98	0.36

Fig. 8 shows the results of training and testing time measurements for NF-ToN-IoT-V2 and IoTID20 dataset. The results show that the RF model produced the longest training time, and the NB model produced the shortest. The RF model also required longer testing time than the other models. It classified 13,552,397 instances within 189.26 s, which means $\approx 71,607$ instances within 1 s. On the other hand, the DT model produced the shortest testing time and could classify $\approx 2,991,699$ instances in 1 s. The RF model excelled in accuracy, precision, recall, F1 score, and FPR because it could produce the best performance. In contrast, the DT model excelled in training and testing time because it could classify samples in the shortest time (0.334 μ s for one instance). Like the time measurement on the IoTID20 dataset, RF requires more training time than the other models, while the NB model requires the least training time. When testing using 80% of the IoTID20 dataset, the performance of the DT model still outperformed it by producing the shortest testing time. The DT model could classify 500,332 samples in 0.19 s or detect 1 sample in 0.38 μ s.

In addition, we investigated and compared the results of training and testing time measurements for those selected for features and those not selected for features. In testing with feature selection, we selected data $k = 35$ with a test size of 80% for NF-ToN-IoT-V2 and $k = 59$ with a test size of 80% for IoTID20. We choose the same size at 80% of the entire dataset for tests without feature selection. Machine learning models required a longer time to train and test datasets without feature selection. In other words, applying feature selection using the Chi-square algorithm effectively reduced ML models' training and testing time.

In the test without feature selection, the DT model detected attacks in 80% of the IoTID20 dataset of 500,332 instances in 0.27 s, whereas, with feature selection, the RF took only 0.19 s. Also, when tested on the 80% NF-ToN-IoT-V2 dataset, DT could classify 13,552,397 instances in 4.53 s with feature selection, but it took 4.87 s without it. With the feature selection stage, adaBoost and XGBoost were able to classify 13,552,397 instances in 183.57 s and 8.8 s, whereas, without feature selection, AdaBoost and XGBoost needed 198.44 s and 11.41 s, respectively. By conducting FS, the RF model took 935.32 s to train the NF-ToN-IoT-V2 dataset with $k = 35$, whereas, without FS, it was 1007.29 s. The RF model classified 80% of the NF-ToN-IoT-V2 without FS for 196.35 s, whereas if using FS with $k = 35$, the time required was 189.26 s.

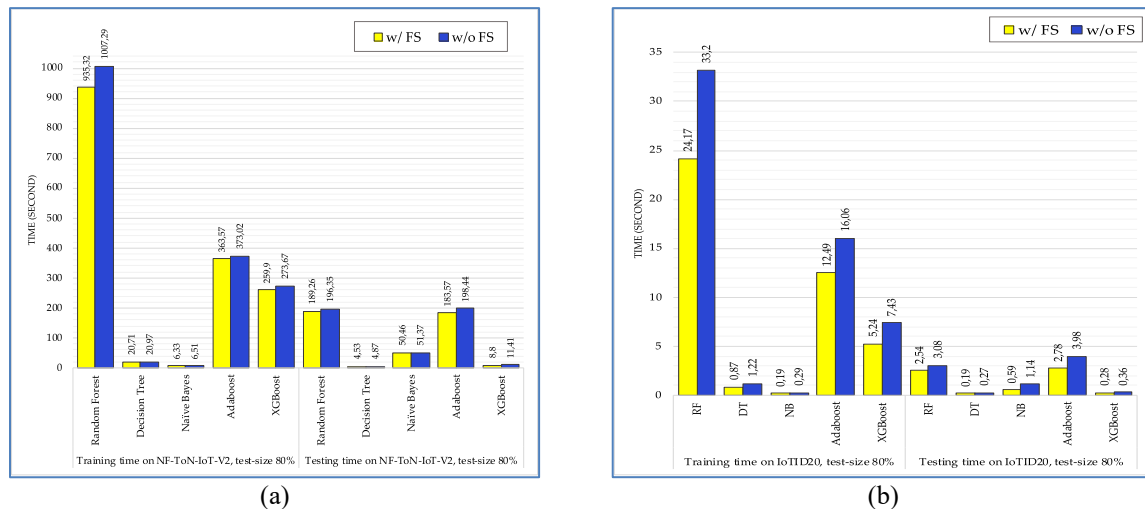


Fig. 8. Training and testing time comparison with and without feature selection on NF-ToN-IoT-V2 and IoTID20 datasets. Feature selection reduces computation time, supporting efficient deployment in real-time IoT systems.

The reduction of training and testing time indicates better computational efficiency Fig. 8. The balance between computational efficiency and model complexity must be considered during the actual implementation of IoT systems because many IoT devices operate in resource-limited environments, such as low battery power, simple processors, and memory limitations. The findings indicate that using chi-square feature selection techniques can decrease the dimensionality of the dataset and enhance the efficiency of training and testing time. This selection process also indirectly impacts energy resource consumption. So, this is important in IoT systems requiring fast response and limited resources, such as real-time security devices and systems. Despite the reduction in training time, the accuracy rate remains high (achieved by RF with a value above 95%), indicating that the model has become more efficient without reducing its detection ability. The short training time allows the model to be developed and quickly adapted to new attack patterns.

Along with current technology development, IoT applications are predicted to generate much greater traffic in the coming years. Of course, it will also impact the increase in cybercrime: the more cybercrime increases, the more diverse the attacks are. Therefore, a short and efficient training time is needed to develop and update the model to be more adaptive to new attack patterns that are developing dynamically. A shorter training duration can save computing resources, namely CPU, RAM, and energy resources. Running the model on IoT devices with limited power and capabilities is crucial. Minimizing testing duration is essential for the rapid identification and response to risks. The more rapidly a system identifies an attack, the less the danger of system failure, data breach, or service interruption diminishes. For example, in a smart healthcare system, rapid detection enables instant identification and cessation of attacks on medical devices, thereby averting extensive harm. Similarly, early detection of an attack on the power grid in a smart grid allows stakeholders to take quick action before a blackout or widespread system failure occurs.

Furthermore, learning techniques also must be considered to balance computational efficiency and model complexity. Artificial Intelligence (AI), primarily through Deep Learning (DL) and Machine Learning (ML) approaches, has become an essential solution in intrusion detection systems (IDS) for cybersecurity, including in IoT environments. However, its implementation faces challenges due to power and memory limitations on

IoT devices. Although effective in managing complex data and providing high accuracy, DL algorithms require larger resources and longer testing and training times. In contrast, ML algorithms are lighter in terms of complexity, require shorter training times, and are more efficient to implement in resource-limited IoT systems. Therefore, we choose an efficient and lightweight security approach that adapts to the character of the IoT network by using machine learning algorithms and implementing feature selection.

For future research directions, we propose a hybrid method, developing an intrusion detection system (IDS) that employs classification and clustering algorithms, to enhance accuracy and significantly reduce training and testing (detection) time in multi-class classification scenarios. The dataset will first be grouped using a clustering algorithm, then each group will be trained using a machine learning classification algorithm so that more specific detection models will be produced. This approach is considered adequate because some classification models have limitations in detecting new attacks. We expect the built IDS to efficiently group data and remain adaptive to unknown threats by combining supervised (classification) and unsupervised (clustering) methods.

4. CONCLUSION

This study evaluated the performance of various machine learning algorithms, Random Forest, Decision Tree, Naïve Bayes, AdaBoost, and XGBoost, in building intrusion detection models for IoT networks using real datasets: NF-ToN-IoT-V2 and IoTID20. Applying the Chi-Square feature selection effectively reduces irrelevant attributes, improving detection performance and computational efficiency. Among all models, Random Forest consistently achieved the highest accuracy, 96.58% for NF-ToN-IoT-V2 and 98.89% for IoTID20. Compared to the Decision Tree (DT), the Random Forest (RF) model produces higher accuracy. This is because RF is built from a set of decision trees. Combining many decision trees to make decisions makes RF more robust to errors from individual trees. Meanwhile, the Decision Tree model outperforms other algorithms regarding training and testing time, highlighting its suitability for time-sensitive and resource-constrained environments.

The actual implementation of IDS for IoT networks should consider the balance between computational efficiency and model complexity, as numerous IoT devices operate in resource-limited environments. In this study, the chi-square feature selection technique enhances the accuracy and efficiency of training and testing time. The RF model that outperforms in terms of accuracy performance increases its accuracy by up to 0.42% on the NF-ToN-IoT-V2 testing and 0.16% on the IoTID20 testing by implementing the Chi-Square algorithm. The DT model with the fastest training and testing time reduces its time utilization by 6.98% on the NF-ToN-IoT-V2 testing and 29.63% on the IoTID20 testing through feature selection.

Our proposed future study is combining classification and clustering techniques to improve accuracy and reduce training and testing (detection) time in multi-class classification scenarios for IDS. After a clustering technique groups the dataset, a machine learning classification algorithm will train each group to create more specialized detection models. We hope this future study can improve detection accuracy and reduce training and attack detection duration.


REFERENCES

- [1] P. M. Chanal and M. S. Kakkasageri, "Security and privacy in IoT: a survey," *Wireless Personal Communications*, vol. 115, no. 2, pp. 1667-1693, 2020, doi: 10.1007/s11277-020-07649-9.
- [2] A. Yastrebova, R. Kirichek, Y. Koucheryavy, A. Borodin, and A. Koucheryavy, "Future Networks 2030: Architecture & Requirements," *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 1-8, 2018, doi: 10.1109/ICUMT.2018.8631208.
- [3] H. F. Atlam and G. B. Wills, "IoT security, privacy, safety and ethics," *Digital twin technologies and smart cities*, pp. 123-149, 2020, doi: 10.1007/978-3-030-18732-3_8.
- [4] R. Chaganti, W. Suliman, V. Ravi, and A. Dua, "Deep Learning Approach for SDN-Enabled Intrusion Detection System in IoT Networks," *Information*, vol. 14, no. 1, p. 41, 2023, doi: 10.3390/info14010041.
- [5] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Generation Computer Systems*, vol. 127, pp. 276-285, 2022, doi: 10.1016/j.future.2021.09.027.
- [6] E. Gyamfi and A. Jurcut, "Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets," *Sensors*, vol. 22, no. 10, p. 3744, 2022, doi: 10.3390/s22103744.
- [7] K. Albulayhi and F. T. Sheldon, "An adaptive deep-ensemble anomaly-based intrusion detection system for the internet of things," in *2021 IEEE World AI IoT Congress (AllIoT)*, 2021: IEEE, pp. 0187-0196, doi: 10.1109/AllIoT52608.2021.9454168.

- [8] K. Albulayhi, A. A. Smadi, F. T. Sheldon, and R. K. Abercrombie, "IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses," *Sensors*, vol. 21, no. 19, p. 6432, 2021, doi: 10.3390/s21196432.
- [9] P. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, "Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: A Survey," *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3220622.
- [10] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "A machine learning security framework for iot systems," *IEEE Access*, vol. 8, pp. 114066-114077, 2020, doi: 10.1109/ACCESS.2020.2996214.
- [11] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "SMOTE-DRNN: A Deep Learning Algorithm for Botnet Detection in the Internet-of-Things Networks," *Sensors*, vol. 21, no. 9, p. 2985, 2021, doi: 10.3390/s21092985.
- [12] D. Unal, M. Hammoudeh, M. A. Khan, A. Abuarqoub, G. Epiphaniou, and R. Hamila, "Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things," *Computers & Security*, vol. 109, p. 102393, 2021, doi: 10.1016/j.cose.2021.102393.
- [13] R. Elsayed, R. Hamada, M. Hammoudeh, M. Abdalla, and S. A. Elsaid, "A Hierarchical Deep Learning-Based Intrusion Detection Architecture for Clustered Internet of Things," *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, p. 3, 2022, doi: 10.3390/jsan12010003.
- [14] Y. Yang, S. Tu, R. H. Ali, H. Alasmay, M. Waqas, and M. N. Amjad, "Intrusion detection based on bidirectional long short-term memory with attention mechanism," 2023, doi: 10.32604/cmc.2023.031907.
- [15] R. Zhao *et al.*, "A novel intrusion detection method based on lightweight neural network for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9960-9972, 2021, doi: 10.1109/JIOT.2021.3119055.
- [16] R. Damasevicius *et al.*, "LITNET-2020: An Annotated Real-World Network Flow Dataset for Network Intrusion Detection," *Electronics*, vol. 9, no. 5, p. 800, 2020, doi: 10.3390/electronics9050800.
- [17] H. Zhang, B. Zhang, L. Huang, Z. Zhang, and H. Huang, "An Efficient Two-Stage Network Intrusion Detection System in the Internet of Things," *Information*, vol. 14, no. 2, p. 77, 2023, doi: 10.3390/info14020077.
- [18] H. Alkahtani and T. H. H. Aldhyani, "Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms," *Complexity*, vol. 2021, p. 5579851, 2021/07/07 2021, doi: 10.1155/2021/5579851.
- [19] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a Standard Feature Set for Network Intrusion Detection System Datasets," *Mobile Networks and Applications*, vol. 27, pp. 1-14, 02/01 2022, doi: 10.1007/s11036-021-01843-0.
- [20] G. Dong, & Liu, H. , *Feature Engineering for Machine Learning and Data Analytics* 1st ed. CRC Press, 2018.
- [21] T. Wisanwanichthan and M. Thammawichai, "A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM," *IEEE Access*, vol. 9, pp. 138432-138450, 2021, doi: 10.1109/ACCESS.2021.3118573.
- [22] J. Fuhr, F. Wang, and Y. Tang, "MOCA: A Network Intrusion Monitoring and Classification System," *Journal of Cybersecurity and Privacy*, vol. 2, pp. 629-639, 08/15 2022, doi: 10.3390/jcp2030032.
- [23] S. M. Kasongo and Y. Sun, "A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System," *IEEE Access*, vol. 7, pp. 38597-38607, 2019, doi: 10.1109/ACCESS.2019.2905633.
- [24] I. Ullah and Q. H. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," in *33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020*, Canada, 2020: Springer, pp. 508-520, doi: 10.1007/978-3-030-47358-7_52.
- [25] H. Kang, D. H. Ahn, G. M. Lee, J. Do Yoo, K. H. Park, and H. K. Kim, "IoT network intrusion dataset. IEEE Dataport," ed, 2019.
- [26] H. M. Alshahrani, "CoLL-IoT: A Collaborative Intruder Detection System for Internet of Things Devices," *Electronics*, vol. 10, no. 7, p. 848, 2021, doi: 10.3390/electronics10070848.
- [27] D. Sharpe, "Chi-square test is statistically significant: Now what?," *Practical Assessment, Research, and Evaluation*, vol. 20, no. 1, p. 8, 2015, doi: 10.7275/tbfa-x148.
- [28] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400-412, 2014, doi: 10.1109/TDSC.2014.2355839.

BIOGRAPHY OF AUTHORS

Nadia Thereza received a bachelor's degree in electrical engineering from Sriwijaya University, South Sumatra, Indonesia, in 2012 and a master's degree in electrical engineering from the School of Electrical Engineering and Informatics, Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2015. She is currently pursuing a Ph.D. degree in electrical engineering at the Faculty of Engineering, Universitas Indonesia (UI), Depok, Indonesia. Since 2019, she has been a lecturer at the department where she got her bachelor's degree in electrical engineering, Universitas Sriwijaya, Indonesia. Her research interests include cyber security, network security, and the Internet of Things.

She can be contacted at: nadia.thereza@ui.ac.id. Orcid id:  <https://orcid.org/0000-0001-8311-8634>

Pardomuan Raja Harahap received a bachelor's degree in informatics engineering from Sriwijaya University, South Sumatra, Indonesia, in 2012. Since 2014, he has been an IT developer at PT. Perkebunan Mitra Ogan, Palembang, South Sumatra. In 2019, he became an IT manager at PT. Rajawali Nusindo. Currently, he is an assistant vice president in the information technology development (IT Development) division at PT. Rajawali Nusantara Indonesia. His areas of interest include networking and infrastructure, machine learning, software development and programming, data science and analytics, and information systems management. He can be contacted at: raja.if07@gmail.com