electronics

MDPI

Article

SUKRY: Suricata IDS with Enhanced kNN Algorithm on Raspberry Pi for Classifying IoT Botnet Attacks

Irfan Syamsuddin [1,*] and Omar Mohammed Barukab [2]

[1] Department of Computer and Networking Engineering, State Polytechnic of Ujung Pandang, Makassar 90245, Indonesia

[2] Faculty of Computing and Information Technology-Rabigh, King Abdulaziz University, Jeddah 21911, Saudi Arabia; obarukab@kau.edu.sa

* Correspondence: irfans@poliupg.ac.id

Abstract: The focus of this research is the application of the k-Nearest Neighbor algorithm in terms of classifying botnet attacks in the IoT environment. The kNN algorithm has several advantages in classification tasks, such as simplicity, effectiveness, and robustness. However, it does not perform well in handling large datasets such as the Bot-IoT dataset, which represents a huge amount of data about botnet attacks on IoT networks. Therefore, improving the kNN performance in classifying IoT botnet attacks is the main concern in this study by applying several feature selection techniques. The whole research process was conducted in the Rapidminer environment using three prebuilt feature selection techniques, namely, Information Gain, Forward Selection, and Backward Elimination. After comparing accuracy, precision, recall, F1 score and processing time, the combination of the kNN algorithm and the Forward Selection technique (kNN-FS) achieves the best results among others, with the highest level of accuracy and the fastest execution time among others. Finally, kNN-FS is used in developing SUKRY, which stands for Suricata IDS with Enhanced kNN Algorithm on Raspberry Pi.

Keywords: botnet attack; IoT; machine learning; kNN; feature selection; Suricata; Raspberry Pi

Citation: Syamsuddin, I.; Barukab, O.M. SUKRY: Suricata IDS with Enhanced kNN Algorithm on Raspberry Pi for Classifying IoT Botnet Attacks. *Electronics* **2022**, *11*, 737. https://doi.org/10.3390/electronics11050737

Academic Editors: Piyush Kumar Shukla, Manoj Kumar, Xiaochun Cheng and Prashant Kumar Shukla

Received: 3 February 2022
Accepted: 24 February 2022
Published: 27 February 2022

1. Introduction

The growing use of IoT in many areas of life, such as Smart Cities, has resulted in the creation of a massive complicated network of linked things (apps, devices, and people) [1]. It is estimated that 50% of the world's needs will greatly depend on IoT-based services in various sectors in the near future [2,3]. Along with such dependence, the threat of cyber attacks on the IoT networks is increasing and becoming more damaging [4]. Among the most vulnerable security risks in the IoT networks are man-in-the-middle attacks, eavesdropping, and malware such as botnets. Malware, or malicious software, is a specific cyberattack that has continued to develop since the era of traditional Internet networks and has become a serious threat in today's IoT environment. Botnets are one of several types of malware commonly used by criminals to carry out cyberattacks on IoT networks [5]. A botnet is basically a group of computers running malware controlled by hackers (usually called "botmasters"). Botnets turn computers into cyberattack forces, usually for spam, fake websites, DoS (Denial of Service) attacks, viruses, and information gathering through phishing and fraud [6]. Botnets themselves have undergone a significant evolution, making them more diverse and more difficult to detect. Traditional intrusion detection systems (IDS) capabilities are now lagging and less effective in detecting this new version of botnet attacks [7].

Intrusion detection systems (IDS) generally use signature-based and anomaly-based detections to detect cyber threats. As a result, signature-based detections are becoming increasingly rare. Predefined attack signatures are used to create signature-based detection systems. As a result, this technique is incapable of detecting novel attack types or variants of known attacks [8]. Anomaly detection systems, on the other hand, have a significant false-positive rate, despite being better at detecting novel types of cyber threats [7,8]. Furthermore, creating an anomaly detection system is complicated by the presence of so many distinct types of IoT devices. As a result, several studies have concluded that IDS is only effective against existing botnets and fails to detect and block new and distinct botnet versions [9]. Machine learning (ML) was proposed as a way to close the gap [10,11] since it improves the effectiveness and intelligence of IDS.

There are two main problems identified from previous research. First, kNN usually shows a lack of performance in dealing with large datasets in the IoT cyberattack domain. Second, there have been very few studies involving real IDS implementation in Raspberry Pi for IoT networks.

Therefore, we would like to propose developing a low-cost and lightweight intelligent intrusion detection system based on the Raspberry Pi machine. Then, for machine learning, the kNN algorithm is chosen considering its simplicity, effectiveness, and robustness, which makes it appropriate for use on resource-constrained devices, such as the Raspberry Pi.

To tackle the low-performance issue, this study aims to apply and evaluate several feature selection techniques in combination with the kNN algorithm to find the best-performing one using Rapidminer software. Finally, the best among them will be selected to establish a novel Suricata with an enhanced kNN algorithm on the Raspberry Pi for a real implementation called SUKRY.

The main contributions of this research are as follows:

- SUKRY, a novel Suricata IDS with an enhanced kNN algorithm on the Raspberry Pi machine.
- The use of five evaluation factors (accuracy, precision, recall, F1 score and execution time) to justify the best among three feature selection techniques in enhancing the performance of the kNN algorithm is novel.
- The application of kNN-FS (a combination between the kNN algorithm and the Forward Selection technique) as the core machine learning model in developing SUKRY.

The rest of the paper is organized as follows. Section 2 discusses the relevant literature concerning machine learning applications in dealing with IoT security issues. Section 3 describes the methodology used in conducting the study. Section 4 presents the data analysis, while the discussion is provided in Section 5. Eventually, the conclusion is given in the last section.

2. Relevant Studies

Research by Liao et al. [12] in 2002 was one of the first studies to consider potential machine learning algorithms to enhance intrusion detection theoretically. In this study, they justified the effectiveness of the kNN algorithm by having simplicity, robustness, and ease of use. Then, Binkley and Singh [13] also introduced several machine learning algorithms in dealing with botnet attack identification. Traditional signature-based IDS was questioned in this study in terms of its usefulness in recognizing new botnet variations, and it was suggested that machine learning be used to address the concerns. Later, many machine learning algorithms were adopted and exemplified in several studies. For example, Support Vector Machine in anomaly traffic detection, which showed an improvement in botnet identification rate [14], automating DDoS attack identification with several machine learning algorithms [15], DDoS attack mitigation ability after using machine learning approaches [16], the use of machine learning techniques to classify flooding network attacks on networks [17], and many others.

Eslahi et al. [18] constructed a relevant survey to address the emergent impact of botnets by examining their characteristics, the state of the art of machine learning-based detection, and future challenges in deploying secure IoT environments. A further review was also carried out by Simkhada et al. [19] to compare the advantages and disadvantages of various machine learning methods in botnet classification. Rashid et al. [20] discussed various cyberattacks on IoT-based Smart Cities by exploring attack and anomaly detection techniques based on machine learning algorithms (LR, SVM, DT, RF, ANN, and kNN). Moreover, Dwibedi et al. [21] focused their comparative study on different datasets by employing ML algorithms such as Random Forest (RF), Support Vector Machines (SVMs), Keras Deep Learning models, and XGBoost. They also suggested making the best model suitable for real-time environments. Then, advanced machine learning models were surveyed and evaluated by Pacheco and Sun [22] against several IDS datasets, such as UNSWNB15 and Bot-IoT. They demonstrated how several attacks were able to effectively degrade the overall performance of SVM, DT, and RF using two IDS datasets. They found RF was shown as the most resilient classifier, while SVM was the least robust on both datasets.

Aswal et al. [23] conducted another interesting investigation on botnet identification on the Internet of Vehicles (IoV). To deal with the IoV attack classification problem, many machine learning algorithms (NB, kNN, LR, LR, CART, and SVM) were chosen. In terms of TPR and FPR, it was discovered that the CART method outperformed other machine learning algorithms. Similarly, Hasan et al. [24] did a comparative investigation. Several machine learning algorithms (LR, SVM, DT, RF, and ANN) were examined for their ability to effectively anticipate attacks and abnormalities on IoT devices. It was demonstrated that RF outperforms other technologies. This conclusion was comparable to that of Bedi et al. [25], who investigated irregularities on an IoT network by comparing ANN, LR, RF, SVM, and DT. They also found RF performed better than the others.

A big data framework was discussed in [26] to detect peer-to-peer botnet attacks by employing a Random Forest algorithm. Furthermore, several approaches applied different algorithms in identifying and tackling various botnet attacks, such as parallel random forest by Chen et al. [27]. In addition, a comparison performance of three algorithms (NB, kNN and DT) for mobile botnet examined by Yusof et al. [28], a parallel multiclassification algorithm for big IoT data analysis by Duan et al. [29], and research by Vengatesan et al. [30] in predicting malware attacks in IoT by using big data analytics. Furthermore, Marjani, et al. [31] presents a comprehensive survey in this field by highlighting its architecture, various approaches and future research challenges.

Gadelrab et al. [32] used a machine learning strategy based on statistical features to detect botnets. Later, Hoang and Nguyen [33] created a botnet detection model based on machine learning and validated its performance using popular machine learning techniques using the Domain Name Service. The results of the experiments demonstrated that the machine learning algorithm can be used to detect botnets successfully, with the Random Forest method producing the best detection accuracy among the others.

The hybrid technique is helpful in this field. A hybrid is a method of combining different machine learning techniques to improve botnet detection accuracy. By merging the C5 and SVM algorithms, Khraisat et al. [34] developed a novel technique to detect botnet attacks, which resulted in greater accuracy for the hybrid approach. Almashhadi [35] shows that utilizing hybrid machine learning to analyze DNS traffic irregularities during the botnet lifetime improves accuracy. Similarly, Khan et al. [36] used a hybrid technique to improve botnet attack similarity identification. They used experiments to prove the effectiveness of their strategy. However, the hybrid approach requires high computational resources.

Rambabu and Venkatram [37] proposed an ensemble classification using traffic flow metrics for DDoS attacks in IoT networks. Using cross-validation, they addressed the importance of the ensemble approach towards DDoS defense accuracy with fewer false alarms. Another ensemble approach was studied by Khraisat et al. [38] to protect IoT devices from cyberattacks. They combined the C5 classifier and one class of support vector machine classifier to develop a novel ensemble Hybrid Intrusion Detection System (HIDS). Likewise, an AdaBoost ensemble learning method based on DT, NB, and ANN was introduced by Moustafa et al. [39] to evaluate and detect malicious events using UNSW-NB15 and NIMS botnet datasets.

In addition, the use of machine learning frameworks is exemplified in several studies. A Weka-based comparative study of various machine learning algorithms was explored by Farhat et al. [40]. They used the NSL-KDD dataset for comparison purposes. Likewise, the use of the Weka tool was confirmed as efficient by Celil and Dener [41] in classifying and detecting anomalous activities within IoT networks. A similar study by Soe et al. [42] implemented machine learning-based IDS using a new feature selection algorithm. The whole machine learning analysis with several algorithms was conducted within the Weka environment. Then, they suggested evaluating other machine learning algorithms with lower computational requirements and relatively simple implementations.

Based on the aforementioned studies, the research gap is identified as follows. Among many machine learning algorithms applied in dealing with cyberattacks on IoT networks, kNN often shows lower performance in comparison to other machine learning algorithms. This is due to the fact that most datasets relating to botnet attacks in the IoT networks are usually large in size causing kNN to require more computational cost and extra processing time.

In this study, we are interested in kNN, a supervised machine learning algorithm that is simple to implement, robust and with lower computational requirements applicable for Raspberry Pi implementation. Although in 2002 it was already suggested in [12], very few studies in terms of IoT-related botnet attacks have shown the best results for kNN accuracy. Churcher and Ullah [43] found kNN as a suitable model to be used for intrusion detection, thanks to its simplicity and robustness. Similarly, Mrabet and Belguith [44] argued kNN's important advantage is simplicity and ease of application, however, kNN is incapable of handling large datasets. The high similarity between the closest and farthest neighbors on the basis of distance functions and weights is the main cause for kNN's poor performance when dealing with large datasets, according to Wazirali [45].

Therefore, we aim to enhance kNNs performance, by proposing the implementation of feature selection techniques. Instead of Weka [40–42], Rapidminer is selected as the environment to conduct the analysis since Rapidminer offers several in-built feature selection techniques and other visual functions [46–48]. Moreover, Lee et al. [49] reported the effective use of in-built feature selection techniques in the Rapidminer environment to handle high dimensionality datasets. The next step in our research is to turn the enhanced kNN model into an actual IDS application that uses a Raspberry Pi engine for intelligent botnet detection.

3. Methodology

This section shows stages to conduct the research in a systematic and directed manner which can be seen in Figure 1. It starts with dataset selection, followed by data preprocessing, then combining particular feature selection techniques with kNN to train and test the dataset, and finally compares them in terms of accuracy and speed of execution time.

3.1. Dataset

The raw data used are from a dataset from the Bot-IoT dataset [50] as presented in Table 1, which is grouped into five classes, namely DDoS, Reconnaissance, DoS, Normal and Theft. The reason to choose this dataset is that it represents a realistic IoT environment [51,52] and also a balanced class between normal and abnormal classes [53,54]. The size of the dataset is approximately 0.78 GB and it has about three million records. About 70% of the dataset is used for training, while the rest is for model testing.

3.2. Dataset Preprocessing

As the dataset is well prepared as mentioned in earlier studies [50–54], it undergoes relatively simpler preprocessing stages, such as removing noises, missing values, tacking inconsistent or redundant data as well as data cleaning. Then, a role is assigned to the target attribute as a label. The process of setting the target attribute as a label is depicted in the Figure 2.

3.3. Feature Selection Techniques and kNN Algorithm

In this step, we apply three built-in feature selection techniques provided in the Rapidminer environment, namely Information Gain (IG), Forward Selection (FS) and

Backward Elimination (BE). Each of the three feature selection techniques will be combined with the k-Nearest Neighbor (kNN) algorithm.

Information gain is a feature selection technique based on weight calculation. Information Gain compares attribute weights to class attributes to calculate attribute weights. The higher an attribute's weight, the more important it is thought to be. At the same time, other low-weight features are eliminated [46].

Forward Selection measures each feature individually, then continues the process by adding and measuring new features until the best combination of features is found. This technique starts with an empty attribute and then adds each new attribute for comparison in each cycle. Later, only the attributes that provide the highest performance improvement are included in the selected features. The low ones will be removed [49].

Backward Elimination is a feature selection strategy that begins with the entire set of characteristics and removes each remaining attribute in each round. The performance of each eliminated attribute is assessed using the inner operators, and only the attribute with the smallest loss in performance or that has no meaningful effect on the dependent variable or output prediction is removed from the selection. After removing the least important features, only those with the best performance remain [46].

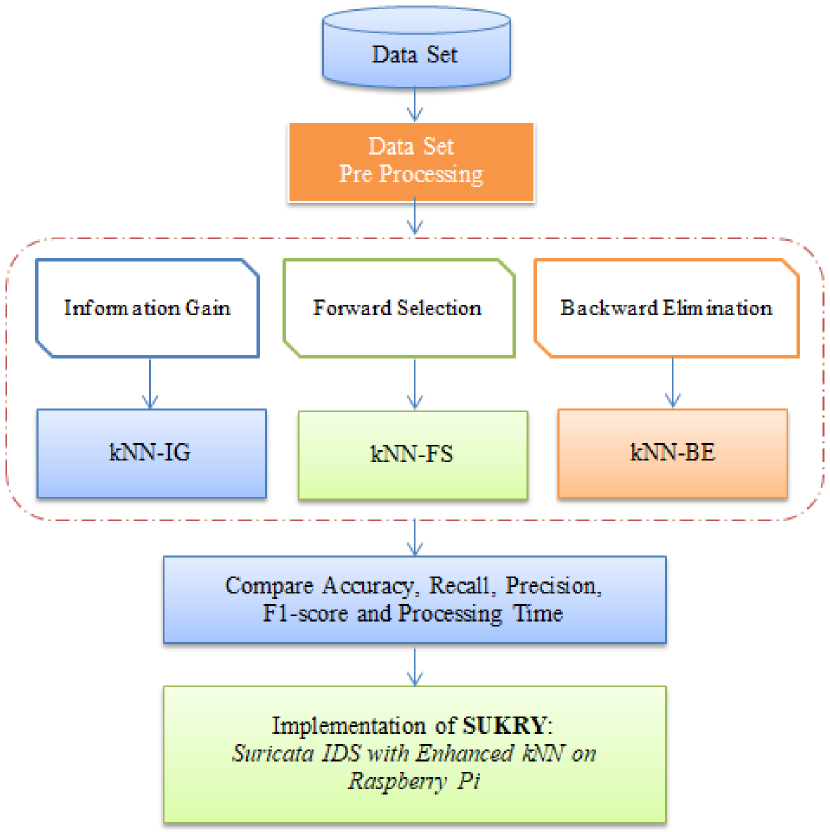

Data Set
Data Set Pre Processing
Information Gain
Forward Selection
Backward Elimination
kNN-IG
kNN-FS
kNN-BE
Compare Accuracy, Recall, Precision, F1-score and Processing Time
Implementation of SUKRY: Suricata IDS with Enhanced kNN on Raspberry Pi

Figure 1. Research steps.

The kNN algorithm is a simple supervised machine learning approach that uses the majority of the k-Nearest Neighbor categories to classify the outcomes of a new instance. The purpose of this approach is to categorize new objects using the training set's properties and samples. The kNN method predicts new instance values and classifies new objects based on k of their nearest neighbors, where k is a predefined positive integer [43]. Its pseudo-code is presented in Figure 3.

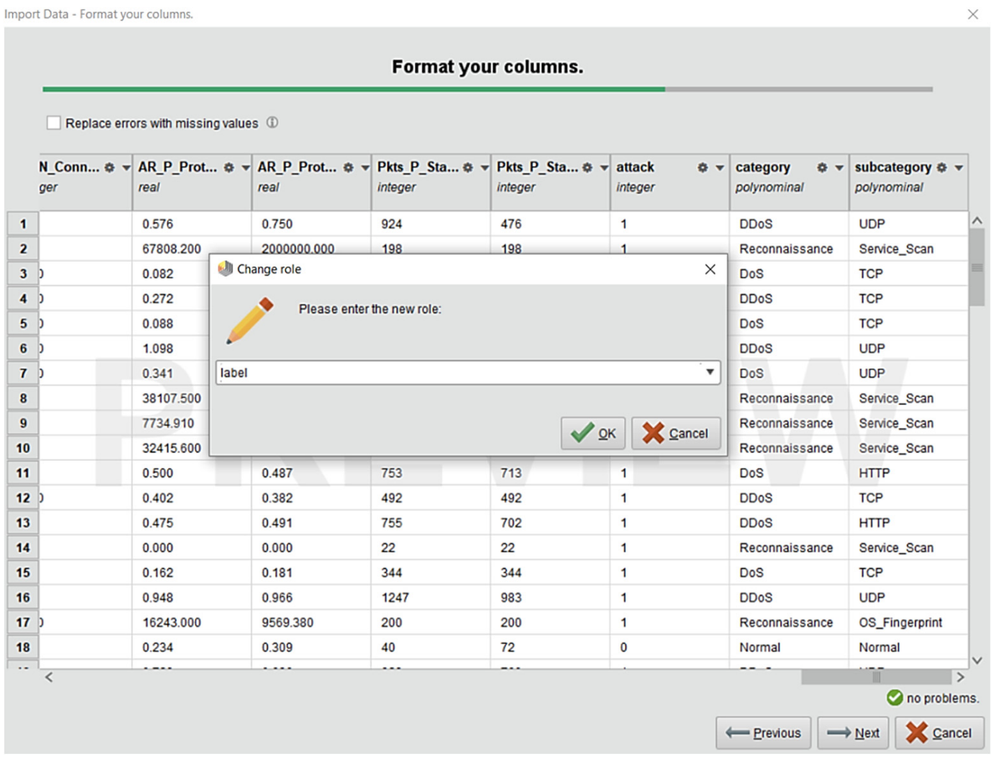
Table 1. Dataset and its features.

Features	Description
pkSeqID	Row Identifier
Stime	Record start time
Flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
Saddr	Source IP address
Sport	Source port number
daddr	Destination IP address
port	Destination port number
Pkts	Total count of packets in transactions
Bytes	Total number of bytes in transactions
state	Transaction state
state_number	Numerical representation of feature state
Ltime	Last time record
Seq	Argus sequence number
Dur	Record total duration
mean	Average duration of aggregated records
Stddev	Standard deviation of aggregated records
Sum	Total duration of aggregated records
Min	Minimum duration of aggregated records
Max	Maximum duration of aggregated records
Spkts	Source-to-destination packet count
Dpkts	Destination-to-source packet count
Sbytes	Source-to-destination byte count
Dbytes	Destination-to-source byte count
Rate	Total packets per second in transaction
Srate	Source-to-destination packets per second
Drate	Destination-to-source packets per second
TnBPSrcIP	Total Number of bytes per source IP
TnBPDstIP	Total Number of bytes per Destination IP.
TnP_PSrcIP	Total Number of packets per source IP.
TnP_PDstIP	Total Number of packets per Destination IP.
TnP_PerProto	Total Number of packets per protocol.
TnP_Per_Dport	Total Number of packets per port
AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP. (calculated by pkts/dur)
AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP.
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.
AR_P_Proto_P_Sport	Average rate per protocol per sport
AR_P_Proto_P_Dport	Average rate per protocol per port
Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP.
Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP.
Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Category	Traffic categories
Subcategory	Traffic subcategory

The combination of each feature selection technique with the kNN algorithm is called, kNN-IG, kNN-FS and kNN-BE. Their performances will be compared in the following section.

3.4. Performance Analysis

There are five factors that will be taken into account to determine whether kNN-IG, kNN-FS and kNN-BE show increasing performance or not. The factors are accuracy, precision, recall, F1 score and processing time. The highest level of the four factors (accuracy, precision, recall, F1 score) and the shortest processing time determine the best feature selection technique to improve the performance of kNN classifying botnet attacks based on the given dataset.


Import Data - Format your columns.

Format your columns.

☐ Replace errors with missing values

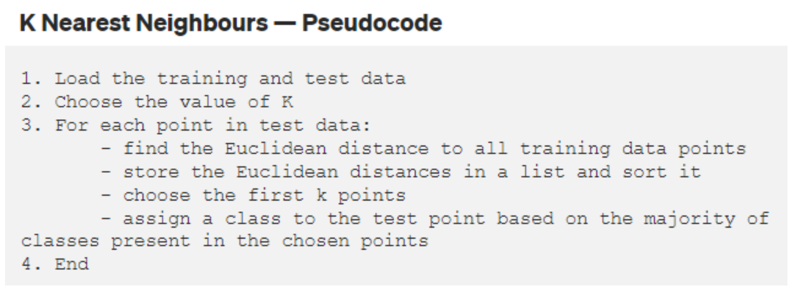
	N_Conn... ger	AR_P_Prot... real	AR_P_Prot... real	Pkts_P_Sta... integer	Pkts_P_Sta... integer	attack integer	category polynominal	subcategory polynominal
1		0.576	0.750	924	476	1	DDoS	UDP
2		67808.200	2000000.000	198	198	1	Reconnaissance	Service_Scan
3	)	0.082					DoS	TCP
4	)	0.272					DDoS	TCP
5	)	0.088					DoS	TCP
6	)	1.098					DDoS	UDP
7	)	0.341					DoS	UDP
8		38107.500					Reconnaissance	Service_Scan
9		7734.910					Reconnaissance	Service_Scan
10		32415.600					Reconnaissance	Service_Scan
11		0.500	0.487	753	713	1	DoS	HTTP
12	)	0.402	0.382	492	492	1	DDoS	TCP
13		0.475	0.491	755	702	1	DDoS	HTTP
14		0.000	0.000	22	22	1	Reconnaissance	Service_Scan
15		0.162	0.181	344	344	1	DoS	TCP
16		0.948	0.966	1247	983	1	DDoS	UDP
17	)	16243.000	9569.380	200	200	1	Reconnaissance	OS_Fingerprint
18		0.234	0.309	40	72	0	Normal	Normal

Change role

Please enter the new role:

label

Figure 2. Data preprocessing in Rapidminer.


K Nearest Neighbours — Pseudocode

```
1. Load the training and test data
2. Choose the value of K
3. For each point in test data:
        - find the Euclidean distance to all training data points
        - store the Euclidean distances in a list and sort it
        - choose the first k points
        - assign a class to the test point based on the majority of
classes present in the chosen points
4. End
```

Figure 3. kNN algorithm.

In doing so, the confusion matrix plays an important role. The confusion matrix is a method used to describe whether or not the classifier recognizes tuples from different classes. The confusion matrix is illustrated by a table that shows the total data that are correctly categorized and the total data that are incorrectly categorized. In this study, the confusion matrix with five labels is shown in Table 2.

Table 2. Confusion matrix with five labels.

	A	**B**	**C**	**D**	**E**	**Precision**
A = DdoS	X11	X12	X13	X14	X15	P1
B = Recons	X21	X22	X23	X24	X25	P2
C = DoS	X31	X32	X33	X34	X35	P3
D = Normal	X41	X42	X43	X44	X45	P4
E = Theft	X51	X52	X53	X54	X55	P5
Recall	R1	R2	R3	R4	R5	

Several measurements could be obtained from the confusion matrix table, such as accuracy, recall, precision and F1 score. The formula of each is described as follows:

$$\text{Accuracy} = \frac{\text{X11} + \text{X22} + \text{X33} + \text{X44} + \text{X55}}{\text{Total Data}} \tag{1}$$

$$\text{Recall} = \frac{\text{X11}}{\text{X11} + \text{X21} + \text{X31} + \text{X41} + \text{X51}} \tag{2}$$

Then the total recall is:

$$\overline{\text{x}}\text{Recall} = \frac{\text{R1} + \text{R2} + \text{R3} + \text{R4} + \text{R5}}{5} \tag{3}$$

$$\text{Precision} = \frac{\text{X11}}{\text{X11} + \text{X12} + \text{X13} + \text{X14} + \text{X15}} \tag{4}$$

Then total precision is:

$$\overline{\text{x}}\text{Precision} = \frac{\text{P1} + \text{P2} + \text{P3} + \text{P4} + \text{P5}}{5} \tag{5}$$

$$\text{F1 score} = \frac{2 * \text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})} \tag{6}$$

In terms of execution time, Rapidminer provides records to show how long a process is being conducted by kNN with each feature selection technique. Using the given execution time data from Rapidminer, the performance of each combination of kNN could be obtained for comparison. The best-performing feature selection techniques with the kNN algorithm will be selected at this stage.

3.5. SUKRY Implementation

The last step is to implement the selected machine learning model into Suricata IDS on the Raspberry Pi machine. This approach is called SUKRY, a novel Suricata IDS with an enhanced kNN algorithm on a Raspberry Pi machine.

4. Analysis and Discussion

This section presents an in-depth analysis of how three different feature selection techniques, Information Gain, Forward Selection and Backward Elimination, would affect the performance of kNN. The performance analysis is based on five factors: accuracy, precision, recall, F1 score and processing time. It is divided into three sections which are called kNN-IG, kNN-FS and kNN-BE as follows.

4.1. kNN Algorithm with Information Gain (kNN-IG)

Information Gain (IG) is the first feature selection technique applied to the dataset in order to measure the most influential features among existing features of the Botnet-IoT dataset. IG feature selection is simply called the Filter method in Rapidminer and Figure 4 represents the process of IG implementation on the dataset in the Rapidminer environment.

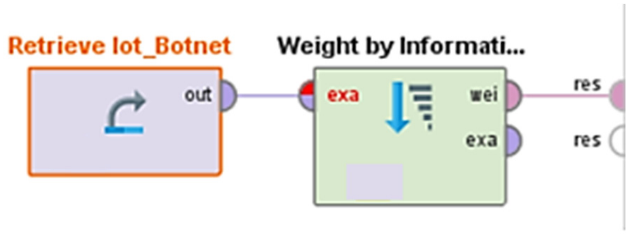


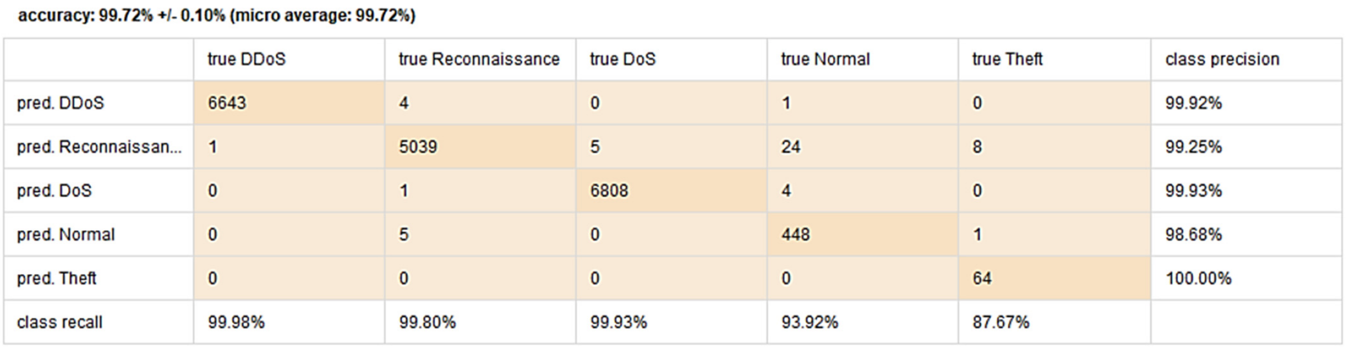
Figure 4. Information Gain.

Table 3 shows the number of features selected by performing the IG feature selection in Rapidminer.

Table 3. Reduced features by Information Gain.

Original Number of Features	Number of Features after IG
46	23

Once selected features are obtained, the kNN algorithm proceeds to the machine learning analysis, which is called kNN-IG. Figure 5 shows the results of kNN-IG in the confusion matrix.



accuracy: 99.72% +/- 0.10% (micro average: 99.72%)

	true DDoS	true Reconnaissance	true DoS	true Normal	true Theft	class precision
pred. DDoS	6643	4	0	1	0	99.92%
pred. Reconnaissan...	1	5039	5	24	8	99.25%
pred. DoS	0	1	6808	4	0	99.93%
pred. Normal	0	5	0	448	1	98.68%
pred. Theft	0	0	0	0	64	100.00%
class recall	99.98%	99.80%	99.93%	93.92%	87.67%	

Figure 5. Confusion matrix of kNN-IG.

The confusion matrix presents how many data are correctly and incorrectly predicted in each class by kNN-IG. In this case, the DDoS class has 6643 pieces of data that are predicted correctly, only one piece of data was incorrectly predicted. The Reconnaissance class has 5039 pieces of data correctly predicted while 10 others are incorrect. Similarly, the DoS class has 6808 pieces of data correctly predicted and five data with an incorrect class. The normal class has 448 pieces of data predicted correctly and 29 pieces of data incorrectly predicted. Finally, there are 64 pieces of data correctly predicted for the Theft class, while seven pieces of data with an incorrect class.

Based on these, the performance of kNN-IG could be determined by calculating scores of accuracy, recall, precision and F1 score. The calculation processes are shown below.

$$\text{Accuracy} = \frac{\text{X11} + \text{X22} + \text{X33} + \text{X44} + \text{X55}}{\text{Total Data}} = \frac{6643 + 5039 + 6808 + 448 + 64}{19056} = \frac{19002}{19056} = 99.72\%$$

Then, we calculate the individual recall score of each class. For example, the following is the recall score for the DDoS class:

$$\text{Recall(DDoS)} = \frac{\text{X11}}{\text{X11} + \text{X21} + \text{X31} + \text{X41} + \text{X51}} = \frac{6643}{6643 + 1 + 0 + 0 + 0} = \frac{6643}{6644} = 99.98\%$$

By using the same calculation, all classes obtain their recall scores as follows, DDoS (99.98%), Reconnaissance (99.80%), DoS (99.93%), Normal (93.92%) and Theft (87.67%).

Then the average recall score is obtained as follows:

$$\bar{\text{x}}\text{Recall} = \frac{\text{R1} + \text{R2} + \text{R3} + \text{R4} + \text{R5}}{5} = \frac{99.98 + 99.80 + 99.93 + 93.92 + 87.67}{5} = 96.26\%$$

Next, we calculate the individual precision score of each class. For example, the following is the precision score for the DDoS class:

$$\text{Precision(DDoS)} = \frac{\text{X11}}{\text{X11} + \text{X12} + \text{X13} + \text{X14} + \text{X15}} = \frac{6643}{6643 + 4 + 0 + 1 + 0} = \frac{6643}{6648} = 99.92\%$$

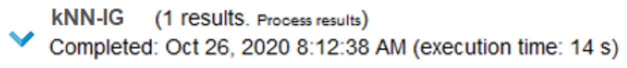
Then, by using the same calculation, all classes obtained their precision scores as follows, DdoS (99.92%), Reconnaissance (99.25%), DoS (99.93%), Normal (98.68%) and Theft (100%). Then the average precision score is obtained as follows:

$$\bar{x}\text{Precision} = \frac{\text{P1} + \text{P2} + \text{P3} + \text{P4} + \text{P5}}{5} = \frac{99.92 + 99.25 + 99.93 + 98.69 + 100}{5} = 99.56\%$$

Next, based on recall and precision scores, the F1 score is calculated as follows:

$$\text{F1 score} = \frac{2*\text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})} = \frac{2 * 97.82 * 99.77}{97.82 + 99.77} = 97.88\%$$

Finally, Rapidminer records the processing time of kNN-IG. Figure 6 shows an execution time of kNN-IG of 14 s.



kNN-IG (1 results. Process results)
Completed: Oct 26, 2020 8:12:38 AM (execution time: 14 s)

Figure 6. Execution time for kNN-IG.

4.2. kNN Algorithm with Forward Selection (kNN-FS)

Forward Selection (FS) is the second feature selection technique applied to the dataset in order to measure the most influential features among existing features of the Botnet-IoT dataset. FS feature selection is classified as the Wrapper method in Rapidminer. Figure 7 represents the process of FS implementation on the dataset in the Rapidminer environment.

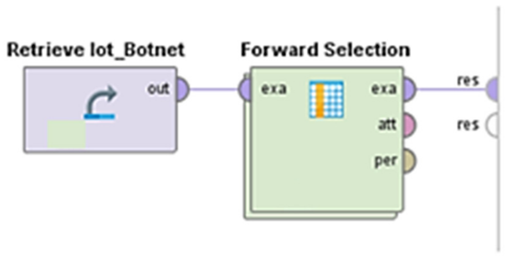


Figure 7. Forward Selection.

After the selection process is carried out, the number of the selected features by the Forward Selection technique is decreased, as can be seen in Table 4.

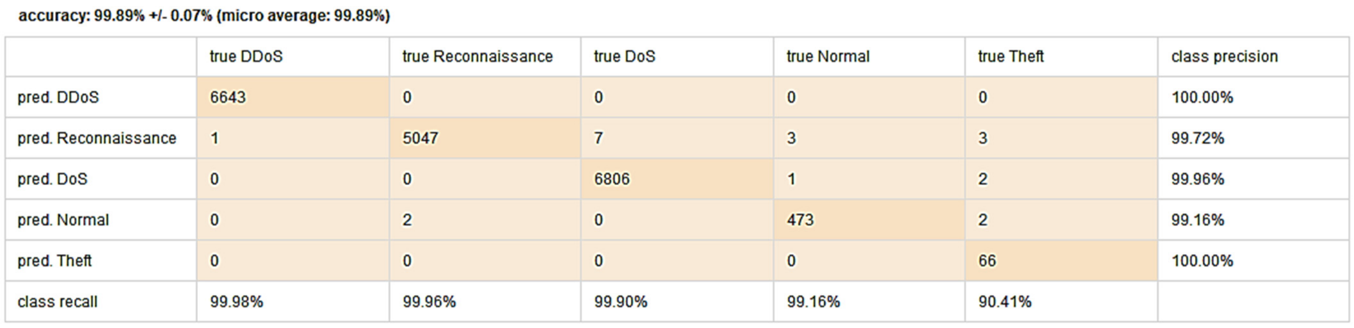
Table 4. Reduced features by Forward Selection.

Original Number of Features	Number of Features after FS
46	8

Then the kNN algorithm is applied to this new dataset, this process is named kNN-FS. Figure 8 shows the results of kNN-FS implementation on the dataset in the confusion matrix.

As can be seen in the confusion matrix of kNN-FS, the DDoS class has 6643 pieces of data that are predicted correctly, with only one piece of data incorrectly predicted. The Reconnaissance class has 5047 pieces of data correctly predicted while two others had a different class.

Similarly, the DoS class has 6808 pieces of data correctly predicted and seven pieces of data with an incorrect class. Then, there are 473 pieces of data predicted correctly in the Normal class while four other pieces of data were incorrectly predicted. Finally, the Theft class has 66 correct pieces of data and seven pieces of data with an incorrect class. Based on these, the performance of the kNN-FS could be determined by calculating scores of accuracy, recall, precision and F1. The calculation processes are shown below.



accuracy: 99.89% +/- 0.07% (micro average: 99.89%)

	true DDoS	true Reconnaissance	true DoS	true Normal	true Theft	class precision
pred. DDoS	6643	0	0	0	0	100.00%
pred. Reconnaissance	1	5047	7	3	3	99.72%
pred. DoS	0	0	6806	1	2	99.96%
pred. Normal	0	2	0	473	2	99.16%
pred. Theft	0	0	0	0	66	100.00%
class recall	99.98%	99.96%	99.90%	99.16%	90.41%	

Figure 8. Confusion matrix of kNN-FS.

$$\text{Accuracy} = \frac{\text{X11} + \text{X22} + \text{X33} + \text{X44} + \text{X55}}{\text{Total Data}} = \frac{6643 + 5047 + 6806 + 473 + 66}{19056} = \frac{19035}{19056} = 99.89\%$$

Furthermore, we calculate individual recall scores for each class. For example, the following is the recall score for the DDoS class:

$$\text{Recall}(\text{DDoS}) = \frac{\text{X11}}{\text{X11} + \text{X21} + \text{X31} + \text{X41} + \text{X51}} = \frac{6643}{6643 + 1 + 0 + 0 + 0} = \frac{6643}{6644} = 99.98\%$$

Using the same process, we obtain recall scores for all classes as follows, DDoS (99.98%), Reconnaissance (99.96%), DoS (99.90%), Normal (99.16%) and Theft (90.14%). Then the average recall is obtained as follows:

$$\bar{\text{x}}\text{Recall} = \frac{\text{R1} + \text{R2} + \text{R3} + \text{R4} + \text{R5}}{5} = \frac{99.98 + 99.96 + 99.90 + 99.16 + 90.14}{5} = 97.82\%$$

Next, we calculate an individual precision score for each class. For example, the following is the precision score for the DDoS class:

$$\text{Precision}(\text{DDoS}) = \frac{\text{X11}}{\text{X11} + \text{X12} + \text{X13} + \text{X14} + \text{X15}} = \frac{6643}{6643 + 0 + 0 + 0 + 0} = \frac{6643}{6643} = 100\%$$

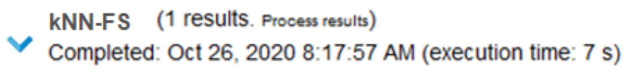
Using the same formula, scores are obtained as follows, DDoS (100%), Reconnaissance (99.72%), DoS (99.96%), Normal (99.16%) and Theft (100%). Then the average precision score is obtained as follows:

$$\bar{\text{x}}\text{Precision} = \frac{\text{P1} + \text{P2} + \text{P3} + \text{P4} + \text{P5}}{5} = \frac{100 + 99.72 + 99.96 + 99.16 + 100}{5} = 99.77\%$$

Next, based on recall and precision scores, the F1 score is obtained as follows:

$$\text{F1 score} = \frac{2{*}\text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})} = \frac{2 * 97.82 * 99.77}{97.82 + 99.77} = 98.78\%$$

Finally, Rapidminer logs the processing time of kNN-FS. Figure 9 shows an execution time by kNN-FS of 7 s.



kNN-FS (1 results. Process results)
Completed: Oct 26, 2020 8:17:57 AM (execution time: 7 s)

Figure 9. Execution time for kNN-FS.

4.3. kNN Algorithm with Backward Elimination (kNN-BE)

Backward Elimination (BE) is the last feature selection technique applied to the dataset in order to choose the most significant features among the existing ones in the Botnet-IoT dataset. The BE feature selection is also classified as the Wrapper method in Rapidminer.

Figure 10 represents the process of BE implementation on the dataset in the Rapidminer environment.

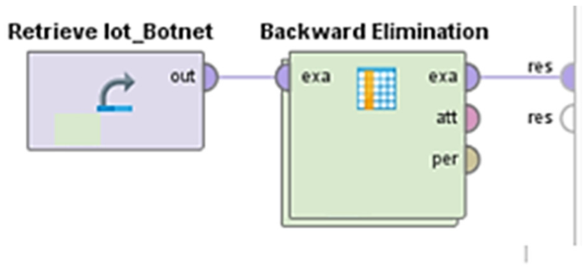


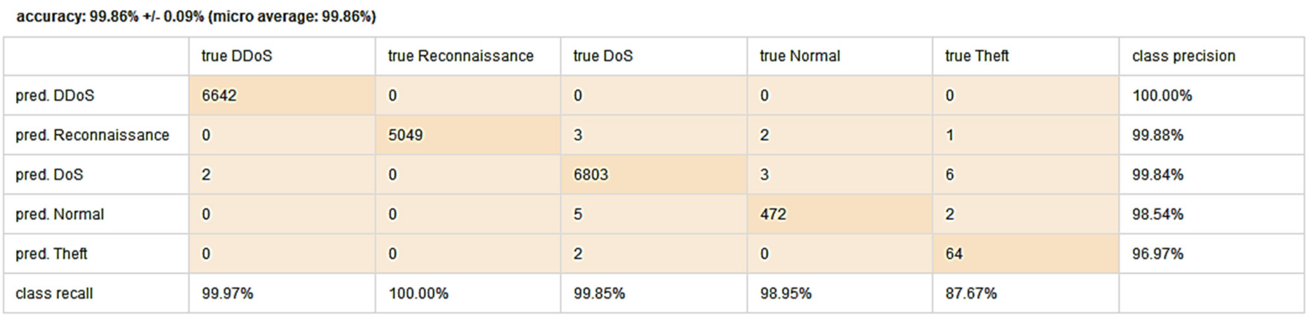
Figure 10. Backward Elimination.

This step reduced the number of features in the dataset according to the Backward Elimination technique, which can be seen in Table 5.

Table 5. Reduced features by Backward Elimination.

Original Number of Features	Number of Features after BE
46	35

Based on the result of Backward Elimination above, we continue applying the kNN algorithm to it, and this process is named kNN-BE. Figure 11 shows the results of kNN-BE in the confusion matrix.



accuracy: 99.86% +/- 0.09% (micro average: 99.86%)

	true DDoS	true Reconnaissance	true DoS	true Normal	true Theft	class precision
pred. DDoS	6642	0	0	0	0	100.00%
pred. Reconnaissance	0	5049	3	2	1	99.88%
pred. DoS	2	0	6803	3	6	99.84%
pred. Normal	0	0	5	472	2	98.54%
pred. Theft	0	0	2	0	64	96.97%
class recall	99.97%	100.00%	99.85%	98.95%	87.67%	

Figure 11. Confusion matrix of kNN-BE.

According to the kNN-BE confusion matrix, the DDoS class has 6642 pieces of data that are correctly predicted, while two pieces of data were incorrectly predicted. Then, the Reconnaissance class has all 5049 pieces of data correctly predicted. For the DoS class, there are 6803 correctly predicted pieces of data, whereas 10 pieces of data were incorrectly classed. Then, 472 pieces of data were predicted correctly for the Normal class, while five pieces of data were incorrectly classed. Finally, the Theft class has 64 pieces of data that are predicted correctly and nine pieces of data that were incorrect.

In order to measure the performance of kNN-BE, several scores should be obtained, namely, accuracy, recall, precision and F1 score, as well as execution time. The following is the calculation to obtain the accuracy for kNN-BE.

$$\text{Accuracy} = \frac{\text{X11} + \text{X22} + \text{X33} + \text{X44} + \text{X55}}{\text{Total Data}} = \frac{6642 + 5049 + 6803 + 472 + 64}{19056} = \frac{19030}{19056} = 99.86\%$$

Next, we calculate the recall for DDoS class as follows:

$$\text{Recall(DDoS)} = \frac{\text{X11}}{\text{X11} + \text{X21} + \text{X31} + \text{X41} + \text{X51}} = \frac{6642}{6642 + 0 + 2 + 0 + 0} = \frac{6642}{6644} = 99.97\%$$

Using the same calculation we obtain the recall scores for all classes as follows, DDoS (99.97%), Reconnaissance (100%), DoS (99.85%), Normal (98.95%) and Theft (87.67%). Then, the average recall score is obtained as follows:

$$\bar{x}\text{Recall} = \frac{\text{R1} + \text{R2} + \text{R3} + \text{R4} + \text{R5}}{5} = \frac{99.97 + 100 + 99.85 + 98.95 + 87.67}{5} = 97.29\%$$

Next, we calculate an individual precision score for each class. For example, the following is the precision score for the DDoS class:

$$\text{Precision(DDoS)} = \frac{\text{X11}}{\text{X11} + \text{X12} + \text{X13} + \text{X14} + \text{X15}} = \frac{6643}{6643 + 0 + 0 + 0 + 0} = \frac{6643}{6643} = 100\%$$

Using a similar formula we obtain precision scores for all classes as follows, DDoS (100%), Reconnaissance (99.88%), DoS (99.84%), Normal (98.54%) and Theft (96.97%). Then, the average precision is obtained as follows:

$$\bar{x}\text{Precision} = \frac{\text{P1} + \text{P2} + \text{P3} + \text{P4} + \text{P5}}{5} = \frac{100 + 99.88 + 99.84 + 98.54 + 96.97}{5} = 99.05\%$$

Next, based on recall and precision scores, the F1 score is calculated as follows:

$$\text{F1 score} = \frac{2{*}\text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})} = \frac{2 * 97.29 * 99.05}{97.29 + 99.05} = 98.16\%$$

Finally, Rapidminer automatically records the processing time of kNN-BE. Figure 12 shows an execution time by kNN-BE of 18 s.

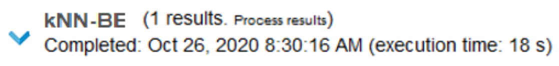

kNN-BE (1 results. Process results)
Completed: Oct 26, 2020 8:30:16 AM (execution time: 18 s)

Figure 12. Execution time for kNN-BE.

4.4. Performance Comparison

This section presents the whole performance comparison between kNN-IG, kNN-FS and kNN-BE obtained previously using Rapidminer, as presented in Table 6. It is clearly visualized in Figures 13 and 14. While Figure 13 depicts the level of accuracy, recall, precision, and F1 score, Figure 14 reveals execution time results.

Table 6. Overall results.

	Accuracy (%)	Recall (%)	Precision (%)	F1 Score (%)	Time (s)
kNN-IG	99.72	96.26	99.56	97.88	14
kNN-FS	99.89	97.82	99.77	98.78	7
kNN-BE	99.86	97.29	99.05	98.16	18

It is clearly seen that kNN-FS, which represents the Forward Selection-based kNN algorithm outperforms the other two models (kNN-IG and kNN-BE). The kNN-FS achieves the best results of all aspects, accuracy (99.89%), recall (97.82%), precision (99.77%), F1 score (98.78%) (see Figure 13) and also the fastest execution time of 7 s (see Figure 14).

The next position goes to the kNN-BE, which outperforms kNN-IG in three factors, namely, accuracy (99.86%), recall (96.29%), and F1 score (98.16%) while achieving the lowest results in precision (99.05%) and the longest execution time of 18 s. Finally, kNN-IG only shows better results than kNN-BE in two factors, F1 score (97.88%) and 14 s for execution time while accounting for the lowest results among others in accuracy (99.72%), recall (96.26) and precision (99.56%). It is found that the implementation of Forward Selection on the given dataset significantly improves the performance of the kNN algorithm in classifying botnet attacks in IoT networks.

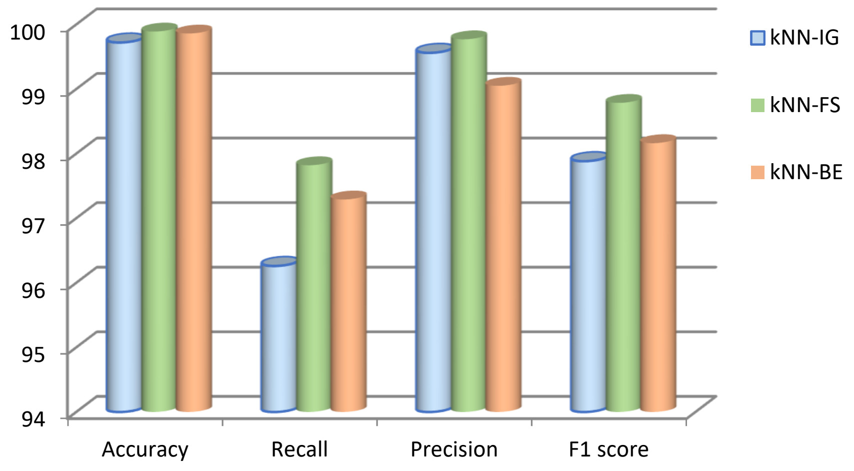


Figure 13. Level of accuracy, recall, precision and F1 score.

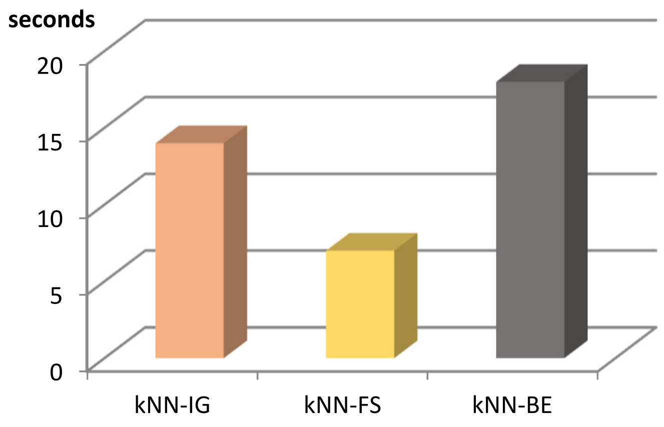


Figure 14. Level of execution time.

4.5. SUKRY Implementation

Based on our results, the kNN-FS model is selected for SUKRY implementation. This last section describes the implementation of SUKRY in the Raspberry Pi machine. The following steps describe how SUKRY is deployed:

- Obtain Raspberry Pi OS from https://www.raspberrypi.com/software/ (accessed on 20 December 2020) [55];
- Install Raspberry Pi OS on the Raspberry Pi machine;
- Obtain Suricata from https://suricata.io/download/ (accessed on 20 December 2020) [56];
- Install Suricata on Raspberry Pi OS;
- Obtain OPNIDS from https://github.com/OPNids (accessed on 20 December 2020) [57];
- Install OPNIDS over Suricata to enable machine learning models being implemented over Suricata;
- Running kNN-FS model using OPNIDS.

The deployment steps above are useful for future studies in reproducing the experiment [58]. In the near future, the hardware of SUKRY will be packed properly for commercialization purposes. The current deployment of SUKRY is presented in Figure 15.

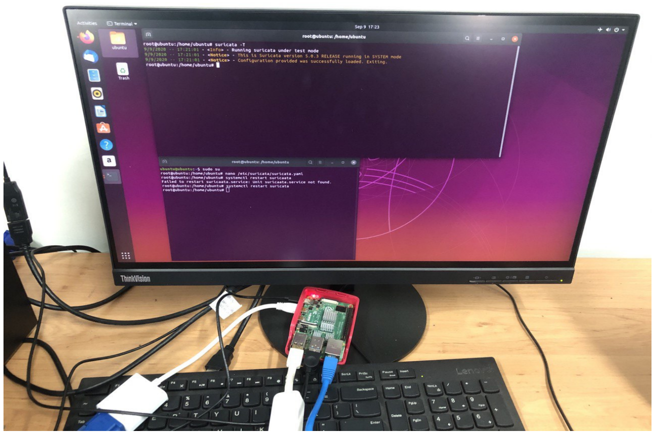


Figure 15. Deployment of SUKRY.

5. Discussion

Compared to previous studies, our study showed a significant improvement in kNN performance. First, recent research by Bahsi et al. [59] implemented a hybrid machine learning approach to detect botnets in IoT networks, which obtains the following results: DT (98.9%) and kNN (94.9%). Secondly, a novel EDIMA botnet detector by Kumar and Lim [60], EDIMA, was applied in IoT networks using three machine learning algorithms with an accuracy of RF (88.8%), kNN (94.44%), and GNB (77.78%). All of these comparisons are presented in Table 7.

Table 7. Accuracy comparison with previous studies.

Approach	Accuracy (%)			
	kNN	**DT**	**RF**	**GNB**
SUKRY	99.89	-	-	-
Bahsi et al. [42]	94.90	98.9	-	-
Kumar and Lim [43]	94.44	-	88.8	77.78

While previous related studies commonly focus on the level of accuracy in determining the best machine learning algorithm, we argue through this study that execution time should be considered an important factor as well. This is particularly true for crucial cases such as cyberattacks in IoT. Unfortunately, none of these previous studies [42,43] provided information about it. As a result, a performance comparison from an execution time perspective could not be presented. SUKRY's 7-min execution time, on the other hand, is a notable breakthrough for the application of the kNN algorithm.

In our view, both accuracy level and execution time factors must be considered equally in detecting any IoT-related cyberattacks. Otherwise, the whole IoT network and all vital services that depend on it will be seriously affected, such as Smart Cities, Smart Grid, and many others, since cyberattacks usually occur in a very short time.

SUKRY is a novel approach to establishing Suricata-based IDS using hybrid kNN and Forward Selection over Raspberry Pi. SUKRY's advantage lies in its high level of accuracy in detecting potential cyber threats in a relatively short time. We affirm that both the high accuracy and short execution time offered by SUKRY are essential factors in securing the IoT environment from botnet attacks more effectively and efficiently.

6. Conclusions

This study introduces SUKRY, a novel Suricata IDS using an enhanced kNN algorithm on the Raspberry Pi machine. The first issue to solve is the low-performance problems of the k-Nearest Network algorithm, particularly in dealing with large datasets such as the Botnet-IoT dataset. By using the Rapidminer environment, three feature selection techniques

(Information Gain, Forward Selection, and Backward Elimination) are selected to reduce the number of features in the dataset before being processed with the kNN algorithm. The three combinations called kNN-IG, kNN-FS, and kNN-BE are then evaluated and compared in terms of accuracy, precision, recall, F1 score, and processing time.

It is found that kNN-FS or the kNN algorithm with the Forward Selection technique accounted for the highest performance score among others in terms of accuracy (99.89%), precision (99.77%), recall (97.82%), and F1 score (98.78%), as well as accounted for the shortest execution time (7 s). These achievements outperform previous kNN-related botnet IoT studies. As a result, the kNN-FS model is selected for the implementation of SUKRY, an intelligent IDS on the Raspberry Pi machine. In the near future, the physical look of SUKRY will be improved properly for commercialization purposes.

Author Contributions: Conceptualization, I.S.; methodology, I.S.; validation, I.S.; formal analysis, I.S. and O.M.B.; investigation, I.S.; resources, I.S. and O.M.B.; writing—original draft preparation, I.S. and O.M.B.; writing—review and editing, I.S. and O.M.B.; supervision, O.M.B.; funding acquisition, O.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset used in this study is the Botnet-IoT dataset obtained from https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 20 December 2020).

Acknowledgments: This study was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah. The authors, therefore, acknowledge with thanks to DSR for technical and financial supports.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sicari, S.; Rizzardi, A.; Coen-Porisini, A. 5G in the internet of things era: An overview on security and privacy challenges. *Comput. Netw.* **2020**, *179*, 107345. [CrossRef]
2. Stoyanova, M.; Nikoloudakis, Y.; Panagiotakis, S.; Pallis, E.; Markakis, E.K. A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1191–1221. [CrossRef]
3. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial internet of things: Challenges, opportunities, and di-rections. *IEEE Trans. Industr. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
4. Agadakos, I.; Chen, C.Y.; Campanelli, M.; Anantharaman, P.; Hasan, M.; Copos, B.; Lindqvist, U. Jumping the air gap: Mod-eling cyber-physical attack paths in the Internet-of-Things. In Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy, Dallas, TX, USA, 3 November 2017; pp. 37–48.
5. Radanliev, P.; De Roure, D.C.; Nicolescu, R.; Huth, M.; Montalvo, R.M.; Cannady, S.; Burnap, P. Future developments in cyber risk assessment for the internet of things. *Comput. Ind.* **2018**, *102*, 14–22. [CrossRef]
6. Bertino, E.; Islam, N. Botnets and internet of things security. *Computer* **2017**, *50*, 76–79. [CrossRef]
7. Sun, L.; Du, Q. A Review of Physical Layer Security Techniques for Internet of Things: Challenges and Solutions. *Entropy* **2018**, *20*, 730. [CrossRef] [PubMed]
8. Zitta, T.; Neruda, M.; Vojtech, L. The security of RFID readers with IDS/IPS solution using Raspberry Pi. In Proceedings of the 2017 18th International Carpathian Control Conference, Sinaia, Romania, 28–31 May 2017; pp. 316–320.
9. Tirumala, S.S.; Sathu, H.; Sarrafzadeh, A. Free and open source intrusion detection systems: A study. In Proceedings of the 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China, 12–15 July 2015; Volume 1, pp. 205–210.
10. Guo, Z.; Harris, I.G.; Jiang, Y.; Tsaur, L.F. An efficient approach to prevent battery exhaustion attack on BLE-based mesh networks. In Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC), Santa Clara, CA, USA, 26–29 January 2017; pp. 1–5.
11. Anthi, E.; Williams, L.; Burnap, P. Pulse: An Adaptive Intrusion Detection for the Internet of Things. In Proceedings of the Living in the Internet of Things: Cybersecurity of the IoT-2018, London, UK, 28–29 March 2018; p. 35.
12. Liao, Y.; Vemuri, V. Use of K-Nearest Neighbor classifier for intrusion detection. *Comput. Secur.* **2002**, *21*, 439–448. [CrossRef]
13. Binkley, J.R.; Singh, S. *An Algorithm for Anomaly-Based Botnet Detection*; SRUTI 6; USENIX: Berkeley, CA, USA, 2006; p. 7.
14. Kondo, S.; Sato, N. Botnet traffic detection techniques by C&C session classification using SVM. In *Advances in Information and Computer Security*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 91–104, ISBN 9783540756507.
15. Seufert, S.; O'Brien, D. Machine learning for automatic defence against distributed denial of service attacks. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, Scotland, 24–28 June 2007; IEEE: Piscataway, NJ, USA, 2007.

16. Vargas, H.; Lozano-Garzon, C.; Montoya, G.A.; Donoso, Y. Detection of Security Attacks in Industrial IoT Networks: A Blockchain and Machine Learning Approach. *Electronics* **2021**, *10*, 2662. [CrossRef]
17. Berral, J.L.; Poggi, N.; Alonso, J.; Gavaldà, R.; Torres, J.; Parashar, M. Adaptive distributed mechanism against flooding network attacks based on machine learning. In *Proceedings of the 1st ACM workshop on Workshop on AISec–AISec '08, Alexandria, VA, USA, 27 October 2008*; ACM Press: New York, NY, USA, 2008.
18. Eslahi, M.; Salleh, R.; Anuar, N.B. Bots and botnets: An overview of characteristics, detection and challenges. In Proceedings of the 2012 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 23–25 November 2012; IEEE: Piscataway, NJ, USA, 2012.
19. Simkhada, E.; Shrestha, E.; Pandit, S.; Sherchand, U.; Dissanayaka, A.M. Security threats/attacks via botnets and botnet detection & prevention techniques in computer networks: A review. In *Proceedings of the Midwest Instruction and Computing Symposium (MICS)*; North Dakota State University: Fargo, ND, USA, 2019.
20. Rashid, M.; Kamruzzaman, J.; Hassan, M.; Imam, T.; Gordon, S. Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques. *Int. J. Environ. Res. Public Health* **2020**, *17*, 9347. [CrossRef]
21. Dwibedi, S.; Pujari, M.; Sun, W. A comparative study on contemporary intrusion detection "datasets" for machine learning research. In *Proceedings of the 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 9–10 November 2020*; IEEE: Piscataway, NJ, USA, 2020.
22. Pacheco, Y.; Sun, W. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets. In Proceedings of the 7th International Conference on Information Systems Security and Privacy, Austria, Vienna, 11–13 February 2021.
23. Aswal, K.; Dobhal, D.C.; Pathak, H. Comparative analysis of machine learning algorithms for identification of BOT attack on the Internet of Vehicles (IoV). In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020; IEEE: Piscataway, NJ, USA, 2020.
24. Hasan, M.; Islam, M.; Zarif, M.; Hashem, M. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet Things* **2019**, *7*, 100059. [CrossRef]
25. Bedi, P.; Mewada, S.; Vatti, R.; Singh, C.; Dhindsa, K.; Ponnusamy, M.; Sikarwar, R. Detection of attacks in IoT sensors networks using machine learning algorithm. *Microprocess. Microsyst.* **2021**, *82*, 103814. [CrossRef]
26. Singh, K.; Guntuku, S.C.; Thakur, A.; Hota, C. Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. *Inf. Sci.* **2014**, *278*, 488–497. [CrossRef]
27. Chen, J.; Li, K.; Tang, Z.; Bilal, K.; Yu, S.; Weng, C.; Li, K. A parallel random forest algorithm for big data in a spark cloud computing environment. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 919–933. [CrossRef]
28. Yusof, M.; Saudi, M.M.; Ridzuan, F. A new mobile botnet classification based on permission and API calls. In Proceedings of the 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 6–8 September 2017; IEEE: Piscataway, NJ, USA, 2017.
29. Duan, M.; Li, K.; Liao, X.; Li, K. A parallel multiclassification algorithm for big data using an extreme learning machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2337–2351. [CrossRef]
30. Vengatesan, K.; Kumar, A.; Parthibhan, M.; Singhal, A.; Rajesh, R. Analysis of Mirai botnet malware issues and its prediction methods in internet of things. In *Lecture Notes on Data Engineering and Communications Technologies*; Springer International Publishing: Cham, Switzerland, 2020; pp. 120–126, ISBN 9783030246426.
31. Marjani, M.; Nasaruddin, F.; Gani, A.; Karim, A.; Hashem, I.A.T.; Siddiqa, A.; Yaqoob, I. Big IoT data analytics: Architecture, opportunities, and open research challenges. *IEEE Access* **2017**, *5*, 5247–5261.
32. Gadelrab, M.S.; ElSheikh, M.; Ghoneim, M.A.; Rashwan, M. BotCap: Machine Learning Approach for Botnet Detection Based on Statistical Features. *Int. J. Commun. Netw. Inf. Secur.* **2018**, *10*, 563–579.
33. Hoang, X.; Nguyen, Q. Botnet detection based on machine learning techniques using DNS query data. *Future Internet* **2018**, *10*, 43. [CrossRef]
34. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A novel ensemble of Hybrid Intrusion Detection System for detecting Internet of Things attacks. *Electronics* **2019**, *8*, 1210. [CrossRef]
35. Al-Mashhadi, S.; Anbar, M.; Hasbullah, I.; Alamiedy, T.A. Hybrid rule-based botnet detection approach using machine learning for analysing DNS traffic. *PeerJ. Comput. Sci.* **2021**, *7*, e640. [CrossRef]
36. Wang, W.; Shang, Y.; He, Y.; Li, Y.; Liu, J. BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Inf. Sci.* **2020**, *511*, 284–296. [CrossRef]
37. Rambabu, K.; Venkatram, N. Ensemble classification using traffic flow metrics to predict distributed denial of service scope in the Internet of Things (IoT) networks. *Comput. Electr. Eng.* **2021**, *96*, 107444. [CrossRef]
38. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine. *Electronics* **2020**, *9*, 173. [CrossRef]
39. Moustafa, N.; Turnbull, B.; Choo, K. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 4815–4830. [CrossRef]
40. Farhat, S.; Abdelkader, M.; Meddeb-Makhlouf, A.; Zarai, F. Comparative study of classification algorithms for cloud IDS using NSL-KDD dataset in WEKA. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; IEEE: Piscataway, NJ, USA, 2020.

41. Celil, O.K.U.R.; Dener, M. Detecting IoT Botnet Attacks Using Machine Learning Methods. In Proceedings of the 2020 International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 3–4 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 31–37.
42. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features. *Electronics* **2020**, *9*, 144. [CrossRef]
43. Churcher, A.; Ullah, R.; Ahmad, J.; ur Rehman, S.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W. An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. *Sensors* **2021**, *21*, 446. [CrossRef] [PubMed]
44. Mrabet, H.; Belguith, S.; Alhomoud, A.; Jemai, A. A Survey of IoT Security Based on a Layered Architecture of Sensing and Data Analysis. *Sensors* **2020**, *20*, 3625. [CrossRef]
45. Wazirali, R. An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation. *Arabian J. Sci. Eng.* **2020**, *45*, 10859–10873. [CrossRef]
46. Kotu, V.; Deshpande, B. *Predictive Analytics and Data Mining: Concepts and Practice with Rapidminer*; Morgan Kaufmann: Burlington, MA, USA, 2014.
47. Epishkina, A.; Zapechnikov, S. A syllabus on data mining and machine learning with applications to cybersecurity. In Proceedings of the 2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC), Moscow, Russia, 6–8 July 2016; IEEE: Piscataway, NJ, USA, 2016.
48. Panthong, R.; Srivihok, A. Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm. *Procedia Comput. Sci.* **2015**, *72*, 162–169. [CrossRef]
49. Lee, S.; Schowe, B.; Sivakumar, V.; Morik, K. *Feature Selection for High-Dimensional Data with Rapidminer*; Universitätsbibliothek Dortmund: Dortmund, Germany, 2012.
50. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
51. Ge, M.; Fu, X.; Syed, N.; Baig, Z.; Teo, G.; Robles-Kelly, A. Deep learning-based intrusion detection for IoT networks. In Proceedings of the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), Kyoto, Japan, 1–3 December 2019; IEEE: Piscataway, NJ, USA, 2019.
52. Alejandre, F.V.; Cortes, N.C.; Anaya, E.A. Feature selection to detect botnets using machine learning algorithms. In Proceedings of the 2017 International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 22–24 February 2017; IEEE: Piscataway, NJ, USA, 2017.
53. Su, S.; Sun, Y.; Gao, X.; Qiu, J.; Tian, Z. A correlation-change based feature selection method for IoT equipment anomaly detection. *Appl. Sci.* **2019**, *9*, 437. [CrossRef]
54. Shobana, M.; Poonkuzhali, S. A Novel Approach for Detecting IoT Botnet Using Balanced Network Traffic Attributes. In *Service-Oriented Computing—ICSOC 2020 Workshops*; Springer International Publishing: Cham, Switzerland, 2021; pp. 534–548, ISBN 9783030763510.
55. Raspberry, O.S. Available online: https://www.raspberrypi.com/software/ (accessed on 20 December 2020).
56. Suricata. Available online: https://suricata.io/download/ (accessed on 20 December 2020).
57. OpNIDS. Available online: https://github.com/OPNids (accessed on 20 December 2020).
58. Muñoz, A.; Farao, A.; Correia, J.R.C.; Xenakis, C. P2ISE: Preserving Project Integrity in CI/CD Based on Secure Elements. *Information* **2021**, *12*, 357. [CrossRef]
59. Bahsi, H.; Nomm, S.; La Torre, F.B. Dimensionality reduction for machine learning based IoT botnet detection. In Proceedings of the 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 18–21 November 2018; IEEE: Piscataway, NJ, USA, 2018.
60. Kumar, A.; Lim, T.J. EDIMA: Early detection of IoT malware network activity using machine learning techniques. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; IEEE: Piscataway, NJ, USA, 2019.