International Journal of Safety and Security Engineering
Vol. 15, No. 6, June, 2025, pp. 1165-1172
Journal homepage: http://iieta.org/journals/ijsse

Comparative Analysis of Two-Step Machine Learning Models for Botnet SPAM Detection

Thoriq Afif Habibi[1], Tohari Ahmad[1*], Dandy Pramana Hostiadi[2], Muhammad Aidiel Rachman Putra[1], Ntivuguruzwa Jean De La Croix[1,3], Md. Sagar Hossen[4], Abdulati K. S. Jahbel[5], Royyana M. Ijtihadie[1]

[1] Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia
[2] Department of Magister Information Systems, Institut Teknologi dan Bisnis STIKOM Bali, Denpasar 80234, Indonesia
[3] College of Science and Technology, University of Rwanda, Kigali 3900, Rwanda
[4] Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh
[5] NSR International for Information Technology, Tripoli 1000, Libya

Corresponding Author Email: tohari@its.ac.id

https://doi.org/10.18280/ijsse.150608

Received: 26 April 2025
Revised: 30 May 2025
Accepted: 15 June 2025
Available online: 30 June 2025

Keywords:
botnet, SPAM detection, machine learning, two step classification, network security, network traffic analysis

ABSTRACT

A botnet refers to a group of devices that have been infected with malicious software, allowing them to be controlled to carry out harmful activities such as identity theft, denial-of-service attacks (DDoS), personal data theft, click fraud, and SPAM distribution. Among these activities, SPAM is the most prevalent type of cyber-attack in today's digital landscape, often aimed at stealing personal information and spreading infections to new devices. Research has shown that using a multistep classification approach can enhance the performance of models designed to detect cyber-attacks. However, the optimal combination of classifiers for identifying SPAM within botnet activities has not yet been established. This study introduces a method for detecting botnet SPAM through a two-step classification process, utilizing two types of classifiers chosen from a set of three: Decision Tree, Naïve Bayes, and Logistic Regression. In the first step, the model categorizes data into normal activity and botnet activity. In the second step, it further classifies botnet activities into SPAM and non-SPAM categories. The method was evaluated using the NCC-2 sensor 3 public dataset, which comprises various types of simultaneous botnet attacks, including SPAM. This dataset has an imbalance proportion, with most network traffic consisting of normal activity, followed by non-SPAM botnet activity, while SPAM botnets represent the smallest group. The experimental results revealed that employing the Decision Tree algorithm in both stages of the classification process achieved the best outcomes. The performance metrics for this proposed method showed an accuracy of 98.96%, a precision of 99.01%, a recall of 98.96%, and an F1-score of 98.98%.

1. INTRODUCTION

The massive usage of technology has always led to an increasing number of cyber-attacks. The amount of financial gain becomes one of the attacker's motivations for seeking security vulnerability [1]. One method of exploiting these vulnerabilities is using malicious software, known as malware, distributed to the target devices [2, 3]. Malware is a harmful program that can damage systems, gain data, take unauthorized access, and any other malicious activity [4]. Various types of malwares are used today, including robot networks called botnets. Botnet attacks involve infecting devices to form a network that can be fully managed by a bot master (attacker) [5]. The attacker manipulates infected devices, commonly called zombies, to achieve various goals, for example click fraud, Distributed Denial of Service (DDoS), and phishing.

Spamming is an activity that sends a large volume of unsolicited messages to the target device [6]. Spammers usually use botnets because of their ability to change the sender's IP address [7]. Besides, SPAM was an attack with the highest increment, by over 15%, or reaching 30.6% among the whole traffic [8]. In addition, SPAM is also used as an initial infection to form a new bot device by sending a message attached with infected code, malicious URL, or phishing to the target device [9]. Thus, the detection of SPAM botnets is needed to prevent the expansion of botnet networks and reduce other types of botnet attacks.

Research in the field of botnet detection has progressed through various methods, starting with machine learning [10-12] and deep learning [13], and moving towards transfer learning [14] and hybrid approaches [5]. Most existing studies have primarily concentrated on distinguishing between normal traffic and botnet traffic. However, as the demand for models that can also identify SPAM activities has increased, there is a growing need for multi-class detection models. Among the various studies focusing on this area, two-stage detection has emerged as particularly effective [4, 15]. Despite this, there has yet to be a comprehensive examination of the best combinations of algorithms for two-step multi-class detection,

especially in relation to SPAM botnet detection. Therefore, to develop a two-stage detection model that can classify data as normal, non-SPAM botnets, or SPAM botnets, it is essential to conduct a comparative analysis to identify the most effective combinations of different machine learning algorithms. This approach will help improve the response to botnet attacks right from the initial infection phase, which is often indicated by SPAM activities.

This study presents a comparative analysis of SPAM botnet detection models using a two-step machine learning classification approach. In the first step, the model classifies network activity into two classes: normal and botnet. Next, the botnet label results in the first step are reclassified in the second step into two classes, namely, non-SPAM botnet and SPAM botnet. Model classifiers for the first and second steps are combined from three well-known classification algorithms, namely Decision Tree (DT), Naïve Bayes (NB), and Logistic Regression (LR). A comparison among nine possible classifier combinations was conducted in this research to determine the best model. This detection model not only identifies botnet activities but also SPAM botnet activities, which allows the prevention of botnet spread. The contributions of this article are summarized as follows:

(1) *A Two-Stage Botnet Detection Framework:* We propose a novel machine learning-based model that first classifies network activity as normal or botnet and then further distinguishes botnet traffic into Non-SPAM and SPAM botnet categories. This hierarchical approach enhances detection specificity and enables early identification of SPAM-related threats.

(2) *Comprehensive Evaluation of Classifier Combinations*: The model uses Decision Tree, Naïve Bayes, and Logistic Regression to construct nine possible classifier pairings across both stages. A detailed performance comparison identifies the most effective combination, demonstrating improved accuracy matching the state-of-the-art.

This paper is structured into five sections, namely: (I) Introduction, (II) Related Works, (III) Proposed Method, (IV) Result and Discussion, and (V) Conclusion. Section I provides a general description of botnets and the problems they cause. Section II discusses other research related to this research. Section III details the proposed SPAM botnet detection method. Section IV presents the results of the experiment and the analysis. Finally, Section V offers a conclusion.

2. RELATED WORKS

Previous researchers have proposed a model to detect botnet activity by grouping network packets into: normal and botnet. They implement various methods to enhance the performance of the detection model, such as machine learning (ML) [10, 11], deep learning (DL) [13], anomaly detection [12], transfer learning [14], and hybrid analysis [5]. Even though multilabel botnet detection is quite challenging, research has been performed with multilabel classifiers to identify botnet families [16], botnet scenarios [4], and SPAM botnets [15].

Hasan et al. [16] highlighted the challenge of implementing Deep Learning methods in resource-constrained IoT devices due to the large network traffic data and memory space required. This paper proposed a botnet detection framework that leverages a multistep process to identify a botnet activity and then classify each into a botnet family. At first, this framework builds a state transition matrix by doing data and feature extraction using the Zeek Network Analysis Framework. The state transition frequency is then used to construct Markov Chains to model the behaviors of each different bot. The framework utilizes a Class-specific Cost Regulation Extreme Learning Machine (CCR-ELM) for botnet detection to overcome the imbalance dataset problem. This proposed framework presents a better performance comparison of classical machine learning algorithms while still having a problem with memory space usage.

Popoola et al. [4] addressed the challenge of detecting multiclass botnet attacks by creating a detection model that employs hybrid deep learning techniques. This model classifies network activities into five categories: DDoS, DoS, normal operations, reconnaissance, and information theft. Their method utilizes a long short-term memory autoencoder (LAE) to create a simplified set of features and a deep bidirectional long short-term memory (BLSTM) network to classify incoming data packets. This approach effectively reduces issues associated with underfitting and overfitting. Test results indicate that their method achieved a significant reduction in data size by 91.89% and demonstrated strong generalization capabilities. However, while the model addresses the challenges of underfitting and overfitting, it does not consider the problem of imbalanced data. Nevertheless, the performance of the model was evaluated using the Matthews Correlation Coefficient (MCC), a metric that offers a fair and balanced view of classification performance, especially in situations involving imbalanced datasets.

The SPAM botnet detection has been introduced in the study [15]. This study presents a method for detecting SPAM within botnets using a multi-step decision tree (DT) algorithm. The process begins with data preparation, where relevant features are selected and categorical variables are encoded. Network activities are then labeled as normal, non-SPAM, or SPAM botnet activities. Following this, the data is divided into different sets for machine learning classification. The initial step focuses on distinguishing between normal and botnet activity, while the second step further classifies the botnet activities into SPAM and non-SPAM categories. To address the risk of imbalanced data distribution, this research ensured that the data splitting took into account the proportions of the various classes. The proposed approach outperforms a direct multi-label decision tree classification across five different depth settings. However, the study primarily focuses on comparing the effectiveness of single-step versus multi-step detection using decision trees. A more detailed investigation is needed to determine which combinations of algorithms work best for tasks involving two-step multi-class detection.

3. PROPOSED METHOD

This study proposes a model for detecting botnet SPAM attacks in the network using a two-step classification approach. The proposed model comprises five main processes, namely: (1) Data Labeling, (2) Categorical Data Encoding, (3) Feature Selection, (4) Data Splitting, and (5) Training the Model using Two-Step Machine Learning Classification. A flowchart illustrating this process is provided in Figure 1.

3.1 Data labelling

Classification requires a target feature during the training and testing phases. The bidirectional network flow (binetflow)

NCC-2 dataset [17] has an 'ActivityLabel' feature with values of 0 for normal activities and 1 for botnet activities. A 'Label' feature also has string values such as "flow=background," "flow=From-Botnet-V44-ICMP. To perform a 3-class classification, the 'ActivityLabel' feature alone is insufficient. Therefore, an analysis of 'Label' is needed to specify the class for every network activity. Data labeling categorizes network activities based on the 'Label' feature values into a normal, SPAM and non-SPAM botnets. Network activity with the 'Label' containing the terms 'spam' and ' botnet' are classified into class 2 (SPAM botnet). If only 'botnet' is present, that computer network activity is classified into class 1 (non-SPAM botnet). Activities without these two words are classified into class 0 (normal).

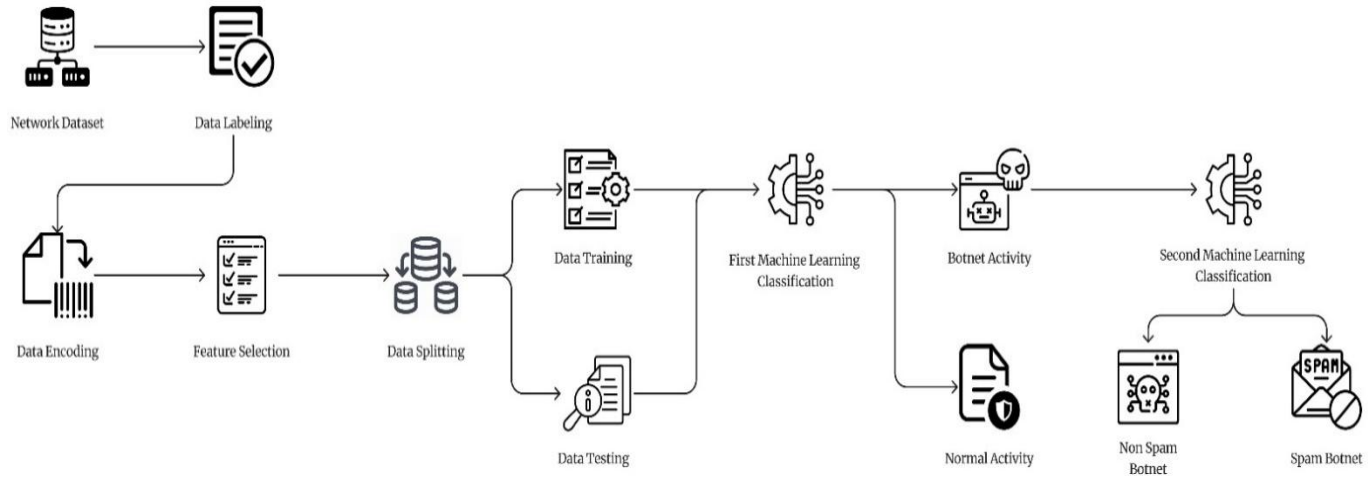

Figure 1. The flow of the proposed method

3.2 Feature selection

In this study, we conducted feature selection as part of our machine learning process. We identified three specific criteria for filtering features: categorical features with excessive data variation, features with a high percentage of missing values, and features that represent labels. Any feature that met one of these three criteria was excluded from our analysis.

To convert categorical features into numerical format, we utilized Dummy Variable Encoding. This method generates (unique_value - 1) new features from a categorical variable, where n represents the number of unique values in that variable. Consequently, a categorical feature with many unique values results in a greater number of new features compared to one with fewer unique values. However, if a feature shows too much data variation, the resulting increase in new features can negatively impact the performance of machine learning classification [18]. Therefore, we decided to remove features with high data variation.

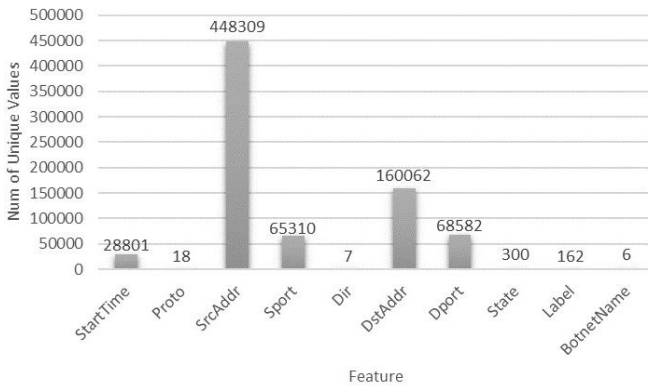

Figure 2. Number of unique values on categorical features

From the botnet dataset analyzed in this study, there are at least ten categorical features, including ‘BotnetName’, ‘Proto’, ‘SrcAddr’, ‘DstAddr’, ‘Sport’, ‘Dport’, ‘State’, ‘Dir’, ‘Label’, and ‘StartTime’. Our analysis indicated that the categorical features ‘Proto’, ‘Dir’ and ‘BotnetName’ had a limited number of unique values (Figure 2). Hence, we chose to exclude these three features from the classification process. We further examined the relationship between these features with low value variations and the primary label, discovering that ‘Proto’ had the most significant influence on our three-class classification. As a result, ‘Proto’ was the only feature retained for further analysis, while the others were discarded.

In the classification stage of machine learning, it is crucial to address missing values, as they can significantly affect detection performance. We removed the two features with the highest proportions of missing values: ‘dTos’ with 7.75% missing data (301,103 entries) and ‘sTos’ with 0.78% missing data (30,137 entries). Lastly, we eliminated any features that indicated labels, specifically retaining ‘ActivityLabel’ and ‘SensorId’, both of which were also removed from consideration in the subsequent processing.

3.3 Data encoding

As discussed in the earlier section, this study employs Dummy Variable Encoding to transform categorical features. This method involves examining the differences among values in categorical features and then creating new features by excluding one of the variations. Specifically, the number of new features created will equal the total number of unique values in the original feature minus one. After these new features are created, each will indicate the presence or absence of the original value using 1 or 0. This indicates that only one of the new features will have a value of 1 at any given time, while the others will be 0. Consequently, the total number of new features will not match the total unique values of the original feature, as omitting that one feature ensures that the combination remains distinct, with only 0 values in the new features. For instance, if the ‘Proto’ feature has 18 unique values, it will result in 17 new features. Combined with the

existing numerical features, this results in a total of 21 features carried forward for further analysis.

3.4 Data splitting

As shown in Table 1, the number of Non-SPAM and SPAM botnet data is highly imbalanced compared to normal data. Therefore, data must be separated based on its class before being split. Each data was then divided into training and testing data with the same proportion, 70%:30%, and into training data and testing data with an equal proportion of 70%:30%. This portion was adopted from the study [19], which resulting optimal performance for machine learning. Data splitting will result in 6 datasets: (1) normal training data, (2) normal testing data, (3) non-SPAM botnet training data, (4) non-SPAM botnet testing data, (5) SPAM botnet training data, and (6) SPAM botnet testing data. Subsequently, the data is combined back into a single testing dataset and a single training dataset. The process of this step is shown in Figure 3.

Table 1. Amount of data for each class

	Total	Percentage (%)
Normal	3,591,792	92.43
Non spam botnet	271,000	6.97
Spam Botnet	23,000	0.59

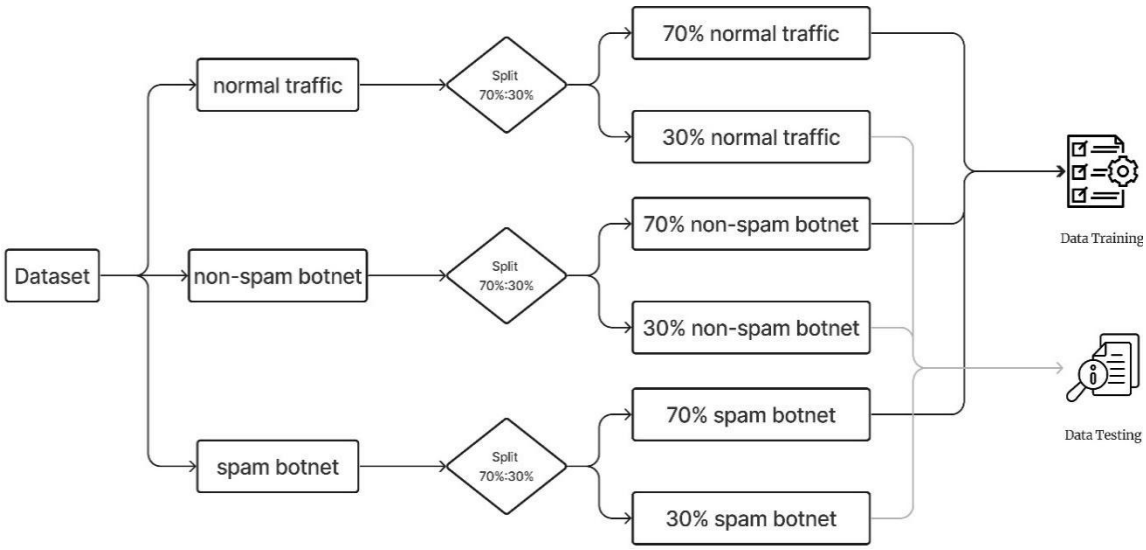

Figure 3. Data splitting flowchart

3.5 Two-step machine learning classification

This process became the main contribution to building a detection model. Training Data was used to train the model, which was conducted with a two-step classification. Each step involves binary classification of network activities. The first step aims to classify network activities into normal (0) and botnet (1). In step two, data is classified as botnet (1) and then classified into non-SPAM and SPAM botnet classes. This process is shown in Algorithm 1.

Algorithm 1. Two Step Spam Botnet Classification

Input: dataTest, dataTrain
Output: performanceScore

columns = ['Dur', 'TotPkts', 'TotBytes', 'SrcBytes', 'esp', 'gre', 'icmp', 'igmp', 'ipv6', 'ipv6-icmp', 'ipx/spx', 'llc', 'pim', 'rarp', 'rsvp', 'rtcp', 'rtp', 'tcp', 'udp', 'udt', 'unas']
Classifier(): Initialize an instance of machine learning classifier /* Here, we use DecisionTree, LogisticRegression, and NaiveBayes */
Classifier.fit(x, y): Training a machine learning model based on training data's feature (x) and labels (y)
Classifier.predict(x): Predict label using trained model based on testing data's feature (x)
len(var): length of variable
performanceScore(test, predict): Calculate accuracy, precision, recall, and f1-score of machine learning model

Function BotnetPartition(data)
/* Separate feature and label in botnet data */
x← []; y ← [];
for index ← 0 to len(data) **do**
x ← x + [data [index][columns]]
if data [index][‘MultiClassLabel’] = 0 **do**
y ← y + [0]
else do
y ← y + [1]
return x, y

Function SpamPartition(data)
/* Separate feature and label in spam botnet data */
x← []; y ← [];
for index ← 0 to len(data) **do**
if data[index][‘MultiClassLabel’] = 1 **do**
x [index] ← data [index][columns]
y [index] ← 0
else if data[index][‘MultiClassLabel’] = 2 **do**
x [index] ← data [index][columns]
y [index] ← 1
return x, y

Step 1: Normal and Botnet Classification
botnetClassifier ← Classifier()
xBotTrain, yBotTrain ← BotnetPartition (dataTrain)
xBotTest, yBotTest ← BotnetPartition (dataTest)
botnetClassifier.fit (xBotnetTrain, yBotnetTrain)
yBotPredict ← botnetClassifier.predict (xBotnetTest)
Step 2: Non-Spam and Spam Botnet Classification
spamClassifier ← Classifier()
spamTest, nonSpamTest ← []
for index ← 0 to len (dataTest) **do**
if yBotPredict [index] = 1 **do**
spamTest ← spamTest + [dataTest[index]]
else do
nonSpamTest←nonSpamTest + [dataTest[index]]

```
    xSpamTrain, ySpamTrain ← SpamPartition (dataTrain)
    xSpamTest, ySpamTest ← SpamPartition (spamTest)
    spamClassifier.fit (xSpamTrain, ySpamTrain)
    ySpamPredict ← spamClassifier.predict (xSpamTest)
Step 3: Evaluate Model Performance
    multiTest ← spamTest + nonSpamTest
    yMultiTest ← multiTest['MultiClassLabel']
    yMultiPredict ← []
    for index ← 0 to len(ySpamPredict) do
        yMultiPredict ← yMultiPredict + [ySpamPredict[index] +
        1]
    for index ← 0 to len(nonSpamTest) do
        yMultiPredict ← yMultiPredict + [0]
    return performanceScore(yMultitest, yMultiPredict)
```

The classification process uses three algorithms: DT, NB, and LR. Each algorithm is implemented in the first step and combined with the same or different algorithm in the second step. There are 8 combinations, namely: (1) DT-DT, (2) DT-LR, (3) DT-NB, (4) NB-NB, (5) NB-DT, (6) NB-LR, (7) LR-LR, (8) LR-NB, and (9) LR-DT.

4. RESULT AND DISCUSSION

This section analyses the performance of botnet SPAM detection using two-step classification algorithms from eight classifier combinations. The first subsection explains the dataset used for training and testing detection models. Then, in the second subsection, the discussion about detection model performance is provided based on accuracy (A), precision (P), recall (R), and F1-score (F1). The model performance is compared with each other among 8 classifier combinations.

4.1 Dataset

The detection model utilizes the NCC-2 sensor 3, which examines bidirectional network flow data. The dataset consists of eighteen unique features, including ten categorical and eight numerical. The NCC-2 dataset was developed through a simulation designed to mimic different types of attacks occurring at the same time. This approach enables the examination and analysis of multiple threat scenarios in a controlled environment. It encompasses four different attacks: Distributed Denial of Service (DDoS), Click Fraud, Port Scanning (PS), and SPAM. The dataset records eight hours of network activity involving five specific botnets: NSIS.ay, Neris, Murlo, Rbot, and Virus. Given the different types of attacks represented, this dataset is classified as imbalanced when segmented into normal, non-SPAM botnet, and SPAM botnet. The normal class is the most prevalent in testing data, containing 1,077,251, while the non-SPAM botnet class includes 81,096, and the SPAM botnet class has the fewest data at 6,989.

4.2 Result analysis

Data labelling creates a new column named 'MultiClassLabel' containing a class for each data. The value for this column is '0' for normal activity, '1' for non-SPAM botnet, and '2' for spam Botnet. Then, four numerical and one categorical feature are selected during the feature selection process. These selected features are 'Dur', 'TotPkts', 'TotBytes', 'SrcBytes', and 'Proto'. Since 'Proto' is categorical, we encode it using dummy variable encoding, which converts 'Proto' into 17 numerical features. In total, 21 numerical features will be brought into the machine-learning model. The dataset consists of 92.43% normal activity, 6.97% non-SPAM botnet, and 0.59% SPAM botnet, as shown in Table 1. Each class is then split with a proportion of 70%:30% for training and testing.

A detection model was built using a multistep classification algorithm and a combination of 3 classifiers: DT, NB, and LR. This research combined three classification algorithms with similar and different algorithms. The detection model was tested using four evaluation matrixes: A, P, R, and F1. Table 2 shows the model performance for all 8-combination classifiers. It indicates that the DT-DT combination has the best A with 98.96%, followed by DT-LR, DT-NB, LR-DT, LR-LR, LR-NB, NB-DT, NB-LR, and NB-NB.

The confusion matrix for each model is also presented in Figures 4 and 5. It shows that the model predicts too much data as normal when using LR as the first step classifier. It will affect the performance because some botnet data that should be processed in the second step is classified as normal. The data imbalance between normal and botnet amplifies the impact. The confusion matrix also shows that the LR-LR combination predicts 93 data as 'SPAM Botnet', while the LR-NB combination does not predict any data as 'Non-SPAM Botnet'. While the LR-DT combination yields improved results compared to the former two, it still falls short of addressing the deficiencies in the initial step. Notably, all three models exhibit P, R, and F1 metrics below 50% for botnet and botnet SPAM.

The DT model combination demonstrates better performance than other models because of its ability to effectively manage complex data patterns and its robustness in dealing with unbalanced data. This strength makes the DT model particularly useful in binary classification tasks, which were structured in this study across two phases using a two-step classification approach. However, the DT-DT combination shows limitations in accurately identifying botnet SPAM classes.

Table 2. Detection performance comparison

Method	Normal (%)			Non-Spam Botnet (%)			Spam Botnet (%)			Weighted Avg. (%)			A (%)
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
DT-DT	99.63	99.34	99.49	93.38	94.83	94.10	69.13	87.35	77.18	99.01	98.96	98.98	98.96
DT-LR	99.63	99.34	99.49	87.55	95.57	91.38	94.54	36.39	52.55	98.76	98.70	98.64	98.70
DT-NB	99.63	99.34	99.49	93.84	48.73	64.15	13.18	92.60	23.07	98.71	95.78	96.57	95.78
NB-NB	91.85	28.35	43.33	3.25	17.72	5.50	1.78	99.80	3.51	85.14	28.04	40.46	28.04
NB-DT	91.85	28.35	43.33	6.57	65.56	11.94	27.65	92.23	42.55	85.53	31.32	41.14	31.32
NB-LR	91.85	28.35	43.33	6.53	66.44	11.89	32.61	36.41	34.41	85.56	31.05	41.09	31.05
LR-LR	92.67	98.89	95.68	13.81	2.67	4.48	0	0	0	86.63	91.60	88.76	91.60
LR-NB	92.67	98.89	95.68	0	0	0	10.37	23.41	14.37	85.73	91.55	88.53	91.55
LR-DT	92.67	98.89	95.68	24.65	2.04	3.76	18.02	23.41	20.37	87.49	91.70	88.83	91.70

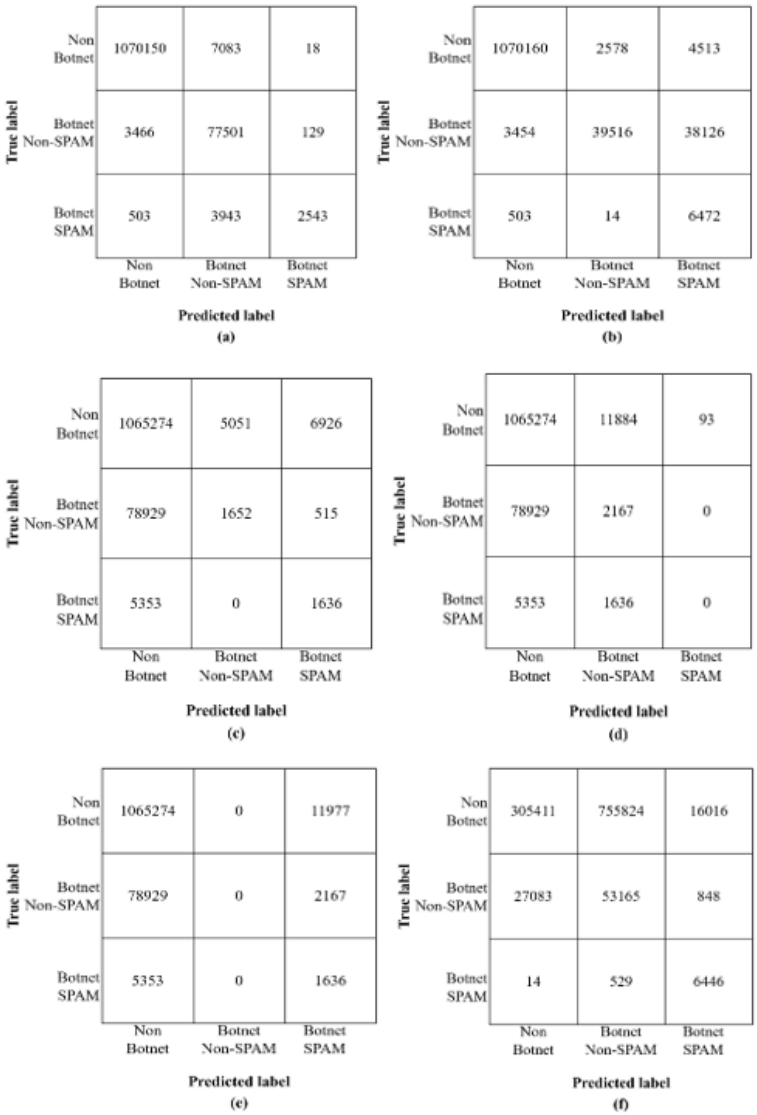

Figure 4. Confusion matrix (a) DT-DT, (b) DT-LR, (c) DT-NB, (d) NB-NB, (e) NB-DT, (f) NB-LR

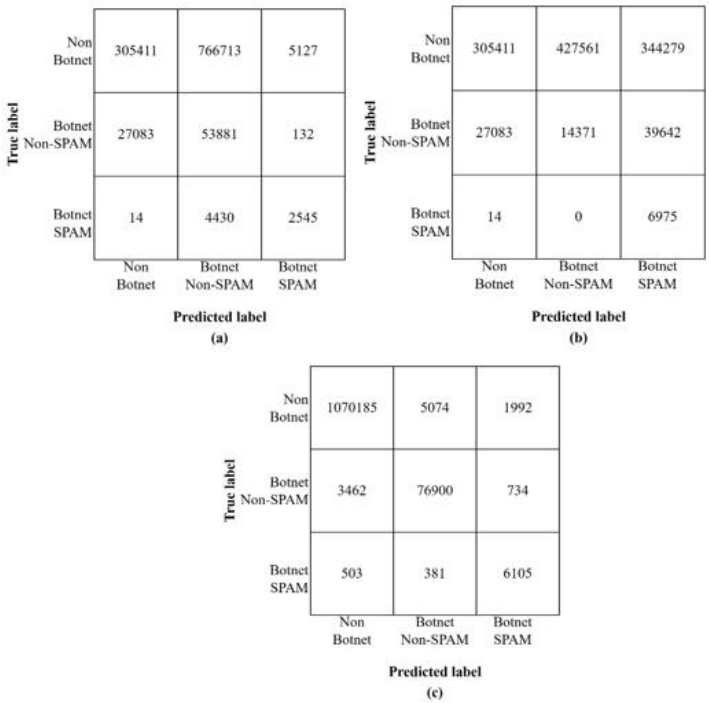

Figure 5. Confusion matrix (a) LR-LR, (b) LR-NB, (c) LR-DT

As illustrated in Figure 4(a), a considerable amount of SPAM botnet data is misclassified as non-SPAM botnets. This challenge likely arises from the similar traits shared among different types of botnet attacks.

Additionally, other model combinations also face difficulties in recognizing botnet SPAM classes. Often, these models are forced to trade off detection accuracy for other classes. This challenge stems from the nature of botnet SPAM activities, which occupy a grey area between normal and non-SPAM botnets behavior, making them particularly hard to detect. Future research could explore additional factors such as target variation and attack intensity, as botnet SPAM activities typically launch attacks in a dispersed manner, resulting in significant variability in targets.

Furthermore, as shown in Table 2, the proposed method utilizing the DT model in both phases achieves the highest performance among the combinations evaluated, with an accuracy of 98.96%, a weighted precision of 99.01%, a weighted recall of 98.96%, and a weighted F1 score of 98.98%.

5. CONCLUSION

This research proposed a classification method using a multistep algorithm to detect SPAM Botnet activity. The proposed methods comprise five main processes, namely: (1) Data Labeling, (2) Categorical Data Encoding, (3) Feature Selection, (4) Data Splitting, and (5) Training the Model using Two-Step Machine Learning Classification. Data labelling categorizes network activities based on 'Label' feature values into each class. Then, the process continues with categorical data encoding, which is performed on the relevant feature, 'Proto', using dummy variable encoding. At the end of this process, there are 25 numerical features with one target feature. The third process then removed two features, 'sTos' and 'dTos', based on high missing values, and the other two features, 'ActivityLabel' and 'SensorId', that function as labels. In Data Splitting, data from previous processes is divided into training and testing data with equal proportions for each class, 70%:30%. The process was then followed by training the detection model using a two-step classification that combined three classifiers: DT, NB, and LR. In the first step, the detection model classified network traffic to: normal and botnet. That identified as a botnet will take the further step to be classified as a non-SPAM and SPAM botnets. By implementing this approach, the system enables the detection of SPAM activities and classifies network activities into ' Normal', 'Non-SPAM Botnet', and 'SPAM Botnet'. The experiment result using eight combinations of classifiers shows that the proposed model performs best in the detection model with a DT in both steps. This model gets an accuracy value of 98.96%, 99.01% precision weighted average, 98.96% recall weighted average, and 98.98% F1-score weighted average.

In the future, it will be important to explore effective techniques for dealing with imbalanced data. Some of these techniques include under sampling, oversampling, and data augmentation. These approaches can help models better recognize patterns in the minority class. However, it is crucial to maintain the quality of the data throughout these processes to ensure that the essential characteristics of the minority group are preserved. An examination of factors like the variation in attack targets is essential, given the nature of SPAM attacks, which tend to occur in a dispersed manner, leading to significant differences in target selection. Finally, exploring the application of deep learning techniques for detecting multiple classes in scenarios with unbalanced data could prove advantageous. It is important to evaluate algorithms that are designed to handle these imbalanced conditions, such as Convolutional Neural Networks (CNNs) that utilize class weights or BLSTM/GRU models with Focal Loss, for their effectiveness and advantages.

Furthermore, other datasets can be taken to evaluate the performance of the method. These include CTU-13 [20] and NCC [21].

ACKNOWLEDGMENT

This research was supported by Institut Teknologi Sepuluh Nopember under the 2025 Research Scheme.

REFERENCES

[1] Chng, S., Lu, H.Y., Kumar, A., Yau, D. (2022). Hacker types, motivations and strategies: A comprehensive framework. Computers in Human Behavior Reports, 5: 100167. https://doi.org/10.1016/j.chbr.2022.100167

[2] Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M. (2022). A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. Future Generation Computer Systems, 130: 1-18. https://doi.org/10.1016/j.future.2021.11.030

[3] Yang, B., Yu, Z., Cai, Y. (2022). Malicious software spread modeling and control in cyber–physical systems. Knowledge-Based Systems, 248: 108913. https://doi.org/10.1016/j.knosys.2022.108913

[4] Popoola, S.I., Adebisi, B., Hammoudeh, M., Gui, G., Gacanin, H. (2020). Hybrid deep learning for botnet attack detection in the Internet-of-Things networks. IEEE Internet of Things Journal, 8(6): 4944-4956. https://doi.org/10.1109/JIOT.2020.3034156

[5] Putra, M.A.R., Ahmad, T., Hostiadi, D.P., Ijtihadie, R.M. (2024). Botnet sequential activity detection with hybrid analysis. Egyptian Informatics Journal, 25: 100440. https://doi.org/10.1016/j.eij.2024.100440

[6] Bhuiyan, H., Ashiquzzaman, A., Juthi, T.I., Biswas, S., Ara, J. (2018). A survey of existing e-mail spam filtering methods considering machine learning techniques. Global Journal of Computer Science and Technology, 18(2): 20-29.

[7] Hachem, N., Mustapha, Y.B., Granadillo, G.G., Debar, H. (2011). Botnets: Lifecycle and taxonomy. In 2011 Conference on Network and Information Systems Security, La Rochelle, France, pp. 1-8. https://doi.org/10.1109/SAR-SSI.2011.5931395

[8] Bignell, F. (2023). Spam Rates Double and Ransomware Worsens Finds Acronis in Cyberthreats Report. The Fintech Times. https://Thefintechtimes.Com/Spam-Rates-Double-and-Ransomware-Worsens-Finds-Acronis-in-Cyberthreats-Report/.

[9] Eslahi, M., Salleh, R., Anuar, N.B. (2012). Bots and botnets: An overview of characteristics, detection and challenges. In 2012 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, pp. 349-354.

https://doi.org/10.1109/ICCSCE.2012.6487169
[10] Hussain, F., Abbas, S.G., Pires, I.M., Tanveer, S., Fayyaz, U.U., Garcia, N.M., Shahzad, F. (2021). A two-fold machine learning approach to prevent and detect IoT botnet attacks. IEEE Access, 9: 163412-163430. https://doi.org/10.1109/ACCESS.2021.3131014
[11] Velasco-Mata, J., González-Castro, V., Fernández, E.F., Alegre, E. (2021). Efficient detection of botnet traffic by features selection and decision trees. IEEE Access, 9: 120567-120579. https://doi.org/10.1109/ACCESS.2021.3108222
[12] Ramesh, R.B., Thangaraj, S.J.J. (2023). Analyzing and detecting botnet attacks using anomaly detection with machine learning. In 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 911-915. https://doi.org/10.1109/ICIRCA57980.2023.10220903
[13] Lo, W.W., Kulatilleke, G., Sarhan, M., Layeghy, S., Portmann, M. (2023). XG-BoT: An explainable deep graph neural network for botnet detection and forensics. Internet of Things, 22: 100747. https://doi.org/10.1016/j.iot.2023.100747
[14] Rabhi, S., Abbes, T., Zarai, F. (2023). Transfer learning-based Mirai botnet detection in IoT networks. In 2023 International Conference on Innovations in Intelligent Systems and Applications (INISTA), Hammamet, Tunisia, pp. 1-5. https://doi.org/10.1109/INISTA59065.2023.10310379
[15] Putra, M.A.R., Ahmad, T., Ijtihadie, R.M., Hostiadi, D.P. (2023). Detecting botnet spam activity by analyzing network traffic using two-stack decision tree algorithms. In 2023 International Conference of Computer Science and Information Technology (ICOSNIKOM), Binjia, Indonesia, pp. 1-6. https://doi.org/10.1109/ICoSNIKOM60230.2023.10364480
[16] Hasan, N., Chen, Z., Zhao, C., Zhu, Y., Liu, C. (2022). IoT botnet detection framework from network behavior based on extreme learning machine. In IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, pp. 1-6. https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798307
[17] Putra, M.A.R., Hostiadi, D.P., Ahmad, T. (2022). Botnet dataset with simultaneous attack activity. Data in Brief, 45: 108628. https://doi.org/10.1016/j.dib.2022.108628
[18] Leygonie, R., Lobry, S., Vimont, G., Wendling, L. (2023). Transforming multidimensional data into images to overcome the curse of dimensionality. In 2023 IEEE International Conference on Image Processing (ICIP), Kuala Lumpur, Malaysia, pp. 700-704. https://doi.org/10.1109/ICIP49359.2023.10222192
[19] Kim, J., Shim, M., Hong, S., Shin, Y., Choi, E. (2020). Intelligent detection of IoT botnets using machine learning and deep learning. Applied Sciences, 10(19): 7009. https://doi.org/10.3390/app10197009
[20] Garcia, S., Grill, M., Stiborek, J., Zunino, A. (2014). An empirical comparison of botnet detection methods. Computers & Security, 45: 100-123. https://doi.org/10.1016/j.cose.2014.05.011
[21] Hostiadi, D.P., Ahmad, T. (2021). Dataset for Botnet group activity with adaptive generator. Data in Brief, 38: 107334. https://doi.org/10.1016/j.dib.2021.107334