Machine Learning-Based Approach for Real-Time Detection of Botnet Activities within Wireless Network Infrastructures

Okafor Adanna P., Asogwa T.C.

Department of Computer Science, Enugu State University of Science and Technology, Nigeria.

DOI: https://doi.org/10.51244/IJRSI.2025.12110189

Received: 21 November 2025; Accepted: 29 November 2025; Published: 09 December 2025

ABSTRACT

In recent years, botnet attacks have emerged as one of the most prevalent and sophisticated cybersecurity threats, exploiting network vulnerabilities to compromise system integrity, confidentiality, and availability. Traditional security mechanisms, such as signature-based intrusion detection systems, struggle to keep pace with the dynamic and evolving nature of these threats. This study presents a machine learning-based approach for real-time detection of botnet activities within wireless network infrastructures. Using the Kaggle Malware Traffic Analysis Knowledge Dataset (MTA-KDD'19) and the dataset underwent preprocessing procedures including data cleaning, normalization, transformation, and class balancing using SMOTE. Three machine learning algorithms such as Decision Tree, Random Forest, and Artificial Neural Network (ANN) which were implemented and evaluated based on accuracy, precision, recall, and F1-score where the experimental results revealed that the Random Forest classifier achieved the highest performance accuracy of 99.93%, outperforming the Decision Tree and Neural Network models. The findings demonstrate that Random Forest provides superior generalization and robustness in classifying malicious and benign network traffic. The study concludes that machine learning models, particularly ensemble methods, can significantly enhance proactive threat detection and serve as a foundation for real-time cyber defence systems against botnet attacks.

Keywords: Botnet Detection; Machine Learning; Random Forest; Decision Tree; Artificial Neural Network (ANN); Cybersecurity

INTRODUCTION

Over the past few years, cyberspace has seen an upsurge in more advanced attacks, and botnet attacks are one of the most common attack models applied in the process of committing this criminal offence (Khan and Mailewa, 2023). A network of computers or devices that are controlled by one individual is called Botnet. These hacked machines are commonly called bots or zombies, and they can be remotely controlled as they can be infected with harmful software (Anwar and Saravanan, 2022).

A botnet is mainly used to carry out different coordinated operations by being undetected by the network owners (Mousavi et al., 2020). The attack used different kinds of malwares, including phishing, worms, Trojan horses, ransomware and spywares to attack the security of a network and take advantage of the vulnerabilities. Phishing attacks are carried out through botnets and include misleading strategies to defraud users into giving their sensitive information, whereas worms and Trojan horses are self-replicatory and maliciously code-carrying. Ransomware will however encrypt files which it will demand to be paid with and spy ware will monitor and collect user information discreetly.

A botnet threat has multidimensional effects that may have a devastating effect on the integrity, availability, and confidentiality of a system. BOTnet threat signs can consist of the presence of an unusual network traffic pattern, excessive use of bandwidth, slowdowns in the operation of the system without any apparent reason, the unauthorised access to sensitive information, and the apparatus increase (Joshi et al., 2022). Also, frequent system crashes, pop-ups, and changed browser settings are also characteristic of possible botnet activity (Lo et al., 2023). The most appropriate way of addressing risks posed by botnets is to establish a strong cybersecurity

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

system, perform a regular security assessment, and train users on the risks of recognising and preventing phishing attacks and suspicious activities. Early warning and anticipatory action are also essential in reducing the impact that the threats of botnets can cause (Habor et al., 2021).

Nasir et al. (2023) argue that such classical security measures as encryption, and botnet intrusion detection systems may not be able to maintain the pace with the dynamicism and variation of such threats. The attackers also use various tricks that include social engineering, zero-day exploits, and polymorphic malware, and it is difficult to foresee and prevent security breaches with the help of these tricks. Additionally, Ayo et al. (2023) found out that the mere number of gadgets linked to networks such as IoT devices under different security postures create new attack vectors. Security vulnerabilities of IoT devices, the lack of proper authentication systems, and absence of encryption algorithms provide attackers with the possibility to intrude and breach systems (Abrantes et al., 2021). These environments are so big and varied that it is hard to impose universal security controls on all equipment and platforms. Moreover, the safety of data transmission and storage, integrity of cloud-based programmes, and unauthorised access becomes a significant issue (Nazir et al., 2023).

Organisations require a complex solution to cybersecurity in order to deal with these issues. This involves the incorporation of the newest threat detection systems, anomaly detection through machine learning algorithms, and behaviour analytics in detecting patterns that may be indicator of a potential security incident (Ayo et al., 2023; Ulagwu-Echefu et al., 2021). This is necessary to reduce the effects of attacks and vulnerabilities in real time, because continuous monitoring and real-time response institutions are required. This study aims at exploring the different methods used in detection and control of botnet, their areas of weaknesses, and how the problem can be solved to future threats of botnet by critical infrastructures of the network.

METHODOLOGY

The methodology used for this research is abehavioural driven development approach. The reason was because it allows for quality assurance in realization of machine learning based projects. In addition, it allows the integration of ideas from diverse domain experts in interdisciplinary research to help in the realization of the new system. In achieving this, the steps to be applied are data collection of botnet threat features considering malware as the threat class. This data upon collection will undergo several processing steps such as visualization, imputation, normalization and transformation, before transferring into three suitable machine learning algorithms for training and then generate the model for threat classification.

Data Collection

The Kaggle Malware Traffic Analysis Knowledge Dataset 2019 (MTA-KDD'19) dataset, which has been updated and improved especially for training and assessing machine learning-based malware traffic analysis algorithms, is the dataset used for the system's implementation. In the UNSW Canberra Cyber Range Lab, a realistic network environment was designed in order to build the BoT-IoT dataset. Both regular and botnet traffic were present in the network environment. The source files for the dataset are offered in a variety of forms, such as the produced argus files, the original PCAP files, and CSV files. To aid in the tagging procedure, the files were divided into assault categories and subcategories. Legitimate traffic in the dataset originates from pcap files classified as Normal in MCFP. These files total more than 7GBytes in size and generated around 40,000 samples. Malware traffic originates from the MTA repository and consists of almost 4.8 GB of ≃ 2112 pcap files. The period covered by these observations is June 2013–August 2019. Although it is especially targeted at machine learning algorithms, the final dataset is not skewed by any one application, and the entire procedure may be automated to maintain its accuracy.

The dataset consists of network traffic attributes representing a number of different aspects of packet behaviour and flow properties. The TCP flag distributions (FinFlagDist, SynFlagDist, RstFlagDist, PshFlagDist, and AckFlagDist) indicate the frequency of occurrence of specific TCP control flags, which can be used to identify potential patterns of abnormal traffic or an attack pattern. The dominance of various network protocols in the captured flow is emphasized by the use of protocol-based features like DNSoverIP, TCPoverIP, and UDPoverIP. Whereas inter-arrival time values (MaxIAT, MinIAT, AvgIAT) are useful in analysing the time differences

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

between successive packets, which is useful in determining abnormalities such as denial-of-service attacks, packet length values (MaxLen, MinLen, StdDevLen, and AvgLen) provide information on the variability of the traffic.

Other flow-based variables such as FlowLEN, FlowLENrx (data on the total number of the flow and received flow), and NumPorts (data on the number of the unique ports utilized) provide an overview of connection dynamics. Several packet-level statistics, such as pktsIOratio, 1stPktLen and repeated_pkts_ratio, may indicate asymmetric traffic patterns, lost packets or retransmissions. The diversity of communication can be determined using such characteristics as NumCon (number of connections) and NumIPdst (number of distinct destination IPs) that are necessary to distinguish between reasonable surfing and potentially malicious activity. Finally, the label column serves as the classification target of the supervised learning activities, allowing the model to differentiate between malevolent and benign network flows. Start flow, DeltaTimeFlow and HTTPpkts provide information that is temporal and protocol specific.

Data Processing

To enhance the applicability of the proposed machine learning algorithms for threat classification, the collected data is prepared at this step using a variety of techniques. Data cleaning is the initial step in data processing, which includes identifying missing values, removing rows with missing values, and replacing them with mean or median values. The z-score standardisation technique is used in the data normalisation process (Deshmukh and Wangikar, 2011). The scale numerical characteristics are then subjected to data normalisation such that their range of values is comparable (Amato, 2023). Using One-Hot encoding, data transformation is similarly used to change categorical variables into numerical values (Samuels, 2024). Additionally, data augmentation is used to create synthetic examples for the minority classes using SMOTE and balance imbalanced datasets (Alberto et al., 2018). Finally, data is split into training and testing sets. The ratio of data split adopted in this study is 70:30 which is a common split ratio.

Neural Network

Neural network is a sort of artificial intelligence that seeks to emulate the way a human brain operates. A neural network functions by establishing connections between processing elements, which are the computer equivalent of neurones, as opposed to utilising a digital model where all calculations involve manipulating zeros and ones. The output is determined by the weights and organisation of the links (Islam et al., 2019).

An artificial neural network is a group of interconnected units, also known as neurones, that draws inspiration from the brain. Neurones can communicate with one another through their connections. The weight or strength of the signal is determined by a real number value carried by each connection (Islam et al., 2019; Sochima et al., 2025). The number of artificial neurones, or units, that make up a typical neural network can range from a few dozen to hundreds, thousands, or even millions. These units are stacked in layers, each of which links to the layers on either side. Some of them, referred to as input units, are made to take in different types of external information that the network will try to understand, identify, or process in some other way. Other units, referred to as output units, are located on the other side of the network and indicate how it reacts to the knowledge it has acquired. The bulk of the artificial brain is made up of one or more layers of hidden units that are positioned between the input and output units. The majority of neural networks have complete connectivity, meaning that every output and hidden unit is linked to every other unit in the layers on either side. A number known as a weight, which can be either positive (if one unit stimulates another) or negative (if one unit suppresses or inhibits another), represents the connections between one unit and another. One unit's impact on other increases with its weight. (This is similar to how real brain cells communicate with each other through microscopic openings called synapses) (Murphy, 2012; Chidi et al., 2024).

Stepwise of the neural network

The algorithm below outlines the step-by-step process of a feedforward artificial neural network.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

Step 1: initialize the network structure

Define the number of layers

Choose the number of neurons in each layer

Randomly initialize weights and bias for all neurons

Step 2: input data feeding

Provide the input vector to the input layer

These values are passed to the first hidden layer neurons

Step 3: forward propagation

Compute the weighted sum of inputs for each neuron in the hidden layer $z = w.x + b$

Where Z= weighted sum, b =bias, X = input vector and w = weighted vector apply an activation function(sigmoid) to get the output of each neuron repeat this for all neurons in the hidden layer the final output layer receives values from the last hidden layer, processes them and gives the prediction.

Step 4: compute the loss

Compare the predicted output with the actual output use a cross-entropy for classification as loss function

Step 5: backward propagation

Calculate the gradient of the loss with respect to each weight using the chain rule.

Compute how much each neuron contributed to the error.

Propagate the errors backward through the network.

Step 6: Update Weights and Biases

Use an optimization algorithm to update weights

Step 7: Repeat (Train)

Repeat steps 2 to 6 for multiple **epochs** until the network learns the patterns.

Step 8: Evaluate the Model

After training, evaluate the ANN on unseendata.

Measure performance using accuracy and F1-score.

Step 9: stop

Decision Tree

Fayang (2016) explains that a decision tree is typically a predictive model that is developed based on a tree-time structure in the machine learning domain. Its inner nodes normally examine a characteristic (characterized by Liduo, 2016), whereas its outer ones are the final classification. The model is able to tackle many of the underlying problems (Mingyue, 2016), including optimization issues, multi-step decision-making problems, etc. It also plays a part in reproducing the process of decision making. Moreover, it is also capable of decomposing complex processes into a variety of simple decisions (Yanli, 2015), which in its turn can provide a transparent

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

overview of the entire decision-making process. A decision tree is simply a tree structure that is used in sorting out cases.

In the context of machine learning and data mining (Zhiyong in 2015), the complexity of the machine learning algorithms can become an obstacle in the issue of understanding, and the desire to match these expectations is not always fulfilled. Nonetheless, decision tree model is a good solution. One of the machine learning algorithms that are popular is decision tree algorithms. Decision trees are an ever-changing field and, as a result, give rise to various; several algorithms have been developed. The main point of Classification is to determine to which of the predefined category an object belongs to. It is not only that classification is a widespread problem, but a basis to more complicated problems in the making of choices. It is also the most eminent algorithm family of machine learning and data mining technology (Zhiyong 2015 research).

Stepwise decision tree algorithm

The algorithm below outlines the step-by-step process of a decision tree algorithm

Step 1: Start at the Root Node

1. Evaluate all features in the dataset to determine the best one for the initial split using a splitting criterion.

Step 2: Split the Dataset at the Root

1. Based on the selected feature, divide the dataset into subsets.
2. Create a **decision node** for each resulting condition or value.

Step 3: Build Subtrees Recursively

For each subset, repeat the process:

i. Evaluate the best feature for splitting.

ii. Create a **new decision node**.

iii. Split again as necessary.

This forms a **subtree**, like the left side of your image.

Step 4: Continue Splitting Until a Stopping Condition is Met

Stop splitting if:

i. All samples in the node belong to the same class.

ii. No remaining features to split.

iii. Stopping rule was met

Step 5: Assign Leaf Nodes

Once a node cannot be split further, mark it as a **leaf node**.

Assign the most frequent class label for classification

Step 6: Repeat the Process for All Branches

Apply steps 3–5 to all branches until the entire tree is built.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

This includes nested decision nodes, like the bottom level of your image.

Step 7: Use the Tree for Prediction

To make a prediction for a new instance:

i. Start at the root.

ii. Follow the decision path based on feature values.

iii. Return the class or value at the reached **leaf node**.

Step 8: stop

Random Forest

In order to generate hypotheses that aid in decision-making in the future, predictive models are built by examining the characteristics of predictive factors. By examining the properties of the variables used for forecasting, predictive models are created, and the outcomes are hypo theses that can be investigated empirically. The accuracy of these models depends on methods for estimating errors. While predictive data mining uses supervised machine learning techniques, meta datamining frequently uses unsupervised techniques. Several decision trees are generated in order to create random forests. This is accomplished by selecting input features at random and collecting random data samples using Bootstrap samples.

One can arrive at a conclusion considerably faster by removing unnecessary branches. The tree's root no de is the parent node, and its children are the other nodes. When the main tree is split up, new branches and subtrees are produced, creating a subtree. The goal variable distinguishes the two main types of decision trees that are included in machine learning (Salman et al. 2024).

Stepwise random forest algorithm

The algorithm below outlines the step-by-step process of a random forest algorithm

Step 1: Prepare the Dataset

Start with the complete training dataset containing input features and corresponding output labels.

Step 2: Create Multiple Bootstrap Samples

Randomly select multiple subsets of the datato create different training sets for each decision tree. This is known as **bootstrap sampling**.

Step 3: Grow Each Decision Tree

For each bootstrap sample, build a **decision tree**:

i. At each node, choose a random subset of features.

ii. Select the best feature from this subset using a splitting criterion (Information Gain).

iii. Split the data based on that feature and continue recursively until a stopping condition is met (min samples).

Step 4: Make Predictions with All Trees

To classify a new instance:

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

i. Pass the instance through **each decision tree** in the forest.

ii. Each tree gives a predicted class.

Step 5: Perform Majority Voting

Collect the predicted classes from all trees.

Use **majority voting** to decide the final class label (i.e., the class with the most votes becomes the final prediction).

Step 6: Output the Final Class

The class label determined by the majority vote becomes the final predicted class for the input instance.

Step 7: stop

System Implementation

The initial process in the training of machine learning models (random forest, decision tree, and neural network) is to prepare the data in the Google Collab environment through Python programming language. The first step in this case is the dataset processing, which involves breaking it down into features (X) and target (y), filling the blank values, and encoding categorical variables. The normalisation then follows to ensure that the characteristics are normalised in a similar manner which is essential in models like the neural networks. After data preparation, a usual split of 70/30 is done to divide the data into training and testing sets. The training data is then used to train the models and this involves optimising an internal parameter(weights or splits) of the model to the training data. This is repeated till the model reduces the error or is at an acceptable level of performance. The implementation area layout is given in Figure 1 in which the algorithms will be trained.

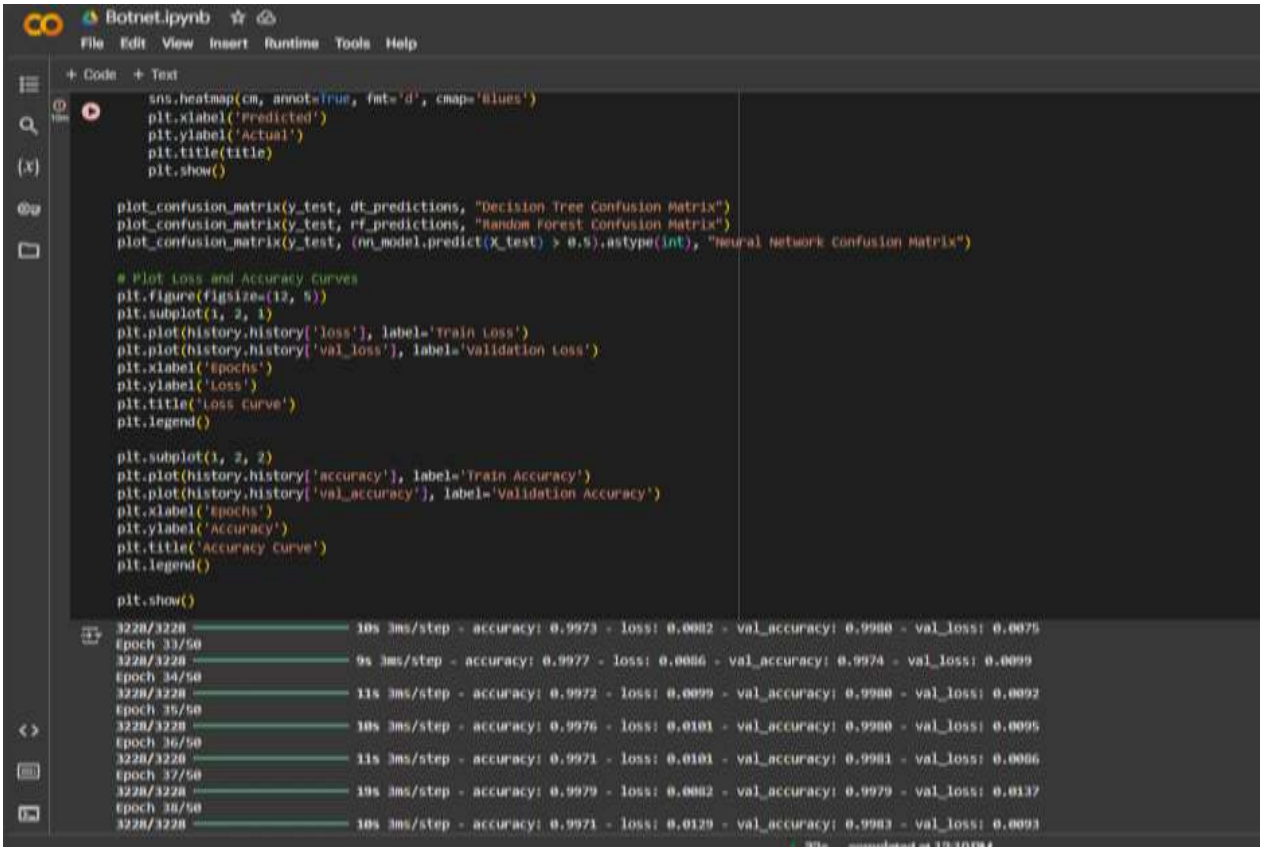

Figure 1: System Implementation environment

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

Assessing the models' performance on the testing set comes next after they have been trained. Metrics like accuracy, precision, recall, and F1-score are used to evaluate the predictions made by the trained model with the actual values of the target variable on the test data.

SYSTEM RESULTS

This section presents the comprehensive performance outcomes of the various proposed machine learning algorithms on proposed dataset.

Results of the Decision Tree Algorithm

The Decision Tree model was a very predictive model with an incredible accuracy rate of 99.99% when applied on the given data. Another fact that supports this is the classification report as the values of the precision, recall and F1-score of both classes (0 and 1) are all 1.00(or close to 1). A precision of 100 means that all of the cases that were predicted to fall into a particular classification were correct and a recall of 100 indicates that the model could find all cases of every classification correct. It is also equally good in identifying true positives and deceiving negatives, the F1-score, which is a balance of accuracy and recall is also 1.00.

When trained on highly structured or unbalanced data, decision trees have a tendency to memorise patterns, which might result in high accuracy on the test set but perhaps poor generalisation to new data. Additional validation using cross-validation, pruning strategies, or ensemble approaches like Random Forest may be helpful to guarantee robustness. To ascertain whether the model is indeed generalising successfully, it might also be helpful to analyse cases that were incorrectly categorised, if any, and assess performance on a different dataset. The performance accuracy of the decision tree algorithm is presented in Figure 2

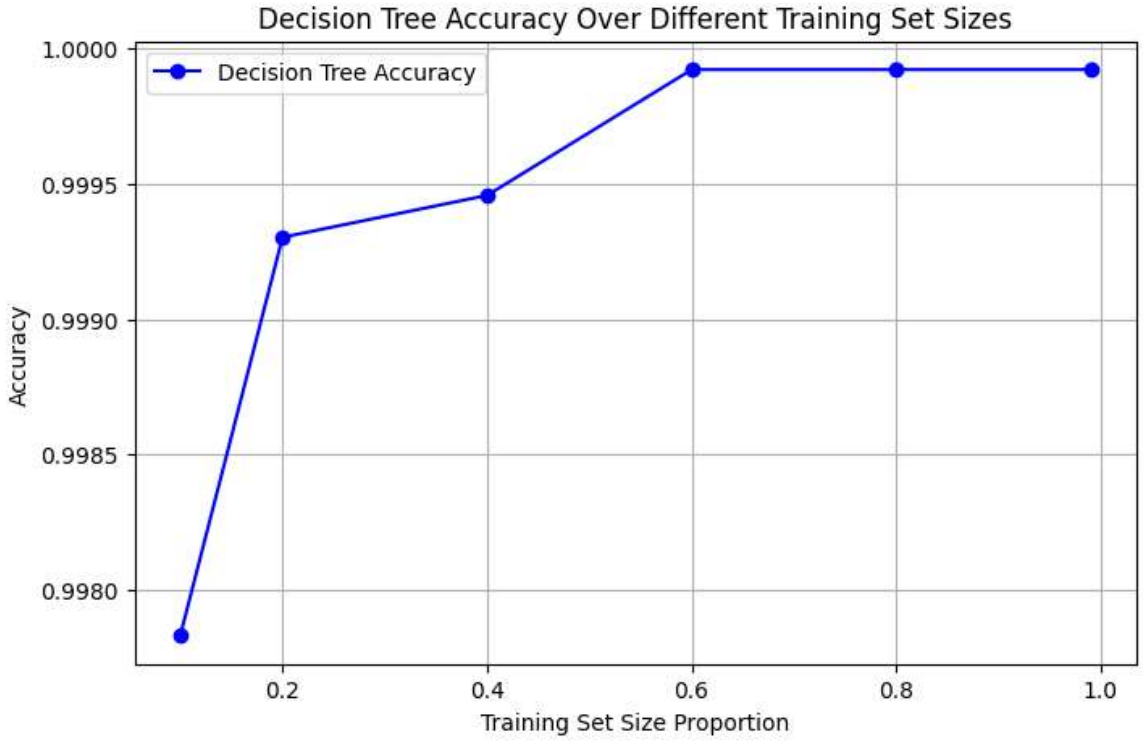

Figure 2: Decision Tree Accuracy Results

The classification performance of the Decision Tree model is visually represented by the confusion matrix in Figure 3. It demonstrates that there were no false positives since the model properly identified 6,062 occurrences of class 0 and 6,848 cases of class 1. One false negative occurred, though, when a class 1 instance was mistakenly identified as class 0. This yields a 99.99% total accuracy, demonstrating the model's almost flawless forecasting power.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

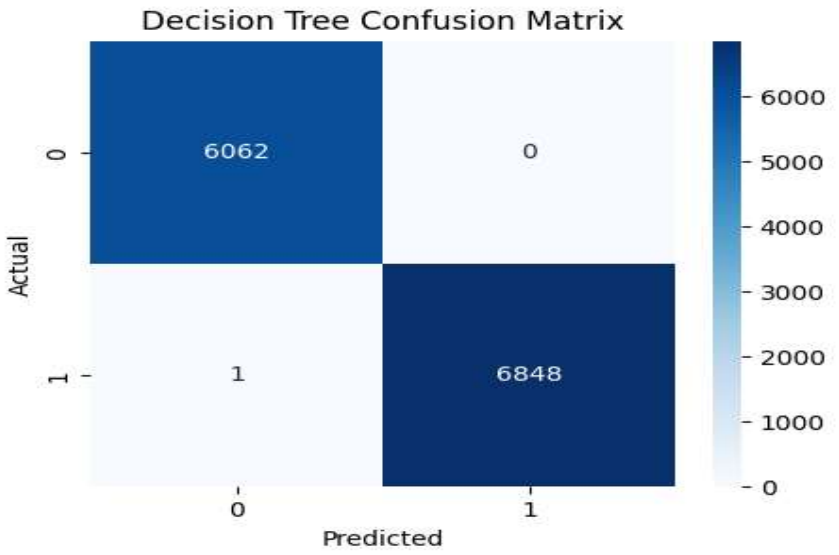

Figure 3: Decision Tree Confusion Matrix

As seen in Figure 3, the model performs exceptionally well in differentiating between the two classes because there are no false positives (misclassifying class 0 as class 1) and just one false negative. Overfitting, which occurs when a model memorises patterns unique to a dataset rather than adapting effectively to fresh, unseen data, is a danger raised by such high accuracy. Techniques like feature significance analysis, cross-validation, and testing on a separate dataset should be taken into consideration in order to further verify the model. The performance of the decision tree training outcomes is shown in Table 1.

Table 1: Performance Result of Decision Tree

Metric	**Class 0**	**Class 1**	**Overall**
True Positives (TP)	6,062	6,848	12,910
False Positives (FP)	0	0	0
False Negatives (FN)	1	0	1
Precision	1.00	1.00	1.00
Recall	1.00	1.00	1.00
F1-Score	1.00	1.00	1.00
Accuracy	-	-	99.99%

Despite these excellent results presented in Table 1, such near-perfect scores may indicate overfitting to the training data.

Results of the Random Forest Algorithm

With an overall accuracy of 99.93%, the Random Forest model's findings show remarkable performance. As can be seen in Table 4, the classification report displays precision, recall, and F1-score values of 1.00 for both classes, suggesting that the model identified almost all occurrences correctly. In particular, the model showed 100% accuracy and recall for Class 0 (negative cases), which means that neither false positives nor false negatives occurred. Similarly, for Class 1 (positive instances), the recall score of 1.00 confirms that nearly all actual positives were identified correctly, with minimal misclassification. The macro and weighted averages also

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

indicate balanced performance across both classes, reinforcing the robustness of the model. The Random Forest performance accuracy over different training set sizes is presented in Figure 4.

Figure 4: Random Forest Accuracy Results

Table 2: Performance Results of Random Forest Algorithm

Metric	**Class 0 (Negative)**	**Class 1 (Positive)**	**Macro Avg**	**Weighted Avg**
Precision	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00
F1-Score	1.00	1.00	1.00	1.00
Support	6062	6849	12911	12911
Accuracy	99.93%	99.93%	99.93%	99.93%

Even though Table 2 displays nearly flawless results, this high accuracy might be a sign of overfitting, in which the model retains patterns from the training set instead of effectively generalising to new data. To make sure the model is still applicable in practical situations, feature selection or dimensionality reduction strategies might be investigated if the dataset includes highly correlated features.

Figure 5 displays the Random Forest classifier's confusion matrix, which shows that the model achieves near-perfect accuracy and performs remarkably well in categorising both classes. 6,058 occurrences of Class 0 (negative cases) and 6,844 instances of Class 1 (positive cases) are accurately classified. Nevertheless, there are five false negatives (where Class 1 was mistakenly categorised as Class 0) and four false positives (where Class 0 was mistakenly classified as Class 1). The incredibly low error rate indicated by these misclassifications shows that the model is capable of strong generalisation and successfully separates the two classes.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

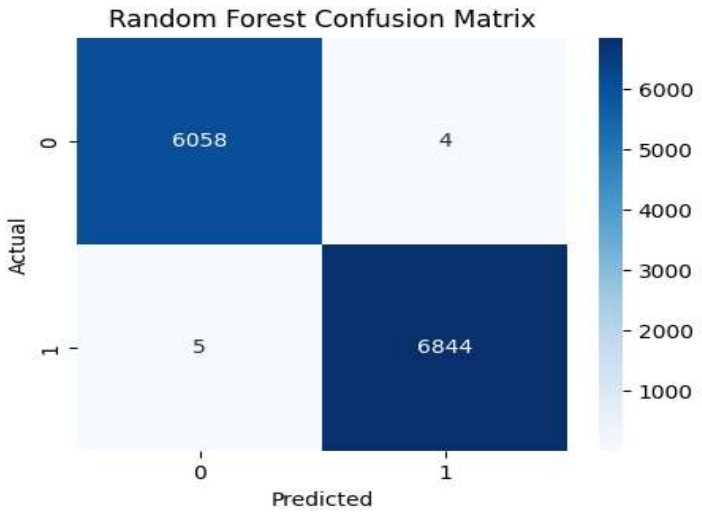

Figure 5: Random Forest Confusion Matrix

The existence of false positives and false negatives, as seen in Figure 5, indicates areas where minor enhancements might be made despite its remarkable performance. While false negatives could result in the loss of significant cases that ought to have been detected, false positives could result in needless alerts or interventions. However, the Random Forest model turns out to be a very dependable classifier due to its high precision and recall values, as well as its overall accuracy of 99.93%. Its performance indicates that it can handle intricate data patterns with a small error margin.

Results of the Neural Network Algorithm

The ANN algorithm's results show how well a neural network model trained for classification performs. With a corresponding loss of 0.0079 and an accuracy of 99.87% on the last training session, the model appears to have learnt to discriminate between classes with a very high degree of precision. The model's good generalisation to unknown data is confirmed by the total test accuracy of 99.85%, which shows little overfitting. The low loss number, which indicates low prediction mistakes, lends more credence to the idea that the model is well-optimized.

Given its high accuracy, the model appears to be quite successful in classifying the input data. But even with these encouraging outcomes, it's crucial to make sure the model hasn't just internalised the training data. To verify whether the model is indeed operating as anticipated, more research would be helpful, such as examining confusion matrices and misclassification rates. Figure 6 displays the neural network implementation's loss and accuracy curves across 50 epochs.

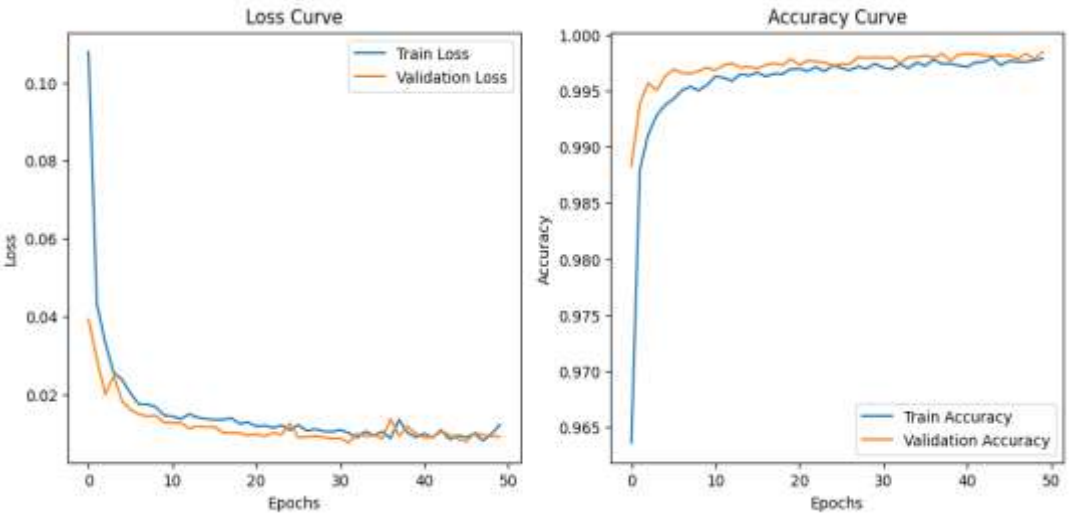

Figure 6: Loss and Accuracy Curve of the ANN algorithm

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

A thorough understanding of the model's performance during training is offered by the loss and accuracy curves shown in Figure 6. The training and validation losses dramatically drop across the epochs in the loss curve (left), suggesting that the model is successfully picking up on the patterns in the data. There is little overfitting, indicating that the model generalises effectively to new data, as seen by the strong relationship between validation loss and training loss.Both training and validation accuracy increase gradually across the epochs, approaching 99.85% accuracy on the accuracy curve (right).

The neural network model's confusion matrix offers details about how well it performs in classification. The model showed a high degree of accuracy by accurately classifying 6832 cases of class 1 and 6059 instances of class 0. Three false positives, however, occurred when the model predicted class 1 instead of class 0, and seventeen false negatives occurred when the model predicted class 0 instead of class 1. The fact that these mistakes are negligible in relation to the entire sample size indicates that the model's predictions are quite dependable. The neural network implementation's confusion matrix is displayed in Figure 7.

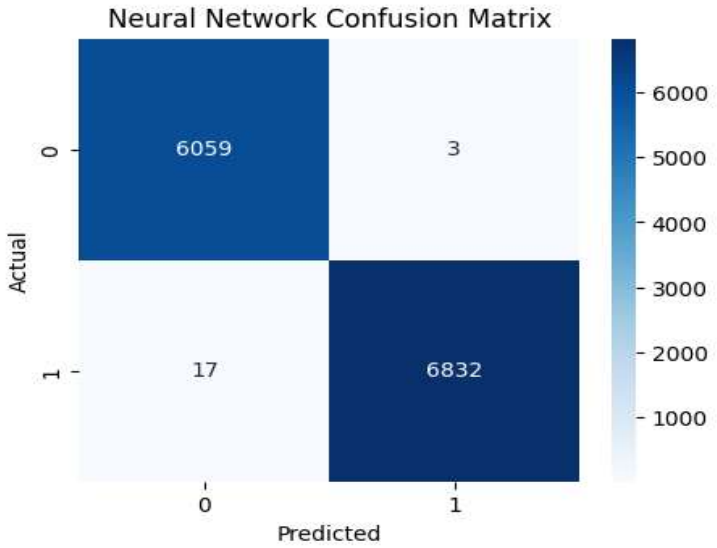

Figure 7: Neural Network Confusion Matrix

With very few misclassifications, the model performs remarkably well, according to the general trend in the confusion matrix shown in Figure 7. The model may be a little more conservative in forecasting the positive class since the number of false negatives is somewhat larger than the number of false positives. Nonetheless, this low misclassification rate is unlikely to have a major influence on the model's efficacy considering its total accuracy of 99.85%. Techniques like modifying class weights or fine-tuning the classification threshold might be investigated to further improve the model and lower the tiny percentage of false negatives.

CONCLUSION

The study presents the effectiveness of machine learning techniques for real-time cyber defenceon wireless networks. By collecting and processing botnet attack data from Kaggle, the study developed and trained three machine learning models such as Decision Tree, Random Forest, and Neural Network for the detection and mitigate cyber threats.The study compared their performances in terms of accuracy, precision, recall and F1-score. The comparative analysis of these models revealed that the Random Forest classifiers achieved highest performance accuracy and reliability among the other models in detecting malicious activities within the network infrastructure.

The results underscore the importance of integrating machine learning into cybersecurity frameworks for proactive threat detection. The recommended random forest model can serve as a foundation for developing a real-time defence mechanism, improving the security of wireless networks against botnet attacks. Future work may focus on optimizing these models for large-scale deployments and incorporating adaptive learning techniques to enhance resilience against evolving cyber threats.

REFERENCES

1. Abrantes, R., Mestre, P., & Cunha, A. (2021). Exploring dataset manipulation via machine learning for botnet traffic. CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2021.
2. Anwar, F., & Saravanan, S. (2022). Comparison of artificial intelligence algorithms for IoT botnet detection on Apache Spark platform. Procedia Computer Science, 215, 499–508. https://doi.org/10.1016/j.procs.2022.12.052
3. Ayo, F., Awotunde, J., Folorunso, S., Adigun, M., & Ajagbe, S. (2023). A genomic rule-based KNN model for fast flux botnet detection. Egyptian Informatics Journal, 24(2), 313–325. https://doi.org/10.1016/j.eij.2023.05.002
4. Cabello-Solórzano, K., Ortigosa de Araujo, I., Peña, M., Correia, L., & Tallón-Ballesteros, A. J. (2023). The impact of data normalization on the accuracy of machine learning algorithms: A comparative analysis. In P. García Bringas et al. (Eds.), 18th international conference on soft computing models in industrial and environmental applications (SOCO 2023) (pp. 373–382). Springer. https://doi.org/10.1007/978-3-031-42536-3_33
5. CHIDI, E. U., UDANOR, C. N., & ANOLIEFO, E. (2024). Exploring the Depths of Visual Understanding: A Comprehensive Review on Real-Time Object of Interest Detection Techniques. Preprints. https://doi.org/10.20944/preprints202402.0583.v1
6. Dai, Y. (2015). Analysis of decision tree algorithm in data mining and its application. Science and Technology Communication, 7(23), 33–34.
7. Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research, 61, 863–905. https://doi.org/10.1613/jair.5682
8. Harbor M.C, Eneh I.I., Ebere U.C. (2021). Nonlinear dynamic control of autonomous vehicle under slip using improved back-propagation algorithm. International Journal of Research and Innovation in Applied Science (IJRIAS); Vol. 6; Issue 9; https://rsisinternational.org/journals/ijrias/DigitalLibrary/volume-6-issue-9/62-68.pdf
9. Hou, L. (2016). Application study of decision tree algorithm in engineering quality supervision decision support system [Doctoral dissertation, Guizhou University].
10. Joshi, C., Ranjan, R., & Bharti, V. (2022). A fuzzy logic based feature engineering approach for botnet detection using ANN. Journal of King Saud University – Computer and Information Sciences, 34(8), 6045–6056. https://doi.org/10.1016/j.jksuci.2021.06.018
11. Khan, S., & Mailewa, A. (2023). Discover botnets in IoT sensor networks: A lightweight deep learning framework with hybrid self-organizing maps. Microprocessors and Microsystems, 97, Article 104753. https://doi.org/10.1016/j.micpro.2022.104753
12. Li, M. (2016). Application of decision tree algorithm in bank telemarketing [Doctoral dissertation, Huazhong University of Science and Technology].
13. Li, W. (2014). Application and parallel study of decision tree algorithm [Doctoral dissertation, University of Electronic Science and Technology]. Advances in Intelligent Systems Research, 161.
14. Lo, 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, Article 100747. https://doi.org/10.1016/j.iot.2023.100747
15. Mohaiminul, I., Chen, G., & Jin, S. (2019). An overview of neural network. American Journal of Neural Networks and Applications, 5(1), 7–11. https://doi.org/10.11648/j.ajnna.20190501.12
16. Mousavi, S., Khansari, M., & Rahmani, R. (2020). A fully scalable big data framework for botnet detection based on network traffic analysis. Information Sciences, 512, 629–640. https://doi.org/10.1016/j.ins.2019.10.018
17. Murphy, K. P. (2012). Machine learning: A probabilistic perspective. The MIT Press.
18. Nasir, M., Arshad, J., & Khan, M. (2023). Collaborative device-level botnet detection for internet of things. Computers & Security, 129, Article 103172. https://doi.org/10.1016/j.cose.2023.103172
19. Nazir, A., He, J., Zhu, N., Wajahat, A., Ma, X., Ullah, F., Qureshi, S., & Pathan, M. (2023). Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets.

Journal of King Saud University - Computer and Information Sciences, 35(10), Article 101820. https://doi.org/10.1016/j.jksuci.2023.101820

20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html
21. Salman, H., Kalakech, A., & Steiti, A. (2024). Random forest algorithm overview. Babylonian Journal of Machine Learning, 2024, 69–79. https://doi.org/10.58496/BJML/2024/007
22. Samuels, J. A. (2024). One-hot encoding and two-hot encoding: An introduction [Preprint]. ResearchGate. https://doi.org/10.13140/RG.2.2.12345.67890 (Note: Preprint DOI; full publication pending).
23. Sochima V.E. Asogwa T.C., Lois O.N. Onuigbo C.M., Frank E.O., Ozor G.O., Ebere U.C. (2025)"; Comparing multi-control algorithms for complex nonlinear system: An embedded programmable logic control applications; DOI: http://doi.org/10.11591/ijpeds.v16.i1.pp212-224
24. Song, Y. Y., & Lu, Y. (2015). Decision tree methods: Applications for classification and prediction. Shanghai Archives of Psychiatry, 27(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044
25. Ulagwu-Echefu A., Eneh. I.I. Ebere U.C. (2021). Enhancing realtime supervision and control of industrial processes over wireless network architecture using model predictive controller. International Journal of Research and Innovation in Applied Science (IJRIAS); vol 6; Issue 9. https://rsisinternational.org/journals/ijrias/DigitalLibrary/volume-6-issue-9/56-61.pdf