Volume 5, Issue III, March, 2026; No. 87, pp. 1106-1121
Submitted 2/2/2025; Final peer review 7/4/2026
Online Publication 12/4/2026
Available Online at http://www.ijortacs.com

ENSEMBLE MACHINE LEARNING-BASED BOTNET DETECTION AND REAL-TIME MITIGATION SYSTEM

[1]Nkechi Oji, [2]Ogochukwu Okeke C., [3]Ike J. Mgbeafulike
[1]Department of Computer Engineering, Federal University of Technology, Owerri (FUTO), Imo State, Nigeria.

[1,2,3]Department of Computer Science; Chukwuemeka Odumegwu Ojukwu University, Uli, Anambara State, Nigeria
Author Email: [1]nkechi.oji@futo.edu.ng, [2]ogoookeke@yahoo.com

Corresponding Author's Email: [1]nkechi.oji@futo.edu.ng

Abstract

This study developed an effective botnet detection and management system using machine learning techniques. An ensemble model combining Artificial Neural Network (ANN), Support Vector Machine (SVM), and Decision Tree (DT) algorithms was employed to accurately identify botnet activities. Datasets from the University of Nigeria, Nsukka, and CTU-13 were used, encompassing multiple botnet types and relevant network features. Feature selection and transformation techniques, including analysis of variance (ANOVA) and Principal Component Analysis (PCA), optimized the datasets for model training. A decision-based algorithm was incorporated to isolate infected devices and generate real-time alerts, enhancing network security. Experimental results showed that ANN and SVM achieved high individual accuracies of 0.98, while DT achieved 0.79. The ensemble model outperformed individual classifiers, achieving an accuracy of 0.99, recall of 0.99, and precision of 0.96, demonstrating superior reliability in detecting botnet threats. Comparative evaluation with existing approaches indicated that the proposed system not only delivers high detection rates but also integrates incident response for real-time threat isolation and logging, improving practical cybersecurity performance. In conclusion, the study demonstrates that ensemble machine learning, combined with effective feature engineering and real-time mitigation mechanisms, significantly enhances botnet detection in dynamic network environments. The findings underscore the value of leveraging complementary strengths of multiple classifiers, while future work could focus on detecting emerging botnet variants and further improving model adaptability for evolving cybersecurity challenges.

Keywords: Botnet Detection, Machine Learning, Ensemble Model, ANN, SVM

1. INTRODUCTION

The term "botnet" refers to a connected network of malware-infected devices that are controlled by hackers (Lo et al., 2023). To put it simply, a botnet is a robot network of compromised

devices that cybercriminals frequently use for a variety of cybercriminal activities (Nasir et al., 2023). In other words, it is a network of compromised computers, referred to as "bots" or "zombies," that are under the control of a single entity, typically a malicious actor or a group of cybercriminals. When the attacker initiates the threat, through the controlled server, multiple botnet nodes (zombies) are used to multiply the threat and then send to the victim server which is overwhelmed and then shutdown. These compromised computers are often infected with malware that allows the attacker to control them remotely (Abrantes et al., 2021).The individuals or organizations behind botnets use them for various malicious purposes, including launching coordinated attacks, distributing spam, conducting distributed denial-of-service (DDoS) attacks, and other forms of cybercrime (Arshad et al., 2023). The operator of the botnet can exercise control through the use of command and control (C&C) software, enabling them to orchestrate the actions of the compromised devices via communication channels established through standard network protocols (Moorthy and Nathiya, 2023).

Over the years, Machine learning (ML) has emerged as the dominant approach in solving various problems, such as pattern recognition, clustering, time series, and fitting tasks. Its success can be attributed to the high success rate of its algorithms and their reliability when applied in real-world scenarios. Notably, ML applications for addressing cyber threats have become a prevailing trend, with many algorithms employed for the classification and prediction of threats in wireless networks (Manasrah et al., 2022).

According to Joshi et al. (2022), ML involves extracting valuable information from vast repositories of data and leveraging this knowledge to make accurate decisions that can effectively solve problems. In the context of wireless network security, a multitude of ML algorithms, including but not limited to support vector machine, k-NN, decision trees, deep learning, Naïve Bayes, and others, have been utilized. These algorithms which presents the taxonomy of ML algorithms. Across various studies, they have all demonstrated acceptable success rates in dealing with threats in wireless networks, although there is still room for further improvements.

Researchers have explored various methodologies, including artificial intelligence (AI), deep learning, machine learning (ML), and ensemble techniques, to fortify networks against botnet intrusions. A prevalent theme across studies is the utilization of diverse datasets such as CTU-13, ISOT, and N-BaIoT, each containing distinct attributes and attack scenarios. Techniques like support vector machines (SVM), decision trees, and hybrid quantum-classical deep learning have emerged as promising approaches, offering high accuracy rates ranging from 92% to 99.7%. Moreover, innovative strategies like edge computing, ensemble classification and reinforcement learning have been proposed to bolster IoT device security and mitigate the impact of botnet threats effectively.Furthermore, studies have addressed the evolving landscape of distributed denial-of-service (DDoS) attacks, particularly in IoT and wireless network environments. Researchers have explored a plethora of ML algorithms such as Naïve Bayes, random forest, and K-nearest neighbors (KNN) to detect and mitigate DDoS intrusions. Applying techniques like autoencoders, deep neural networks, and artificial bee colony-BP neural network algorithms,

detection accuracy rates have been enhanced, exceeding 90% in several instances. Moreover, the development of real-time detection schemes and the integration of novel protection strategies underscore the continuous efforts to fortify network infrastructures and mitigate the disruptive effects of DDoS attacks on critical systems.

In addition, studies have emphasized the importance of dataset quality, model reliability, and real-time response in combating evolving cyber threats. Techniques like signature-based intrusion detection, ensemble machine learning, and quantum-classical hybrid models showcase the multifaceted approaches adopted to address the dynamic nature of cyber threats. However, challenges persist, including the need for improved dataset manipulation, enhanced training data quality, and the exploration of novel detection paradigms such as dynamic hashing and database fragmentation. As research endeavors progress, the synergy between AI, ML, and cybersecurity frameworks offers promising avenues for fortifying network resilience and thwarting malicious activities in an increasingly interconnected digital landscape.

Many works have been presented on cyber threats detection systems, considering various botnet attack model. Among the studies, one of the most recent is Ayo et al., (2023) who applied genomic rule-based KNN model for fast flux detection of botnet, however despite the ability of the proposed machine learning model, the study did not consider the dynamic characteristics of botnet and this remained a research gap.

2. RESEARCH METHODOLOGY

The methodology used for this work is a combination of agile and method. In realizing this methodology first a data model which captures the dynamic characteristics of Botnet will be developed using data augmentation approach, then the collected data will be used to train selected machine learning algorithms such as artificial neural network, decision tree and support vector machine respectively to generate an ensemble model for the classification of botnet. The model will be evaluated with various metrics which define success of cyber security model and analyze the results for recommendations to facilitate the design of future cyber security frameworks. To validate the model, comparative analysis will be performance with other state of the art algorithms and the results will be discussed to determine the best.

2.1 Data collection

Data of botnet used for this project is collected from African center of excellence, ICT Lab, University of Nigeria, Nsukka. This serves as the primary source of data collection. The data contain six classes of botnet such as dynamer, taoboa, open candy, cridex, dridex and yakes features. The sample size is 13299. The secondary data set is the CTU-13 Botnet data collected from Kaggle repository. The CTU-13 Botnet Dataset is a well-known dataset designed for the evaluation of botnet detection systems. It was created by the Czech Technical University (CTU) and contains labeled network traffic data captured during real-world botnet infections. The dataset includes various botnet attack scenarios, featuring network traffic from multiple botnet families such as the Dynamer, Taobao, and Dridex botnets, among others. It consists of 10 different attributes, each with a combination of benign and malicious traffic, capturing a wide range of botnet activities totaling a sample size of 298,00 features. The dataset provides detailed

network traffic features like packet sizes, flow duration, protocol types, and other relevant attributes, making it a valuable resource for training and testing machine learning models for botnet detection. The overall numbers of features are 43099 features of botnet.

2.2 The Analysis of Variance (ANOVA) for feature selection

The ANOVA feature selection method is a statistical approach used in this work to identify the most relevant botnet features in the dataset. It works by comparing the variances of continuous features across different classes of the target variable. It operates by assessing how much a feature can distinguish between the different classes by calculating the ratio of variance between the classes (inter-class variance) to the variance within each class (intra-class variance). Features that exhibit a large difference between classes and minimal variation within each class are deemed more informative. This is quantified using the F-statistic, where a higher value indicates a stronger ability to differentiate the classes. Features with the highest F-statistic values are selected for model training, enabling the creation of a more efficient and accurate model by focusing on the most significant predictors.

Algorithm 1: The ANOVA feature selection algorithm

1. Start
2. Input data
3. Divide the data: The data is split based on the target variable into different groups.
4. Calculate the variance: For each feature, calculate intra-class variance
5. Compute the F-statistic: $F = \frac{variance\ between\ group}{Variance\ \ within\ group}$
6. Rank the features: The features are ranked based on their importance.
7. Select the best features
8. Return as feature output
9. End

2.3 Principal Component Analysis (PCA) for feature transformation

Principal Component Analysis (PCA) operated by standardizing the data first, to ensure each feature contributes equally. Then, the covariance matrix is calculated to examine relationships between features. The eigenvalues and eigenvectors of this matrix are computed, with the eigenvectors representing the directions (principal components) of maximum variance. These components are ranked by eigenvalues, and the top components are selected to reduce the dimensionality of the dataset. The data is then projected onto these principal components, creating a smaller, more informative set of features that can be used for further analysis or model training, thus improving efficiency and performance.

2.4 Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) consists of interconnected layers of neurons (nodes), where each neuron processes information and passes it to the next layer. The model uses a training process called back propagation, where the error between the predicted and actual outputs is propagated back through the network, adjusting the weights of connections to minimize the error.

Algorithm 2: Artificial Neural Network (ANN) Training

1. Input data
2. Normalize the data and split it into training, validation, and testing datasets.

3. Network Architecture% Define the architecture of the neural network,
4. Forward Propagation% Pass input data through the network..
5. Loss Calculation % Calculate the loss
6. Back-propagation % Compute the gradients of the loss function
7. Update the weights and biases using an optimization algorithm
8. Model Evaluation
9. Validate the model on the validation set after each epoch to monitor performance.
10. Meeting stopping criteria
11. Generate model

2.5 Support Vector Machine

Support Vector Machine (SVM) works by finding the hyperplane that best separates data points of different classes with the maximum margin. It differentiate between benign and malicious network traffic by mapping input features to a higher-dimensional space and identifying the optimal boundary between classes. The model uses support vectors, the data points closest to the decision boundary, to define this margin, and the result is the robust classifier model.

Algorithm 3: Stepwise Training of the SVM Algorithms

1. Input data
2. Normalize the input data
3. Split the dataset into training and testing sets.
4. Select Linear Kernel Function
5. Solving the Optimization Problem
6. Use a quadratic programming solver to find the optimal values for the weight vector
7. Model Evaluation
8. Validate the trained model
9. Adjusting the regularization and the kernel parameters
10. Generate the model
11. End

2.6 Decision Tree

A Decision Tree (DT) is a tree-like model used for classification and regression tasks. It splits the dataset into subsets based on feature values, creating branches that lead to decision nodes, ultimately ending in leaves representing the predicted class. For botnet detection, the tree's nodes represent tests on different network features (e.g., traffic volume, packet type), and the branches represent outcomes based on these tests. The model recursively partitions the data to find the best feature splits, which help classify traffic as either benign or indicative of botnet activity. Decision Trees are easy to interpret, making them valuable for explaining the reasoning behind botnet detection.

Algorithm 4: Decision Tree (DT) Training

1. Start
2. Input Data
3. Split the dataset into training and validation sets.
4. Root Node Selection
5. Start with the root node and compute the Gini Index for all features.

6. Recursive Splitting% Repeat the process for each subset
7. Pruning% Apply pruning to remove branches that do not improve accuracy
8. Model Evaluation
9. Generate classifier
10. End

2.7 Ensemble Model

An Ensemble Model combines multiple individual models (ANN, SVM, and DT) to improve the accuracy and robustness of predictions. It works by aggregating the outputs of various classifiers to make a final decision, using method majority voting leading to a more accurate and reliable botnet detection system.

2.8 Botnet Detection Classifier

The Botnet Detection Classifier works by analysing various network features, such as packet size, frequency of requests, and patterns of communication between devices. The classifier is trained using botnet data, where the traffic is marked as either "botnet-related" or "benign." Based on these labels, the classifier learns to detect anomalies in the traffic that suggest botnet behaviour and flags suspicious activities for further investigation.

Algorithm 5: Decision-Based Algorithm for Botnet Isolation and Alert Generation

Assumptions:

Input: Threshold Values: if the confidence level is above 90%, the prediction is considered valid).

Output: Isolation Decision: Whether the device should be isolated from the network.

Alert Generation: A message or notification to be sent to the administrator about the infected device.

Algorithm Steps:

1. Start
2. Receive Classification Model Output
3. Check Confidence Threshold
   - If
     - Confidence score is above the threshold% (Greater than 90%)
       - Trigger alert response
   - Else If
     - Confidence score is below the threshold
       - Flag the result as inconclusive
   - Return to step 2.
4. Isolate Infected Device
5. For devices identified as Infected
   - Block IP
   - Disconnect from Network
6. Log the Incident
   - Record the details of the infected device: IP address, Timestamp
7. Generate an alert with log information
8. Notify Network Administrators
9. Send the alert to the administrators dashboard

10. End

4.7 System Implementation

The implementation of the botnet classifier system in PYTHON involves designing a robust framework for detecting and classifying botnet activities using machine learning techniques. The system begins by importing input data collected from user devices, such as network traffic logs or behavioral metrics. The feature selection and extraction process uses PYTHON's built-in tools like `pca()` for dimensionality reduction and custom scripts for extracting statistical features, including packet rate, connection duration, or unusual resource usage. Pre-processing steps, such as normalization, are performed to standardize the data, ensuring efficient model training. PYTHON's `fitensemble()` function or the Classification Learner toolbox is utilized to develop an ensemble model comprising machine learning classifiers like DT, SVM, and ANN. A voting mechanism is integrated to combine the outputs of individual classifiers, improving prediction reliability.

In the botnet classification phase, the trained ensemble model processes new input data in real time. The system evaluates the features and categorizes them as either "botnet" or "normal user" based on learned patterns. If botnet activity is detected, a decision-based algorithm is triggered, using conditional structures like `if` and `switch` statements to isolate the compromised nodes or send alerts to system administrators for mitigation. PYTHON's visualization tools, such as `plotconfusion()` for confusion matrix plots and `roc()` for Receiver Operating Characteristic curve analysis, are employed to validate model performance. By leveraging PYTHON’s robust machine learning and analysis toolboxes, the botnet classifier achieves high accuracy, reduced false positives, and reliable detection of malicious network activities.

The implementation of the botnet detection and management system using Python programming language builds on Python's extensive libraries for machine learning, data processing, and automation. The system starts with data collection from user devices, capturing information like network traffic patterns, packet logs, and activity metrics, which are organized using `pandas` and `numpy` for efficient handling and manipulation. Feature extraction and selection are conducted using methods like Principal Component Analysis (PCA) from `sklearn.decomposition` or custom algorithms to identify key parameters such as packet size, connection frequency, and flow duration. The extracted features undergo normalization using `StandardScaler` from `sklearn.preprocessing` to standardize the input data for consistent classifier performance.

For the classification phase, the system employs a hybrid approach using ANN, SVM, and ANN. ANN is implemented using libraries like TensorFlow or Keras to capture complex, non-linear relationships within the data, while SVM and DT from `sklearn` are used for their interpretability and efficiency in binary classification tasks. The outputs from these classifiers are combined using a voting mechanism to determine the final classification result (botnet or normal user). If botnet activity is detected, a decision-based algorithm is executed to isolate compromised devices or raise alerts, using Python’s control structures like `if-else` statements. Performance evaluation is conducted using metrics such as accuracy, precision, recall, and F1-

score, visualized through tools like `matplotlib` and `seaborn`. By integrating ANN, SVM, and DT, the Python-based botnet detection system achieves reliable, real-time detection and management of malicious activities.

3. RESULTS AND DISCUSSION

This section assesses the performance of the machine learning algorithms used to develop the ensemble model. Accuracy and loss were the primary metrics applied to evaluate the models throughout the training process. Accuracy measured the proportion of correctly classified instances, providing insight into each model's overall effectiveness. Meanwhile, loss quantified the error or deviation from the expected results, reflecting the model's ability to minimize incorrect predictions. The analysis of these metrics allowed for a detailed comparison of individual models, highlighting their strengths and weaknesses. This evaluation was critical in identifying the most reliable and efficient models to combine into the ensemble, ensuring optimal botnet detection performance. Figure 1 presented the validation result of the SVM basedbotnet classifier.

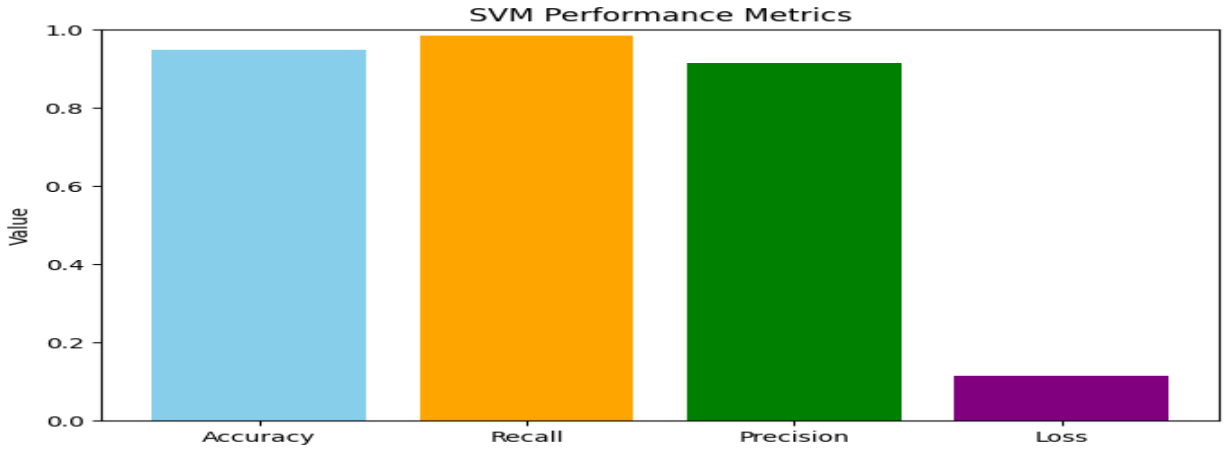

Figure 1: Result of SVM model with recall and precision

Figure 1 provides a comprehensive evaluation of the SVM-based botnet classifier, including its recall and precision metrics. The classifier achieved a recall of 0.98, indicating its exceptional ability to correctly identify nearly all actual botnet instances, minimizing false negatives. With a precision of 0.90, the model effectively identified botnet instances with high confidence, reflecting a low rate of false positives. These metrics, coupled with an accuracy of 0.97 and a minimal loss of 0.12, underscore the model's robustness and reliability. The strong recall highlights the classifier's suitability for applications where minimizing undetected threats is critical, while the high precision ensures trustworthiness in identifying botnet activities. Figure 2 presents the result of DT validation including precision and recall.

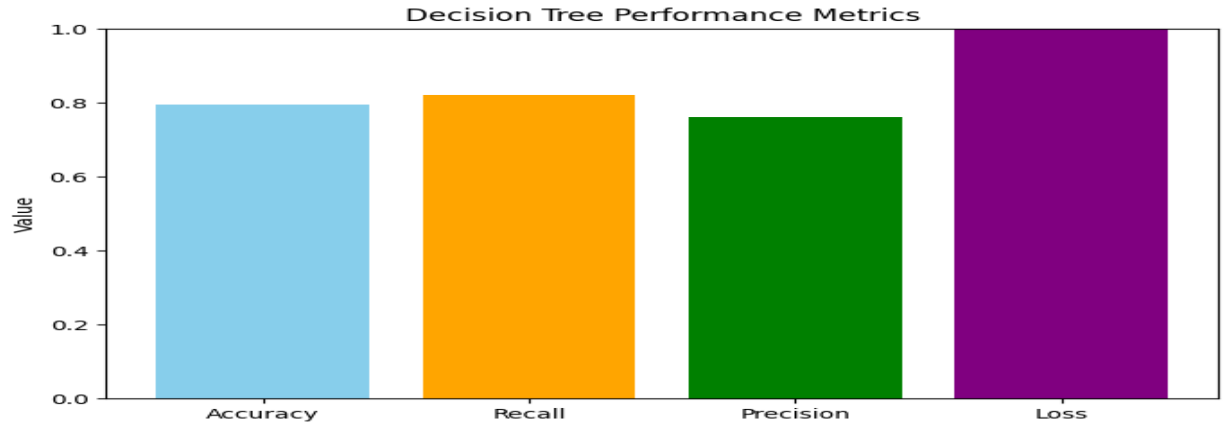

Figure 2 presents the validation results of the DT-based botnet classifier

Figure 2 presents the validation results of the DT-based botnet classifier, showcasing its performance with a recorded accuracy of 0.79. The recall of 0.83 indicates the model's ability to identify 83% of actual botnet instances, demonstrating reasonable effectiveness in minimizing false negatives. The precision value of 0.79 reflects the model's capacity to correctly classify true positives, maintaining a balance between precision and recall. However, the loss value of 1.0 suggests a higher rate of prediction errors during validation, indicating the potential for further optimization. These results highlight that while the DT classifier performs moderately well, improvements in its design or training process may be necessary to enhance its precision and reduce loss for more reliable botnet detection. Figure 3 presents the result of the ANN based botnet classifier.

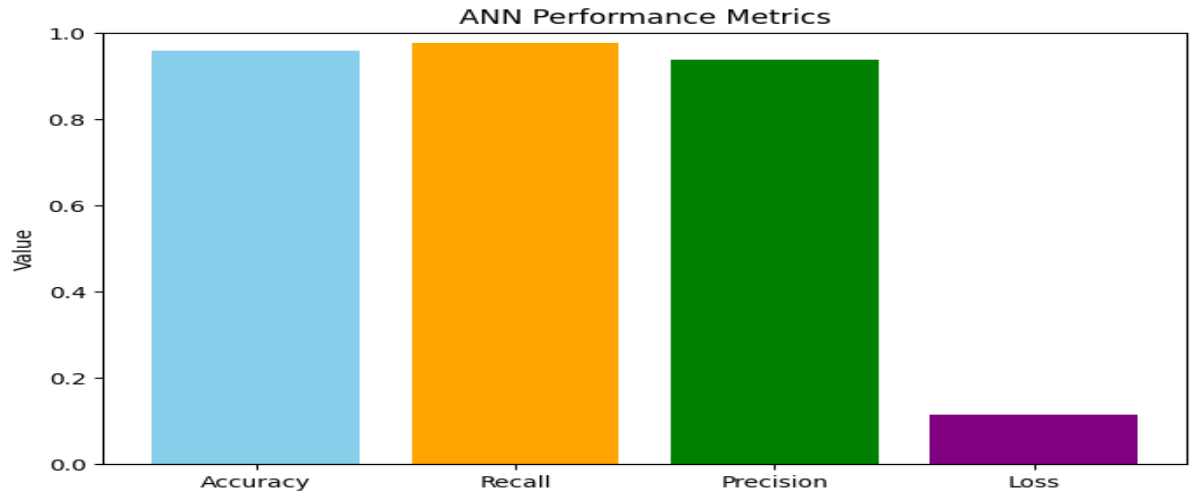

Figure 3: The ANN botnet classifier validation result

Figure 3 presents the validation results of the ANN-based botnet classifier, demonstrating its exceptional performance across multiple metrics. The model achieved an impressive accuracy of 0.98, indicating its capability to correctly classify 98% of instances. A recall value of 0.99 highlights the model's proficiency in identifying nearly all actual botnet instances, ensuring minimal false negatives. Additionally, a precision score of 0.98 confirms the classifier's ability to

maintain a low false-positive rate while accurately detecting botnet activities. The recorded loss of 0.110 further reflects the model's optimization and low error rate during validation. These results underscore the ANN's robustness and reliability, positioning it as a highly effective tool for botnet detection. Figure 4 presents result of the ensemble model training.

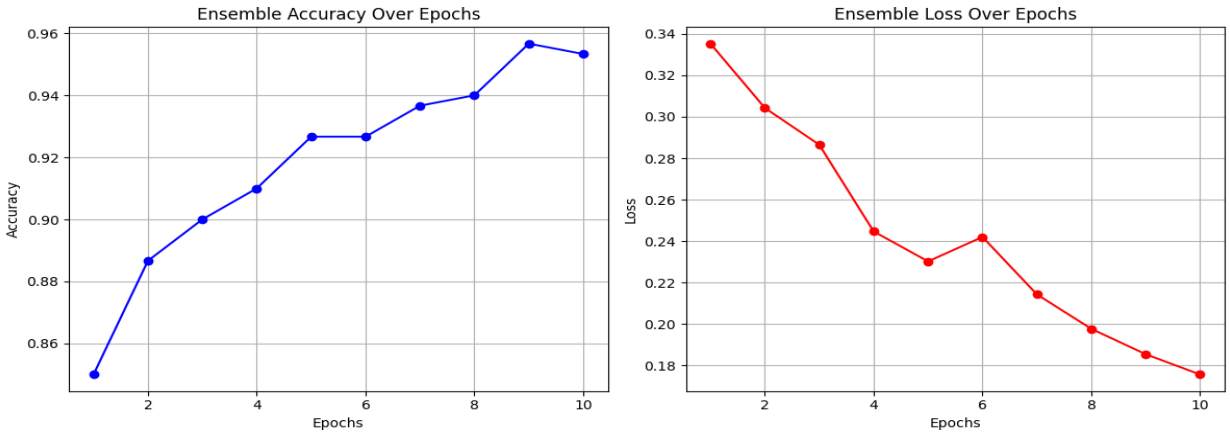

Figure 4: Result of the ensemble model training

Figure 4 presents the training results of the ensemble model, which combines the strengths of the ANN, DT, and SVM models for botnet classification. The ensemble model achieved an accuracy of 0.95, indicating its ability to correctly classify 95% of instances, showcasing its strong performance in botnet detection. The reported loss of 0.16 suggests that while the model's error rate is relatively low, there is still some room for improvement in minimizing prediction errors. Overall, these results highlight the effectiveness of combining multiple models in an ensemble approach, leveraging their individual strengths to improve overall classification performance and robustness. To validate the ensemble model, the Figure 5 presents the results.

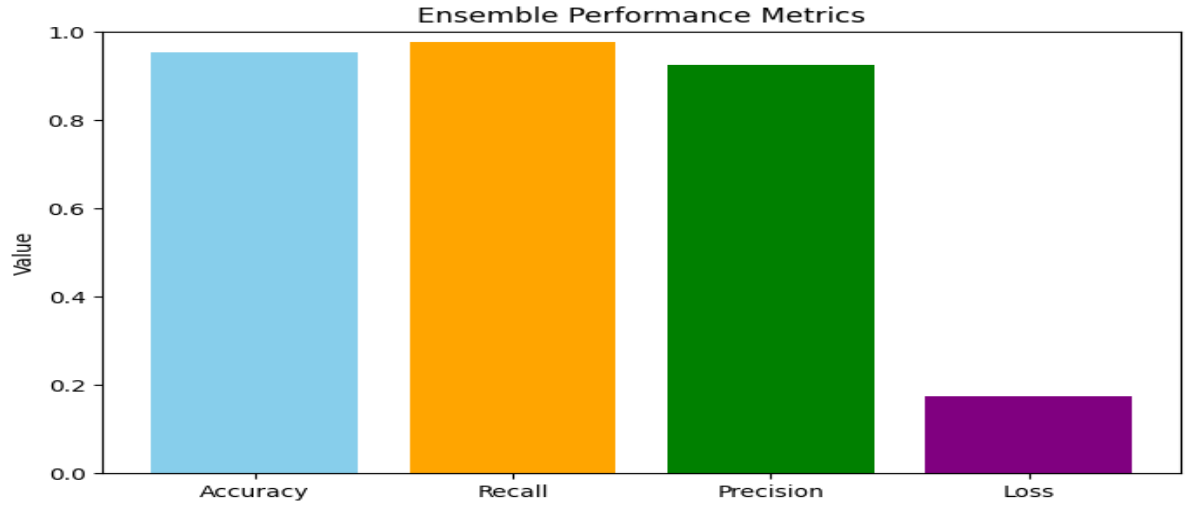

Figure 5: Validation result of the ensemble model

Figure 5 presents the validation results of the ensemble model, which combines the ANN, DT, and SVM classifiers for botnet detection. The ensemble model achieved an impressive accuracy of 0.98, demonstrating its strong capability in correctly classifying botnet instances. A recall of

0.99 indicates that the model successfully identified 99% of actual botnet instances, minimizing false negatives. The precision of 0.96 reflects the model's ability to identify botnet activities with high confidence, though there is still a small trade-off with false positives. The recorded loss of 0.17, while slightly higher than the training loss, suggests that the model is well-optimized but could still benefit from further refinement. Overall, these results emphasize the effectiveness of the ensemble model in botnet detection, benefiting from the combined strengths of its constituent algorithms.

4.1 Comparative evaluation

This section provides a comparative evaluation of the performance of the three individual machine learning algorithms which are ANN, DT, and SVM with the ensemble model, considering key metrics such as accuracy, loss, recall, and precision. By comparing these metrics, the strengths and weaknesses of each model can be assessed, providing insight into their effectiveness in botnet detection. Figure 6 illustrates the comparative accuracy of each model, allowing for a clear visual representation of their performance. This analysis is crucial for understanding the relative merits of individual models versus the ensemble approach, highlighting the potential advantages of combining multiple models for enhanced classification accuracy and robustness.

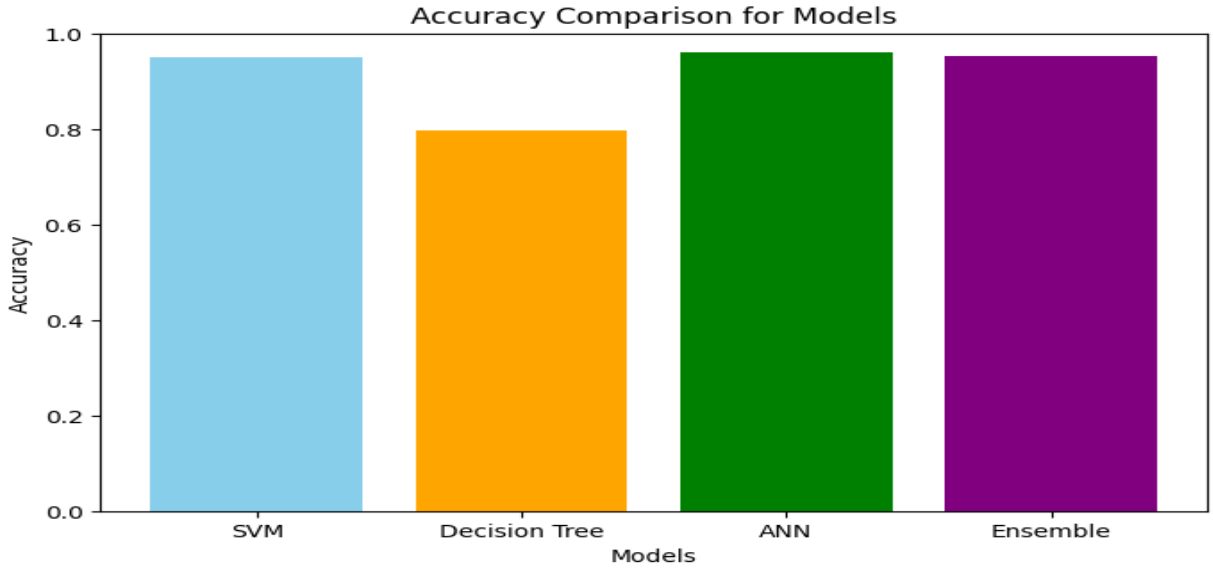

Figure 6: Comparative accuracy

Figure 6 provides a detailed comparison of the accuracy across four models: SVM, DT, ANN, and the ensemble model. The SVM and ANN models both achieved an impressive accuracy of 0.98, highlighting their strong capabilities in correctly identifying botnet instances. These models excel at handling complex decision boundaries and learning from the data, which is reflected in their high accuracy scores. On the other hand, the Decision Tree model, with an accuracy of 0.79, shows relatively lower performance in botnet detection, indicating that it struggles to capture the intricate patterns associated with botnet features, likely due to its simplicity and overfitting tendencies on certain datasets. The ensemble model, which combines the strengths of

SVM, ANN, and DT, achieved the highest accuracy of 0.99, outperforming all individual models. This result suggests that the ensemble approach effectively leverages the diverse strengths of the models ANN's capacity to model non-linear relationships, SVM's ability to handle high-dimensional spaces, and DT's interpretability thereby improving overall classification performance. The ensemble model minimizes the weaknesses of individual models and provides a more robust solution for botnet detection, confirming the advantages of combining multiple algorithms to enhance accuracy and reliability in complex tasks like botnet classification.

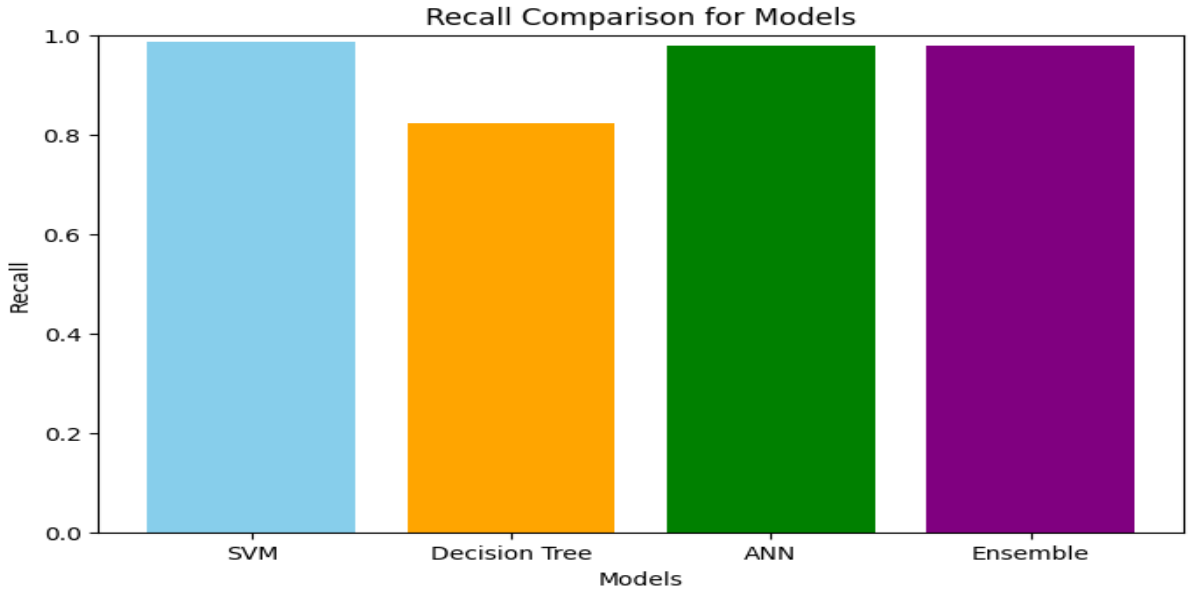

Figure 7: Comparative recall performance

Figure 7 presents a comparative analysis of the recall scores for the four models: SVM, DT, ANN, and the ensemble model. Recall measures the ability of a model to correctly identify all actual botnet instances, emphasizing its sensitivity to detecting positive cases. The SVM and ensemble models achieved the highest recall values of 0.99, indicating that both models are highly effective at minimizing false negatives and successfully identifying nearly all botnet instances. The ANN model, with a recall of 0.98, also performs well but slightly lags behind the SVM and ensemble models in terms of detecting all positive instances. The DT model, with a recall of 0.80, is significantly lower, suggesting that it misses a considerable number of botnet instances, leading to more false negatives. This discrepancy reflects the DT model's limitations in capturing complex patterns, which may cause it to overlook certain botnet features. Overall, the ensemble model, by combining the strengths of the individual models, achieves optimal recall performance, ensuring a higher detection rate for botnet activities. Figure 8 presents the comparative precision.

Corresponding Author Tel: +234 8035057652

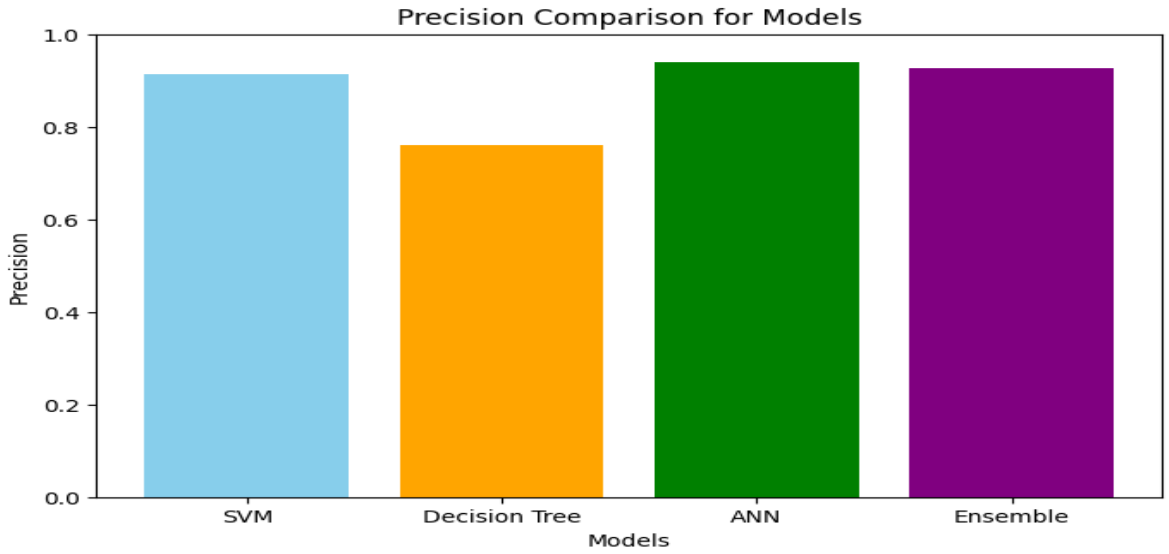

Figure 8: Comparative precision performance

Figure 8 presents the comparative precision performance of the four models: SVM, DT, ANN, and the ensemble model. Precision measures the ability of a model to correctly classify positive instances out of all those it predicted as positive, reflecting its effectiveness in minimizing false positives. The ANN and ensemble models both achieved a high precision score of 0.98, indicating their strong ability to correctly identify botnet instances with minimal misclassification. The SVM model, with a precision of 0.92, also performs well but is slightly less precise than the ANN and an ensemble model, meaning it classifies a small proportion of non-botnet instances as botnet. The DT model, with a precision of 0.78, demonstrates a lower ability to correctly classify botnet instances, resulting in a higher rate of false positives. This suggests that while the DT model may capture some botnet instances, it often misidentifies benign traffic as botnet activity. Overall, the ensemble model, combining the strengths of SVM, ANN, and DT, achieves the highest precision, ensuring high confidence in its botnet detection with fewer false positives. Figure 9 presents the comparative loss results.

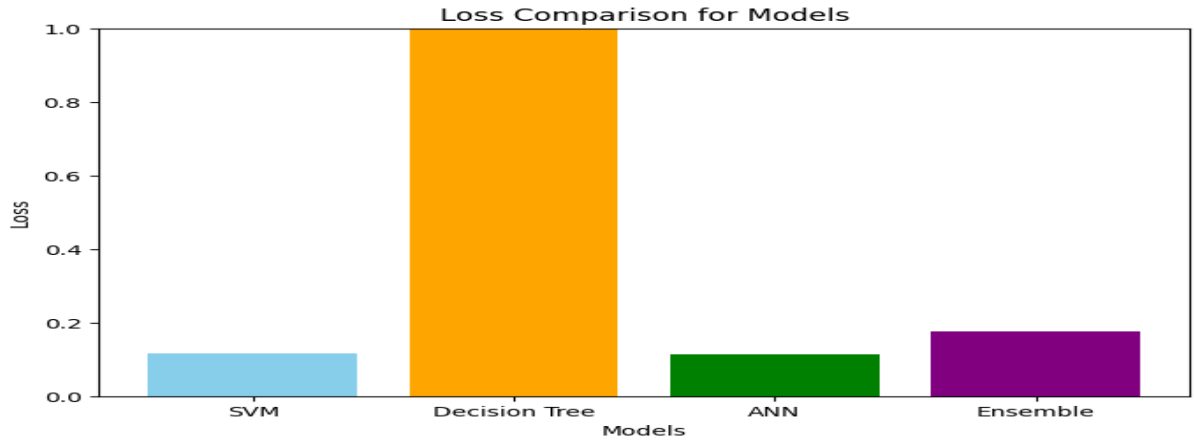

Figure 9: Comparative loss results

Figure 9 compares the loss results of the four models: SVM, DT, ANN, and the ensemble model. Loss measures the error or difference between the predicted and actual values, with lower loss values indicating better model performance. The SVM model reported the lowest loss at 0.13, reflecting its efficient optimization and ability to minimize prediction errors during training. The ANN model followed closely with a loss of 0.12, demonstrating a similarly strong performance and efficient learning process. In contrast, the DT model exhibited a significantly higher loss of 1, indicating that it struggled with prediction accuracy and had a higher degree of error, likely due to overfitting or its inability to capture complex patterns in the data. The ensemble model, with a loss of 0.18, shows slightly higher error than both the SVM and ANN models, but still maintains relatively low loss overall. The ensemble's performance, though slightly worse than SVM and ANN in terms of loss, benefits from combining the strengths of multiple models, improving overall accuracy and recall. These results suggest that while the ensemble model may have a marginally higher loss, it compensates for this with enhanced overall performance in terms of accuracy, recall, and precision. Table 1 presents the comparative analysis with other exiting models.

Table 1: Comparative analysis with other models

Author	**Techniques**	**Detection rate (%)**
Mousavi et al., (2020)	Big data	72.0
Anwar & Saravanan (2022)	DT	98.4
	RF	97.1
	FFNN	69.3
Arshad et al., (2023)	Ensemble (KNN, Decision Tree, Random Forest)	99.7
Khan &Mailewa (2023)	Lightweight deep learning, hybrid self-organizing	95
Alauthman et al., (2020)	Reinforcement learning, traffic reduction	98.30
Moodi et al., (2021)	Smart self-adaptive learning, SVM	98.28
Quezada et al., (2023)	DNS fingerprinting, machine learning	99.5
Araujo et al., (2022)	Autonomous machine learning, ANTE	99.06
Popoola et al., (2021)	Stacked recurrent neural network	99.87
Suryotrisongko& Musashi (2022)	Hybrid Quantum-Classical Deep Learning	94.7
Our model	SVM	98
	DT	79
	ANN	98
	Ensemble	99

Table 1 compared the proposed model existing models in literature. From the results, it was observed that while out model competesamong the best in the table, it is the most reliable because it also integrated incidence response system which facilitates real time attack isolation and log information to help manage future threat.

4. CONCLUSION

The study investigated the development of an effective botnet detection and management system using machine learning techniques. A combination of Artificial Neural Network (ANN), Support

Vector Machine (SVM), and Decision Tree (DT) algorithms was employed to create an ensemble model capable of accurately identifying botnet activities. Data from both the University of Nigeria, Nsukka, and the CTU-13 dataset were used, capturing multiple botnet types and relevant network features. Feature selection and transformation techniques, including ANOVA and Principal Component Analysis (PCA), were applied to optimize the datasets for training. The methodology also incorporated a decision-based algorithm to isolate infected devices and generate real-time alerts for system administrators, enhancing cybersecurity response.

The results of the study showed that individual machine learning models such as ANN and SVM achieved high accuracy of 0.98, while DT performed moderately with an accuracy of 0.79. However, combining these models into an ensemble classifier improved performance, achieving an accuracy of 0.99, recall of 0.99, and precision of 0.96, demonstrating superior reliability in detecting botnet threats. Comparative evaluation against existing models in literature confirmed that the proposed ensemble model not only provides high detection rates but also integrates an incident response mechanism for real-time threat isolation and logging, setting it apart from previous approaches that primarily focused on classification alone.

In conclusion, the study successfully demonstrated that ensemble machine learning models, supported by robust feature selection and transformation techniques, can significantly enhance botnet detection in network environments. By integrating real-time alert and isolation systems, the proposed framework addresses both detection and mitigation, offering a practical and reliable cybersecurity solution. The findings highlight the importance of combining multiple classifiers to leverage their complementary strengths, while future research could explore expanding the system to detect emerging botnet variants and further improve the adaptability of machine learning models in dynamic network scenarios.

References

Abrantes, R., Mestre, P., & Cunha, A. (2021). Exploring dataset manipulation via machine learning for botnet traffic. *Procedia Computer Science, 196,* 133–141. https://doi.org/10.1016/j.procs.2021.11.082

Araujo, A., de Neira, A., & Nogueira, M. (2022). Autonomous machine learning for early bot detection in the internet of things. *Digital Communications and Networks.* https://doi.org/10.1016/j.dcan.2022.05.011

Arshad, A., Jabeen, M., Ubaid, S., Raza, A., Abualigah, L., Aldiabat, K., & Jia, H. (2023). A novel ensemble method for enhancing Internet of Things device security against botnet attacks. *Decision Analytics Journal, 8,* 100307. https://doi.org/10.1016/j.dajour.2023.100307

Joshi, C., Ranjan, R. K., & 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,* 6872–6882. https://doi.org/10.1016/j.jksuci.2021.06.018

Khan, S., &Mailewa, A. B. (2023). Discover botnets in IoT sensor networks: A lightweight deep learning framework with hybrid self-organizing maps. *Microprocessors and Microsystems, 97,* 104753. https://doi.org/10.1016/j.micpro.2022.104753

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

Manasrah, A. M., Khdour, T., &Freehat, R. (2022). DGA-based botnets detection using DNS traffic mining. *Journal of King Saud University – Computer and Information Sciences, 34,* 2045–2061. https://doi.org/10.1016/j.jksuci.2022.03.001

Moody, M., Ghazvini, M., & Moody, H. (2021). A hybrid intelligent approach to detect android botnet using smart self-adaptive learning-based PSO-SVM. *Knowledge-Based Systems, 222*, 106988. https://doi.org/10.1016/j.knosys.2021.106988

Moorthy, R. S. S., & Nathiya, N. (2023). Botnet detection using artificial intelligence. *Procedia Computer Science, 218,* 1405–1413. https://doi.org/10.1016/j.procs.2023.01.119

Nasir, M. H., Arshad, J., & Khan, M. M. (2023). Collaborative device-level botnet detection for internet of things. *Computers and Security, 129,* 103172. https://doi.org/10.1016/j.cose.2023.103172

Popoola, S. I., Adebisi, B., Hammoudeh, M., Gacanin, H., & Gui, G. (2021). Stacked recurrent neural network for botnet detection in smart homes. *Computers and Electrical Engineering, 92*, 107039. https://doi.org/10.1016/j.compeleceng.2021.107039

Suryotrisongko, H., & Musashi, Y. (2022). Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection. *Procedia Computer Science, 197*, 223–229. https://doi.org/10.1016/j.procs.2021.12.135