JOURNAL OF ELECTRONIC VOLTAGE AND APPLICATION
e-ISSN: 2716-6074

JEVA

Vol. 6 No. 2 (2025) 61-72
https://publisher.uthm.edu.my/ojs/index.php/jeva

Anomaly Detection Using Artificial Intelligence Methods: Support Vector Machine and Decision Tree Algorithms

Emmanuel A. Ubom[1]*, Andikan E. Okoro[1]

[1] *Department of Electrical and Electronic Engineering, Faculty of Engineering, Akwa Ibom State University, Ikot Akpaden, Mkpat Enin, 524106, Akwa Ibom Sate, NIGERIA*

*Corresponding Author: emmmanuelubom@aksu.edu.ng
DOI: https://doi.org/10.30880/jeva.2025.06.02.006

Article Info

Received: 14 April 2025
Accepted: 1 November 2025
Available online: 9 December 2025

Keywords

Anomaly detection, machine learning, decision tree, support vector machine

Abstract

For anyone using a network, safeguarding personal or sensitive information is a top priority. Anomaly detection plays a crucial role in spotting irregular patterns that could signal malicious behaviour and help to prevent data breaches, defend against cyberattacks, and ensure the integrity of network resources. Hence, this study explores how machine learning can be used to detect unusual or suspicious activity in network traffic, a key step in protecting systems from threats like Denial of Service (DoS) attacks, unauthorised access, and probing attempts. Using the improved NSL-KDD dataset, which builds on the original KDD Cup 1999 dataset, two popular algorithms: Decision Trees and Support Vector Machines (SVM) with a polynomial kernel were tested. The models were applied to classify network traffic as either normal or potentially harmful, helping assess their effectiveness in identifying anomalies and supporting stronger network security. The Decision Tree model demonstrated a remarkable accuracy of 99.7% with an F-score of 0.997, showcasing its robustness in detecting prevalent attack patterns with high precision. Conversely, the Polynomial Kernel SVM exhibited a slightly lower accuracy of 99.5% but demonstrated stronger generalisation across various classes, reflected by a macro average F-score of 0.72. This indicates its superior capability to handle diverse and complex anomalies. The findings highlighted that both models are highly effective for intrusion detection, with the Decision Tree excelling in overall accuracy and the SVM offering a more balanced performance across a range of attack types. This research contributes to the ongoing development of intrusion detection systems, providing valuable insights into the trade-offs between model complexity, computational efficiency, and detection accuracy in real-time network security contexts.

1. Introduction

A major concern of any network user is the security of his or her information, and anomaly detection is essentially the identification of unusual activity in the network that may indicate malicious intent, and help prevent loss of important information, cyberattacks and the protection of network resources. Anomalies within network traffic often signify potential security threats such as Denial of Service (DoS) attacks, unauthorised access attempts, probing, and other forms of intrusion. It is crucial to find these problems in oreder to protect the network's core functions, its trustworthiness, security, and of course stability in the delivery of its services. The NSL-KDD dataset is widely utilised for evaluating intrusion detection systems (IDS) [1] and recent research continues to leverage it for this purpose [2]- [4]. It was developed as an enhancement over the original KDD Cup 1999 dataset, which had

several limitations, including redundant records and an imbalanced class distribution, leading to poor generalisation of models trained on it. NSL-KDD addresses these issues by removing duplicates and providing a more balanced distribution of normal and attack data points, thus presenting a more realistic challenge for evaluating machine learning models for anomaly detection.

Two popular machine learning algorithms employed for anomaly detection in network security are Decision Trees and Support Vector Machines (SVM) with a polynomial kernel [5], [6]. These methods take distinct approaches to identifying anomalous behaviour in network traffic. Decision Trees are a supervised learning algorithm that recursively splits the data based on the most informative features, resulting in a tree-like structure where each branch represents a decision rule and each leaf node represents a classification outcome. The model functions by identifying the feature that best separates the data into distinct classes, continuing this process until a stopping criterion is reached, such as a minimum number of samples in each leaf node. Decision Trees are renowned for their interpretability and simplicity, making them a popular choice for network intrusion detection. They are effective in handling both numerical and categorical features, which is particularly advantageous when analysing the NSL-KDD dataset that contains various features such as protocol type, service type, and connection duration [7]. However, Decision Trees are susceptible to overfitting, particularly when the dataset is noisy or imbalanced, as is common in network traffic data.

Conversely, Support Vector Machines (SVM) with a polynomial kernel provide a more complex approach by seeking to find the optimal hyperplane that separates normal and anomalous traffic. SVMs are particularly adept at handling high-dimensional data, as they can map inputs into a higher-dimensional space using kernel functions [8], [9]. The polynomial kernel introduces non-linearity into the model, allowing SVM to capture intricate relationships between features that may represent complex attack patterns. This capability is particularly beneficial in network security, where cyberattacks often exhibit non-linear and multi-dimensional characteristics. SVMs with a polynomial kernel excel in scenarios where decision boundaries are not easily separable by a straight line, making them well-suited to detecting previously unseen or novel types of intrusions. However, this approach is computationally intensive, especially when dealing with large datasets like NSL-KDD, and it can encounter scalability challenges [10]- [12].

This study evaluates both Decision Trees and SVM with a polynomial kernel applied to the NSL-KDD dataset to evaluate their performance in detecting network anomalies. The objective of this research is to examine the trade-offs between these two algorithms in terms of computational efficiency, detection accuracy, and model complexity. By comparing their performance on the NSL-KDD dataset, this study seeks to determine which algorithm is better suited for real-time network intrusion detection, balancing the demands for accuracy and computational feasibility.

2. Review of Related Works

In the article referenced as [13], the authors used some machine learning algorithms to classify network traffic using the NSL-KDD dataset. After a thorough data preprocessing, Decision Trees, Naïve Bayes, k-Nearest Neighbours (KNN), Random Forests, Gradient Boosted Trees, and Support Vector Machines (SVM) were used to study and foretell potential network intrusions. Model performance was assessed using several evaluation tools, including the confusion matrix, accuracy (ACC), Receiver Operating Characteristic (ROC) curve, and Area Under the Curve (AUC). The findings showed that when careful preprocessing is done, these algorithms could achieve very high accuracy in sniffing out attacks, and this offers insights of great worth into the relative strengths of different classes of models.

Similarly, in [14], the authors called attention to the increasing essentiality for strong Intrusion Detection Systems (IDS) to fight the incessant threat of cyberattacks as a result of the widespread adoption of smart technologies that rely on internet access. They noted that attackers continually evolve their tactics to bypass conventional defences, posing significant challenges to the development of effective IDS solutions. In tackling this issue, the study made use of multiple machine learning classifiers like SVM, Naïve Bayes, Random Forest, Extra-Trees Classifier, Logistic Regression, and Decision Tree on the NSL-KDD dataset to differentiate between a normal and an intrusive traffic. The authors were able to improve model performance by taking out unnecessary data that needs to be removed before analysis. Their results showed that Extra-Trees Classifier, Random Forest, and Decision Tree were more accurate in classification and performed above 99% on all the attack categories, showing both how effective and computationally efficient these models are in real-world IDS applications.

In [15], the researchers tackled the growing challenge of keeping network communications secure, most especially with the rise of Internet of Things (IoT) devices, by focusing on methods for detecting unusual or suspicious activity. While tools like Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are commonly used, the study pointed out that not enough attention has been given to exploring different types of SVMs and how their settings affect performance. To address this gap, the authors tested various kernel functions to find the best configuration for SVM. They used the NSL-KDD dataset, which is known for having uneven class distribution, to prove how reliable their improved model is. Their experiments showed that a well-tuned SVM

J. of Electronic Voltage and Application Vol. 6 No. 2 (2025) p. 61-72

reached an impressive 99.9% accuracy, beating other methods on key performance measures and proving it's a strong choice for spotting anomalies in network security.

In [16], the researchers used Decision Tree, Random Forest, and SVM algorithms to spot unusual activity in IoT networks. They worked with two datasets, namely NSL-KDD and UNSW-NB15, and focused on improving performance through feature selection and cross-validation. Their findings showed that Random Forest and SVM delivered strong results, with high accuracy and low false-positive rates, making them more practical options for detecting anomalies in real-world IoT systems. In [17], the study compared how well different AI model including SVM, Decision Tree, and ensemble methods performed in detecting anomalies within edge computing environments, using the NSL-KDD dataset. The authors found that SVM and Decision Tree models stood out, especially in resource-limited edge computing setups, offering a good balance between accuracy and efficiency.

3. Methodology

Machine learning algorithms played a key role in analysing and making sense of the data processed in the Google Colaboratory environment. Using Python and the scikit-learn library, the study applied these algorithms to make predictions and classify data based on patterns found in historical records.

Among these, the Decision Tree algorithm stood out as a simple yet powerful tool widely used for both classification and regression tasks [18], [19]. It works by repeatedly splitting the dataset into smaller groups based on the most important feature at each step. This creates a tree-like structure, where each branch represents a decision rule and each leaf shows an outcome [20].

The process starts at the root, where the algorithm picks the most informative feature, using criteria like Gini impurity or information gain, and splits the data based on its values. This continues at each level of the tree, building a set of rules designed to make each group as consistent as possible. The goal was to reduce errors and improve the accuracy of predictions.

One major advantage of Decision Trees is that they're easy to understand. Their rule-based structure mirrors how people often make decisions, which makes them highly interpretable. They're also flexible, able to work with both numerical and categorical data. However, Decision Trees can sometimes overfit the data, especially when the dataset is noisy or complex. They're also sensitive to changes in the training data, which can affect how well they perform on new, unseen data.

Fig. 1 shows the decision boundaries created by a Decision Tree when applied to a synthetic dataset..

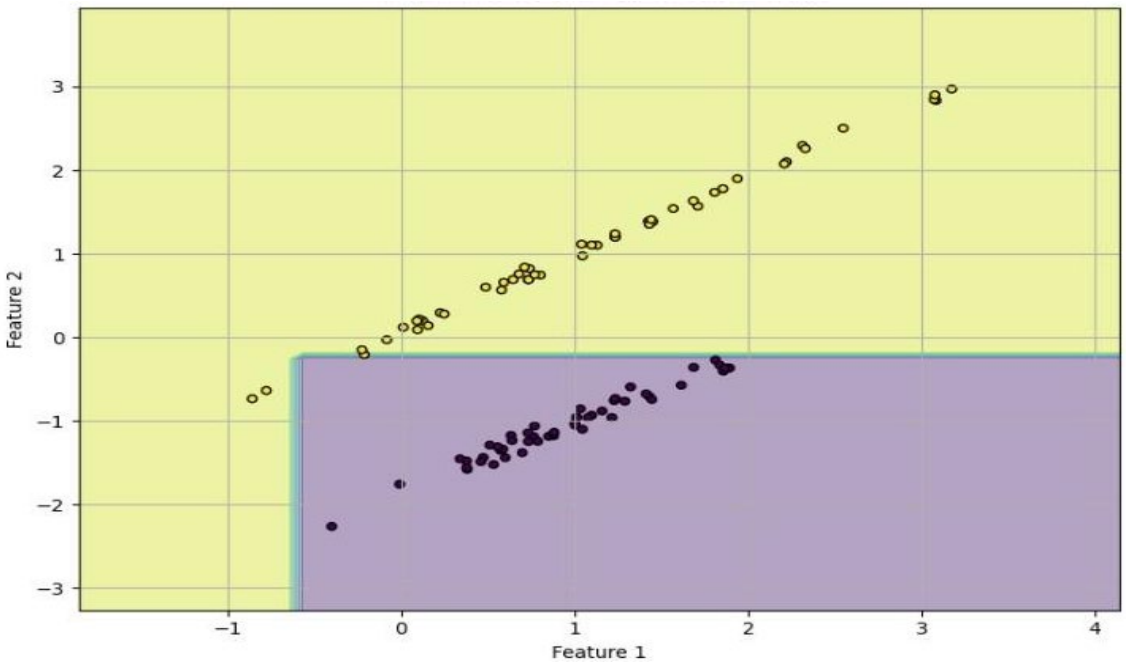

Fig. 1 *Decision boundaries of decision tree*

Support Vector Machine (SVM) is a supervised learning algorithm primarily used for classification tasks, although it can also handle regression problems. At its core, SVM seeks to find the optimal hyperplane that separates data points belonging to different classes. The goal is to maximise the margin, the distance between the closest points of opposing classes—which helps improve the classifier's generalization and robustness [15], [21].

When data isn't linearly separable, SVM employs kernel functions to transform the input space into a higher-dimensional feature space where a linear separation becomes possible. One widely used kernel is the polynomial kernel, which introduces non-linearity by considering polynomial combinations of the input features. The degree of the polynomial is a tunable parameter that controls the complexity of the decision boundary, allowing the model to capture more nuanced patterns in the data [15].

SVMs with polynomial kernels are especially effective in scenarios involving non-linear relationships and complex, multi-dimensional datasets. One of the standout strengths of SVM is its ability to maintain high classification accuracy even in high-dimensional spaces, where the number of features may exceed the number of samples. However, this performance comes with trade-offs: SVMs can be computationally demanding and memory-intensive, particularly when applied to large-scale datasets.

Fig. 2 depicts the decision boundary formed by an SVM using a polynomial kernel on a synthetic dataset.

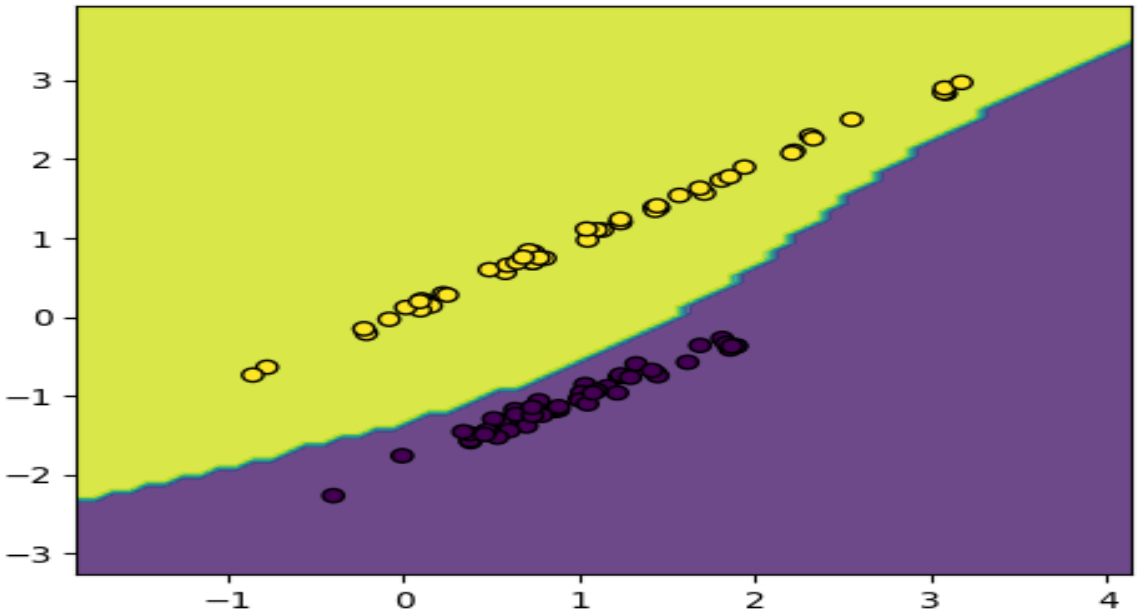

Fig. 2 *SVM polynomial kernel*

3.1 Implementation Tools

The study employed supervised learning algorithms to classify network traffic data, utilising historical observations to train models capable of generalising to previously unseen instances. Two algorithms namely; Decision Tree and Support Vector Machine (SVM) were selected owing to their fundamentally distinct learning paradigms and performance characteristics, thereby facilitating a meaningful comparative analysis.

The implementation was conducted in the Python programming environment using Google Colaboratory. Key libraries included pandas for data manipulation, scikit-learn for model development, and matplotlib and seaborn for data visualisation.

The Decision Tree classifier was instantiated via sklearn.tree.DecisionTreeClassifier. This model operates by recursively partitioning the dataset based on the most informative features at each node, as determined by an impurity metric such as Gini impurity or entropy. The result is a hierarchical structure of decision rules, forming a tree-like model. For this study, the classifier was configured with Gini impurity and a fixed random state to ensure reproducibility. All other parameters were retained at their default values.

The Support Vector Machine classifier was implemented using sklearn.svm.SVC with a polynomial kernel, enabling it to handle non-linearly separable data by projecting it into a higher-dimensional space. The algorithm seeks to identify the optimal hyperplane that maximises the margin between distinct classes, relying solely on the most critical data points—termed support vectors—to define the decision boundary. Given the sensitivity of kernel-based methods to input feature scales, feature standardisation was performed using StandardScaler. Moreover, SVM performance is notably influenced by hyperparameter optimisation, particularly with respect to the regularisation parameter (C), polynomial degree (degree), and kernel coefficient (gamma), all of which govern model complexity and decision boundary flexibility.

Conceptually, SVM focuses on maximising the separation margin between classes by constructing an optimal hyperplane, guided primarily by support vectors, those instances nearest to the boundary. In contrast, the Decision Tree algorithm divided the dataset into branches based on feature-based conditions, continuing the process iteratively until subsets are homogeneous or a predefined stopping criterion is satisfied.

J. of Electronic Voltage and Application Vol. 6 No. 2 (2025) p. 61-72

3.2 Implementation Procedure

The anomaly detection procedure comprised several key stages, each designed to prepare the data, apply appropriate machine learning algorithms, and evaluate the resulting model performance. A concise overview of the adopted methodology is presented below:

Data Collection and Preprocessing: The process commenced with the acquisition of a suitable dataset, such as the NSL-KDD, which contains network traffic data annotated with instances of normal and anomalous behaviour. Preprocessing was undertaken to ensure the quality of data and its suitability for analysis. This included handling missing values, eliminating duplicate records, and performing feature scaling through normalisation or standardisation. Such measures enhance model stability and contribute significantly to improved predictive performance.

Feature Selection and Engineering: Subsequent steps involved the identification and extraction of features deemed relevant for effective anomaly detection. Techniques such as feature scaling, dimensionality reduction, and domain-informed feature engineering were applied. Proper feature selection not only enhances the model's sensitivity to anomalous patterns but also reduces computational overhead by eliminating irrelevant or redundant attributes.

Data Splitting: The dataset was partitioned into training and testing subsets. The training data was utilised to build the machine learning models, whereas the test data served as an independent benchmark for evaluating model generalisation. This division is essential to prevent overfitting and to gauge model performance on previously unseen data. The ratio was 70:30.

Model Training: Machine learning models were then trained using the prepared training data. The Random Forest algorithm constructs an ensemble of decision trees and aggregates their predictions through majority voting, offering robustness and resistance to overfitting. The Support Vector Machine (SVM) algorithm, particularly with a polynomial kernel, transforms the input data into a higher-dimensional space to identify non-linear decision boundaries. It then seeks an optimal hyperplane that maximises the margin between classes.

Model Evaluation: Trained models were evaluated using performance metrics such as accuracy, precision, recall, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). These metrics provide a comprehensive assessment of the models' ability to correctly identify anomalies while minimising false positives and false negatives.

Visualisation: Visualisation techniques were employed to support result interpretation and model diagnostics. Tools such as heatmaps, confusion matrices, and ROC curves offered intuitive insights into classification performance and the spatial distribution of anomalies within the dataset.

3.3 Evaluation Metrics

Performance metrics in machine learning are essential for evaluating the effectiveness and reliability of models. They provide insights into how well a model is performing and help guide improvements. Key performance metrics include:

Accuracy: The ratio of correctly predicted instances to the total number of instances. It provides a general measure of how often the model is correct, but may be misleading if the dataset is imbalanced [22]. The mathematical formula is given as equation (1) [23]

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

Precision: The ratio of true positive predictions to the sum of true positives and false positives. It measures the model's ability to correctly identify positive instances, minimizing false positives. It was calculated using equation (2) [23].

$$\frac{TP}{TP + FP} \tag{2}$$

Recall (Sensitivity): The ratio of true positive predictions to the sum of true positives and false negatives. It evaluates the model's ability to identify all relevant positive instances, minimizing false negatives. It is mathematically represented by equation (3) [23].

$$\frac{TP}{TP + FN} \tag{3}$$

F1-Score: The harmonic mean of precision and recall, providing a single metric that balances both aspects. It is useful when you need a balance between precision and recall and when there is an uneven class distribution [23] The mathematical formula used in the analysis is given in equation (4) [23].

$$F1\ Score = 2\left[\frac{(Precison \times Recall)}{(Precision + Recall)}\right] \quad (4)$$

Heatmap: In analysing the NSL-KDD dataset for intrusion detection, heatmaps serve as an effective tool for visualising and interpreting the data. They provide a colour-coded representation of feature distributions, correlations, and model performance, making it easier to identify patterns and relationships within the dataset. It is easier to observe correlations between different features, understand the distribution of attack types across feature values, and assess the performance of machine learning algorithms through metrics such as confusion matrices.

These metrics help in assessing how well the models performed on tasks such as classification, detection, and prediction, guiding the development and fine-tuning of machine learning algorithms.

4. Results and Discussion

4.1 Class Mapping

Table 1 shows the attack class mapping.

Table 1 *Attack class mapping*

Class Label	Classification	Description of Class
0	Back	This is a form of attack that attempts to overwhelm a system or a resource, thereby causing a denial of service.
1	Buffer Overflow	When a program's buffer overflow weakness is exploited to execute malicious code.
2	FTP Write	This attack class describes unauthorised users' attempts to write or transfer files via the FTP service.
3	Guess Password	When an unauthorised user attempts to gain access by guessing passwords over and over again.
4	IMAP	Represents Internet Message Access Protocol, an attack involving illegal access or an attempt to manipulation of email accounts using IMAP
5	IPSweep	This class involves IP range scanning to identify live hosts on a network.
6	Load	An attack where malicious packet is concocted to cause a system to respond to itself.
7	Loadmodule	This is the loading of a malicious module or code into a system to exploit risks.
8	Multihop	When a network attack involves secondary or alternate hosts to hide the true origin of the attack.
9	Neptune	Represents a denial-of-service attack that spates the target with traffic.
10	Nmap	When an attack uses Nmap tool to discover and map network hosts.
11	Normal	Represents normal network traffic, with no malicious intent.
12	Perl	Involves exploiting loopholes in Perl scripts or applications.
13	PHF	Means "Personal Home Page," when there is an attempts to utilise the CGI program meant for accessing personal webpages.
14	Pod	When there is an attempts to overload the network resources.
15	Portsweep	The acts of scanning host in search of open ports for possible attack use.
16	Rootkit	This attack represents illegal software installation to facilitate illegal access with concealed identity and presence.
17	Satan	When the tool Satan, which performs security assessments is used to attack.
18	Smurf	A denial-of-service that overload network with spoofed ICMP echo requests.

J. of Electronic Voltage and Application Vol. 6 No. 2 (2025) p. 61-72

Class Label	Classification	Description of Class
19	Spy	Spying a network in search of important information.
20	Teardrop	A denial-of-service attack that fragments packets developed to crash target systems.
21	Warezclient	Refers to a client involved in downloading or distributing pirated software.
22	Warezmaster	Stands for a central coordinating server for the pirated software distribution.

Fig. 3 shows the performance of the NSL-KDD dataset using the Decision Tree (DT) algorithm, and Fig. 4 shows the performance of the Support Vector Machine (SVM) model. Both models delivered strong results, showing high accuracy and impressive F-scores. However, there were clear differences in how each algorithm performed across specific metrics.

4.2 Result of Decision Tree Algorithm

The Decision Tree model performed impressively overall. It got about 99.7% of predictions right, which shows very high accuracy. Its F-score was also strong at 0.997, meaning it handled precision and recall well, especially for the more common classes in the dataset. But when looking at the macro average F-score, which treats all classes equally, the score dropped to 0.68. That suggests that the model struggles with less frequent or minority classes. On the flip side, the weighted average F-score was a perfect 1.00, showing that the model did an excellent job overall, mostly because it nailed predictions for the dominant classes.

4.3 Result of Support Vector Machine Polynomial Kernel Algorithm

The SVM model also delivered excellent results, with a 99.5% accuracy rate, just slightly below the Decision Tree, but still very good performance. Its F-score of 0.995 shows it handled precision and recall well, reliably identifying both positive and negative cases. Interestingly, the macro average F-score came in at 0.72, which is higher than the Decision Tree's. That suggests the SVM was more consistent across all classes, even the ones with fewer examples. Like the Decision Tree, its weighted average F-score was a perfect 1.00, showing outstanding overall performance, especially for the dominant classes in the dataset.

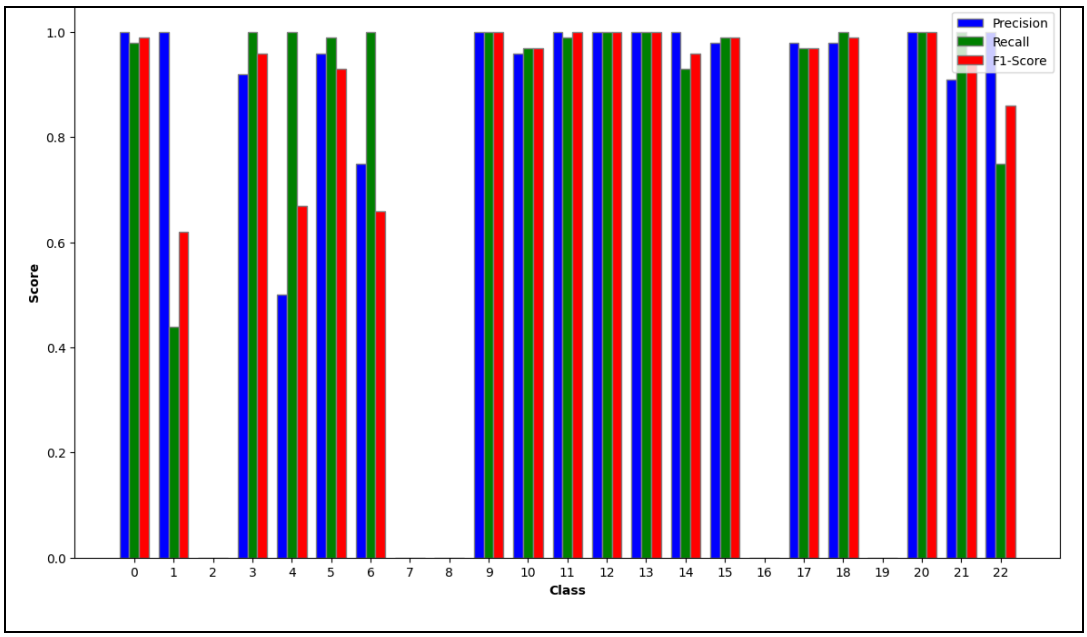

Fig. 3 *Performance of NSL KDD dataset using decision tree algorithm*

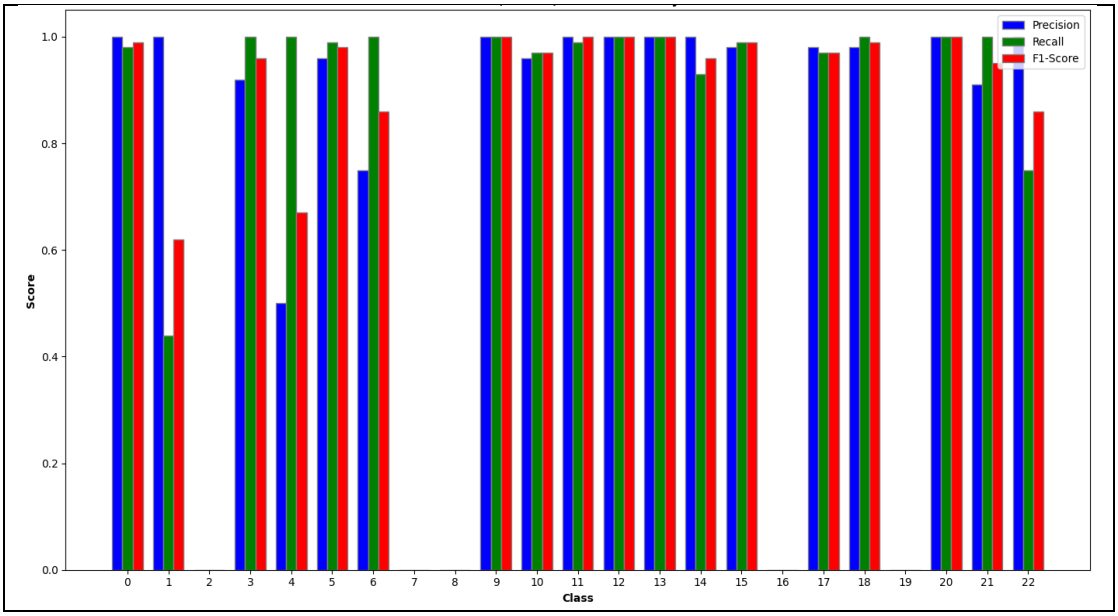

Fig. 4 *SVM performance using NSL KDD dataset*

Fig. 5 illustrates the confusion matrix for the Decision Tree (DT) model while Fig. 6 presents the confusion matrix for the Support Vector Machine (SVM) model, and the heatmap is shown in Fig. 7.

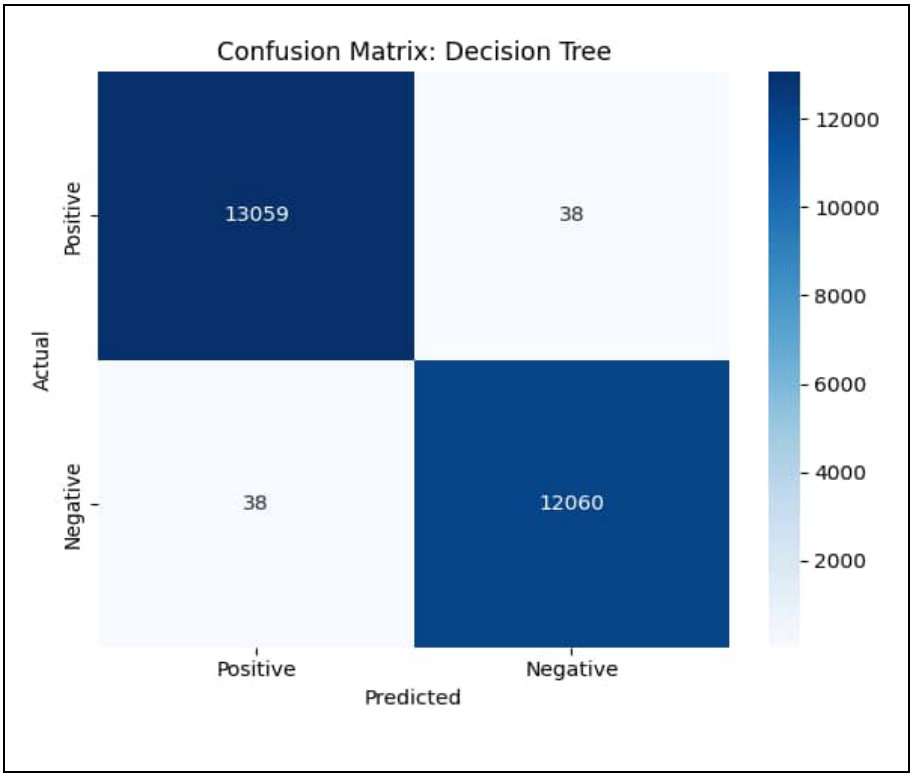

Fig. 5 *DT Confusion matrix*

The confusion matrix shows that the model is highly accurate, with very few misclassifications in overall rating. What's especially notable is the balance (38 false positives and 38 false negatives) which suggests the model isn't favoring one class over the other. This kind of symmetry points to a well-calibrated model with no obvious bias, making its predictions more trustworthy across both classes.

J. of Electronic Voltage and Application Vol. 6 No. 2 (2025) p. 61-72

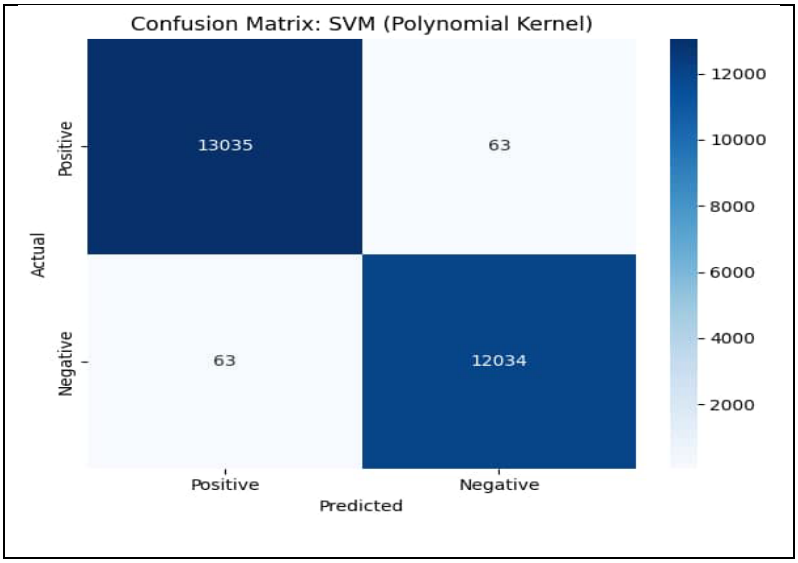

Fig. 6 *Confusion matrix using SVM (Polynomial kernel)*

The confusion matrix in Fig. 6 highlights the SVM model's strong performance, with very few misclassification errors overall. What stands out is the balance: 63 false positives and 63 false negatives. This symmetry indicates that the model treats both classes fairly, without leaning toward one or the other, a sign of well-balanced predictive capability.

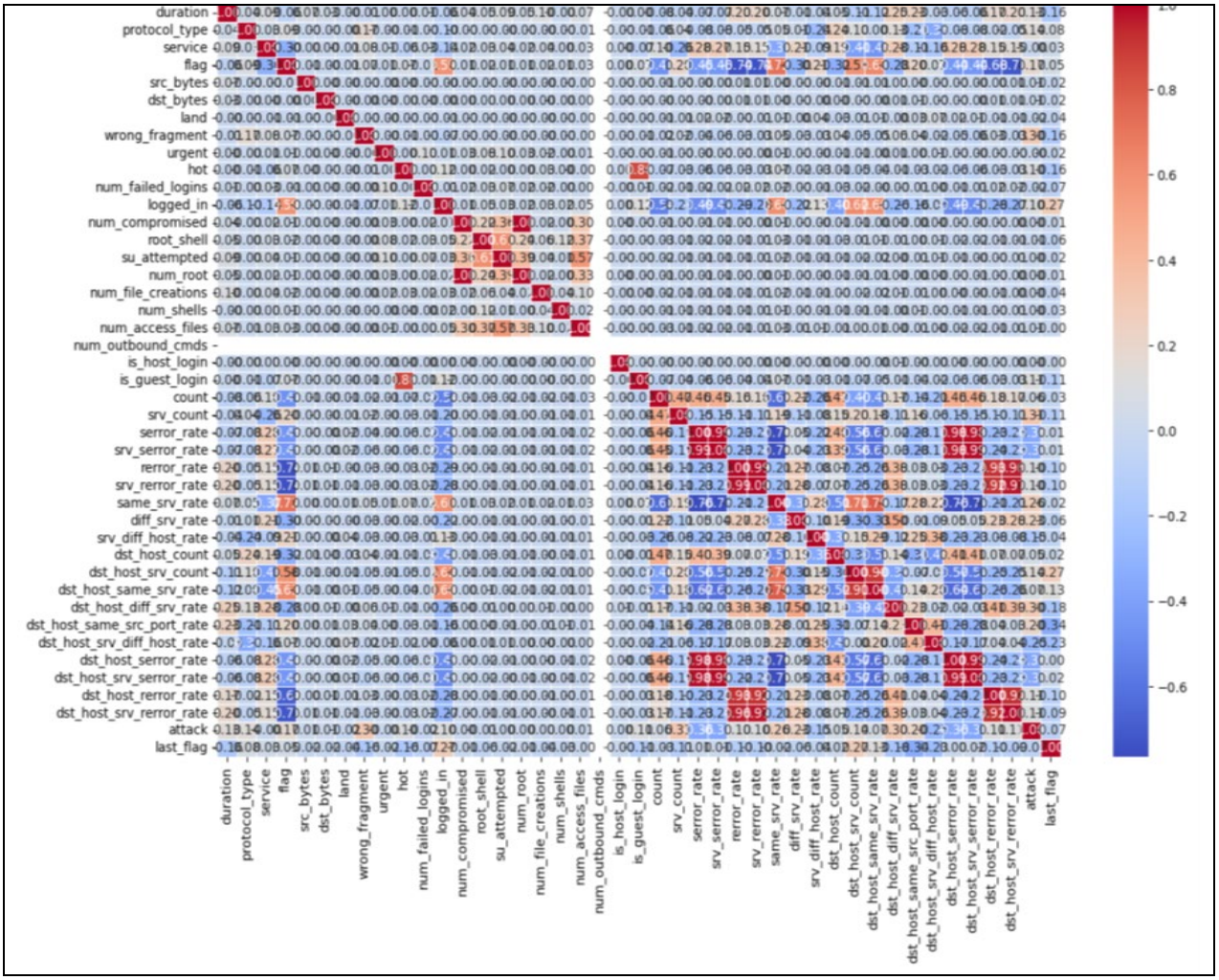

Fig. 7 *Heatmap*

4.4 Discussion

When comparing the Decision Tree and Polynomial Kernel SVM models, both proved to be very effective at classifying network data, though they each have their own strengths.

The Decision Tree model performed slightly better overall, with an accuracy of 99.7%. This means it was able to correctly identify nearly all the data points. Its F-score of 0.997 shows that it maintained a good balance between precision (how many of the identified anomalies were actually correct) and recall (how many actual anomalies it managed to catch). However, when looking at how it performed across all types of data, the macro average F-score was 0.68, which suggests it didn't do as well with less common types of data. Still, its weighted average F-score of 1.00 shows that it handled the most frequent types of data extremely well.

On the other hand, the Polynomial Kernel SVM model had a slightly lower accuracy of 99.5%, but that's still excellent. While it didn't outperform the Decision Tree in terms of raw numbers, it's known for being better at handling complex patterns, especially when the data isn't straightforward or when there are many features involved. Its F-score of 0.995 reflects a similarly robust precision–recall balance. More notably, the macro average F-score of 0.72 surpasses that of the Decision Tree, suggesting improved generalisation across all classes, including minority ones. The weighted average F-score of 1.00, confirms the SVM model's strength in handling the more frequent classes.

While both models demonstrated high accuracy and strong classification capabilities, the Decision Tree slightly outperforms in terms of raw accuracy and F-score, making it well-suited for datasets with dominant classes and where anomaly detection is critical. Interestingly, the Polynomial Kernel SVM showed stronger generalisation across different types of data, especially when the class distribution was uneven. This is reflected in its higher macro average F-score, which suggests it's a more balanced option when working with datasets that contain a mix of common and rare categories.

That said, neither model was perfect. Both made classification errors, pointing to deeper challenges that can affect how well these algorithms perform. One major issue lies in the dataset itself. The NSL-KDD dataset used in this study has a noticeable imbalance, as, some types of attacks appeared far more frequently than others. This can cause models to favour the majority classes, making them less effective at detecting the rarer, but often more critical, types of attacks.

5. Conclusion

While considering network security, effective anomaly detection is critical for identifying threats and maintaining system integrity. This study compared the performance of two machine learning models: Decision Tree and Polynomial Kernel Support Vector Machine (SVM) and applied them to the NSL-KDD dataset, with a focus on classifying network traffic as either normal or anomalous.

The Decision Tree model achieved an impressive accuracy of 99.7% and an F-score of 0.997, indicating its strong ability to detect anomalies while maintaining a balance between precision and recall. Its hierarchical structure makes it especially effective for large datasets with many features, allowing it to form clear decision boundaries. The weighted average F-score of 1.00 re-echoed its strength in identifying the dominant classes. However, the macro average F-score of 0.68 reveals a drop in performance across less frequent classes, suggesting the model may struggle with minority attack types. The Polynomial Kernel SVM, while slightly behind in overall accuracy at 99.5%, also delivered a high F-score of 0.995, showing reliable precision and recall. Notably, its macro average F-score of 0.72 outperformed the Decision Tree, indicating more consistent classification across all classes, including rare or complex anomalies. This broader generalisation is especially valuable in network security, where threats can vary widely. Like the Decision Tree, the SVM also achieved a weighted average F-score of 1.00, confirming its effectiveness with the majority classes.

It is clear that both models proved highly capable, but each offers distinct advantages. While DT was best suited for environments prioritising detection of common threats and interpretability. Its simplicity and speed make it ideal for real-time systems. Polynomial Kernel SVM on the other hand, was more robust in handling diverse and infrequent threats, making it preferable in scenarios requiring high generalisation and nuanced classification. These findings reinforce the growing role of machine learning in communication systems, with applications extending beyond intrusion detection to areas like spectrum management [24], [25] and network performance evaluation [26], [27]. Looking at the future, there is a need to address class imbalance with a more balanced dataset.

Acknowledgement

This research was self-funded by the authors.

Conflict of Interest

The authors declare no conflict of interest regarding the publication of this paper.

Author Contribution

*The authors confirm their respective contributions to this study as follows: Emmanuel Ubom was responsible for the **conception and design of the study**. Andikan Okoro conducted the **data collection**. Both authors, Emmanuel Ubom and Andikan Okoro, jointly performed the **analysis and interpretation of the results**. The initial **draft of the manuscript** was prepared by Andikan Okoro and subsequently reviewed by Emmanuel Ubom. Both authors were involved in the **review processes** and have read and approved the final version of the manuscript.*

References

[1] Divekar, A., Parekh, M., Savla, V., Mishra, R., & Shirole, M. (2018) Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives, 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, Nepal, 1–8, https://doi.org/10.1109/CCCS.2018.8586840.

[2] Olugbenga, O.E.M., Kolajo, T., and Babatunde, J.A. (2024) Development and Evaluation of a Hybrid Machine Learning-Based Intrusion Detection System Using NSL-KDD Dataset, USci, vol. 3, no. 3, 277–283.

[3] Alrayes, F.S., Zakariah, M., Amin, S.U., Khan, Z.I., and Alqurni, J.S. (2024) CNN Channel Attention Intrusion Detection System Using NSL-KDD Dataset, Computers, Materials & Continua, vol. 79, no. 3, 4319–4347.

[4] Sukhadeo, B.S., Patil, R.N., Atole, R., Sinkar, Y.D., Patkar, U.C., and Chopade, R. (2023) MLIDS: A Machine Learning-Based Intrusion Detection System Using the NSLKDD Data, International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 4s, 167–179.

[5] Ao, H. (2021) Using machine learning models to detect different intrusion on NSL-KDD, 2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE), SC, USA, 166–177, https://doi.org/10.1109/CSAIEE54046.2021.9543241.

[6] Corli, S., Moro, L., Dragoni, D., Dispenza, M., and Prati, E. (2025) Quantum Machine Learning Algorithms for Anomaly Detection: A Review, Future Generation Computer Systems, vol. 166, Article no. 107632

[7] Sangve, S. M., and Thool, R. (2015) A formal assessment of anomaly network intrusion detection methods and techniques using various datasets, 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Davangere, India, 267–272, https://doi.org/10.1109/ICATCCT.2015.7456894.

[8] Sekhar, C., Pavani, K., and Rao, M. S. (2021) Comparative analysis on intrusion detection system through ML and DL techniques: Survey, 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nagpur, India, 1–5, https://doi.org/10.1109/ICCICA52458.2021.9697291.

[9] Guido, R., Ferrisi, S., Lofaro, D., and Conforti, D. (2024) An Overview on the Advancements of Support Vector Machine Models in Healthcare Applications: A Review, Information, vol. 15, no. 4, Article no. 235.

[10] Riswanto, U. (2025) What is the Polynomial Kernel? Unlocking the Power of Non-linear SVMs, Medium, Jul. 4. [Online]. Available: https://ujangriswanto08.medium.com/what-is-the-polynomial-kernel-unlocking-the-power-of-non-linear-svms-ed7619299ca1.

[11] Singh, S., and Banerjee, S. (2020) Machine learning mechanisms for network anomaly detection system: A review, 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 0976–0980, https://doi.org/10.1109/ICCSP48568.2020.9182197.

[12] Drogkoula, M., Kokkinos, K., and Samaras, N. (2023) A Comprehensive Survey of Machine Learning Methodologies with Emphasis in Water Resources Management, Applied Sciences, vol. 13, no. 22, Article no. 12147.

[13] Hong, R. F., Horng, S. C., and Lin, S. S. (2021) Machine learning in cyber security analytics using NSL-KDD dataset, 2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Taichung, Taiwan, 260–265, https://doi.org/10.1109/TAAI54685.2021.00057.

[14] Abrar, I., Ayub, Z., Masoodi, F., and Bamhdi, A. M. (2020) A machine learning approach for intrusion detection system on NSL-KDD dataset, 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 919–924, https://doi.org/10.1109/ICOSEC49089.2020.9215232.

[15] Agrawal, A. P., and Singh, N. (2021) Comparative analysis of SVM kernels and parameters for efficient anomaly detection in IoT, 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 1–6, https://doi.org/10.1109/ISCON52037.2021.9702398.

[16] Islam, S. H., Ab Rahman, M., Khan, S., Jan, M. A., and Lloret, J. (2020) Anomaly detection system using machine learning algorithms for IoT-based smart environments, IEEE Access, 8, 114066–114077, https://doi.org/10.1109/ACCESS.2020.3003711.

[17] Raj, J. G., Jolfaei, A., Rawat, D. B., and Masud, M. M. (2021) Artificial intelligence–enabled intrusion detection for edge–cloud computing: Performance evaluation and comparison, IEEE Transactions on Industrial Informatics, 17(10), 7081–7090, https://doi.org/10.1109/TII.2020.3045820.

[18] Yilmaz, A. A. (2022) Intrusion detection in computer networks using optimized machine learning algorithms, 2022 3rd International Informatics and Software Engineering Conference (IISEC), Ankara, Turkey, 1–5, https://doi.org/10.1109/IISEC56263.2022.9998258.

[19] Blockeel, H., Moro, L., Dragoni, D., Dispenza, M., and Prati, E. (2023) Decision Trees: From Efficient Prediction to Responsible AI, Frontiers in Artificial Intelligence, vol. 6, Article no. 1124553.

[20] Abedinia, A., and Seydi, V. (2024) Building Semi-Supervised Decision Trees with Semi-CART Algorithm, International Journal of Machine Learning and Cybernetics, vol. 15, 4493–4510.

[21] Mhamdi, L., Hamdi, H., and Mahmood, M. A. (2023) Network intrusion detection in software-defined network using deep and machine learning, GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 2692–2697, https://doi.org/10.1109/GLOBECOM54140.2023.10437050.

[22] Pervez, M. S., and Farid, D. M. (2014) Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs, The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Dhaka, Bangladesh, 1–6, https://doi.org/10.1109/SKIMA.2014.7083539.

[23] Andikan E. Okoro, Emmanuel A. Ubom, and Ubong S. Ukommi. (2024) Evaluation of Machine Learning Model for Network Anomaly Detection: Support Vector Machine, Journal of Engineering Research and Reports, 26(8), 326-342.

[24] Ubom, E. A., Akpanobong, A. C., and Ukommi, U. (2019) Spectrum occupancy in rural Nigeria: A case for a lightly licensed spectrum band for rural broadband enhancement, International Journal of Computer Science & Information Technology (IJCSIT), 11(4), 81–99, https://doi.org/10.5121/ijcsit.2019.11407.

[25] Ubom, E. A., and Ukommi, U. S. (2022) Comparative evaluation of spectrum occupancy of the broadcasting bands in urban, sub-urban, and rural environments, Nigerian Journal of Technology (NIJOTECH), 41(6), 1008–1016, https://doi.org/10.4314/njt.v41i6.11.

[26] Ubom, E. A., Akpanobong, A. C., and Abraham, I. I. (2019) Characterization of indoor propagation properties and performance evaluation for 2.4 GHz band Wi-Fi, International Journal of Wireless & Mobile Networks (IJWMN), 11(2), 27–38, https://doi.org/10.5121/ijwmn.2019.11203.

[27] Ukommi, U., Ubom, E. A., and Abraham, I. I. (2020) Outdoor 2.4 GHz band Wi-Fi networks performance characterisation, International Journal of Engineering Research & Technology (IJERT), 9(8), 207–212.