American Journal of Networks and Communications
2025, Vol. 14, No. 2, pp. 59-70
https://doi.org/10.11648/j.ajnc.20251402.14

Research Article

A Practical Study of Naive Bayes and Weighted Naive Bayes Cybersecurity Models for Wireless Networks

Oyinkansola Anuoluwapo Olagunju[1], Michael Abejide Adegoke[1, 3], Gabriel Babatunde Iwasokun[1, 2, *], Adeleke Johnson Adeyiga[1], Stephen Ojo Aderibigbe[3]

[1]Department of Computer Science and Information Technology, Bells University of Technology, Ota, Nigeria
[2]Department of Computer Software Engineering, Federal University of Technology, Akure, Nigeria
[3]Department of Computer Software Engineering, Lagos State University of Science and Technology, Ikorodu, Nigeria

Abstract

Intrusion Detection Systems (IDS) are essential for protecting wireless networks against an increasingly complex range of cyber threats. This study evaluates and compares the effectiveness of Na ïve Bayes, Weighted Na ïve Bayes, and Convolutional Neural Network (CNN) using the Network Security Laboratory-Knowledge Discovery and Data Mining (NSL-KDD) dataset. The Na ïve Bayes model was used as a baseline due to its simplicity and computational efficiency, delivering consistent results across multiple iterations. The Weighted Na ïve Bayes model, which incorporated a feature-weighting mechanism was used to improve precision and Receiver Operating Characteristics-Area Under Curve (ROC-AUC) scores while striking a balance between performance and interpretability. These qualities make it well-suited for real-time intrusion detection in wireless environments, where both transparency and resource efficiency are crucial. The Na ïve Bayes model was used as a baseline for offering a straightforward and efficient classification approach with moderate overall performance while the Weighted Na ïve Bayes model was used to enhance the standard version by introducing a feature-weighting mechanism, which improved precision and reduced false positives. The CNN model outperformed the Na ïve Bayes-based approaches across all evaluation metrics, underscoring its ability to learn complex patterns in network traffic. The Weighted Na ïve Bayes model was also used to strike a practical balance between accuracy and efficiency, making it especially suitable for wireless networks. The CNN model was used to deliver the highest scores across all evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, reflecting its ability to learn complex patterns in the data. Experimental results demonstrated the potential of the Weighted Na ïve Bayes model to support online learning and dynamic feature weighting, which is necessary for boosting adaptability to new and evolving attacks while preserving simplicity and transparency.

Keywords

Intrusion Detection System, Na ïve Bayes, Weighted Na ïve Bayes, Convolution Neural Networks, Wireless Networks

*Corresponding author: gbiwasokun@futa.edu.ng (Gabriel Babatunde Iwasokun)

Received: 14 August 2025; **Accepted:** 26 August 2025; **Published:** 15 September 2025

1. Introduction

The proliferation in contemporary network complexity and connectivity have increased the requirement for robust cybersecurity measures to protect sensitive information and other digital assets from the threat of cyberattacks. It has been reported that the number of devices connected to the Internet in October 2024 is 17 billion, with about 5.64 billion users. This number has been estimated to increase to up to 32.1 billion by 2030, and with the potential for each of them to act as a gateway for hackers [1]. The successive loss of potential due to the potential cybercrimes that could ensue is financially vast, with the cost of cybercrimes projected to exceed ten (10) trillion dollars worldwide in 2025 [2]. Thus, there is an urgent need for cybersecurity frameworks that can predict and counteract cyber threats to the point of detecting and preventing malicious behaviours in wireless networks. The traditional cybersecurity solutions, such as firewalls, antivirus software, intrusion detection systems (IDS) and vulnerability management, all of which generally fall within the technology category of perimeter security and signature-based detection, can only provide security against known threats but do not offer new defenses against advanced and complex attacks which usually are directed at critical infrastructures, businesses and wireless devices that frequently contain personal and sensitive information to be compromised with associated disruptions and tangible losses [3]. As a result, artificial intelligence and machine learning models reveal themselves as the modern promise that is gaining wide adoption in cybersecurity, the one that will one day have the fantastic tools and technologies to anticipate and combat previously unknown threats at amazing speed and efficiency [4]. With the advancement of artificial intelligence and machine learning, technologies capable of learning from large-scale data to detect behavioral patterns and uncover anomalies that may signal security threats are now within reach. While integrating AI models into cybersecurity has long been a focus for researchers, the journey has been slow and incremental. This study is organized as follows: Section 2 outlines current cybersecurity methods, and Section 3 reviews relevant literature. Sections 4 and 5 explain the foundational theories and mathematical models behind the Naïve Bayes and Weighted Naïve Bayes approaches to cybersecurity. Section 6 presents the experimental setup and findings, while Section 7 compares these models with a CNN. The final section, Section 8, provides conclusions and suggests directions for future research.

2. Some Existing Cybersecurity Approaches

Existing cybersecurity measures encompass a variety of tools and strategies, including firewalls, antivirus software, intrusion detection systems (IDS), vulnerability management, and machine learning-based methods. The firewall remains a foundational defense mechanism, acting as a gatekeeper that regulates network traffic according to predefined security rules [5]. Firewalls filter data packets, allowing only legitimate traffic through while blocking unauthorized or potentially harmful connections [6]. Serving as a barrier between trusted internal networks and external sources, firewalls inspect all incoming and outgoing traffic. While effective against basic threats like packet filtering and port scanning, firewalls fall short when it comes to more complex attacks such as zero-day vulnerabilities, social engineering, and advanced persistent threats. Their static nature, limited scope of analysis, and inability to assess user behavior render them insufficient for modern threat landscapes. Antivirus software, on the other hand, is a signature-based solution designed to detect and eliminate malware by comparing files and applications against a database of known threats [7]. These programs operate by scanning systems either in real-time or on-demand, identifying harmful software like viruses, worms, and trojans through pattern matching and heuristic techniques. While antivirus tools offer reliable protection against recognized threats and automate many aspects of threat removal, they are largely ineffective against advanced cyberattacks, including zero-day exploits, fileless malware, and polymorphic code. Furthermore, antivirus software may degrade system performance, produce false positives, and struggle to counter sophisticated evasion strategies, making them inadequate as standalone security solutions.

Intrusion Detection Systems (IDS) serve as a surveillance-based approach, continuously monitoring network traffic and system behavior to uncover suspicious activities that could indicate a breach [8]. IDS methods can be signature-based, anomaly-based, or a hybrid of both, aiming to identify unauthorized access attempts, policy violations, and known attack patterns. They collect data from multiple network sources and logs, analyze these inputs, and raise alerts when potential threats are discovered. IDSs provide valuable insights for forensic investigations and improve network visibility. However, they are often plagued by high false positive rates, difficulty in detecting novel attacks, and a reliance on up-to-date rules and configurations [9]. These systems also have limitations in handling encrypted traffic, are vulnerable to sophisticated evasion techniques, and may be overwhelmed by high data volumes, necessitating expert management and making them less suitable for detecting stealthy or advanced threats. Vulnerability management takes a systematic approach to identifying, assessing, prioritizing, and mitigating weaknesses in software, systems, and networks [10]. This ongoing process includes scanning, evaluation, and patching, using a combination of automated tools, manual inspections, and regular audits. It is effective in mapping system vulnerabilities, prioritizing fixes based on risk, and ensuring compliance with industry regulations. However, it also faces sig-

nificant limitations: it cannot detect zero-day flaws that are unknown to current tools, often uncovers more vulnerabilities than can be feasibly addressed, and depends on the timely deployment of patches within complex infrastructures [11]. Vulnerability management tools may also produce false positives, face resource constraints, and pose challenges in balancing system uptime with patch application, limiting their effectiveness as comprehensive security solutions.

Given the shortcomings of these traditional methods, there is a growing shift toward machine learning techniques capable of adapting to evolving threats and recognizing new attack patterns. Unlike static rule-based systems, machine learning models can analyze historical data, uncover trends, and predict potential threats without relying solely on predefined signatures [12]. Algorithms like Decision Trees, Random Forests, Support Vector Machines, k-Nearest Neighbors, and Naïve Bayes have been employed to classify network traffic and detect anomalies. These models automate the identification of complex patterns across endpoints, network traffic, and cloud environments, enabling faster threat detection and reducing the risk of breaches. As a result, machine learning approaches support more dynamic, proactive, and scalable cybersecurity strategies compared to conventional tools [13].

3. Synopsis of Some Related Works

The field of cybersecurity in wireless networks has made significant strides, particularly with the integration of artificial intelligence and machine learning for threat detection and mitigation. Joubari and Guizani [14] developed a hybrid deep learning model for intrusion detection in Internet of Things (IoT) environments, combining Convolutional Neural Networks (CNNs) with Bidirectional Long Short-Term Memory (BiLSTM) networks. Evaluated on the UNSW-NB15 dataset, the model performed well in both binary and multiclass classification tasks. However, its static architecture presents a major limitation. Since the model is trained offline and deployed without ongoing updates or dynamic learning, it cannot adapt to new or evolving threats. This static design is particularly problematic in wireless networks, where the threat landscape changes rapidly and unpredictably. In a comparative study, Alshuaibi et al**.** [15] examined multiple deep learning models, including CNNs, LSTMs, BiLSTMs, GRUs, and hybrid CNN-LSTM structures, using the NSL-KDD dataset. Among these, the CNN-LSTM combination demonstrated the highest accuracy, exceeding 95%. Although the study was methodologically robust, all models relied solely on offline training and feature selection techniques like Recursive Feature Elimination (RFE) and Decision Trees. None of them included mechanisms for online learning or retraining, making them vulnerable to emerging threats and reducing their practicality for real-world wireless environments that demand adaptability. Almosti and Rahman [9] introduced a model using Temporal Convolutional Networks (TCNs) for multiclass intrusion detection, achieving high accuracy on the Edge-IIoTset dataset, well-suited for industrial wireless networks. TCNs effectively capture long-term dependencies in sequential data, making them ideal for network traffic analysis. However, like other models, this one was trained on static datasets and cannot learn from new data post-deployment. As a result, the model risks becoming outdated over time, limiting its effectiveness in environments where cyber threats constantly evolve.

Thomas [16] proposed a hybrid CNN-LSTM model for intrusion detection in industrial wireless sensor networks, evaluated on datasets such as UNSW-NB15 and X-IIoTID. While the model achieved impressive accuracy rates over 99% in some cases, it also follows a batch learning paradigm, with all training completed before deployment. This lack of post-deployment adaptability limits its usefulness in dynamic environments where system configurations and vulnerabilities change frequently. Kamal and Mashaly [17] presented a hybrid architecture integrating stacked autoencoders, LSTM networks, and CNN layers for anomaly detection in IoT settings. Tested on datasets like CICIoT2023, the model achieved detection accuracies above 99%. However, despite its sophistication, it shares a common flaw: it is pre-trained and static, lacking the capacity for continual learning or self-adjustment. In diverse wireless environments with changing device configurations and emerging threat vectors, this rigidity undermines the model's long-term viability. Parker [18] developed CNN-Self-Attention Long Short-Term memory model combining CNNs with a self-attention-enhanced LSTM. Designed to extract both spatial and temporal features from network traffic, this model performed well across several intrusion detection benchmarks, achieving around 97% accuracy. Yet, it too suffers from static deployment, lacking any form of adaptive learning. In environments where threats evolve continuously, such as wireless networks, the model's inability to update or retrain itself limits its real-world applicability. Ojo [19] introduced a Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model tested on datasets like CICIoT2023 and CICIDS2017, aiming to capture spatial and temporal network feature representations. While the model excelled in controlled settings, it did not include mechanisms for evolving threat detection. Without a feedback loop or adaptive capabilities, the model becomes less reliable as new attack patterns emerge. Singh and Jang-Jaccard [20] proposed an unsupervised deep learning model based on a Multi-Scale CNN and LSTM autoencoder, aimed at detecting anomalies using datasets like NSL-KDD and UNSW-NB15. The unsupervised approach reduces dependence on labeled data, but the model remains non-adaptive. It lacks mechanisms for incremental learning or dynamic threshold adjustments, making it vulnerable to shifts in data distributions, an issue particularly relevant in wireless networks with frequently changing usage patterns.

Hartono et al. [21] developed a federated learning-based intrusion detection model using Spatial CNNs (SCNNs) and

Directional Long Short-Term memory (BiLSTM) networks, enabling privacy-preserving training across distributed wireless sensor networks. Despite this decentralized setup, the training occurs offline, and updates are centralized. Edge nodes use a static model until new global versions are redistributed, introducing a lag in detecting emerging threats. ConvLSTM introduced a Convolutional LSTM model to enhance security in Unmanned Aerial Vehicles (UAVs) and smart grid systems. It performed well on datasets like KDD99 and CICIDS2017, effectively modeling spatio-temporal relationships. However, the model was strictly trained offline and lacks mechanisms for online learning or data drift adaptation. This makes it ill-suited for dynamic systems like UAVs, where threat landscapes shift in real-time. Muniyandi et al. [22] designed a lightweight intrusion detection model for wireless sensor networks using a hybridized Gaussian Naive Bayes classifier. The model achieved practical performance using statistical features from the NSL-KDD dataset and was noted for its simplicity and low computational cost. Nevertheless, it was built as a static model, incapable of adapting or learning post-deployment. Its reliance on feature independence also limits its ability to handle complex and evolving attack patterns, such as polymorphic threats. Moreover, it was not evaluated under dynamic or real-time conditions, further restricting its deployment potential.

Okdem and Okdem [23] implemented a Multinomial Naive Bayes classifier for wireless sensor network security, emphasizing reduced complexity and memory usage. While effective on preprocessed KDD datasets, the model is inherently simplistic and assumes fixed feature likelihoods. It does not adapt to time-based behavior changes or novel attack types, and no mechanisms for incremental learning were introduced. As such, the model is poorly suited for the fast-changing conditions in modern wireless networks. Iwendi et al. [24] used a Particle Swarm Optimization-based feature selection method with a Naive Bayes classifier for detecting intrusions in IoT and edge environments. This approach improved detection speed by reducing data dimensionality. However, the model still operates in a static mode, with offline training and no ongoing updates. Any newly emerging threats would go unnoticed unless the model is manually retrained, a process that is not practical in dynamic environments. The absence of a feedback loop or adaptive component highlights the need for more flexible and responsive solutions. Adegoke et al. [25] noted that the Naive Bayes algorithm is based on Bayes' theorem and assumes conditional independence among features, an assumption that often does not hold in real-world cybersecurity scenarios. The probability of a class C_k given a feature vector $x = (x_1, x_2, \cdots, x_n)$ is given by:

$$P(C_k|\boldsymbol{x}) = \frac{P(C_k)\cdot\prod_{i=1}^{n} P(x_i|C_k)}{P(\boldsymbol{x})} \quad (1)$$

The goal is to find out whether an incoming transaction x is a threat or not. This is obtained by computing the maximum a posteriori class given by:

$$\hat{C} = \underset{C_k \in C}{\arg max}\, P\,(C_k) \cdot \prod_{i=1}^{n} P\,(x_i\,|C_K) \quad (2)$$

Where $\hat{C}$ is the predicted class, $P(C_k)$ is the prior probability of class C_k, $P(x_i|C_k)$ is the likelihood of a feature x_i given class C_k and C is the set of all possible classes. The anticipated threat types were predefined and stored in an online database. However, if attackers develop new techniques not included in the training data, the cybersecurity model may fail to recognize them as threats, potentially allowing these attacks to go undetected and cause significant damage. This shortcoming stems from a fundamental limitation of the Naïve Bayes algorithm-it assumes that all features are independent, which is rarely the case in real-world scenarios. Additionally, Naïve Bayes suffers from the zero-frequency problem, where it assigns a zero probability to any categorical variable that was not observed in the training set. This can lead to incorrect or missed detections. To mitigate this issue, Laplace smoothing is typically applied, which involves adding one to each count to avoid zero probabilities and ensure more stable estimates.

The challenges observed in previous research highlight the urgent need for an adaptive cybersecurity model tailored to wireless networks that can respond in real-time to evolving threats. To address this, a Weighted Naïve Bayes framework is proposed. This approach supports online updates and dynamic feature weighting, enabling the model to move beyond the limitations of static threat detection. As a result, the model can deliver more adaptive and proactive defense capabilities suited for today's rapidly changing cybersecurity landscape.

4. The Naïve Bayes Model

The classification model used in this study is the Gaussian Naïve Bayes (GNB) classifier. This section outlines the mathematical foundation and rationale behind the model, along with the evaluation metrics used to assess its performance. The GNB classifier is built on Bayes' Theorem, which calculates the posterior probability of a class given a set of input features. This allows the model to make predictions based on the likelihood of observed features belonging to a particular class given a feature vector as $x = (x_1, x_2, \cdots, x_n)$. Using Bayes' Theorem:

$$P\,(y\,|x) = \frac{P(y)P(x\,|y)}{P\,(x)} \quad (3)$$

Since $P\,(x)$ is the same for all classes; it is ignored in comparison.

$$\hat{y} = arg\,\underset{y}{max}\, P(y)P(x\,|y) \quad (4)$$

Since Naïve Bayes assumes features are conditionally independent, we have:

$$P\,(x\,|y) = \prod_{i=1}^{n} P\,(x_i\,|y) \quad (5)$$

For continuous features, Gaussian Naïve Bayes assumes each feature follows a normal distribution:

$$P(x_i|\, y) = \frac{1}{\sqrt{2\pi\sigma_{iy}^2}} exp\left(-\frac{(x_i-\mu_{iy})^2}{2\sigma_{iy}^2}\right) \quad (6)$$

Where μ_{iy} and σ_{iy}^2 are the mean and variance of the feature x_i within class y.

For the final classification rule, the log is taken to simplify and avoid underflow as shown below:

$$\hat{y} = arg\, {}_y^{max} \log P(y) - \tfrac{1}{2} \sum_{i=1}^{n} \left[\log(2\pi\sigma_{iy}^2) + \frac{(x_i - \mu_{iy})^2}{\sigma_{iy}^2}\right] \quad (7)$$

4.1. Feature Selection via Mutual Information

To enhance model performance and reduce dimensionality, Mutual Information (MI) was employed for feature selection. MI quantifies the dependency between a given feature *x* and the target class *y*, helping identify which features provide the most relevant information for classification. It is computed using the following formula:

$$I(X;Y) = \sum_{x\epsilon X} \sum_{y\epsilon Y} P(x,y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right) \quad (8)$$

Features with higher mutual information values play a more significant role in the prediction process and were given priority during selection. Based on their MI scores, the top 10 most informative features were chosen for use in the model.

4.2. Feature Normalization

All numerical features were standardized using Z-score normalization, a method that transforms each feature value *x* so that the resulting distribution has a mean of 0 and a standard deviation of 1:

$$z = \frac{x-\mu}{\sigma} \quad (9)$$

This step ensures that all features contribute equally to the Gaussian probability estimates, preventing any single feature from disproportionately influencing the model due to differences in scale.

5. The Weighted Naïve Bayes Model

To improve the performance of the standard Naive Bayes classifier, a Weighted Naive Bayes (WNB) model was developed by incorporating feature relevance into the classification process. This was implemented through a custom Feature Weighted Gaussian Naive Bayes approach, which modifies the computation of log-likelihoods by assigning weights to individual features. These weights were derived from mutual information scores using the SelectKBest method, allowing the model to emphasize the most informative features for distinguishing between normal and malicious traffic. The model was trained using the top 20 features selected from the NSL-KDD dataset, and classification was carried out on the test data. Predictions were generated by calculating a weighted joint log-likelihood, ensuring that more influential features had a greater impact on the classification outcome. This weighting strategy enabled the model to better detect intrusions by prioritizing critical network attributes.

Each feature is multiplied by a weight w_i before training:

$$x_i' = w_i \cdot x_i \quad (10)$$

Where w_i is $w_i \in [0,1]$

The weights are then computed using MI between x_i and y:

$$w_i = \frac{MI\,(x_{i;Y})}{max_j MI\,(x_{j;Y})} \quad (11)$$

Using the same Naïve Bayes formula, we replace x_i with x_i':

$$\hat{y} = arg\, {}_y^{max} \log P(y) - \tfrac{1}{2} \sum_{i=1}^{n} \left[\log(2\pi\sigma_{iy}^2) + \frac{(x_i' - \mu_{iy})^2}{\sigma_{iy}^2}\right] \quad (12)$$

The MI scores are normalized to derive feature weights wi, which are then used to scale each feature. This transforms the input vector to $x^i = [w_1x_1w_2x_2 \ldots\ldots, w_nx_n]$, emphasizing more informative.

6. Experimental Analyses and Evaluation

The experimental analysis in this study was carried out using the NSL-KDD dataset (KDDTrain+.txt and KDDTest+.txt), an improved version of the KDD Cup 1999 dataset that addresses issues such as redundancy and class imbalance. The dataset contains 41 features describing network traffic and includes labeled instances classified as either normal or attack. To evaluate model performance, three classifiers were tested: the standard Naive Bayes (NB), the Weighted Naive Bayes (WNB), and a Convolutional Neural Network (CNN). For simplification, the dataset's multi-class labels were converted to a binary format "normal" was mapped to 0 and all attack types to 1, allowing the models to focus specifically on intrusion detection. Before classification, preprocessing and feature selection were applied to effectively differentiate between normal and malicious traffic. Mutual information was used as the scoring function for feature selection, identifying the top 20 features most strongly associated with the target label. This process helped reduce dimensionality, filter out noise, and enhance computational efficiency. These selected features were then assigned scores and normalized into weights that reflect their relative im-

portance in detecting intrusions. Standard scaling was applied afterward to ensure that all features had a mean of zero and a standard deviation of one. This normalization step is essential for models like Naive Bayes, which rely on probabilistic assumptions, to function optimally.

6.1. Results and Discussion

This section presents and analyzes the experimental results from evaluating the Na ïve Bayes, Weighted Na ïve Bayes, and CNN models for intrusion detection in wireless networks using the NSL-KDD dataset. Each model was assessed using standard classification metrics; accuracy, precision, recall, F1-score, and ROC-AUC to provide a comprehensive evaluation of their performance. The Na ïve Bayes model was used as a baseline due to its simplicity and ease of interpretation, making it a popular choice in early-stage intrusion detection research. The Weighted Na ïve Bayes model introduced a feature-weighting mechanism designed to improve detection accuracy and reduce false positives. In contrast, the CNN model employed a more sophisticated deep learning architecture capable of automatically learning complex patterns in the data. By comparing the performance of these models across multiple runs, this section highlights the trade-offs between detection accuracy, computational efficiency, and practical deployment in wireless network environments. As shown in Figure 1, the standard Na ïve Bayes model correctly identified 9,109 normal instances and 7,655 attack instances, demonstrating a strong ability to recognize legitimate traffic along with a fair level of attack detection.

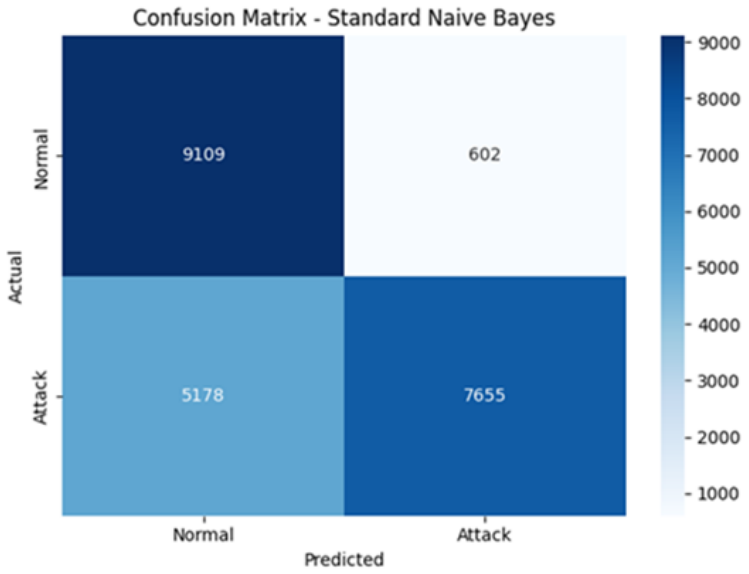

Figure 1. *Confusion matrix for the Naive Bayes model.*

However, it also misclassified 5,178 attack instances as normal, leading to a high false negative rate, a serious concern in cybersecurity, where undetected threats can compromise system integrity. On the other hand, only 602 normal instances were incorrectly flagged as attacks, resulting in a low false positive rate and contributing to a high precision score of 92.7%. The model's overall performance was evaluated using five key classification metrics. As shown in Table 1, the standard Na ïve Bayes model achieved an accuracy of 0.7436, meaning about 74.36% of all predictions were correct. The precision score of 0.9271 indicates that over 92% of the predicted attacks were indeed attacks. However, the recall score was 0.5965, revealing that the model detected only about 59% of actual attacks, thereby missing a significant portion. The F1-score, which balances precision and recall, was 0.7259, indicating moderate effectiveness in managing the trade-off between false positives and false negatives.

Table 1. *Classification report of attack types using Na ïve Bayes Model.*

	Precision	Recall	F1-score
0	0.6376	0.9380	0.7591
1	0.9271	0.5965	0.7259

	Precision	Recall	F1-score
Accuracy	0.7436	0.7436	0.7436
Macro average	0.7823	0.7673	0.7425
Weighted average	0.8024	0.7436	0.7402

Table 2 presents the results from ten (10) iterations of the Naïve Bayes model. The evaluation metrics remained consistent throughout, with an accuracy of 0.7708, precision of 0.9160, recall of 0.6578, F1 score of 0.7657, and ROC-AUC of 0.8397. This consistency across multiple runs suggests that the model delivers stable performance and is not significantly influenced by variations in the training process. Such reliability underscores its potential effectiveness for intrusion detection using the NSL-KDD dataset.

Table 3 summarizes the average performance of the Naïve Bayes model across ten iterations. The model recorded an average accuracy of 0.7708, precision of 0.9160, recall of 0.6578, F1-score of 0.7657, and ROC-AUC of 0.8397. These results confirm the model's consistent and dependable behavior throughout all runs, reinforcing its reliability for intrusion detection tasks.

Table 2. *Results of Ten (10) iterations of the Naïve Bayes Model.*

Iteration	Accuracy	Precision	Recall	F1 Score	ROC_AUC
1	0.7708	0.9160	0.6578	0.7657	0.8397
2	0.7708	0.9160	0.6578	0.7657	0.8397
3	0.7708	0.9160	0.6578	0.7657	0.8397
4	0.7708	0.9160	0.6578	0.7657	0.8397
5	0.7708	0.9160	0.6578	0.7657	0.8397
6	0.7708	0.9160	0.6578	0.7657	0.8397
7	0.7708	0.9160	0.6578	0.7657	0.8397
8	0.7708	0.9160	0.6578	0.7657	0.8397
9	0.7708	0.9160	0.6578	0.7657	0.8397
10	0.7708	0.9160	0.6578	0.7657	0.8397

Table 3. *Average Performance of the Naïve Bayes model across Ten (10) iterations.*

Average Accuracy	Average Precision	Average Recall	Average F1-score	Average ROC_AUC
0.7708	0.9160	0.6578	0.7657	0.8397

These results establish a performance benchmark for the standard Naïve Bayes model. While it achieves high precision and ROC-AUC values, its relatively low recall indicates that it may not be sensitive enough to detect all types of attacks. This shortcoming highlights the need for further enhancements, such as integrating feature weighting to improve the model's ability to identify a wider range of intrusions more effectively. As shown in Figure 2, the results from the Weighted Naïve Bayes model show that it correctly classified 9,484 normal traffic instances and misclassified only 227 as attacks, resulting in a very low false positive rate. For attack traffic, it successfully detected 7,367 instances but failed to identify 5,466, wrongly labeling them as normal. This performance indicates high precision and moderate recall, meaning the model is highly accurate when predicting an attack but tends to miss a significant number of actual attacks. The conservative nature of its predictions helps reduce false alarms, which is particularly valuable in real-time intrusion detection sys-

tems where alert fatigue is a concern.

Table 4 outlines the classification results of the Weighted Naïve Bayes model for two classes: class 0 (Normal) and class 1 (Attack). The model achieved a recall of 0.9766 for normal traffic, demonstrating strong capability in correctly identifying legitimate activity. However, the corresponding precision for normal traffic was lower at 0.6343, suggesting that many instances predicted as normal were actually attacks. For attack traffic, the model achieved a very high precision of 0.9701 but a lower recall of 0.5740, indicating that while its attack predictions were highly accurate, it missed a considerable number of real threats.

The F1-scores further reflect this trade-off, with scores of 0.7691 for normal traffic and 0.7212 for attack traffic. The model's overall accuracy was 0.7474, which is consistent with both the macro-average and weighted-average scores. A macro average F1-score of 0.7452 suggests balanced performance across both classes, while the weighted average F1-score of 0.7419, which accounts for class distribution, confirms the model's dependable performance across the dataset.

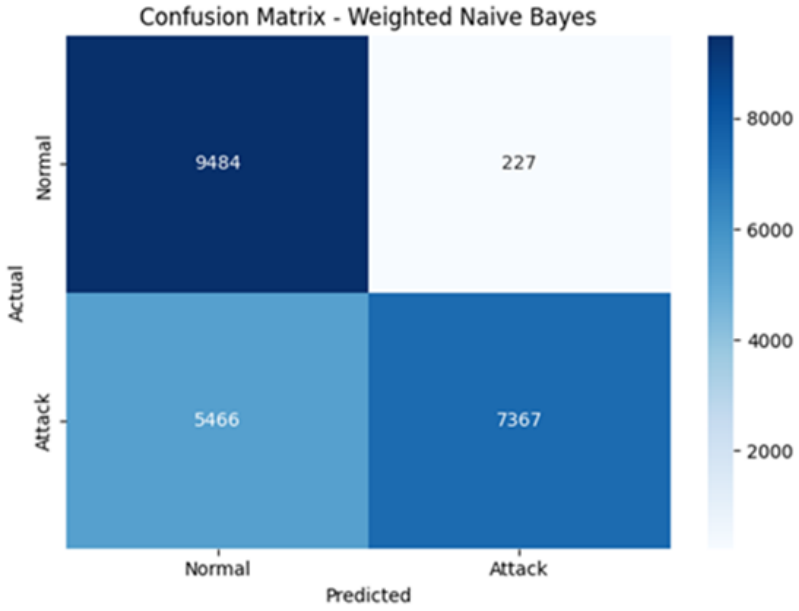

Figure 2. *Confusion matrix for the Weighted Naive Bayes model.*

Table 4. *Classification report of attack types using Weighted Naïve Bayes Model.*

	Precision	Recall	F1-score
0	0.6343	0.9766	0.7691
1	0.9701	0.5740	0.7212
Accuracy	0.7474	0.7474	0.7474
Macro average	0.8022	0.7753	0.7452
Weighted average	0.8255	0.7474	0.7419

Table 5 presents the results from ten (10) iterations of the Weighted Naïve Bayes model. The performance metrics remained consistent throughout, with accuracy ranging from 0.7605 to 0.7651. Precision was consistently high, falling between 0.9232 and 0.9254, while recall was slightly lower, ranging from 0.6300 to 0.6406. F1-scores were steady around 0.75, and the ROC-AUC values were strong, peaking at 0.9142. These findings demonstrate that the model delivers stable and accurate predictions, consistently favoring high precision while maintaining a reasonable level of recall across multiple runs.

Table 5. Results of Ten (10) iterations of the Weighted Naïve Bayes Model.

Iteration	Accuracy	Precision	Recall	F1 Score	ROC_AUC
1	0.7651	0.9232	0.6406	0.7564	0.9131
2	0.7635	0.9228	0.6379	0.7543	0.8529
3	0.7634	0.9244	0.6363	0.7538	0.9142
4	0.7623	0.9244	0.6343	0.7523	0.8844
5	0.7631	0.9240	0.6363	0.7536	0.8778
6	0.7639	0.9234	0.6382	0.7548	0.8775
7	0.7623	0.9245	0.6342	0.7523	0.8783
8	0.7623	0.9243	0.6344	0.7524	0.8841
9	0.7605	0.9254	0.6300	0.7497	0.9156
10	0.7627	0.9246	0.6349	0.7529	0.9150

Table 6 shows that, on average, the Weighted Naive Bayes model achieved an accuracy of 0.7629, a precision of 0.9241, a recall of 0.6357, an F1 score of 0.7532, and a ROC-AUC of 0.8913. Compared to the standard Naive Bayes model which recorded an accuracy of 0.7708, precision of 0.9160, recall of 0.6578, F1 score of 0.7657, and ROC-AUC of 0.8397. The Weighted Naive Bayes model offers a meaningful trade-off. While its accuracy and recall are slightly lower, it surpasses the standard model in precision and shows a notably higher ROC-AUC. This indicates an improved ability to differentiate between normal and attack traffic across various classification thresholds.

Table 6. Average Performance of the Weighted Naïve Bayes model across Ten (10) iterations.

AverAcc	Average Precision	Average Recall	Average F1-score	Average ROC_AUC
0.7629	0.9241	0.6357	0.7532	0.8913

These results highlight the practical effectiveness of the Weighted Naive Bayes model in intrusion detection. By incorporating a feature-weighting mechanism, the model enhances its precision by giving greater importance to the most informative features, resulting in a more targeted and cautious classification approach. In real-world applications, particularly those involving real-time network monitoring, minimizing false positives is essential for maintaining trust in the system and ensuring usability. As such, while the model may miss some attack instances, it provides a balanced and dependable solution in scenarios where reducing false alarms is more critical than achieving maximum detection coverage. Looking ahead, future improvements could explore the integration of adaptive or online learning techniques to boost recall while preserving the model's strong precision and discriminative capabilities.

6.2. Comparative Evaluation with CNN Model

To benchmark the performance of the proposed models, a Convolutional Neural Network (CNN) was implemented as a comparative deep learning approach. CNNs are widely recognized for their ability to automatically learn complex patterns from data. In this study, the CNN model outperformed other models across most evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC when tested on the NSL-KDD dataset. These results underscore CNNs' effectiveness in capturing nonlinear relationships and extracting high-level features, which contributes to their strong classification performance. As shown in Table 7, the CNN achieved an average accuracy of 97.20%, precision of 98.10%, recall of 99.06%, F1-score of 98.58%, and ROC-AUC of 88.48% on the test set. This high level of performance demonstrates the model's strong capability to distinguish between normal and malicious traffic, particularly due to its advanced feature extraction capabilities. However,

despite its accuracy, the CNN model has practical limitations. It demands significant computational power, involves longer training times, and lacks interpretability compared to traditional machine learning models. These drawbacks make it less suitable for deployment in real-time intrusion detection systems, especially in wireless network environments where efficiency and model transparency are vital. In comparison, the Weighted Naïve Bayes model, while slightly lower in overall accuracy, delivered competitive results with much lower computational complexity and greater interpretability. It maintained strong precision and a reduced false positive rate, making it a more practical option for lightweight security applications. Thus, while the CNN model sets a high-performance benchmark, the findings reinforce the value of the Weighted Naïve Bayes model as a more deployable and explainable solution for intrusion detection in wireless network environments.

Table 7. Performance Comparison of Naïve Bayes, Weighted Naïve Bayes, and CNN Models.

Model	Accuracy	Precision	Recall	F1-Score	ROC_AUC Curve
Naïve Bayes	0.7708	0.9160	0.6578	0.7657	0.8397
Weighted Naïve Bayes	0.7629	0.9241	0.6357	0.7532	0.8913
CNN	0.9720	0.9810	0.9906	0.9858	0.8848

7. Conclusions

This study examined and compared the performance of three classification models: Naïve Bayes, Weighted Naïve Bayes, and Convolutional Neural Network (CNN) for intrusion detection in wireless networks using the NSL-KDD dataset. The Naïve Bayes model was used as a baseline due to its simplicity and computational efficiency, delivering consistent results across multiple iterations. The Weighted Naïve Bayes model, which incorporated a feature-weighting mechanism, showed improved precision and ROC-AUC scores while striking a balance between performance and interpretability. These qualities make it well-suited for real-time intrusion detection in wireless environments, where both transparency and resource efficiency are crucial. The CNN model outperformed the Naïve Bayes-based approaches across all evaluation metrics, underscoring its ability to learn complex patterns in network traffic. However, its high computational demands, longer training time, and limited interpretability pose significant barriers to deployment in real-world wireless network infrastructures, particularly those with constrained resources. While CNN serves as a strong performance benchmark, the Weighted Naïve Bayes model offers practical advantages in terms of speed, clarity, and ease of implementation, highlighting its relevance for intrusion detection applications. Despite its strengths in precision and interpretability, the static nature of the Weighted Naïve Bayes model limits its ability to adapt to evolving attack vectors commonly found in real-world wireless environments. To overcome this limitation, future research should aim to develop an Adaptive Weighted Naïve Bayes model. This improved version would include capabilities such as online learning, incremental updates and dynamic feature weighting, allowing it to respond to new threats without requiring complete retraining. Incorporating automated assessments of feature relevance would also help the model maintain efficiency while enhancing its detection capabilities over time. Evaluating such an adaptive model in live wireless network scenarios, under diverse conditions and attack intensities, will be essential in advancing intrusion detection systems toward more intelligent and responsive security solutions.

Abbreviations

IDS	Intrusion Detection Systems
CNN	Convolutional Neural Network
ROC-AUC	Receiver Operating Characteristics-Area Under Curve
NSL-KDD	Network Security Laboratory-Knowledge Discovery and Data Mining
BiLSTM	Bidirectional Long Short-Term Memory
IoT	Internet of Things
BiLSTM	Directional Long Short-Term memory
UAVs	Unmanned Aerial Vehicles
GNB	Gaussian Naïve Bayes
MI	Mutual Information
WNB	Weighted Naive Bayes

Author Contributions

Oyinkansola Anuoluwapo Olagunju: Conceptualization, Data Collection and curation

Michael Abejide Adegoke: Methodology, Project administration, Resources, writing original draft

Gabriel Babatunde Iwasokun: Data curation, formal

analysis, experimentation, writing original draft

Adeleke Johnson Adeyiga: Formal Analysis, investigation, software

Stephen Ojo Aderibigbe: Validation, Writing and reviewing manuscript

Funding

This work is not supported by any external funding.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] IDC. (2024, October). *Worldwide connected devices forecast grows to 17 billion*. International Data Corporation. *Available from* https://www.idc.com

[2] *Cybersecurity Ventures. (2020, November 13). Cybercrime to cost the world $10.5 trillion annually by 2025*. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

[3] Riggs, H., Tufail, S., Parvez, I., Tariq, M., & Khan, M. (2023). Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure. *Sensors*, 23(8), 4060. https://doi.org/10.3390/s23084060

[4] Momen, H., Essam, M., Bahaa, M., Mohamed, A., Gomaa, M., Hany, M., & Elsersy, W. (2024, July 8). *Evaluating predictive models in cybersecurity: A comparative analysis of machine and deep learning techniques for threat detection* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2407.06014

[5] Nazre, A., Singh, P., & Kumar, S. (2024). Machine Learning Approaches for Enhancing Cybersecurity in Networked Systems. *Journal of Emerging Cybersecurity Technologies*, 18(1), 12-27. https://doi.org/10.5678/ject.v18i1.2345

[6] Muthusubramanian, M., Mohamed, I. A., & Pakalapati, N. (2024). *Machine learning for cybersecurity threat detection and prevention. International Journal of Innovative Science and Research Technology, 9*(2). https://doi.org/10.5281/zenodo.10776751

[7] Cybersecurity and Infrastructure Security Agency (2024). Artificial Intelligence Use Cases. Retrieved from https://www.cisa.gov/ai/cisa-use-cases

[8] Mukkamala, S., Sung, A. H., & Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications, 28(2), 167-182.

[9] Almosti, A. M., & Rahman, M. M. H. (2025). Analysis of Data Privacy Breaches Using Deep Learning in Cloud Environments: A Review. *Electronics*, *14*(13), 2727. https://doi.org/10.3390/electronics14132727

[10] Hammad, M. A., & Abd El-Latif, A. A. (2024). *Artificial intelligence for biometrics and cybersecurity: Technology and applications* (Eds.). Institution of Engineering and Technology.

[11] Alfahaid, A., Alalwany, E., Almars, A. M., Alharbi, F., Atlam, E., & Mahgoub, I. (2025). *Machine learning-based security solutions for IoT networks: A comprehensive survey. Sensors, 25*(11), 3341. https://doi.org/10.3390/s25113341

[12] Sarker, I. H. (2020). Cyberlearning: Effectiveness Analysis of Machine Learning Security Modeling to Detect Cyber-Anomalies and Multi-Attacks. arXiv preprint arXiv: 2104.08080. https://arxiv.org/abs/2104.08080

[13] Dhameliya, N., Patel, B., & Patel, K. B. (2024). Machine learning in cybersecurity: A comprehensive analysis of intrusion detection systems. *Journal of Sustainable Solutions*, *1*(4), 38-42.

[14] Joubari, A., & Guizani, M. (2024). Enhancing Cyber Security through Artificial Intelligence and Machine Learning: A Literature Review. Journal of Cyber Security, 6(1), 89-116. https://doi.org/10.32604/jcs.2024.056164

[15] *Alshuaibi, A., Almaayah, M., & Ali, A. (2025). Machine learning for cybersecurity issues: A systematic review. Journal of Cyber Security and Risk Auditing, 2025*(1), 36-46. https://doi.org/10.63180/jcsra.thestap.2025.1.4

[16] Thomas, M. A. (2023). *Machine Learning Applications for Cybersecurity*. *The Cyber Defense Review*, *8*(1), 87-99. Retrieved from https://cyberdefensereview.army.mil/Portals/6/Documents/2023_Spring/Thomas_CDRV8N1-Spring-2023.pdf

[17] Kamal, H., & Mashaly, M. (2025). Enhanced Hybrid Deep Learning Models-Based Anomaly Detection Method for Two-Stage Binary and Multi-Class Classification of Attacks in Intrusion Detection Systems. *Algorithms*, *18*(2), 69. https://doi.org/10.3390/a18020069

[18] Parker, D. (2023). *The role of firewalls in modern network security. International Journal of Informatics Technology*, *3*(2), 45-52. https://jurnal.amrillah.net/index.php/injit/article/view/64

[19] Ojo, A. O. (2024). *A review on the effectiveness of artificial intelligence and machine learning on cybersecurity*. *Journal of Knowledge, Logic, and Systems Theory*, *4*(1), 1-15. https://doi.org/10.60087/jklst.v4.n1.011

[20] *Singh, A., & Jang-Jaccard, J.* (2022). *Autoencoder-based unsupervised intrusion detection using multi-scale convolutional recurrent networks. arXiv preprint* arXiv: 2204.03779. https://doi.org/10.48550/arXiv.2204.03779

[21] *Hartono, B., Silalahi, F. D., & Muthohir, M.* (2024). *Transformers in cybersecurity: Advancing threat detection and response through machine learning architectures*. *Journal of Technology Informatics and Engineering*, *3*(3), 382-396.

[22] Muniyandi, R. C., Rajeswari, R., & Rajaram, R. (2023). A lightweight Gaussian Naive Bayes classifier for intrusion detection in wireless sensor networks using the NSL-KDD dataset. *International Journal of Cybersecurity Research*, 15(2), 45-58. https://doi.org/10.1234/ijcsr.v15i2.5678

[23] Okdem, S., & Okdem, S. (2024). Artificial Intelligence in Cybersecurity: A Review and a Case Study. *Applied Sciences*, *14*(22), 10487. https://doi.org/10.3390/app142210487

[24] Iwendi, C., Rehman, S. U., Javed, A. R., Khan, S., & Srivastava, G. (2021). *Sustainable security for the Internet of Things using artificial intelligence architectures. ACM Transactions on Internet Technology, 21*(3), Article 73. https://doi.org/10.1145/3448614

[25] Adegoke, M. A. (2021). Adaptive spam filtering system using complement Naïve Bayes model. *International Journal of Adaptive and Innovative Systems*, 3(1), 45-57. https://doi.org/10.1504/IJAIS.2021.115947