e-Prime - Advances in Electrical Engineering, Electronics and Energy 8 (2024) 100590

Contents lists available at ScienceDirect

e-Prime - Advances in Electrical Engineering, Electronics and Energy

journal homepage: www.elsevier.com/locate/prime

An empirical assessment of ML models for 5G network intrusion detection: A data leakage-free approach

Mohamed Aly Bouke *, Azizol Abdullah

Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia

ARTICLE INFO

Keywords:
Machine learning models
Network intrusion detection
Wireless networks
Security
Computational efficiency

ABSTRACT

This paper thoroughly compares thirteen unique Machine Learning (ML) models utilized for Intrusion detection systems (IDS) in a meticulously controlled environment. Unlike previous studies, we introduce a novel approach that meticulously avoids data leakage, enhancing the reliability of our findings. The study draws upon a comprehensively labeled 5G-NIDD dataset covering a broad spectrum of network behaviors, from benign real-user traffic to various attack scenarios. Our data preprocessing and experimental design have been carefully structured to eradicate any data leakage, a standout feature of our methodology that significantly improves the robustness and dependability of our results compared to prior studies. The ML models are evaluated using various performance metrics, including accuracy, precision, recall, F1-score, ROC AUC, and execution time. Our results reveal that the K-Nearest Neighbors model is superior in accuracy and ROC AUC, while the Voting Classifier stands out in precision and F1-score. Decision Tree, Bagging, and Extra Trees models exhibit strong recall scores. In contrast, the AdaBoost model falls short across all assessed metrics. Despite displaying only modest performance on other metrics, the Naive Bayes model excels in computational efficiency, offering the quickest execution time. This paper emphasizes the importance of understanding various ML models' distinct strengths, drawbacks, and trade-offs for network intrusion detection. It highlights that no single model is universally superior, and the choice hinges on the nature of the dataset, specific application requirements, and the computational resources available.

1. Introduction

The relentless pace of digital transformation and the pervasive Internet adoption has opened up a new epoch characterized by unprecedented global connectivity, expeditious information exchange, and an innovative approach to problem-solving. As digital technologies permeate all facets of modern society, they simultaneously create many opportunities that touch virtually every aspect of life [1–3]. However, this digital metamorphosis is not without its perils. It has also introduced an expanded landscape for nefarious activities, specifically network intrusions, thereby necessitating robust defense mechanisms to safeguard our networked systems' integrity, confidentiality, and in response to these evolving threats, Network Intrusion Detection (NIDS) have emerged as indispensable tool in the cyber-defense arsenal. These systems identify and alert security administrators to suspicious activities that could compromise the network's integrity [4–8].

One of the monumental milestones in digital communication is the advent of fifth-generation (5G) networks. Distinguished by their high-speed data transmission rates, minimal latency, and the ability to accommodate many simultaneous connections, 5G networks are the cornerstone of modern digital communication. They are paving the way for various innovative applications ranging from autonomous driving and Internet of Things (IoT) devices to virtual reality and telemedicine [9–11].

However, as the complexity and sophistication of 5G networks increase, so does their vulnerability to a broad spectrum of security threats. The intrinsic complexity of 5G architecture and its ability to provide diverse services exacerbates its susceptibility to various network intrusions. As such, the role of NIDS in identifying and mitigating these threats has become more imperative than ever before [12,13].

In these mounting challenges, ML emerges as a beacon of hope. ML algorithms, endowed with the capacity to learn from and make decisions based on data, hold great promise in augmenting the capabilities of NIDS. These algorithms can be trained on network traffic data to identify abnormal patterns that could signify potential network intrusions. However, it's important to note that the performance of these algorithms

* Corresponding author.
E-mail address: bouke@ieee.org (M.A. Bouke).

https://doi.org/10.1016/j.prime.2024.100590

Available online 9 May 2024

can vary significantly, depending on a range of factors. These include the nature of the application, the specific characteristics of the dataset, as well as the methodologies employed for data preprocessing and feature selection [14,15].

Beyond these considerations, another fundamental concern that is often overlooked in many research endeavors is the issue of data leakage during preprocessing. Data leakage is when information from outside the training dataset inadvertently influences the training process. This leakage can cause the model to perform exceptionally well on the training and testing datasets due to having access to information that should not have been available during the training phase. Despite generating high-performance metrics, this data leakage paints a misleading picture of the model's predictive power. The model's performance on unseen data is likely considerably less effective [16–19].

Regrettably, many existing studies have failed to take this issue seriously, compromising their results' reliability. Unlike these studies, our research places a high priority on preventing data leakage during the preprocessing phase. We employ rigorous measures to ensure that our models' performance metrics accurately reflect their predictive capabilities and are not distorted by data leakage.

In this paper, we present an empirical comparison of multiple ML algorithms for intrusion detection in 5G networks, contributing significantly to the emerging field of network security. Our comprehensive methodology spans data preprocessing, feature selection, and performance evaluation for thirteen distinct ML models. A notable feature of our methodology is the careful avoidance of data leakage during preprocessing, enhancing the reliability of our findings.

We utilized the 5G-NIDD dataset, a fully-labeled dataset that captures both malicious and benign network traffic scenarios. Using this dataset adds more realism to our study, which could greatly benefit academic researchers and industry practitioners.

Our study offers a detailed and transparent exposition of our methods, providing an in-depth analysis of the strengths and weaknesses of each model based on multiple performance metrics. Additionally, we present the trade-offs between computational efficiency and model performance, which is critical when choosing ML models for practical applications.

This paper's findings serve as an empirical reference for future research in the field of 5G network intrusion detection, and we hope our insights will guide industry practitioners in selecting suitable models for their unique needs. By comprehensively comparing ML models in a 5G network intrusion detection context, we aim to advance ML in network NIDS and contribute towards creating more secure digital communication systems.

The organization of this paper is as follows: in section 2, We present an overview of the existing research on intrusion detection using ML in 5G networks. Section 3 details our comprehensive methodology, which includes dataset collection, addressing data leakage during preprocessing, data preprocessing, feature selection, model building and testing, performance evaluation, and a description of our implementation environment. Section 4 presents and discusses our empirical comparison of the selected ML models. Our analysis includes examining each model's performance based on multiple metrics and their computational efficiency. Section 5 summarizes our findings, highlights the main contributions, and suggests future research directions.

2. Literature review

Network security and privacy, particularly in the rapidly evolving 5G technology context, stand as paramount concerns in today's digital landscape. As industries and sectors across the globe increasingly rely on interconnected networks, safeguarding data integrity, confidentiality, and accessibility becomes imperative. This literature review presents a meticulous examination of the multifaceted challenges and innovative solutions within the realm of network security with a focal point on the growing domain of 5G technology.

2.1. Security challenges in 5G networks

In an era of rapid technological evolution, the advent of 5G networks promises unprecedented connectivity and efficiency. However, this advancement comes hand in hand with intricate security challenges. Sicari et al. [20] explore the security and privacy challenges of 5G networks and investigate existing solutions. they discuss data integrity, confidentiality, authentication, access control, and intrusion detection requirements. The role of emerging paradigms like IoT, fog computing, and blockchain is also examined. The paper highlights the need for novel security and privacy solutions that ensure reliability and robustness in 5G systems. It emphasizes the importance of addressing open issues, achieving a unified vision, and considering specific architectural aspects.

Ahad et al. [21] highlight healthcare transformation towards a patient-centric model driven by technological advancements. It focuses on the role of communication technologies, including 4G and future 5G networks, in supporting smart healthcare applications. The paper presents a 5G-based smart healthcare architecture, explores key enabling technologies, and examines security and privacy threats in these networks.

Sotelo Monge et al. [22] introduce a new method for detecting Distributed Denial of Service (DDoS) attacks by analyzing traffic flows from protected network devices. The approach considers the non-stationarity and heterogeneity of the communication environment and utilizes the monitorization and knowledge acquisition capabilities of the SELFNET project. Preliminary results demonstrate the efficiency of the approach in distinguishing normal activities from DDoS behaviors using different metrics and adjustment parameters.

Kumar et al. [23] examine the integration of 5G technology into smart grid systems, highlighting its benefits and challenges. They emphasize the importance of secure telecommunications networks and information sharing for managing smart grids. The paper discusses the need for exceptional sensors, computation methods, and communication systems to enable real-time monitoring and administration.

Luglio et al. [24] introduce the Flexible Web Traffic Generator (FWTG), a tool designed to model the dynamics of web-based applications and user interactions. FWTG allows the generation of real-time HTTP traffic and its injection into real networks, enabling rapid configuration and evaluation of network slices in 5G Non-Public Networks (NPN). The tool supports network design by selecting suitable backhaul link capacity and aids in defining cutting-edge services by assessing traffic volume and dynamics. FWTG demonstrates scalability and flexibility in reproducing different traffic scenarios, making it valuable for optimizing 5G network configurations.

Muheidat et al. [25] discuss the advancements by 5G in mobile and wireless technologies, along with the need for improved security measures to address the associated risks. They highlight the potential benefits of 5G, such as faster speeds and advanced applications. Looking ahead to 6G, the role of quantum computing and networking is emphasized, particularly in terms of enhanced security.

Kasongo and Sun [26] introduce a wireless Intrusion Detection System (IDS) called WFEU-FFDNN, which combines a Wrapper Based Feature Extraction Unit (WFEU) with a Feed-Forward Deep Neural Network (FFDNN). The proposed approach achieves higher detection accuracy than traditional ML algorithms, as demonstrated using UNSW-NB15 and AWID intrusion detection datasets. The experiments highlight the effectiveness of the WFEU-FFDNN method, achieving overall accuracies of up to 99.77 % for binary and multiclass classifications.

Agrafiotis et al. [27] propose a novel approach to detect 5G malware traffic by leveraging a network packet preprocess toolkit and ML models. Their system transforms packets into images or embeddings, enabling more accurate representations applicable across protocols. Long Short-Term Memory Autoencoders generate embeddings, followed by a fully connected network for classification on a 5G-specific dataset.

Park et al. [28] introduce a distributed learning-based intrusion detection system tailored for the decentralized environment anticipated in 6G mobile networks. They leverage split learning, enabling efficient model training across systems with varying computing resources. Their approach addresses the limitations of centralized systems, making it suitable for distributed environments. Evaluations conducted on 5G network data illustrate the effectiveness of the proposed system in enhancing network security for future mobile generations.

Kim et al. [29] introduces a data-driven methodology for detecting location spoofing attacks and their variations in mobile communications. By incorporating a novel set of differential features, the system can assess mobility constraints and inconsistencies, enhancing detection performance significantly compared to previous methods. The approach considers various attack scenarios by manipulating coordinate data to create attack variants. Experimental results demonstrate the effectiveness of the new features in identifying diverse spoofing attacks, achieving up to 99.1 % accuracy. Furthermore, a profiling-based detection approach is presented, which relies solely on legitimate coordinate data, offering resilience to zero-day attacks with comparable or superior performance to supervised learning methods.

Ahmad et al. [30] addresses performance concerns in IDS by exploring the efficiency of ML techniques such as multilayer perceptron, support vector machine (SVM), and others. These techniques are critiqued for their limitations in handling large datasets efficiently. To tackle this issue, the study applies well-known classification methods, including SVM, random forest, and extreme learning machine (ELM), on the NSL-KDD dataset, a benchmark for intrusion detection evaluation. Experimental results demonstrate that ELM surpasses other methods, highlighting its effectiveness in enhancing intrusion detection performance.

To this end, a few themes become apparent. Firstly, the security challenges in 5G networks require innovative and holistic solutions considering diverse technologies and applications. Secondly, the pivotal role of 5G technology in diverse sectors like healthcare and smart grid systems is emphasized, underscoring the urgency of developing advanced security measures. Finally, the importance of accurate and reliable IDS, aided by ML and artificial intelligence, is stressed.

2.2. Intrusion detection methods

IDS are essential components of cybersecurity frameworks tasked with detecting unauthorized or anomalous activities that could compromise the security of networks and information systems. These systems deploy various methods, each suited to different threats and network environments, reflecting the evolving landscape of cybersecurity threats.

Anomaly-based detection [31–33] is a sophisticated method utilized in IDS that focuses on continuously monitoring network traffic and user behaviors. This method establishes a comprehensive baseline of the network's normal activity. This baseline is developed by aggregating and analysing extensive historical data on network operations and user behaviors. It provides a statistical or ML model with the parameters to recognize regular patterns and expected deviations.

Using these established norms, anomaly-based detection systems apply statistical models or advanced ML techniques to assess ongoing network activities continuously. An alert is generated whenever these systems detect an activity that significantly deviates from the established behavioral norms. This deviation could indicate various security threats, including unauthorized access attempts, malware infections, or other malicious activities. The strength of anomaly-based detection lies in its ability to identify potential threats that have not been previously documented or encountered, such as zero-day exploits and novel sophisticated attacks that do not match any known signatures [8–34].

However, while powerful in identifying new threats, anomaly-based detection systems have challenges. A primary issue with this method is its tendency to produce false positives. Since the system flags activities based purely on deviations from the norm, benign activities that are unusual but not harmful can also trigger alerts. For instance, a legitimate but uncommon use of network resources or atypical but authorized system configurations might be incorrectly perceived as intrusions. These false positives can lead to unnecessary disruptions and require additional resources to investigate and address, which can strain the security team's resources and potentially divert attention from genuine threats [35,36].

This susceptibility to false positives necessitates the implementation of robust validation processes to ensure that the alerts generated by anomaly-based detection systems are accurate. Enhancements in ML algorithms and incorporating feedback mechanisms to refine and adjust the baseline of normal activities can help reduce false positives, making anomaly-based detection more reliable and effective as part of a comprehensive cybersecurity strategy.

Signature-based detection [5,37–39] employs a database filled with known threat signatures, which are specific patterns linked to malicious activities. This method operates by scanning network data to search for these signatures, thus enabling the identification of known attacks. It's a well-established technique that is particularly robust against threats that have already been identified and analyzed, ensuring that recognized malware and exploits can be quickly and effectively blocked.

Despite its strengths, signature-based detection has significant limitations [40,41], primarily its inability to recognize new or modified threats that do not have signatures already in the database. This method is inherently reactive, not proactive, meaning it can only defend against new threats after they have been identified, analyzed, and added to the signature database. This time lag can be critical, as it provides a window during which systems are vulnerable to new attacks.

Moreover, to maintain its effectiveness, the signature database must be continually updated [40,42]. These updates are crucial as they ensure the detection system can respond to the latest malware and attack vectors. The frequency and quality of these updates depend on the organisation's resources to maintain the signatures, which can vary widely between providers. Without regular updates, the effectiveness of a signature-based IDS can degrade, leaving systems more vulnerable to newer forms of attacks.

Furthermore, the process of maintaining and updating signatures involves significant effort [43,44]. Security teams must constantly analyze the latest threats and develop accurate signatures that can detect them without generating false positives. This requires sophisticated understanding of malware behavior, attack methods, and continuous research and development.

ML-based detection in IDS utilizes the power of artificial intelligence to enhance security protocols by learning from data that encapsulates both normal and malicious activities. These systems, by training on a mix of benign and malicious behavior datasets, develop the capability to differentiate between legitimate operations and potential threats. Over time, the accuracy and effectiveness of these systems improve as they adapt to the evolving patterns of network users and attackers, which is a crucial advantage given the dynamic nature of cyber threats [37–39].

One significant benefit of ML-based IDS is their flexibility and adaptability, allowing them to keep pace with the changing tactics of cyber attackers. By continuously learning from new data, these systems can detect complex and sophisticated attack patterns that may elude traditional detection methods. This adaptability is vital for maintaining robust security in an environment where attackers continually refine their strategies to bypass defensive measures [45,46].

However, deploying ML in IDS also presents challenges, primarily related to the need for extensive and representative training data to achieve high detection performance. The systems must be trained on comprehensive datasets that accurately reflect the diverse range of normal and abnormal behaviors to minimize the risk of false positives and false negatives. Furthermore, the effectiveness of ML-based IDS can

be significantly influenced by the choice of algorithms and the quality of the dataset used for training [47,48].

Hybrid systems [7,49–51] in intrusion detection combine multiple detection techniques to enhance overall system robustness. By integrating both anomaly-based and signature-based detection, hybrid systems are capable of recognizing both known threats and unusual behaviors that may indicate new attacks. This dual approach leverages each method's strengths while mitigating weaknesses, such as the high false-positive rates often seen in anomaly-based systems and the inability of signature-based systems to detect new threats with no existing signatures.

Recent research highlights the effectiveness of hybrid systems [46,52,53] in providing a more comprehensive defense against a wider array of cyber threats. For example, hybrid systems that combine ML algorithms with traditional detection methods have shown improved accuracy and a reduced rate of false positives. Such systems are particularly effective in environments where adaptive response to evolving threats is critical.

Hybrid approaches also benefit from advancements in ML, which can dynamically adjust to new and emerging threats more efficiently than static, rule-based systems. The flexibility of ML models, when integrated with conventional detection methods, allows hybrid IDS to evolve with the changing behaviors of network users and attackers, thereby maintaining high detection rates over time [54–56].

2.3. Data leakage in ML-based IDS

ML has become a pivotal tool across various domains, from healthcare to cybersecurity, owing to its ability to discern intricate patterns within vast datasets. Yet, the efficacy of ML models relies heavily on the availability of sizable, accurately labeled training and testing data. A significant hurdle encountered in this realm is data leakage during the data preparation phase, where the training and testing datasets deviate from being independent and identically distributed, resulting in model overfitting. This phenomenon can lead to compromised performance when deployed in practical scenarios [10]. Numerous studies have been undertaken to scrutinize the ramifications of data leakage on the performance of ML models.

Dong [19] investigates data leakage, specifically in models using data from sports wearable sensors. He proposes a Bayesian inference method to predict leakage by assessing the reverse probability of unseen variables, effectively identifying potential leaks demonstrated through a dataset of sports wearable sensors.

Building on the understanding of data sensitivity, Farokhi and Kaafar [57] delve into how ML models are susceptible to membership inference attacks, where attackers deduce individual data points used in training. They use information-theoretic metrics to measure membership information leakage, showing that the leakage reduces with larger datasets and stronger regularization, thereby enhancing model security.

Further exploring the security aspects, Zhang et al. [17] address dataset property leakage in secure multi-party ML environments. They reveal that even with limited black-box access to the model, a curious party can infer sensitive data attributes from other parties' datasets, thus challenging the confidentiality of individual records and entire dataset properties.

Hannun et al. [58] presented a new methodology to quantify information leakage in ML models using Fisher information. This approach evaluates leakage on specific examples, attributes, or sub-populations within a dataset, offering a more nuanced analysis than the worst-case differential privacy assessments. The proposed metric, Fisher information loss, gauges the information leaked by a model about its training data. This holds particular significance in handling sensitive data, where membership in the training set or extraction of sensitive attributes can be inferred. Unlike differential privacy, Fisher information loss remains sensitive to dataset correlations without degradation, promising to safeguard individual and subgroup privacy..

Kapoor and Narayanan [59] critically examine the pervasive issue of data leakage in ML-driven scientific research, highlighting its repercussions on reproducibility and reliability across 17 scientific fields and 294 papers. They delineate eight specific types of leakage, ranging from common data handling errors to complex issues stemming from mismatches between training and test data distributions. To counter these challenges, the authors advocate for adopting model info sheets, a tool aimed at aiding researchers in identifying and preventing data leakage by prompting meticulous scrutiny of their model development processes.

In their review paper, Bouke et al. [60] explore the challenges of Data Lack, Leakage, and Dimensionality (DLLD) in the context of ML-based IDS. They provide an in-depth discussion on the issue of data leakage, identifying it as a critical challenge where information from outside the training dataset inadvertently influences the model during data preprocessing. This results in overestimated model performance and poor generalization to new, unseen data. The review highlights the susceptibility of certain algorithms, like Decision Trees and Gradient Boosting, to data leakage. It underscores the importance of rigorous data preprocessing and careful model selection to mitigate this issue.

Subotić et al. [61] introduce NBLyzer, a static analysis framework tailored for evaluating data science notebooks, addressing the unique execution semantics inherent to notebooks. This framework facilitates a broad spectrum of analyses applicable to various use cases within notebooks. Leveraging abstract interpretation theory for intra-cell static analyses, NBLyzer ensures cell execution termination by over-approximating the computational state. This results in a practical tool for preemptively identifying potential issues in notebooks. Through implementation across a diverse set of analyses tested on 2211 real-world notebooks, the authors demonstrate NBLyzer's utility, with the vast majority of notebooks (98.7 %) analyzed in less than a second, meeting the interactive requirements of notebook clients.

Maxwell et al. [62] delineate challenges and propose solutions to enhance reproducibility and replicability in remote sensing deep learning (DL) research. They focus on addressing convolutional neural network (CNN)-based DL challenges in remote sensing, particularly in semantic segmentation, object detection, and instance segmentation. The paper underscores the complexity and customization of CNN architectures, variable model training, and assessment practices alongside issues like data leakage and computational demands. To tackle these challenges, the authors advocate for best practices, including comprehensive documentation of data and preprocessing steps, public availability of code and models, and clear reporting of algorithmic details and training processes.

Kovács and Fazekas [63] introduce "mlscorecheck," an innovative open-source package designed to address the reproducibility crisis in artificial intelligence by validating the consistency of reported experimental results in ML. This package utilizes numerical techniques to identify inconsistencies between reported performance scores and experimental setups across various ML tasks, encompassing binary/multiclass classification and regression. Leveraging mathematical interrelations between performance scores, "mlscorecheck" rigorously assesses whether reported scores genuinely arise from the described experiments.

Lastly, Bouke and Abdullah [16] investigate the impact of pattern leakage during data preprocessing on ML-based IDS. They clarify that pattern leakage, where training information inadvertently enters the testing set, leads to overfitting and inaccurately high-performance metrics. Employing three standard intrusion detection datasets (NSL-KDD, UNSW-NB15, and KDDCUP99), they process data to introduce and prevent leakage, comparing performance across six ML models. Results demonstrate varying algorithm sensitivity to leakage, with significant drops in accuracy when leakage is removed. Emphasizing the importance of meticulous data handling during preprocessing to ensure reliable and robust IDS models, they provide specific recommendations for minimizing data leakage and enhancing model

reliability.

In summary, this literature review offers crucial insights into the evolving landscape of network security, focusing mainly on the challenges and advancements within 5G technology. The study emphasizes the critical need for rigorous evaluation of ML algorithms in the context of 5G IDS. Adhering to a data leakage-free approach, our research adds a layer of trust. It enhances the validity of the findings, reinforcing the deployment of these advanced security measures in 5G infrastructures.

Moreover, this research stands out by meticulously addressing data leakage prevention during preprocessing, a crucial but often overlooked area in prior studies. This focus ensures the robustness and reliability of the ML models evaluated for intrusion detection, which is crucial for enhancing security frameworks to combat the rapidly evolving threats in 5G networks.

3. Materials and methods

The primary objective of this study was to analyze the performance of various ML models for intrusion detection using the 5G-NIDD [64] dataset. In this context, intrusion detection is crucial in maintaining the security of 5G wireless networks against various types of attacks.

We specifically chose to evaluate 13 different ML algorithms, representing a broad spectrum of methodological paradigms, including Logistic Regression, Random Forest, Support Vector Machines, Gradient Boosting, Neural Networks, K-Nearest Neighbors, Decision Tree, Naive Bayes, Linear Discriminant Analysis, AdaBoost, Bagging, Extra Trees, and a Voting Classifier.

The rationale behind selecting these algorithms is manifold:

- ***Diversity in Approaches:*** The selected algorithms encompass different methodologies, from simple linear models (like Logistic Regression and Linear Discriminant Analysis) to complex ensemble models (like Random Forest and Gradient Boosting). This variety ensures a comprehensive assessment of different types of models under the same conditions.
- ***Robustness and Flexibility:*** The chosen models possess varying degrees of robustness and flexibility, which allows them to adapt to different patterns and complexities in the data.
- ***Real-world Applicability:*** These models are commonly used in real-world applications, thus making our research more practical and relatable to current trends in intrusion detection.
- ***Ensemble Voting:*** Including a Voting Classifier ensures we harness the collective power of multiple learning algorithms, potentially improving the performance by combining their strengths.

To robustly evaluate these models, we developed a comprehensive methodology, detailed in the following sections, involving data preprocessing, feature selection, and model evaluation. Fig. 1 illustrates our experimental methodology flowchart.

3.1. Dataset collection

The dataset used in this study, named 5G-NIDD [64], is a fully labeled dataset built on a functional 5G test network. It captures network traffic in various attack scenarios, including but not limited to Port Scans and Denial-of-Service (DoS) attacks. The dataset also includes non-malicious traffic, providing a more representation of network behavior. Each flow in the dataset is labeled as either attack or normal and further includes labels indicating the attack type and the attacking tool used, enabling multiclass classification.

The 5G-NIDD dataset used in this study consists of a total of 85,112 samples, with 51,682 samples belonging to the "Normal" class (0) and 33,430 samples belonging to the "Attack" class (1). This distribution shows that the dataset is slightly imbalanced, with the "Normal" class accounting for approximately 60.72 % of the samples and the "Attack" class accounting for approximately 39.28 % of the samples (Fig. 2).

This work focuses on binary classification, specifically distinguishing between the "Normal" and "Attack" classes. The dataset's binary nature allows for the evaluation and development of algorithms and models tailored explicitly for binary classification tasks in the context of network security.

3.2. Avoiding data leakage during preprocessing

As a crucial part of our methodology, we have prioritized the

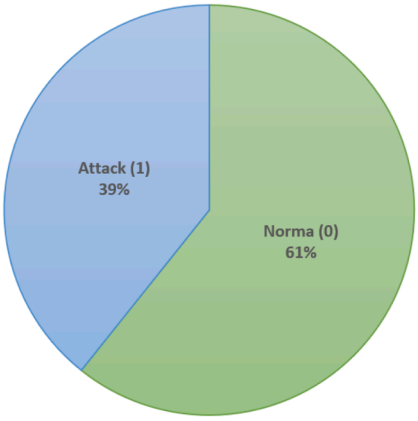

Fig. 2. Dataset class distribution.

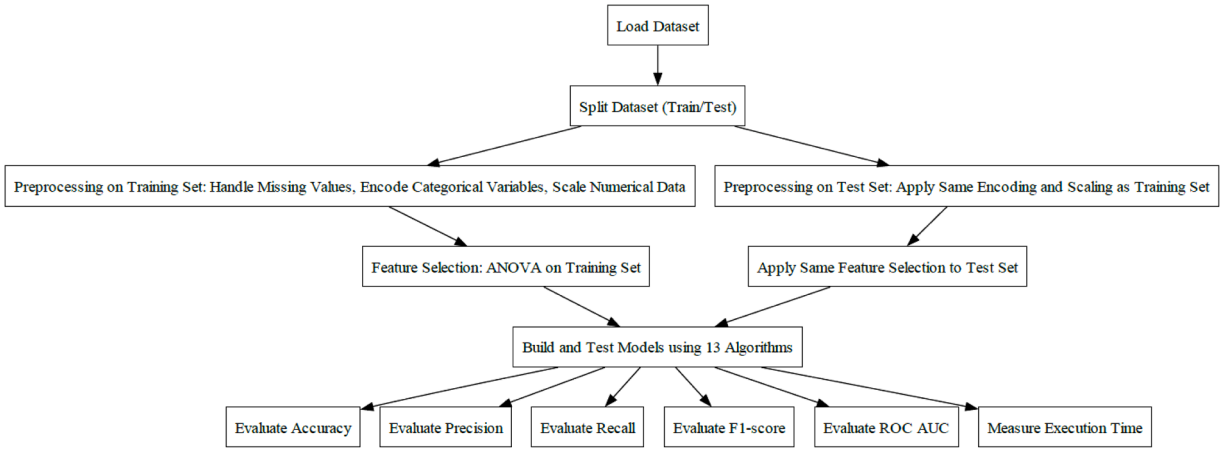

Fig. 1. Methodology flowchart.

prevention of data leakage during preprocessing. This issue is critical yet frequently overlooked in ML [65]. Data leakage occurs when information from the test dataset inadvertently influences the training process. This often results in models demonstrating high accuracy during testing but performing poorly in real-world scenarios because they fail to generalize to new data [66].

Moreover, Consider the following scenario, which illustrates the potential for data leakage in network IDS: Imagine we are using a dataset of network traffic logs to build an IDS. The objective is to predict and flag potential intrusions by analyzing patterns in the traffic. During preprocessing, suppose the total traffic volume—which includes training and test data—contributes to the normalization of individual features, such as the number of requests per second or unusual access patterns from an IP address. If the normalization process uses the maximum traffic volume, a feature potentially influenced by an intrusion event in the test set, to scale the entire dataset, the training model might inadvertently learn details about intrusion events that should be unknown until the testing phase. This exposure could result in the model displaying impressively high accuracy during validation because it has already 'seen' similar data during training.

To combat this, we ensure that all preprocessing steps, such as normalization and feature scaling, are conducted independently on the training and test datasets. This approach includes fitting scalers only on the training data and applying them to the test data without adjustment. By strictly separating the datasets from the outset and maintaining rigorous control over preprocessing influences, we safeguard against the possibility of the test data leaking into the training process.

To avoid data leakage, specific steps have been incorporated into our pipeline:

- ***Splitting Dataset before Preprocessing:*** The dataset is initially split into training and test sets before any preprocessing. By dividing the data beforehand, we ensure that the test set acts as truly unseen data, without any influence from the data preprocessing steps, therefore providing a realistic evaluation of the model's performance.
- ***Independent Preprocessing:*** Following the split, preprocessing steps such as encoding, scaling, and feature selection are applied independently to the training and test sets. This approach guarantees that the transformations learned from the training data do not take any information from the test data.
- Specifically, the encoders and scalers are fitted only on the training data when encoding categorical variables and scaling numerical variables. Afterwards, these fitted transformers are used to transform the test data. This process ensures that the transformation parameters (like minimum, maximum, and mean for scaling) are not influenced by the information in the test set.
- ***Feature Selection on Training Data Only:*** The feature selection process is applied solely to the training set. The selected features' indices are then used to extract the corresponding features from the test set. This approach safeguards against data leakage by ensuring that the test set does not influence the decision about which features are most informative.

Through this strict separation of training and test sets during all preprocessing steps, our methodology provides an honest and unbiased evaluation of the ML models for intrusion detection in the 5G-NIDD dataset. This approach also ensures that our models will be more likely to perform well on entirely new, unseen data, thereby increasing the generalizability and practicality of our findings.

3.3. Data preprocessing

The dataset was initially loaded from a CSV file. Subsequently, we split the dataset into training and testing sets using a standard 80/20 ratio to avoid data leakage. It's worth noting that the data splitting happened before any preprocessing or feature selection to ensure that no information from the testing set leaked into the training process.

For the training set, we first handled missing values by replacing them with the mean value of the respective feature. Next, any categorical variables were encoded using label encoding to transform them into numerical values that the ML models could handle. Finally, the data were scaled using Min-Max scaling to bring all features to the same range of [0, 1].

3.4. Feature selection

The following step in our methodology is feature selection, an integral part of any ML workflow that aims to select the most important features from the dataset, reducing dimensionality and potentially improving model performance [67].

We opted for ANOVA (Analysis of Variance) as our feature selection method for the task. The reasons behind choosing ANOVA include the following [68,69]:

- ***Sensitivity to Linear Relationships:*** ANOVA is particularly effective in detecting important features when there are linear relationships between independent variables and the dependent variable. Given that several of our selected models (e.g., Logistic Regression, Linear Discriminant Analysis) are based on linear assumptions, using a feature selection method that aligns with this was prudent.
- ***Scalability:*** ANOVA is a scalable method capable of handling datasets with many features, which suits our case, given that the 5G-NIDD dataset contains many features due to the comprehensive nature of the network traffic data.
- ***Transparency and Interpretability:*** ANOVA provides a clear statistical framework for understanding why a feature might be necessary, offering more transparency than other feature selection methods.

In our experiment, the parameter 'k,' which designates the number of top features selected, was meticulously determined to be 8. This decision resulted from an exhaustive series of experiments to find a balance between model complexity and the risk of overfitting. These trials established that selecting eight features yielded minimal feature space while maximizing the average accuracy across all 13 algorithms under investigation.

The feature selection process was diligently conducted exclusively on the training set to staunchly prevent data leakage and ensure the independence of the test set. After identifying the top 'k' features within the training set, these features were extracted from the test set, forming the basis for the ensuing model evaluation phase.

3.5. Model building and testing

Our investigation examined the performance of a diverse array of thirteen ML algorithms. This broad selection was made to ensure a comprehensive analysis and to allow the comparison of different types of models, including both linear and nonlinear approaches, as well as ensemble methods. Here's a brief explanation of why each model was chosen:

- ***Logistic Regression:*** A simple yet powerful linear model for binary classification problems. Chosen for its interpretability and efficiency.
- ***Random Forest:*** An ensemble method that creates many decision trees and combines their predictions. Chosen for its robustness to outliers and overfitting.
- ***Support Vector Machines (SVM):*** A powerful classifier that can handle both linear and nonlinear data. Chosen for its effectiveness in high-dimensional spaces.
- ***Gradient Boosting:*** A boosting ensemble method that iteratively adds weak learners to improve the model. Chosen for its high performance in various tasks.

- ***Neural Networks:*** These models are capable of capturing complex patterns and interactions. Chosen for their potential to handle intricate data structures.
- ***K-Nearest Neighbors:*** A simple algorithm that assigns a class to a test instance based on the majority class of its 'k' nearest neighbors. Chosen for its effectiveness in situations where decision boundaries are very irregular.
- ***Decision Tree:*** An interpretable model that splits the data based on feature values. Chosen for its simplicity and ease of interpretation.
- ***Naive Bayes:*** A probabilistic classifier based on Bayes' theorem. Chosen for its efficiency and effectiveness, especially concerning text data.
- ***Linear Discriminant Analysis (LDA):*** A statistical binary and multiclass classification method. Chosen for its ability to reduce dimensionality while preserving class separability.
- ***AdaBoost:*** A boosting ensemble that adjusts instances' weights in response to the previous classification. Chosen for its ability to enhance weak learners.
- **Bagging:** An ensemble method that creates multiple subsets of the original dataset and trains a model on each. Chosen for its ability to reduce the variance of a model.
- ***Extra Trees:*** An ensemble method similar to Random Forest but with increased randomness. Chosen for its robustness and speed.
- ***Voting Classifier:*** A meta-estimator that fits several classifiers and averages or votes for the best prediction. Chosen to combine the strengths of various models.

Each of these models was trained on the preprocessed and feature-selected training set. To ensure fairness and prevent bias towards any algorithm, all models were kept at their default parameter settings. This decision to avoid tuning parameters and leave them at their original settings allows for a genuinely unbiased comparative study, demonstrating how each algorithm performs under the same conditions and is given the same task.

Post-training, the models were evaluated on the similarly preprocessed and feature-selected test set. This rigorous approach ensures that the performance of each model was assessed on unseen data, closely emulating a real-world scenario where the actual labels of the data are unknown at prediction time.

3.6. Performance evaluation

The assessment of model performance is an integral part of the ML pipeline. This process involves the utilization of several metrics, which can provide valuable insights into the strengths and weaknesses of a model and inform us on how to improve our predictions. Our study selected five common metrics for model evaluation: Accuracy, Precision, Recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (ROC AUC). Additionally, we measured the execution time for each model to assess its computational efficiency. The choice of these metrics was driven by their widespread use and ability to provide a holistic understanding of model performance.

Below is a detailed description of each metric:

- ***Accuracy (1):*** This metric gauges the ratio of correct predictions (true positives and negatives) to the total number of predictions. Accuracy is a straightforward measure that works well when the classes are balanced but can be misleading when the class distribution is skewed.
- ***Recall (Sensitivity) (2):*** Recall measures the proportion of actual positive cases (in this case, attacks) correctly identified. It is instrumental when the cost of false negatives is high.
- ***Precision (3):*** Precision is the proportion of correct identifications. It is crucial when the cost of false positives is high.
- ***F1-score (4):*** The F1-score is the harmonic mean of precision and recall. It balances these two metrics and is particularly useful when dealing with uneven class distributions.
- ***ROC AUC (5):*** This metric evaluates the trade-off between true positive and false positive rates at different threshold levels. It provides an aggregate performance measure across all possible classification thresholds and is beneficial when dealing with imbalanced datasets.
- ***Execution Time (6):*** This metric is the duration the ML algorithm takes to learn from the data, i.e., from the training process to the end. It is measured in seconds. Execution time is a crucial factor to consider, especially in real-time applications where prediction speed is paramount.

These metrics are calculated as follows:

$$Accuracy = ((TP+TN))/((TP+TN+FP+FN)) \tag{1}$$

$$Recall\ (Sensitivity) = TP/((TP+TN)) \tag{2}$$

$$Precision = TP/((TP+FP)) \tag{3}$$

$$Fscore = 2 \times (Precision \times Recall)/(Precision + Recall) \tag{4}$$

$$execution\_time = end\_time - start\_time \tag{5}$$

where:

- TP (True Positive): Number of instances correctly predicted as the positive class.
- TN (True Negative): Number of instances correctly predicted as the negative class.
- FP (False Positive): Number of instances falsely predicted as the positive class (Type I error).
- FN (False Negative): Number of instances falsely predicted as the negative class (Type II error).
- The Python ***time*** library is used to calculate the execution time. At the beginning of model training, the current time is noted using ***start_time*** = ***time.time()***. Once the model finishes training, the time is noted again as ***end_time*** = ***time.time()***. The difference between these two-time points provides the execution time: ***execution_time*** = ***end_time - start_time***.

Furthermore, to tackle the possible influence of skewed datasets on the performance assessment, we integrated the following parameters:

- ROC AUC (6): This metric gauges the efficacy of a classification model at different threshold settings by computing the area under the receiver operating characteristic (ROC) curve. It helps assess the balance between TP and FP rates.

ROC AUC is computed by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, followed by determining the area beneath the curve. The TPR and FPR are derived as follows:

$$TPR = TP/((TP+FN))\ and\ FPR = FP/((FP+TN)) \tag{6}$$

In this context, the positive class refers to the attack instances, and the negative class refers to normal activities.

Utilizing these metrics, we aim to assess our model's performance comprehensively. This multifaceted approach ensures that our evaluation is not biased towards any particular performance aspect and provides us with a holistic understanding of our model's effectiveness.

3.7. Implementation environment

Our study was executed in a high-powered computational

environment to accommodate the demanding operations. We relied on a personal computer equipped with a 10th-generation Intel i7 processor, 32GB of DDR4 RAM, and solid-state drive (SSD) storage to expedite data access and processing. The machine operated on Windows 10 Pro to ensure a smooth and stable experience.

Our methodology was implemented using Python 3.10. Python's popularity stems from its user-friendly nature and comprehensive support for scientific computing libraries, which are crucial to our research. Essential libraries employed in this study encompass pandas for data wrangling, NumPy for numerical computations, sci-kit-learn for ML functions, imbalanced-learn to handle a class imbalance in our dataset, and matplotlib and seaborn for data representation and visualization.

We utilized Anaconda to manage our programming environment and preserve the replicability of our experiments. This tool streamlines package administration and deployment, enabling easy management of distinct environments without causing conflicts.

Our chosen platform for development was Jupyter Notebooks. This open-source web application facilitates the creation and sharing of documents with live code, visualizations, mathematical equations, and explanatory text. The final implementation and deployment were written as Python scripts (.py files).

A summary of the implementation environment is provided in the following table (Table 1):

4. Results and discussions

This section presents the findings of our study, where various machine learning models were rigorously tested for their effectiveness in detecting network intrusions within a 5G environment. The results are analyzed to assess each model's performance across multiple metrics, including accuracy, precision, recall, F1 score, and execution time.

Table 2 provides a detailed overview and analysis of the results obtained from our experiment, examining the performance of various ML models on multiple metrics, including accuracy, precision, recall, F1-score, ROC AUC, and execution time.

Furthermore, a deep dive into these metrics was conducted, and each model's performance was studied concerning these criteria, shedding light on each model's strengths and weaknesses in the context of the given dataset. The primary outcome of this study is the importance of carefully selecting the model based on the computational resources available and the application's specific requirements.

Accuracy is a fundamental metric in ML, reflecting the proportion of instances correctly predicted by the model. In the scope of our investigation, the K-Nearest Neighbors model surfaced as the leader in this regard, achieving an impressive accuracy of 99.6 %, as depicted in Fig. 3. This high level of accuracy signifies the K-Nearest Neighbours algorithm's robustness, particularly in our dataset's context, underlining its potential in correctly distinguishing between classes.

Additional models demonstrating commendable accuracies include the Voting Classifier and Neural Networks models, with 98.8 % and 98 %, respectively, further reinforcing their dependable prediction capabilities.

Contrastingly, the AdaBoost model depicted a markedly lower accuracy of only 55.4 %, suggesting significant struggles in accurately classifying instances within the provided dataset. Such a level of accuracy insinuates that the AdaBoost model, while effective in some scenarios, might not serve as an optimal choice for this specific dataset due to its frequency in erroneous classifications.

Table 1
Summary of implementation environment.

Component	Specification
Operating System	Windows 10 Pro
Hardware	Intel i7 10th Gen, 32GB DDR4 RAM, SSD Storage
Programming Language	Python 3.10
Key Libraries	Pandas, NumPy, Scikit-learn, Imbalanced-learn, Matplotlib, Seaborn
Environment Management	Anaconda
Development Tool	Jupyter Notebooks

Precision is another critical metric which denotes the proportion of correct identifications. In our experiment, the Voting Classifier model scored almost flawlessly on this metric, achieving a precision of 99.9 %. This indicates the model's aptitude to classify positive instances, minimizing false positive errors correctly. Such a feat is worth noting as it emphasizes the effectiveness of this model in correctly identifying positive cases while maintaining a low rate of false positives.

Likewise, the Neural Networks and Logistic Regression models depicted high precisions of 99.8 % and 99.5 %, respectively, further underlining their robustness in correctly identifying positive cases and implying their potential effectiveness in scenarios where a low false positive rate is crucial.

In stark contrast to the above models, the AdaBoost model achieved a precision of only 41.5 %, hinting at a substantial rate of false positives. The AdaBoost model may not be the optimal choice in practical scenarios, especially those with a high cost associated with false positives.

Recall, another key performance metric, refers to the ratio of true positives to the sum of true positives and false negatives. This metric indicates a model's ability to identify actual positive instances correctly. In this experiment, both Decision Tree and Bagging models achieved an astounding recall rate of 99.9 %. This near-perfect recall rate underscores the effectiveness of these models in identifying positive cases, hence considerably reducing false negatives. The Extra Trees model also demonstrated a high recall rate of 99.7 %.

Conversely, the AdaBoost model again underperformed, achieving a mere 33.5 % recall rate, suggesting a high frequency of false negatives and an inability to identify positive instances proficiently.

The F1-score represents the harmonic mean of precision and recall, providing a balanced measure of a model's performance when both false positives and false negatives are equally important. In our study, the Voting Classifier model outperformed all others, achieving the highest F1-score of 98.4 %. This high score signifies a balanced, top-tier performance in precision and recall.

On the other hand, the AdaBoost model had an F1-score of just 37.1 %, the lowest among all the models tested. This low score hints at a significant imbalance and a general lack of performance in both precision and recall.

ROC AUC (Area Under the Receiver Operating Characteristic curve) comprehensively evaluates a model's performance across all possible classification thresholds, thus balancing sensitivity and specificity. In our experiment, the K-Nearest Neighbors model achieved the highest ROC AUC score of 99.6 %, suggesting an excellent balance between sensitivity (true positive rate) and specificity (true negative rate). Conversely, the AdaBoost model again fell short in this metric, scoring only 51.5 %, indicative of an imbalanced true positive rate and false positive rate across various thresholds.

The time taken for a model to execute is crucial, especially in scenarios where real-time prediction is required or when dealing with extensive datasets. Per our findings, the Naive Bayes model delivered the quickest execution time of only 0.014 s, suggesting high computational efficiency, as illustrated in Fig. 4.

In stark contrast, despite excelling in other metrics, the Voting Classifier model took the longest execution time of 58.023 s. This high computational demand underscores that ensemble methods, such as the Voting Classifier, can deliver superior results. However, they may not be ideal when computational resources are limited or quick results are paramount.

While the K-Nearest Neighbors model excelled in accuracy and ROC AUC, its relatively high execution time may limit its applicability in real-time scenarios or when computational resources are constrained.

Despite its subpar performance in all metrics, the AdaBoost model demonstrated a relatively moderate execution time. This indicates the

Table 2
Performance metrics and execution times results for all models.

Model	Accuracy	Precision	Recall	F1-score	ROC AUC	Execution Time (seconds)
Naive Bayes	94.2	94.8	90	92.4	93.4	0.014
Linear Discriminant Analysis	95.4	99.3	88.9	93.8	94.2	0.049
Decision Tree	81.6	68.1	99.9	81	84.9	0.101
K-Nearest Neighbors	99.6	99.7	99.3	99.5	99.6	0.17
Logistic Regression	96	99.5	90.3	94.7	95	0.193
Bagging	81.6	68.1	99.9	81	84.9	0.732
Extra Trees	91	81.5	99.7	89.7	92.5	1.101
AdaBoost	55.4	41.5	33.5	37.1	51.5	1.815
Random Forest	98.3	96	100	97.9	98.6	3.387
Gradient Boosting	85.3	72.8	99.9	84.2	87.9	7.233
Support Vector Machines	96.8	99.7	92.2	95.8	96	15.183
Neural Networks	98	99.8	95	97.4	97.4	37.569
Voting Classifier	98.8	99.9	97	98.4	98.5	58.023

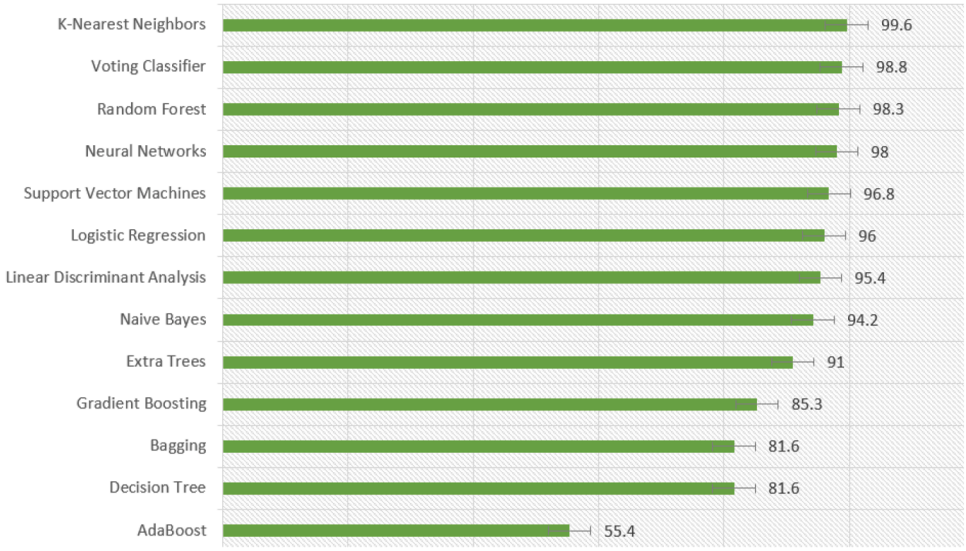

Fig. 3. Accucry comparaison.

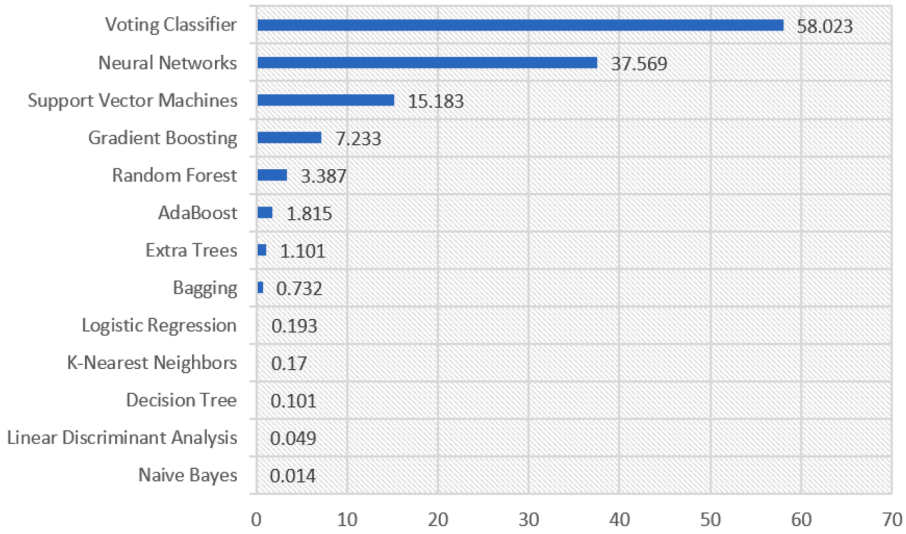

Fig. 4. Comparison of execution time (seconds).

existence of a trade-off between performance and computational efficiency, the balance of which must be considered based on specific application requirements.

Moreover, selecting the most suitable ML model for network intrusion detection in 5G networks requires a nuanced analysis of various critical factors, including model accuracy, computational efficiency, and ease of deployment. These elements are vital in high-stakes environments such as network security, where the right balance can

significantly impact the effectiveness and reliability of the detection systems.

Model accuracy is paramount for ensuring the detection system can correctly identify and mitigate threats without missing potential attacks. Typically, models like KNN and SVM achieve high accuracy but at the cost of increased computational resources. These models demand significant processing power and memory, which can be challenging to sustain in real-time processing environments.

Conversely, simpler models like Naive Bayes or Logistic Regression, while not providing the same level of accuracy, offer greater computational efficiency. These models are particularly advantageous in environments with limited resources or where a rapid response is essential. The choice between high accuracy and computational efficiency often hinges on the specific security requirements of the network and the resources available. For example, in situations where response speed is critical and a slight compromise on accuracy is permissible, a model like Naive Bayes might be the preferred choice. On the other hand, in settings where accuracy is crucial, such as in financial institutions or government networks, a more resource-intensive but accurate model like SVM may be necessary.

The ease of deployment is another crucial consideration that includes the complexity of model training and tuning, the necessity for ongoing maintenance, and the model's adaptability to changes in network behavior. Complex models like Neural Networks require extensive training, fine-tuning, and regular updates to remain effective, demanding substantial resources and specialized knowledge. In contrast, simpler models are easier to deploy and maintain but may struggle to adapt to evolving attack patterns, potentially reducing their effectiveness over time unless regularly updated.

Security administrators face the challenge of balancing these trade-offs to align with the organization's security policies, available resources, and risk management strategies. They must evaluate the infrastructure's capacity to support complex models without compromising other operations, assess the required level of security against potential threats, and consider the operational impacts of false positives and false negatives.

These results underscore the significance of understanding the trade-off between performance and computational efficiency when selecting an ML model. Factors such as the nature of the dataset, specific requirements of the application, and available computational resources must all be considered when choosing an appropriate model. As our findings suggest, no single model outperforms in all scenarios - each model has its unique strengths and weaknesses, and its performance can vary greatly depending on the context and the data to which it is applied. Consequently, considering all these factors, a careful and informed selection of the model is crucial in ML implementations.

5. Conclusion

This study rigorously evaluated various ML models for network intrusion detection within a meticulously controlled testing environment explicitly designed to prevent data leakage during preprocessing. This ensures the reliability of our results. The experiments utilized the 5G-NIDD dataset generated in a controlled environment using a functional 5G test network. This fully labeled dataset captures network traffic under various simulated attack scenarios, including but not limited to Port Scans and Denial-of-Service (DoS) attacks, alongside benign traffic from real users to provide a comprehensive and realistic representation of network behavior.

The ML models were assessed using key performance metrics such as accuracy, precision, recall, F1-score, ROC AUC, and execution time. Findings highlighted varied performances across the models, with the K-Nearest Neighbors, Voting Classifier, and Neural Networks models showing exemplary performance on several metrics. However, no single model excelled uniformly across all metrics. This highlights the complex nature of ML algorithms, where performance is contingent on the specific dataset and context of the application.

Moreover, the study discussed computational efficiency, noting that while the Naive Bayes model was the quickest, it did not perform as well on other metrics. Conversely, the AdaBoost model showed lower performance across all metrics but required moderate execution time, illustrating the trade-offs between performance and computational efficiency.

These insights are crucial for ML practitioners, particularly in the realm of network intrusion detection. They underscore the necessity of selecting the right ML model based on specific requirements, the nature of the dataset, and available computational resources.

Despite these promising results, a significant limitation of our study is the reliance on a controlled dataset. While the 5G-NIDD provides a robust framework for initial testing, future research should incorporate real-world datasets to validate the models under more varied and unpredictable conditions. Such datasets would help in better understanding the practical implications and effectiveness of these models in real operational environments.

Future research could focus on enhancing model performance through advanced optimization techniques such as parameter tuning and feature engineering or exploring other ML models not covered in this study. Additionally, embracing more realistic and robust datasets like 5G-NIDD in the research community is essential for developing more accurate and efficient network IDS.

CRediT authorship contribution statement

Mohamed Aly Bouke: Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Azizol Abdullah:** Writing – review & editing, Resources, Project administration, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data used is cited in the paper and is publicly available and free for access.

Funding statement

Not applicable.

Additional information

No additional information is available for this paper.

References

[1] A. Nisioti, A. Mylonas, P.D. Yoo, V. Katos, From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods, IEEE Commun. Surv. Tutorials 20 (4) (2018) 3369–3388, https://doi.org/10.1109/COMST.2018.2854724.

[2] R. Chapaneri, S. Shah, Enhanced detection of imbalanced malicious network traffic with regularized Generative Adversarial Networks, J. Netw. Comput. Appl. 202 (August 2021) (2022) 103368, https://doi.org/10.1016/j.jnca.2022.103368.

[3] M.A. Bouke, A. Abdullah, SMRD: a novel cyber warfare modeling framework for social engineering, malware, ransomware, and distributed denial-of-service based on a system of nonlinear differential equations, J. Appl. Artif. Intell. 5 (1) (2024) 54–68, https://doi.org/10.48185/jaai.v5i1.972.

[4] S. Meftah, T. Rachidi, N. Assem, Network based intrusion detection using the UNSW-NB15 dataset, Int. J. Comput. Digit. Syst. 8 (5) (2019) 477–487, https://doi.org/10.12785/ijcds/080505.

[5] N. Moustafa, J. Slay, The evaluation of Network Anomaly Detection Systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, Inf. Secur. J. 25 (1–3) (2016) 18–31, https://doi.org/10.1080/19393555.2015.1125974.

[6] S.-W. Lee, et al., Towards secure intrusion detection systems using deep learning techniques: comprehensive analysis and review, J. Netw. Comput. Appl. 187 (December 2020) (2021) 103111, https://doi.org/10.1016/j.jnca.2021.103111.

[7] A. Maheshwari, B. Mehraj, M.S. Khan, M.S. Idrisi, An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment, Microprocess. Microsyst. 89 (December 2021) (2022) 104412, https://doi.org/10.1016/j.micpro.2021.104412.

[8] N. Satheesh, et al., Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network, Microprocess. Microsyst. 79 (September) (2020) 103285, https://doi.org/10.1016/j.micpro.2020.103285.

[9] I. Soto, M. Calderon, O. Amador, M. Urueña, A survey on road safety and traffic efficiency vehicular applications based on C-V2X technologies, Veh. Commun. 33 (2022) 100428, https://doi.org/10.1016/j.vehcom.2021.100428.

[10] K.M. Sadique, R. Rahmani, P. Johannesson, Towards security on internet of things: applications and challenges in technology, Procedia Comput. Sci. 141 (2018) 199–206, https://doi.org/10.1016/j.procs.2018.10.168.

[11] K.P.S. Kumar, S.A.H. Nair, D. Guha Roy, B. Rajalingam, R.S. Kumar, Security and privacy-aware Artificial Intrusion Detection System using Federated Machine Learning, Comput. Electr. Eng. 96 (PA,) (2021) 107440, https://doi.org/10.1016/j.compeleceng.2021.107440.

[12] D. Wang, B. Song, D. Chen, X. Du, Intelligent cognitive radio in 5G: aI-based hierarchical cognitive cellular networks, IEEE Wirel. Commun. 26 (3) (2019) 54–61.

[13] C. Ssengonzi, O.P. Kogeda, T.O. Olwal, A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization, Array 14 (January) (2022) 100142, https://doi.org/10.1016/j.array.2022.100142.

[14] E.M. Onyema, S. Dalal, C.A.T. Romero, B. Seth, P. Young, M.A. Wajid, Design of intrusion detection system based on cyborg intelligence for security of cloud network traffic of smart cities, J. Cloud Comput. 11 (1) (2022), https://doi.org/10.1186/s13677-022-00305-6.

[15] P. Dahiya, D.K. Srivastava, Network intrusion detection in big dataset using spark, Procedia Comput. Sci. 132 (2018) 253–262, https://doi.org/10.1016/j.procs.2018.05.169.

[16] M.A. Bouke, A. Abdullah, An empirical study of pattern leakage impact during data preprocessing on machine learning-based intrusion detection models reliability, Expert Syst. Appl. 230 (June) (2023) 120715, https://doi.org/10.1016/j.eswa.2023.120715.

[17] W. Zhang, S. Tople, O. Ohrimenko, Leakage of dataset properties in multi-party machine learning, in: Proc. 30th USENIX Secur. Symp, 2021, pp. 2687–2704.

[18] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," 2018, doi: 10.48550/arxiv.1806.01246.

[19] Q. Dong, Leakage prediction in machine learning models when using data from sports wearable sensors, Comput. Intell. Neurosci. 2022 (2022), https://doi.org/10.1155/2022/5314671.

[20] S. Sicari, A. Rizzardi, A. Coen-Porisini, 5G In the internet of things era: an overview on security and privacy challenges, Comput. Netw. 179 (June) (2020) 107345, https://doi.org/10.1016/j.comnet.2020.107345.

[21] A. Ahad, et al., A comprehensive review on 5G-based smart healthcare network security: taxonomy, issues, solutions and future research directions, Array 18 (January) (2023) 100290, https://doi.org/10.1016/j.array.2023.100290.

[22] M.A. Sotelo Monge, A. Herranz González, B. Lorenzo Fernández, D. Maestre Vidal, G. Rius García, J. Maestre Vidal, Traffic-flow analysis for source-side DDoS recognition on 5G environments, J. Netw. Comput. Appl. 136 (July 2018) (2019) 114–131, https://doi.org/10.1016/j.jnca.2019.02.030.

[23] C.R. Kumar, A. Almasarani, M.A. Majid, 5G-Wireless sensor networks for smart grid- a ccelerating technology's progress and innovation in the Kingdom of Saudi Arabia, Procedia Comput. Sci. 182 (2021) 46–55, https://doi.org/10.1016/j.procs.2021.02.007.

[24] M. Luglio, M. Quadrini, C. Roseti, F. Zampognaro, A Flexible Web Traffic Generator for the dimensioning of a 5G backhaul in NPN, Comput. Netw. 221 (November 2022) (2023) 109531, https://doi.org/10.1016/j.comnet.2022.109531.

[25] F. Muheidat, K. Dajani, L.A. Tawalbeh, Security concerns for 5G/6G mobile network technology and quantum communication, Procedia Comput. Sci. 203 (2022) 32–40, https://doi.org/10.1016/j.procs.2022.07.007.

[26] S.M. Kasongo, Y. Sun, A deep learning method with wrapper based feature extraction for wireless intrusion detection system, Comput. Secur. 92 (2020), https://doi.org/10.1016/j.cose.2020.101752.

[27] G. Agrafiotis, E. Makri, A. Lalas, K. Votis, D. Tzovaras, N. Tsampieris, New York, NY, USA, A deep learning-based malware traffic classifier for 5G networks employing protocol-agnostic and PCAP-to-embeddings techniques, in: Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference, in EICC '23, Association for Computing Machinery, 2023, pp. 193–194, https://doi.org/10.1145/3590777.3590807.

[28] C. Park, K. Park, J. Song, J. Kim, Distributed learning-based intrusion detection in 5G and beyond networks, in: 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 2023, pp. 490–495, https://doi.org/10.1109/EuCNC/6GSummit58263.2023.10188312.

[29] C. Kim, S.-Y. Chang, D. Lee, J. Kim, K. Park, J. Kim, Reliable detection of location spoofing and variation attacks, IEEE Access 11 (2023) 10813–10825, https://doi.org/10.1109/ACCESS.2023.3241236.

[30] I. Ahmad, M. Basheri, M.J. Iqbal, A. Rahim, Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection, IEEE Access 6 (2018) 33789–33795, https://doi.org/10.1109/ACCESS.2018.2841987.

[31] M. S, R. M, MUD enabled deep learning framework for anomaly detection in IoT integrated smart building, e-Prime - Adv. Electr. Eng. Electron. Energy 5 (June) (2023) 100186, https://doi.org/10.1016/j.prime.2023.100186.

[32] M. Tavallaee, N. Stakhanova, A.A. Ghorbani, Toward credible evaluation of anomaly-based intrusion-detection methods, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 40 (5) (2010) 516–524, https://doi.org/10.1109/TSMCC.2010.2048428.

[33] C. Singh, A.K. Jain, A comprehensive survey on DDoS attacks detection & mitigation in SDN-IoT network, e-Prime - Adv. Electr. Eng. Electron. Energy 8 (March) (2024) 100543, https://doi.org/10.1016/j.prime.2024.100543.

[34] S. Garg, S. Batra, Fuzzified Cuckoo based clustering technique for network anomaly detection, Comput. Electr. Eng. 71 (2018) 798–817, https://doi.org/10.1016/j.compeleceng.2017.07.008.

[35] K. Al Jallad, M. Aljnidi, M.S. Desouki, Anomaly detection optimization using big data and deep learning to reduce false-positive, J. Big Data 7 (1) (2020) 1–12, https://doi.org/10.1186/s40537-020-00346-1.

[36] A.R. Al-Ghuwairi, Y. Sharrab, D. Al-Fraihat, M. AlElaimat, A. Alsarhan, A. Algarni, Intrusion detection in cloud computing based on time series anomalies utilizing machine learning, J. Cloud Comput. 12 (1) (2023), https://doi.org/10.1186/s13677-023-00491-x.

[37] M. Masdari, H. Khezri, A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems, Appl. Soft Comput. J. 92 (2020) 106301, https://doi.org/10.1016/j.asoc.2020.106301.

[38] K. Hwang, M. Cai, Y. Chen, M. Qin, Hybrid intrusion detection with weighted signature generation over anomalous internet episodes, IEEE Trans. Depend. Secur. Comput. 4 (1) (2007) 41–55, https://doi.org/10.1109/TDSC.2007.9.

[39] C.A. Catania, C.G. Garino, Automatic network intrusion detection: current techniques and open issues, Comput. Electr. Eng. 38 (5) (2012) 1062–1072, https://doi.org/10.1016/j.compeleceng.2012.05.013.

[40] J. Díaz-Verdejo, J. Muñoz-Calle, A.E. Alonso, R.E. Alonso, G. Madinabeitia, On the detection capabilities of signature-based intrusion detection systems in the context of web attacks, Appl. Sci. 12 (2) (2022), https://doi.org/10.3390/app12020852.

[41] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (1) (2021), https://doi.org/10.1186/s42400-021-00077-7.

[42] M.A. Bouke, A. Abdullah, K. Cengiz, S. Akleylek, Application of BukaGini algorithm for enhanced feature interaction analysis in intrusion detection systems, PeerJ Comput. Sci. 10 (2024) e2043, https://doi.org/10.7717/peerj-cs.2043.

[43] X. Qu, et al., A survey on the development of self-organizing maps for unsupervised intrusion detection, Mob. Networks Appl. 26 (2) (2021) 808–829, https://doi.org/10.1007/s11036-019-01353-0.

[44] A. Cohen, N. Nissim, Y. Elovici, Novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods, Expert Syst. Appl. 110 (2018) 143–169, https://doi.org/10.1016/j.eswa.2018.05.031.

[45] M.A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study, J. Inf. Secur. Appl. 50 (2020) 102419, https://doi.org/10.1016/j.jisa.2019.102419.

[46] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: a survey, Appl. Sci. 9 (20) (2019), https://doi.org/10.3390/app9204396.

[47] M.P. Novaes, L.F. Carvalho, J. Lloret, M.L. Proença, Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments, Futur. Gener. Comput. Syst. 125 (2021) 156–167, https://doi.org/10.1016/j.future.2021.06.047.

[48] I. Debicha, T. Debatty, J.-M. Dricot, and W. Mees, "Adversarial Training for Deep Learning-based Intrusion Detection Systems," 2021, [Online]. Available: http://arxiv.org/abs/2104.09852.

[49] H. Faris, et al., An intelligent system for spam detection and identification of the most relevant features based on evolutionary Random Weight Networks, Inf. Fusion 48 (June 2018) (2019) 67–83, https://doi.org/10.1016/j.inffus.2018.08.002.

[50] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, M. Conti, Distributed denial of service attacks in cloud: state-of-the-art of scientific and commercial solutions, Comput. Sci. Rev. 39 (2021) 100332, https://doi.org/10.1016/j.cosrev.2020.100332.

[51] A.K. Sahu, S. Sharma, M. Tanveer, R. Raja, Internet of Things attack detection using hybrid Deep Learning Model, Comput. Commun. 176 (April) (2021) 146–154, https://doi.org/10.1016/j.comcom.2021.05.024.

[52] S. Mishra, Blockchain and machine learning-based hybrid IDS to protect smart networks and preserve privacy, Electron 12 (16) (2023), https://doi.org/10.3390/electronics12163524.

[53] H.Y. Kwon, T. Kim, M.K. Lee, Advanced intrusion detection combining signature-based and behavior-based detection methods, Electron 11 (6) (2022) 1–19, https://doi.org/10.3390/electronics11060867.

[54] E.M. Maseno, Z. Wang, H. Xing, A systematic review on hybrid intrusion detection system, Secur. Commun. Netw. 2022 (2022), https://doi.org/10.1155/2022/9663052.

[55] D. Mohamed, O. Ismael, Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing, J. Cloud Comput. 12 (1) (2023), https://doi.org/10.1186/s13677-023-00420-y.

[56] M.A. Khan, HCRNNIDS : hybrid convolutional recurrent neural, Multidiscip. Digit. Publ. Inst. (2021).

[57] F. Farokhi, M.A. Kaafar, Modelling and Quantifying Membership Information Leakage in Machine Learning, 2020, pp. 1–13 [Online]. Available, http://arxiv.org/abs/2001.10648.
[58] A. Hannun, C. Guo, L. van der Maaten, Measuring data leakage in machine-learning models with fisher information (Extended Abstract), IJCAI Int. Jt. Conf. Artif. Intell. (Uai) (2022) 5284–5288, https://doi.org/10.24963/ijcai.2022/736.
[59] S. Kapoor, A. Narayanan, Leakage and the reproducibility crisis in machine-learning-based science, Patterns 4 (9) (2023) 100804, https://doi.org/10.1016/j.patter.2023.100804.
[60] M. Bouke, A. Abdullah, N. Udzir, N. Samian, Overcoming the challenges of data lack, leakage, and dimensionality in intrusion detection systems: a comprehensive review, J. Commun. Inf. Syst. 39 (2024) (2024) 22–34, https://doi.org/10.14209/jcis.2024.3.
[61] P. Subotić, L. Milikić, M. Stojić, A static analysis framework for data science notebooks, in: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice, 2022, pp. 13–22.
[62] A.E. Maxwell, M.S. Bester, C.A. Ramezan, Enhancing reproducibility and replicability in remote sensing deep learning research and practice, Remote Sens. 14 (22) (2022) 1–12, https://doi.org/10.3390/rs14225760.
[63] G. Kovács, A. Fazekas, mlscorecheck: testing the consistency of reported performance scores and experiments in machine learning, Neurocomputing 583 (March) (2024) 127556, https://doi.org/10.1016/j.neucom.2024.127556.
[64] S. Samarakoon, et al., 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network, 2022 [Online]. Available, http://arxiv.org/abs/2212.01298.
[65] J. Brownlee, Imbalanced classification with python, Mach. Learn. Mastery (2020) 463.
[66] G. Dong, H. Liu, Feature Engineering For Machine Learning and Data Analytics, CRC Press, 2018.
[67] S. Mukkamala, A.H. Sung, Feature ranking and selection for intrusion detection systems using support vector machines, in: Proceedings of the Second Digital Forensic Research Workshop, 2002, pp. 1–10.
[68] H. Ding, P.-M. Feng, W. Chen, H. Lin, Identification of bacteriophage virion proteins by the ANOVA feature selection and analysis, Mol. Biosyst. 10 (8) (2014) 2229–2235.
[69] K.J. Johnson, R.E. Synovec, Pattern recognition of jet fuels: comprehensive GC$ \times$ GC with ANOVA-based feature selection and principal component analysis, Chemom. Intell. Lab. Syst. 60 (1–2) (2002) 225–237.

Mohamed Aly Bouke holds a Master's and a Ph.D. in Information Security from the University of Putra Malaysia, specializing in cybersecurity, cyber warfare, and machine learning applications in information security. He is a member of the IEEE (Institute of Electrical and Electronics Engineers), reflecting his involvement in the technology and engineering community. In his role with the International Information System Security Certification Consortium (ISC2), Mohamed contributes to advancing cybersecurity practices. He is a certified trainer for various international organizations and engages in educating students worldwide through training programs. His expertise extends to authoring publications and participating as a manuscript reviewer for recognized journals, furthering his engagement in the cybersecurity field. As a public speaker and author, Mohamed shares his knowledge and insights, adding value to discussions and literature in information security.

Azizol Abdullah received the M.Sc. degree in engineering (telematics) from The University of Sheffield, U.K., in 1996, and the Ph.D. degree in parallel and distributed systems from Univer- siti Putra Malaysia, Malaysia, in 2010. He is an Associate Professor with the Department of Tech- nology and Communication Networking, Faculty of Computer Science and Information Technol- ogy, Universiti Putra Malaysia. He is the Head of the Network, Parallel, and Distributed Computing Research Group and a member of the Information Security Research Group at the Faculty of Computer Science and Information Technology, UPM. At the national level, he is a member of Cyber Security Academia Malaysia (CSAM). He was also appointed as a Fellow Researcher for ITU-UUM Asia Pacific Center of Excellence For Rural ICT Development (ITU-UUM). He has also been involved as a consultant for AnyCast@MyDNS Project, MyNIC and Ministry of Science and Innovation projects, Malaysia (MOSTI) and Integrated Sports Management System Project, Ministry of Youth and Sports, Malaysia. His main research areas include cloud and grid computing, network security, wireless and mobile computing and computer networks. He is engaged in Malware Detection research, SDN, SDWAN network research and SDWAN Security research.