Nigerian Journal of Operations Research (NIJOR)

https://nijor.org.ng/
pISSN: 2141-7938
eISSN: 2141-7938

Towards Real-Time Network Intrusion Detection: An Enhanced Feature Selection Framework Using Variance Analysis and K-Means Clustering

Usman Ahmad Baba [1,*], Etemi Joshua Garba[2], Asabe Sandra Ahmadu[2], Maria Lapina[3 & 4]

[1] Department of Data Science and Artificial Intelligence, Faculty of Computing, Modibbo Adama University, Yola, Adamawa State, Nigeria; usmanbabatela@gmail.com (U. A. B) - https://orcid.org/0000-0001-7270-2018
[2] Department of Computer Science, Faculty of Computing, Modibbo Adama University, Yola, Adamawa State, Nigeria; e.j.garba@mau.edu.ng (E.J.G); ahmaduasabe@mau.edu.ng (A.S.A)
[3] Department of Computational Mathematics and Cybernetics, North-Caucasus Federal University, 355017, Stavropol, Russia; mlapina@ncfu.ru (M.L)
[4] Trusted AI Research Center, RAS, 109004 Moscow, Russia;

ARTICLE INFO

Article history:
Received 19 April 2026
Revised 14 May 2026
Accepted 9 June 2026
Available online 29 June 2026

Keywords:

Network Intrusion Detection Systems; Feature Selection; Random Forest; Support Vector Machine; XGBoost; NSL-KDD; UNSW-NB15.

ABSTRACT

The cybersecurity threat landscape is increasingly dynamic, making Network Intrusion Detection Systems (NIDS) essential to cyber defense. NIDS face challenges such as network traffic imbalance, high false alarm rates, and suboptimal detection accuracy. To address these issues, we integrated variance analysis and k-means clustering, thereby enhancing computational efficiency across seven machine learning models evaluated on the UNSW-NB15 and NSL-KDD datasets. Key metrics included Accuracy, Precision, Recall, F1-Score, Model Training Time (MTT), and Inference Time (IT). Our feature selection technique achieved notably lower MTT and IT than existing methods, particularly with the XGB and RF models, which saw MTT drop by 80% and IT by 60% compared to the baseline. This efficiency is pivotal for real-time intrusion detection and resource-constrained NIDS, enabling swift threat response and optimized resource use. These computational gains emphasize the technique's potential for real-time network security.

1. Introduction

From a Cybersecurity threat perspective, the last several years have been the most eventful, especially during and after the COVID-19 pandemic. Globally, the number of connections and devices is growing faster than the population and the number of internet users, according to Cisco (2020). Also, in recent times, the rise of novel technologies has heightened concern about cybercrime. The tactics of cybercriminals are continually advancing as they seek to exploit newly discovered vulnerabilities, posing a greater risk to individuals and organizations worldwide. Sophisticated attack vectors such as Advanced Persistent Threats (APT), phishing, distributed denial-of-service (DDoS), Business Email Compromise (BEC), and ransomware are now prevalent, posing high costs to businesses and individuals (Park *et al.*, 2023). This is essentially due to the continuous expansion of networks, increasing complexity, the vast amounts of data and applications involved, and the rapid transition to an increasingly digitalized economy, especially in Africa's largest economy, Nigeria (Interpol, 2023).

[*] *Corresponding author.*
E-mail address: usmanbabatela@gmail.com

A total of 37 African nations have established universal access funds to extend Internet coverage across their countries. As a result, the African continent now boasts one of the fastest rates of connectivity expansion globally (Interpol, 2023) – this indicates a significant increase in cybercrime and threats faced by organizations.

Increased digitization makes strong cybersecurity more important than ever; hence, intelligent systems must be able to achieve a goal even in the presence of adversaries. These goals (security policies) are key characteristics of information that make it valuable to an entity or organization. For any entity or organization to protect its assets, the goal often falls into one of three categories: to protect the confidentiality, integrity, and availability of information (Whitman & Mattord, 2019). After identifying assets and their values, information systems must include the appropriate mechanisms or controls to enforce security policies to protect those assets. These security controls can typically be categorized into detection and prevention mechanisms. Prevention mechanisms prevent security policies from being violated, e.g., passwords, encryption, biometric authentication, firewalls, etc., while detection mechanisms detect violations, e.g., motion sensors, honeypots, Intrusion Detection System (IDS), etc. (Gollmann, 1999).

Protecting assets against present-day threats and learning how to identify and assess future threats are crucial aspects of cybersecurity. It involves understanding the risks and rewards for adversaries, as well as the costs and benefits for individuals and organizations (Chaudhary *et al.*, 2023). Achieving comprehensive cybersecurity is challenging but essential, and efforts must be made to secure all possible entry points while understanding that an adversary only needs to find one vulnerability. Security Mechanisms must include early detection and prevention of intrusions that will/may compromise the confidentiality, integrity, and availability of assets (Gollmann, 1999). The focus should not solely be on prevention but also on detection. Even if an adversary breaches defenses, quick detection can minimize the impact on the affected party and disrupt the adversary's advantage. Prioritizing proactive security measures can dissuade potential attackers, ultimately making an individual or organization a less appealing target (Kaur *et al.*, 2023).

Intrusion Detection is a technique in computer networks for detecting various forms of attack on a network or system (Buczak & Guven, 2016). An intrusion detection system is a piece of hardware or an application that monitors for policy violations or malicious behavior on a system. There are two main types of intrusion detection systems: network-based and host-based (Mahmoud *et al.*, 2025). One system that keeps an eye on all the traffic entering a network is called a network-based intrusion detection system (NIDS). In contrast, a host-based intrusion detection system (HIDS) resides on a host machine and monitors all activities running on that machine (Whitman & Mattord, 2019). Network traffic is compared against a knowledge base to differentiate between normal and malicious activity patterns in a network-based IDS (Acharya & Singh, 2017).

Network-based IDS (NIDS) are of two main types: signature-based and anomaly-based. Signature-based IDS scans all incoming data packets for known malicious activity by comparing them to a database of signatures. These signatures consist of attributes that specify predetermined attack patterns (Tamy *et al.*, 2021). An anomaly-based IDS, on the other hand, typically relies on knowledge of normal behavior to build a baseline or model and then checks and compares network activity against the baseline using statistical techniques so it can flag any abnormal behavior (Kayacik *et al.*, 2005).

Because it searches for any abnormal behavior, an anomaly-based IDS can detect new forms of attack; however, it requires significant processing power and overhead and is also prone to false alarms (Whitman & Mattord, 2019). Determining effective IDS models for normal behavior – which normally requires human intervention - can be a very demanding task, especially in large and complex systems due to the amount of generated network traffic. To curb the challenges and improve legacy security solutions, the attention of both industry and academia has turned to intelligent security solutions, such as Artificial Intelligence (AI) and machine learning, to enhance the efficiency of IDSs (Abdulhammed

et al., 2019; Al-Turaiki & Altwaijry, 2021; Aljehane *et al.*, 2024; Bhati *et al.*, 2020; Mohammed *et al.*, 2023; Singh & Keikhosrokiani, 2023).

Machine Learning (ML) is a branch of AI that focuses on developing systems that can improve their performance through experience, i.e., by learning from data (Baba *et al.*, 2018). ML approaches are broadly divided into supervised and unsupervised learning. Supervised learning is where models learn from labeled examples, while in unsupervised learning, patterns within the data are discovered, even without predefined outputs. Classification and regression are examples of supervised learning, while clustering and dimensionality reduction are examples of unsupervised learning. The set of rules that ML algorithms learn is called a model. Once a model has been built, new observations can be fed into it, and predictions will be made for those observations. The process by which the model is learned is called the algorithm (Nsang & Baba, 2018; Rhys, 2020).

Dimensionality reduction is an unsupervised learning technique that simplifies data by reducing its feature space while preserving essential information, i.e., retaining the most informative features and eliminating the irrelevant or redundant ones. It is particularly useful for visualizing or processing high-dimensional datasets (Nsang *et al.*, 2015). Some common approaches to dimensionality reduction include Principal Component Analysis (PCA), Variance, Random Projection (RP), and Singular Value Decomposition (SVD), among others (Baba *et al.*, 2018). However, many existing dimensionality reduction techniques may not be optimized for the unique characteristics of intrusion detection datasets. This is because of the complex nature of network data, which can lead to compromised detection accuracy, loss of critical information, or excessive computation time (Akashdeep *et al.*, 2017; Shandilya *et al.*, 2023).

ML is not only used to defend against cyberattacks, but also to conduct cyberattacks (Xu, 2020). This means that soon there will be competition in using Machine Learning and other Artificial Intelligence technologies for both cyberattacks and defense against them. Therefore, Machine Learning is an indispensable component in securing cyberspace (Adenusi *et al.*, 2025; Chinnasamy *et al.*, 2025; Fang & Leng, 2025). This makes it necessary to stay up to date in developing new tools that will aid in the battle against cyberattacks, and ultimately, an aggressive push is expected towards proactive and advanced analytics that can augment the network defense systems in efforts to identify new threats and variants of earlier threats (Hande & Muddana, 2020; Kaur *et al.*, 2023; Khraisat et al., 2019; Singh & Keikhosrokiani, 2023).

The domain of network intrusion detection systems faces a significant and persistent challenge in efficiently analyzing vast, complex datasets, as it primarily deals with many observations gathered from network traffic (Nimbalkar & Kshirsagar, 2021). Issues such as increased computational time, overfitting, and reduced model interpretability affect traditional machine learning methods when dealing with high-dimensional data (Acharya & Singh, 2017). By transforming data into a reduced feature space while preserving relevant information, dimensionality reduction offers a potential solution to these challenges (Baba *et al.*, 2018). However, Potential information loss, decreased detection accuracy, and subpar performance are all possible outcomes if current dimensionality reduction approaches are not properly tailored to NIDS's unique requirements (Akashdeep *et al*., 2017; Bansal & Kaur, 2019; Shandilya et al., 2023; Zong *et al.*, 2019).

According to Alsaleh and Binsaeedan (2021), data preprocessing and model training are key factors that determine the final detection capabilities and efficiency of Network Intrusion Detection Systems. The features considered during preprocessing and model training have a significant impact on NIDS detection rate, making this stage the most important in the IDS process. Furthermore, redundant, imbalanced, and ineffective features commonly contribute to the variability in classifier performance across datasets (Wu & Li, 2021). There are substantial time and labor costs associated with this preprocessing stage.

To address these challenges, researchers have proposed approaches that optimize IDS efficiency by integrating feature selection and multiple machine learning techniques (Aljawarneh *et al.*, 2018; Almasoudy *et al.*, 2020; Bojarajulu *et al.*, 2021; Damtew *et al.*, 2023; Faris *et al.*, 2023; Hao *et al.*, 2018; Iwendi *et al.*, 2020; Kamarudin et al., 2019; Lifandali et al., 2023; Malhotra & Sharma, 2019; Mohammadi *et al*., 2019; Nimbalkar & Kshirsagar, 2021; Sumaiya Thaseen & Aswani Kumar, 2017; Tamy *et al.*, 2021; Zhang *et al*., 2016; Zhang *et al.*, 2023). A lot of these approaches, however, yielded unsatisfactory results in terms of network traffic packet data imbalance issues, detection accuracy, detection rate, precision, recall, and false alarm rate in the NIDS (Dubey & Bhujade, 2021; Hao *et al*., 2018).

In this context, there is a strong need for more approaches tailored to NIDS requirements. This research attempts to bridge this gap by proposing a novel approach that integrates two separate statistical techniques - variance analysis and k-means clustering preservation. This research, therefore, aims to select an optimized subset of features from the original dataset, thereby enhancing the performance of ML algorithms applied to NIDS.

2. Literature Review

Acharya and Singh (2017) proposed a hybrid intrusion detection model that combines the intelligent water drops (IWD) method to reduce features and the support vector machine (SVM) to evaluate the reduced features. The performance of the proposed hybrid technique was compared with that of SVM, Naïve Bayes, K-means, and a combination of Genetic Algorithm (GA) and SVM, on the KDD CUP 99 dataset. IWD + SVM outperformed GA + SVM, achieving a detection rate of 97.19% compared to 99.41%. With an accuracy of 99.09%, the proposed model outperformed GA + SVM with 97.76%. The precision and false alarm rates for the proposed model are 99.12% and 1.41%, respectively, whereas GA + SVM achieves 98.11% and 2.40%, respectively. Overall, the performance of the proposed model was better than that of the other existing models it was compared against. However, the model was not evaluated on recently captured data.

Hao et al. (2018) proposed a new Feature Selection technique that combines five feature selection methods to choose the most important features by using the majority voting rule. The authors then used an SVM with the proposed technique (25 features) to improve IDS performance. Experiments on the NSL-KDD dataset show that the proposed technique outperforms SVM without Feature Selection. However, the authors reported a relatively low accuracy of 76.46% and did not address the dataset's imbalance. There is also no report of the preprocessing steps used.

A dimensionality reduction method that reduces the dimension of data using a combination of Symmetric Uncertainty, Chi-Squared, and ReliefF to select the most important features was proposed by Bansal and Kaur (2019). The authors used the NSL-KDD and CICIDS 2017 datasets for the experiment. They compared the performance of their proposed technique with that of the classifiers without it and found that the classifiers' performance improved when their technique was used. They used four machine learning algorithms: Extreme Gradient Boosting Machine (XGBoost), Support Vector Machine (SVM), Neural Network, and Conditional Inference Tree (CTree)Nnet, and reported that XGBoost outperformed the other three classifiers with an accuracy of 98.85% when the proposed technique was used, as against 94.3% without the proposed technique. However, the authors failed to address the data imbalance in the datasets.

A NIDS model was developed by Kim *et al.* (2020). The authors employed Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to address challenges in network intrusion detection. They conducted evaluations using well-established metrics, including Precision, Accuracy, Recall, and F1-Score, on two diverse datasets: CSE-CIC-IDS2018 and KDD CUP99. While previous deep learning-based studies on the KDD dataset typically focused on binary classifications, distinguishing benign traffic from attacks across all categories, this work innovatively addressed the multiclass classification problem. Specifically, it aimed to distinguish among the four distinct attack

categories within the KDD dataset. However, this study had certain gaps, such as the lack of feature selection, and was limited in scope, focusing primarily on Denial-of-Service (DOS) attacks. Additionally, the proposed CNN model outperformed the RNN model in terms of accuracy, especially on the KDD dataset, where it consistently achieved 99% accuracy in both binary and multiclass classifications. However, the resource-intensive nature of the process and the RNN model's lower performance, especially on the CSE-CIC-IDS 2018 dataset, have identified areas where further research and optimization may be useful for IDS.

Using the CICIDS2017 dataset, which was created for IDS research, Alfrhan et al. (2020) conducted experiments with three classifiers – random forest, naive Bayes, and k-nearest neighbors (K-NN) – on the entire unbalanced dataset. After that, the authors used the SMOTE method to compare the results of balanced and unbalanced datasets. Based on the overall performance of the Decision Forest, SVM, and Decision Jungle classifiers, applying the SMOTE approach improved performance. However, neither Feature Selection nor normalization techniques were reported by the authors during preprocessing.

Another study that was recently conducted by Ashiku and Dagli (2021) worked on enhancing IDS efficacy using deep learning techniques. To address the challenge of identifying known and zero-day network behavioral features, they used a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The authors acknowledged deep learning's potential to enhance IDS systems with adaptive learning capabilities. They used a deep learning classification architecture combined with semi-dynamic hyperparameter tuning. Their proposed approach outperformed similar deep learning-based network IDSs by leveraging enhancements to multiclass models. The overall accuracy rates of 95.4% and 95.6% that were achieved for both pre-partitioned and user-defined multiclass classifications highlight this enhancement. However, the study did not include feature selection procedures or data on the False Alarm Rate; the model also has a low detection rate, among other shortcomings.

Another research was conducted by Alsaleh and Binsaeedan (2021) on the impact of feature selection on anomaly intrusion detection in network security. The authors proposed the Salp Swarm Algorithm (SSA) as a powerful tool for enhancing feature selection in ML-based IDSs. The research showed that the choice of functions during data preprocessing and model training significantly affects the speed of intrusion detection. They overcame this obstacle by using SSA to improve model performance by selecting the most appropriate features. The network anomaly detection model was built using two classification methods: naive Bayes (NB) and extreme gradient boosting (XGBoost), in combination with SSA (i.e., SSA-XGBoost and SSA-NB).

The results of the research demonstrated the effectiveness of using SSA to select features, achieving a high detection rate, a low false alarm rate, and high accuracy. The suggested model demonstrated exceptional detection capabilities, surpassing other approaches with a recall of 99.1% and a False Negative Rate of 0.6%. However, it's worth noting that the proposed technique was outperformed by another algorithm (SVM-SPDO-BPSO), and the authors did not report on the Inference Time of the proposed technique, indicating that further improvements are needed in feature selection techniques to increase the detection rate. To improve the model's overall performance, it is crucial to address data imbalance and investigate techniques tailored to different datasets.

Another common challenge in this domain is the variability of classifiers across datasets, which is often an obstacle in real-world applications. This aligns with the problem addressed in Wu and Li (2021) and is often attributed to redundant features. A comprehensive framework incorporating feature selection and multiple ML techniques was by Wu and Li (2021). The feature selection approaches used in the work include Correlation-based Filter (CBF), Fast Correlation-based Filter (FCBF), and Consistency approaches (CM), along with Random Forest and Back Propagation algorithms on two well-known intrusion detection datasets, namely KDD CUP99 and Honeynet. Their proposed method

achieved 94.8% accuracy, underscoring the need for more innovative approaches to improve NIDS detection accuracy.

Dubey and Bhujade (2021) proposed two (2) feature selection techniques - Dense FR and Sparse FR - which are used to generate ideal feature subsets for ML-based IDS. Their findings suggest that, when using Naïve Bayesian Classifiers and One vs. Rest Classifiers via Logistic Regression, the proposed techniques perform better than a Multilayer Perceptron. The authors were silent about the dataset's imbalance and concluded that their technique might not be the ideal approach to feature selection, even though it slightly boosts the performance of some classifiers.

Herrera-Semenets *et al.* (2021) proposed a novel feature selection algorithm tailored to IDS. The algorithm offers a more comprehensive approach to feature selection by combining many indicators. However, the authors did not directly compare the proposed feature selection algorithm with existing algorithms on the same dataset, thereby providing a more reliable estimate of its performance. Much of the work we have reviewed in the NIDS domain focuses on improving feature selection methods, particularly filter-based methods that have been proven effective at identifying relevant features in this domain. These reviewed studies noted that one notable limitation that leads to bias in feature selection is reliance on a single indicator, which can inadvertently exclude relevant features.

Classification algorithms work best on uniformly distributed datasets. In addition, most real-world applications typically exhibit an uneven distribution of classes. Class imbalance is therefore acknowledged as a difficult issue for data mining and machine learning methods. The class with a larger number of instances in an unbalanced dataset is referred to as the major class, and the class with a relatively smaller number of instances is referred to as the minor class. The main cause of class imbalance is that most instances and examples come from the majority class, whereas the minority class has few. Many algorithms prioritize classifying the major class while neglecting or even misclassifying the minority class, leading to this issue.

Based on the above literature, building IDS models using old datasets, low detection rates, data imbalance, high inference time, and the lack of comparison between Feature Selection algorithms proposed in several studies and other baseline algorithms on the same dataset are among the major concerns in NIDS research. Hence, this work set out to address these lingering issues by proposing a novel Dimensionality Reduction technique tailored to the specific demands of Network Intrusion Detection Systems to enhance accuracy, efficiency, model training time, and inference time.

3. Methodology

This section presents the NIDS architecture, which models the steps involved in selecting features from the datasets used to build the ML models. Figure 1 shows a Machine Learning pipeline that uses the UNSW-NB15 and NSL-KDD datasets to build a network intrusion detection system. Two datasets were used mainly because a single dataset would not be sufficient to draw conclusions about the robustness of the methods involved (Alam *et al.,* 2025).

Exploratory Data Analysis (EDA) is first conducted on the datasets to gain deeper insights, and then data cleaning is performed to remove redundant data and fill in missing values. One-Hot Encoding for categorical variables, and Min-Max Normalization to scale features are the first steps in the pipeline's data pre-processing phase. After that, we use the Synthetic Minority Over-sampling Technique (SMOTE) to fix the dataset's class imbalance. The data is then split into a training and a test set during the train-test split. The training set is passed on to the feature selection stage.

At the next stage, models are trained using several classification algorithms, including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), Naive Bayes (NB), XGBoost, and a Multi-Layer Perceptron (MLP). After training, the features selected from the test dataset are used in the final evaluation phase. Model performance is then assessed using metrics such as Accuracy, Precision, Recall, and F1-score, along with computational indicators such as Model Training Time (MTT) and Inference Time (IT).

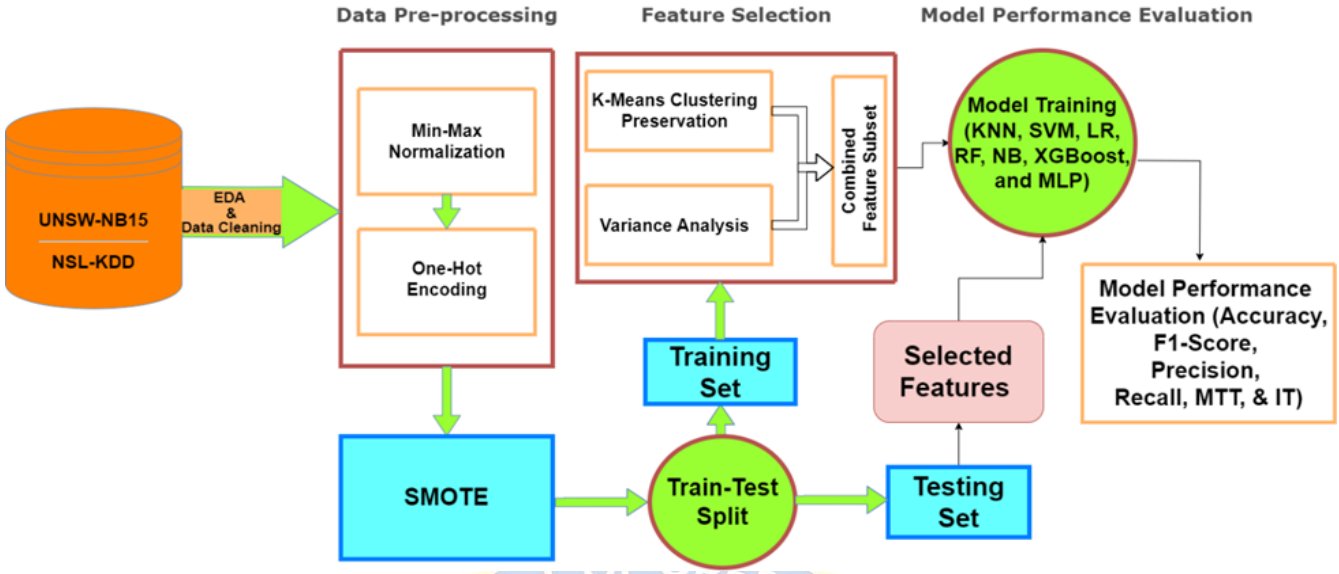

Figure 1. Network Intrusion Detection System Architecture.

3.1 Data Collection and Preprocessing

This section provides basic information about the datasets used for the work. The datasets are collected from various sources, including NSL-KDD and UNSW-NB15. These datasets contain various types of cyberattacks, including probing, denial-of-service, and other forms. While UNSW-NB15 includes more relevant and diverse forms of cyber-attacks, NSL-KDD is an improved version of the KDD Cup 1999. Preliminary data processing includes handling missing values, removing irrelevant features, and normalizing the data. The missing values are processed using conditional calculation.

One-Hot Encoding (OHE) was used to transform categorical features in both datasets. In OHE, each distinct category is assigned its own column in the resulting representation, allowing algorithms to precisely interpret the data and retain categorical information (Raschka & Mirjalili, 2019). Mathematically, OHE is represented as in (1).

$$OHE(x_i) = \begin{cases} 1, & if\ x_i = category \\ 0, & Otherwise \end{cases} \quad (1)$$

Where x_i represents the categorical variable and category represents one of the categories within the variable.

Classification techniques perform well when applied to a dataset that is evenly distributed. However, real-world applications often exhibit uneven class distributions. In an unbalanced dataset, the class with the greater number of instances is the major class, while the class with a relatively smaller number of instances is the minor class. The imbalance problem refers to a situation in which the majority class contains many more examples than the minority class.

Both sets of data showed class imbalance, with a disproportionately high number of harmless cases compared to attack cases. To address the potential bias caused by this imbalance and ensure that the models generalize effectively, the Synthetic Minority Over-sampling Technique (SMOTE) was utilized, as shown in Figure 2.

SMOTE functions by creating synthetic (artificial) instances of the minority class (attacks), effectively balancing the class distributions. In other words, it is a data augmentation technique that generates synthetic samples for the minority class to balance the class distribution in the dataset.

Empirical evidence shows that this method enhances classifier performance on imbalanced datasets by preventing models from being biased towards the majority class (Chawla et al., 2002).

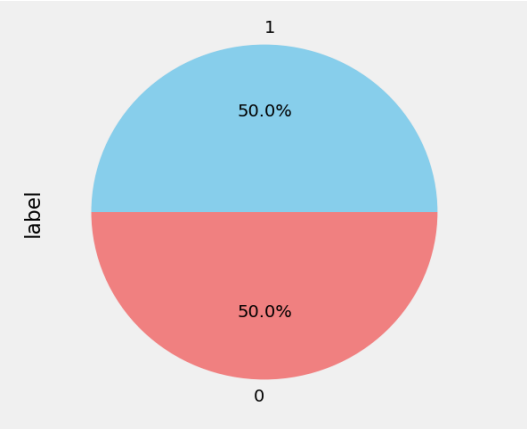

Figure 2. Distribution of the target variable after applying SMOTE to both datasets.

Normalization is an important part of many ML algorithms to make sure that all features have an equal impact on the model's predictions. The process of normalization prevents features with larger ranges from dominating the learning process (Raschka & Mirjalili, 2019). In this work, all features were normalized before the data was fed into the machine learning models to ensure they were all on the same scale. Numerical data is commonly scaled to the interval [0, 1] using techniques such as Z-scores and min-max scaling. In particular, the Min-Max normalization method was used in this research, which, as shown in (2), maps feature values to the interval [0, 1].

$$X_{Normalized} = \frac{X - X_{Min}}{X_{Max} - X_{Min}} \quad (2)$$

where X represents the original feature value, X_{Min} is the minimum value of the feature, and X_{Max} is the maximum value of the feature (De Amorim et al., 2023).

3.2 Feature Selection

After the preprocessing step and SMOTE, the feature selection phase follows. Finding the most important features in the preprocessed datasets is the goal at this stage. Using a combination of two statistical metrics, variance and k-means clustering preservation, the preprocessed and split datasets are subjected to the PFS technique to identify the most pertinent features in the dataset.

Features with the highest variability are identified using variance analysis, potentially capturing significant data patterns (Karami *et al.*, 2023). K-means clustering preservation ensures that features maintain the underlying data structure, fostering the discovery of intricate intrusion patterns (Baba et al., 2018). To obtain a score for each feature, the proposed technique combines the metrics by assigning weights to each and summing them. To find the optimal combination of weights that maximizes the performance of the ML algorithms, we use a grid search.

3.2.1 Variance

Features with higher variance are more likely to capture significant data trends; this is where the variance approach focuses (Karami *et al.*, 2023). Algorithm 1 shows the process for applying the variance technique to reduce dataset D to dataset DR.

$$DR = D\ (:, Ir), \quad (3)$$

where DR has the same number of rows as D and r columns, and the i-th column of DR is the column of the original dataset D with the i-th largest variance (Baba *et al.*, 2018).

Algorithm 1: Feature Selection based on Variance Approach

Input: D: Original dataset with n features. r: Number of top features to select.
Output: D_R: Reduced dataset containing top r features.

1. Initialize an empty set $I \leftarrow \emptyset$.
2. Calculate the variance σ_j^2 for each dimension (feature) $j \; in \; D$.
3. Sort all dimensions in descending order based on their variances: $i_1, i_2, \ldots, i_n$, where $\sigma_{i_1}^2 \geq \sigma_{i_2}^2 \geq \cdots \geq \sigma_{i_n}^2$.
4. Select the top r dimensions: $I \leftarrow \{i_1, i_2, \ldots, i_r\}$
5. Construct the reduced dataset D_R by extracting columns corresponding to the indices in I from the original dataset : $D_R = \{d \in D \mid feature \; index \; of \; d \in I\}$.
6. Return D_R.

3.2.2 K-Means Clustering Preservation

The goal of the k-means clustering preservation technique, often referred to as the top-down approach (Nsang, 2011), is to find a lower-dimensional representation of the high-dimensional dataset D that preserves the k-means clustering structure of the original data as much as possible. The process is given in Algorithm 2.

Algorithm 2: K-means Preservation Feature Selection

Input: D (dataset), p (features), m (target)
Output: D_R (reduced dataset)

1. $R \leftarrow K - means(D)$
2. $L_{curr} \leftarrow \{1, \ldots, p\}$,
3. $t \leftarrow p - 1$
4. **while** $t \geq m$ **do**
5. $\quad LC \leftarrow \{C \subset L_{curr} : |C| = t\}$
6. $\quad AM \leftarrow \emptyset$
7. $for \; each \; C \in LC \; do$
8. $R_C \leftarrow K - means(D|_C)$
9. $score \leftarrow RandIndex(R, R_C)$
10. $AM \leftarrow AM \cup \{(C, score)\}$
11. $end \; for$
12. $L_o \leftarrow arg \; arg \; (AM)$
13. $L_{curr} \leftarrow L_o$
14. $t \leftarrow t - 1$
15. $end \; while$
16. $D_R \leftarrow D|_{L_o}$
17. $return \; D_R$

The key idea here is to iteratively reduce the number of attributes by one at each step, while preserving the k-means clustering as much as possible. The algorithm starts with all p-1 attribute combinations, finds the best one, then considers all p-2 attribute combinations from the best p-1 set, and so on, until reaching the desired m dimensions.

This top-down approach ensures that the final lower-dimensional representation Dl closely retains the clustering structure of the original high-dimensional data D, as measured by the chosen clustering performance measure (Rand).

3.2.3 Proposed feature selection technique

The proposed feature selection technique combines the variance approach and the k-means clustering preservation approach by assigning weights to each metric and summing them to obtain a score for each feature. The weights are optimized using a grid search, which evaluates the best

combination of weights to maximize the performance of the ML algorithms. The Algorithm for the Proposed Feature Selection (PFS) technique is given in Algorithm 3.

Algorithm 3: PFS Algorithm

Input: Original dataset with n samples and m features
Output: Reduced feature subset for intrusion detection

1. $\forall j \in \{1, \dots, p\}: \sigma_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_{ij} - \underline{x_j}\right)^2$
2. $L_1 = sort(\{1, \dots, p\})\ by\ \sigma^2$
3. $R = arg\ min_S \sum_{i=1}^{k}\sum_{x \in S_i}||x - \mu_i||^2$
4. $\forall j \in \{1, \dots, p\}: S_j = Silhouette(D_j, R)$
5. $L_2 = sort(\{1, \dots, p\})\ by\ S$
6. $\forall j \in \{1, \dots, p\}: Score(j) = 0.6 \cdot pos(j, L_1) + 0.4 \cdot pos(j, L_2)$
7. $I_{best} = \{j_1, j_2, \dots, j_k\}\ where\ Score(j)\ is\ max\ D_{PFS} = D|_{I_{best}}$
8. $Return\ D_{PFS}$

The algorithm's operating principle is as follows:

Step 1: Feature Selection using Variance Analysis

i. Calculate the variance of each feature across the dataset.
ii. Generate a list L1 such that L1[i] = p if p is the i-th attribute with the largest variance.

Step 2: Clustering Preservation using K-means

i. Apply the K-means clustering algorithm on the original dataset.
ii. Obtain cluster assignments for each data point.
iii. For each feature in the original dataset, calculate its silhouette score based on how well it preserves the k-means clustering of the original dataset.
iv. Generate a list L2 such that L2[i] = p if p is the i-th attribute that best preserves k-means clustering.

Step 3: Combined Feature Subset

i. The feature selection criteria from Steps 1 and 2 are combined to form the overall selection criterion, which is then used to select k attributes from the original dataset to produce the reduced dataset.
ii. In combining the feature selection criteria from Steps 1 to 2, variance is given a weighting of 6, and k-means clustering preservation is given a weighting of 4.

3.2.4. Correlation-based Feature Selection

Machine learning uses the dimensionality reduction approach known as Correlation-based Feature Selection (CFS) to identify relevant features in a dataset. By considering both the individual predictive power and the redundancy among features, the CFS evaluation process determines the value of a subset of features. The fundamental premise is that effective feature subsets consist of features that are highly correlated with the target class and uncorrelated with one another (Hall, 1999). This reduces redundancy and enhances the model's overall predictive power. The merit of a feature subset S consisting of k features is given in (4).

$$Merit\ (S) = \frac{k.\underline{r_{cf}}}{\sqrt{k + k(k-1).\underline{r_{ff}}}} \tag{4}$$

where k is the number of features in the subset S, $\underline{r_{cf}}$ is the average correlation between the features and the class label, $\underline{r_{ff}}$ and is the average inter-correlation among the features in the subset S.

The numerator $k.\underline{r_{cf}}$ represents the total correlation of all features with the class. Multiplying the average feature-class correlation $\underline{r_{cf}}$ by the number of features k gives a measure of how well the features, collectively, correlate with the class.

The denominator $\sqrt{k + k(k-1).\underline{r_{ff}}}$ represents a measure of redundancy among the features. The term k accounts for the individual contribution of each feature, while $k(k-1).\underline{r_{ff}}$ accounts for the pairwise correlations among the features, adjusting for redundancy.

3.3 Classification Techniques

In the classification stage, the selected features are used to train and test various ML models. According to Wolpert and Macready (Wolpert & Macready, 1997), the "No Free Lunch" theorem and related empirical evidence suggest that no classification algorithm is universally superior. The effectiveness of an algorithm depends on the context, i.e., the dataset and the problem at hand. To determine which algorithm works best for a specific problem, researchers usually try multiple algorithms and often use cross-validation to identify the best-performing one. A brief description of the algorithms used in this work is covered in the following subsections.

3.3.1 K-Nearest Neighbor

One of the simplest and most straightforward ML algorithms is the K-Nearest Neighbors (KNN) algorithm. The core idea behind KNN is to determine the class of a new data instance by considering the labels of its k closest neighbors in the dataset. *K* in KNN is a parameter that specifies how many nearby points are considered and is chosen by the user, usually as a relatively small positive integer (Chio & Freeman, 2018; Cover & Hart, 1967). Figure 3 illustrates the KNN method in a two-dimensional space, where data points belong to two categories: rectangles for Class 1 and triangles for Class 2. The aim here is to predict the class of the new unclassified data point, which is represented by a question mark ('?'). This will be done based on its nearest neighbors.

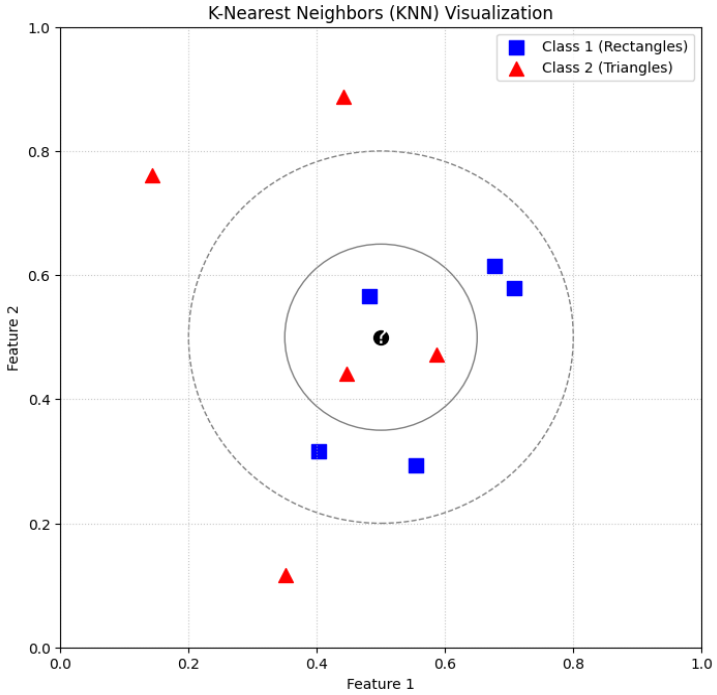

Figure 3. Visualizing the K-Nearest Neighbor classification process

The first step is to determine the nearest neighbors, and the second step is classification based on the chosen K value and a majority vote. If K is small (e.g., 3), the algorithm considers the three closest neighbors to the unclassified point. If K is larger (e.g., 5), it considers a larger neighborhood. The unclassified point ('?') is then classified into the class that most of its nearest neighbors belong to.

In essence, the KNN algorithm uses proximity to classify new data points. By adjusting the value of K and observing the decision boundaries (inner and outer circles), we can understand how the algorithm adapts to classify data points based on their nearest neighbors in a 2-dimensional feature space.

3.3.2 Naive Bayes

Naive Bayes (NB) is a probabilistic classifier based on Bayes' Theorem, assuming that the features are conditionally independent given the class label. It is a simple yet very effective method. The probability of a class C given a feature vector $x_1, x_2, \dots, x_n$ is given in (5).

$$P(x) \propto P(C) \prod_{i=1}^{n} P(C) \quad (5)$$

where P(C) is the prior probability of class C, and $P(C)$ is the likelihood of feature x_i given class C (Bonaccorso, 2017; Chio & Freeman, 2018; Murphy, 2012).

3.3.3 Random Forest

Random Forest (RF) is an ensemble technique used for classification and regression applications. During training, it generates many decision trees and then combines their predictions to improve the classifier's overall performance and accuracy. To improve generalization and decrease overfitting, each tree in the forest is trained on a randomly selected subset of features and a random subset of data (Breiman, 2001; Chio & Freeman, 2018). Equation (6) summarizes the RF prediction process.

$$\hat{y} = majority\ vote\ (T_1(x), T_2(x), \dots, T_n(x)) \quad (6)$$

where $T_i(x)$ represents the prediction from the i-th decision tree.

3.3.4 Logistic Regression

For binary classification, one commonly used statistical method is Logistic Regression (LR). It predicts the probability that an input belongs to a specific class. A classification decision is made based on the output, which is a probability value between 0 and 1 (Chio & Freeman, 2018; Raschka & Mirjalili, 2019).

The Logistic Regression model is summarized in (7)

$$P(x) = \sigma(w \cdot x + b) \quad (7)$$

where $\sigma(z) = \frac{1}{1+ e^{-z}}$ is the sigmoid function, w is the weight vector, x is the feature vector, and b is the bias term.

*3.3.5 Support Vector Machin***e**

Supervised learning techniques such as Support Vector Machines (SVMs) are useful for regression and classification tasks. Its primary function is to find the hyperplane that divides the feature space into classes. SVMs aim to maximize the margin between support vectors (data points from different classes) to optimize their separation. Several domains use SVMs, including intrusion detection, pattern recognition, image classification, bioinformatics, and text classification. SVMs excel in problems with complex decision boundaries (Chio & Freeman, 2018; Mello & Ponti, 2018).

The decision function for a linear support vector machine is given by (8).

$$f(x) = \text{sign}(w \cdot x + b) \quad (8)$$

where w is the weight vector, x is the feature vector, and b is the bias term. The objective is to maximize the margin $\frac{2}{\|w\|}$ subject to the constraint that all data points are correctly classified [69]. The sign function outputs the sign of its input.

The sign function $\text{sign}(z) = \{+1, if\ z > 0;\ 0, if\ z = 0;\ -1, if\ z < 0.$

3.3.6 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a popular ensemble learning algorithm. XGBoost utilizes machine learning algorithms within the Gradient Boosting framework. It is a very portable, adaptable, and efficient distributed gradient boosting library that has been optimized. Due to its reputation for

speed and efficiency, XGBoost is widely used with structured/tabular data and is commonly employed in many domains that work with high-dimensional data (Chen et al., 2018). Equation (9) shows the prediction of an ensemble of K trees.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{9}$$

where f_k represents the *k*-th tree in the ensemble. Each tree is trained to minimize the objective function given in (10).

$$L = \sum_i l\,(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{10}$$

where l is a differentiable loss function (e.g., squared loss for regression), and $\Omega(f_k)$ is a regularization term to prevent overfitting

3.3.7 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a type of artificial neural network composed of several interconnected layers: an input layer, one or more hidden layers, and an output layer. In this architecture, neurons are connected through weighted links, allowing information to pass from one layer to the next. To form a fully connected structure that enables the network to learn complex patterns in the data, each neuron in each layer typically connects to every neuron in the following layer. MLPs can model complex data interactions since they employ non-linear activation functions (Mello & Ponti, 2018). Equation (11) shows the output of a neuron in an MLP.

$$h_j = \phi\left(\sum_i w_{ij} x_i + b_j\right) \tag{11}$$

where h_j is the activation of the neuron, w_{ij} are the weights, x_i are the inputs, b_j is the bias term, and ϕ is the activation function (e.g., sigmoid, tanh, ReLU). The final output layer typically uses a softmax function for classification, as shown in (12).

$$P(x) = \frac{e^{z_k}}{\sum_j e^{z_j}} \tag{12}$$

where z_k is the weighted sum of inputs to the *k*-th output neuron (Goodfellow et al., 2016).

3.4 Model Evaluation

The model evaluation stage assesses the PFS's performance using metrics derived from the Confusion Matrix. These metrics include accuracy, precision, recall, F1-score, False Positive Rate (FPR), and False Negative Rate (FNR). Here, the inference time (IT) and the model training time (MTT) are also computed. The evaluation is carried out using cross-validation, which repeatedly splits the data into training and test sets and averages performance across the splits. To evaluate the performance of the PFS, we can compare how well models constructed using all features from the UNSW-NB15 and NSL-KDD datasets perform to those selected using the PFS. Additionally, the performance of an existing feature selection approach (CFS) is compared to that of PFS.

Also measured are the inference time (IT) and the model training time (MTT). How long it took to train the ML model on the dataset is shown by MTT. It demonstrates how well the training process uses computational resources. It is preferable to have a lower MTT for quicker model building. Furthermore, the time it takes for the trained model to generate predictions using unseen data is known as Inference Time, and it is also assessed in this study. Rapid response times are essential to reduce attacks in the ever-changing cybersecurity landscape, and IT plays a key role in real-time intrusion detection.

3.4.1 Cross Validation

One of the statistical methods used to test a model's ability to generalize is cross-validation (CV). This method involves splitting the data into distinct subsets, training the model on some of them, and evaluating its performance on the remaining subsets. This is depicted in Figure 4. Using multiple train-test splits provides a reliable measure of how the model will perform on unseen data (Talukder *et al.*, 2024). CVs are of different types, including k-fold, leave-one-out (LOOCV), and stratified k-fold.

As illustrated in Figure 4, the model is trained k times, with each fold serving as the test set exactly once. LOOCV is a special case of K-fold, where k equals the total number of data points. In LOOCV, each data point is used once as the test set, while the model is trained on all the remaining $n - 1$ points.

To maintain class proportions in classification tasks, Stratified k-Fold CV partitions the data into k folds so that the target variable is equally distributed across all folds (Talukder et al., 2023). CV makes better use of scarce data by estimating model performance more accurately than a single train-test split and by ensuring that every data point is used for both training and testing. On the other hand, it requires repeated model training, which can be computationally costly for complex models and large datasets (James *et al.*, 2015).

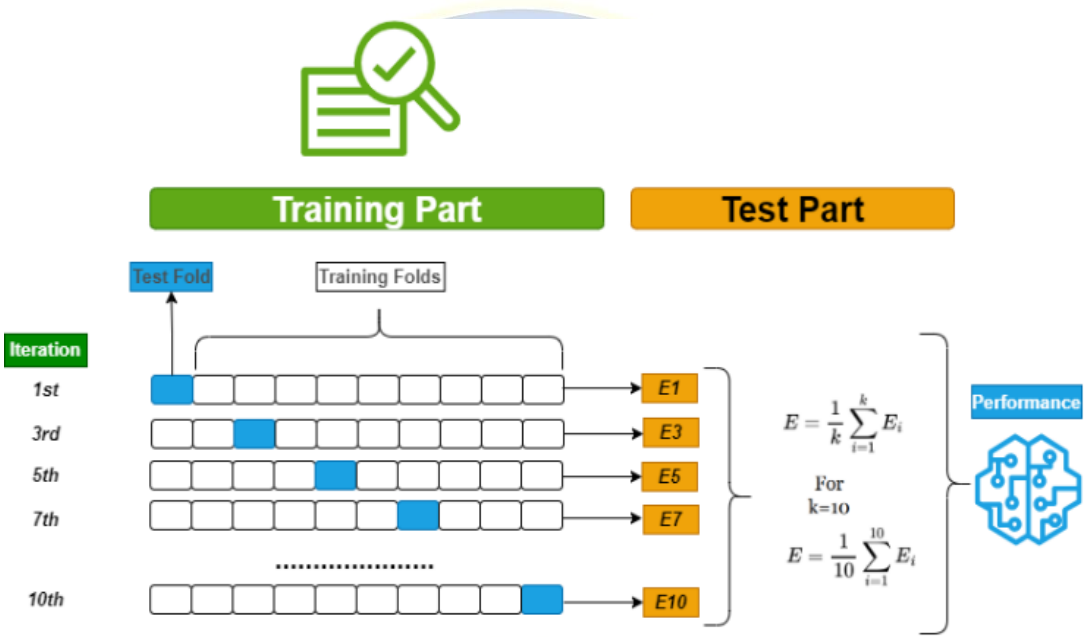

Figure 4. 10-Fold Cross Validation (Zhang *et al.*, 2016).

3.4.2 Confusion Matrix

The performance of an algorithm can be visualized in a tabular form using a confusion matrix. The confusion matrix is a 2x2 table that summarizes the performance of a classification algorithm for a binary classification problem, which is common in NIDS (e.g., distinguishing between normal and malicious traffic). The matrix can also be used for multi-class classification (Thomas et al., 2020).

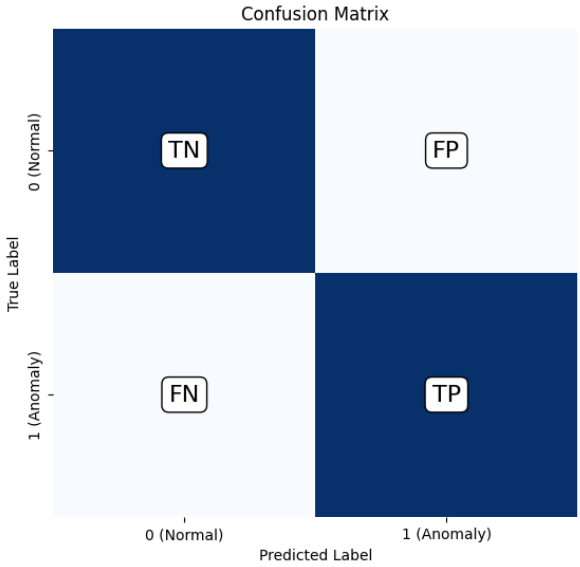

Figure 5. Confusion matrix

For reference, the confusion matrices depicted in this work contain four distinct combinations of true and predicted labels, distributed across their rows and columns. Each row corresponds to the true/actual labels of the instances in the dataset, and each column corresponds to the predicted labels assigned by the model, as captured in Figure 5. The row labeled 1 corresponds to the true label of "attack" instances, and the row labeled 0 corresponds to the true label of "normal" instances. The column labeled 0 corresponds to the predicted label of "normal" instances, and the column labeled 1 corresponds to the predicted label of "attack" instances. Therefore, each of the four (4) cells in the matrix represents the intersection between the actual and predicted labels, distributed across its rows and columns as follows:

i. **True Negatives (TN):** Instances that are negative (normal) and predicted as negative. These are in the top-left cell (row 1, column 1).
ii. **False Positives (FP):** Instances that are negative (normal) but predicted as positive (anomalies). These are in the top-right cell (row 1, column 2).
iii. **False Negatives (FN):** Instances that are positive (anomalies) but predicted as negative (normal). These are in the bottom-left cell (row 2, column 1).
iv. **True Positives (TP):** Instances that are positive (anomalies) and predicted as positive. These are in the bottom-right cell (row 2, column 2).

The four (4) cells can be used to compute various performance metrics as captured below. Note that the values range from 0 to 1 (or 0% to 100%). For some metrics, the higher the better, while for others, the lower the better (Acharya & Singh, 2017; Murphy, 2012)..

a) **Accuracy:** The ratio of correctly predicted instances to the total number of instances (both normal and attack). It provides an overall measure of the performance of the model but can be misleading if the dataset is imbalanced. It is computed using the in (13). A higher proportion of accurate predictions is indicated by a higher value, which is better.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

b) **Precision:** This shows the percentage of positive occurrences that were accurately predicted relative to the total number of positive instances predicted. It demonstrates the model's efficacy in reducing false alarms by showing how many of the detected intrusions are intrusions. Eq. (14) provides the formula for computing precision from the confusion matrix. In this case, higher values indicate fewer false positives, which is also better. It indicates that the system is more likely to be right when it detects an intrusion.

$$Precision = \frac{TP}{TP + FP} \quad (14)$$

c) Recall (Sensitivity or True Positive Rate): This is the percentage of positive instances correctly predicted relative to all positive instances. By evaluating the model's sensitivity, we assess its ability to detect all actual intrusions, thereby indicating how well it reduces false positives. Eq. (15) provides the formula for its computation. A lower number of missed intrusions indicates a higher recall value, which is more desirable.

$$Recall = \frac{TP}{TP + FN} \quad (15)$$

d) F1 Score: The idea here is to provide a single measure that shows the balance between precision and recall. It is the harmonic mean of Precision and Recall, which is useful when balancing them is needed, especially with an imbalanced dataset. Higher values are also preferable in this case. Equation (16) provides the formula for its computation.

$$F1\ Score = 2 \cdot \frac{Precision\ \cdot Recall}{Precision + Recall} \quad (16)$$

e) False Positive Rate (FPR): Here, we can compute the ratio of falsely predicted positive instances to all actual negative instances using the in (17). It shows the possibility that NIDS may detect normal traffic as an intrusion, reducing the system's usability by generating false alarms. For FPR, lower values indicate fewer false alarms.

$$FPR = \frac{FP}{FP + TN} \quad (17)$$

f) False Negative Rate (FNR): Here we can compute the ratio of falsely predicted negative instances to all actual positive instances using the in (18). It measures the likelihood that intrusions will go undetected by NIDS, thereby degrading the system's security effectiveness. For FNR, lower values are better, indicating that fewer intrusions are missed by the system.

$$FNR = \frac{FN}{FN + TP} \quad (18)$$

4. Results

The experimental results using the NSL-KDD and UNSW-NB15 datasets are presented in this section. We evaluated the performance of seven Machine Learning models for Network Intrusion Detection System in our experiment: RF, MLP, KNN, LR, NB, SVM, and XGBoost. We focused on evaluating key performance metrics, including Model Training Time and Inference Time, by considering 1. All the features from each of the UNSW-NB15 and NSL-KDD datasets, 2. The Features selected by our developed feature selection algorithm from each of the UNSW-NB15 and NSL-KDD datasets, and 3. The Features selected by another existing feature selection algorithm (CFS) from each of the UNSW-NB15 and NSL-KDD datasets. The results in this section are presented based on the performance evaluation of both datasets used in this study: UNSW-NB15 and NSL-KDD.

4.1 Performance Evaluation Results for Unsw-Nb15 and NSL-KDD Datasets

This section presents the results of seven (7) NIDS models trained using all the features (NO FS) of the UNSW-NB15 dataset, as well as features selected using the proposed feature selection (PFS) technique and the correlation-based feature selection (CFS) method. Key performance metrics, such as precision, accuracy, F1-score, recall, model training time, and inference time, are also presented.

Table 1. Performance Evaluation Results of NIDS models on the UNSW-NB15 dataset

Model	Accuracy			Precision			Recall			F1-score		
	NO FS	PFS	CFS	NO FS	PFS	CFS	NO FS	PFS	CFS	NO FS	PFS	CFS
KNN	0.92	0.83	0.92	0.98	0.88	0.93	0.88	0.81	0.91	0.93	0.84	0.92
NB	0.65	0.75	0.75	1.00	0.84	0.89	0.36	0.68	0.63	0.53	0.75	0.74
SVM	0.92	0.76	0.89	0.96	0.75	0.91	0.88	0.85	0.88	0.92	0.80	0.90
XGB	0.93	0.93	0.96	0.96	0.96	0.96	0.92	0.92	0.96	0.94	0.94	0.96
LR	0.91	0.76	0.85	0.95	0.75	0.88	0.87	0.85	0.85	0.91	0.80	0.86
RF	0.94	0.94	0.96	0.96	0.96	0.96	0.92	0.92	0.96	0.94	0.94	0.96
MLP	0.96	0.87	0.95	0.97	0.93	0.94	0.96	0.83	0.96	0.97	0.88	0.95

Table 2: The results of evaluating the accuracy of NIDS models based on the NSL-KDD dataset

Model	Accuracy			Precision			Recall			F1-score		
	NO FS	PFS	CFS	NO FS	PFS	CFS	NO FS	PFS	CFS	NO FS	PFS	CFS
KNN	0.99	0.96	0.98	0.99	0.94	0.98	0.99	0.97	0.98	0.99	0.96	0.98
NB	0.81	0.82	0.87	0.99	0.75	0.89	0.62	0.94	0.83	0.76	0.84	0.86
SVM	0.96	0.83	0.96	0.97	0.77	0.97	0.94	0.92	0.96	0.96	0.84	0.96
XGB	1.00	0.98	0.98	1.00	0.96	0.98	1.00	0.99	0.99	1.00	0.98	0.98
LR	0.96	0.82	0.92	0.96	0.77	0.93	0.95	0.89	0.92	0.96	0.83	0.92
RF	1.00	0.94	0.98	1.00	0.97	0.98	0.99	0.91	0.98	0.99	0.94	0.98
MLP	0.99	0.92	0.98	0.99	0.96	0.98	0.99	0.86	0.98	0.99	0.91	0.98

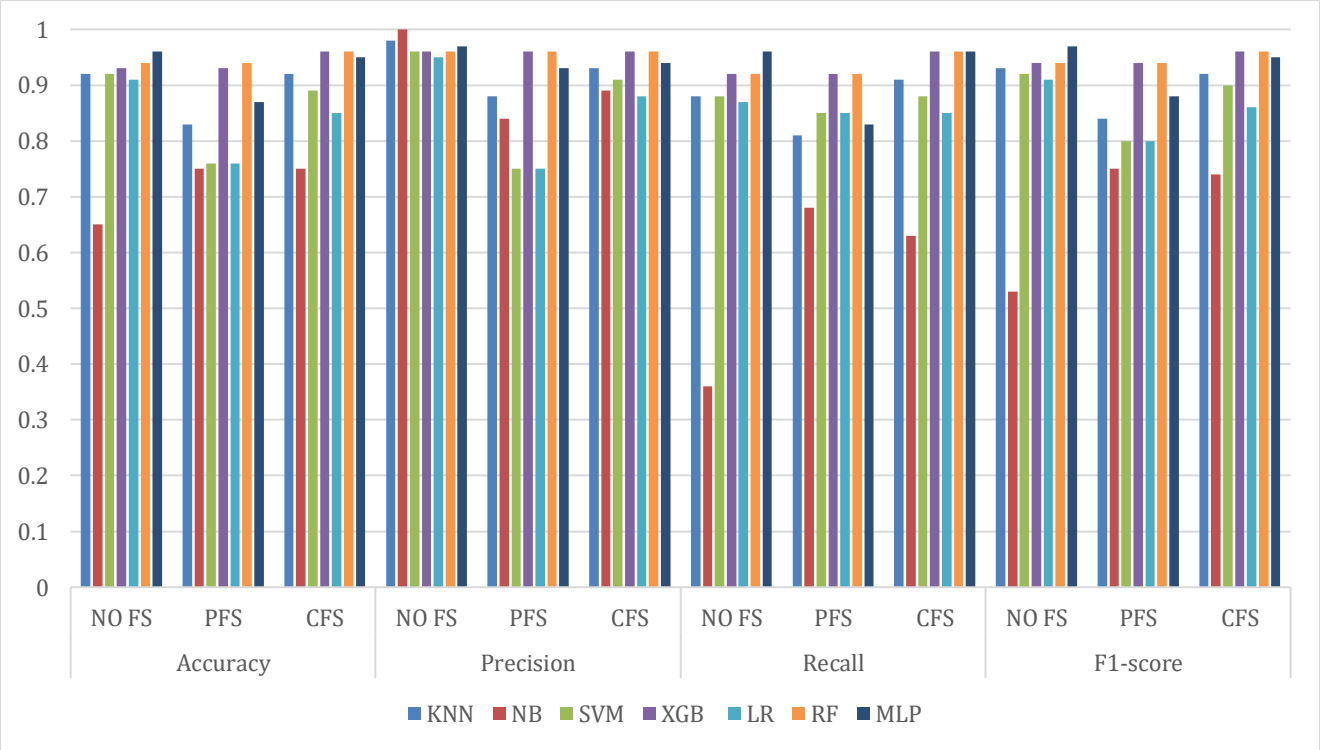

Figure 6: Clustered Column Chart depicting the performance – measured using various metrics - of the 7 models built using "NO FS", "PFS", and "CFS" on the UNSW-NB15 dataset.

Figure 7: Clustered Column Chart depicting the performance – measured using various metrics - of the 7 models built using "NO FS", "PFS", and "CFS" on the NSL-KDD dataset.

Table 3: Model Training Time and Inference Time of 7 NIDS models on the UNSW-NB15 dataset

Model	MTT (s)			IT (s)		
	NO FS	PFS	CFS	NO FS	PFS	CFS
KNN	0.12	0.03	0.04	19.99	13.90	11.79
NB	0.34	0.07	0.06	0.08	0.02	0.00
SVM	268.34	209.74	170.49	59.62	22.81	89.12
XGB	4.62	1.01	1.34	0.12	0.05	0.05
LR	2.26	0.58	0.88	0.02	0.00	0.00
RF	18.55	17.21	15.73	0.61	0.47	0.59
MLP	321.62	114.36	133.89	0.18	0.09	0.09

Table 4: Model Training Time and Inference Time of 7 NIDS models on the NSL-KDD dataset

Model	MTT (s)			IT (s)		
	NO FS	PFS	CFS	NO FS	PFS	CFS
KNN	0.11	0.04	0.04	31.90	23.08	27.62
NB	0.22	0.06	0.06	0.06	0.03	0.02
SVM	328.35	615.20	139.62	60.64	35.52	63.29
XGB	5.32	0.89	1.41	0.17	0.08	0.08
LR	3.23	0.88	0.93	0.04	0.00	0.00
RF	20.61	10.12	11.53	1.35	0.67	0.82
MLP	236.45	126.52	106.89	0.08	0.09	0.05

5. Discussion

This section discusses the implications of the findings of this work in the context of the research objectives. The results obtained demonstrate the high accuracy of the proposed feature selection method and its effect on NIDS developed using features from the UNSW-NB15 and NSL-KDD datasets.

5.1 Discussion of Performance Evaluation Results of UNSW-NB15 Dataset

In the context of network intrusion detection, it is essential to achieve high accuracy, precision, recall, and F1-score to reliably identify hostile network activity while minimizing false positives and false negatives. PFS, as depicted in Figure 7 and Table 1 exhibited substantial improvements in several essential metrics across various models in comparison to the "NO FS" and CFS baseline. This demonstrates a significant decrease in false negatives (improved recall) and a more optimal trade-off between precision and recall (higher F1-score).

RF model achieved an accuracy of 94%, indicating that about 94% of all instances in the test set are correctly classified as either "attack" or "normal" with all the features being used, as well as when PFS was used. The model achieved 96% accuracy with CFS. While accuracy is an important metric, it may not provide a complete picture, especially in imbalanced datasets where one class is predominant. Therefore, in actual NIDS settings, accuracy alone might not be sufficient to evaluate the model's effectiveness.

Precision measures the model's ability to correctly identify "attack" instances without misclassifying too many "normal" instances as "attack." The NB model achieved 100% precision across all features (Table 1). This shows that all instances predicted as "attack" are indeed "attack". High precision means that when the model predicts an instance as "attack," it is very likely to be correct. High precision is necessary in NIDS; it enables the system to minimize false alarms, thereby reducing the workload of security analysts in determining whether an event is an actual attack.

Recall measures the model's ability to capture all actual "attack" instances without missing too many. As shown in Table 1, the RF model achieved a recall of 92%, indicating that it correctly identifies about 92% of all actual "attack" instances when using "NO FS' and "PFS". F1 score provides a balanced measure of a model's overall performance by taking the harmonic mean of precision and recall. The RF model in this case achieved an F1 score of 94%, reflecting a balance between precision and recall. When F1 score is high, it means that recall and precision are well balanced. This balance is particularly useful in situations of uneven class distribution or when both false positives and false negatives are equally important, as in NIDS.

The model training time of the NB model reduced drastically from 0.34 Seconds (for the "NO FS" scenario) to 0.07 seconds (in the "PFS" scenario), and the inference time from 0.08 seconds (in the "NO FS" scenario) to 0.02 seconds (in the "PFS" scenario), which shows a significant improvement as depicted in Table 3.

Similarly, the other models mostly demonstrated a comparable performance. The XGB model, for instance, has a similar accuracy (0.93), precision (0.96), recall (0.92), and F1-Score (0.94) to the "PFS". The PFS demonstrated a significant decrease in Model Training Time from 4.62 seconds down to 1.01 seconds. The RF model also demonstrated comparable performance to the PFS, with a slight decrease in Model Training and Inference times. This implies the potential for zero false positives and false negatives, which is highly desirable in network security applications.

In addition, the PFS also led to faster training times for most models compared to the "NO FS" situation, with reductions ranging from a few seconds (e.g., XGB, LR, and RF) to minutes (e.g., SVM and MLP) as shown in Table 3.

Remarkably, the CFS method exhibited enhancements above the "NO FS" baseline in numerous instances, as can be gathered from Table 1. However, the PFS generally outperformed CFS or achieved results comparable to CFS on the UNSW-NB15 dataset. This indicates that the PFS is more appropriate for selecting the most distinguishing features that are relevant to network intrusion detection.

5.2 Discussion of Performance Evaluation Results of NSL-KDD Dataset

The PFS, as shown in Figure 8 and Table 3, exhibited substantial improvements in several essential metrics across various models in comparison to the "NO FS" baseline. For the NSL-KDD dataset, the NB model built using the features selected by the PFS achieved a slightly higher accuracy of 0.82, compared to 0.81 in the "NO FS" scenario. This was achieved even though the MTT recorded for the "PFS" is about one-quarter of the training time taken by "NO FS," and the inference time is half of the time taken by the "NO FS". These are depicted in Table 4 and Figure 8.

The XGB model achieved an accuracy of 0.98 with the "PFS" technique, which is remarkably high and comparable to its performance with "NO FS" and "CFS," as shown in Table 3. The precision for "PFS" is also excellent at 0.96, indicating a low false-positive rate. The recall score of 0.99 also suggests that the XGB model with "PFS" correctly identified almost all positive instances (attacks). Consequently, the F1-score of 0.98 demonstrates a good balance between precision and recall.

Similarly, the radio frequency model demonstrated high performance when using the PFS method, achieving an accuracy of 0.94 and 0.97. The recall rate of 0.98 indicates that the model detected most attacks, resulting in an F1 score of 0.94, as shown in Figure 4.21. Both the XGB and RF models had low values of FPRs (0.03 and 0.09, respectively) and FNRs (0.01 and 0.01, respectively) compared to PFS, which once again highlights their reliability in minimizing false alarms and missed detections.

While the SVM model's performance with the PFS is lower than with NO FS and CFS, with an accuracy of 0.83, precision of 0.77, recall of 0.92, and F1-score of 0.84, it is important to note that the variations in performance compared to other feature selection techniques should be expected in some algorithms and do not necessarily diminish the significance of the "PFS" approach.

In addition to the other performance metrics, the time complexity of the classification models and feature selection techniques is a crucial factor to consider when implementing NIDS in real-world scenarios. The PFS appears to offer notable computational efficiency gains, making it a promising approach for real-time or resource-constrained NIDS deployments.

As shown in Table 4, the XGB, MLP, and RF models consistently outperformed other models, with the PFS technique demonstrating remarkably low model training times (MTT) and inference times (IT). Specifically, the XGB model with PFS had an MTT of 0.89 seconds and an IT of 27.62 seconds, while the XGB model with NO FS had an MTT of 5.32 seconds and IT of 0.17 seconds. The RF model with PFS exhibited an MTT of 10.12 seconds and an IT of 0.67 seconds, while the RF model with NO FS had an MTT of 20.61 seconds and IT of 1.35 seconds. The MLP model with PFS exhibited an MTT of 126.52 seconds and an IT of 0.09 seconds, while the MLP model with NO FS had an MTT of 236.45 seconds and IT of 0.08 seconds. Compared to other models like SVM (with PFS), with significantly higher MTT (615.2 seconds) and a lower IT (35.52 seconds), the low computational times obtained here are impressive.

6. Conclusion

This study proposed an Enhanced Multi-Metric Feature Selection Technique — integrating variance analysis with k-means clustering — to improve the performance of Network Intrusion Detection Systems. The proposed approach addresses the challenges of irrelevant and noisy features by identifying the most pertinent features in the dataset, resulting in improved classification performance, computational efficiency, and adaptability compared to existing methods.

The technique demonstrated exceptional inference and training times across evaluated models, consistently outperforming the baseline (no feature selection) and achieving comparable or superior results to correlation-based feature selection (CFS). Notably, the combination of the proposed technique with XGB and RF models yielded the most significant computational gains, enabling timely detection and response to potential threats while optimizing the use of available hardware resources. These characteristics make the approach particularly well-suited for real-time intrusion detection and resource-constrained NIDS deployments, such as embedded systems and edge devices.

Overall, the proposed feature selection technique enhances the reliability and efficiency of NIDS pipelines, providing a promising foundation for scalable and lightweight network security solutions. Future work should explore its time complexity in high-dimensional feature spaces and its integration with ensemble models to further extend its applicability across diverse NIDS implementations.

Using the CICIDS2017 dataset, which was created for IDS research, Alfrhan et al. (2020) conducted experiments with three classifiers – random forest, naive Bayes, and k-nearest neighbors (K-NN) – on the entire unbalanced dataset. After that, the authors used the SMOTE method to compare the results of balanced and unbalanced datasets. Based on the overall performance of the Decision Forest, SVM, and Decision Jungle classifiers, applying the SMOTE approach improved performance. However, neither Feature Selection nor normalization techniques were reported by the authors during preprocessing.

Author Contributions: Conceptualization, U.A.B.; methodology, U.A.B.; formal analysis, U.A.B.; writing—original draft preparation, U.A.B.; writing—review and editing, U.A.B. and M.L; supervision, E.J.G and A. S. A.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets supporting the conclusions of this article are available in the UNSW repository found at https://research.unsw.edu.au/projects/unsw-nb15-dataset and Kaggle repository found at https://www.kaggle.com/datasets/hassan06/nslkdd

Conflicts of Interest: The authors declare no conflicts of interest.

Acknowledgements: Not applicable

References

Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., & Abuzneid, A. (2019). Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. *Electronics*, *8*(3). https://doi.org/10.3390/electronics8030322

Acharya, N., & Singh, S. (2017). An IWD-based feature selection method for intrusion detection system. *Soft Computing*, *22*(13), 4407-4416. https://doi.org/10.1007/s00500-017-2635-2

Adenusi, D. A., Oladimeji, O. O., Oyekola, T. A., & Olagunju, K. S. (2025). Data-driven network intrusion detection using optimized machine learning algorithms. *Franklin Open*, *12*, 100339. https://doi.org/https://doi.org/10.1016/j.fraope.2025.100339

Akashdeep, Manzoor, I., & Kumar, N. (2017). A feature reduced intrusion detection system using ANN classifier. *Expert Systems with Applications*, *88*, 249-257. https://doi.org/10.1016/j.eswa.2017.07.005

Al-Turaiki, I., & Altwaijry, N. (2021). A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection. *Big Data*, *9*(3), 233-252. https://doi.org/10.1089/big.2020.0263

Alam, K., Monir, M. F., Hossain, M. J., Uddin, M. S., & Habib, M. T. (2025). Adaptive Defense: Zero-Day Attack Detection in NIDS With Deep Reinforcement Learning. *IEEE Access*, *13*, 116345-116361. https://doi.org/10.1109/ACCESS.2025.3585445

Alfrhan, A. A., Alhusain, R. H., & Khan, R. U. (2020, 9-10 Sept. 2020). SMOTE: Class Imbalance Problem In Intrusion Detection System. 2020 International Conference on Computing and Information Technology (ICCIT-1441),

Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, *25*, 152-160. https://doi.org/10.1016/j.jocs.2017.03.006

Aljehane, N. O., Mengash, H. A., Hassine, S. B. H., Alotaibi, F. A., Salama, A. S., & Abdelbagi, S. (2024). Optimizing intrusion detection using intelligent feature selection with machine learning model. *Alexandria Engineering Journal*, *91*(February), 39-49. https://doi.org/10.1016/j.aej.2024.01.073

Almasoudy, F. H., Al-Yaseen, W. L., & Idrees, A. K. (2020). Differential Evolution Wrapper Feature Selection for Intrusion Detection System. *Procedia Computer Science*, *167*, 1230-1239. https://doi.org/https://doi.org/10.1016/j.procs.2020.03.438

Alsaleh, A., & Binsaeedan, W. (2021). The Influence of Salp Swarm Algorithm-Based Feature Selection on Network Anomaly Intrusion Detection. *IEEE Access*, *9*, 112466-112477. https://doi.org/10.1109/access.2021.3102095

Ashiku, L., & Dagli, C. (2021). Network Intrusion Detection System using Deep Learning. *Procedia Computer Science*, *185*, 239-247. https://doi.org/https://doi.org/10.1016/j.procs.2021.05.025

Baba, U. A., Nsang, A. S., & Adeseye, O. (2018). Three novel approaches to dimensionality reduction. 2018 World Congress in Computer Science, Computer Engineering and Applied Computing, CSCE 2018 - Proceedings of the 2018 International Conference on Artificial Intelligence, ICAI 2018,

Bansal, A., & Kaur, S. (2019). Data dimensionality reduction (DDR) scheme for intrusion detection system using ensemble and standalone classifiers. Communications in Computer and Information Science,

Bhati, B. S., Rai, C. S., Balamurugan, B., & Al-Turjman, F. (2020). An intrusion detection scheme based on the ensemble of discriminant classifiers. *Computers & Electrical Engineering*, *86*. https://doi.org/10.1016/j.compeleceng.2020.106742

Bojarajulu, B., Tanwar, S., & Rana, A. (2021). *A Synoptic Review on Feature Selection and Machine Learning models used for Detecting Cyber Attacks in IoT* 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON),

Bonaccorso, G. (2017). *Machine Learning Algorithms*. Packt Publishing.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153-1176. https://doi.org/10.1109/comst.2015.2494502

Chaudhary, S., Gkioulos, V., & Katsikas, S. (2023). A quest for research and knowledge gaps in cybersecurity awareness for small and medium-sized enterprises. *Computer Science Review*, *50*. https://doi.org/10.1016/j.cosrev.2023.100592

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321-357. https://doi.org/10.1613/jair.953

Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., & Peng, J. (2018). *XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud* 2018 IEEE International Conference on Big Data and Smart Computing (BigComp),

Chinnasamy, R., Subramanian, M., Easwaramoorthy, S. V., & Cho, J. (2025). Deep learning-driven methods for network-based intrusion detection systems: A systematic review. *ICT Express*, *11*(1), 181-215. https://doi.org/https://doi.org/10.1016/j.icte.2025.01.005

Chio, C., & Freeman, D. (2018). *Machine Learning and Security Protecting Systems with Data and Algorithms*. O'REILLY.

Cisco. (2020). *Cisco Annual Internet Report (2018 - 2023)*. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21-27. https://doi.org/10.1109/TIT.1967.1053964

Damtew, Y. G., Chen, H., & Yuan, Z. (2023). Heterogeneous Ensemble Feature Selection for Network Intrusion Detection System. *International Journal of Computational Intelligence Systems*, *16*(1), 9. https://doi.org/10.1007/s44196-022-00174-6

De Amorim, L. B. V., Cavalcanti, G. D. C., & Cruz, R. M. O. (2023). The choice of scaling technique matters for classification performance. *Applied Soft Computing*, *133*, 109924. https://doi.org/10.1016/j.asoc.2022.109924

Dubey, G. P., & Bhujade, D. R. K. (2021). Optimal feature selection for machine learning based intrusion detection system by exploiting attribute dependence. *Materials Today: Proceedings*, *47*, 6325-6331. https://doi.org/https://doi.org/10.1016/j.matpr.2021.04.643

Fang, J., & Leng, F. (2025). Network Security Intrusion Detection System Based on Deep Learning. *Procedia Computer Science*, *261*, 1107-1113. https://doi.org/https://doi.org/10.1016/j.procs.2025.04.692

Faris, M., Mahmud, M. N., Salleh, M. F. M., & Alsharaa, B. (2023). A differential evolution-based algorithm with maturity extension for feature selection in intrusion detection system. *Alexandria Engineering Journal*, *81*, 178-192. https://doi.org/https://doi.org/10.1016/j.aej.2023.09.032

Gollmann, D. (1999). *Computer security*. John Wiley & Sons, Inc. https://doi.org/https://doi.org/10.1002/wics.106

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.

Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning.* University of Waikato]. http://researchcommons.waikato.ac.nz/

Hande, Y., & Muddana, A. (2020). A Survey on Intrusion Detection System for Software Defined Networks (SDN). *International Journal of Business Data Communications and Networking*, *16*(1), 28-47. https://doi.org/10.4018/ijbdcn.2020010103

Hao, Y., Hou, Y., & Li, L. (2018). A Novel Algorithm for Feature Selection Used in Intrusion Detection. In *Innovative Mobile and Internet Services in Ubiquitous Computing* (pp. 967-974). https://doi.org/10.1007/978-3-319-61542-4_98

Herrera-Semenets, V., Bustio-Martínez, L., Hernández-León, R., & van den Berg, J. (2021). A multi-measure feature selection algorithm for efficacious intrusion detection. *Knowledge-Based Systems*, *227*. https://doi.org/10.1016/j.knosys.2021.107264

Interpol. (2023). *African Cyberthreat Assessment Report: Cyberthreat Trends*. https://cybilportal.org/publications/african-cyberthreat-assessment-report-2023-cyberthreat-trends/

Iwendi, C., Khan, S., Anajemba, J. H., Mittal, M., Alenezi, M., & Alazab, M. (2020). The Use of Ensemble Models for Multiple Class and Binary Class Classification for Improving Intrusion Detection Systems. *Sensors (Basel)*, *20*(9). https://doi.org/10.3390/s20092559

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2015). *An Introduction to Statistical Learning with Applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7

Kamarudin, M. H., Maple, C., & Watson, T. (2019). Hybrid feature selection technique for intrusion detection system. *International Journal of High Performance Computing and Networking*, *13*(2).

Karami, S., Saberi-Movahed, F., Tiwari, P., Marttinen, P., & Vahdati, S. (2023). Unsupervised feature selection based on variance–covariance subspace distance. *Neural Networks*, *166*, 188-203. https://doi.org/https://doi.org/10.1016/j.neunet.2023.06.018

Kaur, R., Gabrijelčič, D., & Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. *Information Fusion*, *97*. https://doi.org/10.1016/j.inffus.2023.101804

Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2005). Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. Proceedings of the third annual conference on privacy, security and trust,

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1). https://doi.org/10.1186/s42400-019-0038-7

Kim, J., Kim, J., Kim, H., Shim, M., & Choi, E. (2020). CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. *Electronics*, *9*(6). https://doi.org/10.3390/electronics9060916

Lifandali, O., Abghour, N., & Chiba, Z. (2023). Feature Selection Using a Combination of Ant Colony Optimization and Random Forest Algorithms Applied To Isolation Forest Based Intrusion Detection System. *Procedia Computer Science*, *220*, 796-805. https://doi.org/https://doi.org/10.1016/j.procs.2023.03.106

Mahmoud, L., Liyanage, M., Singla, J., & Gangopadhyay, S. (2025). DSEM-NIDS: Enhanced Network Intrusion Detection System Using Deep Stacking Ensemble Model. *IEEE Open Journal of the Computer Society*, *6*, 955-967. https://doi.org/10.1109/OJCS.2025.3581036

Malhotra, H., & Sharma, P. (2019). Intrusion Detection using Machine Learning and Feature Selection. *International Journal of Computer Network and Information Security*, *11*(4), 43-52. https://doi.org/10.5815/ijcnis.2019.04.06

Mello, R. F. d., & Ponti, M. A. (2018). *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer International Publishing. https://doi.org/https://doi.org/10.1007/978-3-319-94989-5

Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., & Karimipour, H. (2019). Cyber intrusion detection by combined feature selection algorithm. *Journal of Information Security and Applications*, *44*, 80-88. https://doi.org/10.1016/j.jisa.2018.11.007

Mohammed, A., Ribadu, M. B., Tukur, A., & Baba, U. A. (2023). Internet of Things (IoT): A Mechanism for Enhancing Teaching and e-Learning in Higher Educational Institutes (HEIs). *Nigerian Journal of Accounting and Finance*, *15*(1), 175-183.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Nimbalkar, P., & Kshirsagar, D. (2021). Feature selection for intrusion detection system in Internet-of-Things (IoT). *ICT Express*, *7*(2), 177-181. https://doi.org/10.1016/j.icte.2021.04.012

Nsang, A. S. (2011). *An Empirical Study of Novel Approaches to Dimensionality Reduction and Applications* University of Cincinnati]. Cincinnati.

Nsang, A. S., & Baba, U. A. (2018). Investigating the relative performances of assorted dimensionality reduction techniques. 2018 World Congress in Computer Science, Computer Engineering and Applied Computing, CSCE 2018 - Proceedings of the 2018 International Conference on Artificial Intelligence, ICAI 2018,

Nsang, A. S., Edi, D., & Ahanonu, C. (2015). Query-Based Dimensionality Reduction Applied to Images. International Conference on Advances in Big Data Analytics, ABDA'15,

Park, N.-E., Lee, Y.-R., Joo, S., Kim, S.-Y., Kim, S.-H., Park, J.-Y., Kim, S.-Y., & Lee, I.-G. (2023). Performance evaluation of a fast and efficient intrusion detection framework for advanced persistent threat-based cyberattacks. *Computers and Electrical Engineering*, *105*. https://doi.org/10.1016/j.compeleceng.2022.108548

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (Third ed.). Packt Publishing.

Rhys, H. I. (2020). *Machine Learning with R, the tidyverse, and mlr* (First ed.). Manning.

Shandilya, S. K., Choi, B. J., Kumar, A., & Upadhyay, S. (2023). Modified Firefly Optimization Algorithm-Based IDS for Nature-Inspired Cybersecurity. *Processes*, *11*(3). https://doi.org/10.3390/pr11030715

Singh, D. T. S. M. M., & Keikhosrokiani, P. (2023). A systematic literature review for APT detection and Effective Cyber Situational Awareness (ECSA) conceptual model. *Heliyon*, *9*(7), 1-28. https://doi.org/https://doi.org/10.1016/j.heliyon.2023.e17156

Sumaiya Thaseen, I., & Aswani Kumar, C. (2017). Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University - Computer and Information Sciences*, *29*(4), 462-472. https://doi.org/10.1016/j.jksuci.2015.12.004

Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., Alharbi, F., & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, *72*. https://doi.org/10.1016/j.jisa.2022.103405

Talukder, M. A., Islam, M. M., Uddin, M. A., Hasan, K. F., Sharmin, S., Alyami, S. A., & Moni, M. A. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00886-w

Tamy, S., Belhadaoui, H., Rabbah, N., & Rifi, M. (2021). Ensemble learning based feature reduction and selection methods for network intrusion detection system. *Journal of Theoretical and Applied Information Technology*, *99*(13).

Thomas, T., Vijayaraghavan, A. P., & Emmanuel, S. (2020). *Machine Learning Approaches in Cyber Security Analytics*. Springer. https://doi.org/https://doi.org/10.1007/978-981-15-1706-8

Whitman, M. E., & Mattord, H. J. (2019). *Management of Information Security* (6th ed.). Cengage.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67-82. https://doi.org/10.1109/4235.585893

Wu, C., & Li, W. (2021). Enhancing intrusion detection with feature selection and neural network. *International Journal of Intelligent Systems*, *36*(7), 3087-3105. https://doi.org/10.1002/int.22397

Xu, Y. (2020). *Application Research Based on Machine Learning in Network Privacy Security* 2020 International Conference on Computer Information and Big Data Applications (CIBDA), Guiyang, China. https://ieeexplore.ieee.org/document/9148393/

Zhang, F., Chan, P. P. K., Biggio, B., Yeung, D. S., & Roli, F. (2016). Adversarial Feature Selection Against Evasion Attacks. *IEEE Transactions on Cybernetics*, *46*(3), 766-777. https://doi.org/10.1109/tcyb.2015.2415032

Zhang, H., Ge, L., Zhang, G., Fan, J., Li, D., & Xu, C. (2023). A two-stage intrusion detection method based on light gradient boosting machine and autoencoder. *Math Biosci Eng*, *20*(4), 6966-6992. https://doi.org/10.3934/mbe.2023301

Zong, W., Chow, Y.-W., & Susilo, W. (2019). Dimensionality Reduction and Visualization of Network Intrusion Detection Data. In *Information Security and Privacy* (pp. 441-455). Springer International Publishing. https://doi.org/10.1007/978-3-030-21548-4_24