Intrusion Detection Using Machine Learning and Deep Learning

Fatima Jobran ALzaher, Asma AlJarullah
Department of Informatics and Computer Systems, King Khalid University, 61421, Alfara, Abha, Saudi Arabia

***Abstract*—As cyberattacks grow in prevalence, Intrusion Detection Systems (IDS) have become critical for securing network infrastructures. This study proposes an efficient IDS framework utilizing both machine learning (ML) and deep learning (DL) algorithms. The framework is evaluated on the "NF-UNSW-NB15-v2" dataset, which comprises a blend of normal and malicious traffic. A diverse set of advanced models—including Deep Neural Networks (DNN), Long Short-Term Memory (LSTM) networks, eXtreme Gradient Boosting (XGBoost), Random Forest (RF), and K-Nearest Neighbors (KNN)—is deployed for intrusion detection. The approach encompasses both binary classification (normal vs. malicious) and multi-class classification (specific attack categories). Preprocessing steps include feature standardization using StandardScaler, class imbalance correction via SMOTE, and dimensionality reduction through Principal Component Analysis (PCA). Results show that Random Forest and XGBoost models achieve high accuracy in binary classification with F1-scores approaching 0.97, while XGBoost attains the best macro F1-score (0.71) in multi-class tasks. Additionally, RF and XGBoost demonstrate the fastest inference times, underscoring their suitability for real-time deployment. This work contributes a scalable and optimized IDS pipeline for enhancing cybersecurity resilience.**

***Keywords*—Cybersecurity; cyber-attack; intrusion detection system; machine learning; deep learning**

I. Introduction

In recent years, the cyber world witnessed the most phenomenal increase ever of cyber threats that targeted individuals, businesses, and governments. The cost of cybercrime at the global level is anticipated at $9.5 trillion by 2024, which amounts to $26 billion per day or $18 million per minute [1]. The growing rate of cybercrime emphasizes the need for proper cybersecurity controls.

Small and medium-sized businesses are most vulnerable with 69% of these experiencing at least one cyber-attack within the past year [1]. Yet 80% of the SMBs are largely unimplemented with the utilization of Privileged Access Management solutions while fewer than 60% of the enterprises are using vital cybersecurity practices of password managers, two-factor authentication, and cybersecurity education [1]. The shift toward remote work amplified the fear of security since 72% of businesses are concerned about the danger it presents and 80% of cybersecurity professionals confirm that the danger increased since 2020 [1].

Phishing and ransomware attacks grew more complex and more numerous. Security professionals saw 62% more phishing assaults within the recent years with 79% of account takeover attacks originating from the type of phishing attack that occurs with the use of phish emails [1]. The ransomware attacks grew with 70% of the attacks focusing on the small business market and the number of ransomware teams actively present more than doubling year over year [1].

Adversaries now leverage automation and AI to accelerate reconnaissance, weaponization, and evasion, fueling an arms race that compels defenders to adopt more adaptive, data-driven countermeasures [2].

Traditional intrusion detection systems (IDS) that rely on static signatures or hand-crafted rules struggle with previously unseen or rapidly morphing threats [3]. Machine learning (ML) and artificial intelligence (AI) enable dynamic threat modeling, anomaly detection, and behavioral analysis, allowing IDSs to generalize to novel attacks while reducing false positives [3].

Despite notable advancements in IDS research, a critical limitation persists across most studies: the insufficient handling of class imbalance, which leads to poor detection rates for minority attack types. Previous works have primarily focused on maximizing overall accuracy, often at the expense of rare class detection, resulting in inflated performance metrics dominated by frequent categories. Furthermore, inconsistencies in preprocessing pipelines, a lack of standardized evaluation on modern NetFlow-based datasets like NF-UNSW-NB15-v2, and the limited integration of a diverse set of both machine and deep learning techniques within a single, optimized framework limit the robustness and practical applicability of existing IDS solutions. This study aims to bridge these gaps by proposing an integrated pipeline that combines SMOTE-based class balancing, PCA-driven feature reduction, and a comprehensive evaluation of multiple ML and DL models to enhance minority class detection and overall performance on the NF-UNSW-NB15-v2 dataset.

The remainder of the paper is organized as follows. Section II reviews related work on ML/DL-based IDS. Section III details the methodology, including the dataset, preprocessing (standardization, imbalance handling, dimensionality reduction, and train–test split), model specifications, and evaluation metrics. Section IV reports the experimental results for both binary and multi-class settings and examines inference-time performance for real-time applicability. Section V discusses key findings, practical implications, and limitations. Section VI concludes and outlines directions for future research.

II. Related Work

Kasongo and Sun [4] conducted a performance analysis of Intrusion Detection Systems (IDS) by applying a feature

selection algorithm to the UNSW-NB15 dataset. They emphasized the importance of reducing feature dimensionality to enhance IDS accuracy. The study used five machine learning classifiers—SVM, KNN, Logistic Regression, Artificial Neural Networks (ANN), and Decision Trees (DT)—and showed that XGBoost-based feature selection significantly improved model performance, especially for DTs. However, the study did not address class imbalance, which led to lower F1-scores for minority classes.

Kumar et al. [5] proposed an integrated rule-based IDS using both the UNSW-NB15 and RTNITP18 datasets. Their approach employed Decision Tree classifiers (C5, CHAID, CART, and QUEST) to detect five attack types: Generic, Probe, DoS, Exploit, and Normal. The study reported improved accuracy and reduced false alarm rates through rule-based modeling and information gain-based feature selection. Nonetheless, it lacked proper handling of class imbalance and yielded low precision and recall for critical attack types.

More et al. [6] compared several supervised ML techniques for identifying deceitful emails with filtering approaches and through use of the WEKA toolset. The work identifies a weakness in conventional Bayesian filtering, effective in spam filtering but not in high false positive cases. To counteract, several classifiers, such as RF and SVM, have been incorporated and utilized for enhancing accuracy and minimizing false positives in classification. In its use, feature extraction via Naïve Bayes and an evaluation tool developed in WEKA facilitated testing of numerous algorithms for classification in a thorough manner. Experimental tests showed that RF and SVM performed better in enhancing positive and negative actual values and overall accuracy over 96%. The work identifies the use of hybrid classification approaches in improving deceitful message detection and minimizing security threats posed through spammers.

Tahri et al. [7] have designed an IDS with ML algorithms for enhancing network security. As communications through electronic means have increased, IDS proves to be a useful tool for discovering hostile activity in network communications. In the current work, three classifiers, Naïve Bayes, SVM, and KNN, have been compared for performance with two benchmark datasets, namely, “NSL-KDD” and “UNSW-NB15”. In part one of work, three classifiers have been compared with the use of “UNSW-NB15”, and then for a proper analysis, best-performing algorithm is utilized for testing with “NSL-KDD”. As per work, SVM outperforms all classifiers in terms of accuracy consistently, with 97.77% accuracy for “UNSW-NB15” and 97.29% accuracy for NSL-KDD. In conclusion, SVM proves to be an effective intrusion detection classifier, and future work will attempt to make its processing efficient and integrate it in real-time security tools such as a firewall.

Musa et al. [8] review the application of ML algorithms in IDS for enhancing network security through observation of traffic and intrusion activity, and IDS is distinguished between anomaly-based and signature-based detection, with the first identifying abnormalities in behavior and the second employing predefined attack signatures. Various types of ML approaches, including single, hybrid, and ensemble classifiers, are contrasted and compared over seven datasets, with the consequence that single classifiers fall below both ensemble and hybrid classifiers in terms of accuracy and detection performance. Comparison between algorithms including SVM, DTs, RF, and Neural Networks identifies that ensemble approaches, including stacking classifiers, have a significant impact in intrusion detection improvement. Challenges include feature selection improvement, testing over a range of and updated datasets, and minimizing false positive values. Optimizing hybrid models, minimizing computational overload, and enhancing real-time intrusion capabilities have been suggested for future work, according to the authors.

Samantaray et al. in [9] conducted a comparative study on ML model implementation in intrusion detection in IoT-based networks. The research is centered on increasing threats in the security of IoT networks and the need to utilize efficient IDS in order to mitigate them. The research uses the “UNSW-NB15 (DS-1)” and “NF-UNSW-NB15 (DS-2)” datasets in comparing models based on ML like SVM, KNN, Logistic Regression, Naïve Bayes, DT, and RF. The feature scaling is based on a method involving the usage of the MaxAbsScaler algorithm in order to increase efficiency in classification. The results highlight the usage of the RF classifier in achieving the highest precision in generating the most accurate outcome with a gain in the rate of detection from 60% to 94% in the DS-2 dataset. The research focuses on efficiency in ML usage in intrusion detection and supports future research on implementation with improved feature selection and DL.

Sayed et al. in [10] conducted research with a focus on optimizing the efficiency of DNN-driven IoT intrusion detection systems (IDS). Because the IoT devices are under threat and there is a limitation in the process ability and in features in security, the researchers provided two CNN models, namely IoTCNN and MyCNN, with a purpose to classify intrusion in the network. The "NF-UNSW-NB15-v2" dataset was used in the research, and the stream network data was converted into RGB images in order to train the models. Results indicated the efficiency of the models in the detection of various intrusion types, and in the majority of intrusion categories, the precision of the models improved. The research confirms the efficiency of anomaly-based IoT security based on DL and calls for improvement in the handling of imbalances in the class and the optimization of the hyperparameters.

Table I provides a structured summary of key related studies, highlighting the datasets, methodologies, and performance metrics used, which helps position the present work within the broader landscape of IDS research.

Despite notable advancements in IDS research, a recurring limitation across most studies is the insufficient handling of class imbalance and the resulting poor detection of minority attack types. Previous works primarily focused on improving overall accuracy without explicitly addressing the critical challenge of rare class detection, often leading to inflated performance metrics dominated by frequent attack categories. In addition, inconsistencies in preprocessing, lack of standardized evaluation on newer datasets, and limited integration of deep learning techniques further limit the robustness of existing IDS solutions. Building on these gaps, this research proposes an

integrated approach combining SMOTE-based class balancing, PCA-driven feature selection, ML/DL modeling to enhance minority class detection and overall intrusion detection performance on the NF-UNSW-NB15-v2 dataset [20].

TABLE I. SUMMARY OF RELATED WORK

Year	Research Title	Dataset	Machine Learning / Deep Learning Model(s)	Preprocessing	Classification Type	Performance Metrics	Findings	Limitations	Ref
2020	Performance analysis of IDS using a feature selection method on UNSW-NB15	UNSW-NB15 [21]	SVM, KNN, Logistic Regression, ANN, Decision Tree	Min-Max normalization; Feature selection with XGBoost	Binary and Multi-class	Binary: Accuracy 90.85%, Precision 80.33%, Recall 98.38%, F1-score 88.45% Multi-class: Accuracy 77.51%, Precision 79.50%, Recall 77.53%, F1-score 77.28%	Feature selection (XGBoost) improved ML model performance.	No class imbalance handling; poor F1-scores for minority classes.	[4]
2020	An integrated rule-based intrusion detection system on UNSW-NB15 and RTNITP18	UNSW-NB15 [21], RTNITP18 [5]	Decision Trees (C5, CHAID, CART, QUEST)	Feature selection using Information Gain; K-Means clustering	Multi-class	Accuracy 84.83%, Approximate F1-score 68.13%	Rule-based modeling reduced false alarms in IDS.	No class imbalance handling; low precision and recall for critical attacks.	[5]
2015	Evaluation of deceptive mails using filtering & WEKA	SpamBase [22], Ling-Spam [23], Enron [24], PU1 [25], PU2 [25]	Random Forest, SVM, Naïve Bayes	Tokenization; Feature extraction; Term Frequency normalization	Binary	Accuracy: RF 98.9%, SVM 98.4%, NB 93.2%	Ensemble models (Random Forest) achieved high classification accuracy.	No class imbalance handling; no minority class evaluation.	[6]
2022	Intrusion Detection System using machine learning algorithms	UNSW-NB15 [21], NSL-KDD [26]	SVM, KNN, Naïve Bayes	Feature selection using mutual information;	Binary and Multi-class	Binary: SVM Accuracy 97.78% Multi-class: SVM Accuracy 97.29%	SVM achieved high accuracy on IDS datasets without heavy feature engineering.	No class imbalance handling; no minority class evaluation.	[7]
2020	Review of machine learning techniques for IDS across different datasets	KDDCup'99 [27], NSL-KDD [26], Kyoto2006+ [28], AWID [29], CIC-IDS2017 [30], UNSW-NB15 [21], UGR'16 [31]	SVM, Random Forest, Decision Tree, KNN, ANN, XGBoost, AdaBoost	Dataset-specific feature engineering; Standard normalization or scaling where needed	Binary and Multi-class	Binary: Ensemble models achieved >99% Accuracy Multi-class: Accuracy ~0.99, Macro F1 ~0.89	Ensemble methods (e.g., XGBoost) consistently outperformed individual classifiers.	No class imbalance handling; no minority class evaluation.	[8]
2024	Comparative assessment of ML algorithms in IoT-based network intrusion detection	UNSW-NB15 (DS-1), NF-UNSW-NB15 (DS-2) [20]	SVM, KNN, Logistic Regression, Naïve Bayes, Random Forest	MaxAbsScaler normalization	Multi-class	Accuracy: RF 60% (DS-1), 94% (DS-2)	Feature scaling (MaxAbsScaler) improved IDS model stability.	No class imbalance handling; no minority class evaluation.	[9]
2022	Augmenting IoT intrusion detection system performance using deep neural networks	NF-UNSW-NB15-v2 [20]	CNN	FFT-based NetFlow transformation to RGB images; Image normalization	Multi-class	Accuracy ~99% (frequent classes); poor F1-scores for minority classes	CNNs using NetFlow-to-image transformation achieved high accuracy for frequent attacks.	No class imbalance handling; poor F1-scores for minority classes.	[10]

III. METHODOLOGY

The methodology for this project is structured into several essential steps, as depicted in Fig. 1.

The proposed method for intrusion detection using deep learning (DL) and machine learning (ML) follows a structured pipeline:

1) Dataset: The NF-UNSW-NB15 dataset is used, comprising a combination of normal and malicious network traffic. It provides a realistic foundation for evaluating intrusion detection models.

Preprocessing Steps:

- Standardization: Normalizes the data to have the same feature scaling.
- PCA (Principal Component Analysis): Reduces dimensionality in a way that maximizes computational efficiency while retaining significant features.
- Oversampling: Balances the dataset and treats imbalances in classes, optimizing performance on minority attack classes.

2) Model training: Preprocessed data is fed to ML (e.g., XGBoost, RF, KNN) and DL (e.g., DNN, LSTM) models for training.

3) Evaluation metrics: Model performance is evaluated using conventional metrics, including accuracy, precision, recall, F1-score, and AUC-ROC, to ensure robust and comprehensive assessment of intrusion detection effectiveness.

B. Dataset

The NetFlow-based variant of the UNSW-NB15 dataset, referred to as NF-UNSW-NB15, incorporates additional flow-level features and is labeled according to specific attack categories. The original dataset comprises 2,390,275 network flow records, including 95,053 attacks (3.98%) and 2,295,222 benign flows (96.02%). These attack records are further divided into nine subtypes, as summarized in Table II [19]. For this study, the dataset was obtained from Kaggle, where the official version was uploaded by the author after removing duplicate rows, reducing the total count to 1,986,745. Therefore, the dataset used in this research is consistent with the original release, except for the exclusion of duplicates to improve data integrity and processing efficiency.

Mohanad Sarhan et al. [20] proposed a standardized feature set for network intrusion detection datasets to improve detection performance through the application of machine learning techniques. Their approach leverages NetFlow v9 features, which are widely supported by network devices and proven to be effective for traffic analysis. The proposed feature set includes 43 numerical, flow-based attributes designed to facilitate accurate and consistent detection of security events. By promoting dataset standardization, this feature set simplifies model evaluation, enhances compatibility for dataset merging, and supports real-world deployment of intrusion detection systems.

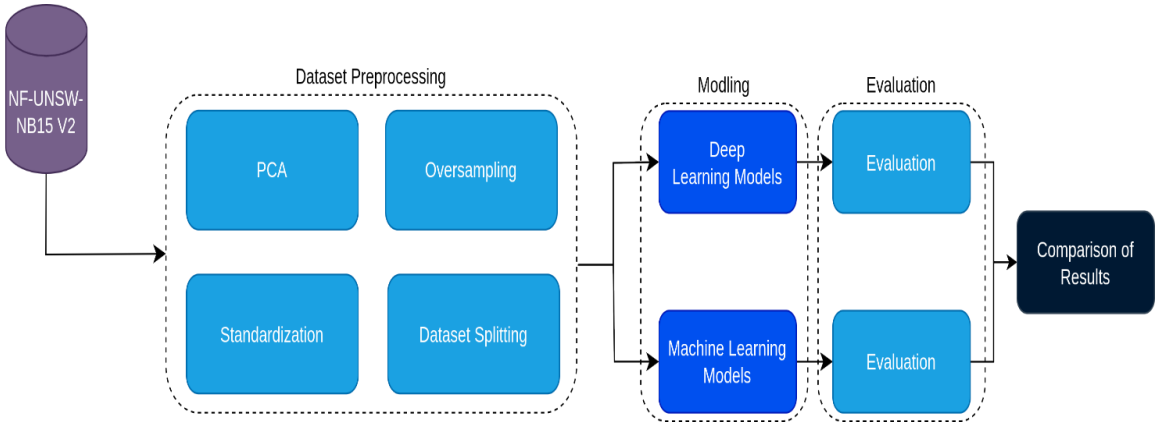

Fig. 1. Proposed method.

TABLE II. ATTACK TYPES IN "NF-UNSW-NB15"

Class	Count	Description
Benign	2295222	Normal, non-malicious network traffic.
Fuzzers	22310	An attack where large volumes of random data are sent to a system to cause crashes and identify security vulnerabilities.
Analysis	2299	A category of threats targeting web applications through ports, emails, and scripts.
Backdoor	2169	A method that bypasses security mechanisms by responding to specially crafted client requests.
DoS	5794	An attack that overwhelms a system's resources to disrupt access to its data or services.
Exploits	31551	Sequences of commands used to manipulate a system by exploiting known vulnerabilities.
Generic	16560	A cryptographic attack that causes collisions in block cipher encryption.
Reconnaissance	12779	Also known as probing, this technique involves gathering information about a network host.
Shellcode	1427	Malicious code designed to take control of a victim's system.
Worms	164	Self-replicating attacks that spread across multiple computers.

TABLE III. FEATURES IN "NF-UNSW-NB15"

	Field Name	Data Type	Description
1	IPV4_SRC_ADDR	String	Source IPv4 address
2	IPV4_DST_ADDR	String	Destination IPv4 address
3	L4_SRC_PORT	Integer	Source port number for IPv4
4	L4_DST_PORT	Integer	Destination port number for IPv4
5	PROTOCOL	Integer	Byte value representing the IP protocol identifier
6	L7_PROTO	Integer	Numeric identifier for the Layer 7 protocol
7	IN_BYTES	Integer	Total incoming bytes
8	OUT_BYTES	Integer	Total outgoing bytes
9	IN_PKTS	Integer	Count of incoming packets
10	OUT_PKTS	Integer	Count of outgoing packets
11	FLOW_DURATION_MILLISECONDS	Float	Duration of the flow in milliseconds
12	TCP_FLAGS	Integer	Aggregated TCP flags
13	CLIENT_TCP_FLAGS	Integer	Aggregated TCP flags from the client side
14	SERVER_TCP_FLAGS	Integer	Aggregated TCP flags from the server side
15	DURATION_IN	Float	Duration of the client-to-server stream (in milliseconds)
16	DURATION_OUT	Float	Duration of the server-to-client stream (in milliseconds)
17	MIN_TTL	Integer	Minimum Time-to-Live (TTL) value observed in the flow
18	MAX_TTL	Integer	Maximum Time-to-Live (TTL) value observed in the flow
19	LONGEST_FLOW_PKT	Integer	Size (in bytes) of the longest packet in the flow
20	SHORTEST_FLOW_PKT	Integer	Size (in bytes) of the shortest packet in the flow
21	MIN_IP_PKT_LEN	Integer	Length of the smallest observed IP packet in the flow
22	MAX_IP_PKT_LEN	Integer	Length of the largest observed IP packet in the flow
23	SRC_TO_DST_SECOND_BYTES	Float	Rate of bytes sent from source to destination (bytes per second)
24	DST_TO_SRC_SECOND_BYTES	Float	Rate of bytes sent from destination to source (bytes per second)
25	RETRANSMITTED_IN_BYTES	Integer	Count of retransmitted TCP bytes from source to destination
26	RETRANSMITTED_IN_PKTS	Integer	Count of retransmitted TCP packets from source to destination
27	RETRANSMITTED_OUT_BYTES	Integer	Count of retransmitted TCP bytes from destination to source
28	RETRANSMITTED_OUT_PKTS	Integer	Count of retransmitted TCP packets from destination to source
29	SRC_TO_DST_AVG_THROUGHPUT	Float	Average throughput (bps) from source to destination
30	DST_TO_SRC_AVG_THROUGHPUT	Float	Average throughput (bps) from destination to source
31	NUM_PKTS_UP_TO_128_BYTES	Integer	Number of packets with an IP size of 128 bytes or less
32	NUM_PKTS_128_TO_256_BYTES	Integer	Number of packets with an IP size between 128 and 256 bytes
33	NUM_PKTS_256_TO_512_BYTES	Integer	Number of packets with an IP size between 256 and 512 bytes
34	NUM_PKTS_512_TO_1024_BYTES	Integer	Number of packets with an IP size between 512 and 1024 bytes
35	NUM_PKTS_1024_TO_1514_BYTES	Integer	Number of packets with an IP size between 1024 and 1514 bytes
36	TCP_WIN_MAX_IN	Integer	Maximum TCP window size from source to destination
37	TCP_WIN_MAX_OUT	Integer	Maximum TCP window size from destination to source
38	ICMP_TYPE	Integer	ICMP type combined with ICMP code (ICMP Type * 256 + ICMP Code)
39	ICMP_IPV4_TYPE	Integer	ICMP type identifier for IPv4
40	DNS_QUERY_ID	Integer	Transaction ID of a DNS query
41	DNS_QUERY_TYPE	Integer	Type of DNS query (e.g., 1 = A, 2 = NS, etc.)
42	DNS_TTL_ANSWER	Integer	Time-to-Live (TTL) value of the first A record, if available
43	FTP_COMMAND_RET_CODE	Integer	Return code for an FTP client command

Table III lists the feature set of the NF-UNSW-NB15 dataset, detailing the flow characteristics captured for intrusion detection analysis.

Fig. 2 illustrates the significant class imbalance in the dataset, with benign (normal) traffic overwhelmingly dominating all categories of malicious traffic. Among the attack types, "Exploits" and "Fuzzers" appear most frequently, while others such as "Reconnaissance," "DoS," and especially "Worms" occur far less often. This imbalance poses challenges for accurate model training and may necessitate the use of resampling methods or advanced techniques to improve detection performance, particularly for minority attack classes.

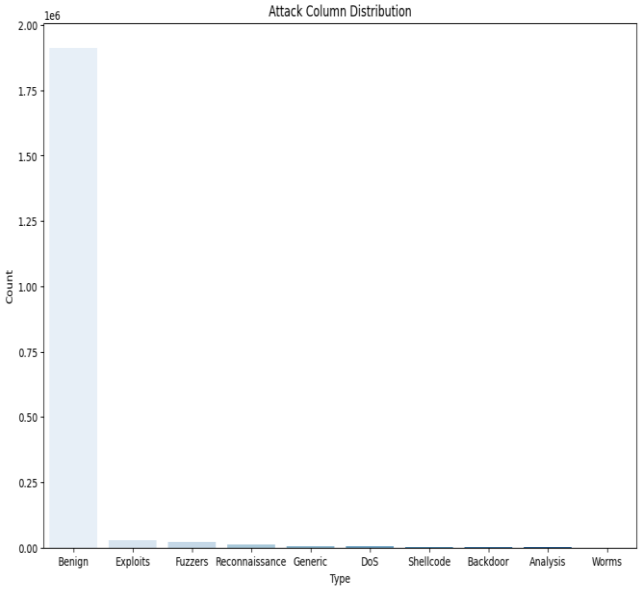

Fig. 2. Attack column distribution.

Fig. 3 displays how normal and malicious flows are distributed in the dataset, revealing a pronounced class imbalance: legitimate traffic vastly outweighs attack traffic.

Such skew can hamper model effectiveness, so remedies like SMOTE oversampling or class-weight adjustment are advisable to achieve more reliable classification.

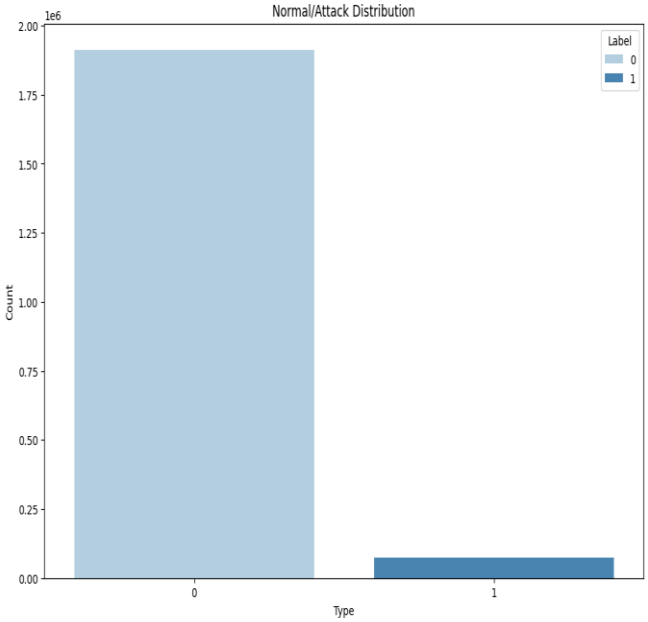

Fig. 3. Label column distribution.

Fig. 4 shows the distribution of different types of attacks in the dataset. "Exploits" is the highest frequency type, with "Fuzzers" and "Reconnaissance" following, and "Worms" and "Analysis" are the least frequent. The imbalanced distribution among attack types implies the need for careful model training in order to correctly identify in every category.

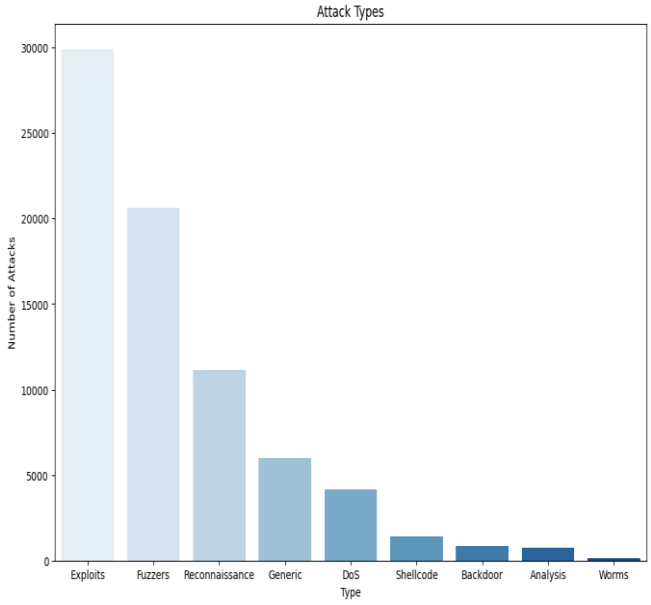

Fig. 4. Count of every attack type.

C. Dataset Preprocessing

To ensure high-quality input for model training, the NF-UNSW-NB15-v2 dataset undergoes a series of preprocessing steps designed to enhance learning efficiency and model performance.

1) Standardization: Standardization was applied using the StandardScaler, which transformed each numerical feature to have a mean of zero and a standard deviation of one. This normalization process ensured that all features contributed proportionally to the learning process, preventing any single feature with a large magnitude from disproportionately influencing the model. Standardization was particularly important for algorithms sensitive to feature scale, such as K-Nearest Neighbors (KNN) and Deep Neural Networks (DNN), which rely on distance-based calculations and gradient-based optimization, respectively. This approach follows best practices outlined in prior literature and common implementations such as scikit-learn [11].

2) Handling class imbalance: To address the class imbalance issue in the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) [32] was applied exclusively to the training set after splitting the data into 70% for training and 30% for testing. This approach was intentionally adopted to prevent data leakage and to ensure an unbiased evaluation of the model's performance on unseen data.

SMOTE generates synthetic samples for minority classes by interpolating between existing instances rather than simply duplicating them. For each minority sample in the training set, the algorithm identifies its k-nearest neighbors (commonly k=5) within the same class. It then selects one neighbor at random and creates a new instance by generating a point along the feature-space line segment connecting the original sample and its neighbor. This synthetic instance inherits the statistical

characteristics of both points, resulting in a new example that is both realistic and non-redundant [32].

By enriching the training set with these synthetic examples, the class distribution becomes more balanced. This allows learning algorithms to be exposed to a wider variety of patterns and variations within the minority classes, which enhances their ability to generalize and improves detection of rare attack types. The test set was left untouched to preserve the original distribution and maintain the integrity of the evaluation.

3) Dimensionality reduction: To address the high dimensionality of the dataset, which consists of 43 flow-based NetFlow features, Principal Component Analysis (PCA) was employed as an effective dimensionality reduction technique. PCA transforms the original feature set into a smaller number of uncorrelated components while retaining the majority of the data's variance. This transformation reduces computational complexity, accelerates model training, and mitigates the risk of overfitting by eliminating redundant or less informative attributes [13].

An initial correlation analysis of the raw NetFlow features revealed strong linear relationships among several variables. For instance, IN_PKTS and OUT_PKTS exhibited a Pearson correlation coefficient of approximately 0.99, while FLOW_DURATION_MILLISECONDS was highly correlated with both DURATION_IN and DURATION_OUT. Similarly, MIN_TTL and MAX_TTL showed correlations exceeding 0.90. These relationships, visualized in Fig. 5, support the application of PCA to reduce multicollinearity and noise within the dataset.

To validate the independence of the PCA components, a correlation matrix was generated to examine the relationships between the extracted components. As shown in Fig. 6, the components are effectively uncorrelated, demonstrating that PCA successfully transforms the original feature space into a set of orthogonal, linearly independent dimensions. This orthogonality reinforces PCA's suitability for improving model robustness and reducing feature redundancy.

Overall, PCA proved especially beneficial for ensemble models such as Random Forest and XGBoost, which can be negatively affected by irrelevant or highly correlated features. Its use enhanced model focus, reduced overfitting, and improved interpretability within the intrusion detection pipeline.

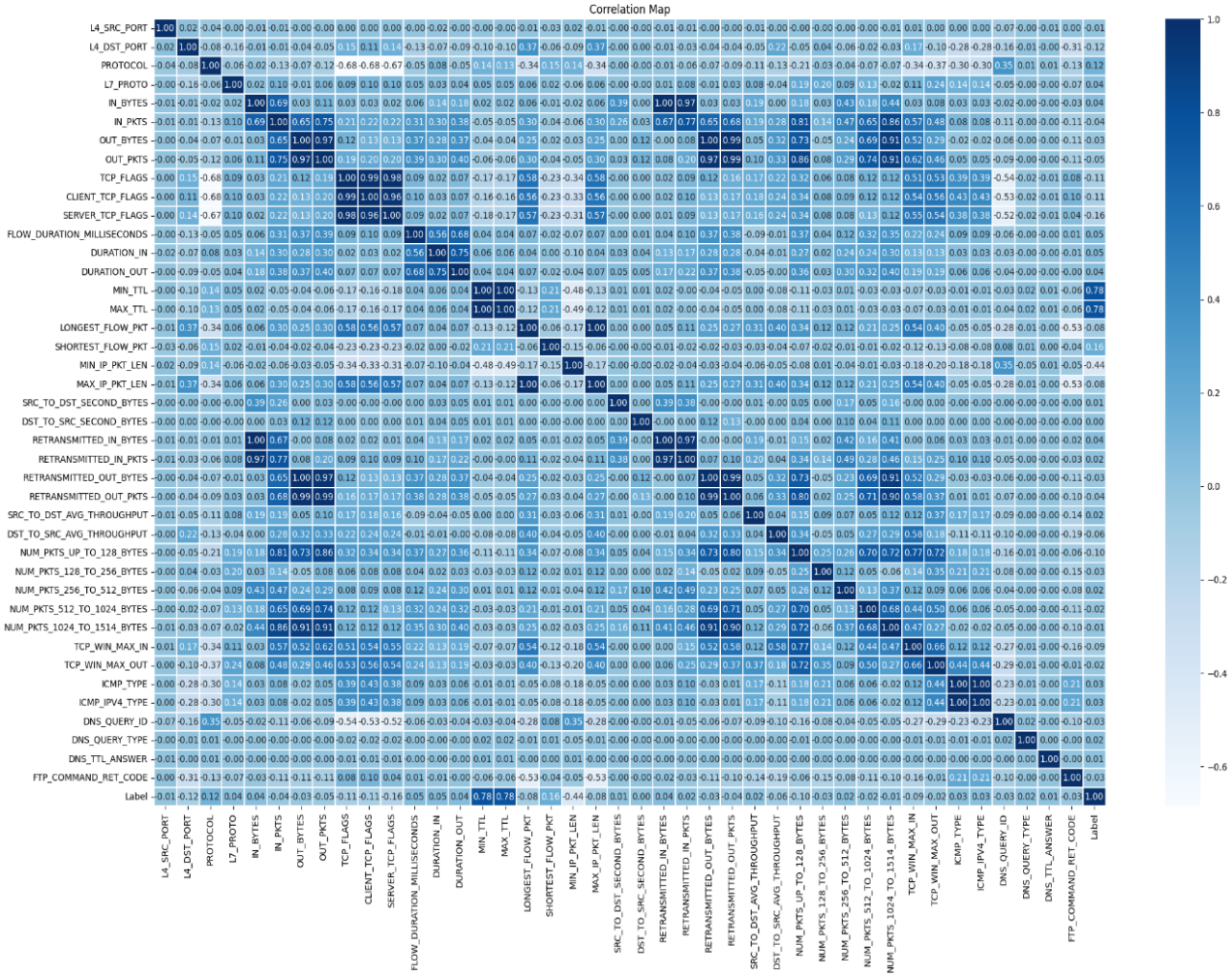

Fig. 5. Correlation heatmap of original NetFlow features.

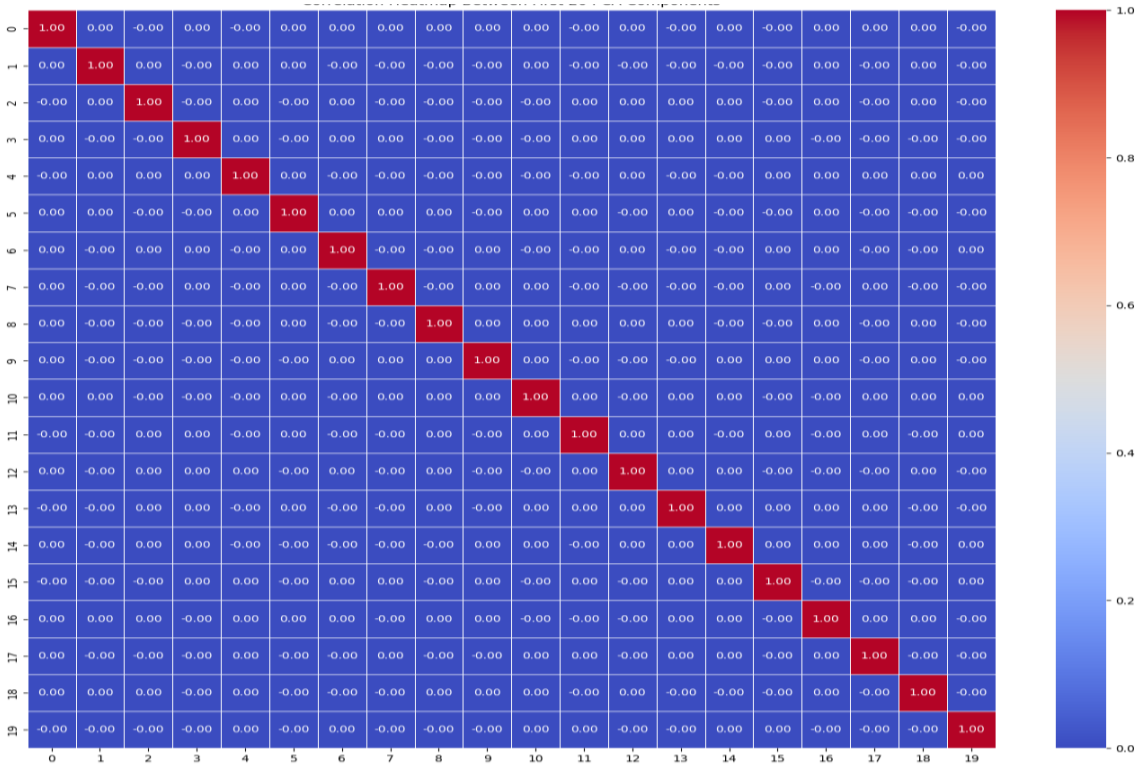

Fig. 6. Correlation between PCA Components (with self-correlation).

4) Train-test split: To evaluate model performance fairly, the dataset is divided into distinct subsets. For traditional machine learning models, the dataset is partitioned into 70% for training and 30% for testing to evaluate generalization performance on unseen data. In the case of deep learning models such as LSTM and DNN, the dataset is further divided into 70% for training and 30% for testing, enabling effective monitoring during training and ensuring robust model evaluation.

D. Modeling

1) Machine learning models: Random Forest (RF), KNN, XGBoost, and SVM are the most effective ML models for the detection of intrusions. The models are good for classification with feature-extracted input but are inefficient with complex temporal relationships within sequential input.

a) K-Nearest Neighbor (KNN): The KNN algorithm is a supervised algorithm in ML most commonly used to classify tasks [11]. It identifies the unlabeled data by considering the label and the available training data's features. The algorithm identifies the data by determining the point's nearest neighbors and the final label by majority voting. Among all the algorithms in ML, the algorithm in the case of the KNN algorithm is unique in terms of ease and interpretability and, in turn, acts to be a commonly used algorithm to classify tasks [12]. Even despite the ease, the algorithm performs exceptionally in the case of solving the classification and the regression problems in different datasets, regardless of size, label distribution, the datasets' noise, and the datasets' ranges [12].

b) Random Forests (RF): RF is a versatile algorithm in ML, famous for reducing the effect of overfitting, a prevalent problem in decision trees (DTs). It performs tasks such as classification, regression, and others by building many DTs during the training period. The algorithm works by evaluating multiple distinct decision trees (DTs) and making the prediction through a voting mechanism. Unlike in the case of a standard DT, in which each internal node gets partitioned by the optimal attribute, RF uses the optimal attribute from a random subset of predictors at each internal node. This randomization serves to enhance model generalization and resistance, and the resulting algorithm is a versatile tool in numerous applications in ML.

c) XGBoost (XGB): XGBoost (XGB) is a Gradient Tree Boosting algorithm powerful enough to solve heavy-scale ML problems in an efficient and effective manner. It possesses great prediction precision and model training speed, and it's the leading performer in all the competitions at Kaggle. The mechanism in XGB lies in the addition of trees in an iterative fashion and the division of the features during the course of the expansion in the tree. The model learns to fit the residuals from the last prediction each time a new tree gets added [13]. Given an input x_i, a true label y_i, and a raw prediction z_i before applying the sigmoid function, according to [14], the XGBoost model defines its objective function in the following equation:

$$L^{(t)} = \sum_{i=1}^{n} \quad l\left(y_i, Z_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + c \quad (1)$$

Where $l(.,.)$ represents the loss function, t denotes the t-th tree, and Ω serves as a penalty for model complexity. The term $\Omega(f_t)$ refers to the regularization penalty, while c is a constant.

The second-order Taylor expansion is given by:

$$f(x+\Delta x) \approx f(x) + f'(x)\Delta x + 1/2\, f''(x)\Delta x^2 \quad (2)$$

By substituting Eq. (2) into Eq. (1), we can obtain the following result.

$$L^{(t)} \approx \sum_{i=1}^{n}\left[l\left(y_i + Z_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2}h_i\left(f_t(x_i)\right)^2\right] \quad (3)$$

where $g_i = \partial\, L/\partial\, z_i$, and $h_i = \partial^2\, L/\partial\, z_i^2$. By eliminating the constant terms, we derive the following simplified objective at step t.

$$L^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i\left(f_i(x_i)\right)^2\right] + \Omega(f_t) \quad (4)$$

The terms g_i and h_i play a vital role in optimizing the XGBoost training process. For binary classification, the model typically employs cross-entropy (CE) as its default loss function.

$$L = -\sum_{i=1}^{n}[y_i log(\acute{y}_i) - (1-y_i)log(1-\acute{y}_i)] \quad (5)$$

In Eq. (5), $\acute{y}_i = 1/[1+exp(-z_i)]$, that is sigmoid is selected as activation. Therefore, we can get:

$$\partial\, \acute{y}_i/\partial\, z_i = \acute{y}_i(1-\acute{y}_i) \quad (6)$$

2) Deep learning models: Deep-learning methods often fall under the umbrella of unsupervised pre-trained networks—architectures that stack many more layers and parameters than typical machine-learning neural nets, earning them the designation "deep."

a) Deep Neural Network (DNN): An Artificial Neural Network (ANN) is a model based on the structure and function of the brain [15]. Since neural networks (NN) are powerful nonlinear discriminators in the event of problems in classification, because they are able to describe any decision boundary in the feature space [16]. In recent years, Deep Neural Networks (DNNs) gained significant interest in intrusion detection research and evolved from Shallow Neural Networks (SNNs). The feature abstraction ability in DNNs and the ability to represent highly complex patterns make them extremely useful in applications in DL. Because of their ability to represent data in a good way, DNNs are in high demand in order to design efficient and robust solutions.

The results are produced in a DNN based on the connection weights and activation functions in the neurons. The DNN is composed of multiple processing layers, and every layer contributes to decision-making and feature extraction. Several hyperparameters dictate the operation of a DNN and are to be determined in advance, including the number of units, number of layers, weights and bias initializers, activation function, regularizer's coefficient, learning rate, and the optimizer. In this DNN model, ReLU activation is applied in the input layer and in every hidden layer. The ReLU function is a piecewise linear function and returns the same input in the situation where the input is a positive number and a value of zero in the situation where the input is a negative number [17]. The neurons activated by this function are also rectified linear activation units.

$$ReLU(x) = max(0, x) \quad (7)$$

b) Long Short-Term Memory (LSTM): The LSTM layer is a special kind of [18] RNN, and the main job of the LSTM layer is to handle sequence data with temporal relationships, such as text and relation data. The LSTM layer consists of three gates, the forget gate, the output gate, and the input gate, and the shared state. The use equation and the LSTM layer's detailed working are presented in the following equations:

$$f_s = \sigma\left(W_f \cdot [h_{s-1}, x_s] + b_f\right) \quad (8)$$

$$i_s = \sigma\left(W_f \cdot [h_{s-1}, x_s] + b_i\right) \quad (9)$$

$$\acute{C} = tanh(W_c \cdot [h_{s-1}, x_s] + b_C) \quad (10)$$

$$C_s = f_s \cdot C_{s-1} + i_s \cdot \acute{C} \quad (11)$$

$$o_s = (W_o[h_{s-1}, x_s] + b_C) \quad (12)$$

$$h_s = o_s \cdot tan(C_s) \quad (13)$$

The input gate controls how the LSTM cell state acquires the information, in Eq. (9). The forget gate controls how the LSTM cell state forget the information, in the Eq. (8). The cell state updates by the Eq. (10) and Eq. (11).

Output Layer:

The output layer generates the predicted sentiment classification of the comment. The final output vector is scaled using the softmax activation function to produce a probability distribution across each class. The equation is as follows:

$$y = softmax\left(W h_{fc} + b\right) \quad (14)$$

E. Evaluation Metrics

1) Confusion matrix: It's a matrix to represent how model classify dataset, also to use to check how good the model's performance of classification. It's comprised by four components.

In a confusion-matrix context, true positives (TP) are cases where an instance is genuinely positive and the model correctly labels it as such, whereas true negatives (TN) are instances that are truly negative and rightly classified as negative. By contrast, false positives (FP) occur when a genuinely negative instance is mistakenly flagged as positive, and false negatives (FN) arise when a genuinely positive instance is incorrectly marked as negative.

2) ROC Curve (Receiver Operating Characteristic Curve): A graph that shows classification performance across all thresholds by plotting the true-positive rate (TPR) against the false-positive rate (FPR), illustrating the trade-off between sensitivity and specificity.

3) Score: The harmonic mean of precision and recall; a single, balanced metric that is especially informative when class distributions are unbalanced. It ranges between 0 and 1, and 1 in the event of optimal precision and recall.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (15)$$

4) Accuracy: The rate of correctly classified observations over the total observations. It is a helpful indicator in the situation where false positives and false negatives are equally distributed.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (16)$$

5) Precision: The proportion of the total predicted positives actually being positives. The higher the precision, the lower the rate of the incorrect positives, and the lowest the incorrect positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (17)$$

6) Recall: The ratio of correctly predicted positive instances to all actual positive instances. A higher recall indicates fewer missed positives and a lower rate of false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (18)$$

IV. Results

This section presents how a range of ML and DL models performed on binary and multi-class intrusion-detection tasks using the "NF-UNSW-NB15" dataset. Before training, we applied PCA to cut dimensionality and boost both speed and accuracy. Every model was tuned with the same hyper-parameter settings to keep the comparison fair. Their effectiveness was gauged with a full suite of metrics—accuracy, precision, recall, F1-score, AUC, and confusion matrices—so we could see each algorithm's strengths and weaknesses in detail. Results are split into two categories: binary detection of normal versus malicious traffic, and multi-class detection that pinpoints the exact attack type.

A. Hyperparameter Configuration

All deep learning models (LSTM and DNN) were trained using a fixed set of hyperparameters to ensure a fair and consistent comparison. For binary classification tasks, the loss function employed was binary crossentropy, while sparse categorical crossentropy was used for multi-class classification. Model optimization was performed using the Adam optimizer with a learning rate of 0.001. Both tasks were trained over 20 epochs with a batch size of 128. Additionally, early stopping was applied based on validation loss to prevent overfitting and ensure optimal generalization performance.

Table IV presents the hyperparameter settings used for model training, including learning configuration, optimization strategy, and loss functions for both binary and multi-class tasks.

B. Binary Classification Results

Before applying SMOTE, all models achieved high overall accuracy (0.99); however, recall and F1-scores—particularly for the minority class—were comparatively lower. For instance, the LSTM model achieved a recall of 0.92 and an F1-score of 0.94, while the DNN model recorded a recall of 0.90. These results indicate reduced sensitivity to minority class detection due to the dataset's class imbalance.

Moreover, AUC scores, which evaluate a model's ability to distinguish between classes, were also affected. Deep learning models such as DNN and LSTM yielded relatively lower AUCs in the range of 0.90–0.92, suggesting less reliable separability between benign and malicious traffic. In contrast, ensemble models like Random Forest and XGBoost performed better, achieving AUC scores around 0.98. Nevertheless, even these models demonstrated measurable improvements after applying SMOTE.

These findings confirm that SMOTE plays a critical role not only in enhancing recall and F1-scores but also in improving the overall discriminative power of classifiers, as reflected in AUC metrics. Table V summarizes the performance of LSTM, DNN, Random Forest, KNN, and XGBoost on the binary classification task prior to applying SMOTE.

After applying SMOTE, all models demonstrate strong performance, with Random Forest and KNN achieving high accuracy and F1-scores. While LSTM and DNN yield slightly lower precision for the Attack class, they still achieve perfect recall, indicating high sensitivity to positive cases. The AUC score of 0.99 or higher across all models confirms excellent separability between classes. Among all models, RF achieved the best overall performance with an F1-score of 0.97, and highly balanced precision and recall values, especially for class 1 (Attack class). Its confusion matrix is very high, showing low misclassificationwith 2308 false positives and 199 false negative, Fig. 7 shows Random Forest confusion matrix that effectively classified both Normal and Attack classes, making it the most reliable model for the binary classification task.

TABLE IV. Hyperparameter Settings

Parameter	Value
Epochs	20
Batch Size	128
Learning Rate	0.001
Optimizer	Adam
Loss Function (Binary)	Binary Crossentropy
Loss Function (Multi)	Sparse Categorical Crossentropy
PCA Components (Binary and Multi)	20

TABLE V. Binary Classification Performance without SMOTE

Model	Accuracy	Precision	Recall	F1-score	AUC
LSTM	0.99	0.97	0.92	0.94	0.92
DNN	0.99	0.97	0.90	0.93	0.90
RF	0.99	0.97	0.98	0.98	0.98
KNN	0.99	0.95	0.95	0.94	0.95
XGBoost	0.99	0.96	0.97	0.97	0.98

Fig. 8 shows LSTM confusion matrix for classifying both Normal and Attack classes.

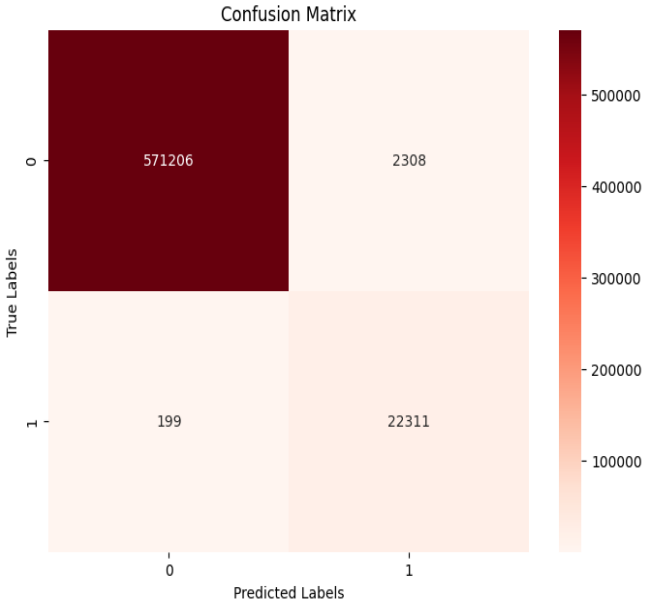

Fig. 7. RF confusion matrix.

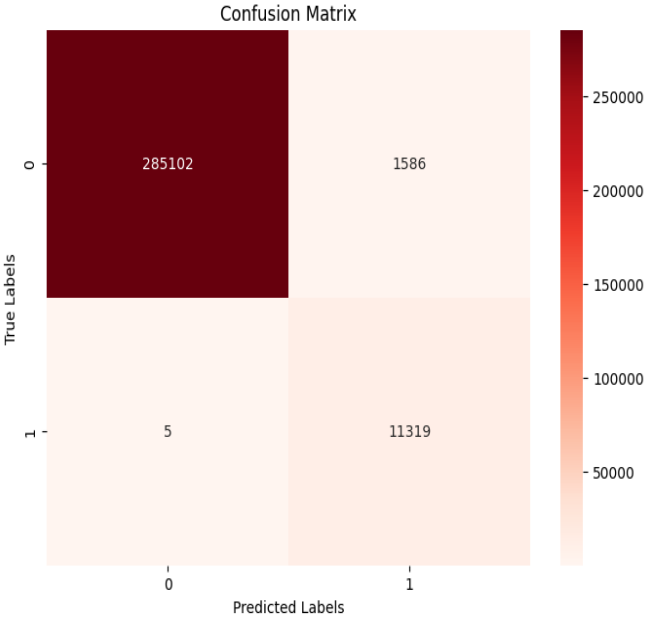

Fig. 8. LSTM confusion matrix.

The confusion matrix of the LSTM model reveals strong sensitivity in detecting attacks, although a slight increase in false positives indicates a trade-off in precision.

Fig. 9 shows DNN confusion matrix for classifying both Normal and Attack classes.

This matrix illustrates the DNN model's good performance, with solid detection capability and also slight false alarm rate, reflecting effective learning of attack patterns.

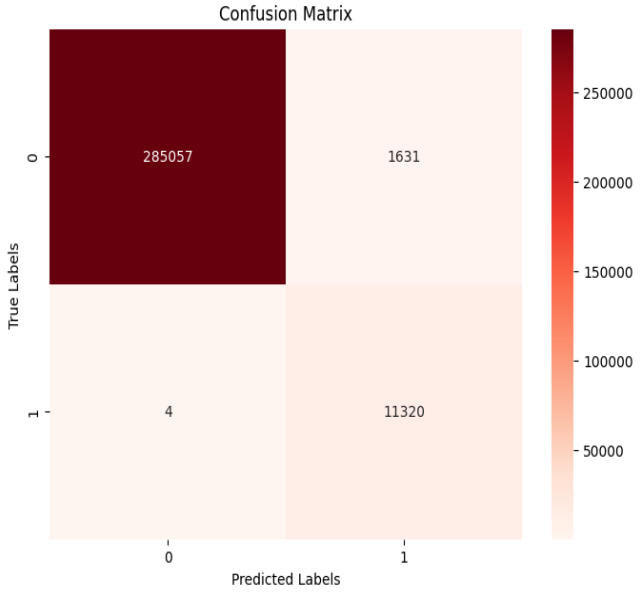

Fig. 9. DNN confusion matrix.

Fig. 10 shows KNN confusion matrix for classifying both Normal and Attack classes.

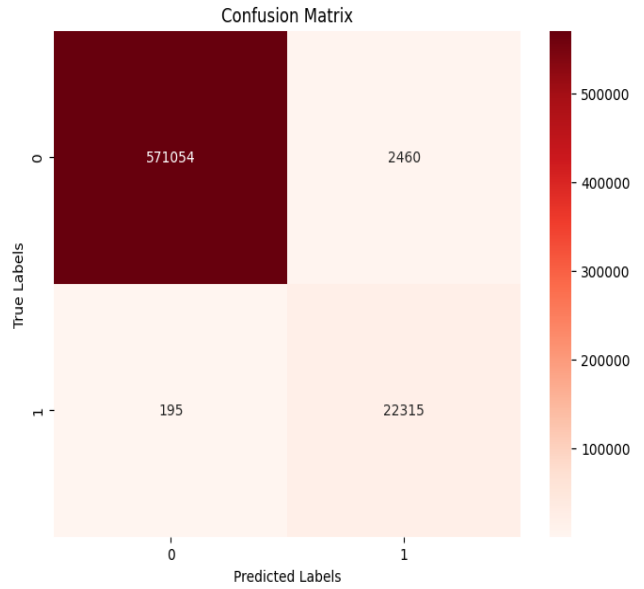

Fig. 10. KNN confusion matrix.

The KNN confusion matrix indicates good accuracy, with slightly higher misclassification of Normal status.

Fig. 11 shows XGBoost confusion matrix for classifying both Normal and Attack classes.

XGBoost exhibits high precision and recall, evident in its compact and clearly defined confusion matrix blocks with 2164 false positives and 239 false negatives, signifying high discriminative power.

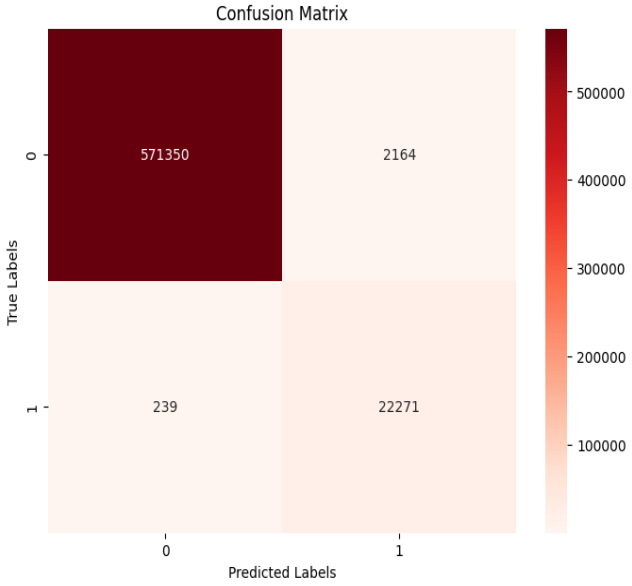

Fig. 11. XGBoost confusion matrix.

Table VI summarizes the performance of LSTM, DNN, RF, KNN, and XGBoost on the binary classification task.

TABLE VI. BINARY CLASSIFICATION PERFORMANCE WITH SMOTE

Model	Accuracy	Precision	Recall	F1-score	AUC
LSTM	0.99	0.94	0.99	0.97	0.99
DNN	0.99	0.94	0.99	0.96	0.99
RF	0.99	0.95	0.99	0.97	0.99
KNN	0.99	0.95	0.99	0.97	0.99
XGBoost	0.99	0.95	0.99	0.97	0.99

C. Multi-Class Classification Results

Unlike binary classification, the multi-class classification task suffered from noticeable performance degradation in the absence of SMOTE, especially in models like DNN, which recorded extremely low precision, recall, and F1-score (all = 0.10) despite an overall accuracy of 0.99. This illustrates that accuracy alone is misleading under class imbalance.

Similarly, LSTM showed moderate metrics (F1 = 0.52), while Random Forest and XGBoost performed relatively better (F1 = 0.64 and 0.62 respectively). AUC scores also reflected this trend—DNN had a poor AUC of 0.50, while XGBoost reached 0.98. These results demonstrate the limitations of models when trained on highly imbalanced data and reinforce the need for resampling methods like SMOTE to achieve fair multi-class performance across all attack types.

Table VII summarizes the performance of LSTM, DNN, RF, KNN, and XGBoost on the multi-class classification task before applying SMOTE.

Table VIII details the multi-class classification performance across the same set of models after SMOTE.

Model performances vary more noticeably in the multi-class setting. LSTM and DNN, while achieving high overall accuracy due to class imbalance in test set, show lower macro-averaged F1-scores, indicating challenges in learning minority classes. In contrast, Random Forest and XGBoost perform robustly across all classes, benefiting from ensemble-based learning. KNN also performs competitively but with slightly lower recall. The XGBoost classifier emerged as a competitive performer in the multi-class setting, achieving a macro F1-score (0.71), reflecting moderate performance across most classes, including minority ones. The confusion matrix of XGBoost reflects moderate performance on minority classes such as 6, 7, 8, and 9 with misclassifications, e.g., 919 false negatives for class 2 and 1455 for class 4 as shown in Fig. 12.

LSTM's multi-class confusion matrix (Fig. 13) shows robust classification for dominant classes but reveals some difficulty in differentiating among minority attack types.

TABLE VII. MULTI-CLASS CLASSIFICATION MACRO-AVERAGED SCORES WITHOUT SMOTE

Model	Accuracy	Precision	Recall	F1-score	AUC
LSTM	0.99	0.57	0.53	0.52	0.92
DNN	0.99	0.10	0.10	0.10	0.50
RF	0.99	0.66	0.62	0.64	0.94
KNN	0.99	0.61	0.58	0.59	0.87
XGBoost	0.99	0.66	0.60	0.62	0.98

TABLE VIII. MULTI-CLASS CLASSIFICATION MACRO-AVERAGED SCORES

Model	Accuracy	Precision	Recall	F1-score	AUC
LSTM	0.99	0.62	0.56	0.56	0.99
DNN	0.99	0.66	0.57	0.59	0.99
RF	0.98	0.68	0.71	0.63	0.99
KNN	0.99	0.69	0.66	0.63	0.89
XGBoost	0.99	0.73	0.80	0.71	0.99

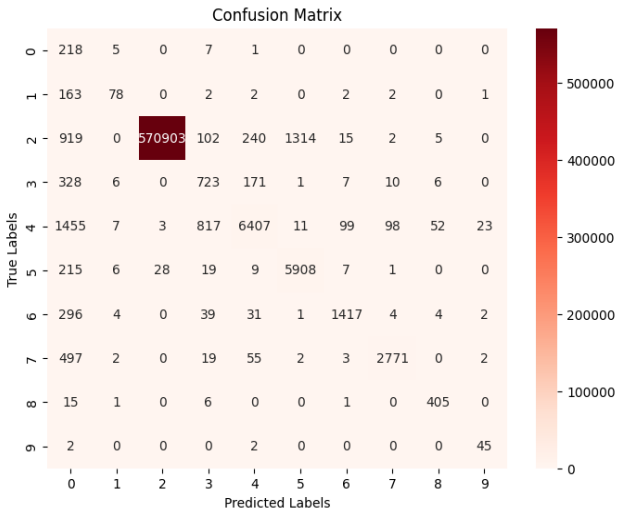

Fig. 12. XGboost confusion matrix in multi-classification.

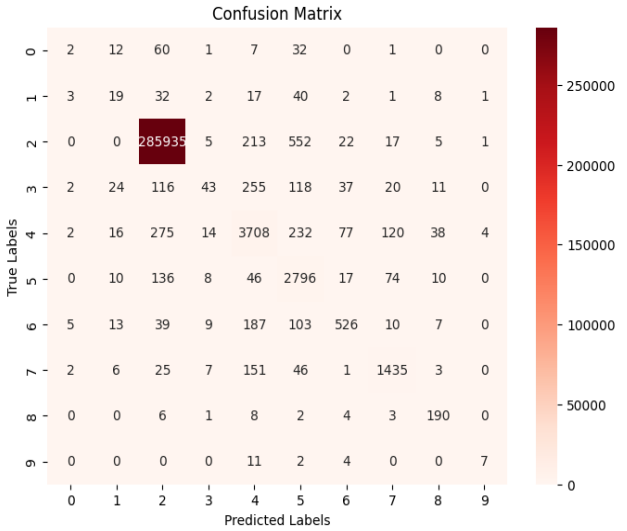

Fig. 13. LSTM confusion matrix in multi-classification

The DNN model performs well across several classes, but the confusion matrix reflects misclassification among minority classes as shown in Fig. 14.

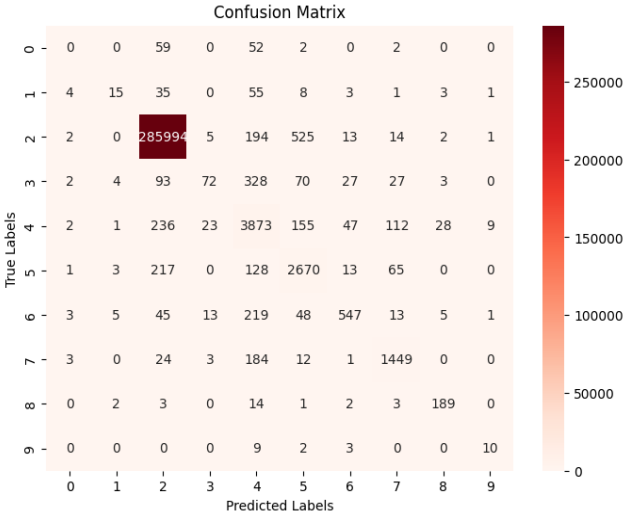

Fig. 14. DNN confusion matrix in multi-classification.

The confusion matrix of KNN (Fig. 15) shows good effectiveness, better than LSTM and DNN, but still less than Xgboost.

Random Forest delivers a compact and efficient confusion matrix, excelling at distinguishing between multiple attack types, especially frequent ones, while showing some overlap in less-represented classes as shown in Fig. 16.

D. Inference Time Evaluation for Real-Time Applicability

In addition to classification performance, prediction time is a critical consideration for real-world deployment, particularly in systems requiring real-time or near-real-time responses. Table IX presents both the total and average prediction times (per 1,000 records) for each model across binary and multi-class classification tasks, providing insight into their computational efficiency.

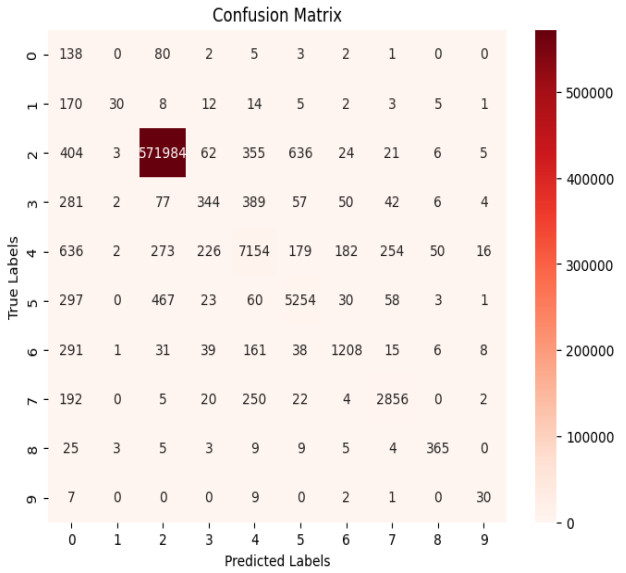

Fig. 15. KNN confusion matrix in multi-classification.

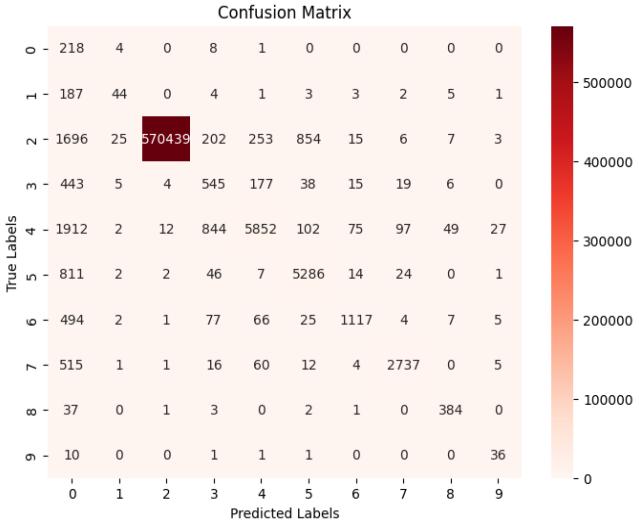

Fig. 16. Random forest confusion matrix in multi-classification.

TABLE IX. PREDICTION TIME FOR EACH CLASSIFIER

Model	Task	Total Time (s)	Avg. Time per 1000 Records (s)
LSTM	Binary	25	0.3575
DNN	Binary	20	0.0700
RF	Binary	4.519	0.0076
KNN	Binary	73.4142	0.1232
XGBoost	Binary	14.323	0.0240
LSTM	Multi-class	22.8	0.07
DNN	Multi-class	20.5489	0.0690
RF	Multi-class	5.0694	0.0128
KNN	Multi-class	39.0577	0.0983
XGBoost	Multi-class	26.7887	0.0449

Random Forest exhibited the fastest prediction times in both binary and multi-class tasks, making it highly suitable for real-time applications. Random Forest also demonstrated strong efficiency, whereas deep learning models, particularly LSTM and KNN, had the longest processing times.

V. Discussion

This study presents a comprehensive evaluation of multiple machine learning (ML) and deep learning (DL) classifiers for both binary and multi-class network intrusion detection using the NF-UNSW-NB15 dataset. The results demonstrate that the proposed models—particularly XGBoost and Random Forest—achieved superior performance, with macro-averaged F1-scores and accuracy exceeding 99% in binary classification. These models also maintained robustness in multi-class scenarios, achieving macro F1-scores of up to 0.71.

In contrast, the recent work by Samantaray et al. [9], which employed traditional ML algorithms on the same dataset, reported significantly lower macro F1-scores—only 0.26 with Logistic Regression and 0.17 with Random Forest—despite using MaxAbsScaler for feature scaling. Although their approach achieved relatively high overall accuracy (up to 94%) in multi-class classification, the low macro-averaged metrics reflect poor generalization across minority classes, indicating that class imbalance was not sufficiently addressed.

TABLE X. Comparative Results of Best Classifiers Across Studies

Study	Model	Dataset	Type	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
This Work	RF	NF-UNSW-NB15-v2	Binary	0.99	0.95	0.99	0.97
This Work	XGBoost	NF-UNSW-NB15-v2	Multi-class	0.99	0.73	0.80	0.71
Kasongo and Sun [4]	XGBoost	NF-UNSW-NB15	Binary	0.91	0.80	0.98	0.88
Samantaray et al. [9]	RF	NF-UNSW-NB15	Multi-class	0.90	0.23	0.20	0.17
Samantaray et al. [9]	KNN	NF-UNSW-NB15	Multi-class	0.93	0.29	0.31	0.26
Sayed et al. [10]	IoTCNN	NF-UNSW-NB15-v2	Multi-class	~0.99	~0.42	~0.63	~0.44
Sharma et al. [33]	RF	UNSW-NB15	Binary	0.87	0.90	0.86	0.86

Similarly, the CNN-based models proposed by Sayed et al. [10], namely IoTCNN and MyCNN, demonstrated limited macro-level performance. For example, the IoTCNN model produced low precision and F1-scores in most attack categories, including F1 = 0.14 for Analysis and 0.13 for DoS, despite exhibiting high overall accuracy. These results suggest that while convolutional models may effectively capture local patterns, they often struggle with inter-class variance and generalization without advanced preprocessing or data augmentation strategies.

In addition to Samantaray et al. [9] and Sayed et al. [10], we include the study by Sharma et al. [33], which evaluates ensemble methods on UNSW-NB15 using a binary setting. Their best model (Random Forest) achieves Accuracy = 0.87, Macro Precision = 0.90, Macro Recall = 0.86, and Macro F1 = 0.86, whereas our binary IDS on NF-UNSW-NB15-v2 attains Accuracy ≈ 0.99 and Macro F1 ≈ 0.97. Thus, our approach improves binary macro-level performance by ≈ 11 points in F1 while also raising accuracy by ≈ 12 percentage points, reinforcing that the proposed SMOTE + PCA + diverse ML/DL pipeline generalizes better under class imbalance than strong ensemble baselines.

Overall, our findings highlight the effectiveness of ensemble models—such as XGBoost and Random Forest—in capturing complex feature relationships and addressing class imbalance. A comparative summary of results is provided in Table X.

VI. Conclusion

This work developed and empirically validated an end-to-end intrusion-detection pipeline that couples rigorous preprocessing (StandardScaler normalization, SMOTE oversampling, and PCA dimensionality reduction) with five classifiers (XGBoost, RF, KNN, DNN, and LSTM) evaluated on the "NF-UNSW-NB15-v2" dataset. The experiments covered both binary (normal vs attack) and nine-class attack identification tasks, using a unified hyper-parameter budget to ensure fair comparison.

Results underscore the strength of ensemble trees for tabular network-flow data: XGBoost and Random Forest achieved high metrics (≈ 0.9960 for accuracy, precision, recall, F1, and AUC) in binary detection, while XGBoost retained a competitive macro F1-score (0.71) and a balanced confusion matrix in multi-class testing. Deep models matched overall accuracy but lagged in macro scores, indicating residual sensitivity to minority classes. Crucially,RF and XGBoost's inference latency—on the order of milliseconds per thousand flows—demonstrates that top-tier accuracy can coexist with real-time throughput, making the approach deployable in production networks.

Although the pipeline raises the state of the art for this dataset, two limitations remain: extremely rare classes such as Worms and Analysis still risk misclassification despite SMOTE, and features were restricted to NetFlow attributes. Future work should explore cost-sensitive or generative resampling, integrate payload-level and temporal-correlation features, and test hybrid architectures (e.g., LSTM embeddings feeding XGBoost) under online-learning and edge-deployment constraints. Overall, the study provides a reproducible blueprint and a strong baseline for practitioners seeking accurate, low-latency, and resource-efficient intrusion detection in modern networked environments.

Acknowledgment

The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP2/318/46.

REFERENCES

[1] L. Cadieux, "20 Shocking Cybercrime Statistics: 2024 Edition," The Devolutions Blog, 4 July 2024. [Online]. Available: https://blog.devolutions.net/2024/07/20-shocking-cybercrime-statistics-2024-edition/?utm_source=chatgpt.com. [Accessed 1 Mars 2025].

[2] M. Sweney, "BT identifying 2,000 signals a second indicating possible cyber-attacks," The Guardian, 12 September 2024. [Online]. Available: https://www.theguardian.com/business/2024/sep/12/hackers-weaponising-ai-for-cybercrime-bt-warns?utm_source=chatgpt.com. [Accessed 1 Mars 2025].

[3] O. Ogundairo and P. Broklyn, "Machine Learning Algorithms for Intrusion Detection Systems," Journal of Cyber Security, 2024.

[4] S. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," Journal of Big Data, vol. 7, no. 1, p. 105, 2020.

[5] V. Kumar, D. Sinha, A. Das, D. S. Pandey and R. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," Cluster Computing, vol. 23, 2020.

[6] S. More and R. Kalkundri, "Evaluation of deceptive mails using filtering & WEKA," in 2015 International Conference on Innovations in Information,Embedded and Communication Systems (ICIIECS), 2015.

[7] R. Tahri, Y. Balouki, A. Jarrar and L. Abdellatif, "Intrusion Detection System Using machine learning Algorithms," ITM Web of Conferences, vol. 46, 2022.

[8] U. S. Musa, M. Chhabra, A. Ali and M. Kaur, "Intrusion Detection System using Machine Learning Techniques: A Review," in Int. Conf. Smart Electron. Commun. (ICOSEC 2020), 2020.

[9] M. Samantaray, R. C. Barik and A. K. Biswal, "A comparative assessment of machine learning algorithms in the IoT-based network intrusion detection systems," Decision Analytics Journal, vol. 11, p. 100478, 2024.

[10] N. Sayed, M. Shoaib, W. Ahmed, S. N. Qasem, A. M. Albarrak and F. Saeed, "Augmenting IoT Intrusion Detection System Performance Using Deep Neural Network," Computers, Materials & Continua, vol. 72, no. 2, p. 3511–3534, 2022.

[11] D. Bzdok, M. Krzywinski and N. Altman, "Machine learning: supervised methods," Nat. Methods, vol. 15, p. 5–6, 2018.

[12] S. Uddin, I. Haque and H. Lu, "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," Sci Rep, vol. 12, no. 6256, 2022.

[13] S. He, B. Li, H. Peng, J. Xin and E. Zhang, "An Effective Cost-Sensitive XGBoost Method for Malicious URLs Detection in Imbalanced Dataset," IEEE Access, vol. 9, pp. 93089-93096, 2021.

[14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in 22nd ACM SIGKDD International Conference, 2016.

[15] S. Jukic, M. Saracevic, A. Subasi and J. Kevric, "Comparison of ensemble machine learning methods for automated classification of focal and non-focal epileptic EEG signals," Mathematics, vol. 8, no. 9, p. 1481, 2020.

[16] S. Zhang, Z. Shen, and H. Yang, "Deep Network Approximation: Achieving Arbitrary Accuracy with Fixed Number of Neurons," Journal of Machine Learning Research, vol. 23, no. 276, pp. 1–60, 2022.

[17] J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," Machine Learning Mastery, 20 August 2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. [Accessed 1 March 2025].

[18] F. Huang, X. Li, C. Yuan, S. Zhang, J. Zhang and S. Qiao, "Attention-Emotion-Enhanced Convolutional LSTM for Sentiment Analysis," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 9, pp. 1-14, 2021.

[19] "Machine Learning-Based NIDS Datasets," The University of Queensland, [Online]. Available: https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA1. [Accessed 1 March 2025].

[20] M. Sarhan, S. Layeghy and M. Portmann, "Towards a Standard Feature Set for Network Intrusion Detection System Datasets," arXiv, 2021.

[21] Nour Moustafa and Jill Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," Military Communications and Information Systems Conference (MilCIS), 2015.

[22] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, Spambase Dataset, UCI Machine Learning Repository, 1999.

[23] I. Androutsopoulos et al., "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages," SIGIR Conference, 2000.

[24] B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," European Conference on Machine Learning (ECML), 2004.

[25] I. Androutsopoulos et al., PU1 and PU2 corpora for spam filtering experiments, 2000.

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," IEEE Symposium on CISDA, 2009.

[27] KDD Cup 1999 Dataset. UCI Machine Learning Repository.

[28] J. Song and H. Kim, "Network intrusion detection based on semi-supervised learning and clustering," Expert Systems with Applications, 2013.

[29] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks," IEEE ICC, 2015.

[30] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," ICISSP, 2018.

[31] M. Fernandez, J. Garcia, A. Sanz, and J. E. Diaz-Verdejo, "UGR'16: A new dataset for evaluation of IDSs," Computer Networks, 2018.

[32] D., Elreedy, A. F., Atiya, & F., Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning," *Machine Learning*, 113:4903–4923, 2024.

[33] N. Sharma, N. S. Yadav, and S. Sharma, "Classification of UNSW-NB15 dataset using Exploratory Data Analysis using Ensemble Learning," EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, vol. 8, no. 29, e4, Oct. 2021.