Al-Aqsa University

Faculty of Computing and Information System

Engineering and Networks Department

Advancing Network Intrusion Detection with the LycoS-Unicas-IDS 2018 Dataset: Tackling Imbalanced Big Data with Machine Learning

Presented to the Faculty of Computer and Information Technology,

Al-Aqsa University.

In Complete Fulfillment of the Requirements of Networks and Mobile Technology Bachelor Degree.

By:

Besan Ayman Abu Zubaida.

Supervised By:

Dr. Hazem El-baz.

Gaza, Palestine.

Spring 2024.

ABSTRACT

The increasing prevalence of cyber threats has made Network Intrusion Detection Systems (NIDS) essential for ensuring the security of interconnected systems. This project aims to address the challenges of big and imbalanced datasets by utilizing the LycoS-Unicas-IDS 2018 dataset, a novel dataset introduced by Cantone et al., which remains largely underutilized in network intrusion detection research. The methodologies employed include Random Under Sampling (RUS), Equal Class Distribution Under Sampling, and Stratified Under Sampling to address data imbalance, Stacking Feature Embedding (SFE) based on clustering results, and Principal Component Analysis (PCA) for dimensionality reduction. The model is trained using four machine learning algorithms: Decision Tree (DT), Random Forest (RF), Extra Trees (ET), and Extreme Gradient Boosting (XGB).

The experimental results demonstrate the efficacy of the proposed approach. For multiclass classification, Equal Class Distribution Under Sampling and Stratified Under Sampling were applied separately, and all four models (DT, RF, ET, XGB) achieved high performance, with accuracies exceeding 98%. Similarly, in binary classification, both Equal Class Distribution Under Sampling and Stratified Under Sampling approaches resulted in strong model performance, with accuracy, F1 score, precision, and recall surpassing 99%. These results indicate that the proposed approach is capable of effectively detecting network intrusions with exceptional precision and reliability across various attack scenarios.

By leveraging the LycoS-Unicas-IDS 2018 dataset, this project ensures the system's capability to handle both known and unknown threats, surpassing existing benchmarks in network intrusion detection. These findings highlight significant progress in the field, demonstrating that the suggested methodology can accurately monitor and identify network traffic intrusions, thereby blocking potential threats and enhancing overall network security in diverse and evolving threat landscapes.

ACKNOWLEDGEMENT

My sincere thanks go to my project supervisor Dr. Hazem Elbaz for his guidance and support throughout the project. His valuable insights and constructive feedback were critical in shaping the direction of this project. I am grateful for the unwavering support of my family and friends who provided encouragement, motivation, and understanding during the ups and downs of the research journey. Finally, I would like to acknowledge the academic community whose scholarly work has laid the foundation for this research. Their contributions have provided invaluable insights, perspectives, and methodologies that have greatly informed and enriched this study.

Table of Contents

List of Figures

Figure Name ***Page***

List of Tables

Table Name ***Page***

List of Abbreviations

IDS: Intrusion Detection System.

ML: Machine Learning.

LycoS18: LycoS-Unicas-IDS2018.

CSE: Communications Security Establishment.

CIC: Canadian Institute for Cybersecurity.

SFE: Stacking Feature Embedded.

RO: Random Oversampling.

RUS: Random Under Sampling.

PCA: Principal Component Analysis.

TCP: Transmission Control Protocol.

ICMP: Internet Control Message Protocol.

PCAP: Packet Capture.

CIA: Confidentiality, Integrity, and Availability.

HIDS: Host-based IDS.

NIDS: Network-based IDS.

DoS: Denial of Service.

DDoS: Distributed Denial of Service.

MANET: Mobile Ad hoc Networks.

WSN: Wireless Sensor Networks.

DL: Deep Learning.

SMOTE: Synthetic Minority Oversampling Technique.

ET: Extra Trees.

ELM: Extreme Learning Machine.

XGB: Extreme Gradient Boosting.

DT: Decision Tree.

ANN: Artificial Neural Network.

LR: Logistic Regression.

KNN: K-Nearest Neighbor.

SVM: Support Vector Machine.

IG: Information Gain.

GR: Grain Ratio.

RF: Random Forest.

MQTT: Message Queuing Telemetry Transport.

PART: Partial Decision Tree.

DNN: Deep Neural Networks.

RNN: Recurrent Neural Networks.

CNN: Convolutional Neural Networks.

IoT: Internet of Things.

LSTM: Long Short-Term Memory.

ACO: Ant Colony Optimization.

GMM: Gaussian Mixture Model.

WDLSTM: Weight-Dropped Long Short-Term Memory.

ASM: Attribute Selection Measure.

GM: Gaussian Mixture.

SFE: Stacking Feature Embedded.

RR: Reduction Ratio

GI: Gini index.

MSE: Mean Squared Error.

TP: True Positive.

TN: True Negative.

FP: False Positive.

FN: False Negative.

AUC: area under the curve.

ROC: Receiver Operating Characteristic.

CV: cross validation.

mRMR: Minimum Redundancy Maximum Relevance.

SDN: Software-Defined Networking.

1 Introduction

1.1 Overview

Cybersecurity has become a global imperative in today's digital age, driven by the critical need to protect systems from unauthorized and malicious intrusions [1]. These intrusions can range from data breaches and information theft to threats that compromise the integrity and functionality of systems. Safeguarding against such threats is essential for ensuring the smooth operation of systems, protecting sensitive data, and maintaining user trust [2]. Intrusion Detection Systems (IDS) have historically been a cornerstone of perimeter security [3, 4]. These systems are designed to identify and respond to suspicious or malicious activities within a network or system. However, traditional signature-based IDS methods, relying on known attack patterns and signatures, have proven inadequate in the face of constantly evolving and sophisticated cyber threats [5, 6]. To address the limitations of traditional IDS, the cybersecurity community has turned to Machine Learning (ML) as a promising solution. ML-enabled IDS leverage behavior analysis to detect anomalies and threats, offering the potential for significantly higher accuracy and faster detection times [7, 8]. This paradigm shift in intrusion detection has the potential to not only enhance security but also reshape the privacy landscape. However, this transition to ML-based intrusion detection has raised concerns regarding both privacy and data science [9, 10]. ML algorithms, while effective at identifying threats, often require access to sensitive data. Balancing the need for security with privacy concerns is a challenge that demands innovative and ethical solutions [11]. However, A major challenge for ML-based IDS systems is the reliance on high-quality datasets that accurately represent real-world traffic, including both benign and attack scenarios. Addressing class imbalance and ensuring high classification accuracy remain critical for effective model performance.

1.2 Problem Definition & Formalization

The development of robust and scalable IDS is paramount in the face of increasingly sophisticated cyber threats. While traditional signature-based IDS systems have been effective in detecting known attacks, their limitations become apparent when dealing with novel and evolving threats. Anomaly-based IDS, on the other hand, can detect zero-day attacks but suffer from high false positive rates and difficulties in adapting to evolving normal behaviour [12].

However, it’s worth noting that many contemporary ML-IDS are often limited by their reliance on small, outdated, and imbalanced datasets for model development [13, 14, 15]. These limitations hinder their ability to accurately detect and respond to modern cyberattacks, which are becoming increasingly complex and adaptive.

My research focuses on addressing these limitations by employing the latest LycoS-Unicas-IDS2018 (LycoS18) dataset, an enhanced Communications Security Establishment-Canadian Institute for Cybersecurity) CSE-CIC-IDS2018) dataset with refined feature extraction techniques [16].

The primary problem lies in designing a ML-based IDS that can effectively detect intrusions on imbalanced and large datasets. **This research aims to enhance the detection capabilities of IDS by integrating techniques like Stacking Feature Embedded (SFE), Random Under Sampling(RUS) and Principal Component Analysis (PCA) to improve model robustness and performance.**

1.3 Research Objectives

1.3.1 Main Objectives

The main objective of this research is to develop a highly accurate and efficient ML-based IDS that can handle imbalanced and large datasets. The model aims to improve detection rates, reduce false positives, and adapt to new threats, thus offering a comprehensive solution for network security.

1.3.2 Secondary Objectives

This project's secondary objectives include:

1. evaluating the performance of various machine learning algorithms to identify the most suitable model for network intrusion detection.
2. Additionally, it aims to address class imbalance within the dataset through appropriate techniques, ensuring robust results.
3. Furthermore, the project also seeks to use advanced data preprocessing techniques, such as Random Under-Sampling (RUS) and Stacking feature embedded - Principal Component Analysis (SFE-PCA), to enhance model performance.

1.4 Research Motivation

The motivation for this research stems from the growing need for more sophisticated intrusion detection mechanisms capable of countering modern-day cyber threats. While earlier datasets like KDD-Cup99 [17] played a pioneering role, it has shortcomings that affect the performance of evaluated systems, leading to poor evaluation of anomaly detection approaches [18]. Even NSL-KDD, an improvement over KDD-Cup99, suffers from issues like skewed target classes, which can hinder the performance of classifiers on certain attack types [19]. While the UNSW-NB15 [20] dataset includes more contemporary attacks than NSL-KDD [21], the CIC-IDS2017 [20] dataset generally offers superior performance in terms of accuracy and false positive rates [22]. Newer offerings, such as CIC-IDS2017 incorporated modern attacks but faced data quality issues like redundant features and misclassified protocols [16].

The LycoS18 [23] dataset emerges as a solution. It is a novel dataset created using LycoStand [24], a reliable feature extractor, to process the well-regarded CSE-CIC-IDS-2018 [20] Packet Capture (PCAP) files. This process rectifies errors present in CIC-IDS2018 and addresses class imbalance by balancing benign and malicious traffic. This enhanced dataset fosters a more realistic foundation for training machine learning models, enabling real-time anomaly detection and improved performance across diverse attack scenarios [16].

This research capitalizes on the strengths of LycoS18 to contribute to advancements in network security. By utilizing a cleaner, more reliable dataset compared to error-prone predecessors like CIC datasets, this work aims to push the boundaries of machine learning-based intrusion detection systems. Building upon past studies and addressing limitations in existing datasets, this research seeks to develop more effective and reliable solutions for

detecting complex, large-scale attacks.

1.5 Time Plan

The project will be carried out over the following phases:

Phase 1: Problem Definition and Literature Review

This phase involves identifying the key challenges in network intrusion detection, reviewing existing solutions, and understanding the capabilities of the LycoS-Unicas-IDS 2018 dataset.

Phase 2: Dataset Analysis and Preprocessing

In this phase, the dataset will be explored in detail to understand its structure and distribution. Key preprocessing tasks, such as handling missing values, addressing data imbalance through random under sampling, and feature selection or engineering will be conducted.

Phase 3: Model Development and Training

Various machine learning models, including Decision Tree (DT), Random Forest (RF), Extra Trees (ET), and Extreme Gradient Boosting (XGB), will be developed and trained using the processed dataset. Techniques like Stacking Feature Embedding (SFE) and Principal Component Analysis (PCA) will be applied to optimize feature representation and dimensionality.

Phase 4: Model Evaluation and Performance Analysis

The trained models will be evaluated using key performance metrics such as accuracy, precision, recall, F1 score, and Area under curve (AUC). Comparative analysis of the models will be performed to determine the best-performing approach for both multilabel and binary classifications.

Phase 5: Results Interpretation and Documentation

This phase focuses on interpreting the findings, documenting the methodology, results, and conclusions, and preparing for the final presentation or project defence.

To ensure smooth progress through these phases, the following 12-week time plan has been developed, outlining key tasks and milestones for each week as illustrated in Figure 1.

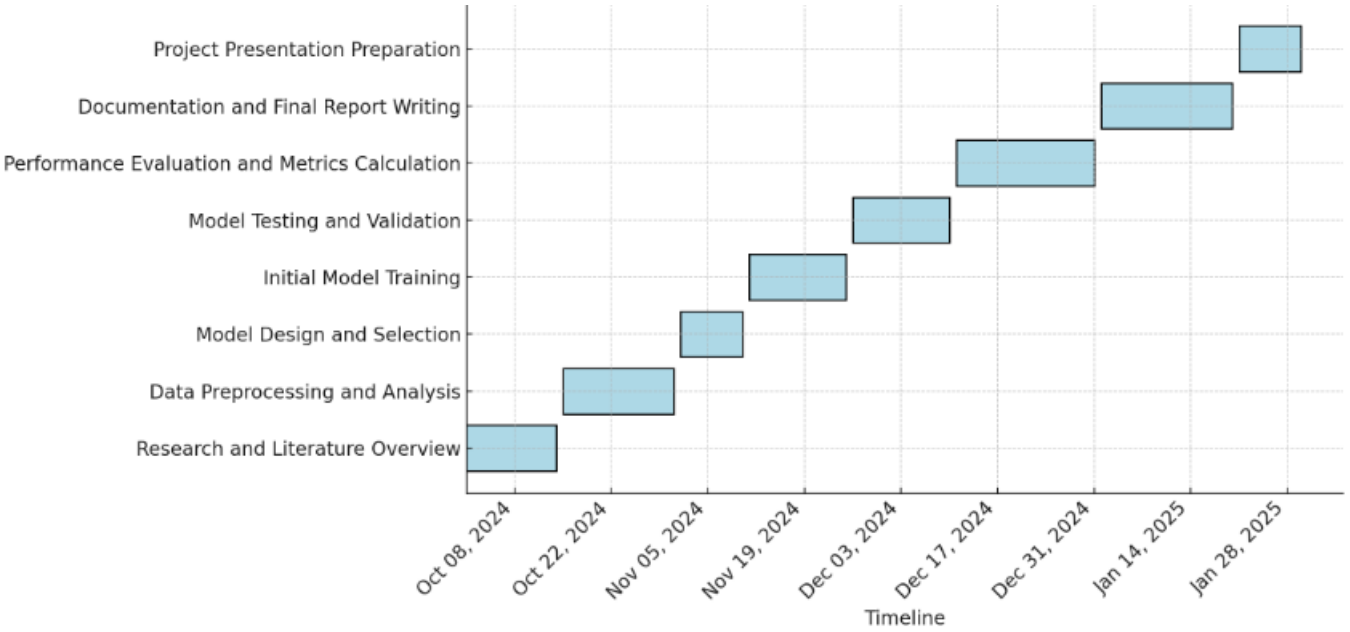

Figure 1: Project Time Plan

1.6 Report Structure

The remainder of this report is organized as follows: Chapter 2 presents the background and context of IDS and cyber-security attacks, emphasizing the role of ML in intrusion detection. Chapter 3 offers a comprehensive literature review on network intrusion detection, covering ML, DL, and hybrid approaches, along with a discussion of the limitations in existing works. Chapter 4 outlines the proposed framework and methodology for model development, including detailed data preprocessing techniques. It also presents an overview of the machine learning algorithms used for intrusion detection, specifically designed to address the challenges posed by large and imbalanced datasets. Chapter 5 describes the experimental setup and evaluation procedures. Chapter 6 presents the results and discussion, offering a comparative analysis with previous studies and assessing time complexity. Chapter 7 concludes the research, summarizing key findings and suggesting potential areas for future work. Finally, Chapter 8 lists all references cited throughout the report.

2 Background

2.1 IDS

The rapid expansion of network interconnections has led to a corresponding growth in the cyber threat landscape, attracting the interest of an increasing number of cyber attackers. This has resulted in the disruption of essential services with significant economic consequences. Globally, cybercrime is estimated to have an impact of around 1 trillion dollars in 2020, with an increase of more than 50% compared to 2018 [25]. As the volume and complexity of cyberattacks continue to rise, protecting network infrastructures has become increasingly challenging. One of the major reasons for this rise is the rapid evolution of attack methodologies, which outpaces traditional security mechanisms. Attackers are constantly discovering new vulnerabilities and creating more effective ways to bypass security defenses, leading to substantial damage for targeted networks. These threats can violate the computer security triad: confidentiality, integrity, and availability (CIA). An intrusion is an attempt to compromise the CIA or evade the security mechanism of a computer or a network [26]. To mitigate these threats, to mitigate these threats, IDS have been developed, designed to monitor network traffic and identify malicious activity that could compromise system integrity. IDS are crafted to uncover and respond to suspicious or malicious activities within a network or system.

IDS can be classified based on methodology into anomaly detection and misuse-based detection. Anomaly detection methods model the user behavior to build a normal (standard) behavior. This method observers the behavior of a user over a period of time and tries to build a model that is close to the normal behavior of the user. Whenever there is an event that deviates from the normal behavior, the IDS detects it as suspicious behavior and raises an alarm which makes it suitable for detecting zero-day attacks. Two major disadvantages are related to this model: 1) the inability to deal with the evolving of normal user behavior, and 2) a high false-positive rate [27]. On the other hand, in the misuse detection or signature-based method, captured events are compared with known attacks or threats for the detection of possible intrusions. The known attacks are often referred to as, signatures [28].This method is more accurate and raises fewer false alarms compared to the anomaly detection method. However, it is not useful for detecting new attacks [29]. To address the shortcomings of traditional IDS, the cybersecurity has turned its attention to ML as a promising solution [29]. IDS can also be classified based on the monitored platform into two types: host-based IDS (HIDS) and network-based IDS (NIDS). HIDS monitors and analyzes the activities on the system (host) where it is deployed. NIDS can be grouped into three categories: statistics-based, knowledge-based, and ML-based [30].The statistics-based paradigm concerns the meticulous examination of each record within a dataset, with the goal of building a statistical model that encompasses the established norms of user behavior within a network [31]. In contrast, the knowledge-based approach is based on attempting to identify the requested actions by leveraging existing system data, including protocol specifications and network traffic instances. Knowledge-based NIDS, guided by human-defined rules, deduces typical system activities and identifies deviations from this benchmark as potential intrusions [32]. A knowledge-based method to attain traffic classification is Deep Packet Inspection. This technique involves examining the content and structure of individual packets, allowing for detailed inspection of packet headers, payload data, and protocol-specific information.

2.2 ML-based IDS

ML-based IDS can automatically learn from network traffic patterns, improving their ability to detect both known and unknown threats, including zero-day attacks. By training ML models on large datasets, these systems can identify

subtle anomalies that may indicate a cyberattack. The application of ML has proven especially valuable in reducing false positives and improving the detection of complex, evolving attack strategies. ML-enabled IDS leverages behavior analysis to detect anomalies and threats, offering the potential for significantly higher accuracy and faster detection times [7, 8]. ML algorithms, while effective at identifying threats, often require access to sensitive data. Balancing the need for security with privacy concerns is a challenge that demands innovative and ethical solutions [33].

However, it's worth noting that many contemporary ML-IDS solutions tend to be limited by their reliance on small, outdated and balanced datasets for model development [13, 34]. The focus on these smaller, often outdated datasets, coupled with imbalances in the data distribution, while facilitating preprocessing and training with diverse ML algorithms, raises questions regarding the practical applicability of these models in real-world scenarios, specifically when dealing with big data. The achievable accuracy of such models often hinges on the intricacies of dataset preprocessing and the selection of suitable algorithms, adding an additional layer of complexity to their effectiveness [35, 36]. For a successful detection of new attacks, a huge amount of data must be used for building a model for what is normal and what is an anomaly. This raised attention to the application of supervised ML algorithms for the efficient understanding of the data and for building a predictive model that can predict new attacks with high-performance rates. The high dimensionality of the data is another issue, as the increase of the feature space with a relatively low number of data, (records) can lead to a "curse of dimensionality" problem that affects the results of classification, and this attracted attention for the application of feature selection and feature reduction for an enhancement of classification [26].

To support the development of more effective ML-based IDS, several datasets have been created such as KDD'99 [17], NSL-KDD, UNSW-Nb15 [20], CICIDS2017 [20], CICIDS2018 [20], LycoS-IDS 2017 and LycoS-Unicas-IDS 2018 [23]. Table 1 illustrates a brief description of each dataset.

Table 1. Description of IDS Datasets.

Data set	Description
KDD'99	It is generated using a simulation of normal and attacks traffic in a military environment (US AirForce LAN). It contains nine weeks of simulation in rat tcpdump files. The dataset is characterized using 41 features related to intrinsic, content, and traffic. Four types of attacks are simulated: Denial of Service (DoS), Prob, U2R, and R2L.
NSL-KDD	It is a modification to the KDD'99 dataset with solving the problems of redundancy, duplicates, the imbalance of data.
UNSW-Nb15	It was created using the IXIA PefectStorm tool to extract normal and attack network traffic based on 100 GB of raw network traffic. It is characterized using 49 features. It consists of around 175 thousand records for training and around 82 thousand records for testing. There are nine types of attacks: Fuzzers, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm.
CICIDS2017	It was created in an emulated environment in a 5-day period. It contains traffic in packet flow and bidirectional flow. 80 features are extracted. Attacks involve: Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS.
CSE-CIC-IDS2018	Released by the Canadian Institute for Cybersecurity in collaboration with CSE, this dataset expanded the traffic generation process using 500 machines on AWS. It includes raw network packets (PCAPs) and CSV files of bidirectional flows, with some features removed for consistency with CIC-IDS2017.
LycoS-IDS2017	Created to correct anomalies in the CIC-IDS2017 dataset, this dataset involved meticulous feature extraction using the LycoStand tool. It addressed issues like feature

	duplication, miscalculations, and misclassification of protocols, ensuring more accurate labeling of network flows.
LycoS-Unicas-IDS2018	Developed using the LycoStand feature extractor on CSE-CIC-IDS2018 PCAP files, this dataset contains 46,950,511 benign samples, which were subsampled to 10 million to balance classes. It aims for optimal labeling precision while addressing previously identified annotation challenges.

Thus, the integration of ML into IDS, supported by datasets such as LycoS18, is a crucial advancement in the field of network security. These systems not only enhance the detection of traditional attacks but also improve the identification of complex, previously unknown threats, making them indispensable in today's cybersecurity landscape.

2.3 Cyber-Security Attacks

IDS are crucial for identifying and mitigating different types of cyberattacks. Cyberattacks can be categorized in various ways, including by their purpose, legal classification, severity, scope, and the types of networks they target. As Uma and Padmavathi [37] point out, these categories help in understanding the diverse range of threats in the digital landscape. One of the primary categories based on the purpose of the attack includes reconnaissance attacks, access attacks, and DoS attacks. A reconnaissance attack is a dangerous type of attack as the attacker trap victims into becoming their friend to extract sensitive information from them [38]. Access attacks, on the other hand, involve gaining unauthorized access to systems. Methods like phishing, social engineering, and man-in-the-middle attacks fall under this category, allowing intruders to exploit security weaknesses to infiltrate networks or devices. In a DoS attack, the attacker exploits the connectivity of the internet to cripple the services offered by the victim site, simply, by flooding a victim site with a high number of requests. It can be a single-source attack or a multi-source attack, where, in the latter, it is referred to as distributed denial of service attack (DDoS) [39]. Another important classification is based on legal distinctions, encompassing cybercrime, cyber espionage, cyber terrorism, and cyberwarfare. Cybercrime, which includes activities like identity theft, involves the illegal use of someone else's data for personal gain [40]. Cyber espionage is concerned with covertly accessing sensitive information, often related to state secrets or corporate intelligence. Cyberterrorism and cyberwarfare represent politically motivated attacks, with the former being carried out by extremist groups and the latter involving state actors in conflicts that use cyberspace as a battlefield. Attacks can also be categorized based on the severity of involvement. Active attacks aim to disrupt or alter system operations, such as in the case of buffer overflow attacks or spoofing, while passive attacks involve monitoring or gathering data without altering the system, for instance, through keylogging [41]. This distinction helps in determining the aggressiveness of an attack and the type of defense mechanism required. In terms of intent, attacks are often divided into malicious and non-malicious categories. Malicious attacks, such as those carried out using viruses, worms, or botnets, are intentional and aimed at causing harm or disruption. On the contrary, non-malicious attacks typically occur due to human error or lack of training and are unintentional, often resulting in minor data breaches or system vulnerabilities. Finally, attacks can be grouped based on the network types they target. Mobile ad hoc networks (MANET) and wireless sensor networks (WSN) are common targets [42]. In MANETs, attacks such as black hole or flood rushing attacks aim to disrupt the network by hijacking or misdirecting traffic. WSNs are also vulnerable to application-layer and network-layer attacks, which exploit various weaknesses in these systems.

3 Literature Review

3.1 State of the Art

In recent years, the application of ML techniques in cybersecurity, particularly in NIDS, has become a critical area of research. ML's ability to uncover hidden patterns in network traffic makes it a powerful tool for identifying anomalies and distinguishing between malicious and legitimate activities. Consequently, various ML, deep learning (DL), and hybrid models have been proposed and refined by researchers to enhance IDS effectiveness.

3.1.1 ML Approaches

Moualla et al. [43] developed a scalable, multiclass ML-based network IDS using the UNSW-NB15 dataset. They addressed the issue of class imbalance using the Synthetic Minority Oversampling Technique (SMOTE) and employed an Extra Trees (ET) classifier combined with an Extreme Learning Machine (ELM) for classification, achieving an impressive accuracy of 98.43%.

Kasongo and Sun [44] presented a filter-based feature-dropping technique on the UNSW-NB15 IDS dataset, employing the Extreme Gradient Boosting (XGB) algorithm, and assessed its performance using various predictive algorithms, including Decision Tree (DT), Artificial Neural Network (ANN), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Support Vector Machine (SVM). They demonstrated that their approach significantly enhanced binary accuracy, increasing it from 88.13% to 90.85%. The overall accuracy rates for binary were 90.85% for DT, 84.4% for ANN, 77.64% for LR, 84.46% for KNN, and 60.89% for SVM. For multiclass, the accuracy rates were 67.57% for DT, 77.51% for ANN, 65.29% for LR, 72.30% for KNN, and 53.95% for SVM, all of which were evaluated using the 19 optimal selected features.

Nimbalkar and Kshirsagar [45] took a different approach by employing Information Gain (IG) and Grain Ratio (GR) for feature selection on the KDDCUP'99 and BOT-IOT datasets, using the JRip classifier. Their model achieved accuracy rates of 99.99% for BOT-IOT and 99.57% for KDDCUP'99.

Additionally, Kumar et al. [46] enhanced detection of DoS attacks using the UNSW-NB15 dataset. By applying IG for feature selection and the C5 classifier, they achieved a remarkable accuracy rate of 99.37%.

Other researchers have explored clustering-based methods. Ahmad et al. [47] introduced a feature clustering ML model to classify flow, Message Queuing Telemetry Transport (MQTT), and TCP data, achieving accuracy rates of 98.67% and 97.37% for binary and multiclass classifications, respectively, on the UNSW-NB15 dataset when trained using RF.

Kshirsagar and Kumar [48] proposed a feature selection technique using IG Ratio, Correlation Ratio, and ReliefF for the CICIDS2017 and KDDCUP'99 datasets, significantly reducing feature dimensionality while maintaining accuracy rates of 99.95% and 99.32% using the Partial Decision Tree (PART) algorithm. The summary of related papers can be found in Table 2.

Table 2: Related Work Summary of Various ML Techniques.

SL. NO.	Author	Algorithms	Dataset	Accuracy	
				Binary	Multi-class
1	[43]	ELM	UNSW-NB15		98.43
2	[44]	DT	UNSW-NB15	90.85	67.57
		ANN		84.4	77.51
		LR		77.64	65.29
		KNN		84.46	72.30
		SVM		60.89	53.95
3	[45]	IG+GR+JRip	KDDCUP'99		99.57
			BOT-IOT		99.99
4	[46]	C5	UNSW-NB15		99.37
5	[47]	RF	UNSW-NB15	98.67	97.37
6	[48]	PART	CICIDS2017		99.95
			KDDCUP'99		99.32

3.1.2 DL Approaches

DL has also been prominently applied in IDS research due to its ability to automatically learn complex representations from large datasets. Aleesa et al. [49] explored the effectiveness of deep neural networks (DNN), recurrent neural networks (RNN), and ANN on the UNSW-NB15 dataset. Their results demonstrated that DNN and ANN outperformed RNN, with DNN achieving an accuracy of 99.22% for binary classification and 95.9% for multiclass classification.

Further advancing DL applications, Choudhary and Kesswani [50] proposed a DNN-based IDS model to identify Internet of Things (IoT)-related attacks, achieving an accuracy of 91.50% on the UNSW-NB15, KDDCUP'99, and NSL-KDD datasets.

Similarly, Kim et al. [51] used convolutional neural networks (CNN) to focus on DoS attacks, demonstrating that CNN models consistently outperformed RNN models on the KDDCUP'99 and CSE-CIC-IDS2018 datasets, with accuracy rates exceeding 99% for binary and multiclass classifications.

Al and Dener [52] combined long short-term memory (LSTM) and CNN architectures with the SMOTE and Tomek Link referred to as the STL method to handle imbalanced datasets, achieving an outstanding accuracy of 99.83% for multiclass classification and 99.17% for binary classification using the CICIDS-001 and UNSW-NB15 datasets. The summary of related papers can be found in Table 3.

Table 3: Related Work Summary of Various DL Techniques

SL. NO.	Author	Algorithms	Dataset	Accuracy	
				binary	Multi-class
1	[49]	DNN	UNSW-NB15	99.92	95.9
		RNN		85.42	85.4
		ANN		99.26	97.89
2	[50]	DNN	KDDCUP'99	91.50	
			NSL-KDD	91.50	
			UNSW-NB15	91.50	
3	[51]	RNN	KDDCUP'99	99	93
			CSE-CIC-IDS2018		65
		CNN	KDDCUP'99	99	99
			CSE-CIC-IDS2018		91.5
4	[52]	LSTM+CNN	CICIDS-001		99.83
			UNSW-NB15	99.17	

3.1.3 Hybrid Approaches

Hybrid approaches, which combine the strengths of multiple algorithms, have also shown promise in improving IDS performance. Bhardwaj et al. [53] presented a hybrid strategy that combines a DNN model with Ant Colony Optimization (ACO) for learning premium hyperparameters for successful DNN classification in a cloud setting. DNN detects attacks more accurately by the usage of ideal settings. They used the CIC-IDS2017 dataset and got a good performance. The detection and accuracy performance are both superior to state-of-the-art approaches, at 95.74% and 98.25%, respectively.

Zhang et al. [54] introduced a novel approach to address the issue of imbalanced intrusion detection by developing the SGM-CNN model, which integrates the SMOTE with a Gaussian Mixture Model (GMM). Their model was evaluated using the UNSW-NB15 and CIC-IDS2017 datasets, achieving impressive results with an accuracy of 99.74% for binary classification and 96.54% for multiclass classification on the UNSW-NB15 dataset. For the CIC-IDS2017 dataset, the model reached a detection rate of 99.85%.

Similarly, Hassan et al. [55] proposed a hybrid DL model combining CNN with Weight-Dropped Long Short-Term Memory (WDLSTM) to enhance network intrusion detection. The CNN extracted key features, while the WDLSTM handled long-term dependencies and reduced gradient vanishing issues. Their model, tested on the UNSW-NB15 dataset, achieved overall accuracy rates of 97.17% for binary classification and 98.43% for multiclass classification. The summary of related papers can be found in Table 4.

Table 4: Related Work Summary of Various Hybrid Approaches.

SL. NO.	Authors	Algorithms	Dataset	Accuracy	
				Binary	Multi-class
1	[53]	DNN+ACO	CIC-IDS2017		98.25
2	[54]	SGM+CNN	UNSW-NB15	99.74	96.54
			CIC-IDS2017		99.85
3	[55]	WDLSTM+CNN	UNSW-NB15	97.17	98.43

3.2 Limitations of Existing Works

Many existing intrusion detection models rely heavily on older datasets such as KDDCUP'99 and NSL-KDD, which present significant limitations in capturing modern cyber threats. These datasets, while historically significant, lack contemporary attack scenarios and do not represent the evolving nature of cyber threats. To address these limitations, the CIC-IDS2017 and CIC-IDS2018 datasets were introduced, offering more recent and relevant attack scenarios. However, despite their widespread use, these newer datasets also have significant issues that the LycoS datasets aim to resolve. Problems like feature duplication, incorrect calculations, and protocol misclassification were prevalent in CIC-IDS2017, leading to the creation of the LycoS17 dataset. For instance, redundant features such as 'Average Packet Size' and 'Packet Length Mean' were corrected, and a more accurate protocol detection mechanism was implemented to resolve misclassifications in ICMP and other protocols. Additionally, the LycoS18 dataset, derived from CIC-IDS2018, introduced the LycoStand tool for more precise feature extraction and improved labeling accuracy. It also addressed class imbalance, a major challenge in CIC-IDS2018, by reducing benign samples to improve computational efficiency and model training. These enhancements make LycoS17 and LycoS18 superior datasets for modern network intrusion detection research, offering more reliable and cleaner data for machine learning applications.

A further limitation is the failure of many existing works to incorporate data-balancing techniques. This results in models that are vulnerable to variations in performance, particularly in terms of false positives and negatives, and may yield suboptimal results in detecting true positive threats. Data imbalance is a common issue in real-world network environments, and ignoring this factor can severely compromise the effectiveness of an IDS. The need for balancing becomes even more apparent when considering large-scale, highly imbalanced datasets where attacks are significantly outnumbered by benign traffic.

Another challenge in prior works is the tendency to utilize the complete feature set during experimentation. While this may improve detection rates under controlled conditions, it also imposes high computational demands, making the models impractical for real-time anomaly detection in large-scale networks. The failure to perform proper feature selection can lead to inefficiencies, as not all features contribute equally to the classification performance. Feature selection, when done correctly, can not only reduce computational overhead but also improve model accuracy by focusing on the most relevant attributes.

Moreover, the majority of the works lack a thorough analysis of time complexity, which is critical for understanding the computational feasibility of these models in real-world applications. In practice, network administrators need solutions that not only provide high detection rates but also operate within acceptable timeframes to prevent latency in security responses.

Lastly, there is a notable absence of the SFE approach in the literature. This method, which incorporates meta-features to enhance model accuracy, has been largely overlooked, despite its potential to significantly improve the comprehensiveness and effectiveness of intrusion detection models. The omission of such advanced techniques limits the depth of intrusion detection capabilities offered by existing solutions.

4 Research Methodology

This section outlines the proposed framework and data preprocessing techniques, including feature resampling, scaling, stacking feature embedding, and feature extraction. Additionally, it provides an overview of the machine learning (ML) algorithms employed for intrusion detection. The proposed framework integrates stacking feature embedding and dimensionality reduction to address the challenges of big and imbalanced datasets, ensuring a robust and secure network by classifying incoming packets as normal or attack traffic. A schematic block diagram of the proposed paradigm is presented in Figure 2. The methodology consists of the following phases:

• Step-1: Data preprocessing begins with essential cleaning and transformation steps, including checking for missing values, removing duplicate rows, merging similar classes with low instance counts, and reducing dataset size by converting data types from `float64` to `float32` and from 'int64' to 'int32'. These steps improve data quality by addressing inconsistencies and optimizing memory usage, ensuring the dataset is ready for analysis.

• Step-2: In the feature scaling step, I use standardization for input features and label encoding for output features. Standardization ensures all input features have the same scale, while label encoding prepares the output features for ML algorithms. These transformations are critical for improving algorithm performance.

• Step-3: In the feature resampling step, I use Equal Class Distribution Under Sampling and Stratified Under Sampling to address dataset imbalance. Equal Class Distribution Under Sampling removes random samples from the majority class to achieve an equal distribution across all classes, while Stratified Under Sampling reduces the majority class while maintaining the proportional representation of each class. These methods effectively balance the dataset, ensuring optimal conditions for training machine learning models.

• Step-4: In the feature extraction step, Dimensionality reduction is achieved through Principal Component Analysis (PCA), which reduces the dataset's features while retaining essential information. By simplifying the dataset, PCA improves computational efficiency and enhances model performance.

• Step-5: In the stacking feature embedded step, I utilize the clustering results as meta-features, to enrich the dataset. These meta-features provide additional insights derived from underlying patterns and structures within the data, contributing to better model performance.

• Step-6: In this phase, the preprocessed dataset is split into training and testing subsets. I employ a k-fold cross-validation technique with k set to 10. This step ensures the model is rigorously evaluated across multiple subsets of the data, enhancing its generalization and accuracy.

• Step-7: In this step, I evaluate the model's performance for both binary class and multi class classification using four established ML algorithms: DT, RF, ET, and XGB. Each model undergoes 10-fold cross-validation to assess its effectiveness in both classification tasks and determine the most suitable algorithm for the IDS.

• Step-8: In the final phase, I assess the model's performance using a comprehensive set of evaluation metrics, including Precision, Recall, Confusion Matrix, Accuracy, F1-score, and the ROC curve. These metrics provide benchmarks for comparing the proposed model with existing approaches, facilitating the selection of the best-performing model for IDS.

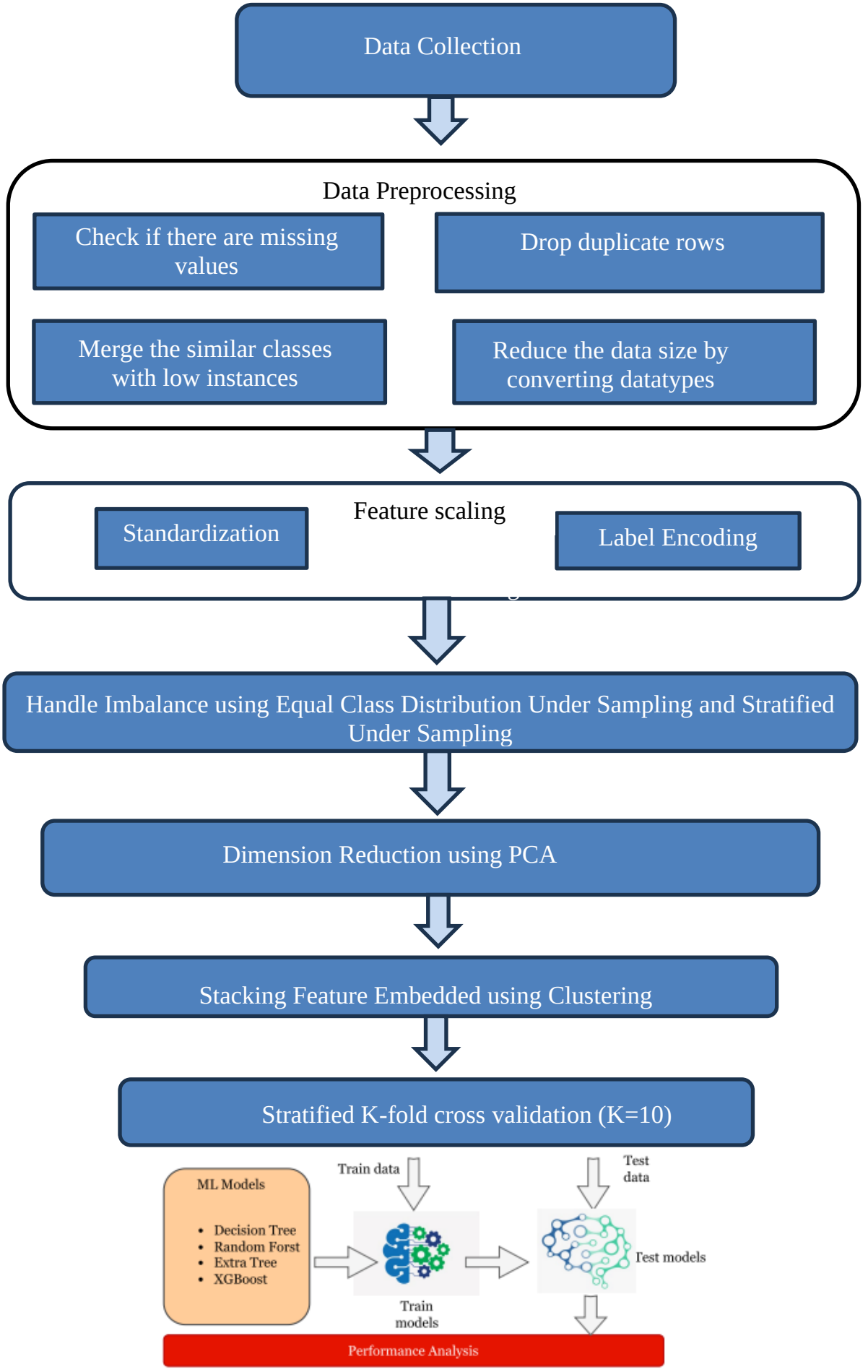

Figure 2: Proposed Framework for Stacking Feature Embedded with PCA for Intrusion Detection

Data collection

For this research, I have utilized a benchmark big dataset for my research schemes, namely LycoS-Unicas-IDS2018 [23]. This benchmark dataset is designed to reflect realistic intrusion detection system environments and includes up-to-date attack categories to detect modern threats effectively. Subsequent sections provide a detailed analysis of this dataset.

LycoS-Unicas-IDS2018

LycoS-Unicas-IDS2018 (LycoS18) [23] is a novel dataset created by us using LycoStand [56], a feature extractor designed to process raw network traffic captured in PCAP files. It was applied to the CSE-CIC-IDS2018 dataset, with flow labelling based on the official attack schedule provided on the dataset's website. While datasets like CIC-IDS2017 have faced criticism for certain annotation anomalies, achieving perfect labelling precision remains a complex challenge. Nonetheless, LycoS18 adopts the most reliable labelling methodology currently available. While certain anomalies have been identified in the annotation of the CIC-IDS2017 dataset, achieving an optimal level of labeling precision remains an intricate challenge. Nevertheless, given the currently available information, the existing labeling methodology represents the most viable approach to address this concern. For the same reason raw network traffic from February 28, 2018, and March 1, 2018, was discarded, as these days included infiltration attacks that could compromise the dataset's labelling accuracy, following similar measures adopted for LycoS-IDS2017. After feature extraction, the dataset included 46,950,511 benign samples, which were randomly subsampled to 10 million. This step was critical for mitigating class imbalance and reducing computational complexity while maintaining dataset integrity [16]. LycoS18 is publicly accessible at https://github.com/MarcoCantone/LycoS-Unicas-IDS2018.

The LycoS18 dataset was designed to address the growing demand for comprehensive datasets tailored for testing intrusion detection systems (IDSs), particularly those focused on network-based anomaly detection. Anomaly detection is a promising approach for identifying emerging threats but is hindered by the inherent complexities of cybersecurity scenarios. Effective anomaly detection systems require extensive testing, which is often limited by the shortcomings of existing datasets. Conventional datasets used for these purposes have shown limitations, stemming from privacy constraints, excessive anonymization, and a lack of representation of contemporary threat trends [29]. LycoS18 aims to overcome these shortcomings by employing a structured methodology for crafting benchmark datasets. It provides 77 features representing various aspects of network traffic and includes 13,691,268 samples, encompassing 13 distinct attack scenarios DoS Hulk, DDoS HOIC, DDoS LOIC-HTTP, FTP-Patator, DoS Slowhttptest, Bot, SSH-Patator, DoS GoldenEye, DoS Slowloris, DDoS LOIC-UDP, Web Attack – Brute Force, Web Attack – XSS, Web Attack – SQL Injection.

Enhanced Characteristics of LycoS18

The LycoS18 dataset serves as an enhanced version of the CIC2018 dataset. The attack infrastructure in CIC2018 includes 50 machines targeting an organization consisting of 5 departments with 420 machines and 30 servers. The network traffic and system logs were meticulously collected and enriched with 80 features extracted using CICFlowMeter-V3 [29]. By improving upon this foundation, LycoS18 provides a more robust and refined dataset for

IDS evaluation.

The LycoS18 dataset represents a valuable resource for advancing research in intrusion detection and supporting the practical implementation of IDSs in real-world cybersecurity environments. Its comprehensive and well-structured nature enables the development of dynamic and adaptable IDSs capable of addressing contemporary threats.
The frequency distribution of attack categories for LycoS18 is detailed in Table 5. The dataset offers researchers and practitioners a powerful tool to systematically evaluate IDSs, contributing significantly to the field of network security.

Table 5: The Frequency Distribution of Attack Categories of the LycoS-Unicas-IDS2018 Dataset

Class Name	**Occurrences**
Benign	10,000,000
DoS Hulk	1,802,966
DDoS HOIC	1,074,379
DDoS LOIC-HTTP	289,328
FTP-Patator	190,300
DoS Slowhttptest	105,550
Bot	96,154
SSH-Patator	92,648
DoS GoldenEye	26,861
DoS Slowloris	10,274
DDoS LOIC-UDP	2,382
Web Attack – Brute Force	260
Web Attack – XSS	116
Web Attack – SQL Injection	50

The distribution of attack categories as multi class and binary class, is demonstrated in Figure 3 and Figure 4.

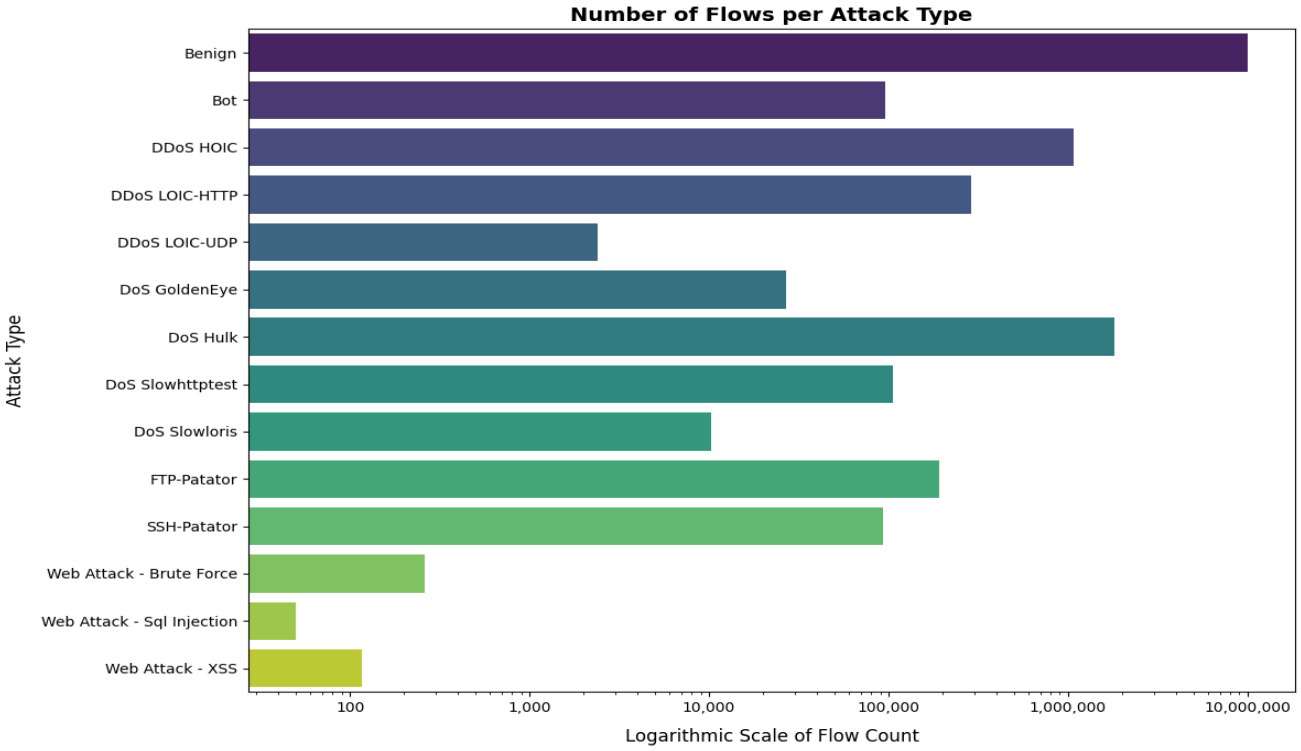

Figure 3: Multi Class Frequency Distribution of the LycoS18 Dataset

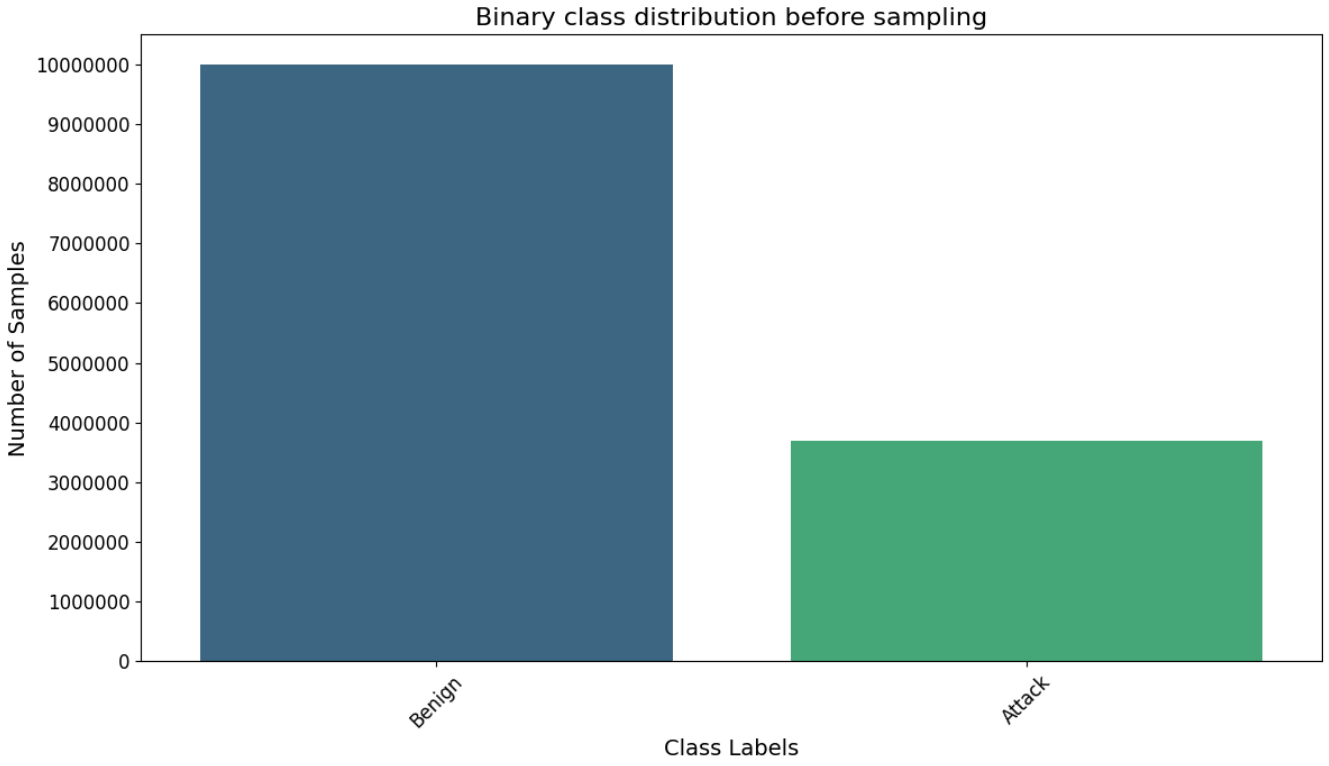

Figure 4: Binary Class Frequency Distribution of the LycoS18 Dataset

Data preprocessing

Data preprocessing is a critical step in developing any ML model. Without proper preprocessing, models can face several challenges, including invalid data handling, overfitting, error-prone predictions, and reduced accuracy. Therefore, preprocessing plays a fundamental role in ensuring the success and reliability of an ML model. In this project, various preprocessing techniques were applied to enhance the quality of the dataset and optimize the model's performance. The steps included the following:

Handling Missing Values and Removing Duplicates

The dataset was initially examined for any missing values; however, the LycoS18 dataset contained no invalid or incomplete entries that could affect the analysis. Additionally, a check for duplicate rows was performed, and all duplicates, except for the first instance, were removed. This process reduced the total number of rows from 13,691,268 to 10,468,590, ensuring the dataset remained clean, representative, and free from redundancy.

Merging Classes with Low Instances

One critical step in the preprocessing phase was merging output classes with a low number of instances into a single class labeled as "Other". This step was crucial for addressing the imbalance in the dataset, where certain classes had disproportionately low representation. By merging these underrepresented classes, the model is less likely to assign disproportionately high weights to these classes during training. The merging process along with dropping duplicates is illustrated in Figure 5, while the Class Distribution after merging the low instances classes and dropping duplicates is presented in Table 6. This approach offers several advantages:

1. **Improved Model Stability**: Low-instance classes can introduce noise and bias, leading to unstable and overfitted models. Combining these into a single class minimizes this risk.
2. **Better Generalization**: By consolidating rare classes, the model focuses on distinguishing meaningful patterns across larger, well-represented classes, improving its ability to generalize to unseen data.
3. **Simplified Training Process**: A reduced number of target classes makes the training process more efficient and manageable, especially for large datasets.
4. **Aligned with Real-world Scenarios**: In practical applications, rare events often require collective handling under broader categories rather than individual treatment. This step ensures the model reflects real-world priorities.

Reducing Dataset Size

To optimize the dataset for training, its size was reduced by converting data types from `float64` to `float32` and from ‘int64’ to ‘int32’. This conversion preserved the precision of the data while significantly reducing memory usage. By minimizing the dataset's size, it became more efficient to train the models, particularly when dealing with large-scale datasets, without compromising the quality of the data or the results.

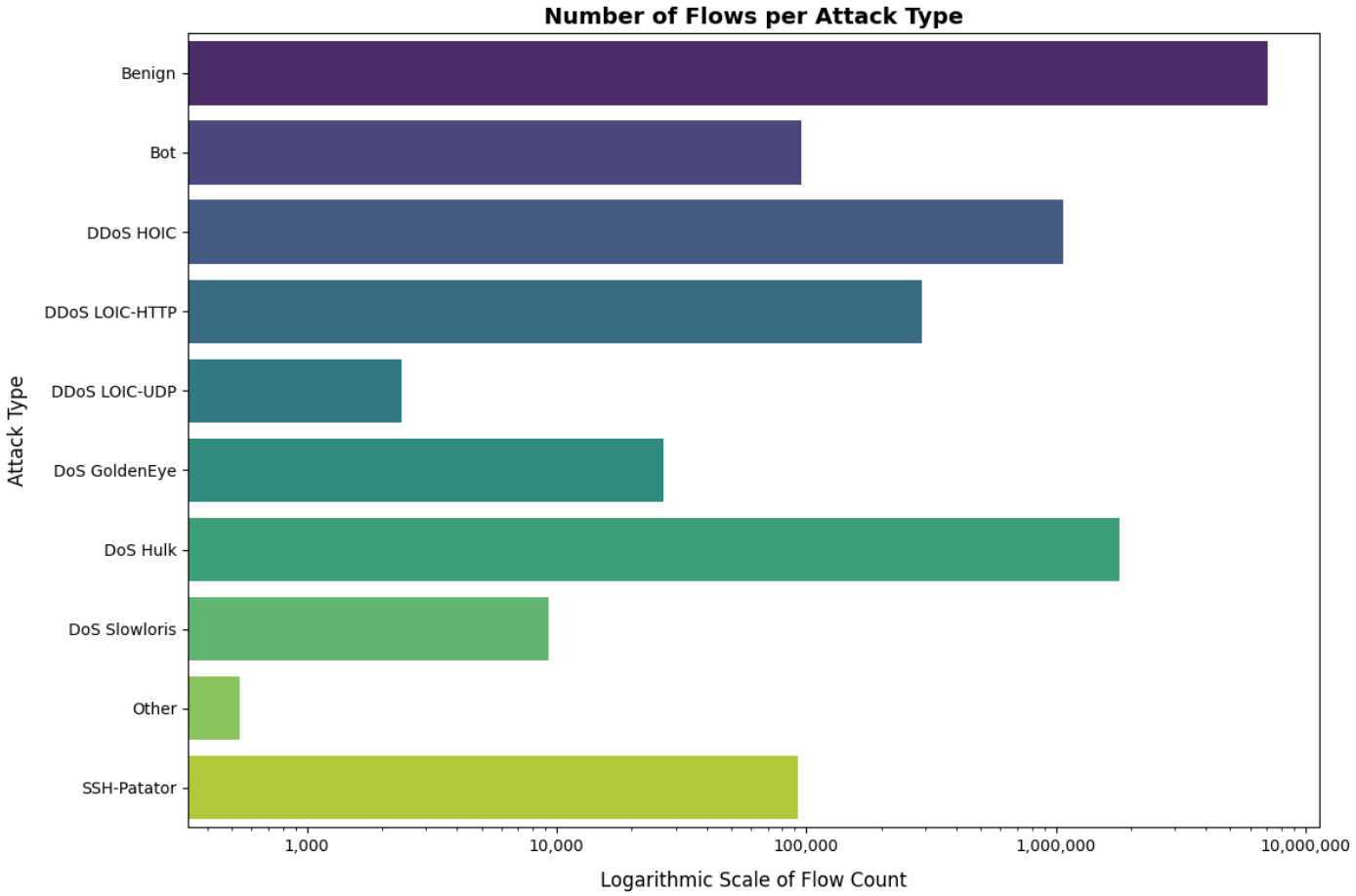

Figure 5: Class Distribution After Merging the Low Instances Classes and Drop Duplicate Rows

Table 6: Class Distribution After Merging the Low Instances Classes and Drop Duplicate Rows

Class Name	**Occurrences**
Benign	7,074,151
DoS Hulk	1,802,966
DDoS HOIC	1,074,376
DDoS LOIC-HTTP	289,328
Bot	96,100
SSH-Patator	92,625
DoS GoldenEye	26,861
DoS Slowloris	9,265
DDoS LOIC-UDP	2,382
Other	536

Feature scaling to normalize the features

Feature scaling is an essential preprocessing step designed to normalize feature values within a consistent range. In this study, I employed both standardization and label encoding techniques to ensure the dataset is well-prepared for machine learning algorithms.

Standardization

Standardization, often referred to as z-score normalization, is a key method for feature scaling. This technique involves subtracting the mean from each feature value and dividing the result by the standard deviation. It is particularly effective when feature values exhibit significant variation within the dataset. After standardization, all features are brought to a common scale with a mean (μ) of zero and a standard deviation (σ) of one, thereby enhancing the accuracy and performance of predictive models. The mathematical formula for z-score normalization is shown in Equation 1.

$$x_{new} = \frac{x-\mu}{\sigma} \qquad (1)$$

In this equation, x represents the original feature value, x_{new} signifies the standardized value, μ corresponds to the mean of the original feature, and σ denotes the standard deviation of the original feature.

While standardization is beneficial for many machine learning algorithms, tree-based models, such as Decision Trees (DT), Random Forests (RF), Extra Trees (ET), and XGBoost (XGB) do not require standardization. This is because these models are invariant to feature scaling. The decision-making process of tree-based models involves splitting the data based on thresholds (e.g., "if feature 1 > 10"), which is based on the relative ordering of feature values, not on their magnitudes. Therefore, these models do not rely on feature scaling to perform well. In this study, Standardization was applied for two key reasons, despite the fact that tree-based models do not require feature scaling. These reasons are critical for the innovative new approach named **SFE-PCA**:

1. **Principal Component Analysis (PCA):** PCA is sensitive to the scale of the data. It works by identifying the directions of maximum variance in the dataset, and if features have different scales, PCA may give undue importance to features with larger magnitudes. Standardizing the data ensures that each feature contributes equally to the analysis.
2. **Clustering (Gaussian Mixture Clustering and K-Means):** K-Means clustering is particularly sensitive to the scale of features because it uses Euclidean distance to measure similarity between data points. Features with larger values can dominate the distance computation if the data isn't scaled. Gaussian Mixture Clustering (GM) also benefits from scaling, though it is somewhat less sensitive to the exact scale of the data compared to K-Means. Therefore, scaling was applied in preparation for these techniques.

Label encoding

Label encoding is a technique used to transform categorical data into numerical values, making it suitable for machine

learning algorithms. Since ML models require numerical inputs, categorical values are converted into integers that correspond to unique classes. Each category is assigned a unique integer ranging from 0 to (n-1), where 'n' is the total number of unique classes. For example, if a dataset contains 11 unique categorical classes, the label encoding process will replace them with integers ranging from 0 to 10. This transformation ensures that categorical data can be efficiently utilized in the model training phase. Table 7 exemplifies the label encoding process. By combining standardization and label encoding, this approach ensures that the input features are normalized and ready for effective machine learning model training.

Table 7: Label Encoding Process

Attack types	Label Encoding
Benign	0
Bot	1
DDoS HOIC	2
DDoS LOIC-HTTP	3
DDoS LOIC-UDP	4
DoS GoldenEye	5
DoS Hulk	6
DoS Slowloris	7
Other	8
SSH-Patator	9

Handling Imbalance Using Equal Class Distribution Under-Sampling and Stratified Under-Sampling

Class imbalance is a significant issue in network intrusion detection datasets, where certain attack types are highly underrepresented compared to benign traffic. In this project, the dataset exhibited extreme imbalance, with the **Benign** class comprising **7,074,151** instances (approximately **67.57%** of the dataset), while several attack classes had only a few hundred instances. For example, the **DDoS LOIC-UDP** class contained only **2,382** samples (**0.0227%**), and the **Web Attack - SQL Injection** class had just **50** samples (**0.00047%**). Without addressing this imbalance, the classifier would be biased towards majority classes, leading to poor detection of rare attacks.

To mitigate this, under-sampling techniques were applied to balance the dataset and ensure that all classes were adequately represented for both **multiclass** and **binary classification** tasks.

Comparison of Resampling Techniques for handling imbalance dataset

There are multiple strategies for handling imbalanced datasets, including oversampling, under-sampling, and hybrid methods.

- **Random Oversampling (RO):** This method involves duplicating instances from minority classes to balance the dataset. While effective in increasing the representation of rare attack types, it introduces a risk of overfitting, as the model may memorize duplicated samples rather than learning meaningful patterns. Given the size of the dataset, oversampling would significantly increase computational complexity. The RO process is depicted in Figure 6

- **Random Under-Sampling (RUS):** This method reduces the majority class instances, making them more comparable in size to the minority classes. While it removes data, it prevents the model from being biased toward frequent attack types. RUS was chosen as the primary technique due to its ability to prevent overfitting and maintain computational efficiency. The RUS process is illustrated in Figure 7
- **Hybrid Sampling (RO + RUS):** A combination of both methods that balances the dataset by increasing minority-class samples while reducing majority-class samples. However, this approach still carries the risk of overfitting due to oversampling. Figure 8 illustrating the hybrid sampling process, showcasing the balance achieved by integrating both RO and RU.

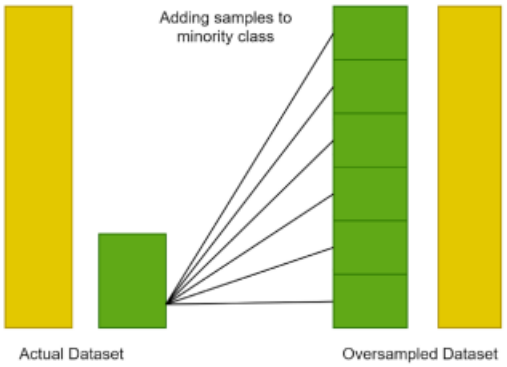

Figure 6: Random Oversampling Process

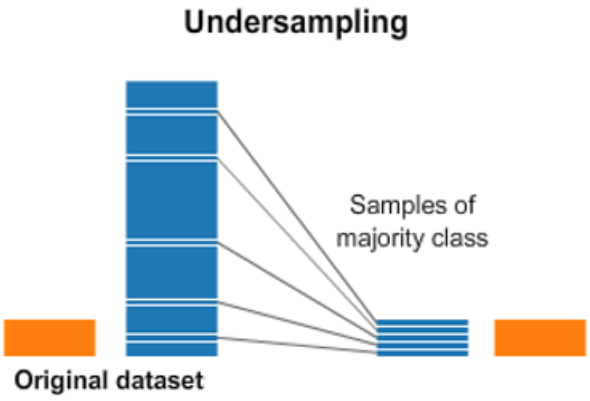

Figure 7: Random Under-Sampling Process

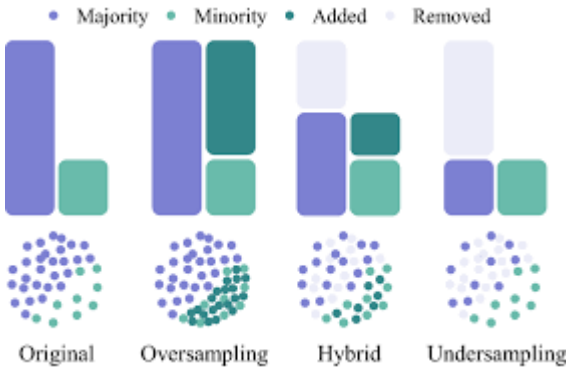

Figure 8: Hybrid Sampling Process

Given the large dataset size and the need for reliable trusted results without the risk of overfitting, **Random Under-Sampling (RUS) was selected as the optimal method, as it reduces class imbalance without artificially inflating minority-class instances, preventing overfitting while preserving meaningful attack patterns**. Table 8 provides a comparative analysis of different resampling techniques, highlighting their effectiveness, impact on model performance, and computational efficiency in handling imbalanced datasets.

Table 8: Effectiveness and Availability of Resampling Techniques for Imbalanced Datasets

Technique	Description	Advantages	Disadvantages	Availability & Impact on Model Performance
Random Oversampling (RO)	Duplicates minority-class samples to balance the dataset.	Prevents data loss and increases representation of rare attack types.	High risk of overfitting, increased computational cost.	Less reliable for trusted results due to overfitting concerns; increases computation time, especially with large datasets due to data duplication.
Random Under-Sampling (RUS)	Reduces majority-class instances to balance the dataset.	Prevents overfitting, computationally efficient.	May lose some information if not applied carefully	**(Chosen Method):** Highly reliable, ensuring trusted results while maintaining model efficiency and requiring less computation time. Achieved high recall, preserving important attack patterns without significant information loss.
Hybrid Sampling (RO + RUS)	Combines oversampling and undersampling for balance.	Moderates class distribution while preserving data.	More complex, still carries risk of overfitting due to oversampling.	Not chosen due to the persistent risk of overfitting and untrusted results; adds unnecessary complexity when RUS alone provides excellent performance.

Multiclass Classification Handling

Table 9 and Figure 9 illustrates the dataset imbalance, it is evident that some attack classes are significantly underrepresented. To handle this, two under-sampling strategies were applied:

Table 9: Multiclass Dataset Imbalance Before Under-Sampling

Class label	Occurrences (after merging and dropping duplicates)	Percentage (%)
Benign	7074151	67.575%
DoS Hulk	1802966	17.2226%
DDoS HOIC	1074376	10.2628%
DDoS LOIC-HTTP	289328	2.7637%
Bot	96100	0.9179%
SSH-Patator	92625	0.8847%
DoS GoldenEye	26861	0.25658%
DoS Slowloris	9265	0.0885%
DDoS LOIC-UDP	2382	0.0227%
Other	536	0.00512%

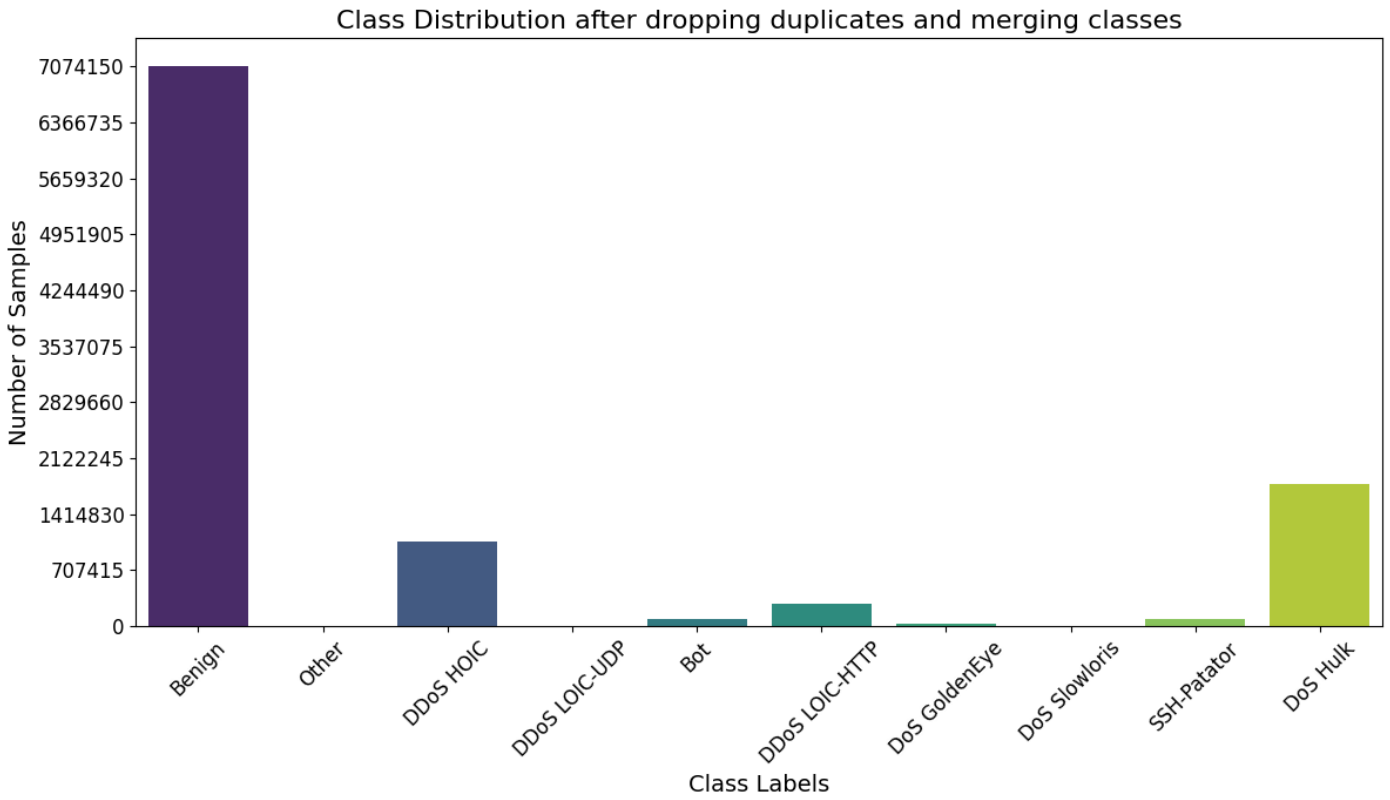

Figure 9: Multi Class Distribution Before Under-Sampling

1. Equal Class Distribution Under-Sampling

In this method, all classes were under-sampled to match the count of the least represented class. This ensures a perfectly balanced dataset where each class has the same number of instances. After applying equal class distribution under-sampling, each class contained **536** instances (matching the smallest class, '**Other'**), as shown in Table 10 and Figure 10.

Table 10: Multiclass Distribution After Equal Class Distribution Under-Sampling

Class Label	Count After Equal Class Distribution
Benign	**536**
DoS Hulk	**536**
DDoS HOIC	**536**
DDoS LOIC-HTTP	**536**
Bot	**536**
SSH-Patator	**536**
DoS GoldenEye	**536**
DoS Slowloris	**536**
DDoS LOIC-UDP	**536**
Other	**536**

Figure 10:Multi Class Distribution After Equal Class Distribution Under-Sampling

This method **completely removes imbalance**, ensuring that all classes are equally represented. However, it results in the loss of a substantial amount of data, particularly from the majority class, which may impact classification performance for more common traffic types.

2. Stratified Under-Sampling (Preserving Proportional Class Distribution)

Rather than making all class sizes equal, this method maintains the proportional representation of each class while reducing the dataset size. This ensures that rare attack types remain underrepresented in a realistic way. After applying stratified under sampling, the new dataset distribution is illustrated in Table 11 and Figure 11.

Table 11: Multiclass Distribution After Stratified Under-Sampling

Class Label	Count After Stratified Under-Sampling
Benign	707415
DoS Hulk	180296
DDoS HOIC	107437
DDoS LOIC-HTTP	92625
Bot	28932
SSH-Patator	9610
DoS GoldenEye	2686
DoS Slowloris	926
DDoS LOIC-UDP	238
Other	53

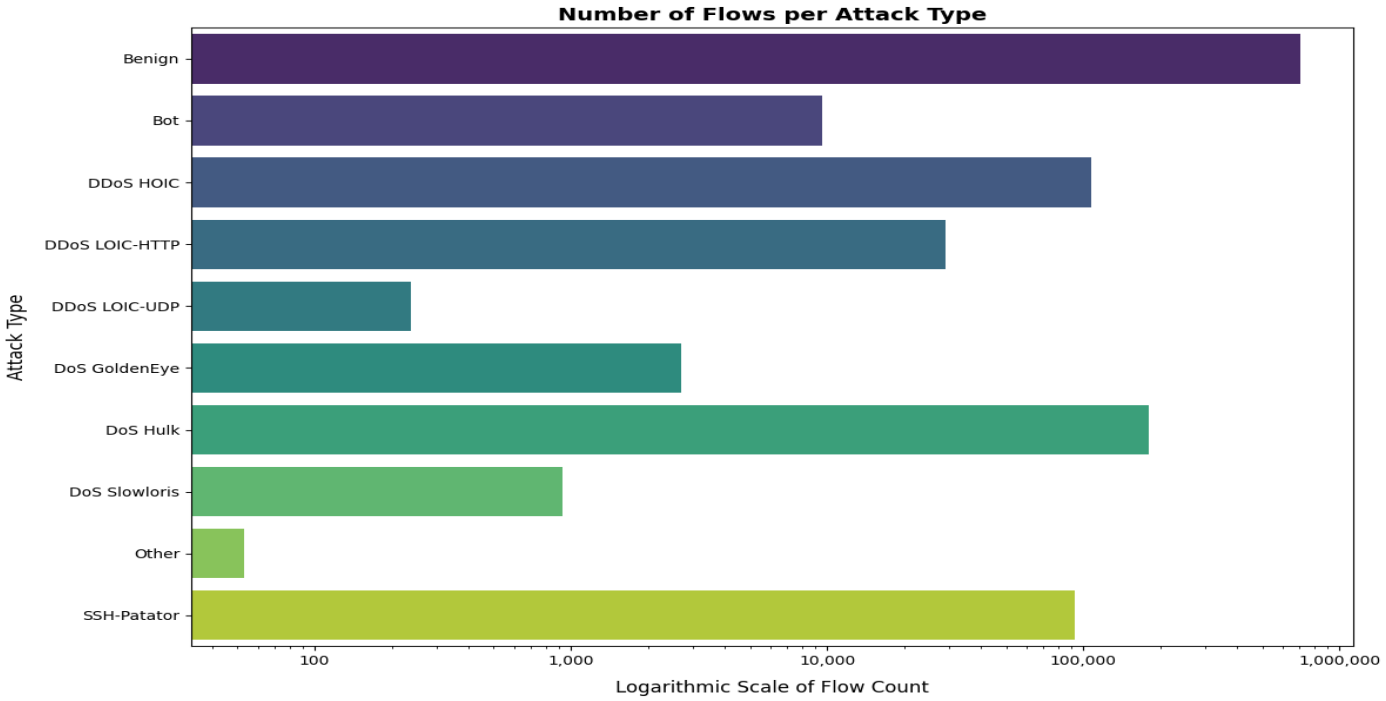

Figure 11: Multi Class Distribution After Stratified Under-Sampling

This method significantly reduces the dataset size while maintaining the **relative proportions** of different classes, preventing excessive data loss from dominant categories while still ensuring that the model is not biased toward frequent attack types.

Binary Classification Handling

For binary classification, the dataset was divided into **Benign** and **Malicious** categories with the following distribution as depicted in Table 12 and Figure 12.

Table 12: Original Binay Class Distribution

Class	Original Count	Percentage (%)
Benign	7,074,151	67.6%
Malicious	3,394,439	32.4%

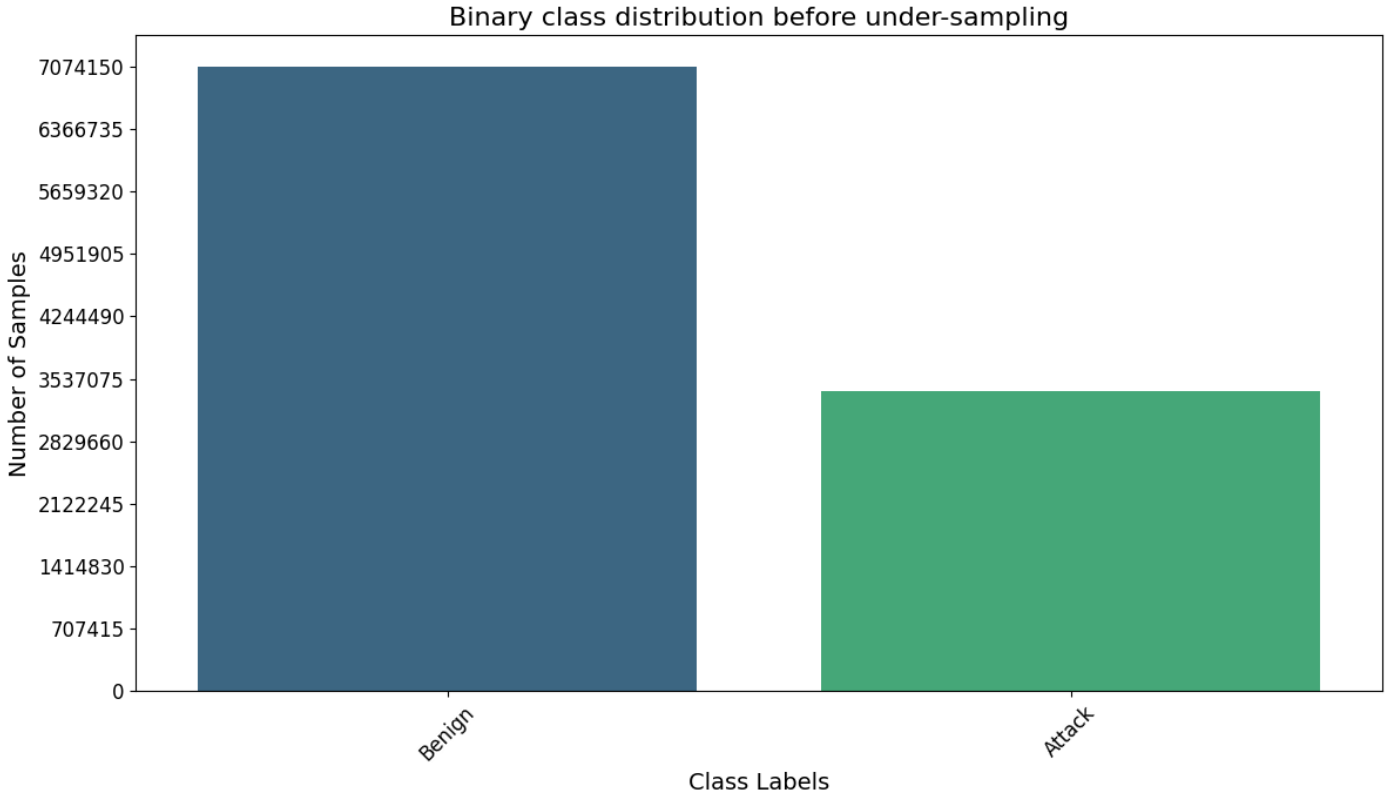

Figure 12: Original Binay Class Distribution

Since this distribution is **not highly imbalanced**, under-sampling was only applied to reduce dataset size and computation time. Two approaches were used:

- **Equal Class Distribution:** Both classes were under-sampled to **3,394,439** samples each, ensuring a perfectly balanced dataset as shown in Figure 13.
- **Stratified Under-Sampling:** The dataset was reduced while maintaining the original **67.6:32.4 ratio**, resulting in **3,537,075** benign samples and **1,697,219** malicious samples as illustrated in Figure 14.

Figure 13: Binary Class Distribution After Applying Equal Class Distribution - Under-Sampling

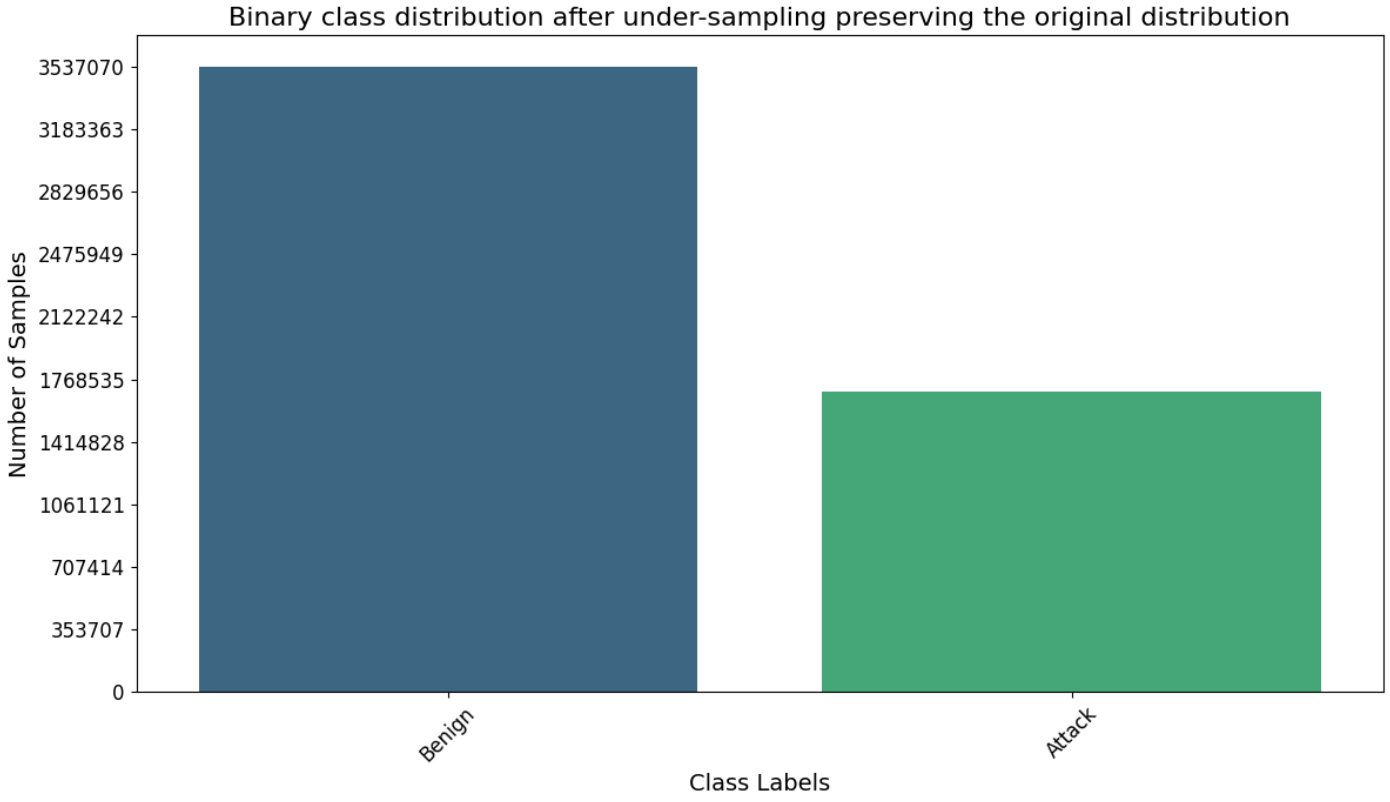

Figure 14: Binary Class Distribution After Applying Stratified Under-Sampling

Random under-sampling was selected as the preferred method due to its ability to **reduce bias, prevent overfitting, and maintain computational efficiency without artificially inflating data**. While equal class distribution under-sampling ensures perfect balance, it leads to significant data loss. Stratified under-sampling, on the other hand, maintains the proportional representation of each class as it better reflects real-world data distribution while reducing dataset size.

Both approaches were applied to **multiclass** and **binary classification**, ensuring that the dataset remains balanced while preserving critical attack patterns.

Stacking Feature Embedded using clustering with PCA

Within my experimental framework, I implemented a new methodology called Stacking Feature Embedded with PCA (SFE-PCA), inspired by the approach outlined in [29]. This methodology integrates clustering techniques firstly with dimensionality reduction then to enhance the performance and efficiency of machine learning models. Unlike this methodology, my adaptation modifies the sequence by applying Principal Component Analysis (PCA) first, followed by clustering, to reduce memory consumption without compromising effectiveness.

The process begins with PCA, a dimensionality reduction technique that transforms the feature set while retaining the most informative components. By doing so, I am able to significantly reduce the dimensionality of the dataset, ensuring that only highly relevant and discriminative features are preserved. This optimization not only improves computational efficiency but also ensures that the input features are well-suited for machine learning model training. Following PCA, I applied the K-Means and Gaussian Mixture (GM) Clustering algorithm to the reduced feature set. Clustering algorithms groups data points based on their intrinsic patterns and structures, identifying clusters that capture underlying relationships within the dataset. To determine the optimal number of clusters, I utilized the Silhouette Method, which evaluates the cohesion and separation of clusters to ensure a robust clustering process. The results of the clustering process were then embedded back into the original feature space as meta-features. This augmentation enriches the dataset with additional layers of information, enabling the machine learning models to capture finer details and complex patterns that might be overlooked by conventional feature engineering approaches.

The integration of PCA and clustering in the SFE-PCA methodology strikes a balance between detailed feature representation and computational efficiency. PCA ensures dimensionality reduction without losing critical information, while clustering introduces a higher level of abstraction by grouping data points based on shared characteristics. The combination of these two techniques creates a focused and enriched feature set, empowering machine learning models to achieve improved accuracy, precision, and overall performance. This innovative approach has demonstrated its ability to capture essential patterns and optimize model performance, making it a valuable contribution to the field of intrusion detection.

Feature extraction using PCA

The challenge of high-dimensional datasets often leads to increased model complexity and the risk of overfitting, ultimately degrading model performance. Reducing the dimensionality of such datasets is crucial to simplify the model, decrease computation time, facilitate data visualization, and eliminate redundant features while retaining the essential information.

Feature reduction transforms a high-dimensional dataset into a lower-dimensional representation by generating new features from the original ones. PCA is a widely used statistical technique for this purpose. It employs an orthogonal transformation to convert correlated variables into uncorrelated ones, enabling both exploratory data analysis and the development of predictive models. Unlike regression, PCA creates a line of best fit, often described as a form of generic factor analysis, making it a valuable unsupervised method for understanding the relationships between variables.

To reduce the dimensions of a dataset from n features to k, the following steps are undertaken:

1. Equalize the data's initial attribute values the dataset by the mean μ and variance, as shown in Equation 2.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \textbf{(2)}$$

Here n represents the instances number and xi represents the data points.

2. Subtract the mean from each feature value: $\boldsymbol{xi} = \boldsymbol{xi} - \mu$.

3. Rescale each vector $xj(i)$ to have unit variance.

$$\boldsymbol{\sigma_j^2} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{\Pi}} \left(\boldsymbol{x_{j(i)}}\right)^{\mathbf{2}} \qquad \textbf{(3)}$$

4. Substitute $xj(i)$ by $\frac{x_{j(i)}}{\sigma}$.

4. Compute the covariance matrix C_M of the standardized dataset.

$$\boldsymbol{c_M} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i}(\boldsymbol{x_i})^{\boldsymbol{T}} \qquad \textbf{(4)}$$

5. Calculate the Eigen-vectors and their related Eigen-values of the covariance matrix C_M.

6. To generate w, Sort the Eigen-vectors in descending order of their eigenvalues and select the top k eigenvectors with the largest Eigen-values

7. Transform the original dataset into the new subspace using the selected eigenvectors (w), as shown in Equation 5.

$$\boldsymbol{y} = \boldsymbol{w^T} * \boldsymbol{x} \qquad \textbf{(5)}$$

where x represents one sample as a d × 1 dimensional vector and y represents the converted k × 1 dimensional vector in the resulting subspace.

The computational complexity of PCA is influenced by the number of features D_p in the dataset [57]. The reduction ratio (RR) is defined as the number of output dimensions divided by the number of input dimensions [58]. A lower RR value corresponds to a higher PCA efficiency.

In my proposed framework, PCA was employed to reduce the dimensionality of the LycoS-18 dataset, ensuring an optimal balance between minimizing feature space and preserving essential information for intrusion detection. The selection of principal components was guided by the criterion of retaining 95% of the variance. Specifically, in multiclass classification, Equal Class Distribution Under-Sampling resulted in 2 selected principal components (PCs), whereas Stratified Under-Sampling required 25 PCs. For binary classification, both Equal Class Distribution Under-Sampling and Stratified Under-Sampling utilized 26 PCs. To provide a clear visualization of this selection process, Figure 15, Figure 16, Figure 17, and Figure 18 illustrate the number of components chosen for each sampling strategy.
The efficiency of this dimensionality reduction can be quantified using the reduction ratio (RR), calculated as the proportion of selected components relative to the original feature set. The RR values for each scenario—binary classification with both Equal Class Distribution Under-Sampling and Stratified Under-Sampling, as well as multiclass classification with both strategies—are presented in Table 13.
Additionally, the overall PCA process is depicted in Figure 19, providing a general conceptual illustration of PCA. it offers a high-level representation, demonstrating the transformation from original features to principal components.

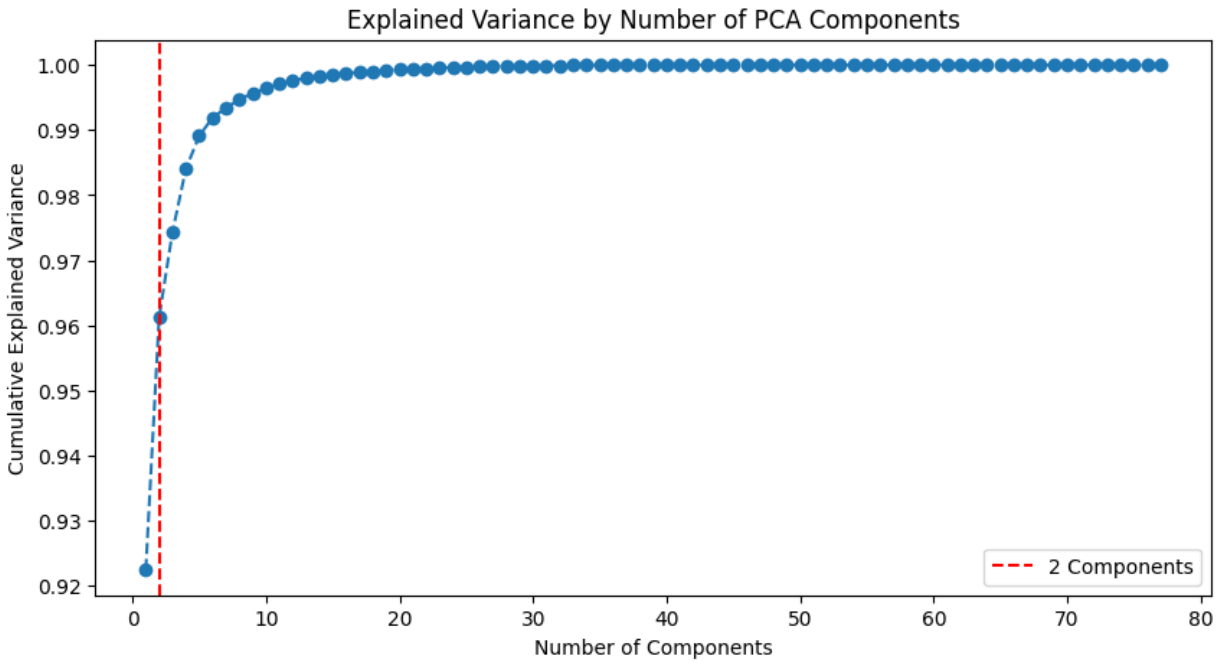

Figure 15: Selected Principal Components for Multiclass Classification (Equal Class Distribution Under-Sampling)

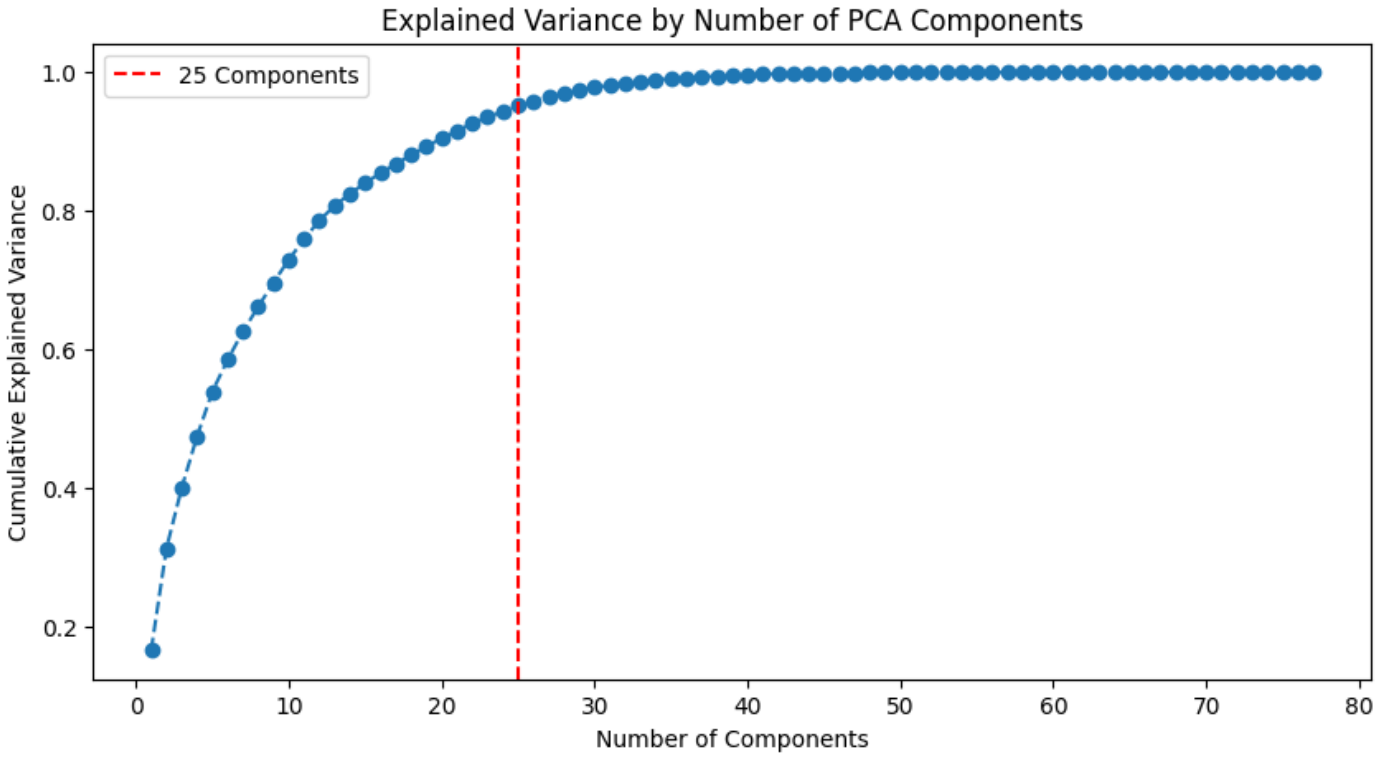

Figure 16: Selected Principal Components for Multiclass Classification with Stratified Under-Sampling

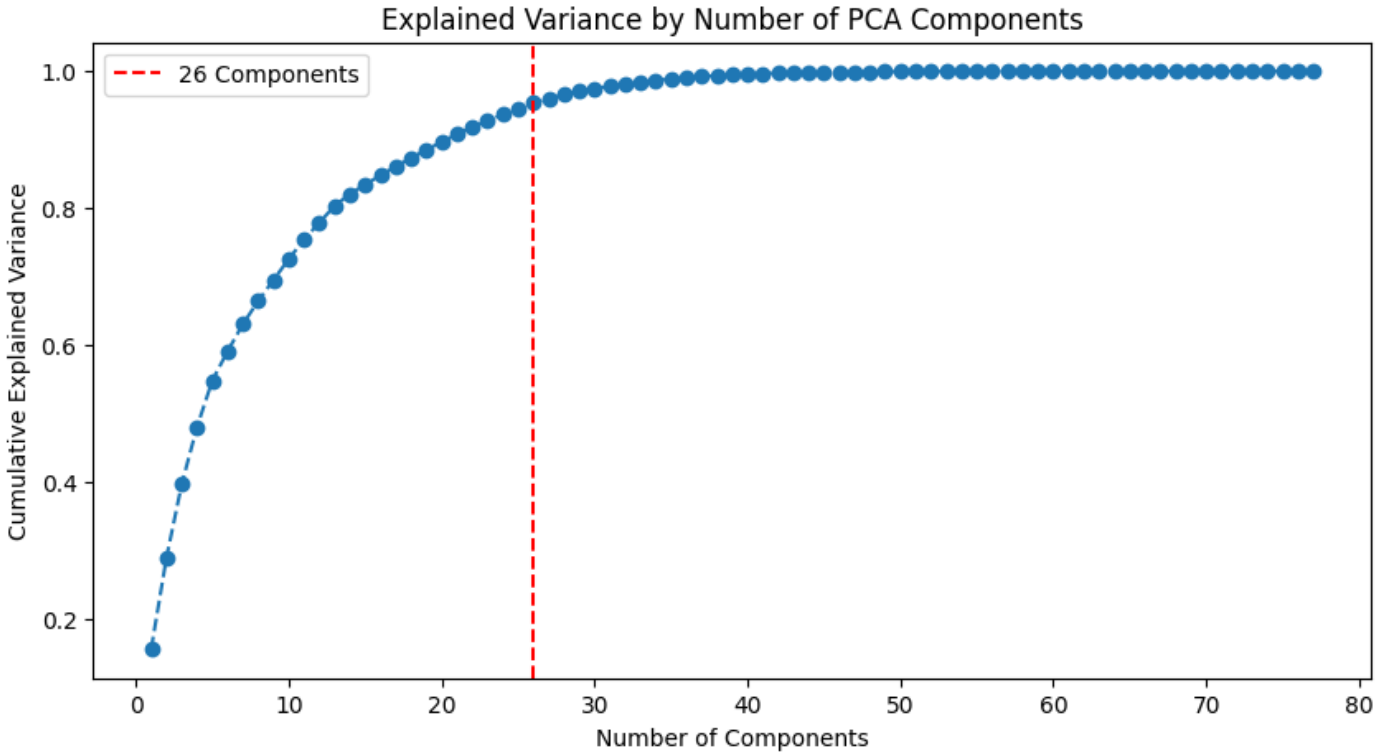

Figure 17: Selected Principal Components for Binary Classification with Equal Class Distribution Under-Sampling

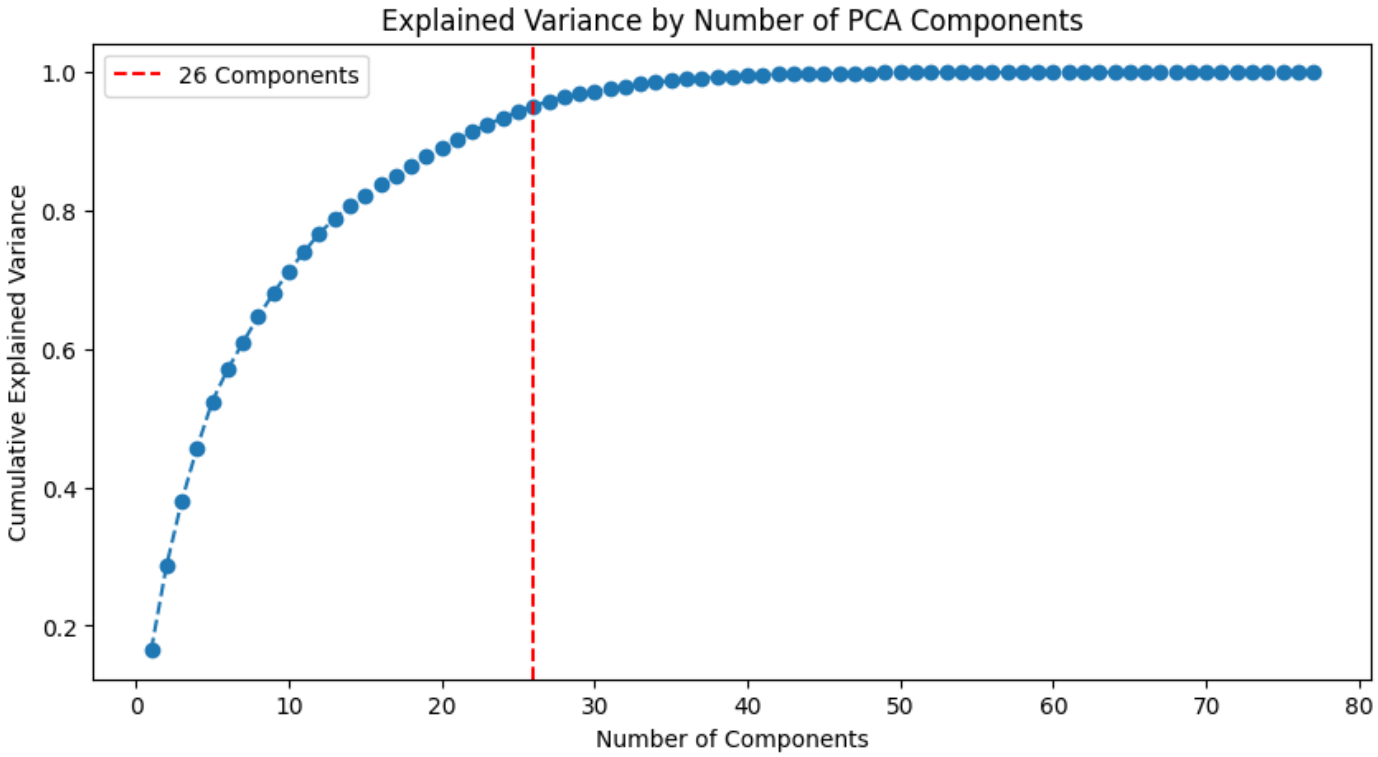

Figure 18: Selected Principal Components for Binary Classification with Stratified Under-Sampling

Table 13: Dimensionality Reduction Ratios for Different Sampling Strategies

Classification Type	**Reduction Ratio (%)**
Multiclass (Equal Class Distribution)	2.60%
Multiclass (Stratified Under-Sampling)	32.47%
Binary (Equal Class Distribution)	33.77%
Binary (Stratified Under-Sampling)	33.77%

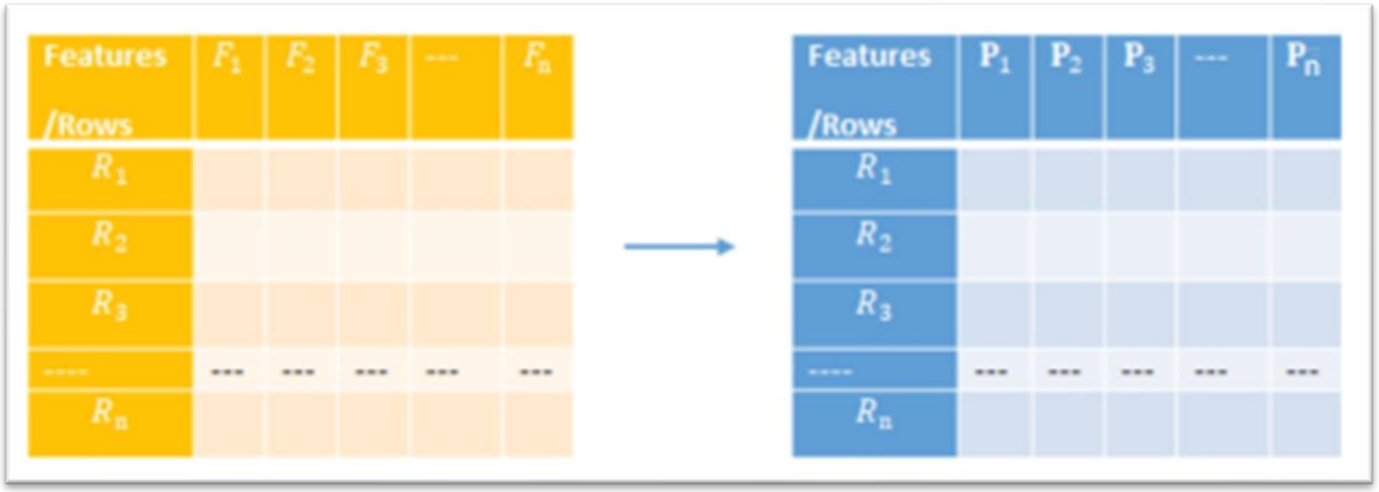

Figure 19: Conceptual Illustration of the PCA Process

Stacking Feature Embedded using clustering

The proposed Stacking Feature Embedded (SFE) methodology is a cornerstone of the used framework for this project. It aims to address the challenges associated with big and imbalanced datasets, particularly in the domain of ML-based network intrusion detection. By integrating clustering techniques with feature embedding, this approach enhances detection accuracy and robustness. The SFE process is illustrated in Figure 20.

The following are the working principles of this approach:

Cluster Formation: The first step involves applying a clustering technique, specifically K-Means and Gaussian Mixture, to group data points based on shared characteristics and patterns. This clustering process uncovers the underlying structure of the data, leading to a deeper understanding of the dataset. A key aspect of this step is determining the optimal number of clusters. In this project, the Silhouette Method was used to assess the most appropriate number of clusters. The silhouette score measures how well each data point fits within its cluster, with higher scores indicating clearer and more distinct clusters. Based on this analysis, two clusters were found to offer the most optimal grouping for the dataset, in combination with applying Stratified Under Sampling for multiclass classification.

Feature Embedding: The clusters generated during the clustering phase are embedded into the original feature space. This embedding process generates additional features, often referred to as meta-dataset points, which capture nuanced relationships and patterns uncovered by the clustering process. These newly created features enrich the dataset by adding an additional layer of information that would otherwise remain undiscovered in traditional feature spaces.

Enhanced Data Representation: The final dataset includes both the original features and the newly embedded meta-dataset points, creating an augmented representation of the data. This enhanced dataset provides a richer and more comprehensive view, enabling the machine learning models to detect subtle patterns and anomalies more effectively.

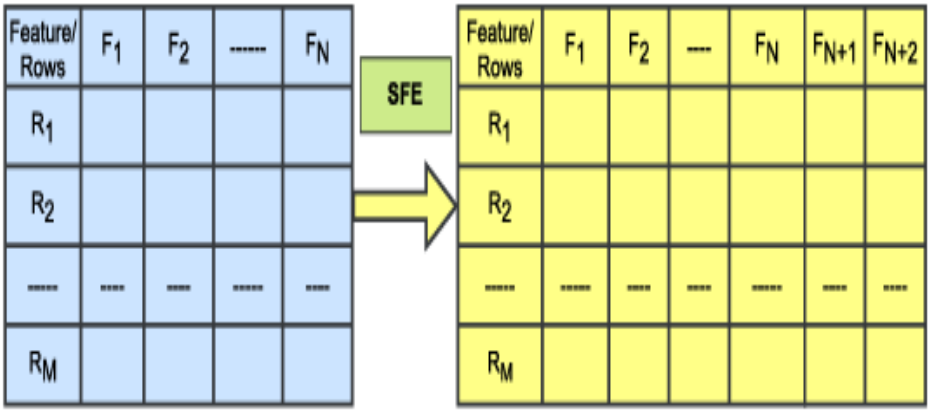

Figure 20: The SFE process

The SFE methodology was designed to overcome the limitations of traditional intrusion detection techniques when applied to big and imbalanced datasets. By combining clustering techniques with feature embedding, this approach achieves several critical objectives:

1. Improved Detection Accuracy: Enriching the feature space with embedded meta-dataset points enhances the models' ability to distinguish between normal and anomalous network traffic.

2. Detection of Subtle Anomalies: The integration of clustering insights enables the detection of fine-grained and previously unnoticed details in network traffic data.

3. Comprehensive Security Posture: The ability to identify previously undetected threats contributes to a more robust and reliable defense mechanism.

4. Enhanced Model Performance: By providing an enriched feature representation, the methodology improves the precision and overall effectiveness of the machine learning models.

This approach represents a significant advancement in the field of network intrusion detection, offering a powerful framework for addressing real-world challenges associated with imbalanced and large-scale network datasets.

Silhouette Methodology for Optimal Clustering

The Silhouette method was employed to identify the optimal number of clusters for the dataset. This method assesses how well data points fit within their assigned clusters, while also ensuring clear separation from other clusters. The silhouette score ranges from -1 to 1, with higher values indicating better-defined clusters. In the binary class classification, using Stratified Under Sampling, the silhouette analysis indicated that three clusters offered the best balance between cohesion and separation. The result of this silhouette analysis is presented in Figure 21, where a line plot displays the silhouette scores for different numbers of clusters. This analysis not only confirms the appropriateness of the selected cluster count but also guarantees that the clustering step aligns with the goal of forming meaningful and informative groupings for this project.

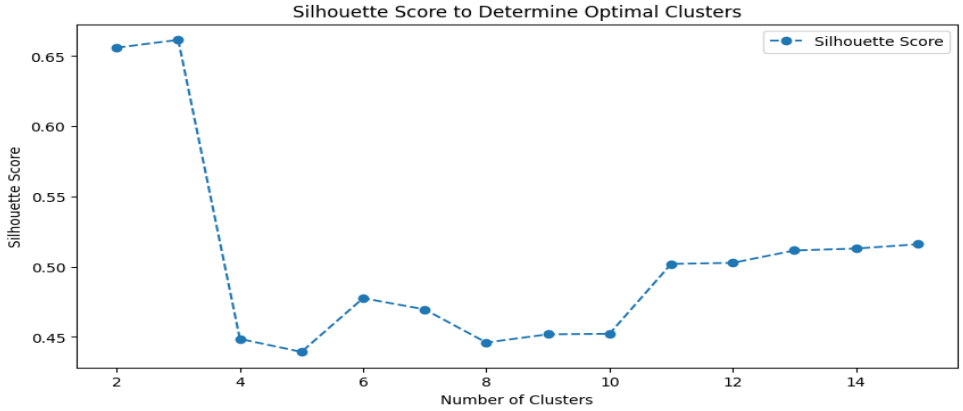

Figure 21: Silhouette Scores for Different Numbers of Clusters

MLAlgorithms

In this section, I have leveraged supervised ML algorithms to evaluate their performance in binary and multi class classification tasks using LycoS18 dataset. The application of ML in detecting network intrusion in the context of security is outlined as follows:

Decision Tree (DT)

The Decision Tree (DT) is a non-parametric supervised ML method designed for both classification and regression problems. It predicts an output value based on decision rules derived from dataset features. This algorithm is intuitive, easy to interpret, and can be represented visually. It also supports handling multi-output problems, making it widely applicable in IDS [59].

The DT comprises two primary types of nodes: decision nodes, which branch out to assist in making decisions, and leaf nodes, which have no branches and provide final outputs. The root node is the starting point for all decision-making processes in the tree. To construct a DT, an Attribute Selection Measure (ASM) is performed on information gain (IG) and the Gini index (GI) to select the feature [60]. IG quantifies the reduction in entropy achieved after a feature split, while GI measures the impurity of a node. Nodes are selected for splitting based on IG, with lower GI values preferred for binary splits.

The pruning process is used to enhance the DT's accuracy by eliminating unnecessary nodes. Two common pruning techniques are Cost Complexity Pruning and Reduced Error Pruning. Equations 7 and 8 represent GI and IG, respectively.

$$\boldsymbol{Gini(D) = 1 - \sum_{i=1}^{n} (p_i)^2} \quad \textbf{(7)}$$

$$\boldsymbol{Gain(A) = Entropy(D) - EntropyA(D)} \quad \textbf{(8)}$$

where,

$$\boldsymbol{Entropy(D) = -\sum_{1=1}^{n} p_i \log_2(p_i)} \quad (9)$$

$$\boldsymbol{Entropy_A(D) = \sum_{i=1} \frac{D_i}{|D|} \times Entropy(D_i)} \quad \textbf{(10)}$$

and the probability of a data point in the subset of D_i of a dataset D is denoted by (P_i).

A Decision Tree is structured like a flowchart, where each internal node represents a test on an attribute, and each branch signifies the outcome of that test. Terminal nodes, or leaf nodes, hold the resulting class label. The tree learns by recursively partitioning the dataset based on attribute values, ceasing when further splits no longer enhance predictions or all subsets at a node share the same target variable value. Classification occurs as instances traverse from the root to leaf nodes, with each decision node applying an attribute test to determine the appropriate branch. This iterative process concludes with a classification at the leaf node. This process is exemplified in Figure 22, which depicts a Decision Tree classifying morning suitability for tennis, illustrating how hierarchical splits based on feature values guide classification decisions—an approach similarly applied in network traffic classification.

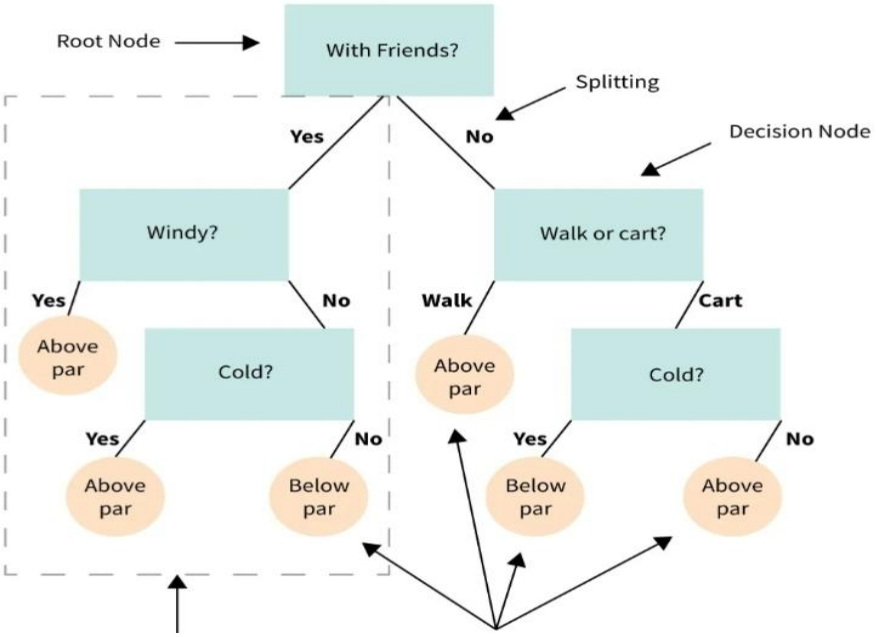

Figure 22: Decision Tree (DT) Model Structure for Classification

Random Forest (RF)

Random Forest (RF) is a prominent supervised ML algorithm based on ensemble learning, which combines multiple classifiers to solve complex problems and improve model performance. RF acts as a meta-predictor, employing averaging techniques to enhance prediction accuracy and mitigate overfitting. It constructs multiple decision tree classifiers using different subsets of data through a bootstrap resampling technique [61].

RF is efficient in training, achieves high accuracy, and performs well with large datasets. It reduces overfitting by aggregating predictions from multiple trees. The final prediction is determined through a voting mechanism, where the majority vote among classifiers decides the outcome. This ensemble approach significantly improves predictive performance [62]. Figure 23 demonstrates how the Random Forest model constructs multiple Decision Trees and averages their predictions to improve classification performance.

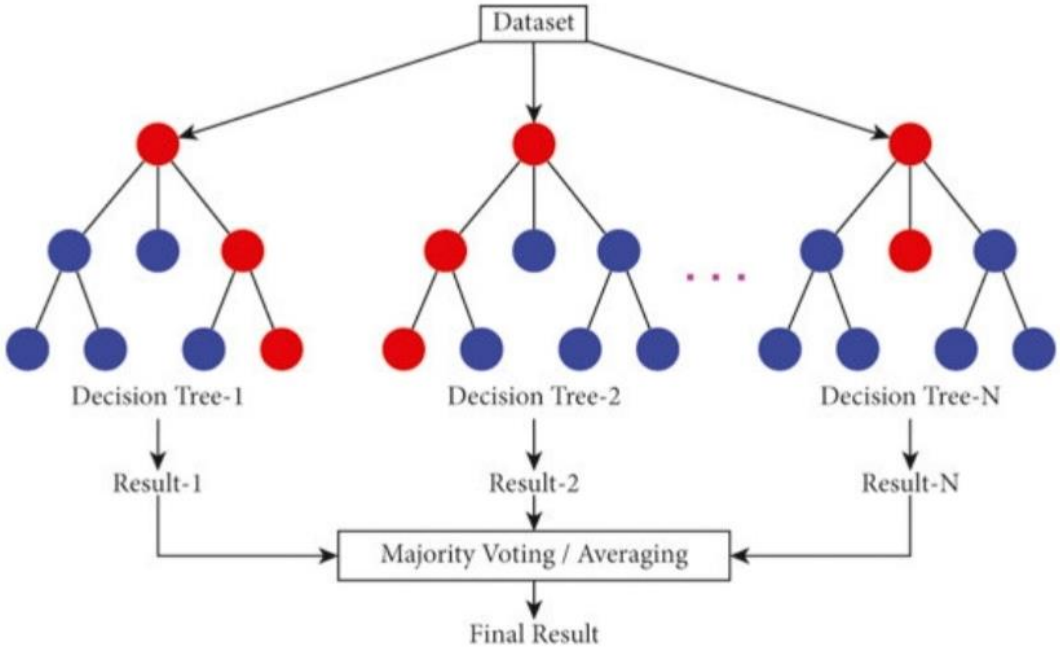

Figure 23: Random Forest (RF) Model Architecture – Ensemble of Decision Trees

Extra Tree (ET)

The Extra Tree (ET), also known as Extremely Randomized Trees, is an ensemble ML technique and meta-estimator. It enhances prediction accuracy by employing a series of highly randomized decision trees applied to various subsets of the data. By averaging the outputs of these trees, ET prevents overfitting.

Like RF, ET is an ensemble model, but it employs a different mechanism. It generates a large number of unpruned decision trees using the entire dataset and splits nodes by selecting random cut-points for features. For classification, majority voting is used to determine outcomes, while for regression, the average is employed. This randomness contributes to robust predictive capabilities [63]. Figure 24 shows how the Extra Trees model differs from Random Forest by selecting splits randomly, leading to increased variance reduction and generalization.

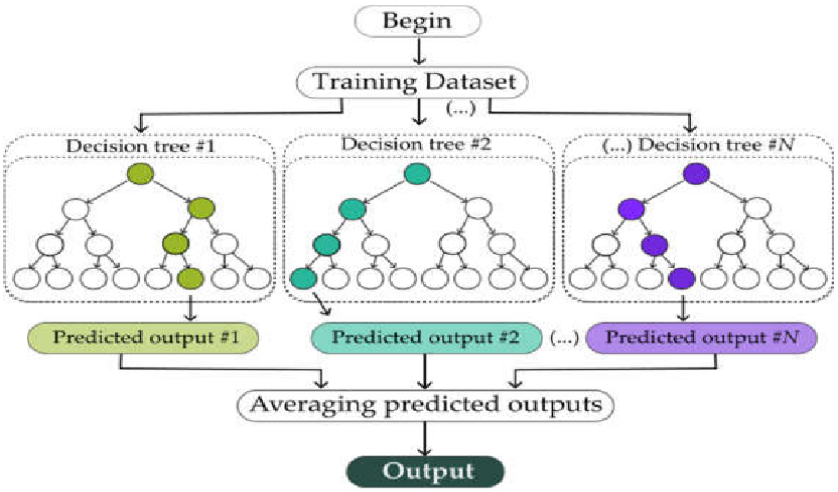

Figure 24: Extra Trees (ET) Model – Randomized Splitting in Decision Trees

Extreme Gradient Boosting (XGB)

Extreme Gradient Boosting (XGB) is a supervised ML algorithm that uses gradient-boosted decision trees to deliver superior speed and performance. XGB is known for its efficiency compared to other gradient boosting methods [64].

The algorithm builds upon residual errors from previous models to refine predictions, employing gradient descent to minimize loss and improve accuracy. When looking for cutting-edge solutions for various ML problems, data scientists have come to appreciate it, as a scalable end-to-end tree-boosting technique [65]. XGB integrates regularization techniques to control model complexity and prevent overfitting. Its objective function, as illustrated in Equation 11, consists of two components: the training loss function and a regularization term (Ω).

$$\boldsymbol{O(\theta) = L(\theta) + \Omega(\theta)} \qquad \textbf{(11)}$$

with θ representing the optimal settings for the training data x_i and the associated labels y_i, and L represents the training loss function, which is a metric for evaluating the model's performance in predicting the training datasets.

For instance, ta straightforward example of a training loss function is the Mean Squared Error (MSE):

$$\boldsymbol{L(\theta) = \sum_i (y_i - \hat{y}_i)^2} \qquad \textbf{(12)}$$

Similarly, the logistic loss function is another commonly used function in classification tasks:

$$\boldsymbol{L(\theta) = \sum_i \left[y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i) \ln\left(1 + e^{-\hat{y}_i}\right) \right]} \qquad \textbf{(13)}$$

The regularization term (Ω) includes parameters for controlling the model's complexity, which helps us to avoid overfitting such as the number of terminal nodes (T), leaf weights (w), pruning factor (γ), and (λ) that is expected to lower the outcome's sensitivity [64], Ω is given by

$$\boldsymbol{\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \|\omega\|^2}$$

XGBoost is known for its speed and high performance, making it particularly well-suited for handling large datasets with reduced computational time, which provides a significant advantage in time-sensitive applications [65]. Figure 25 illustrates how XGBoost sequentially builds decision trees, correcting errors from previous trees to improve overall classification accuracy

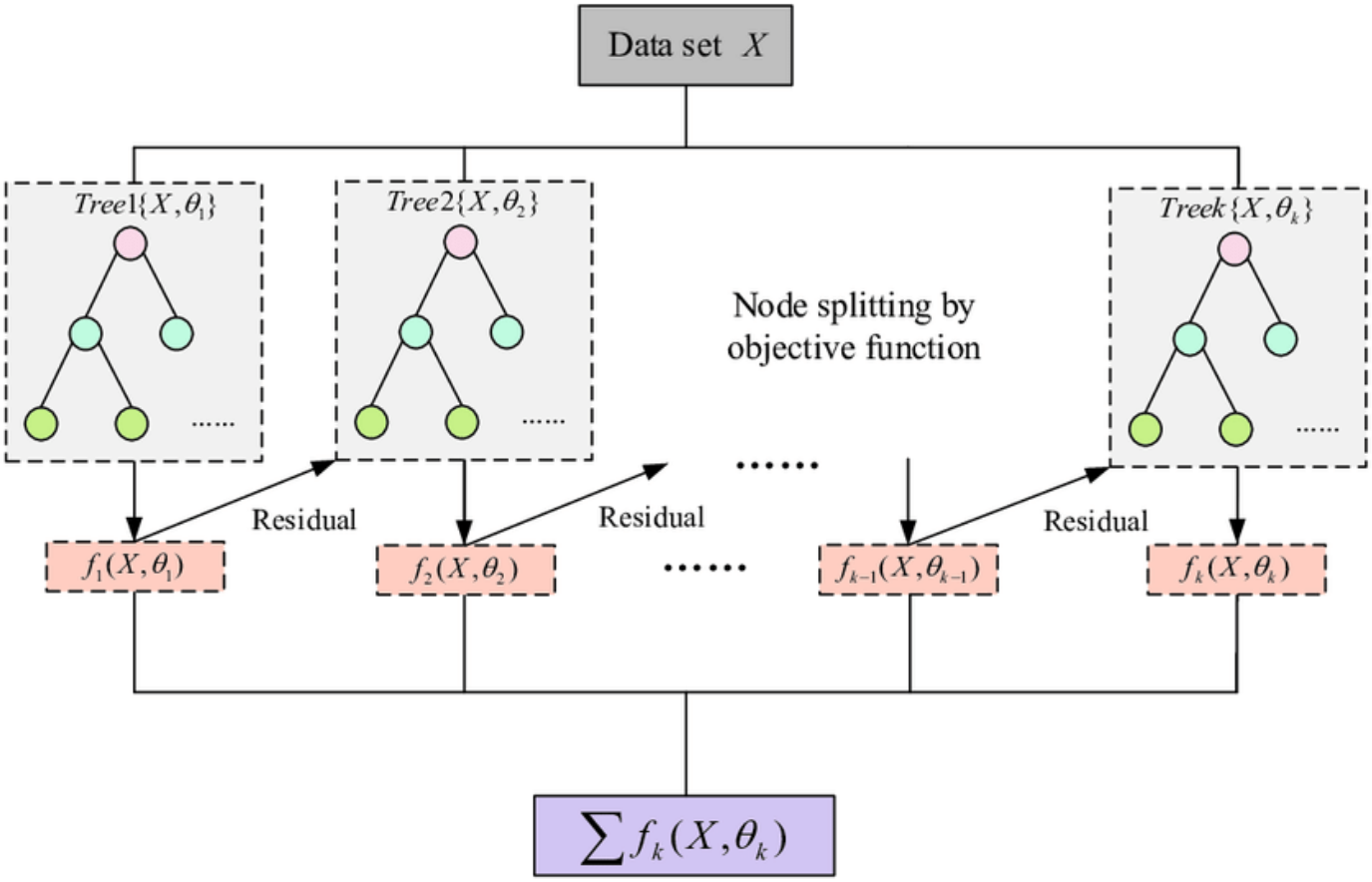

Figure 25: XGBoost (XGB) Model – Boosted Decision Trees for Enhanced Learning

5 Experimental setup and Evaluations

Environment setup

The experiments were conducted on a Core i7 machine with 16 GB of RAM, a configuration that required careful management due to the large 6 GB dataset. Given the resource limitations, I employed several memory optimization techniques to ensure smooth execution and mitigate issues such as system crashes and memory overload. Although the setup proved to be functional, occasional fatal blue screen crashes occurred, typically when running resource-heavy code cells. These errors were likely related to video memory usage and highlight the importance of having a more powerful machine with advanced hardware specifications. A setup with a dedicated GPU and increased RAM would significantly improve efficiency and reduce the likelihood of such errors.

To overcome memory limitations, I implemented a range of strategies. Initially, I loaded the dataset in smaller chunks, which were then concatenated for further processing. I also reduced the feature types from float64 to float32 and from int64 to int32, enhancing computation speed without sacrificing model performance. Additionally, I employed Random Under-Sampling (RUS) to reduce memory consumption and improve efficiency. I also performed Principal Component Analysis (PCA) before clustering to reduce the dimensionality of the dataset, making it more manageable.

Whenever memory was exhausted, I serialized the variables using joblib library to ensure efficient storage and retrieval, cleared the RAM, or restarted the kernel. This approach allowed me to resume work by reloading the necessary variables without any data loss or performance degradation. Importantly, these techniques did not affect the model's performance, ensuring the best results were achieved without overfitting. These efforts allowed me to successfully navigate memory constraints and complete the tasks, all while ensuring the dataset was handled efficiently and memory errors were eliminated despite its large size.

The experiments were seamlessly executed using the Visual Studio Code with Jupyter Notebook extension. I leveraged the Python programming language and a suite of indispensable libraries, including XGBoost, Pandas, Scikit-learn, NumPy, Matplotlib, Seaborn, Imbalanced-learn etc.

For future improvements, running this project on a machine with a dedicated GPU, increased RAM, and more powerful processing capabilities would significantly enhance performance, reduce the likelihood of system errors, and shorten computation times. This environment would be ideal for handling large datasets and complex models more efficiently.

Performance Evaluation Metrics

The performance of the ML models was evaluated using several metrics, including accuracy, precision, recall, F1-score, ROC curves, and confusion matrices.

- **Confusion Matrix**: The Confusion Matrix is a valuable tool for evaluating ML classification performance. This tool tabulates four outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [66]. Table 14 illustrates a confusion matrix where TP represents correctly forecasted positive values, TN indicates accurately projected negative values, FP corresponds to incorrectly anticipated positive values, and

FN signifies inaccurately predicted negative values. This matrix is essential for assessing Recall, F1-score, Accuracy and Precision.

Table 14: Confusion Matrix

	Actual positive	**Actual negative**
Predicted positive	TP	FP
Predicted negative	FN	TN

- **Accuracy**: Represents the proportion of correct predictions to the total number of predictions.

$$\boldsymbol{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (\mathbf{15})$$

- **Precision**: Measures the ratio of correctly predicted positive observations to all predicted positive observations.

$$\boldsymbol{precision} = \frac{TP}{TP+FP} \qquad \mathbf{(16)}$$

- **Recall**: Reflects the proportion of correctly predicted positive observations to all actual positive observations.

$$\boldsymbol{Recall} = \frac{TP}{TP + FN} \qquad \mathbf{(17)}$$

- **F1-Score**: The harmonic mean of precision and recall.

$$\boldsymbol{F1 - Score} = 2 * \frac{(Precision \cdot Recall)}{(Precision + Recall)} \qquad \mathbf{(18)}$$

- **Receiver Operating Characteristic (ROC) Curve**: ROC curves are widely used two-dimensional plots for evaluating the performance of classifiers [67]. These plots effectively illustrate the trade-off between a classifier's sensitivity and specificity across various classification thresholds. This visualization is particularly valuable for selecting classifiers tailored to specific user requirements, often involving varying error costs and accuracy demands, as highlighted in studies [68, 69]. The area under the curve (AUC) quantifies the discriminative capability of the ROC curve, while the curve itself represents a probability-based evaluation of the model's ability to differentiate between categories. The true positive rate is plotted on the Y-axis, while the false positive rate is displayed on the X-axis. An AUC value nearing 1 indicates excellent discrimination between class labels, whereas a value closer to 0 reflects poor predictive ability, akin to random chance. This approach provides an insightful visualization of classification efficiency, as noted by [70]. Classifiers with higher ROC curves are generally regarded as superior, as supported by findings in [71]**.**

K-Fold Cross-Validation (CV)

K-fold cross-validation (CV) is a commonly used technique that divides the training dataset into k smaller subsets, or folds. The model is iteratively trained on k-1 folds and tested on the remaining fold. This process is repeated k times, ensuring that each fold serves as a testing set once. The overall performance metric is then calculated as the average of the results from all folds. In this study, I employ 10-fold CV, where the dataset is partitioned into 90% training data and 10% testing data for each fold. The k-fold CV process is visually represented in Figure 26.

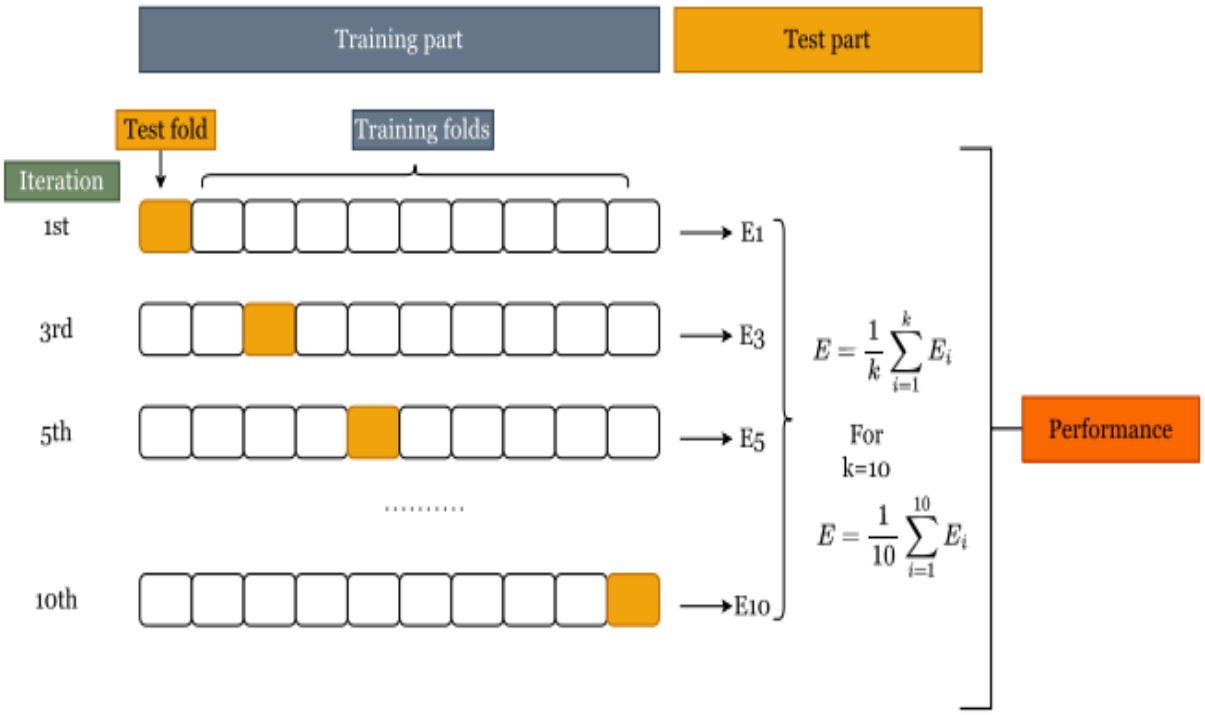

Figure 26: K-fold cross-validation process

6 Results and Discussion

6.1 Multiclass Classification

The multiclass classification task in this study involved highly imbalanced data, as shown by the original class distribution. The Benign class constituted 67.57% of the total dataset, followed by DoS Hulk at 17.22%, and DDoS HOIC at 10.26%. Meanwhile, the minority classes such as Web Attack - SQL Injection, FTP-Patator, and DoS Slowhttptest had extremely low occurrences, contributing less than 0.005% of the dataset. Due to this severe imbalance, handling the dataset imbalance was essential to improve model performance and prevent biases toward majority classes. To address the imbalance, two approaches were utilized. The first approach was Random Under Sampling (RUS) with Equal Class Distribution, where all classes were reduced to have the same number of instances. The second approach was Stratified Under Sampling, where the dataset was reduced while maintaining the proportional representation of each class. The results for both scenarios are analysed below.

6.1.1 Performance with Equal Class Distribution

When all classes were under-sampled to have an equal number of instances, the Decision Tree model achieved 98.28% accuracy, with precision, recall, and F1 scores all closely aligned at 98.28%. Extra Trees performed slightly better, reaching an accuracy of 98.79%, while Random Forest and XGBoost followed with 98.71% and 98.58% accuracy, respectively. The mean ROC scores remained high across all models, with minor variations.

In addition to classification performance, training efficiency varied significantly among the models. XGBoost demonstrated the fastest training time, leveraging its optimized parallelization capabilities, which are particularly beneficial for large datasets. Decision Tree was also computationally efficient, training faster than Extra Trees but slower than XGBoost. Conversely, Random Forest exhibited the slowest training speed due to its ensemble nature, requiring multiple decision trees to be trained independently before aggregation. These computational differences are crucial when considering model deployment in resource-constrained environments or real-time applications. The overall classification performance for this scenario is summarized in Table 15, highlighting each model's accuracy, precision, recall, F1-score, and mean ROC values.

Table 15: Multiclass Classification Performance with Equal Class Distribution

Model	Accuracy	Precision	Recall	F1-score	Mean ROC
DT	**98.28%**	**98.30%**	**98.28%**	**98.27%**	**0.99 ± 0.02**
ET	**98.79%**	**98.80%**	**98.79%**	**98.78%**	**0.99 ± 0.00**
RF	**98.71%**	**98.73%**	**98.71%**	**98.70%**	**0.99 ± 0.00**
XGB	**98.58%**	**98.59%**	**98.58%**	**98.57%**	**0.99 ± 0.00**

To further evaluate model performance, confusion matrices for each classifier offer insights into their classification capabilities. The confusion matrices for Decision Tree (DT), Extra Trees (ET), Random Forest (RF), and XGBoost (XGB) are illustrated in Figure 27, Figure 28, Figure 29, and Figure 30, respectively. These matrices reveal strong classification performance, characterized by high true positive (TP) and true negative (TN) rates, alongside minimal

false positive (FP) and false negative (FN) occurrences, indicating effective differentiation across multiple attack categories.

Additionally, Receiver Operating Characteristic (ROC) curves for DT, ET, RF, XGB are shown in Figure 31, Figure 32, Figure 33, and Figure 34, respectively. It further validates the robustness of these models. The curves demonstrate near-optimal AUC values, reinforcing their ability to distinguish between attack types after applying random under-sampling with equal class distribution.

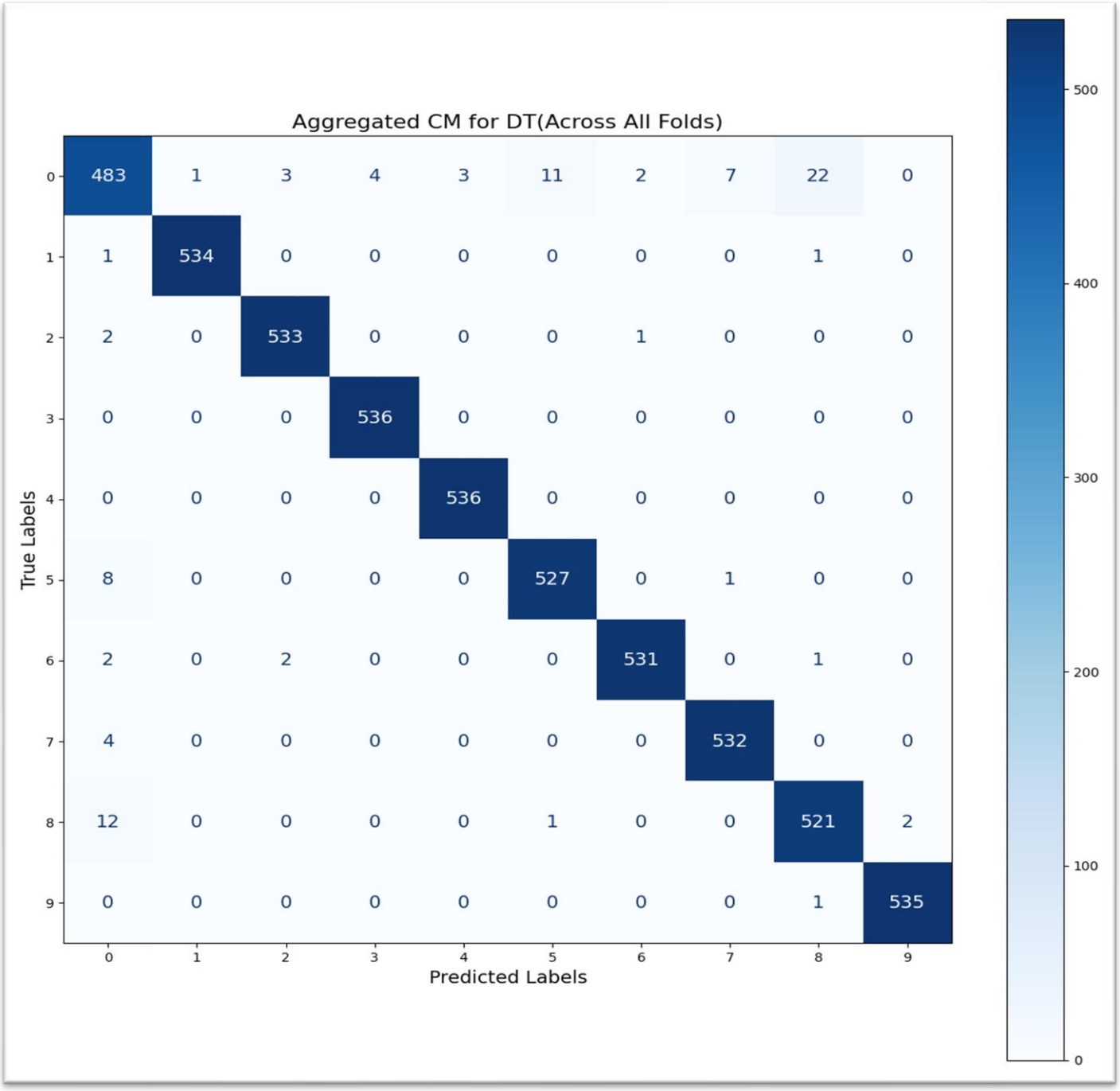

Figure 27: Confusion Matrix for Decision Tree (Multiclass – Equal Class Distribution)

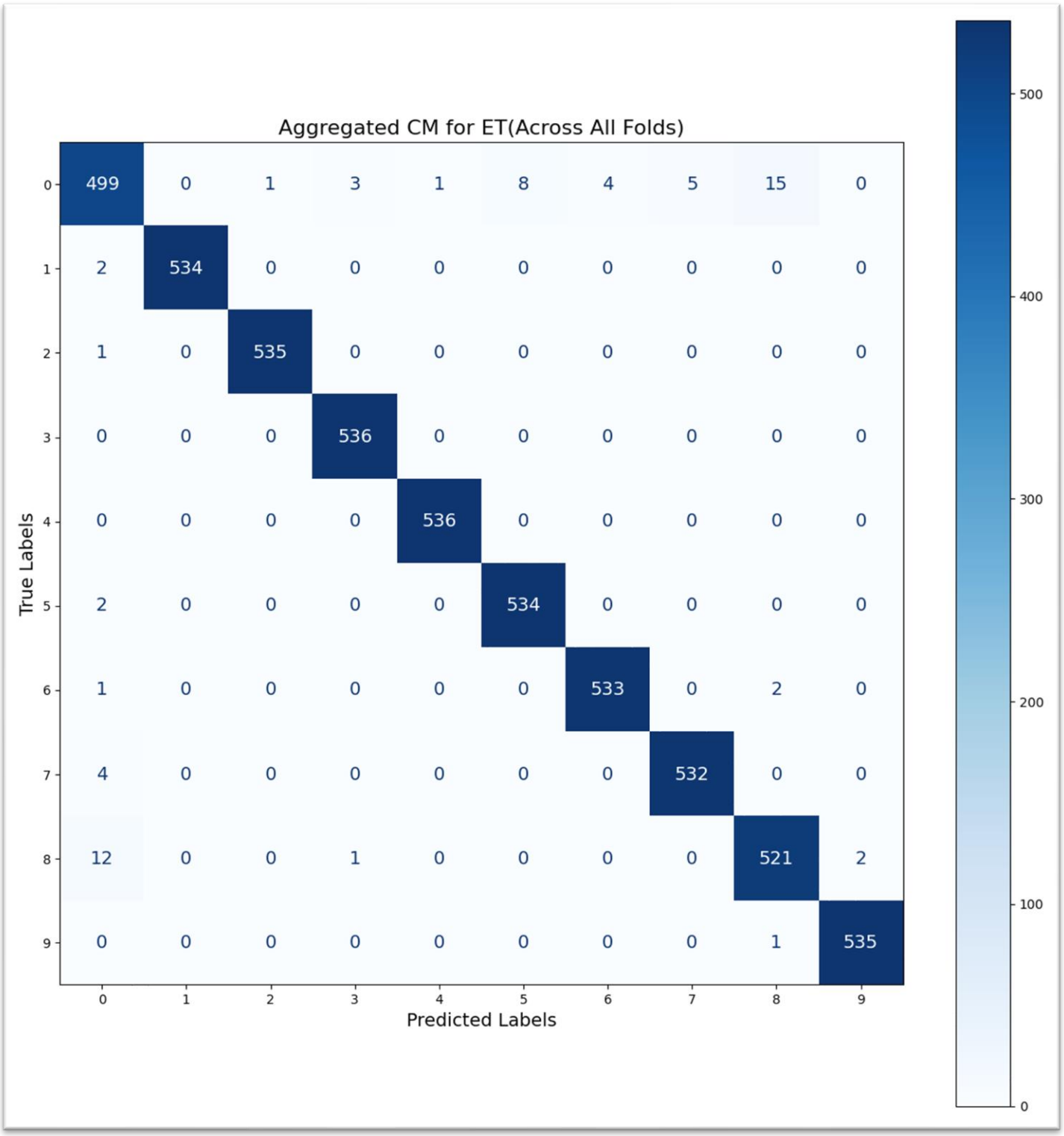

Figure 28: Confusion Matrix for Extra Trees (Multiclass – Equal Class Distribution)

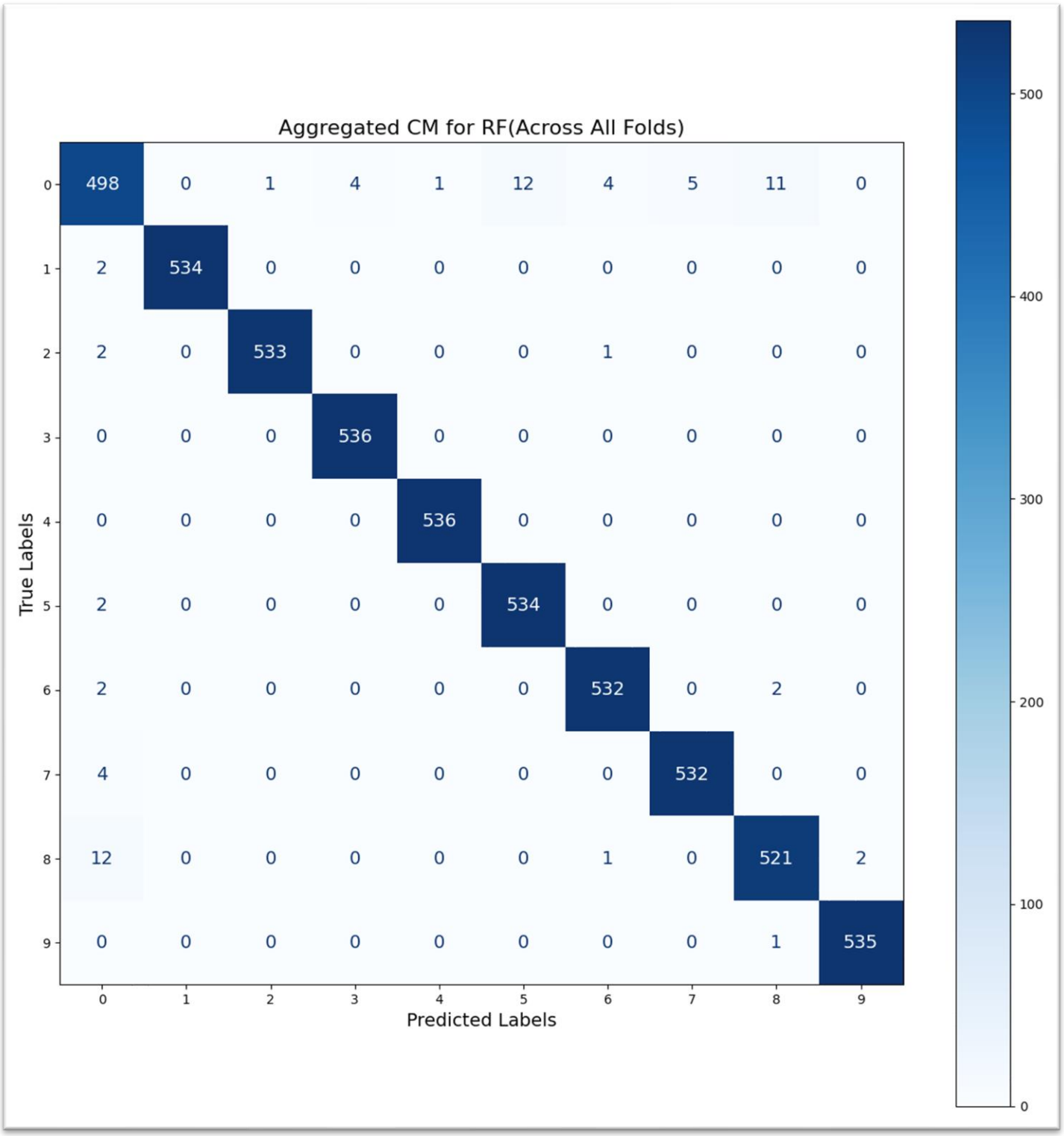

Figure 29: Confusion Matrix for Random Forest (Multiclass – Equal Class Distribution)

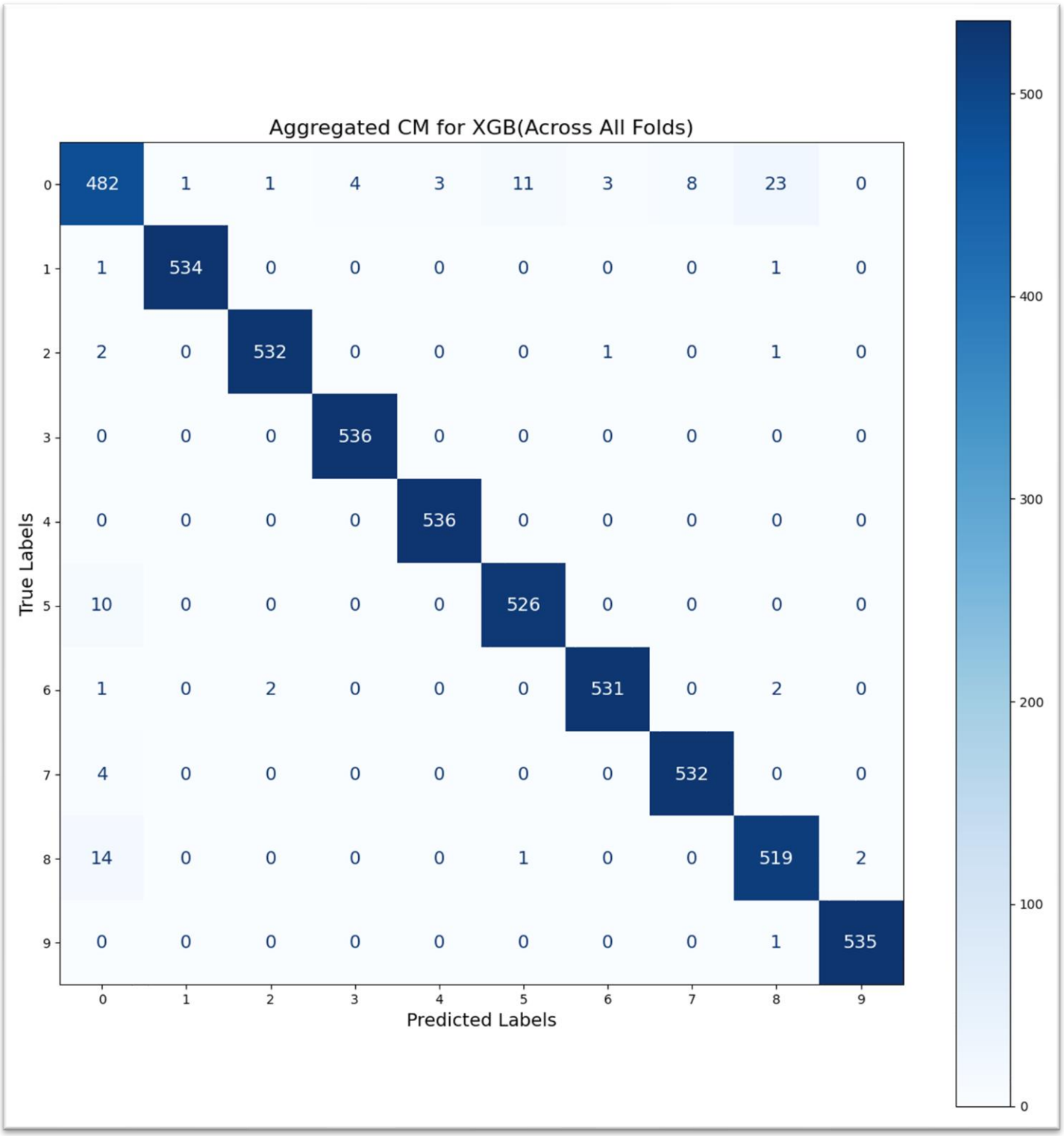

Figure 30: Confusion Matrix for XGBoost (Multiclass – Equal Class Distribution)

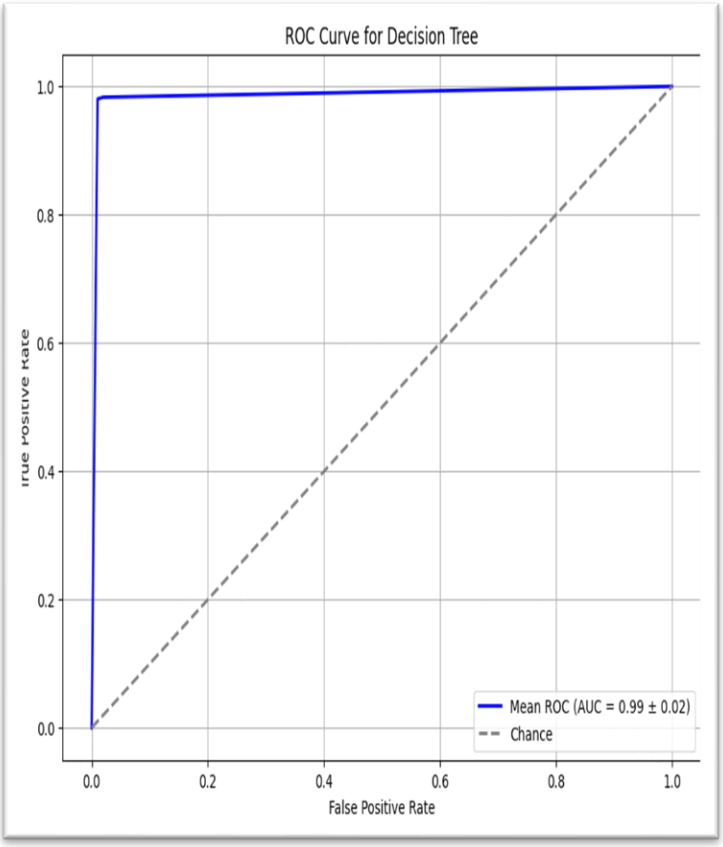

Figure 31: ROC Curve for Decision Tree (Multiclass – Equal Class Distribution)

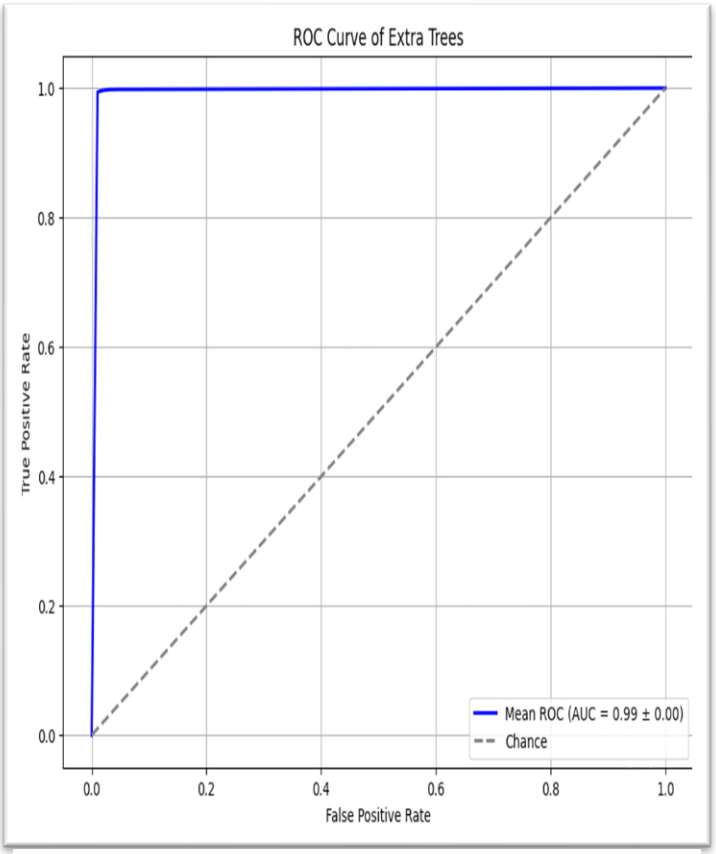

Figure 32: ROC Curve for Extra Trees (Multiclass – Equal Class Distribution)

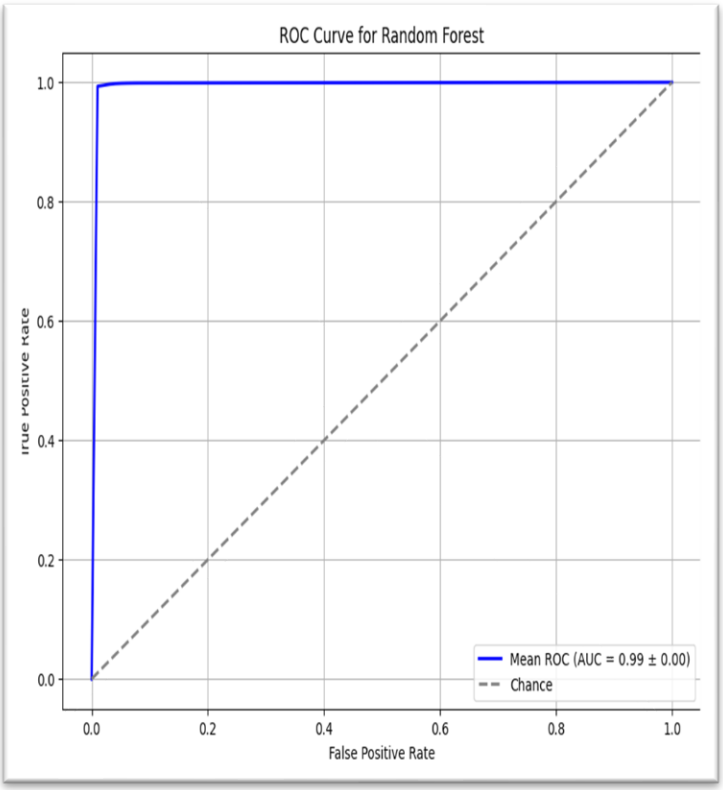

Figure 33: ROC Curve for Random Forest (Multiclass – Equal Class Distribution)

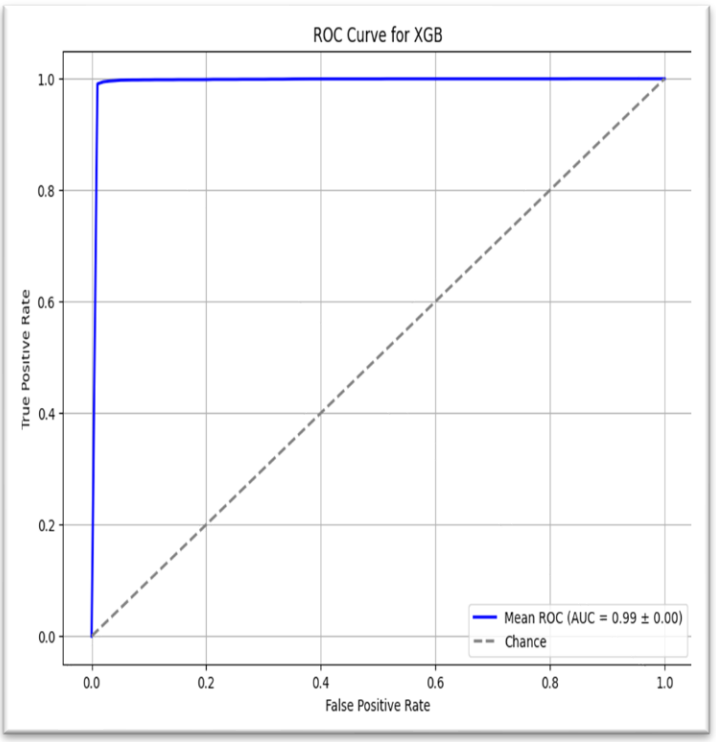

Figure 34: ROC Curve for XGBoost (Multiclass - Equal Class Distribution)

6.1.2 Performance with Stratified Under-Sampling

In the second approach, the proportional distribution for each class were preserved while reducing the dataset size. This method yielded even better performance, with Decision Tree reaching an accuracy of 99.78%, and Extra Trees, Random Forest, and XGBoost all exceeding 99.87%. The Mean ROC scores remained consistent, indicating that this method retained the class distribution advantage while improving model generalization. Stratified under-sampling is considered more reliable as it mirrors real-world data, which typically exhibits imbalances across classes, making these results more trustworthy and applicable in practical scenarios. The detailed classification performance for this approach is summarized in Table 16, showcasing accuracy, precision, recall, F1-score, and mean ROC values.

Table 16: Multiclass Classification Performance with Stratified Under-Sampling

Model	Accuracy	Precision	Recall	F1-score	Mean ROC
DT	99.78%	99.78%	99.78%	99.78%	0.98 ± 0.04
ET	99.87%	99.88%	99.87%	99.87%	0.99 ± 0.02
RF	99.89%	99.89%	99.89%	99.89%	0.99 ± 0.01
XGB	99.89%	99.90%	99.89%	99.89%	0.99 ± 0.02

The impact of Stratified Under-Sampling on model performance was further examined using confusion matrices and ROC curves. The confusion matrices for DT, ET, RF, and XGB are presented in Figure 35, Figure 36, Figure 37 and Figure 38, respectively demonstrating strong classification accuracy, with consistently high TP/TN rates and minimal FP/FN occurrences. This confirms that the models maintain effective attack detection even when the dataset retains its relative class proportions. Similarly, the ROC curves, depicted in Figure 39, Figure 40, Figure 41, and Figure 42 reflect high AUC values across all models, confirming their ability to differentiate attack categories. This indicates that stratified under-sampling allows models to generalize well while preserving the relative distribution of attack classes.

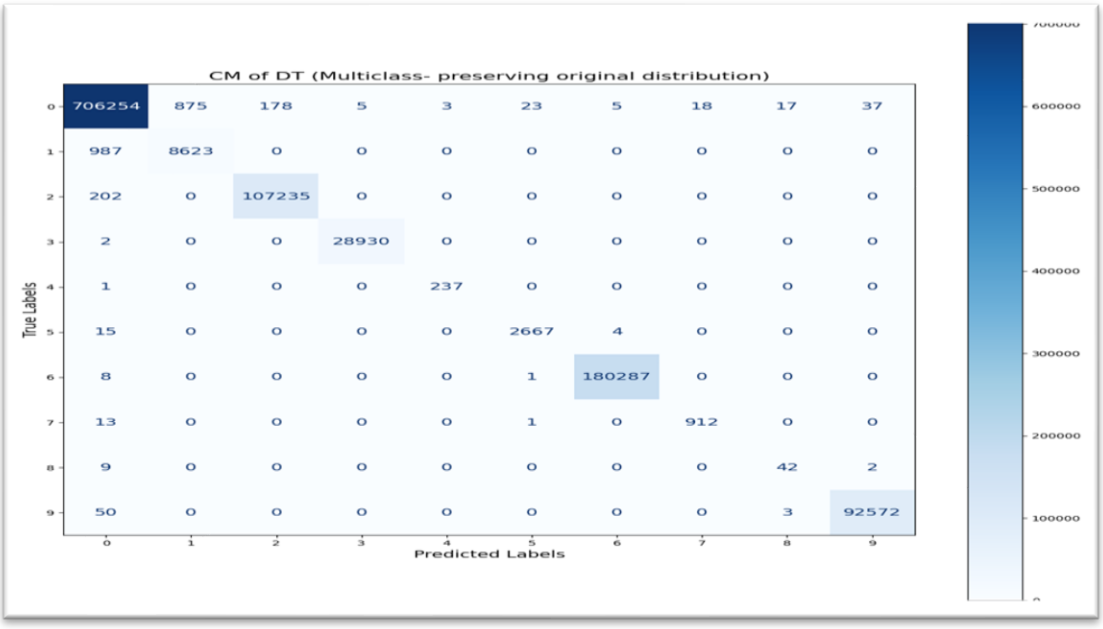

Figure 35: Confusion Matrix for Decision Tree (Multiclass – Stratified Under-Sampling)

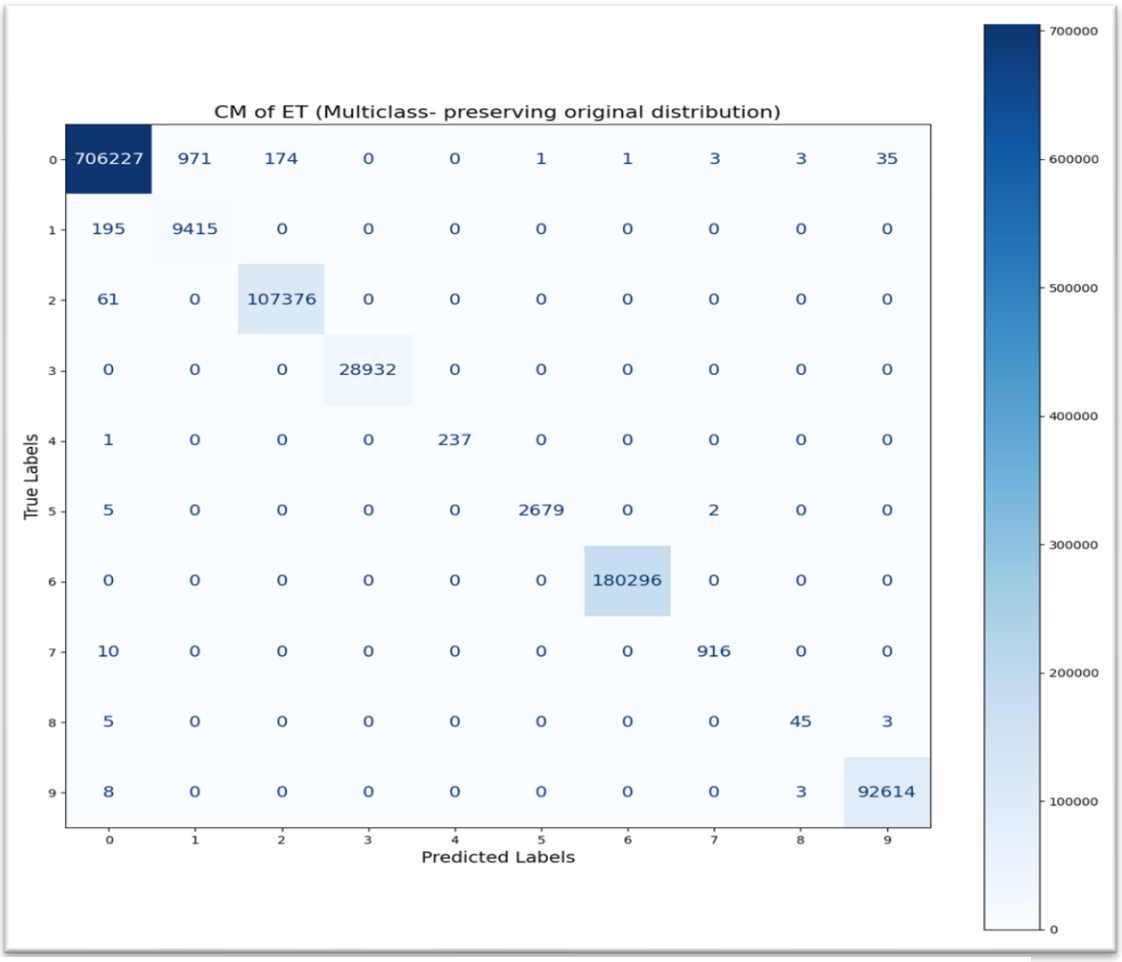

Figure 36: Confusion Matrix for Extra Trees (Multiclass – Stratified Under-Sampling)

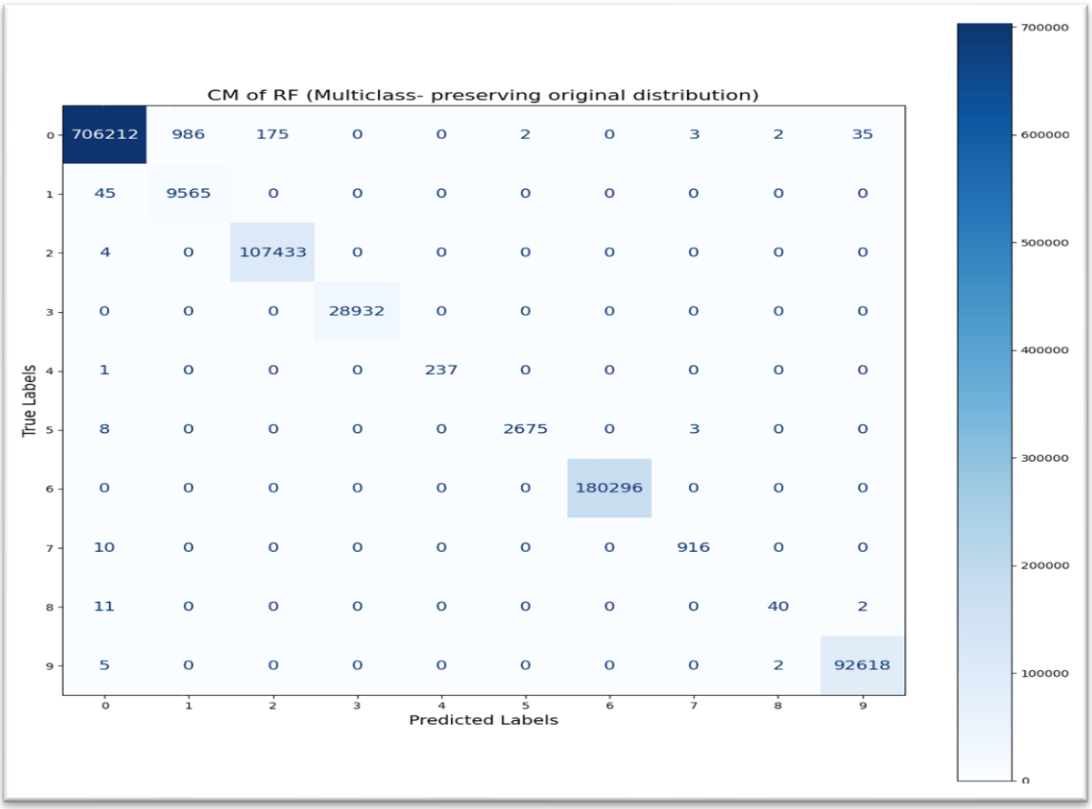

Figure 37: Confusion Matrix for Random Forest (Multiclass – Stratified Under-Sampling)

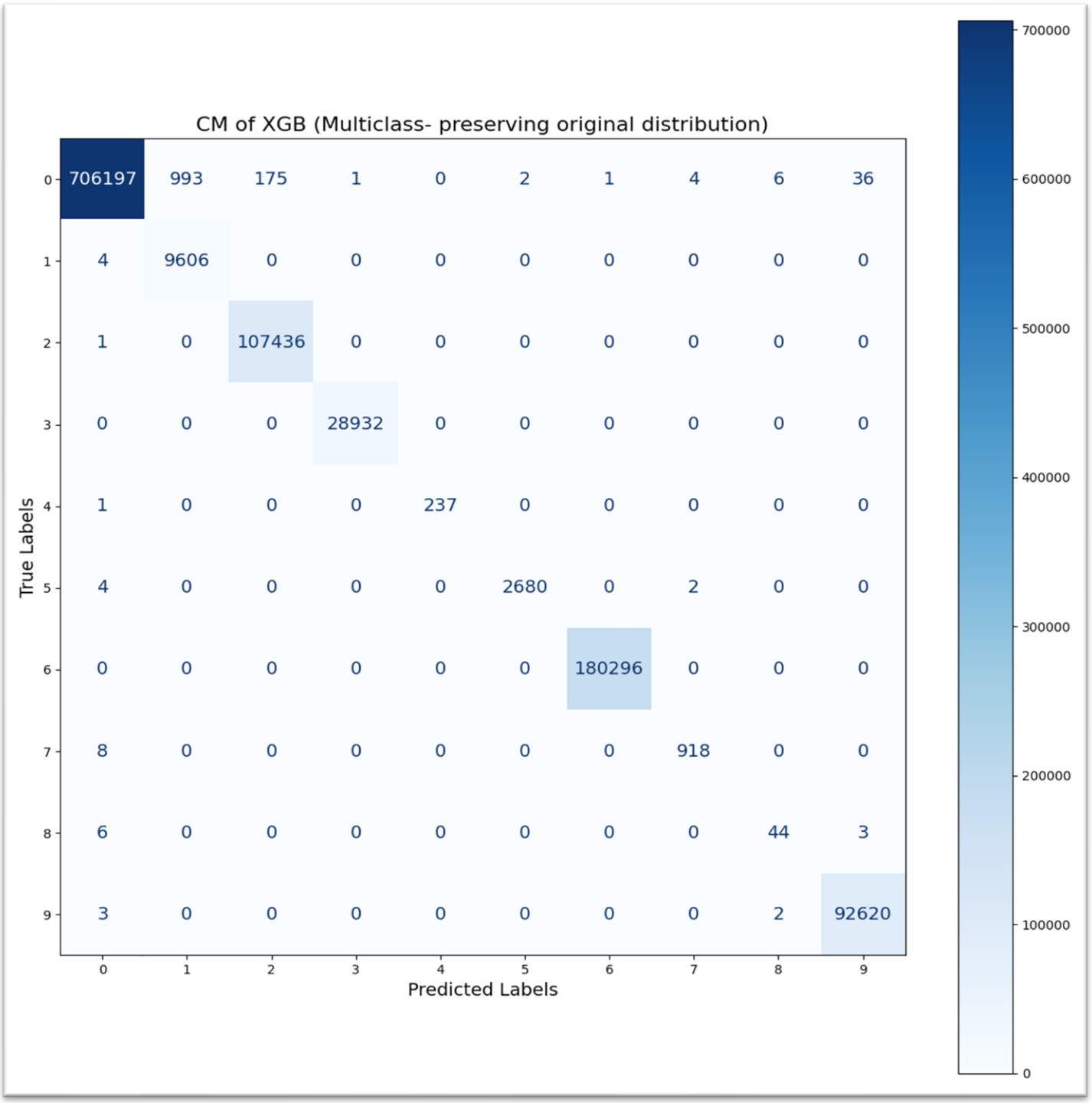

Figure 38: Confusion Matrix for XGBoost (Multiclass – Stratified Under-Sampling)

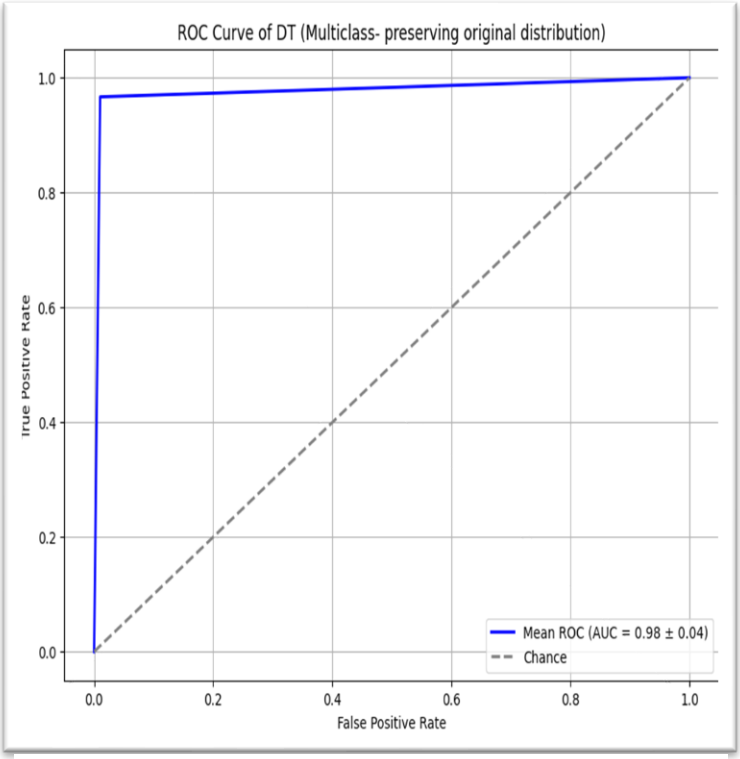

Figure 39: ROC Curve for Decision Tree (Multiclass – Stratified Under-Sampling)

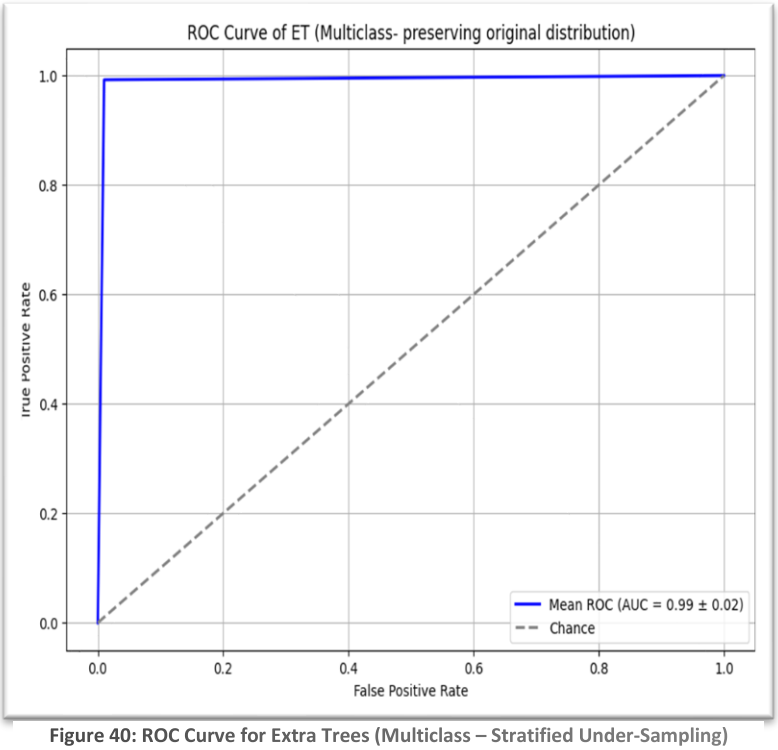

Figure 40: ROC Curve for Extra Trees (Multiclass – Stratified Under-Sampling)

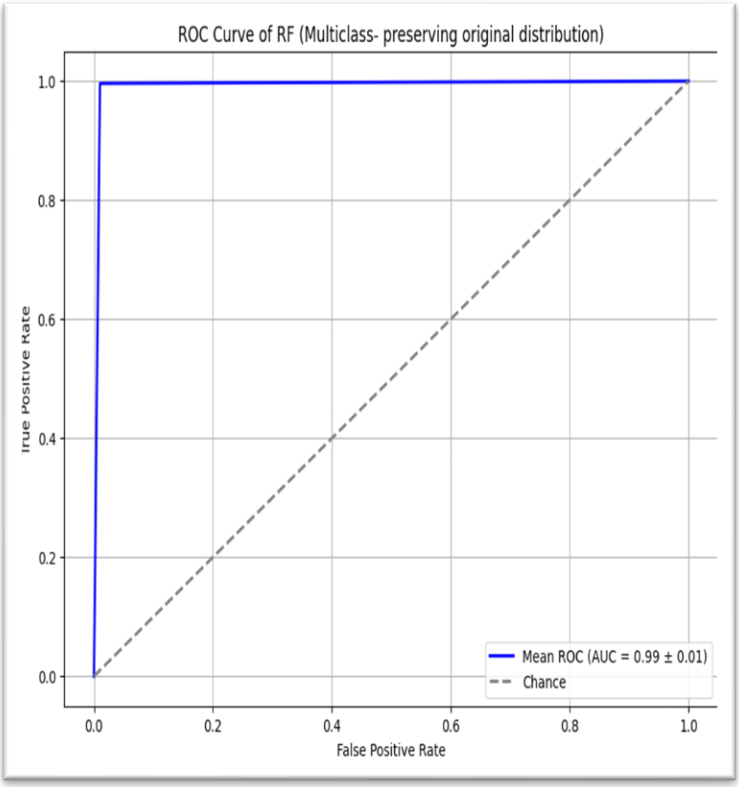

Figure 41: ROC Curve for Random Forest (Multiclass – Stratified Under-Sampling)

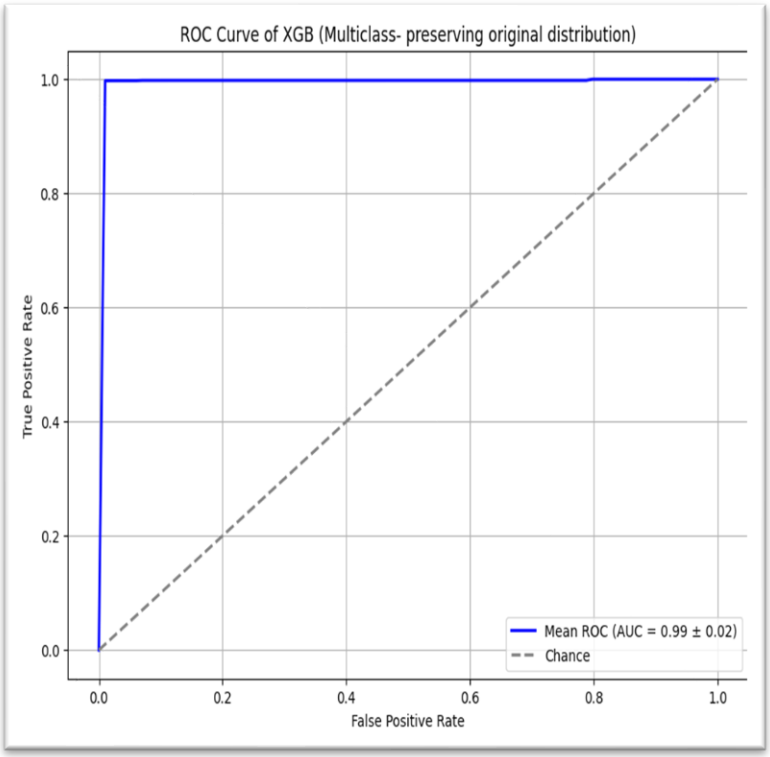

Figure 42: ROC Curve for XGBoost (Multiclass – Stratified Under-Sampling)

6.2 Binary Classification

The original binary class distribution was 67.6% non-intrusion (Benign) and 32.4% intrusion (Attack), indicating that the dataset was not severely imbalanced. However, due to computational constraints and to reduce resource requirements, two under-sampling techniques were applied: Equal Class Distribution Under-Sampling, where both classes were reduced to the same number of samples, and Proportionally Preserved Under-Sampling, where the class ratio was maintained but with fewer total instances.

6.2.1 Performance with Equal Class Distribution

For the scenario where both classes had an equal number of instances, Decision Tree achieved 99.82% accuracy, while Extra Trees, Random Forest, and XGBoost reached an impressive 99.91% across all metrics. The ROC scores remained consistent, indicating high classification effectiveness. The overall classification metrics for this scenario are detailed in Table 17, outlining accuracy, precision, recall, F1-score, and mean ROC values.

Table 17: Binary Classification Performance with Equal Class Distribution

Model	Accuracy	Precision	Recall	F1-score	Mean ROC
DT	99.82%	99.82%	99.82%	99.82%	0.99 ± 0.00
ET	99.91%	99.91%	99.91%	99.91%	0.99 ± 0.00
RF	99.91%	99.91%	99.91%	99.91%	0.99 ± 0.00
XGB	99.91%	99.91%	99.91%	99.91%	0.99 ± 0.00

For the binary classification task, the confusion matrices for DT, ET, RF, and XGB are illustrated in Figure 43, Figure 44, Figure 45, and Figure 46, respectively. These matrices reveal high TP and TN rates, along with minimal FP and FN occurrences, underscoring the models' ability to accurately differentiate between benign and attack traffic when using equal class distribution under-sampling. Likewise, ROC curves, presented in Figure 47, Figure 48, Figure 49, and Figure 50 confirm the models' strong predictive performance, with AUC values approaching 1.0, demonstrating robust classification between the two classes.

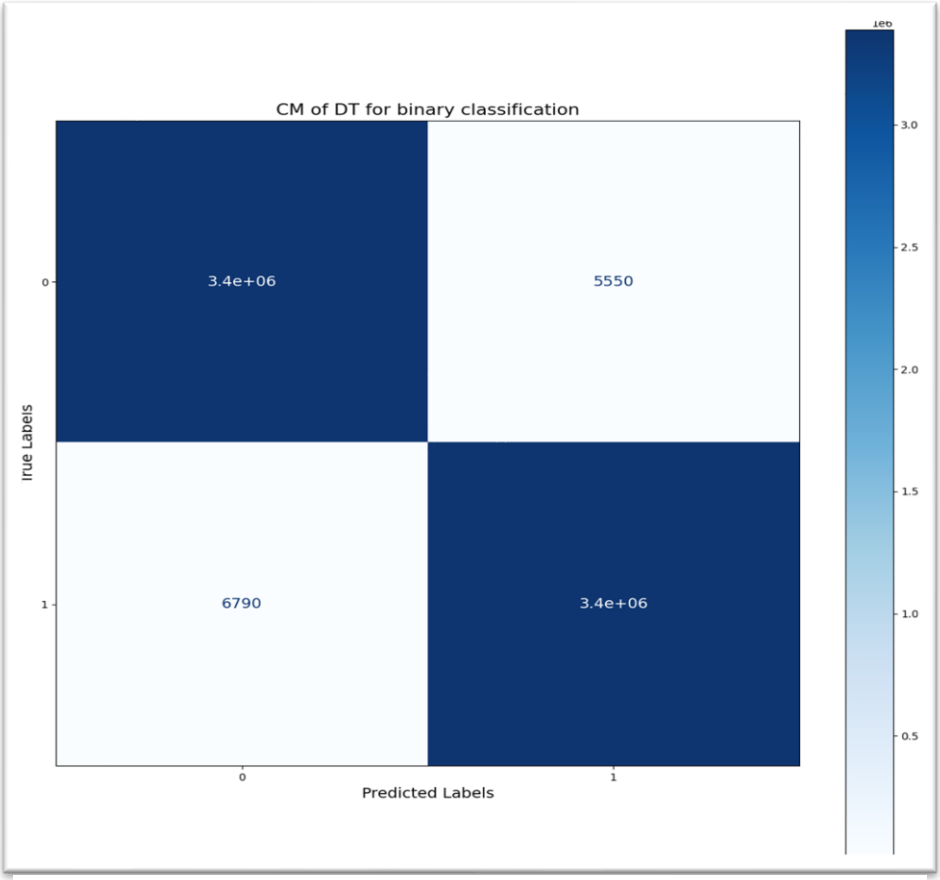

Figure 43: Confusion Matrix for Decision Tree (Binary Classification – Equal Class Distribution)

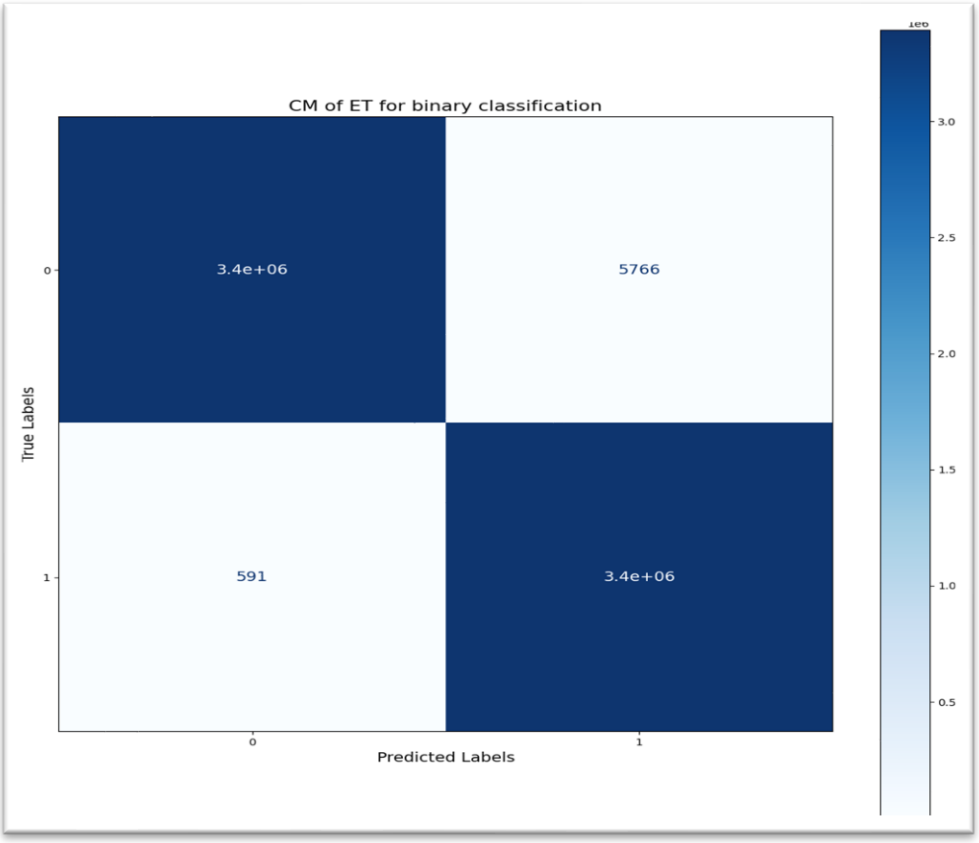

Figure 44: Confusion Matrix for Extra Trees (Binary Classification – Equal Class Distribution)

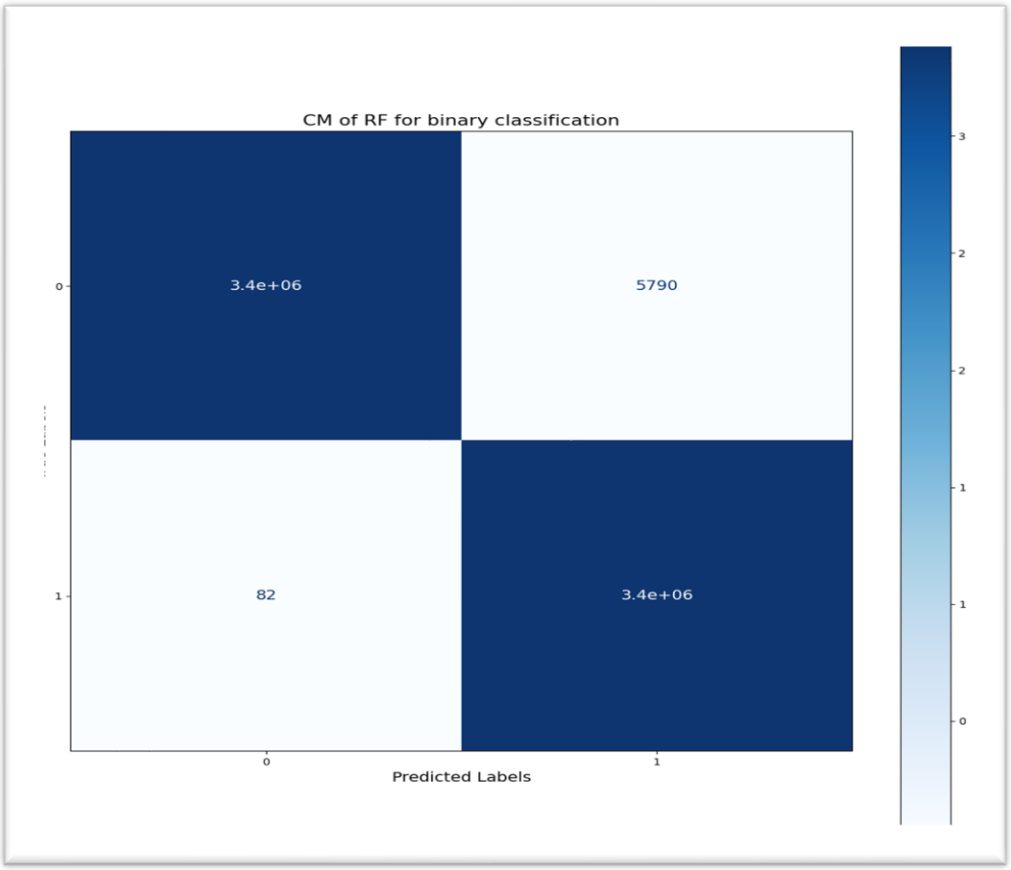

Figure 45: Confusion Matrix for Random Forest (Binary Classification – Equal Class Distribution)

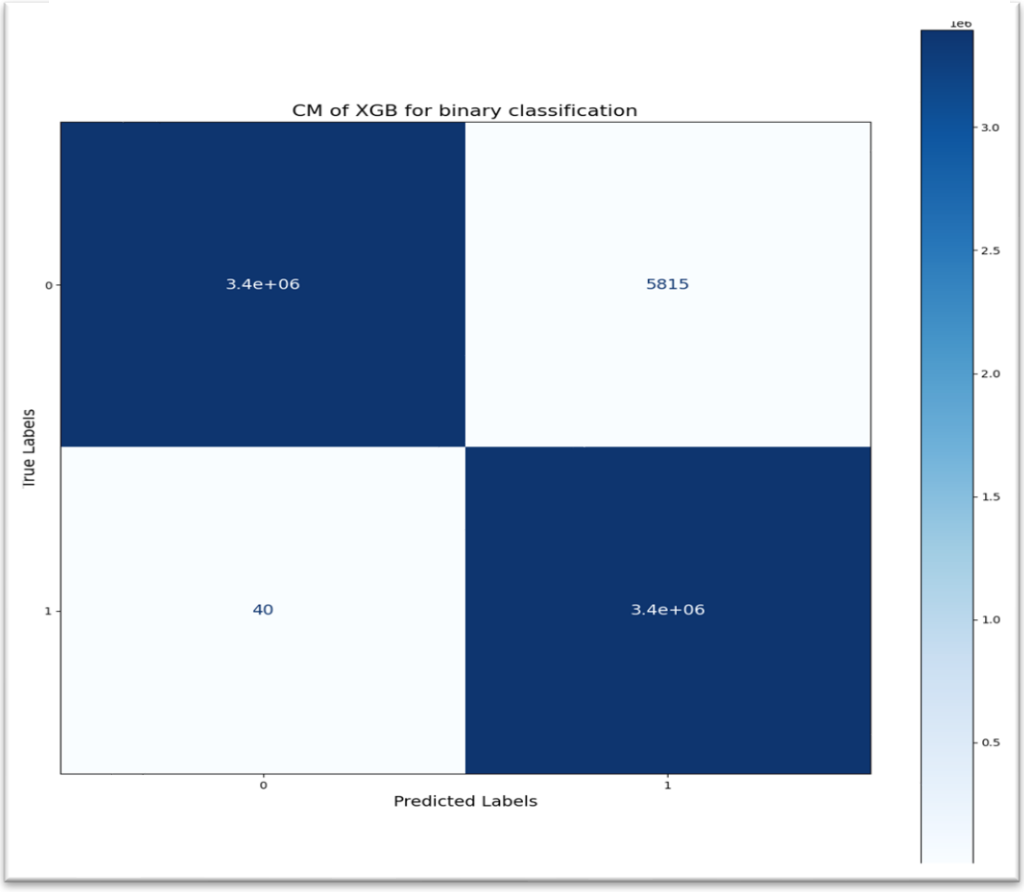

Figure 46: Confusion Matrix for XGBoost (Binary Classification – Equal Class Distribution)

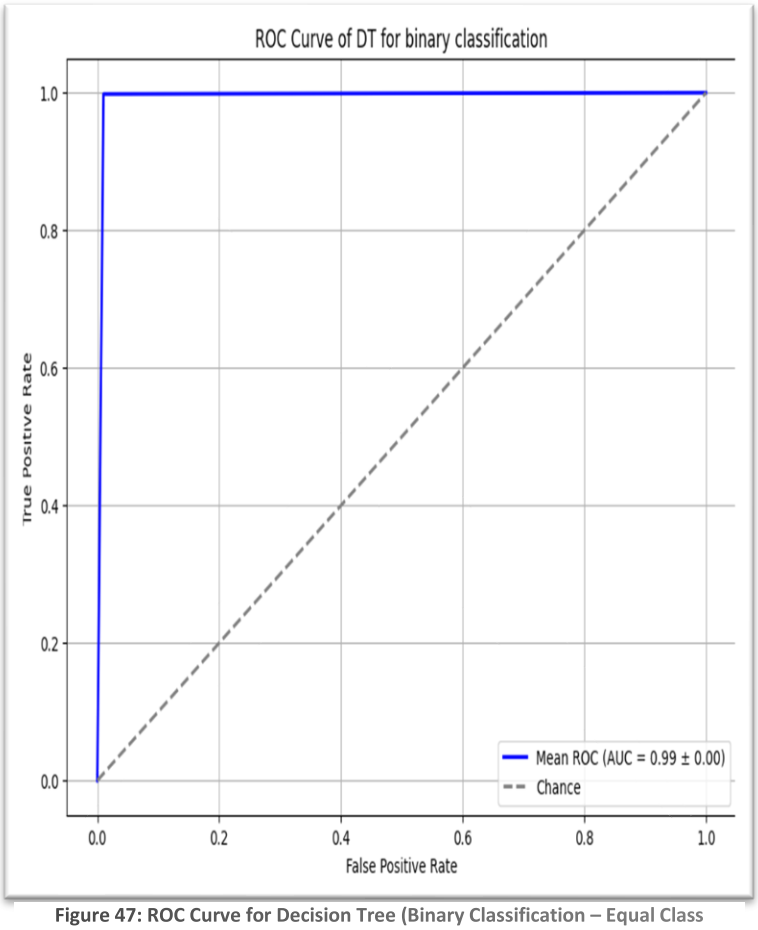

Figure 47: ROC Curve for Decision Tree (Binary Classification – Equal Class Distribution)

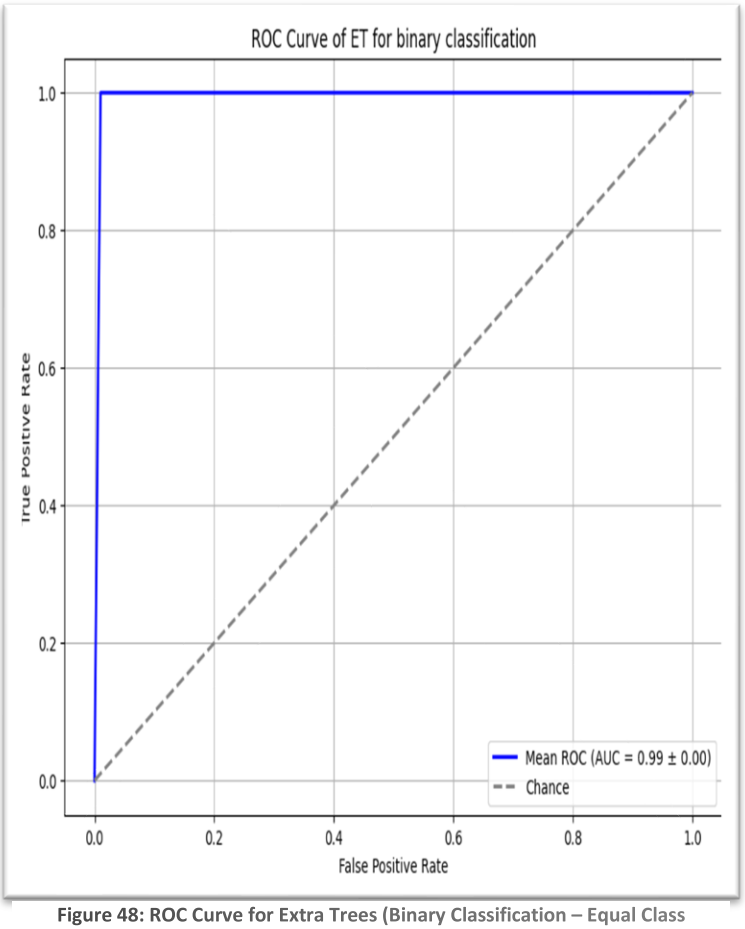

Figure 48: ROC Curve for Extra Trees (Binary Classification – Equal Class Distribution)

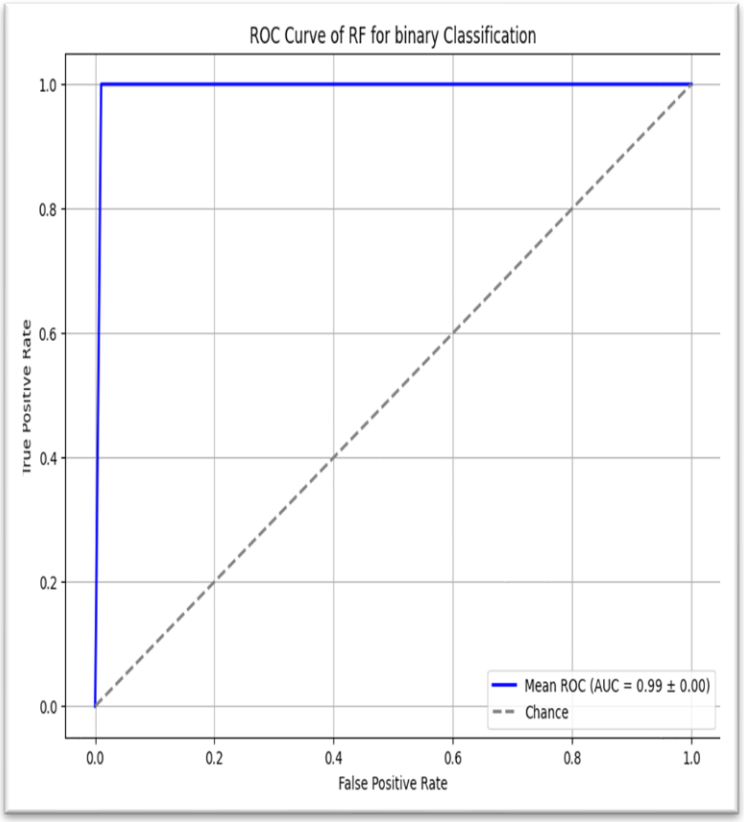

Figure 49: ROC Curve for Random Forest (Binary Classification - Equal Class Distribution)

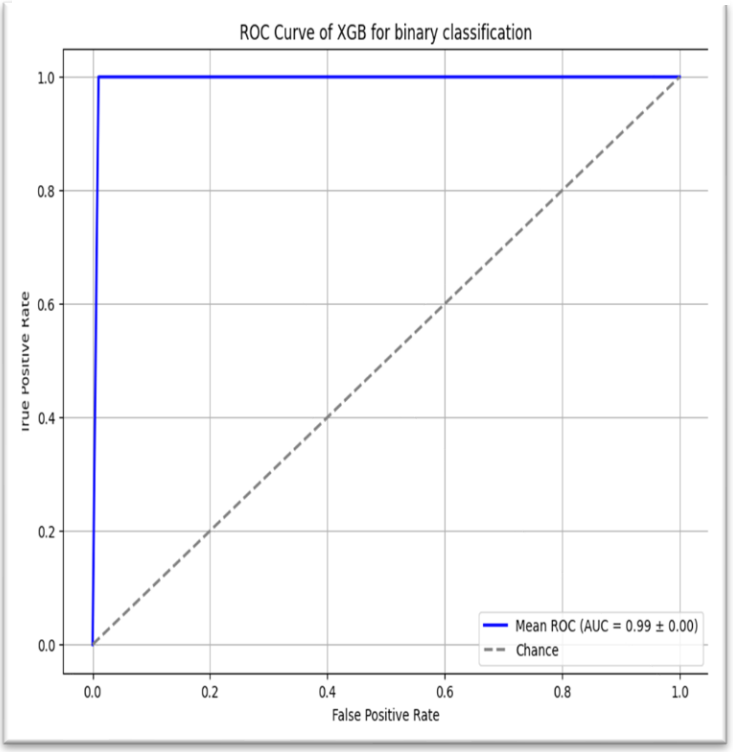

Figure 50: ROC Curve for XGBoost (Binary Classification - Equal Class Distribution)

6.2.2 Performance with Proportionally Preserved Under-Sampling

In the proportionally preserved scenario, all models exhibited consistently high performance. Decision Tree achieved 99.77% accuracy, while Extra Trees, Random Forest, and XGBoost slightly outperformed with accuracies of 99.87%, 99.88%, and 99.88%, respectively. A comprehensive performance evaluation is provided in Table 18, demonstrating the impact of the applied sampling strategy on model accuracy and predictive capability.

Table 18: Binary Classification Performance with Proportionally Preserved Under-Sampling

Model	Accuracy	Precision	Recall	F1-score	Mean ROC
DT	99.77%	99.77%	99.77%	99.77%	0.99 ± 0.00
ET	99.87%	99.87%	99.87%	99.87%	0.99 ± 0.00
RF	99.88%	99.88%	99.88%	99.88%	0.99 ± 0.00
XGB	99.88%	99.88%	99.88%	99.88%	0.99 ± 0.00

For Stratified Under-Sampling, the confusion matrices for DT, ET, RF, and XGB, illustrated in Figure 51, Figure 52, Figure 53, and Figure 54, respectively indicate strong classification performance, with high TP/TN values and low FP/FN rates, demonstrating the effectiveness of this approach in handling class imbalance while preserving relative proportions. Similarly, ROC curves, depicted in Figure 55, Figure 56, Figure 57, and Figure 58 confirm that all models achieve high AUC values, ensuring reliable classification even when attack-to-benign ratios are maintained.

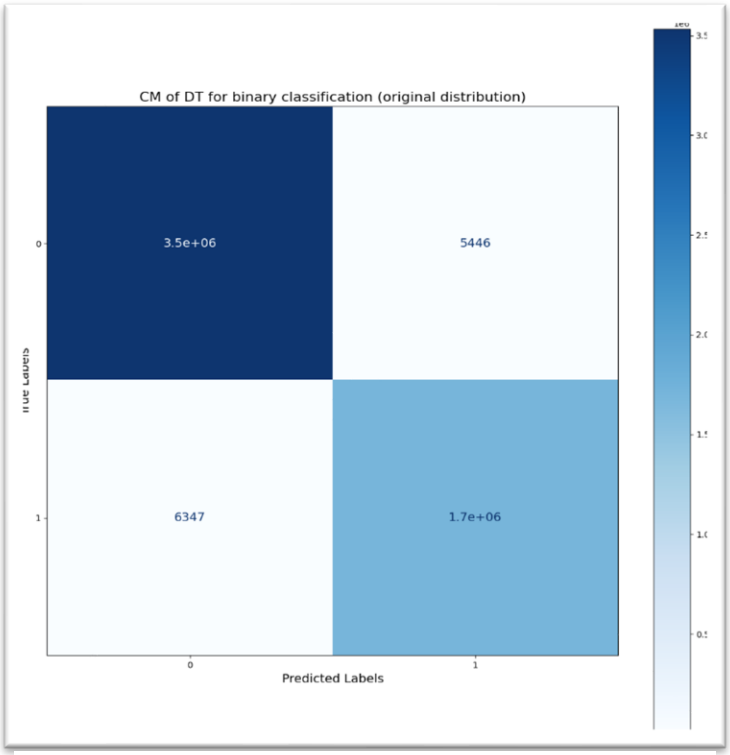

Figure 51: Confusion Matrix for Decision Tree (Binary Classification – Stratified Under-Sampling)

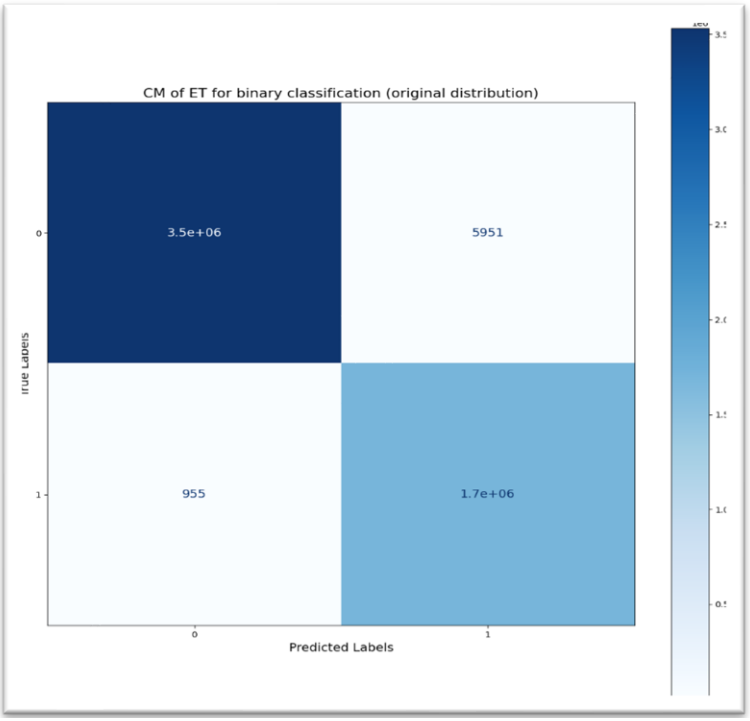

Figure 52: Confusion Matrix for Extra Trees (Binary Classification – Stratified Under-Sampling)

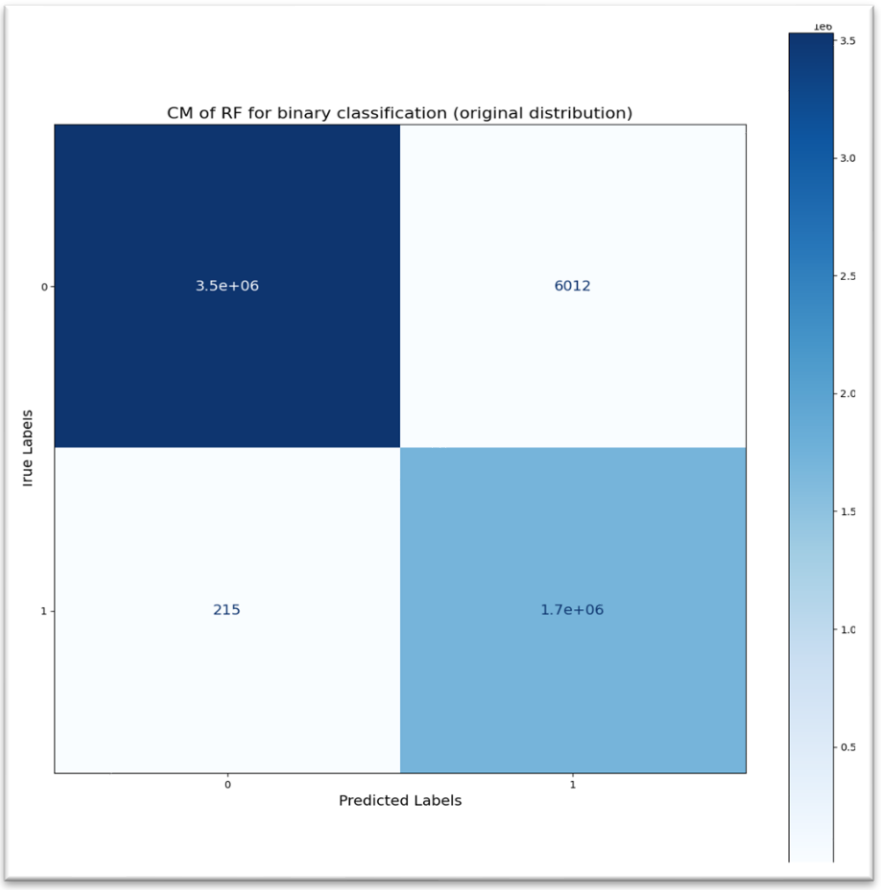

Figure 53: Confusion Matrix for Random Forest (Binary Classification – Stratified Under-Sampling)

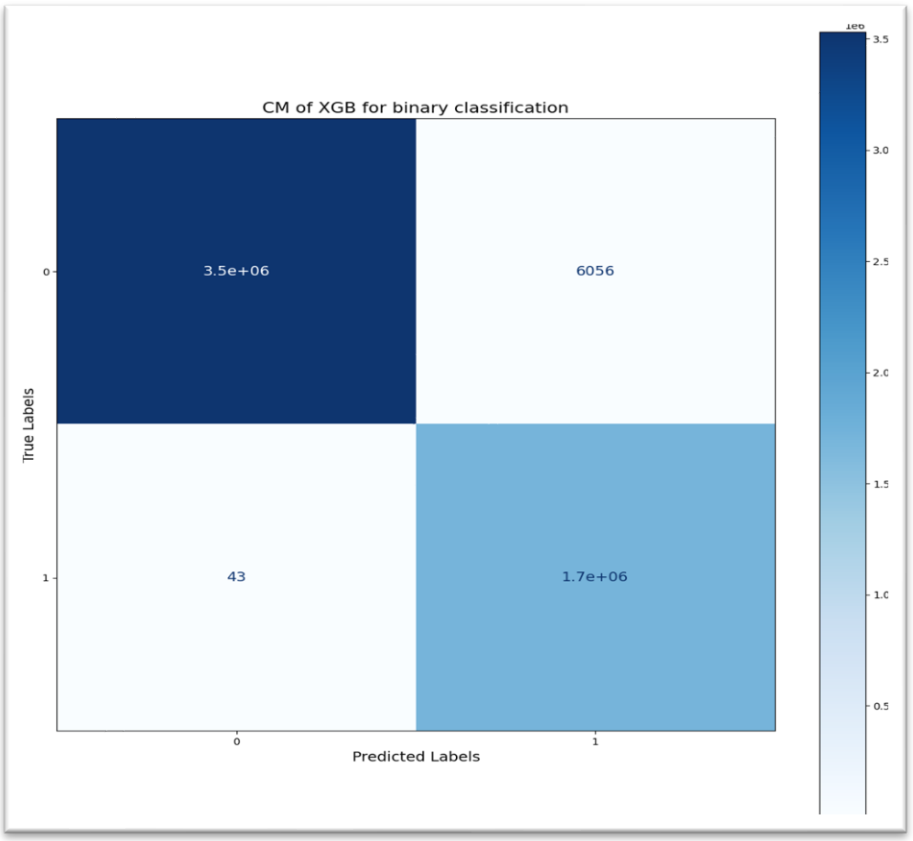

Figure 54: Confusion Matrix for XGBoost (Binary Classification – Stratified Under-Sampling)

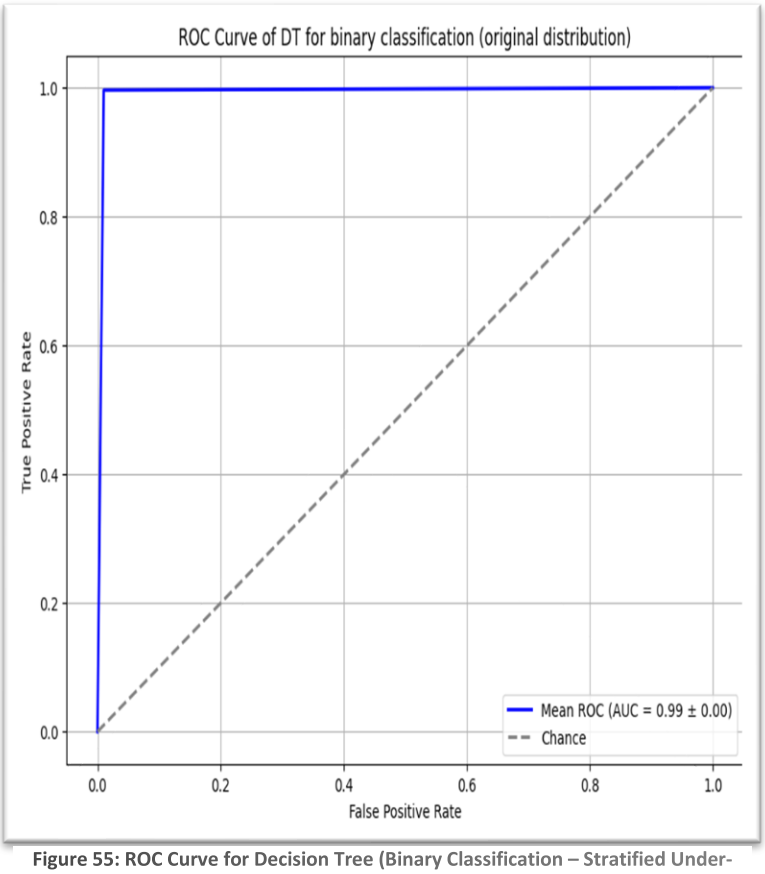

Figure 55: ROC Curve for Decision Tree (Binary Classification – Stratified Under-Sampling)

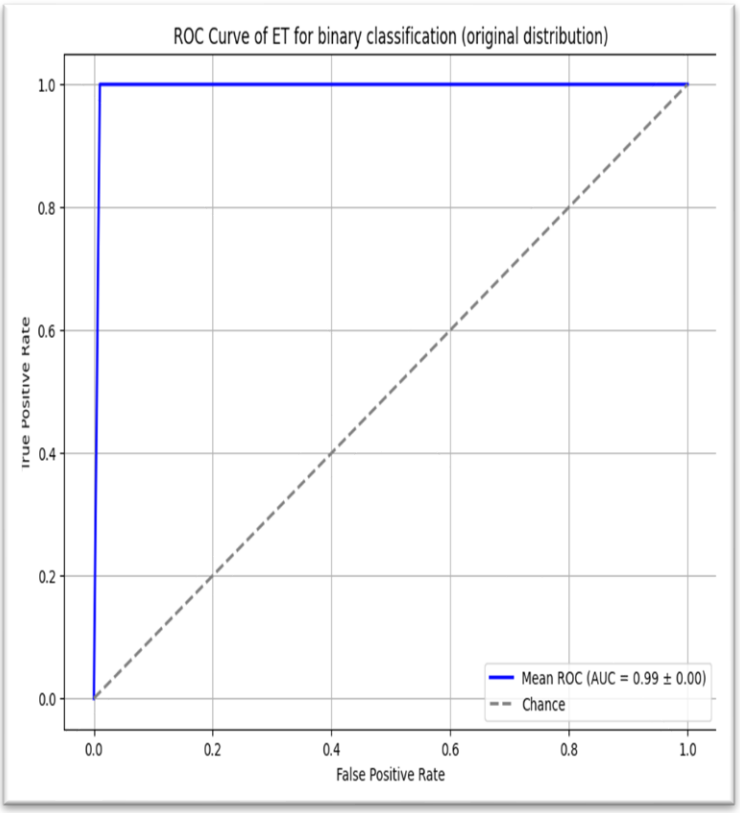

Figure 56: ROC Curve for Extra Trees (Binary Classification – Stratified Under-Sampling)

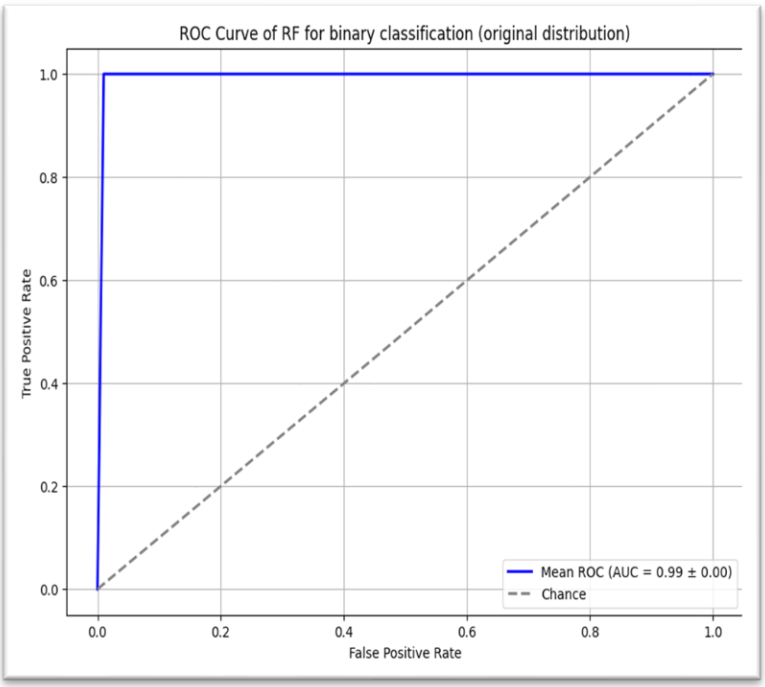

Figure 57: ROC curve for RF (Binary Classification - Stratified Under-Sampling)

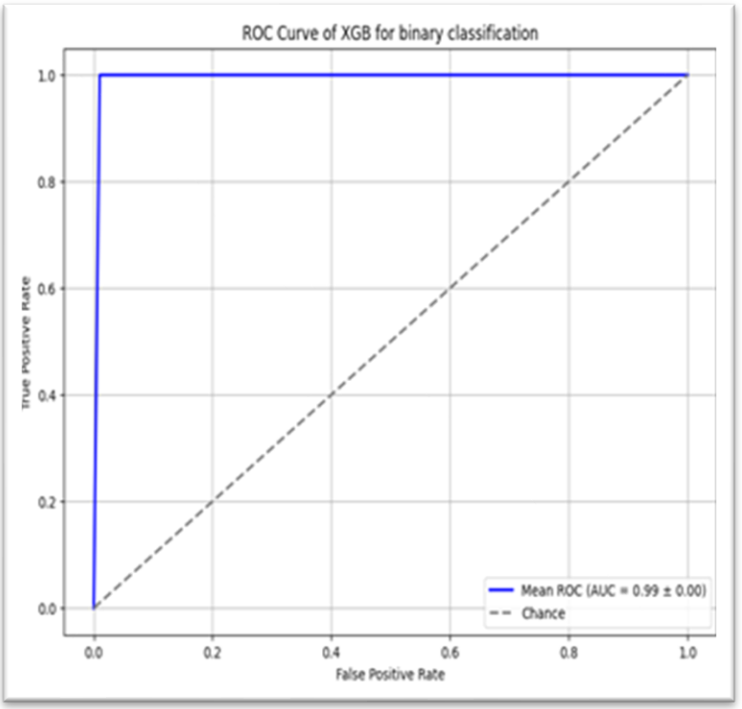

Figure 58: ROC Curve for XGBoost (Binary Classification -Stratified Under-Sampling)

6.3 Results Summary

This study applied two distinct sampling techniques to mitigate dataset imbalance: Equal Class Distribution Under-Sampling, which ensured all classes had the same sample size, and Proportionally Preserved Under-Sampling, which reduced dataset size while maintaining the original class proportions. The results demonstrated that both approaches led to exceptional model performance, with minimal differences in classification metrics. While ensemble models such as Extra Trees and Random Forest consistently outperformed single-tree models like Decision Tree, all classifiers achieved high accuracy, precision, recall, and F1 scores.

Beyond classification effectiveness, computational efficiency played a key role in evaluating model suitability for practical deployment. XGBoost exhibited the fastest training time, making it highly scalable and suitable for large datasets. Decision Tree, while not as fast as XGBoost, maintained a relatively low computational overhead. Extra Trees required more training time due to its reliance on randomized feature selection, while Random Forest was the slowest due to its extensive ensemble structure. These findings suggest that while all models performed well in terms of classification, XGBoost provides a strong balance between accuracy and training efficiency, making it a compelling choice for real-world network intrusion detection systems.

In addition, Stratified Under-Sampling is considered more reliable than equal class distribution under-sampling, as it better reflects real-world data distributions, where class imbalance often occurs. This makes the results from the stratified approach not only highly accurate but also trustworthy for deployment in practical, imbalanced environments.

6.4 Discussion

In this study, I conducted a comparative analysis of my proposed model in relation to the only existing work that uses the LycoS-Unicas-IDS2018 dataset—Cantone et al. (2024) [16]. The results of this comparison, detailed in Table 19, demonstrate the effectiveness of my approach, which integrates SEF-PCA for feature extraction and Random Under-Sampling (RUS) for class balancing. This methodology achieved strong performance across both binary and multiclass classification tasks, showcasing its robustness in handling class imbalance and large-scale datasets.

Table 19: Results Comparison with the Existing Study

Study	Classification Type	Class Balancing	Feature Extraction	Classifiers	F1 Score (%)
This study	Multiclass	Stratified RUS	SFE-PCA	DT	99.78
				ET	99.87
				RF	99.89
				XGB	99.89
	Binary	Equal class distribution RUS		DT	99.82
				ET	99.91
				RF	99.91
				XGB	99.91

	Binary	Stratified RUS		DT	99.77
				ET	99.87
				RF	99.88
				XGB	99.88
[16]	Binary	Stratified RUS	mRMR	LDA	93.44
				DT	99.83
				RF	99.83
				XGB	99.76

The LycoS-Unicas-IDS2018 dataset, though promising, remains underutilized in network intrusion detection research, with Cantone et al. (2024) [16] being the only other study to leverage it for network intrusion detection. My research builds upon their efforts by incorporating both binary and multiclass classification, which extends the applicability of intrusion detection beyond single-attack scenarios. While Cantone et al. focused solely on binary classification, my study addresses a broader range of attacks, providing a more comprehensive evaluation of model performance in real-world settings.

A key methodological distinction lies in the feature selection process. Cantone et al. (2024) [16] employed Minimum Redundancy Maximum Relevance (mRMR), a well-established technique that selects features based on their relevance to the target variable while minimizing redundancy. In contrast, my study utilized Stacking Feature Embedded with PCA (SFE-PCA), which not only selects informative features but also enhances feature representation through dimensionality transformation. While mRMR focuses on reducing redundancy and selecting a compact subset, SFE-PCA leverages feature embedding to refine feature space representation, potentially leading to improved decision boundaries. This methodological difference may contribute to the slight performance improvement observed in my study.

Another notable improvement in my methodology is the use of 10-fold stratified k-fold cross-validation instead of the 80:20 train-test split employed by Cantone et al [16]. This ensures that each fold maintains the overall class distribution, reducing bias and producing more stable performance metrics. Stratified k-fold cross-validation provides a more reliable assessment of model generalization by mitigating the risks of biased training subsets. While Cantone et al. employed tree-based models as well, the improvement in my results can be attributed to the refined methodologies used, particularly the combination of SFE-PCA for feature selection and the implementation of stratified k-fold cross-validation.

While Cantone et al. conducted cross-dataset evaluation, which adds another layer of generalization assessment, my study instead emphasizes enhanced classification techniques and a more balanced feature selection approach. Rather than merely replicating their methodology, my work expands upon it by incorporating multiclass classification, and enhanced feature selection. These refinements make my study a valuable addition to network intrusion detection research, particularly in demonstrating the potential of the LycoS-Unicas-IDS2018 dataset beyond binary classification tasks.

6.5 Time complexity

Time complexity, which refers to the computational time required for an algorithm to execute, is a critical metric for evaluating the efficiency of machine learning models [72]. Efficient operation is especially vital in real-time environments in the context of IDS where prompt detection of network intrusions is crucial. In this study, I analyze the time complexity of the machine learning algorithms employed, namely DT, RF, ET, and XGB.

The time complexity of DT is generally $O(n * m.log(m))$, where 'n' represents the number of data points, and 'm' represents the number of features. DT constructs a hierarchical tree structure by recursively splitting data based on features. RF, which is an ensemble method comprising multiple DTs, has a time complexity of $O(t * n * m * log(m))$, where 't' represents the number of trees. Similarly, ET, another ensemble-based approach, shares the same time complexity as RF, $O(t \cdot n \cdot m \cdot log(m)$, as it builds multiple randomized DTs to enhance performance. On the other hand, the time complexity of XGB varies depending on its implementation, but it is typically $O(t \cdot d)$, where 't' is the number of trees, and 'd' is the depth of each tree.

Table 20 provides a detailed summary of the computational efficiency of these models, highlighting their respective time complexities. These insights are essential for selecting suitable algorithms based on the trade-offs between computational efficiency and predictive performance in IDS applications.

Table 20: Time complexity of ML models in IDS

SI. No.	ML model	Time complexity
1	DT (Decision Trees)	$O(n * m.log(m))$
2	RF (Random Forests)	$O(t * n * m * log(m))$
3	ET (Extra Trees)	$O(t * n * m * log(m))$
4	XGB (XGBoost)	$O(t \cdot d)$

7 Conclusion and Future Work

Conclusion

In conclusion, this research presents a robust framework for network intrusion detection, effectively addressing the challenges of imbalanced data distribution through a systematic approach. Unlike methods relying on Random Oversampling (RO), which risk overfitting due to data replication, this study employs Stratified Random Under-Sampling (RUS) to maintain the natural class distribution, ensuring that the results reliably reflect real-world conditions where benign traffic is significantly more prevalent than attacks. Additionally, this work integrates feature embedding through K-means and Gaussian Mixture (GM) clustering results while leveraging SEF-PCA for dimensionality reduction, enhancing both feature representation and model efficiency. The combination of tree-based models—Decision Tree (DT), Random Forest (RF), Extra Trees (ET), and XGBoost (XGB)—further strengthens detection capabilities, with ensemble methods improving generalization and boosting mechanisms refining predictive performance. The experimental results validate the effectiveness of this approach, achieving high accuracy and F1 scores across both binary and multiclass classification. By developing a methodology that ensures reliable performance on real-world, imbalanced datasets, this research provides valuable insights into optimizing machine learning-driven intrusion detection systems for practical deployment.

Future Work

Future research can explore the integration of federated learning to develop a decentralized network intrusion detection system capable of operating across multiple institutions without compromising data privacy. By training local models on separate datasets and aggregating their knowledge into a global model, this approach can improve generalization across diverse network environments while ensuring compliance with data protection regulations.

Another promising direction involves extending the proposed framework to real-time intrusion detection by integrating it with Software-Defined Networking (SDN) architectures. This would enable adaptive security responses, allowing network administrators to dynamically adjust security policies based on detected threats. Additionally, applying deep learning techniques, such as Transformer-based models, could further enhance the system's ability to detect complex attack patterns and previously unseen threats.

By pursuing these research directions, future work can contribute to the advancement of more adaptive, scalable, and intelligent intrusion detection systems capable of safeguarding modern network infrastructures against evolving cyber threats.

8 Reference

[1] S. Mueller, "Facing the 2020 Pandemic: What does Cyberbiosecurity want us to know to safeguard the future?," *Biosafety and Health*, September 2020.

[2] T. Marwala, Artificial Intelligence, Game Theory and Mechanism Design in Politics, Springer, 2023, p. 135–155.

[3] A. H. George, T. Bhaskar and A. S. George, "Digitally Immune Systems: Building Robust Defences in the Age of Cyber Threats," vol. 01, no. 04 | July-August 2023, August 2023.

[4] H. H. Nguyen, Y. Lim, M. Seo, Y. Jung, M. Kim and W. Park, "Strengthening Information Security Through Zero Trust Architecture: A Case Study in South Korea," in *Intelligent Systems and Data Science*, 2023, pp. 63-77.

[5] A. R. Khan, M. Kashif, R. H. Jhaveri, R. Raut, T. Saba and S. A. Bahaj, "Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions," *Security and Communication Networks*, vol. 2022, July 2022.

[6] M. A. Talukder, F. Hasan, M. Islam, M. A. Uddin, A. Akhter, M. Abu Yousuf, F. Alharbi and M. A. Moni, "A Dependable Hybrid Machine Learning Model for Network Intrusion Detection," December 2022.

[7] M. Schmitt, "Securing the Digital World: Protecting smart infrastructures and digital industries with Artificial Intelligence (AI)-enabled malware and intrusion detection," *Journal of Industrial Information Integration 36(1):100520*, December 2023.

[8] D. Preuveneers and W. Joosen, "Sharing Machine Learning Models as Indicators of Compromise for Cyber Threat Intelligence".

[9] P. Singh and P. Singh, "Artificial Intelligence: The Backbone of National Security in 21st Century," vol. 44, 2023.

[10] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm," *Journal of Information Security and Applications*, February 2019.

[11] N. Allahrakha, "Balancing Cyber-security and Privacy: Legal and Ethical Considerations in the Digital Age," 2023.

[12] Y. Hamid, S. Muthukumarasamy and B. Ranganathan, "IDS Using Machine Learning -Current State of Art and Future Directions," *British Journal of Applied Science & Technolog*, March 2016.

[13] S. M. Istiaque, A. I. Khan, Z. Al Hassan and S. Waheed, "Performance Evaluation of a Smart Intrusion Detection System (IDS)," *European Journal of Engineering and Technology Research 6(2):148-152*, February 2021.

[14] A. Cholakoska, M. Shushlevska, Z. Todorov and D. Efnusheva, "Analysis of Machine Learning Classification Techniques for Anomaly Detection with NSL-KDD Data Set," in *Data Science and Intelligent Systems*, Springer International Publishing, 2021, pp. 258-267.

[15] S. Narayanasami, S. Sengan, S. Khurram, F. Arslan, S. K. Murugaiyan, R. Rajan, V. Peroumal, A. K. Dubey, S. Srinivasan and D. K. Sharma, "Biological Feature Selection and Classification Techniques for Intrusion Detection on BAT," *Wireless Personal Communications*, 29 July 2021.

[16] M. Cantone, C. Marrocco and A. Bria, "On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection," 15 Febraury 2024.

[17] [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[18] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*.

[19] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole, "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018.

[20] [Online]. Available: https://www.unb.ca/cic/datasets/index.html.

[21] M. A. Albahar and . M. Binsawad, "Deep Autoencoders and Feedforward Networks Based on a New Regularization for Anomaly Detection," *Security Communication Networks*, 10 July 2020.

[22] A. H. Farooqi, S. Akhtar, H. Rahman, T. Sadiq and W. Abbass, "Enhancing Network Intrusion Detection Using an Ensemble Voting Classifier for Internet of Things," *Sensors*, 26 December 2023.

[23] [Online]. Available: https://github.com/MarcoCantone/LycoS-Unicas-IDS2018.

[24] A. Rosay , E. Cheval, F. Carlier and P. Leroux, "Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017," in *Proceedings of the 8th International Conference on Information Systems Security and Privacy ICISSP*, 2022.

[25] F. Cremer, B. Sheehan, M. Fortmann, A. N. Kia, M. Mullins, F. Murphy and S. Materne , in *Cyber risk and cybersecurity: a systematic review of data availability*, The Geneva Papers on risk and insurance-Issues and practice, 2022, p. 698–736.

[26] E. E. Abdallah, W. Eleisah and A. F. Otoom, "Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey.," in *The 13th International Conference on Ambient Systems, Networks and Technologies (ANT)*, 2022.

[27] Y. Hamid, S. Muthukumarasamy and B. Ranganathan, "IDS Using Machine Learning - Current State of Art and Future Directions," *Current Journal of Applied Science and Technology*, pp. 1-22, 2016.

[28] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," 2020.

[29] M. A. Talukder, M. Islam, M. A. Uddin and F. Hasan, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, February 2024.

[30] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," December 2019.

[31] Z. Tan, A. Jamdagni, X. He and P. Nanda, "A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis," February 2014.

[32] R. SakilaAnnarasi and S. Sivanesh, "A secure intrusion detection system for MANETs," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014.

[33] N. Allahrakha, "Balancing Cyber-security and Privacy: Legal and Ethical Considerations in the Digital Age," 2023.

[34] S. Narayanasami, S. Sengan, S. Khurram, F. Arslan, S. K. Murugaiyan, R. Rajan, V. Peroumal, A. K. Dubey, S. Srinivasan and D. K. Sharma, "Biological Feature Selection and Classification Techniques for Intrusion Detection on BAT," *Wireless Personal Communications 127(3)*, p. 1–23, July 2021.

[35] S. Norwahidayah, N. Suhana, N. N. Farahah, A. Amirah and N. Liyana, "Performances of Artificial Neural Network (ANN) and Particle Swarm Optimization (PSO) Using KDD Cup ‘99 Dataset in Intrusion Detection System (IDS)," *Journal of Physics Conference Series 1874(1):012061*, May 2021.

[36] B. S. Bhati and C. Rai, "Intrusion detection technique using Coarse Gaussian SVM," *International Journal of Grid and Utility Computing 12(1):27*, January 2021.

[37] M. Uma and P. Ganapathi, "A survey on various cyber attacks and their classification," *International Journal of Network Security 15(6)*, 2013.

[38] X. Li, J. D. Smith, T. N. Dinh and M. T. Thai, "Privacy Issues in Light of Reconnaissance Attacks with Incomplete Information," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016.

[39] A. Hussain, J. Heidemann and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," July 2003.

[40] K. . A. Forcht, E. Kieschnick, D. S. Thomas and J. D. Shorter, "IDENTITY THEFT: THE NEWEST DIGITAL ATTACK".

[41] A. Bhardwaj and S. Goundar, "Keyloggers: silent cyber security weapons," February 2020.

[42] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta and C. Q. Wu, "AVOIDIT: A Cyber Attack Taxonomy".

[43] S. Moualla, K. Khorzom and A. Jafar, "Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset," June 2021.

[44] S. M. Kasongo and Y. Sun , "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset".

[45] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Express 7(4)*, May 2021.

[46] V. Kumar, A. K. Das and D. Sinha, "Statistical Analysis of the UNSW-NB15 Dataset for Intrusion Detection," in *Computational Intelligence in Pattern Recognition*, 2020, pp. 279-294.

[47] M. Ahmad, Q. Riaz, M. Zeeshan, H. Tahir, S. A. Haider and M. . S. Khan, "Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set," *EURASIP Journal on Wireless Communications and Networking*, January 2021.

[48] D. Kshirsagar and S. Kumar, "An efficient feature reduction method for the detection of DoS attack," *ICT Express 7(3)*, January 2021.

[49] A. Aleesa, M. Y. Thanoun, N. M. Sahar and A. A. Mohammed, "DEEP-INTRUSION DETECTION SYSTEM WITH ENHANCED UNSW-NB15 DATASET BASED ON DEEP LEARNING TECHNIQUES," *Journal of Engineering Science and Technology*, vol. 16, February 2021.

[50] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," in *International Conference on Computational Intelligence and Data Science (ICCIDS 2019)*, 2020.

[51] J. Kim, J. Kim, H. Kim and M. Shim, "CNN-Based Network Intrusion Detection against Denial-of-Service Attacks," *Electronics*, 1 June June.

[52] S. AL and M. DENER, "STL-HDL: A new hybrid network intrusion detection system for imbalanced dataset on big data environment," *Computers & Security 110(2):102435*, August 2021.

[53] A. Bhardwaj, V. Mangat and V. Renu, "Hybrid Deep Neural Architecture for Detection of DDoS Attacks in Cloud Computing," in *Intelligent Systems, Technologies and Applications*, 2021, pp. 71-86.

[54] H. Zhang, L. Huang, C. Q. Wu and Z. Li, "An Effective Convolutional Neural Network Based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset," *Computer Networks 177(18):107315*, May 2020.

[55] M. M. Hassan, A. Gumaei, A. Alsanad and M. Alrubaian, "A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment," *Information Sciences 513*, November 2019.

[56] A. Roasy, E. Cheval, F. Carlier and P. Leroux, "Network intrusion detection: A comprehensive analysis of CIC-IDS2017," in *8th International Conference on Information Systems Security and Privacy*, 2022.

[57] H. Zou, T. Hastie and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, May 2004.

[58] B. Surendiran and K. K. Vasan , "Dimensionality reduction using principal component analysis for network intrusion detection," *Perspectives in Science 8(C)*, July 2016.

[59] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour and H. Janicke, "A Novel Hierarchical Intrusion Detection System Based on Decision Tree and Rules-Based Models," in *1st International Workshop on Security and Reliability of IoT Systems - SecRlot 2019*, Greece, 2019.

[60] M. A. Uddin, M. Islam, M. A. Talukder, M. Al, A. Hossain, A. Akhter, S. Aryal and M. Muntaha, "Machine Learning Based Diabetes Detection Model for False Negative Reduction," June 2023.

[61] L. Breiman, "Random Forests," *Machine Learning*, October 2001.

[62] N. Uddin, M. K. U. Ahamed, M. A. Uddin, M. Islam and M. A. Talukder, "An Ensemble Machine Learning Based Bank Loan Approval Predictions System with a Smart Application," *International Journal of Cognitive Computing in Engineering*, January 2023.

[63] P. Geurts, D. Ernst and L. Wehenkel, "Extremely Randomized Trees," *Machine Learning*, April 2006.

[64] T. Chen and T. He, "Higgs Boson Discovery with Boosted Trees," *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, vol. PMLR 42, pp. 69-80 , 2015.

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

[66] M. A. Talukder, M. Islam, M. A. Uddin, A. Akhter, M. A. J. Pramanik, S. Aryal, M. A. Almoyad, K. F. Hasan and M. A. Moni , "An efficient deep learning model to categorize brain tumor using reconstruction and fine-tuning," *Expert Systems with Applications*, May 2023.

[67] A. Akhter, U. K. Acharjee, M. A. Talukder, M. Islam and M. A. Uddin, "A robust hybrid machine learning model for Bengali cyber bullying detection in social media," *Natural Language Processing Journal*, July 2023.

[68] G. Sameera, R. V. Vardhan and K. V. Sarma, "Binary classification using multivariate receiver operating characteristic curve for continuous data," *Journal of Biopharmaceutical Statistics*, 26 May 2015.

[69] I. A. Vergara, T. Norambuena, E. Ferrada, A. W. Slater and F. Melo, "StAR: A simple tool for the statistical comparison of ROC curves," *BMC Bioinformatics*, February 2008.

[70] F. Gorunescu, Data Mining: Concepts, models and techniques, vol. 12, Springer-Verlag Berlin Heidelberg, Intelligent Systems References Library, 2011.

[71] A. Yulianto, P. Sukarno and N. Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset," *Journal of Physics Conference Series,* March 2019.

[72] M. A. Talukdera, M. M. Islama, M. A. Uddina, A. Akhtera, K. F. Hasanb and M. A. Mon, "Machine learning-based lung and colon cancer detection using deep feature extraction and ensemble learning," *Expert Systems with Applications,* June 2022.

[73] S. Seth, G. Singh and K. K. Chahal, "A novel time efficient learning-based approach for smart intrusion detection system," *Journal of Big Data,* August 2021.

[74] M. A. Khan, "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System," *Processes,* May 2021.