

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research



University of Mila
Faculty of Sciences and Technology
Department of Computer Science

Specialty: Artificial Intelligence and its Applications
Master's Thesis

**Analysis and Evaluation of Dimensionality
Reduction Techniques in the Context
of Network Intrusion Detection System**

Presented by:

> **Remmache Yaakoub**
> **Lecheheb Nabil**

Supported by the jury:

President: Mr. Selmane Samir

Rank: M.A.A

Examiner: Mr. Alliouche Abdelaziz

Rank: M.A.A

Supervisor: Ms. Zekiouk Mounira

Rank: M.A.A

Co-supervisor: Mr. Bencheikh Lehocine Madjed

Rank: M.C.A

Academic Year: 2025 – 2026

إِهْدَاء

إلى الذين لم يقرؤوا السطور، لكنهم شعروا بثقل الرحلة في عيني...
إلى أمي وأبي، اللذين كانا يدفعان بالدعاء خلف كل خطوة متعثرة،
ليتحول القلق إلى يقين.

إلى تلك الأيدي الخفية التي رتبت فوضى الأيام خلف الكواليس
لكي أصل إلى هذه اللحظة،
وإلى نفسي القديمة التي وقفت حائرة في البداية، وأهدتني هذا الإنجاز اليوم.

يعقوب - نبيل

Abstract

The rapid expansion of computer networks has increased the frequency and sophistication of cyberattacks, making Intrusion Detection Systems (IDS) a critical line of defense for protecting network infrastructures. Although Machine Learning (ML) has significantly improved the detection capabilities of IDS, modern network traffic often produces massive and high-dimensional datasets. This high dimensionality, commonly referred to as the curse of dimensionality, can lead to high computational complexity, increased model complexity, and overfitting, which may reduce the ability of ML models to accurately detect cyber threats.

This study addresses this issue by comparing several feature selection and feature reduction techniques in the context of ML-based IDS. The adopted techniques include ANOVA and Mutual Information as filter-based methods, Random Forest and Lasso L1 regularization as embedded methods, in addition to two hybrid techniques designed to leverage the advantages of simpler feature selection approaches, and PCA. The experimental evaluation is conducted using two well-known IDS datasets, namely CIC-IDS2017 and CSE-CIC-IDS2018.

All experiments are based on the XGBoost algorithm, which is widely recognized for its strong predictive performance and effectiveness in classification tasks. The obtained results for both binary and multiclass classification show promising performance and provide valuable insights into the role of dimensionality reduction in enhancing the effectiveness, efficiency, and generalization ability of ML-based IDS.

Keywords: Network Intrusion Detection, Intrusion Detection System, Machine Learning, Dimensionality Reduction, Feature Selection, Feature Reduction, CIC-IDS2017, CSE-CIC-IDS2018.

Résumé

L'expansion rapide des réseaux informatiques a entraîné une augmentation de la fréquence et de la sophistication des cyberattaques, faisant des systèmes de détection d'intrusions (Intrusion Detection Systems, IDS) une ligne de défense essentielle pour la protection des infrastructures réseau. Bien que l'apprentissage automatique (ML) ait considérablement amélioré les capacités de détection des IDS, le trafic réseau moderne génère souvent des ensembles de données massifs et de grande dimension. Cette forte dimensionnalité, communément appelée malédiction de la dimensionnalité, peut engendrer une complexité computationnelle élevée, une augmentation de la complexité des modèles ainsi qu'un risque de surapprentissage, ce qui peut réduire la capacité des modèles de ML à détecter avec précision les cybermenaces. Cette étude aborde cette problématique en comparant plusieurs techniques de sélection et de réduction de caractéristiques dans le contexte des IDS basés sur le ML. Les techniques adoptées comprennent ANOVA et l'information mutuelle en tant que méthodes de type filtre, Random Forest et la régularisation Lasso L1 en tant que méthodes intégrées, ainsi que deux approches hybrides conçues pour tirer parti des avantages des méthodes simples de sélection de caractéristiques, en plus de l'analyse en composantes principales (PCA). L'évaluation expérimentale est menée à l'aide de deux ensembles de données IDS bien connus, à savoir CIC-IDS2017 et CSE-CIC-IDS2018. Toutes les expérimentations reposent sur l'algorithme XGBoost, largement reconnu pour ses fortes performances prédictives et son efficacité dans les tâches de classification. Les résultats obtenus, aussi bien pour la classification binaire que multiclasse, montrent des performances prometteuses et fournissent des indications précieuses sur le rôle de la réduction de dimensionnalité dans l'amélioration de l'efficacité, de l'efficacité et de la capacité de généralisation des IDS basés sur le ML.

Mots-clés :Détection d'intrusions réseau, système de détection d'intrusions, apprentissage automatique, réduction de dimensionnalité, sélection de caractéristiques, réduction de caractéristiques, CIC-IDS2017, CSE-CIC-IDS2018

ملخص

أدى التوسع السريع في شبكات الحاسوب إلى زيادة وتيرة الهجمات السيبرانية وتعقيدها، مما جعل أنظمة كشف التسلل (IDS) خط الدفاع الأساسي لحماية البنى التحتية للشبكات. وعلى الرغم من أن تقنيات التعلم الآلي (ML) قد حسّنت بشكل كبير قدرات أنظمة كشف التسلل، فإن حركة مرور الشبكات الحديثة غالباً ما تُنتج مجموعات بيانات ضخمة وعالية الأبعاد. ويؤدي هذا الارتفاع في عدد الأبعاد، المعروف باسم "لعنة الأبعاد" (Curse of Dimensionality)، إلى زيادة التعقيد الحسابي وتعقيد النماذج، بالإضافة إلى الإفراط في التعلّم (Overfitting)، مما قد يحدّ من قدرة نماذج التعلم الآلي على اكتشاف التهديدات السيبرانية بدقة.

تعالج هذه الدراسة هذه المشكلة من خلال مقارنة عدد من تقنيات اختيار الميزات وتقليل الأبعاد في سياق أنظمة كشف التسلل المعتمدة على التعلم الآلي. وتشمل التقنيات المعتمدة تحليل التباين (ANOVA) والمعلومات المتبادلة (Mutual Information) بوصفهما من أساليب الترشيح (Filter-Based Methods)، والغابة العشوائية (Random Forest) وتنظيم لاسو (Lasso L1 Regularization) بوصفهما من الأساليب المدمجة (Embedded Methods)، بالإضافة إلى تقنيتين هجنتين (Hybrid Techniques) صُممتا للاستفادة من مزايا أساليب اختيار الميزات الأبسط، إلى جانب تحليل المكونات الرئيسية (PCA). وقد أُجري التقييم التجريبي باستخدام مجموعتي بيانات معروفتين في مجال كشف التسلل، وهما CIC-IDS2017 و CSE-CIC-IDS2018.

استندت جميع التجارب إلى خوارزمية XGBoost، التي تُعرف على نطاق واسع بأدائها التنبؤي القوي وفعاليتها في مهام التصنيف. وأظهرت النتائج المتحصل عليها في كلّ من التصنيف الثنائي (Binary Classification) والتصنيف متعدد الفئات (Multiclass Classification) أداءً واعدًا، كما قدمت رؤية قيّمة حول دور تقليل الأبعاد في تعزيز فعالية وكفاءة وقدرة أنظمة كشف التسلل المعتمدة على التعلم الآلي على التعميم.

الكلمات المفتاحية: كشف التسلل إلى الشبكات، نظام كشف التسلل، التعلم الآلي، تقليل الأبعاد، اختيار الميزات، اختزال الميزات، CIC-IDS2017، CSE-CIC-IDS2018.

Contents

Abstract	2
Résumé	4
List of Figures	5
List of Tables	7
List of Abbreviations	9
General Introduction	1
1 Intrusion Detection System and ML: Foundations and Challenges	3
1.1 Introduction	4
1.2 Fundamentals of IDSs	4
1.2.1 Definition and Role of IDSs	4
1.2.2 Common Network Security Threats	5
1.3 IDSs Architecture	6
1.4 Taxonomies of IDSs	7
1.4.1 Deployment-based classification	8
1.4.2 Classification by Detection Method	9
1.5 Performance Challenges in IDS	10
1.6 Emerging Technologies for IDS	11
1.7 ML in the Context of Cybersecurity	12
1.7.1 Fundamental Concepts of ML	12
1.7.2 Toward ML-based IDS	14

1.7.3	Industry and Research Adoption of ML-Based IDS	15
1.8	Challenges of ML-Based IDSs	16
1.9	Conclusion	18
2	Network Intrusion Detection Datasets: Characteristics and Dimensionality Challenges	19
2.1	Introduction	20
2.2	Overview of the Main IDS-Related Datasets	20
2.2.1	Evolution of IDS Datasets	20
2.2.2	The KDD Cup 99 Dataset	22
2.2.3	The NSL-KDD Dataset	23
2.2.4	The UNSW-NB15 Dataset	25
2.2.5	The CIC-IDS2017 Dataset	27
2.2.6	The CSE-CIC-IDS2018 Dataset	29
2.3	Comparative Study of Benchmarking Datasets	30
2.4	High-Dimensionality Problem	31
2.4.1	Definition	32
2.4.2	Impact of the High-Dimensionality Problem	32
2.4.3	Dimensionality Reduction Techniques	33
2.5	Key Feature Selection (FS) Methods	33
2.5.1	Filter Methods	34
2.5.2	Wrapper Methods	36
2.5.3	Embedded Methods	36
2.5.4	Hybrid Methods	38
2.6	Key Feature Reduction (FR) Methods	38
2.6.1	Principal Component Analysis (PCA)	39
2.6.2	Linear Discriminant Analysis (LDA)	39
2.7	Conclusion	40
3	Experimental Evaluation of Dimensionality Reduction Techniques	41

3.1	Introduction	42
3.2	Global methodology	42
3.2.1	Datasets Cleaning and Harmonisation	43
3.2.2	Data Splitting	45
3.2.3	Class Imbalance Handling	46
3.2.4	Dimensionality Reduction	48
3.2.5	Data Encoding	49
3.2.6	Features Normalization	49
3.3	Model Selection and Evaluation Strategy	50
3.3.1	Selection of the Learning Model	50
3.3.2	Evaluation Strategy	51
3.3.3	Evaluation Metrics	52
3.4	Tools and Software	53
3.5	Results	53
3.5.1	Multiclass Classification Results	54
3.5.2	Binary Classification Results	57
3.6	Comparative Discussion	61
3.7	Multi-Classification Comparison	61
3.8	Binary Classification Comparison	64
3.9	Conclusion	66
	General Conclusion	67

List of Figures

1.1	IDS architecture.	7
1.2	Taxonomy of IDS based on deployment and Detection Method.	8
1.3	HIDS vs NIDS deployment.	9
1.4	Signature-based vs. Anomaly-based Detection.	10
1.5	Supervised Learning.	13
1.6	Unsupervised learning.	13
1.7	Semi-Supervised Learning.	14
1.8	Reinforcement ML Process.	14
2.1	IDS datasets evolution.	21
2.2	Number of instances by traffic types in the KDD Cup 99 dataset.	23
2.3	Number of Instances by Traffic Types in the NSL-KDD Dataset.	25
2.4	Number of Instances by Traffic Types in the UNSW-NB15 Dataset.	27
2.5	Number of Instances by Traffic Types in the CIC-IDS2017 Dataset.	28
2.6	Number of Instances by Traffic Types in the CIC-IDS2018 Dataset.	29
2.7	Feature Selection Categories.	34
3.1	Global Methodology.	43
3.2	Class Balancing for Multiclass Classification.	47
3.3	Class balancing for binary classification.	47
3.4	Multiclass Classification Confusion Matrices For the Baseline Model.	55
3.5	Performance Curve of XGBoost with Different Numbers of ANOVA Selected Features	55
3.6	Confusion Matrices for ANOVA (55 Features)	56

3.7	Performance Curve of Mutual Information.	58
3.8	confusion matrix and KDE plot for DoS-Slowloris using MI (Threshold = 0.3)	59
3.9	Performance Curve of Random Forest	60
3.10	Confusion Matrix and KDE Probability Distribution for Dos hulk using RF (Threshold = 0.3)	61
3.11	Recall Comparison Across FS Methods – CICIDS2017	62
3.12	Multiclass Classification Comparison of Accuracy and Macro Recall Across Different Feature Selection Tehniques	62
3.13	Per-Class Recall Comparison Across Feature Selection Techniques for Mul- ticlass Classification-CICIDS2018	63
3.14	Comparison of Global Recall Across Features selection techniques.	64
3.15	Recall Improvement of Weakly Detected Attacks Using Mutual Information	65

List of Tables

2.1	KDD Cup 99 Traffic Distribution.	22
2.2	NSL-KDD Dataset Traffic Distribution.	24
2.3	UNSW-NB15 Dataset Traffic Distribution.	26
2.4	CIC-IDS2017 Dataset Traffic Distribution.	28
2.5	CIC-IDS2018 Dataset Traffic Types.	30
2.6	Comprehensive Comparison of Prominent IDS Datasets	31
2.7	Comparison of Filter, Wrapper, and Embedded Feature Selection Paradigms	38
3.1	Attack Categories in CIC-IDS2017 and CIC-IDS2018 Datasets.	45
3.2	Dataset splitting Ratios.	46
3.3	Parameter Configuration of the Feature Selection Methods	48
3.4	Label Encoding for Binary and Multiclass Classification	49
3.5	XGBoost Hyperparameter Configuration for Binary and Multiclass Classifi- cation	51
3.6	Performance Report : (Baseline Model - All Features).	54
3.7	Performance Report : ANOVA (55 Features)	56
3.8	Binary Classification Performance Report-CIC-IDS2017 (Baseline Model) .	57
3.9	Binary Classification Performance Report on CSE-CIC-IDS2018 (Baseline Model)	58
3.10	Binary Classification Performance Report on CSE-CIC-IDS2018 (Mutual Information/ 10 features)	59
3.11	Binary Classification Performance Report on CSE-CIC-IDS2018 (Random Forest/ 40 feature)	60

3.12 Pairwise Feature Intersection Matrix across Feature Selection Techniques. .	64
--	----

List of Abbreviations

AI	Artificial Intelligence
ANOVA	Analysis of Variance
CIC	Canadian Institute for Cybersecurity
CM	Confusion Matrix
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Tree
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
KDE	Kernel Density Estimation
LASSO	Least Absolute Shrinkage and Selection Operator
MI	Mutual Information
ML	Machine Learning
NIDS	Network Intrusion Detection System
PCA	Principal Component Analysis
PCC	Pearson Correlation Coefficient
RF	Random Forest
RFE	Recursive Feature Elimination
SMOTE	Synthetic Minority Over-sampling Technique
SQL	Structured Query Language
XSS	Cross-Site Scripting

General Introduction

In recent decades, digital technologies have developed rapidly, mainly due to the growth of computer networks, Internet-connected devices, cloud computing, and the Internet of Things (IoT). These technologies have improved communication, data sharing, and operational efficiency. However, they have also increased the number of possible entry points for cyberattacks. As a result, cyber threats have become more advanced, dynamic, and automated, creating serious risks for sensitive information and critical infrastructures.

Within this evolving threat landscape, IDS have emerged as an indispensable line of defense. These systems are purposefully designed to continuously monitor and analyze network traffic in real-time, identifying anomalous or malicious activities. However, traditional IDS solutions are increasingly proving inadequate against modern, encrypted, and zero-day attacks, primarily due to their reliance on pre-defined rules. To overcome these limitations, the cybersecurity community has pivoted towards Artificial Intelligence and ML, leveraging their capacity to autonomously extract complex behavioral patterns from historical data.

Nevertheless, deploying ML models for real-time intrusion detection introduces a critical architectural challenge: the **Curse of Dimensionality**. Modern network traffic capture tools extract an immense number of features (attributes) per communication flow to ensure no subtle attack pattern is missed. Feeding this massive, high-dimensional feature space into a ML algorithm drastically increases computational overhead and prolongs training and inference times, which is highly detrimental for real-time detection systems. More critically, high-dimensional data often contains redundant, highly correlated, and noisy features, which can obscure the true underlying attack signatures. This may cause ML models to suffer from severe overfitting, high memory consumption, and a significant degradation in their ability to generalize to novel and unseen network environments.

To mitigate this problem, dimensionality reduction techniques are commonly employed as an essential preprocessing step in ML-based IDSs. These techniques aim to reduce the number of input features while preserving the most relevant and discriminative information

required for accurate attack detection.

In this work, we aim to analyze and compare several dimensionality reduction techniques for ML-based IDS in order to identify the most suitable approaches for reducing data dimensionality while preserving detection performance and model reliability.

This document is structured into three main chapters, organized as follows:

- **Chapter 1: Network Intrusion Detection and ML: Foundations and Challenges.**

This chapter provides a theoretical overview of IDS architectures and detection mechanisms. It also introduces the fundamental concepts of ML and highlights the transition towards ML-based IDS.

- **Chapter 2: Network Intrusion Detection Datasets: Characteristics and Dimensionality Challenges.** This chapter presents an overview of the main benchmark datasets used in network intrusion detection, with particular focus on CIC-IDS2017 and CIC-IDS2018. It highlights their key characteristics, structure, attack categories, and relevance for ML-based IDS evaluation. In addition, the chapter discusses the dimensionality problem and explains the main techniques used to address it, including feature selection and dimensionality reduction methods.

- **Chapter 3: Experimental Evaluation of Dimensionality Reduction Techniques.** This chapter presents the practical implementation of the proposed work. It describes the overall methodology, the integration of dimensionality reduction techniques, and the experimental setup used to train and evaluate the selected models. In addition, it provides a comparative analysis of the obtained results under both internal evaluation and external validation strategies.

Chapter 1

Intrusion Detection System and ML: Foundations and Challenges

1.1 Introduction

The rapid growth of digital communication channels has increased the risks of unauthorized access and data manipulation by malicious actors. As these threats continue to evolve, securing digital infrastructure has become a critical necessity. Therefore, the adoption of advanced protection technologies is essential to strengthen security and protect systems against potential cyberattacks.

IDSs stand as a cornerstone of modern cybersecurity. By maintaining continuous surveillance over network traffic, they ensure the early detection of anomalies, enabling swift responses to prevent significant security breaches.

This chapter presents the fundamental concepts of IDSs, including their definition, objectives, common cybersecurity threats, taxonomy, and architecture. It also discusses the major challenges that affect IDS performance and effectiveness. Furthermore, the chapter provides an overview of ML in the context of cybersecurity and highlights its role in enhancing intrusion detection through intelligent and adaptive threat detection mechanisms.

1.2 Fundamentals of IDSs

1.2.1 Definition and Role of IDSs

An IDS is a security tool that monitors network or system activities to detect suspicious behavior or possible attacks. Its main purpose is to identify intrusions and generate alerts for security administrators.

IDSs are important because they provide an additional layer of defense inside a security architecture. They help organizations detect attacks that may bypass traditional protection mechanisms such as firewalls. By analyzing network traffic or system behavior, an IDS supports early threat detection and helps security teams respond more quickly to potential incidents.[1]

1.2.2 Common Network Security Threats

To design effective defense mechanisms, it is essential to understand the main network security threats. Modern networks are continuously exposed to different attack vectors that can compromise data, services, and system availability. This section presents the most common threats relevant to intrusion detection :

- **Denial of Service (DOS) and Distributed Denial of Service (DDoS):** A DOS attack aims to paralyze a target server or network by overwhelming it with a massive influx of illegitimate traffic, thereby exhausting its computational resources and bandwidth. A DDoS attack represents a more severe evolution of this concept, where the adversary orchestrates the assault using a geographically dispersed compromised devices. This distributed nature not only amplifies the attack's magnitude but also severely complicates mitigation and source traceback efforts. [2].
- **Man-in-the-Middle (MitM) Attacks:** In this sophisticated interception technique, an attacker covertly positions themselves between two communicating nodes, breaching the communication channel without either party's awareness. This strategic placement allows the adversary to seamlessly eavesdrop on confidential exchanges, manipulate data packets in transit, or perform session hijacking, severely compromising data integrity and confidentiality [3].
- **Malware Infections:** Malware represents a broad spectrum of malicious software-encompassing viruses, worms, Trojans, and deployed onto a host system. The objectives behind malware deployment range from covert surveillance and sensitive data exfiltration to widespread operational disruption and extortion. These payloads typically infiltrate networks through compromised email attachments, malicious URLs, or unpatched software vulnerabilities [4].
- **Social Engineering and Phishing:** Unlike direct technical exploits, phishing relies on the psychological manipulation of human users. Attackers masquerade as trustworthy entities across digital communication channels (predominantly email) to deceive

individuals into divulging sensitive credentials or inadvertently executing malicious payloads. Advanced variants, such as spear-phishing, utilize highly personalized contexts to target specific organizational personnel [5].

- **Database Injection Attacks:** These attacks exploit critical flaws in input validation mechanisms within web applications and backend databases. By injecting malicious SQL queries into standard user input fields, adversaries can manipulate the database execution logic. This enables unauthorized actors to bypass authentication, expose sensitive records, or permanently alter database contents [6].
- **Authentication and Password Attacks:** This category involves systematic attempts to compromise user credentials to breach security perimeters. Attackers employ various aggressive methodologies, including automated brute-force scripts, dictionary attacks, and credential stuffing (exploiting passwords reused across multiple platforms). These tactics heavily leverage human tendencies to use weak or predictable authentication phrases [7].
- **Reconnaissance and Probing:** Before launching a full-scale assault, attackers often engage in preliminary data-gathering activities. By systematically scanning network topologies, attackers can map out active hosts, identify open ports, and pinpoint exploitable vulnerabilities, effectively laying the groundwork for subsequent intrusions.

1.3 IDSs Architecture

An effective IDS relies on a modular and coordinated architecture. A standard IDS generally comprises four essential functional components, as illustrated in the high-level structural diagram in Figure 1.1 [8]:

- **Sensors (Data Collection):** These are the primary data acquisition units. They continuously monitor the target environment to capture raw data, such as network packets or operating system logs, and forward this information to the detection engine.

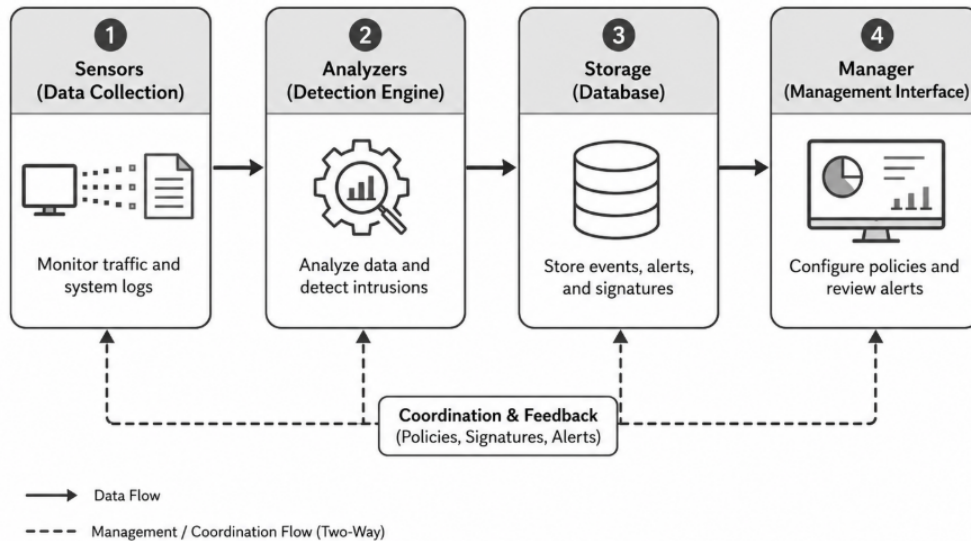


Figure 1.1: IDS architecture.

- **Analyzers (Detection Engine):** Serving as the core analytical brain of the system, analyzers process the raw data collected by sensors. They apply specific detection methodologies to evaluate the data and identify potential malicious activities.
- **Storage (Database):** This component is responsible for archiving historical data, including intercepted events, generated alerts, and known attack signatures. This stored data is critical for post-incident analysis and refining future detection systems.
- **Manager (Management Interface):** The management console acts as the centralized interface for human operators. It is responsible for configuring system policies, displaying real-time alerts, and coordinating automated or manual response protocols.

1.4 Taxonomies of IDSs

IDSs can be classified according to various taxonomies, each reflecting a different aspect of their design, functionality, or operational characteristics. Several classification criteria have been proposed in the literature, including detection methodology, deployment strategy, data source, response mechanism, and analysis approach. Among these, the most widely adopted and relevant classifications are based on the detection method employed and the deployment location within the monitored infrastructure, as comprehensively illustrated in

Figure 1.2 [9]:

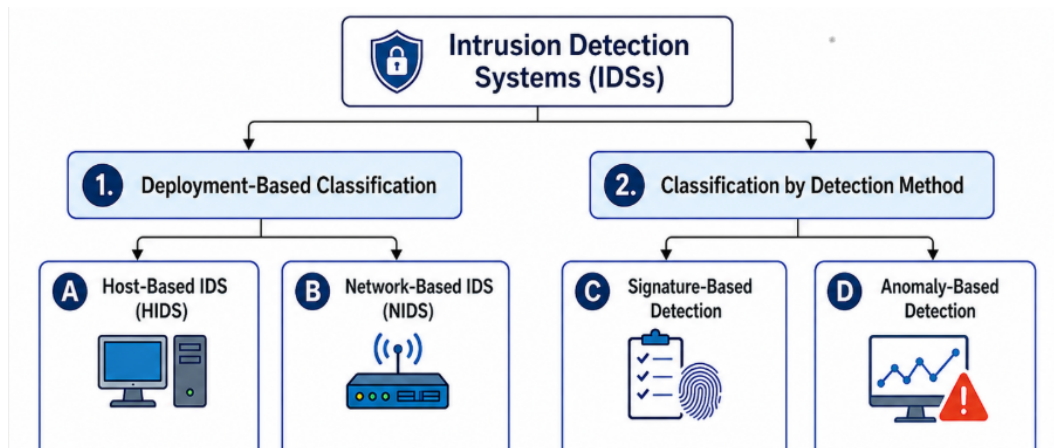


Figure 1.2: Taxonomy of IDS based on deployment and Detection Method.

1.4.1 Deployment-based classification

Deployment-based classification categorizes IDSs according to their placement within the monitored environment. Based on this criterion, IDSs are primarily classified as Host-Based IDS (HIDS) or Network-Based IDS (NIDS) [9]:

- **Host-based IDS (HIDS):** These systems are deployed on individual endpoints, such as critical servers or user workstations. They scrutinize localized system activities, including operating system audit trails, file integrity changes, and application logs. HIDS is highly effective at detecting insider threats or attacks that have successfully penetrated the network perimeter. Some popular HIDS tools include OSSEC, Wazuh, Tripwire, AIDE, Samhain, Fail2ban, and Solar Winds Security Event Manager [9].
- **Network-based IDS (NIDS):** In contrast to HIDS, NIDS operate at the network level. They are strategically positioned at infrastructure choke points, such as switches, routers, or network gateways, to monitor inbound and outbound traffic across an entire network segment. Some popular NIDS tools include Snort, Suricata, Zeek, Bro, Security Onion, and Cisco Secure IDS [9].

Figure 1.3 illustrates the main differences between Host-based IDS and Network-based IDS in terms of deployment location and monitoring visibility.

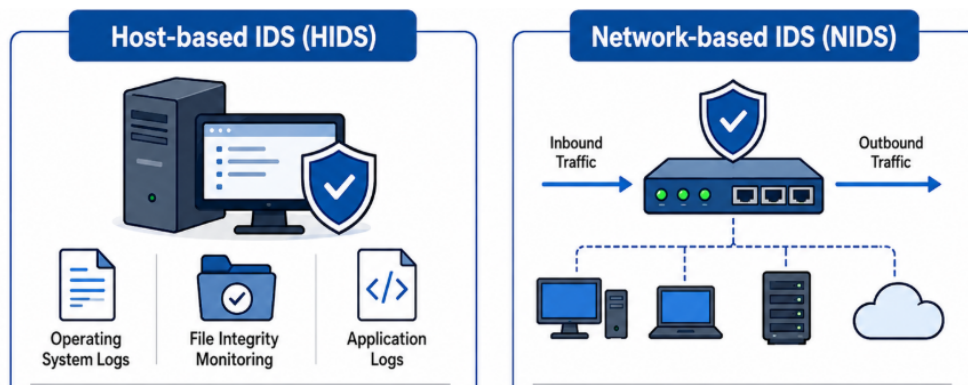


Figure 1.3: HIDS vs NIDS deployment.

1.4.2 Classification by Detection Method

The core mechanism by which an IDS identifies malicious activity defines its detection paradigm. The two foundational approaches are:

- **Signature-based Detection:** This approach relies on a predefined database of known attacks signatures specific patterns. When observed traffic matches a stored signature, an alert is triggered. While highly accurate and computationally efficient for recognized threats, this method is fundamentally flawed against "Zero-day" vulnerabilities and novel attack variations, as it cannot detect what it does not explicitly know.[10]
- **Anomaly-based Detection:** To overcome the limitations of signature-based systems, the anomaly-based approach establishes a statistical baseline of "normal" network or system behavior over time. Any significant deviation from this established profile is classified as a potential intrusion. This paradigm is exceptionally capable of detecting zero-day attacks. However, building an accurate baseline in dynamic network environments is complex, often resulting in higher False Positive Rates (FPR).[10]

Figure 1.4 compares signature-based and anomaly-based detection according to their detection process and alert generation mechanism.

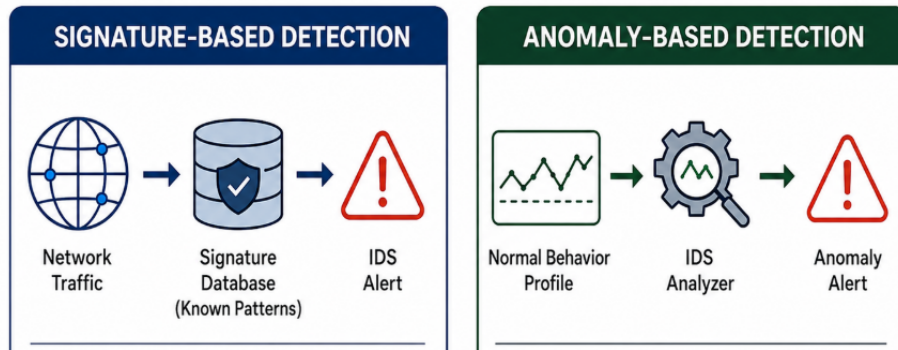


Figure 1.4: Signature-based vs. Anomaly-based Detection.

1.5 Performance Challenges in IDS

Although IDS play a crucial role in identifying malicious activities and protecting computer networks, their performance is still affected by several challenges. These limitations can reduce the accuracy, efficiency, and reliability of IDS. The main performance challenges that affect IDS can be summarized as follows:

- **False Positive (False Alert):** Legitimate activities may be incorrectly classified as malicious, generating unnecessary alerts and reducing analysts' confidence in the IDS. [11]
- **False Negative (False Detection):** Real attacks may remain undetected, allowing malicious activities to continue inside the system without being noticed. [11]
- **Real-time detection:** IDS must analyze traffic or host activities quickly, but real-time detection becomes difficult when traffic volume is high or detection methods are complex.
- **Resource consumption:** IDS may require significant CPU, memory, storage, and bandwidth resources to continuously monitor and analyze system or network activities. This can affect overall system performance and may limit their deployment in resource-constrained environments.[12]
- **Encrypted traffic:** Encryption prevents IDS from fully inspecting packet contents, which limits visibility and may reduce detection accuracy.[12]

- **Adaptability to evolving attacks:** Since attack techniques constantly change, IDS models must be regularly updated, retrained, or adapted to new malicious behaviors.[13]

1.6 Emerging Technologies for IDS

With the continuous evolution of cyber threats and the increasing complexity of modern network environments, traditional IDS face important limitations in terms of scalability, flexibility, and detection accuracy. As a result, several emerging technologies are being integrated into IDS to enhance their performance and adaptability. The most important technologies include:

- **Big Data Analytics:** Big Data Analytics helps IDS process and analyze massive volumes of network traffic, system logs, and security events. It improves the ability of IDS to detect hidden patterns, abnormal behaviors, and large-scale attacks in complex environments. [14]
- **Edge Computing:** Edge computing brings intrusion detection closer to the data source, such as IoT devices, sensors, or local network gateways. This reduces latency, supports faster response, and limits the need to send all data to a centralized server. [15]
- **Blockchain:** Blockchain can support IDS by enabling secure, decentralized, and tamper-resistant sharing of alerts, logs, and threat information. This is useful in collaborative environments where different IDS components need to exchange trusted security data. [16]
- **ML :** ML has significantly enhanced IDSs by enabling them to learn from historical data and automatically distinguish between normal and malicious network traffic patterns. Given its ability to provide adaptive and data-driven detection mechanisms, ML-based IDSs constitute the primary focus of this study, where we investigate their performance and effectiveness in addressing modern cybersecurity threats. [17]

1.7 ML in the Context of Cybersecurity

ML has become a fundamental approach in modern cybersecurity, particularly for improving the performance of IDS . By leveraging data-driven models, ML enables the automatic detection of malicious activities, the identification of abnormal network behavior, and the support of automated security responses. Unlike traditional rule-based methods, ML techniques learn directly from data by capturing complex patterns and relationships, allowing systems to make intelligent decisions without being explicitly programmed. [17]

In this section, we first present the fundamental ML, including its main paradigms: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. We then analyze the evolution toward ML-based IDS, the increasing adoption in both industrial and research domains, and the key challenges that may limit their practical deployment.

1.7.1 Fundamental Concepts of ML

ML is a branch of AI that studies the design of algorithms capable of learning from data and making data-driven predictions or decisions without explicit programming. These algorithms identify hidden patterns and relationships within data, allowing systems to generalize their knowledge to unseen instances and continuously improve their performance as more data becomes available [18]. ML approaches can be divided into four main categories based on how they learn from data:

- **Supervised Learning:** Supervised learning is a ML approach in which an algorithm learns the mapping between input data and corresponding output labels. It is trained using a labeled dataset, where each instance is composed of input features associated with a known target output. The dataset is typically divided into two subsets: a training set, which is used to learn the model parameters, and a test set, which is used to evaluate the model's performance on unseen data. [19]

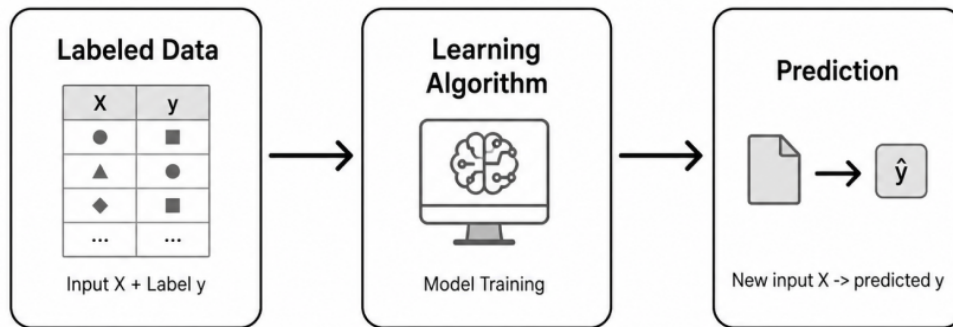


Figure 1.5: Supervised Learning.

- Unsupervised Learning:** Unsupervised learning is the second major category of ML. Unlike supervised learning, it operates on unlabeled datasets, where no predefined output labels are provided and no explicit supervision guides the learning process. The main objective of unsupervised learning algorithms is to discover hidden structures, patterns, or intrinsic relationships within the data. Once trained, these algorithms can analyze new incoming data and assign it to meaningful groups or representations based on the learned structure. [19]

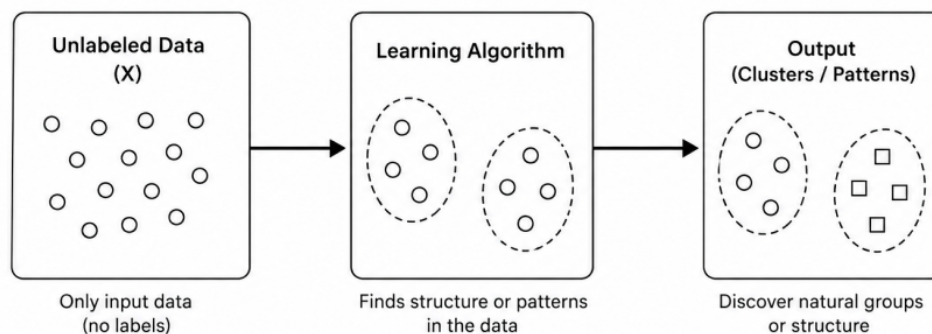


Figure 1.6: Unsupervised learning.

- Semi-Supervised Learning:** Semi-supervised learning is a hybrid approach that combines both supervised and unsupervised learning techniques. In this setting, the model is trained using a dataset that contains a small portion of labeled data along with a large amount of unlabeled data. The inclusion of unlabeled samples helps the model better capture the underlying structure of the data and improves its generalization performance, especially when labeled data is scarce or expensive to obtain. [20]

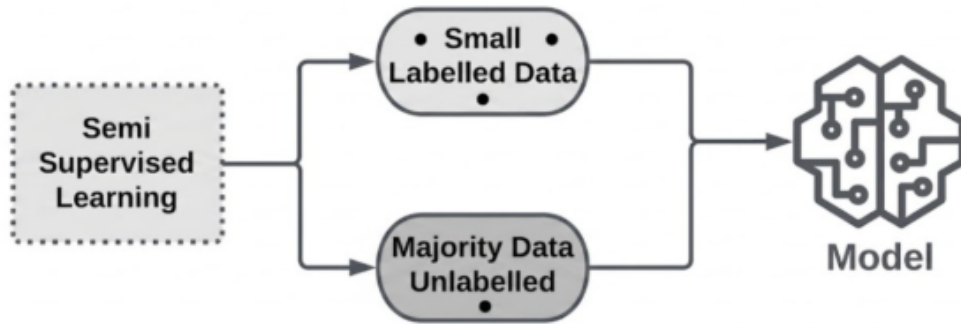


Figure 1.7: Semi-Supervised Learning.

- **Reinforcement Learning (RL):** This approach is based on continuous interaction with an environment. An AI "agent" learns to make decisions through trial and error, receiving rewards for correct actions and penalties for mistakes. [20]

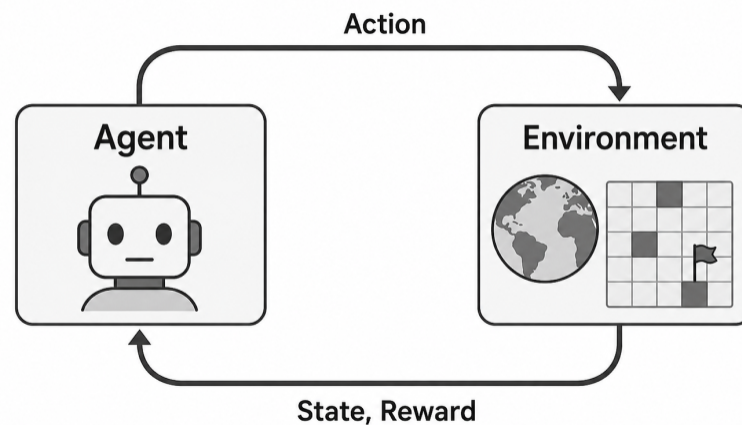


Figure 1.8: Reinforcement ML Process.

1.7.2 Toward ML-based IDS

The evolution of IDSs has gradually moved from traditional rule-based and signature-based approaches toward ML-based solutions. Early IDSs concepts were introduced in the 1980s, particularly with the work of Denning in 1987, which established the basis for intrusion detection using system audit data. During the 1990s and early 2000s, IDSs mainly relied on predefined signatures and expert rules to detect known attacks. Although these methods were effective against previously identified threats, they became less suitable for detecting new, unknown, and evolving attacks.[21]

With the growth of network traffic and the appearance of more complex cyberattacks, ML started to play an important role in IDS research. From the late 1990s, especially after the release of benchmark datasets such as DARPA 1998 and KDD Cup 99, researchers began to train ML models to automatically distinguish between normal and malicious traffic. Unlike traditional IDSs, ML-based IDSs learn patterns directly from data and can identify suspicious behaviors without relying only on fixed signatures. More recent datasets, such as UNSW-NB15, CIC-IDS2017, and CSE-CIC-IDS2018, have further supported the development and evaluation of ML-based IDSs under more realistic network conditions.[22]

Therefore, ML-based IDSs represent an important step toward more adaptive and data-driven security systems. They can improve detection accuracy, reduce manual rule updates, and support the identification of both known and unknown attacks. For this reason, ML-based IDSs constitute the main focus of this study, where different learning strategies and feature selection techniques are investigated to improve detection performance and generalization.[22]

1.7.3 Industry and Research Adoption of ML-Based IDS

The increasing integration of ML into operational security systems is reflected by its deployment in real-world industrial and organizational contexts, where modern security solutions increasingly rely on data-driven models to detect and mitigate cyber threats. For instance, IBM QRadar [23] applies ML analytics to network data for behavioral baselining and anomaly detection, enabling automated threat identification across cloud, and hybrid environments.

Similarly, Cisco Secure Network Analytics uses multilayer ML to detect unknown malware, insider threats, and policy violations in real time.[24]. Microsoft Defender for Endpoint employs hybrid detection with a cloud-powered ML pipeline, analyzing trillions of signals daily through multiple specialized engines to identify new and unknown threats in milliseconds [25].

In parallel, the research community has experienced an explosive growth in papers exploring ML-based IDS, covering a wide range of approaches including supervised learn-

ing, unsupervised anomaly detection, and deep learning architectures. This extensive body of work has significantly broadened the scope of IDS research, enabling new directions in adaptive security, real-time detection, and intelligent cyber defense [26].

1.8 Challenges of ML-Based IDSs

Despite the significant advances achieved by ML techniques in IDS, several fundamental challenges still limit their effectiveness and can be summarized as follows:

- **Dataset Quality and Representativeness:** The performance of ML-based IDS heavily depends on the quality and realism of the training datasets. However, many widely used datasets such as KDD Cup 1999 suffer from being outdated and not representative of modern network traffic, while others may include synthetic or limited attack scenarios. This lack of realism reduces the generalization capability of trained models when deployed in real-world environments.[26]
- **Class Imbalance Problem:** IDS datasets typically exhibit severe class imbalance, where normal traffic significantly outnumbers attack samples, and some attack categories are extremely rare. This imbalance biases ML models toward majority classes, resulting in poor detection of minority but critical attacks. As a result, rare attacks may be overlooked, reducing the reliability of the system in practice.[27]
- **High Dimensionality and Irrelevant Features :** Network traffic data often contains a large number of features, many of which may be redundant, irrelevant, or noisy. High-dimensional feature spaces increase computational complexity and can negatively affect model performance due to overfitting. Therefore, effective feature selection or dimensionality reduction is essential for improving IDS efficiency and accuracy. [28]
- **Adversarial Attacks:** Adversarial attacks pose a significant limitation to ML-based IDSs, where attackers intentionally manipulate input features to mislead the detection model while maintaining malicious functionality. By exploiting weaknesses in the

model's decision boundaries, these subtle perturbations can lead to incorrect classifications of malicious traffic as benign [29].

- **Model Interpretability:** Many high-performing ML models, especially deep learning approaches, operate as black-box systems, making their decision-making process difficult to interpret. In cybersecurity applications, this lack of transparency is a critical limitation, as security analysts need to understand and justify detection results [30].

1.9 Conclusion

This chapter introduced the fundamental concepts of IIDSs, including their definition, role, and main types. It also discussed the transition from traditional IDS approaches to ML-based IDSs, which are more suitable for detecting complex and evolving cyberattacks.

Although ML-based IDSs offer important advantages, they still face several challenges, such as class imbalance, false alarms, poor generalization, and high dimensionality. In this thesis, special attention is given to the dimensionality problem, as network traffic datasets often contain a large number of features that may affect model efficiency and performance.

The next chapter will further explore this challenge by presenting the most important IDS datasets used in the domain and explaining how their characteristics, especially high dimensionality, influence the design and evaluation of ML-based IDSs.

Chapter 2

Network Intrusion Detection Datasets: Characteristics and Dimensionality Challenges

2.1 Introduction

The performance of ML-based IDSs strongly depends on the quality, structure, and characteristics of the datasets used for training and evaluation. As network traffic grows in volume, diversity, and complexity, IDS datasets often become highly dimensional, containing a large number of features that may affect detection efficiency and model performance. Therefore, understanding the main characteristics of intrusion detection datasets, together with the dimensionality problem and the techniques used to address it, is essential for improving detection accuracy, reducing computational complexity, and developing efficient, scalable, and reliable IDSs. [31]

This chapter provides a comprehensive overview of the primary datasets used in IDS benchmarking, highlighting their evolution and key characteristics. It also presents a comparison of these datasets in terms of their properties and suitability for intrusion detection tasks. Furthermore, the chapter discusses the dimensionality problem commonly encountered in these datasets and examines the most widely used feature selection and dimensionality reduction techniques employed to address this challenge.

2.2 Overview of the Main IDS-Related Datasets

ML-based IDSs is traditionally developed and evaluated using benchmark datasets that simulate real-world network traffic. In this section, we examine the historical evolution of these datasets and analyze the specific benchmarks used in contemporary cybersecurity research.

2.2.1 Evolution of IDS Datasets

Historically, the foundation of IDS benchmarking began with the DARPA 1998 project and its widely adopted derivative, the KDD Cup 99 dataset. For nearly a decade, KDD Cup 99 served as the standard for evaluating ML models in cybersecurity. However, researchers eventually identified critical statistical flaws within the dataset, most notably the presence of

a large number of redundant records, which caused learning algorithms to overfit the training data and perform poorly in real-world scenarios. [31]

To mitigate these fundamental issues, the NSL-KDD dataset was introduced as a refined, redundancy-free version of its predecessor. While NSL-KDD successfully provided a more accurate evaluation metric for classic algorithms, the rapid advancement of network technologies and the emergence of novel attack vectors soon rendered its attack signatures obsolete.[32]

Consequently, the focus of modern cybersecurity research has shifted toward more recent and realistic datasets, including UNSW-NB15, CICIDS2017, CSE-CIC-IDS2018, CIC-DDoS2019, Bot-IoT, ToN-IoT, Edge-IIoTset, NF-UQ-NIDS, and the recently introduced FLNET2023 dataset. Unlike earlier benchmark datasets, these contemporary datasets are designed to reflect modern network infrastructures, encompassing enterprise networks, cloud and edge computing environments, Internet of Things (IoT) ecosystems, Industrial Internet of Things (IIoT) deployments, and Federated Learning (FL) architectures. They include a diverse spectrum of sophisticated cyberattacks, such as Distributed Denial-of-Service (DDoS), botnet activities, web-based intrusions, brute-force attacks, infiltration attempts, data exfiltration, and attacks targeting industrial communication protocols.[31]

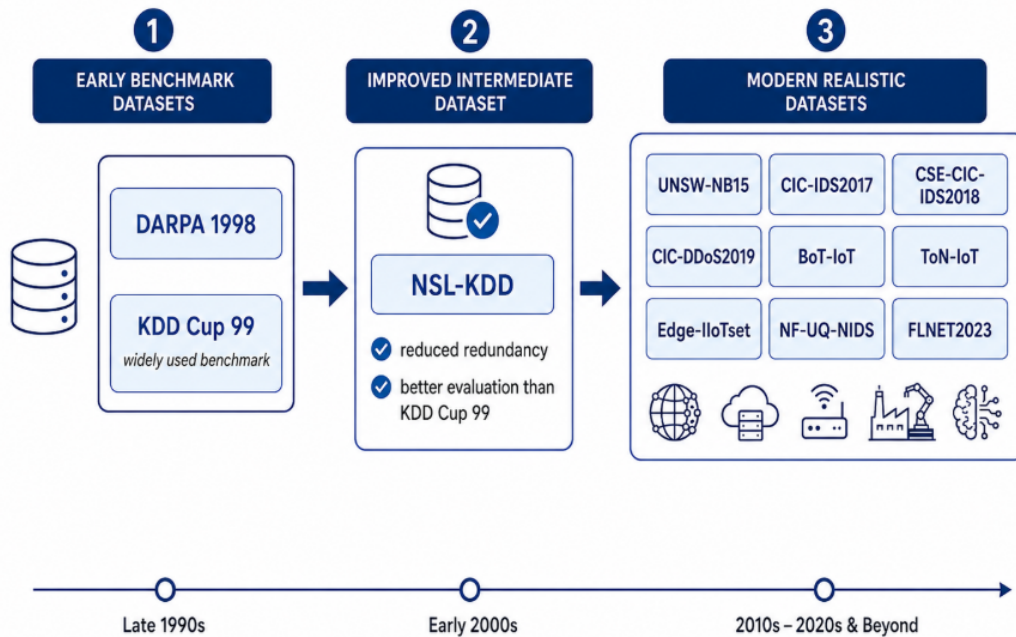


Figure 2.1: IDS datasets evolution.

2.2.2 The KDD Cup 99 Dataset

The KDD Cup 99 dataset remains the most widely cited benchmark in the history of intrusion detection research. It was derived from the 1998 DARPA Intrusion Detection Evaluation program, simulating a military network environment subjected to various cyber-attacks. The dataset was generated over a period of approximately seven weeks within a simulated Local Area Network (LAN) environment. Network traffic was captured using the tcpdump utility, resulting in a total of 4,898,431 connection records. Each records is represented by 41 traffic-related features that describe various characteristics of the observed network activity. The dataset contains both normal and malicious traffic, with attack instances categorized into four major classes: Denial of Service (DoS), Remote-to-Local (R2L), User-to-Root (U2R), and Probing attacks. [33]. Table 2.1 shows the classes included in this dataset.

Traffic			Instances		
Attack	DoS	Back	2203	3883370	3925650
		Land	21		
		Neptune	1072017		
		Pod	264		
		Smurf	2807886		
		Teardrop	979		
	Probe	Ipsweep	12481	41102	
		Nmap	2316		
		PortswEEP	10413		
		Satan	15892		
	U2R	Buffer	30	52	
		Loadmodule	9		
		Perl	3		
		Rootkit	10		
	R2L	Ftp_write	8	1126	
		Guess_pwd	53		
		Imap	12		
		Multihop	7		
		Phf	4		
		Spy	2		
		Warezcclient	1020		
		Warezmaster	20		
Normal					972781
Total					4898431

Table 2.1: KDD Cup 99 Traffic Distribution.

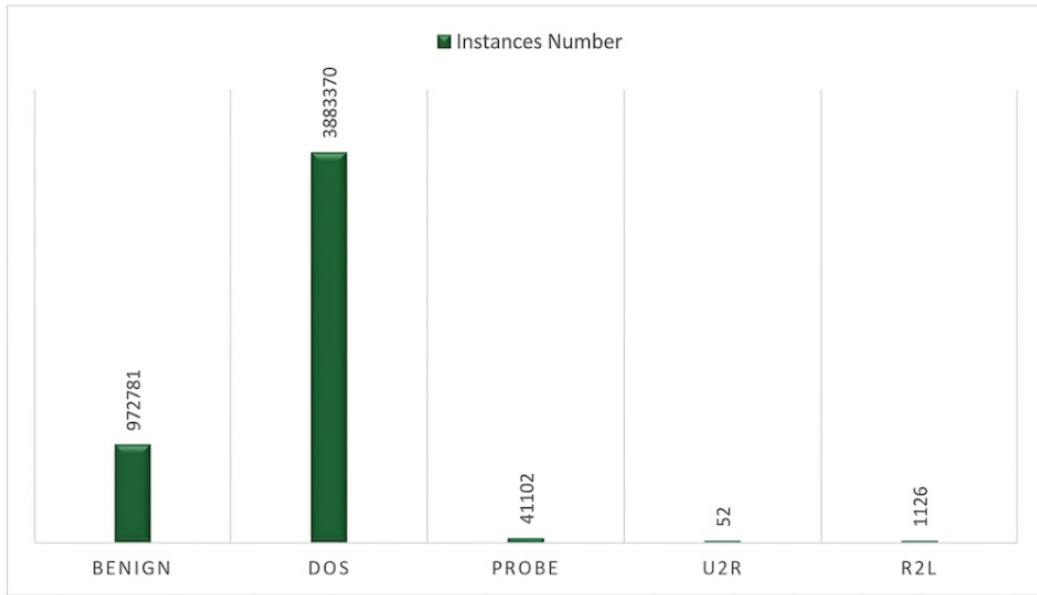


Figure 2.2: Number of instances by traffic types in the KDD Cup 99 dataset.

2.2.3 The NSL-KDD Dataset

To address the significant analytical shortcomings of the KDD Cup 99 dataset, researchers introduced NSL-KDD as a refined and optimized alternative. The primary objective behind the creation of NSL-KDD was to eliminate the massive redundancy present in its predecessor by carefully removing duplicate records from both the training and testing sets. Structurally, the NSL-KDD dataset preserves the same architecture as the original KDD Cup 1999 dataset, retaining its 41 network traffic features and four principal attack categories. However, NSL-KDD offers a more balanced distribution of normal and malicious instances. In the training set, benign traffic accounts for approximately 53.45% of the samples, while attack traffic represents 46.55%. Similarly, in the test set, benign and malicious records constitute 43.08% and 56.92% of the data, respectively. [34]

Traffic Types			Number of Instances					
			Train			Test		
Attacks	DOS	Back	956	45927	58630	359	7460	12833
		Land	18			7		
		Neptune	41214			4657		
		Pod	201			41		
		Smurf	2646			665		
		Teardrop	892			12		
		Mailbomb	0			293		
		Apache2	0			737		
		Udpstorm	0			2		
		Processtable	0			685		
		Worm	0			2		
	Probe	Ipsweep	3599	11656		141	2421	
		Nmap	1493			73		
		Portsweep	2931			157		
		Satan	3633			735		
		Saint	0			319		
		Mscan	0			996		
	U2R	Buffer_overflow	30	52		20	67	
		Loadmodule	9			2		
		Perl	3			2		
		Rootkit	10			13		
		Xterm	0			13		
		Ps	0			15		
		Sqlattack	0			2		
	R2L	Ftp_write	8	995		3	2885	
		Guess_passwd	53			1231		
		Imap	11			1		
		Multihop	7			18		
		Phf	4			2		
		Spy	2			0		
		Warezclient	890			0		
		Warezmaster	20			944		
		Sendmail	0			14		
		Named	0			17		
		Snmpgetattack	0			178		
		Snmpguess	0			331		
		Xlock	0			9		
		Xsnoop	0			4		
		Httpunnel	0			133		
Normal			67343			9711		
Total			125973			22544		

Table 2.2: NSL-KDD Dataset Traffic Distribution.

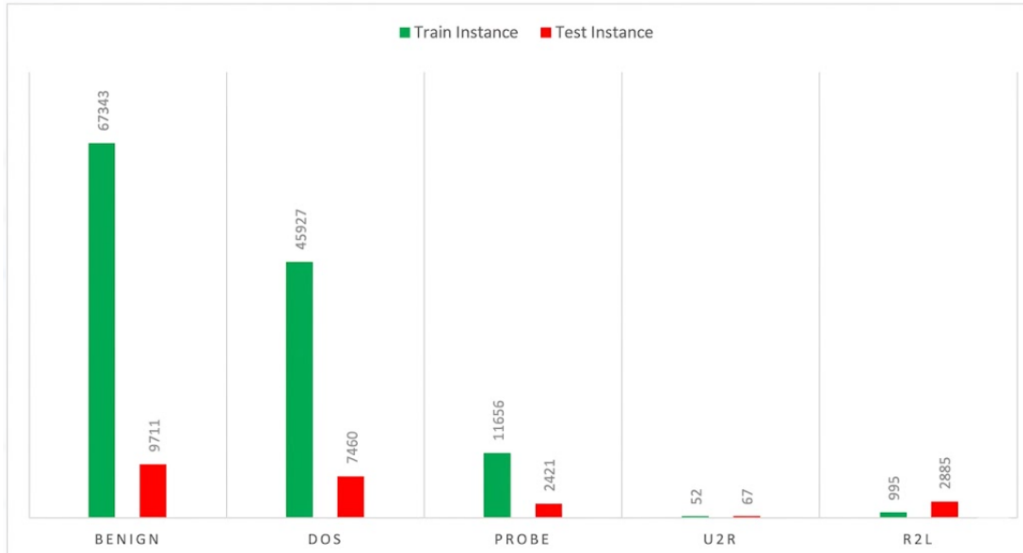


Figure 2.3: Number of Instances by Traffic Types in the NSL-KDD Dataset.

Consequently, it lacks the representation of modern, low-footprint attack vectors and contemporary network topologies. This limitation paved the way for the development of newer, more complex datasets that better reflect today's threat landscape.

2.2.4 The UNSW-NB15 Dataset

Developed by the Cyber Research Group at the Australian Centre for Cyber Security (ACCS), the UNSW-NB15 dataset was designed to address the shortcomings of older benchmark datasets. It was generated using the IXIA PerfectStorm tool, which simulates a mix of modern normal network activities and nine distinct types of contemporary synthetic attacks (e.g., Fuzzers, Backdoors, and Worms). The dataset contains 49 features, which are more diverse and representative of modern network behaviors compared to those found in the KDD and NSL-KDD datasets.

In terms of distribution, the UNSW-NB15 dataset contains a total of 2,540,044 connection records, among which 2,218,761 correspond to normal (benign) traffic, representing approximately 87.38% of the data, while 321,283 records are associated with different types of attacks, accounting for 12.62%.[35]

Traffic Types		Attack subcategory	NB of Instances	
Attacks	Fuzzers	FTP, HTTP, RIP, SMB, Syslog, PPTP, FTP, DCERPC, OSPF, TFTP, DCERPC, OSPF, BGP.	24246	321283
	Reconnaissance	Telnet, SNMP, SunRPC Portmapper (TCP) UDP Service, SunRPC Portmapper (TCP) TCP Service, SunRPC Portmapper (UDP) UDP Service, NetBIOS, DNS, HTTP, SunRPC Portmapper (UDP), ICMP, SCTP, MSSQL, SMTP, Telnet, SunRPC Portmapper (UDP) TCP Service.	13987	
	Shellcode	FreeBSD, HP-UX, NetBSD, AIX, SCO Unix, Linux, Decoders, IRIX, OpenBSD, Mac OS X, BSD, Windows, BSDi, Multiple OS, Solaris.	1511	
	Analysis	HTML, Port Scanner, Spam.	2677	
	Backdoors	/	2329	
	DoS	Ethernet, Microsoft Office, VNC, IRC, RDP, TCP, FTP, LDAP, Oracle, TFTP, DCERPC, XINETD, SNMP, ISAKMP, NTP, Telnet, CUPS, Hypervisor, ICMP, SunRPC, IMAP, Asterisk, Browser, Cisco Skinny, SIP, SMTP, SSL, DNS, IIS Web Server, Miscellaneous, RTSP, Common Unix Print System (CUPS), IGMP, HTTP, NetBIOS/SMB, Windows Explorer.	16353	
	Exploits	Evasions, SCCP, SSL, VNC, Backup Appliance, Browser, ClientSide Microsoft Office, Interbase, Miscellaneous Batch, SOCKS, TCP, Apache, IMAP, Microsoft IIS, ClientSide, ClientSide Microsoft Paint, IDS, SSH, ICMP, DCERPC, FTP, RADIUS, WINS, ClientSide Microsoft, POP3, Unix r Service, Cisco IOS, ClientSide Microsoft Media Player, Damewar, LPD, MSSQL, Office Document, RTSP, SCADA, Webserver, All, LDAP, NNTP, IGMP, Oracle, RDesktop, Telnet, PHP, SMB, SunRPC, Web Application, DNS, SMTP, Browser FTP, Miscellaneous, PPTP, SIP, TFTP.	44525	
	Generic	All, SIP, HTTP, SMTP, IXIA, TFTP, Superflow.	215481	
	Worms	/	174	
Normal				2218761
Total				2540044

Table 2.3: UNSW-NB15 Dataset Traffic Distribution.

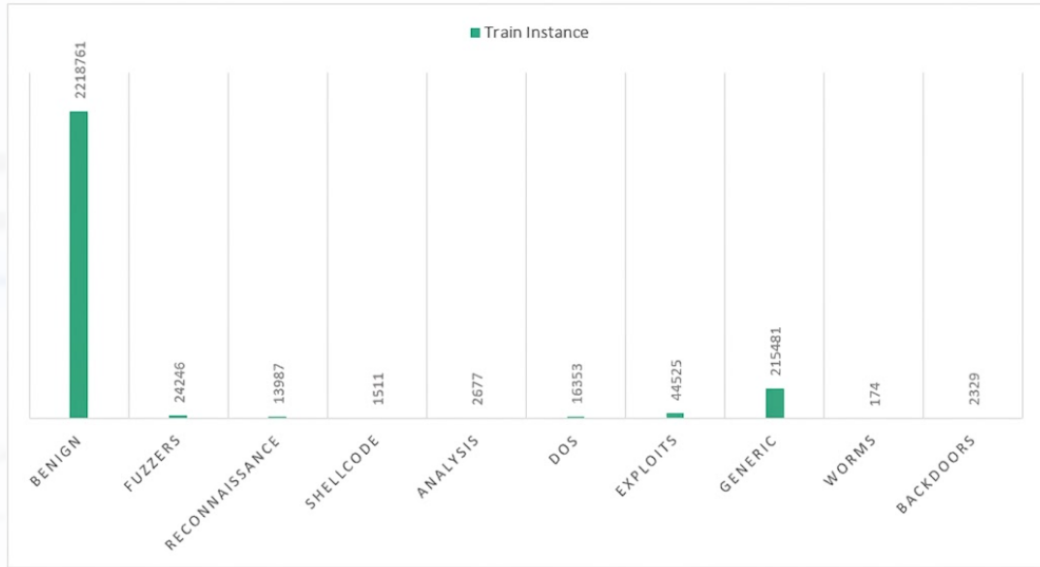


Figure 2.4: Number of Instances by Traffic Types in the UNSW-NB15 Dataset.

2.2.5 The CIC-IDS2017 Dataset

Developed by the Canadian Institute for Cybersecurity (CIC), the CIC-IDS2017 dataset was introduced to provide a reliable and contemporary benchmark that closely reflects real-world network conditions. Unlike earlier datasets, it includes a wide variety of modern attack scenarios, such as DDoS, Brute Force, Web Attacks, and Botnets, embedded within realistic and continuous background traffic. Overall, the CIC-IDS2017 dataset comprises a total of 2,830,743 network flow records, of which 2,049,979 correspond to benign traffic, representing 72.43% of the dataset, while 780,764 records are labeled as attack instances, accounting for 27.57%. In addition, the dataset contains 79 network traffic features extracted from flow-based characteristics, making it suitable for ML-based intrusion detection [36]. It covers 14 distinct types of attacks distributed across eight separate files as detailed in Table (2.4).

Name of CSV Files	Day Activity	Traffic Types	NB of Instances	
Friday-WorkingHours-Afternoon DDos.pcap_ISCX	Friday	BENIGN	97718	225745
		DDoS	128027	
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX	Friday	BENIGN	127537	286467
		PortScan	158930	
Friday-WorkingHours-Morning.pcap_ISCX	Friday	BENIGN	189067	191033
		Bot	1966	
Monday-WorkingHours.pcap_ISCX	Monday	BENIGN	529918	529918
Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX	Thursday	BENIGN	288566	288602
		Infiltration	36	
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX	Thursday	BENIGN	168186	170366
		Web Attack-XSS	652	
		Web Attack-Brute Force	1507	
		Web Attack-Sql Injection	21	
Tuesday-WorkingHours.pcap_ISCX	Tuesday	BENIGN	432074	445909
		FTP-Patator	7938	
		SSH-Patator	5897	
Wednesday-workingHours.pcap_ISCX	Wednesday	BENIGN	440031	692703
		DoS Hulk	231073	
		DoS Slowloris	5796	
		DoS Slowhttptest	5499	
		Heartbleed	11	
		DoS GoldenEye	10293	
Total		15	2830743	

Table 2.4: CIC-IDS2017 Dataset Traffic Distribution.

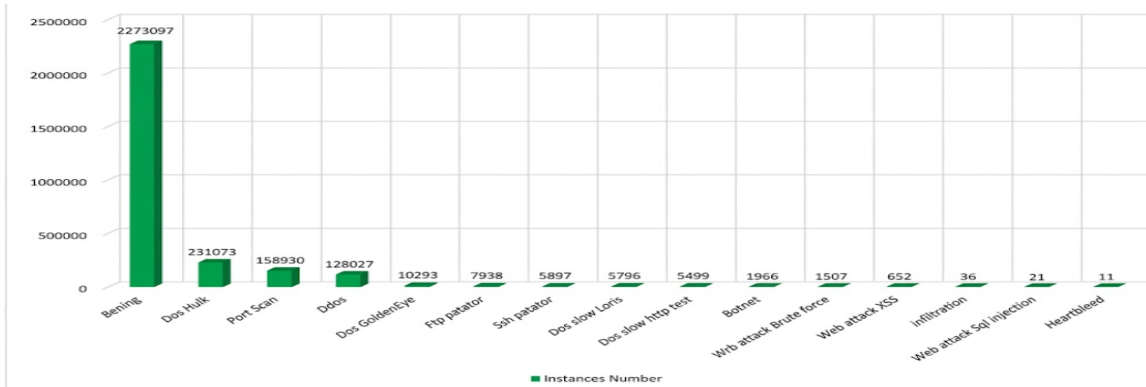


Figure 2.5: Number of Instances by Traffic Types in the CIC-IDS2017 Dataset.

2.2.6 The CSE-CIC-IDS2018 Dataset

Building upon the robust foundation of its 2017 predecessor, the CSE-CIC-IDS2018 dataset was developed as a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The primary motivation behind this iteration was to evaluate IDSs on a substantially larger and more distributed scale. To achieve this, the entire network infrastructure was deployed on Amazon Web Services (AWS), accurately simulating a large-scale modern enterprise cloud environment.

The dataset contains 84 features covering both benign traffic and 14 distinct types of modern cyberattacks. The data was collected over a ten-day period. In total, the CSE-CIC-IDS2018 dataset comprises over 16 million records, including 13,479,671 benign samples (83.07%) and 2,747,397 attack samples (16.93%), reflecting a realistic and large-scale traffic distribution.[36]

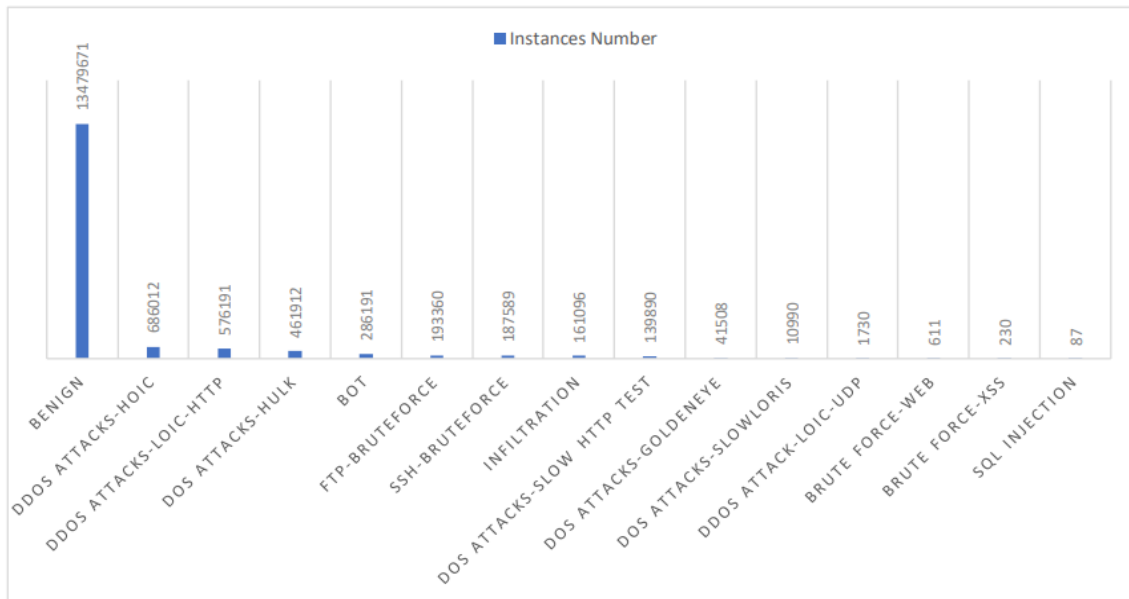


Figure 2.6: Number of Instances by Traffic Types in the CIC-IDS2018 Dataset.

Name of CSV Files	Day Activity	Traffic Types	NB of Instances	
Friday-02-03-2018- _TrafficForML_CICFlowMeter.csv	Friday	Benign	762384	1048575
		Bot	286191	
Friday-16-02-2018- _TrafficForML_CICFlowMeter.csv	Friday	Benign	446772	1048574
		DoS attacks-SlowHTTPTest	139890	
		DoS attacks-Hulk	461912	
Friday-23-02-2018- _TrafficForML_CICFlowMeter.csv	Friday	Benign	1048009	1048575
		Brute Force -Web	362	
		Brute Force -XSS	151	
		SQL Injection	53	
Tuesday-20-02-2018- _TrafficForML_CICFlowMeter.csv	Tuesday	Benign	7372557	7948748
		DDoS attacks-LOIC-HTTP	576191	
Thursday-01-03-2018- _TrafficForML_CICFlowMeter.csv	Thursday	Benign	236632	329266
		Infiltration	92634	
Thursday-15-02-2018- _TrafficForML_CICFlowMeter.csv	Thursday	Benign	996077	1048575
		DoS attacks-GoldenEye	41508	
		DoS attacks-Slowloris	10990	
Thursday-22-02-2018- _TrafficForML_CICFlowMeter.csv	Thursday	Benign	1048213	1048575
		Brute Force -Web	249	
		Brute Force -XSS	79	
		SQL Injection	34	
Wednesday-14-02-2018- _TrafficForML_CICFlowMeter.csv	Wednesday	Benign	667626	1048575
		FTP-BruteForce	193360	
		SSH-Bruteforce	187589	
Wednesday-21-02-2018- _TrafficForML_CICFlowMeter.csv	Wednesday	Benign	360833	1048575
		DDOS attack-HOIC	686012	
		DDOS attack-LOIC-UDP	1730	
Wednesday-28-02-2018- _TrafficForML_CICFlowMeter.csv	Wednesday	Benign	540568	609030
		Infiltration	68462	
Total		15	16227068	

Table 2.5: CIC-IDS2018 Dataset Traffic Types.

2.3 Comparative Study of Benchmarking Datasets

Table 2.6 presents a comparative analysis of several IDS datasets according to different characteristics, including year, number of features, number of instances, data quality, traffic types, and representativeness. The results show that early datasets such as KDD Cup 99 and NSL-KDD are useful historical benchmarks, but they are less representative of modern network traffic. KDD Cup 99 contains many redundant records, while NSL-KDD improves data quality by removing redundancy, but it is still derived from an old dataset.

Compared with these older datasets, UNSW-NB15 provides more realistic traffic and

a larger number of attack types. However, CIC-IDS2017 and CSE-CIC-IDS2018 are more recent and contain larger and more representative traffic samples. They also include a higher number of features and traffic types, making them more suitable for evaluating feature selection techniques. In particular, CIC-IDS2017 contains 79 features and 15 traffic types, while CSE-CIC-IDS2018 contains 84 features and a much larger number of instances.

Based on this comparison, CIC-IDS2017 and CSE-CIC-IDS2018 were selected for the experimental study. These datasets are recent, realistic, highly representative, and contain high-dimensional network traffic data. Therefore, they are suitable for evaluating the effectiveness of feature selection techniques in reducing dimensionality while maintaining good intrusion detection performance.

Attribute	KDD'99	NSL-KDD	UNSW-NB15	CIC-IDS2017	CIC-IDS2018
Year	1999	2009	2015	2017	2018
Train/Test Split	No	Yes	No	No	No
Derived From	Simulated	KDD Cup 99	Realistic Net.	Realistic Net.	Realistic Net.
Features	41	41	49	79	84
Class Balance	Unbalanced	Unbalanced	Unbalanced	Unbalanced	Unbalanced
Time Span	9 weeks	Same as KDD	1 month	5 days	10 days
Instances	4,898,431	147,717	2,540,044	2,830,743	16,227,068
Data Quality	High red.	Red. removed	Realistic/diverse	Very high	Very high
Source	DARPA 1998	CIC & UNB	ACCS	CIC & UNB	CIC & CSE
Traffic Types	5	5	10	15	15
Representative	No	Slightly better	Modern traffic	High	High
Noise Level	High	Lower than KDD	High	Very low	Very low

Table 2.6: Comprehensive Comparison of Prominent IDS Datasets

[37]

2.4 High-Dimensionality Problem

After presenting the main IDS benchmark datasets and their characteristics, an important challenge emerges from their richness of information. Modern IDS datasets often contain a large number of features designed to capture various aspects of network traffic and attack behaviors. For example, widely used datasets such as CIC-IDS2017 and CSE-CIC-IDS2018 each include approximately 80 network flow features, while other recent datasets

may contain an even larger number of attributes. Although these features provide valuable information for distinguishing between normal and malicious traffic, they also introduce the high-dimensionality problem, which has become a major concern in ML-based IDS.

2.4.1 Definition

The high-dimensionality problem in ML often called the curse of dimensionality refers to phenomena that occur when analyzing datasets with a massive number of features (variables)[38].

2.4.2 Impact of the High-Dimensionality Problem

As the number of features increases, the complexity of data analysis and model learning also grows, leading to several challenges that negatively affect detection capability and computational efficiency. The main impacts are presented as follows:

- **Increased Storage Requirements:** High-dimensional network traffic datasets often require substantial storage capacity due to the large volume of extracted features. Managing and storing such datasets can become costly and inefficient in real-world cybersecurity infrastructures.[39]
- **Increased Computational Complexity:** High-dimensional datasets require greater computational resources for data processing, model training, and prediction. The presence of a large number of features increases memory consumption and training time, making real-time intrusion detection more difficult, especially in large-scale network environments.[40]
- **Risk of Overfitting:** When the feature space becomes excessively large, ML models may learn irrelevant patterns and noise instead of meaningful attack behaviors. This phenomenon, known as overfitting, reduces the model's ability to generalize effectively to unseen network traffic.[41]
- **Difficulty in Model Interpretation:** A large number of features complicates the interpretation of model decisions and the identification of the most influential network

attributes. This reduces the transparency and explainability of IDS, particularly in security-critical applications.[42]

2.4.3 Dimensionality Reduction Techniques

To mitigate the "Curse of Dimensionality" and improve the performance of ML-based IDS, researchers employ two primary strategies to reduce the number of features. While both aim to simplify the data, they do so through fundamentally different mechanisms.

- **Feature Selection (FS):** This technique involves selecting a specific subset of the original features that are most relevant to the prediction task . It discards redundant or noisy attributes without changing their original form. For example, in network traffic, we might keep "Protocol" and "Packet Size" but discard "Timestamp." [43].
- **Feature Reduction (FR):** Unlike selection, this approach transforms the original features into a completely new, smaller set of features (often called components). It mathematically projects the high-dimensional data into a lower-dimensional space while trying to preserve as much information as possible.[44].

2.5 Key Feature Selection (FS) Methods

As illustrated in Figure 2.8, feature selection methods are generally categorized into four main types. Each type follows a specific strategy to identify the most relevant features.

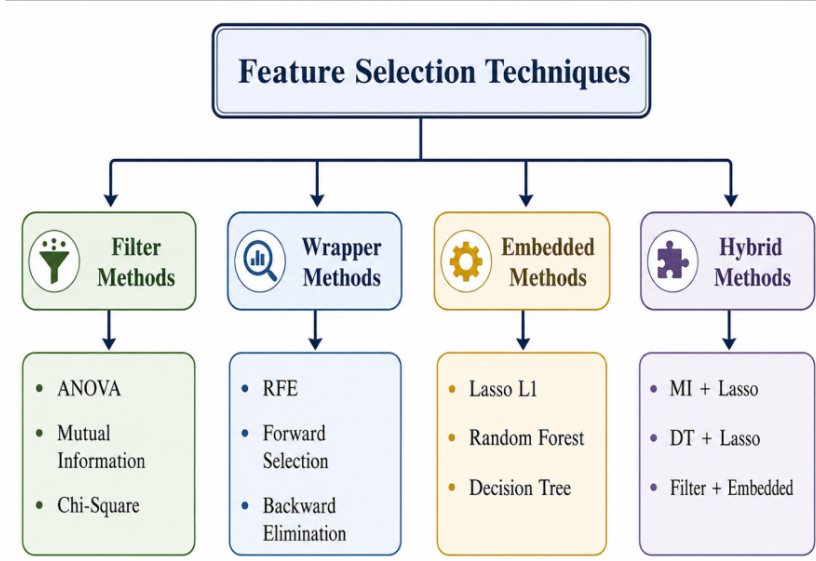


Figure 2.7: Feature Selection Categories.

2.5.1 Filter Methods

Filter methods refer to techniques that rank features based on statistical scores such as mean, variance, and correlation coefficients, allowing for the selection or removal of features independently of a ML algorithm's training process.

These methods are computationally inexpensive, easy to implement. However, they do not consider multivariate interactions among features, as each feature is evaluated individually and assigned a statistical score representing its relevance to the target class.[45] To illustrate this mechanism, we examine two widely adopted filter-based techniques:

- **Mutual Information (MI):** Mutual Information is a powerful filter method that quantifies the mutual dependence between a specific network feature and the target label. It excels at capturing complex non-linear dependencies by measuring the reduction in uncertainty about the target label given the knowledge of the feature [46]. The discrete formulation of MI is calculated as follows:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \quad (2.1)$$

Where:

$I(X;Y)$ The Mutual Information score between the feature X and the target label Y (a higher value indicates greater dependency).

X, Y The discrete state spaces of the network feature and the target label, respectively.

$p(x, y)$ The joint probability mass function of the feature and the target label occurring together.

$p(x)$ The marginal probability of the feature X occurring.

$p(y)$ The marginal probability of the target label Y occurring.

- **Analysis of Variance (ANOVA)** : Analysis of Variance (ANOVA) is a filter-based feature selection method used to measure the relationship between each numerical feature and the target class. It evaluates whether the values of a feature are significantly different from one class to another. In other words, ANOVA checks if a feature can help distinguish between the different classes. This is done using the F-score, which compares the variation between classes with the variation inside each class. A high F-score means that the feature has strong discriminative power and is more useful for classification [47]. The F-score is calculated as follows:

$$F = \frac{\frac{1}{K-1} \sum_{k=1}^K n_k (\bar{X}_k - \bar{X})^2}{\frac{1}{N-K} \sum_{k=1}^K \sum_{i=1}^{n_k} (X_{ik} - \bar{X}_k)^2} \quad (2.2)$$

Where:

F The calculated F-statistic score for the feature.

K The total number of categories (classes) in the target label.

N The total number of samples in the dataset.

n_k The number of samples belonging to class k .

$\bar{X}_k$ The arithmetic mean of the feature values specifically within class k .

$\bar{X}$ The overall mean of the feature values across the entire dataset.

X_{ik} The value of the i -th sample of the feature belonging to class k .

2.5.2 Wrapper Methods

Wrapper methods differ from filter methods by utilizing a specific ML classifier to evaluate the quality of feature subsets. They treat the learning algorithm as a "black box" and use its predictive performance, such as detection accuracy, as the objective function for assessing different combinations of features. To identify the most relevant subset, wrapper methods search through the space of possible feature combinations, repeatedly training and testing the model and selecting the subset that yields the best performance.

Although wrapper methods often achieve superior feature selection results, they are computationally expensive, especially when applied to large-scale, high-dimensional datasets. Evaluating numerous feature subsets requires repeated model training, resulting in significant processing time and memory consumption.[48]

Consequently, applying wrapper methods to massive IDS datasets may be impractical without substantial computational resources. Due to these high hardware requirements and processing delays, wrapper methods were excluded from the practical implementation phase of this thesis.

A prominent example of this category is:

- **Recursive Feature Elimination (RFE):** Recursive Feature Elimination (RFE) is a wrapper-based feature selection method that selects features through a step-by-step elimination process. It starts by training a ML model using all available features. Then, the importance of each feature is estimated according to the model. The least important feature, or group of features, is removed from the dataset. After that, the model is trained again on the remaining features, and the same process is repeated. This continues until the desired number of features is reached. In this way, RFE helps identify the most useful features for improving classification performance.[49]

2.5.3 Embedded Methods

Embedded methods integrate the feature selection process directly into the training phase of a ML algorithm. In this approach, feature importance is learned simultaneously

with the optimization of the model parameters through an iterative process that minimizes the loss function. A regularization term is added to the objective function, which encourages the model to focus on the most informative features. As a result, the model assigns higher weights to relevant features while reducing or setting to zero the weights of irrelevant ones, thereby performing automatic feature selection during training.[45]

A primary example of this category is:

- **LASSO L1 regularization:** Least Absolute Shrinkage and Selection Operator (LASSO) is an embedded feature selection method based on L1 regularization . It is used to select the most relevant features while reducing the influence of less important ones. The main idea of LASSO is to add a penalty term to the learning process, which forces some feature coefficients to become very small or exactly zero.

When the coefficient of a feature becomes zero, this means that the feature does not contribute significantly to the prediction task and can be removed from the dataset. Therefore, LASSO performs feature selection automatically during model training. The strength of this selection process is controlled by the regularization parameter, usually denoted by λ . A higher value of λ increases the penalty and leads to the selection of fewer features. Thus, LASSO helps reduce dimensionality, simplify the model, and limit overfitting.[50].

- **Random Forest (RF) Feature Importance:** Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their results to improve prediction performance. In addition to classification, Random Forest can be used as an embedded feature selection method because it estimates feature importance during the training phase. The importance of a feature is commonly measured using the Mean Decrease in Impurity (MDI), which evaluates how much a feature contributes to reducing impurity when it is used to split the data in decision trees.[51].

In simple terms, a feature receives a high importance score if it frequently helps the trees create better splits and separate the target classes more effectively. Therefore, features that strongly reduce impurity across the forest are considered more relevant. These importance scores are then used to rank the features from the most informative

to the least informative, allowing less useful features to be removed and reducing the dimensionality of the dataset.

2.5.4 Hybrid Methods

Hybrid methods aim to combine the strengths of multiple feature selection techniques to achieve a balance between computational efficiency and predictive performance. By adopting a multi-stage approach, these methods are particularly effective in handling high-dimensional datasets, often outperforming any single feature selection technique.

In general, hybrid feature selection strategies can take different forms. They may combine techniques from different categories, for example integrating Filter and Embedded methods in a sequential pipeline, or they may combine multiple techniques within the same category, such as applying several Embedded methods successively to progressively refine the feature space.[52]

Table 2.7 presents a comparative analysis of the three main categories of feature selection techniques: filter, wrapper, and embedded methods.

Evaluation Criterion	Filter Methods	Wrapper Methods	Embedded Methods
Core Mechanism	Evaluates features based on their intrinsic statistical properties independently of any ML algorithm.	Employs a specific ML classifier as a black box to evaluate feature subsets iteratively through search strategies.	Performs feature selection intrinsically during the model training phase.
Classifier Dependency	Completely independent (Model-agnostic).	Fully dependent on the chosen ML classifier.	Tied to specific learning algorithms that support internal selection.
Computational Complexity	Extremely low; computationally efficient and highly scalable to massive high-dimensional data.	Exceptionally high; requires retraining the model for every evaluated feature subset combination.	Moderate; significantly faster than wrappers as selection occurs in a single training run.
Risk of Overfitting	Very low	High.	Moderate
Feature Interactions	Generally ignores feature interactions, assessing attributes individually .	Successfully captures complex non-linear dependencies between multiple features.	Effectively captures feature interactions.

Table 2.7: Comparison of Filter, Wrapper, and Embedded Feature Selection Paradigms

2.6 Key Feature Reduction (FR) Methods

This section presents the key Feature Reduction (FR) methods adopted in this study to reduce data dimensionality and retain the most relevant information for classification.

2.6.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used feature reduction technique in ML. Its main objective is to reduce the number of features while preserving as much useful information as possible. PCA transforms the original features into a new set of variables called principal components. These components are uncorrelated and are ordered according to the amount of information, or variance, they capture from the original data.

In practice, PCA first standardizes the data so that all features contribute equally. Then, it analyzes the relationships between features and identifies the main directions where the data varies the most. The first principal components contain the largest amount of information, while the last components usually contain less useful information or noise. By keeping only the top (k) components, PCA reduces dimensionality, decreases computational cost, and helps simplify the learning process with limited information loss.[53]

2.6.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised feature reduction technique mainly used for classification tasks. Unlike PCA, which reduces dimensionality by preserving the maximum variance in the data without considering class labels, LDA uses the target labels during the reduction process. Its main goal is to project the original features into a lower-dimensional space where the different classes are better separated.[54].

2.7 Conclusion

This chapter presented the main IDS datasets used in the literature and discussed their evolution from early benchmarks, such as DARPA 1998 and KDD Cup 99, to more recent and realistic datasets, including UNSW-NB15, CIC-IDS2017, CSE-CIC-IDS2018, and other modern datasets. This evolution shows the continuous need for datasets that better reflect current network environments and recent cyberattacks.

The chapter also highlighted the dimensionality problem in IDS datasets, where the large number of features can increase computational cost, affect model performance, and make learning more complex. To address this issue, several feature selection and feature reduction techniques were presented, including filter, wrapper, embedded, hybrid methods and PCA.

Overall, this chapter showed that selecting the most relevant features is an important step in building efficient ML-based IDSs. Feature selection helps reduce data complexity, improve training efficiency, and support better detection performance. These concepts provide the foundation for the experimental methodology presented in the following chapter.

Chapter 3

Experimental Evaluation of Dimensionality Reduction Techniques

3.1 Introduction

This chapter presents the methodology used to compare different feature selection and reduction techniques for ML-based IDSs. It explains how these techniques are applied, tested, and compared in the experiments.

The methodology includes several main steps: dataset preparation, class balancing, feature selection, model training, and performance evaluation. The comparison is done using two classification tasks: binary classification and multi-class classification. Binary classification separates normal traffic from attack traffic, while multi-class classification identifies the type of attack.

Two evaluation strategies are used in this work: internal validation and external validation. In internal validation, training and testing are performed on different subsets of the same dataset. In external validation, the model is trained on one dataset and tested on another independent dataset. CIC-IDS2017 and CSE-CIC-IDS2018 are used for both evaluation strategies.

This chapter does not aim to evaluate all existing feature selection and reduction techniques. Instead, it adopts a set of the most commonly used techniques in the literature to provide a clear and practical comparison. The main goal is to identify which of these techniques can provide good detection results, reduce the number of features, and improve generalization across datasets.

3.2 Global methodology

This section presents the global methodology followed in this study. As illustrated in Figure 3.1, the proposed workflow follows a comprehensive process starting from dataset cleaning and harmonisation, followed by data splitting, training set preparation steps, model training, and performance evaluation. This methodology is designed to ensure a clear and consistent experimental framework for evaluating feature selection and reduction techniques under both internal and external validation strategies.

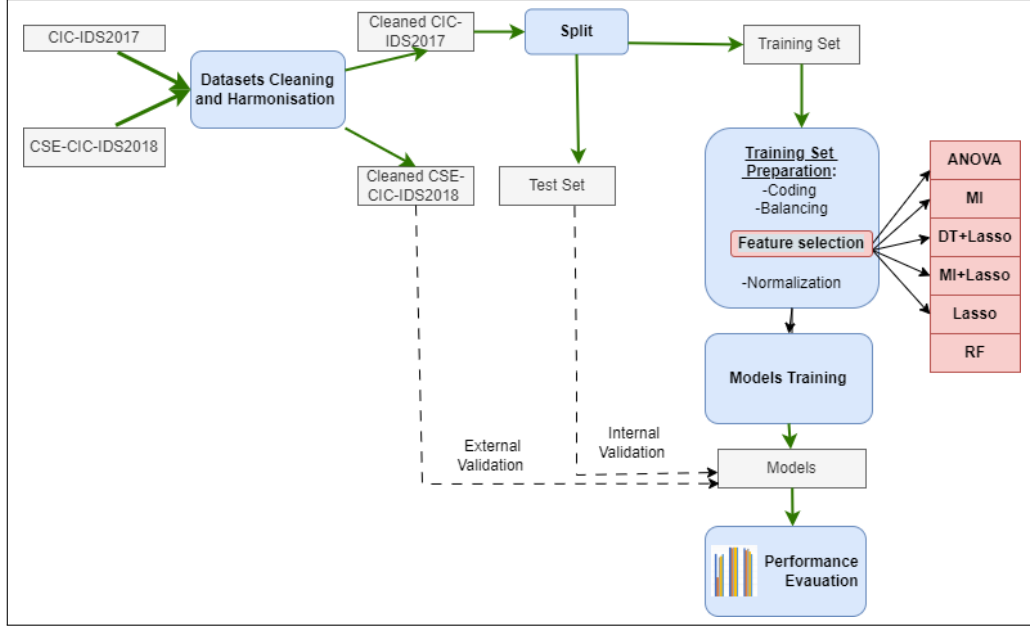


Figure 3.1: Global Methodology.

3.2.1 Datasets Cleaning and Harmonisation

Before applying feature selection techniques and training the ML models, a cleaning and harmonisation step was performed on the selected IDS datasets. This step aims to improve data quality and ensure that the datasets have a consistent structure for fair comparison and evaluation. First, missing values, infinite values, and duplicated records were removed to avoid biased learning and unreliable results. Then, feature alignment was applied to keep only the common and compatible features between the datasets. Finally, attack labels were reorganized into broader categories through an attack grouping strategy, allowing a more consistent representation of attack classes across datasets

- **Handling Missing and Infinite Values:** We converted all infinite computational values to Not-a-Number (*NaN*) and subsequently removed all rows containing missing values to maintain dataset integrity.
- **Addressing Inconsistent Values:** Non-numeric entries found in numerical feature columns were converted into *NaN* values and subsequently removed to ensure data consistency and prevent errors during model training.
- **Removing Duplicate Records:** Redundant observations were identified and elimi-

nated to mitigate potential model bias and prevent overfitting to duplicated samples.

- **Feature Alignment:** Due to inconsistencies in feature naming conventions between the CICIDS2017 and CICIDS2018 datasets, we performed a thorough feature harmonization process. We mapped the feature names of the 2018 dataset to perfectly align with those of the 2017 dataset using a predefined mapping dictionary (e.g., standardizing “Dst Port” to “Destination Port” and “Tot Fwd Pkts” to “Total Fwd Packets”). This ensured a uniform feature space across both datasets.
- **Attack Grouping:** Based on previous studies using the CIC-IDS2017 dataset and considering the class imbalance problem, we adopted an attack grouping strategy to obtain broader and more representative attack categories, as shown in Table 1. Closely related attack types were merged according to their semantic similarity and behavior. More specifically, “DoS GoldenEye,” “DoS Slowhttptest,” and “DoS Slowloris” were grouped into a single DoS class, while “FTP-Patator” and “SSH-Patator” were combined under the Patator class. Similarly, the three web-based attack subtypes, namely “Web Attack – SQL Injection,” “Web Attack – XSS,” and “Web Attack – Brute Force,” were unified into one Web based class. However, “DoS Hulk” was kept as a separate class because it already contains a sufficient number of instances. In addition, due to the very limited number of samples associated with the Infiltration and Heartbleed attacks in CIC-IDS2017, these classes were removed from the experimental analysis to avoid unreliable learning and evaluation results. The same attack grouping strategy was also applied to the CIC-IDS2018 dataset in order to ensure consistent class representation across datasets. Table 3.1 illustrates this grouping strategy.

New Classes	Original Attack Names	CIC-IDS2017 Instances	CIC-IDS2018 Instances
Benign	BENIGN / Benign	2,271,320	13,410,471
DoS Hulk	DoS Hulk	230,124	434,873
DoS	DoS GoldenEye	21,588	71,202
	DoS Slowloris		
	DoS SlowHTTPTest		
DDoS	DDoS / DDoS-LOIC HTTP	128,025	1,246,382
	DDoS HOIC		
	DDoS LOIC UDP		
Patator	FTP-Patator / FTP-BruteForce	13,843	156,668
	SSH-Patator / SSH-Bruteforce		
Web Based	Web Brute Force	2,180	928
	Web XSS		
	SQL Injection		
Botnet	Bot	1,956	282,310
Portscan	PortScan	158,804	–
Dropped	Infiltration	47	–
	Heartbleed		

Table 3.1: Attack Categories in CIC-IDS2017 and CIC-IDS2018 Datasets.

3.2.2 Data Splitting

Following the data preprocessing and feature harmonization stages, the CIC-IDS2017 dataset was partitioned into three distinct subsets: a training set, a validation set, and a testing set. The training set was used to train the ML models, while the validation set was used to tune the model parameters and select the best configuration. The testing set was kept separate and used only for the final evaluation, in order to measure the model performance on unseen data.

The dataset was divided to obtain a final distribution of 70% for training, 10% for validation, and 20% for testing. During the splitting process, a stratified sampling technique

(*stratify*) was applied. This option preserves the original class proportions in each subset, ensuring that both majority and minority attack classes are represented in the training, validation, and testing sets.

Table 3.2 summarizes the distribution ratios adopted for our subsets.

Dataset Subset	Percentage
Training Set	70%
Validation Set	10%
Testing Set	20%

Table 3.2: Dataset splitting Ratios.

3.2.3 Class Imbalance Handling

Network intrusion datasets are highly imbalanced, where some classes contain many more samples than others. This imbalance can bias the model toward majority classes and reduce its ability to detect rare attacks. To address this issue, balancing techniques were applied only to the training set for both binary and multiclass classification tasks. Random undersampling was used to reduce the size of majority classes, while the Synthetic Minority Over-sampling Technique (SMOTE) [55] was used to generate synthetic samples for minority classes. In both classification scenarios, the undersampling and oversampling thresholds were defined experimentally in order to obtain a suitable balance between reducing the dominance of majority classes and improving the representation of minority classes. The use of SMOTE was also motivated by a previous thesis, which showed its effectiveness compared to other oversampling methods on the same dataset.

Multiclass Classification Balancing

For the multiclass classification task, the original training data was highly imbalanced, as shown in the initial class distribution. Some classes contained a large number of samples, such as DoS Hulk, Portscan, and DDoS, while other classes, such as Web-Based and Botnet, had a very limited number of instances. To reduce this imbalance, we applied a balancing strategy based on both random undersampling and oversampling. Classes containing more than 90,000 samples were reduced to 90,000 using random undersampling, while classes

containing fewer than 90,000 samples were increased to the same number using oversampling. Thus, all attack classes were balanced to 90,000 samples.

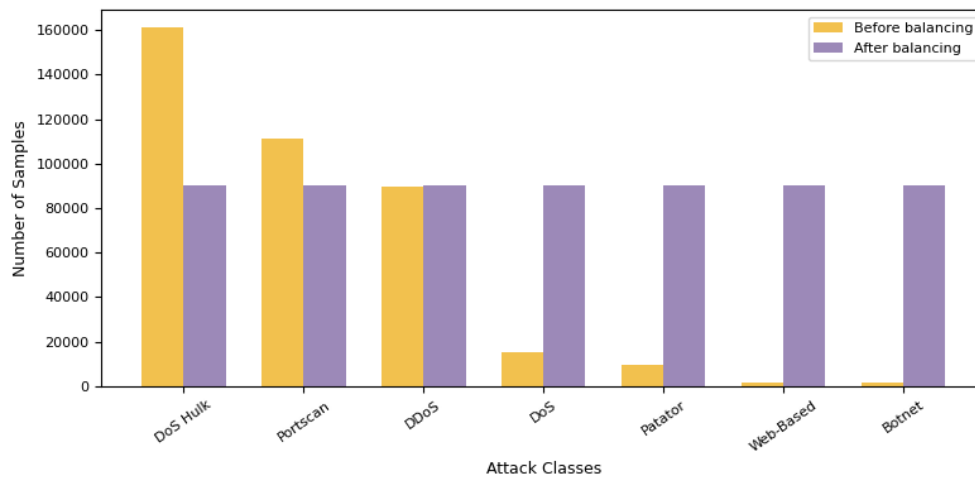


Figure 3.2: Class Balancing for Multiclass Classification.

Binary Classification Balancing

For the binary classification task, the balancing strategy was designed based on the two main classes: Benign and Attack. First, the total number of attack samples used for training was calculated, resulting in 389,564 instances. Since this number was lower than the desired class size, the Attack class was oversampled to reach 500,000 samples. At the same time, the Benign class, which contained a significantly larger number of instances, was reduced to 500,000 samples using random undersampling. As a result, both the Benign and Attack classes were balanced to the same size, ensuring a more equitable training distribution.

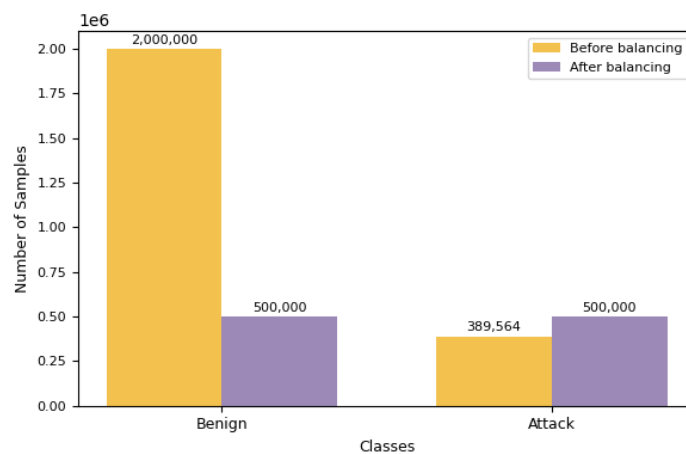


Figure 3.3: Class balancing for binary classification.

3.2.4 Dimensionality Reduction

To ensure a representative evaluation of different feature selection and reduction strategies, methods were selected from all main categories. More specifically, ANOVA and Mutual Information were selected from the filter-based methods, while Random Forest and Lasso L1 regression were adopted from the embedded methods. PCA was also used as a dimensionality reduction technique. In addition, two hybrid approaches were included to combine the strengths of different methods.

Since ANOVA, Mutual Information, Random Forest, Lasso L1 regression, and PCA were already explained in the previous chapter, this section focuses only on the composition of the two adopted hybrid techniques, as follows:

- **Dual-Stage Embedded Method (Decision Tree / Lasso L1):** This method was applied in two steps. First, a Decision Tree using the entropy criterion selected the top 65 features. Then, Lasso was applied to this reduced subset to obtain a final set of 50 features.
- **Two-Stage Hybrid Method (MI / Lasso L1):** In this approach, MI was first applied to select the top 60 most informative features. Then, Lasso L1 was applied to this reduced feature subset to obtain the final set of 50 features.

Table 3.3 summarizes the parameters configuration of each method.

Method	Key Parameters
ANOVA	score_func = f_classif
Lasso (L1 Regularization)	penalty='l1', solver='liblinear', random_state=42, max_iter=1000
Random Forest	n_estimators=100, random_state=42, n_jobs=-1
Dual-Stage Embedded (DT + Lasso)	criterion = 'entropy', penalty = 'l1', solver='liblinear', max_iter=1000, random_state=42
Two-Stage Hybrid (MI + Lasso)	penalty='l1', solver='liblinear', max_iter=1000
PCA (Dimensionality Reduction)	n_components = 35 , output_distribution = 'normal'

Table 3.3: Parameter Configuration of the Feature Selection Methods

3.2.5 Data Encoding

Categorical feature encoding is an important preprocessing step because most ML algorithms require numerical input data. In IDS datasets, some features may contain categorical values, which cannot be directly processed by learning models. Therefore, these values must be transformed into numerical representations before training. Several encoding techniques can be used for this purpose, such as One-Hot Encoding, Label Encoding, Ordinal Encoding, and Binary Encoding [56]. In our case, both CIC-IDS2017 and CSE-CIC-IDS2018 contain only one categorical feature, which corresponds to the traffic label. In this work, we adopted Label Encoding for both binary classification and multiclass classification tasks. This choice was made because Label Encoding provides a simple and efficient way to convert class labels into numerical values while keeping the dataset structure compact.

Classification Task	Class / Attack Category	Encoded Value
Binary Classification	Benign	0
	Attack	1
Multiclass Classification	Botnet	0
	DDoS	1
	DoS	2
	DoS Hulk	3
	Patator	4
	Portscan	5
	Web Based	6

Table 3.4: Label Encoding for Binary and Multiclass Classification

3.2.6 Features Normalization

Network traffic features often have different ranges and scales, which can affect the performance of ML algorithms. When some features have much larger values than others, the model may give them more importance, even if they are not more relevant. This can increase training time and reduce model stability. To solve this problem, feature normalization is applied to put the features on a comparable scale. This helps the model learn patterns more

effectively and improves the efficiency of the training process.[57]

Several feature scaling techniques can be used to reduce differences in feature ranges. Min-Max Scaling transforms feature values into a fixed interval, usually between 0 and 1, making all features share the same scale. Robust Scaling uses the median and the interquartile range, which makes it more suitable when the data contains outliers. Another useful method is the Quantile Transformer, which transforms feature values based on their quantile positions and maps them to a uniform or normal distribution. Overall, these techniques help make the data more consistent and support more stable model training.[58]

In this study, we adopted the *QuantileTransformer* [59] from the *Scikit-Learn* library for both binary and multiclass classification tasks. This choice was based on empirical experiments, where different scaling techniques were tested and compared.

3.3 Model Selection and Evaluation Strategy

After preprocessing the data, this section presents the selected training model and the evaluation strategy. It first introduces XGBoost as the adopted classifier, then describes the internal and external validation strategies, and finally defines the performance metrics used to assess the results.

3.3.1 Selection of the Learning Model

Several ML models can be used for intrusion detection, such as Decision Trees, Random Forest, Support Vector Machine, Logistic Regression, and boosting-based algorithms. In this study, we selected Extreme Gradient Boosting (XGBoost) [60] as the main classification model for both binary and multiclass classification tasks.

XGBoost is an advanced and highly efficient ML algorithm designed to deliver superior predictive performance, particularly when applied to large and complex datasets. It is built upon the principles of decision trees and employs a robust gradient boosting framework to iteratively enhance model accuracy by minimizing the loss function at each sequential step [60].

The choice of XGBoost is motivated by its strong accuracy, its ability to handle large and complex datasets, and its good performance in classification problems. These advantages make it suitable for IDS datasets, which contain many traffic records and different attack patterns. In addition, XGBoost includes regularization techniques that help reduce overfitting and improve generalization.

Table 3.5 presents the main hyperparameter used to configure the XGBoost model for both binary and multiclass classification tasks. The parameter `device='cuda'` was used to enable GPU acceleration, which helps reduce training time when working with large IDS datasets. The selected hyperparameter values were determined experimentally by testing several configurations and choosing the settings that provided the most stable and effective results.

Parameter	Binary Classification	Multiclass Classification
objective	binary:logistic	multi:softprob
n_estimators	300	300
max_depth	8	8
learning_rate	0.05	0.05
subsample	0.8	–
colsample_bytree	0.8	–
reg_alpha	–	1.0
reg_lambda	–	2.0
tree_method	hist	hist
device	cuda	cuda
early_stopping_rounds	20	15
random_state	42	42

Table 3.5: XGBoost Hyperparameter Configuration for Binary and Multiclass Classification

3.3.2 Evaluation Strategy

To ensure a rigorous and unbiased assessment of the XGBoost models, two distinct evaluation scenarios were adopted:

- **Internal Validation:** The model was tested against the remaining 20% unseen testing subset of the CICIDS2017 dataset to evaluate performance on data originating from the same distribution.
- **External Validation:** The model was exposed to the CICIDS2018 dataset. This cross-dataset evaluation strategy is crucial for verifying the model's robustness against

dataset shifts and its ability to detect cyber threats that were not present in the original training environment.

3.3.3 Evaluation Metrics

To rigorously assess the classification performance of the selected techniques, a standardized set of evaluation metrics was utilized. The foundation of these metrics is the Confusion Matrix, a contingency table that visualizes the performance of the algorithm by comparing predicted classifications against the actual ground truth . For a binary or multi-class classification scenario, the confusion matrix components are defined as follows [61]:

- **True Positives (TP):** Malicious flows correctly identified as attacks.
- **True Negatives (TN):** Benign flows correctly identified as normal traffic.
- **False Positives (FP):** Benign flows incorrectly flagged as attacks.
- **False Negatives (FN):** Malicious flows incorrectly classified as benign.

Based on these fundamental components, the following standard mathematical metrics were calculated to quantify the model's effectiveness:

Accuracy: The ratio of correctly predicted observations to the total observations. While intuitive, it can be misleading in highly imbalanced datasets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Precision: The ratio of correctly predicted positive observations to the total predicted positives. It measures the system's exactness and its ability to minimize false alarms.

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

Recall: The ratio of correctly predicted positive observations to all actual observations

in the attack class. It measures the system's sensitivity to detecting real threats.

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

F1-Score: The harmonic mean of Precision and Recall. It is highly useful when seeking an optimal balance between precision and recall, especially in the presence of uneven class distributions.

$$F1-Score = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad (3.4)$$

3.4 Tools and Software

The implementation of the selected feature selection techniques required a scalable software environment capable of handling large volumes of network traffic data and performing computationally intensive ML tasks. The experimental framework was developed using the Python programming language, due to its rich ecosystem of libraries for data preprocessing, ML, and cybersecurity analysis.

To manage computational resources efficiently, all experiments were conducted on Google Colab. This cloud-based platform provides access to GPU acceleration and sufficient memory resources, which are particularly useful when processing large IDS datasets such as CIC-IDS2017 and CSE-CIC-IDS2018. The use of GPU support helped reduce both data processing and model training time, while the available memory space facilitated the manipulation of large data files during preprocessing and experimentation.

3.5 Results

This section presents the experimental results obtained after applying the selected feature selection and dimensionality reduction techniques. The objective is to analyze how each technique affects the performance of the XGBoost classifier in both binary and multiclass classification tasks. For each feature selection technique, XGBoost was trained using different numbers of selected features: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, and 65 features.

The obtained results are first presented graphically to show the evolution of performance according to the number of selected features. Then, the best results are summarized in tables for both internal validation on CIC-IDS2017 and external validation on CSE-CIC-IDS2018. For each technique, the best feature subset was selected based on the external validation results, considering the configuration that provided the best balance between macro recall and accuracy.

For better readability and clarity of the presented results, the graphs illustrating the selected features obtained by the different feature selection techniques are provided in the Appendix.

3.5.1 Multiclass Classification Results

This subsection presents the results obtained for the multiclass classification task using the selected feature selection and reduction techniques.

- **Baseline Model - All Features**

Table 3.6 summarizes the performance metrics of XGBoost model using the complete feature set, before applying any feature selection technique. The model was evaluated through two strategies: internal validation on the CIC-IDS2017 test set and external validation on the independent CSE-CIC-IDS2018 dataset.

Attack Type	Internal Validation			External Validation		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Botnet	1.0000	1.0000	1.0000	1.0000	0.5066	0.6725
DDoS	1.0000	1.0000	1.0000	0.9998	0.9738	0.9866
DoS	0.9970	0.9988	0.9979	0.6922	0.6117	0.6495
DoS Hulk	0.9998	0.9996	0.9997	0.9580	0.9391	0.9485
Patator	0.9993	1.0000	0.9996	0.8881	1.0000	0.9407
Portscan	0.9998	0.9997	0.9997	-	-	-
Web-Based	0.9909	1.0000	0.9954	0.0033	0.6078	0.0066
Accuracy	99.97%			89.67%		
Macro Recall	0.9997			0.7731		

Table 3.6: Performance Report : (Baseline Model - All Features).

Furthermore, Figure 3.4 visually represents the model's misclassification through confusion matrices for both the internal and external evaluations.

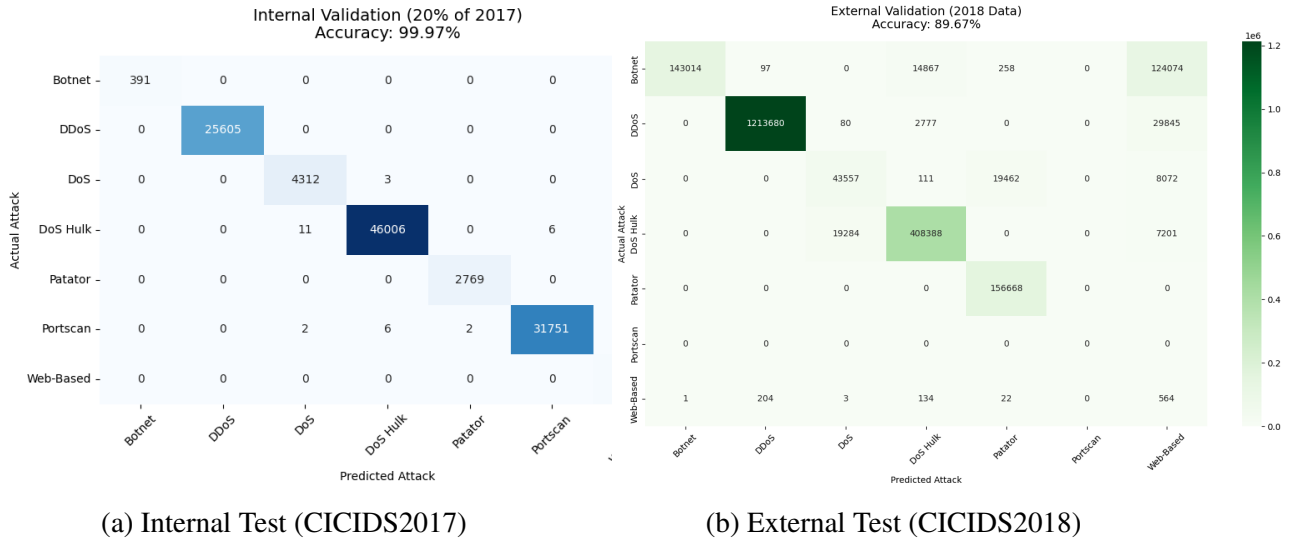


Figure 3.4: Multiclass Classification Confusion Matrices For the Baseline Model.

• Analysis of Variance (ANOVA)

From the performance curve shown in Figure 3.5, the ANOVA method achieved its best results when using the first 55 selected features. This configuration provided an external accuracy of **0.7086** and a macro recall of **0.6822** on CSE-CIC-IDS2018. Table 3.7 presents the detailed performance metrics for this selected feature subset.

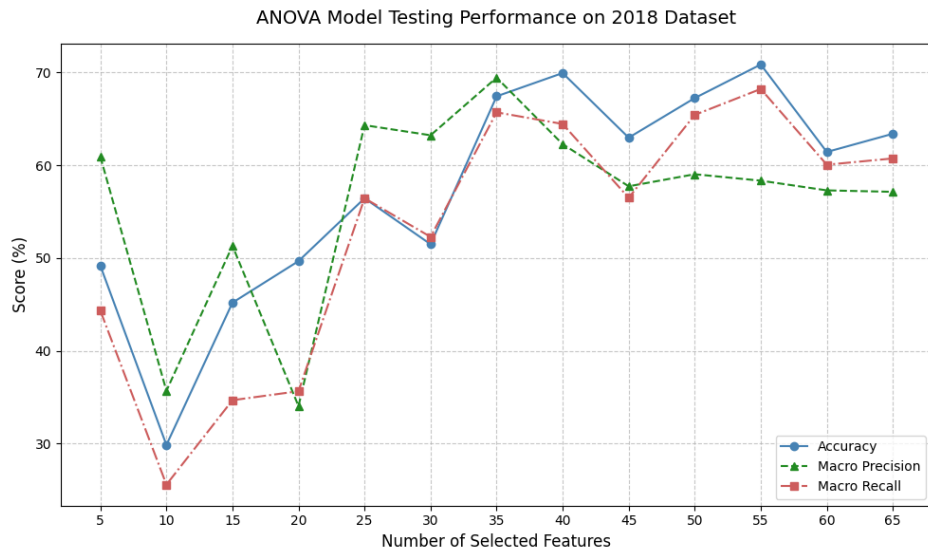
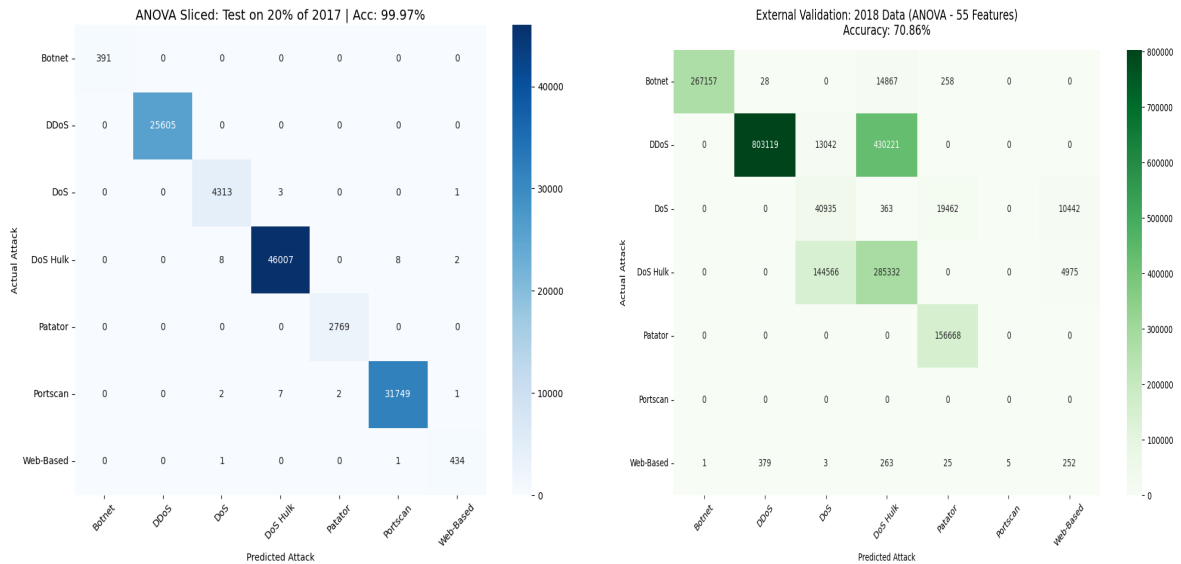


Figure 3.5: Performance Curve of XGBoost with Different Numbers of ANOVA Selected Features

Attack Type	Internal Validation			External Validation		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Botnet	1.0000	1.0000	1.0000	1.0000	0.9463	0.9724
DDoS	1.0000	1.0000	1.0000	0.9995	0.6444	0.7836
DoS	0.9975	0.9991	0.9983	0.2062	0.5749	0.3035
DoS Hulk	0.9998	0.9996	0.9997	0.3903	0.6561	0.4895
Patator	0.9993	1.0000	0.9996	0.8881	1.0000	0.9407
Portscan	0.9997	0.9996	0.9997	-	-	-
Web-Based	0.9909	0.9954	0.9931	0.0161	0.2716	0.0304
Accuracy	99.97%			70.86%		
Macro Recall	0.9991			0.6822		

Table 3.7: Performance Report : ANOVA (55 Features)

Furthermore, Figure 3.6 illustrates the confusion matrices for both evaluations using the selected features.



(a) Internal Test (ANOVA - 55 Features)

(b) External Test (ANOVA - 55 Features)

Figure 3.6: Confusion Matrices for ANOVA (55 Features)

3.5.2 Binary Classification Results

For the binary classification task, the objective is to evaluate the ability of the selected feature selection techniques to distinguish between benign and attack traffic. Since the internal validation on CIC-IDS2017 produced nearly perfect results for all experiments, only the internal validation results of the baseline model on CIC-IDS2017 are presented.

To provide a more detailed external evaluation, the CSE-CIC-IDS2018 dataset was divided into 13 separate files. Each file contains a portion of benign traffic combined with one specific attack type. This organization allows us to identify more clearly where the model maintains good detection performance and where performance drops occur. In addition, all files were evaluated using two decision thresholds, 0.5 and 0.3, in order to analyze the effect of threshold adjustment on detection performance. A Kernel Density Estimation (KDE) plot was also used to better illustrate the performance differences between the two thresholds.

Due to the volume of the thesis, only one selected example of the confusion matrices obtained from these 13 files and one KDE plot is presented, while the complete detailed results are provided in the implementation code.

In this binary classification evaluation, three feature selection techniques were tested: dual-stage (DT+Lasso L1), Mutual Information, and Random Forest.

– Baseline Model

Table 3.8 presents the baseline binary classification results obtained on CIC-IDS2017 using all features, while Table 3.9 reports the corresponding external evaluation results on CSE-CIC-IDS2018 for the same baseline model.

Class	Precision	Recall	F1-score
Benign	0.9996	0.9986	0.9991
Attack	0.9987	0.9996	0.9991
Accuracy			0.9991

Table 3.8: Binary Classification Performance Report-CIC-IDS2017 (Baseline Model)

Traffic Type	Recall (Th. = 0.5)	Recall (Th. = 0.3)
1. Benign	0.9996	0.9997
2. DoS Hulk	0.0135	0.1018
3. DoS SlowHTTPTest	1.0000	1.0000
4. DoS GoldenEye	0.2030	0.4669
5. DoS Slowloris	0.9845	0.9845
6. FTP BruteForce	1.0000	1.0000
7. SSH BruteForce	0.1987	0.1988
8. BruteForce Web	0.0065	0.0065
9. BruteForce XSS	0.0043	0.0043
10. SQL Injection	0.0690	0.0690
11. Botnet	0.0001	0.0004
12. DDoS LOIC HTTP	0.0012	0.0221
13. DDoS HOIC	0.0000	0.0000
14. DDoS LOIC UDP	0.0000	0.0000
Global Attack Recall	0.2078	0.2365

Table 3.9: Binary Classification Performance Report on CSE-CIC-IDS2018 (Baseline Model)

– Filter method : Mutual Information (MI)

Based on the performance curve presented in Figure 3.7, the Mutual Information method obtained its best performance when using 10 features. This feature subset showed the most suitable trade-off between accuracy and recall. The detailed performance results for this configuration are reported in Table 3.10.

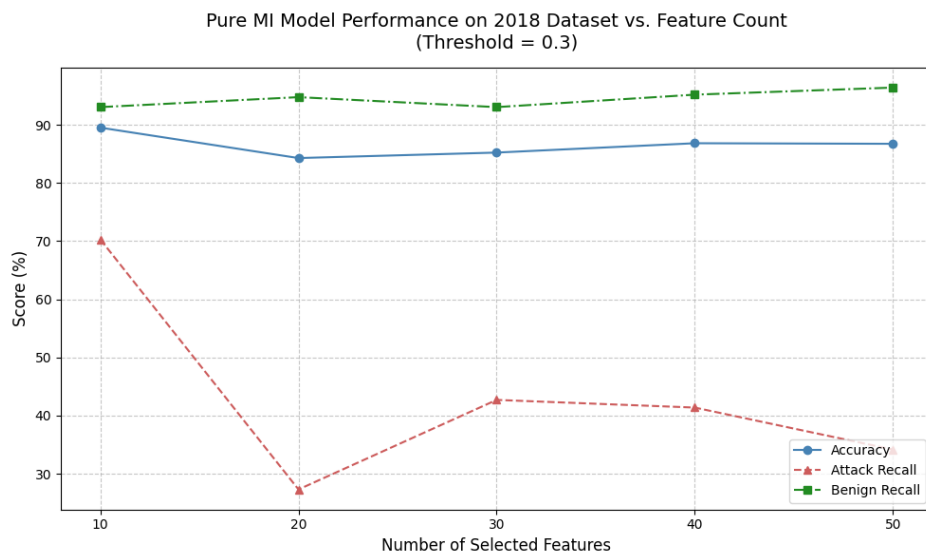


Figure 3.7: Performance Curve of Mutual Information.

Traffic Type	Recall (Th. = 0.5)	Recall (Th. = 0.3)
1. Benign	99.98	99.98
2. DoS Hulk	0.9413	0.9413
3. DoS SlowHTTPTest	0.0000	1.0000
4. DoS GoldenEye	0.3602	0.3628
5. DoS Slowloris	0.9882	0.9953
6. FTP BruteForce	0.0000	1.0000
7. SSH Bruteforce	0.9999	1.0000
8. BruteForce Web	0.6334	0.6367
9. BruteForce XSS	0.4913	0.4913
10. SQL Injection	0.7011	0.7471
11. Botnet	0.4907	0.4907
12. DDoS LOIC HTTP	0.4965	0.4965
13. DDoS HOIC	0.7550	0.7550
14. DDoS LOIC UDP	0.0000	0.0000
Global Attack Recall	0.6101	0.7009

Table 3.10: Binary Classification Performance Report on CSE-CIC-IDS2018 (Mutual Information/ 10 features)

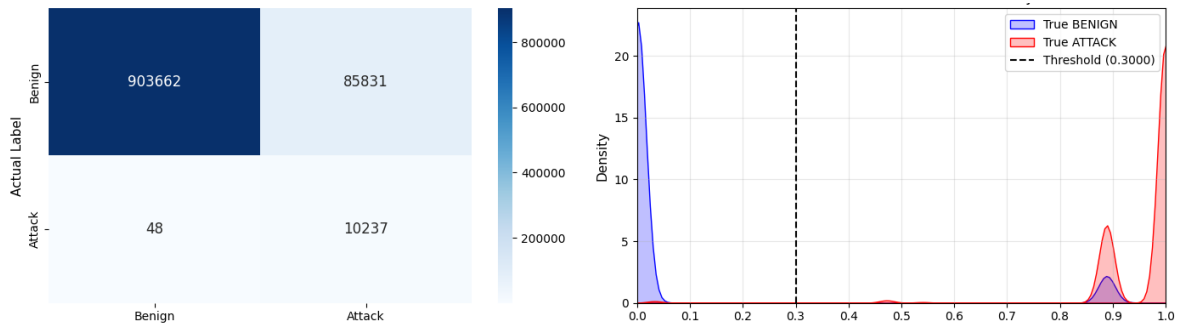


Figure 3.8: confusion matrix and KDE plot for DoS-Slowloris using MI (Threshold = 0.3)

Figure 3.8 presents the confusion matrix and the KDE plot for the **DoS-Slowloris** attack.

– Embedded method: Random Forest (RF)

Based on the performance curve presented in Figure 3.9, the embedded random forest method achieved its best performance when using 40 features. The detailed performance results for this configuration are reported in Table 3.11.

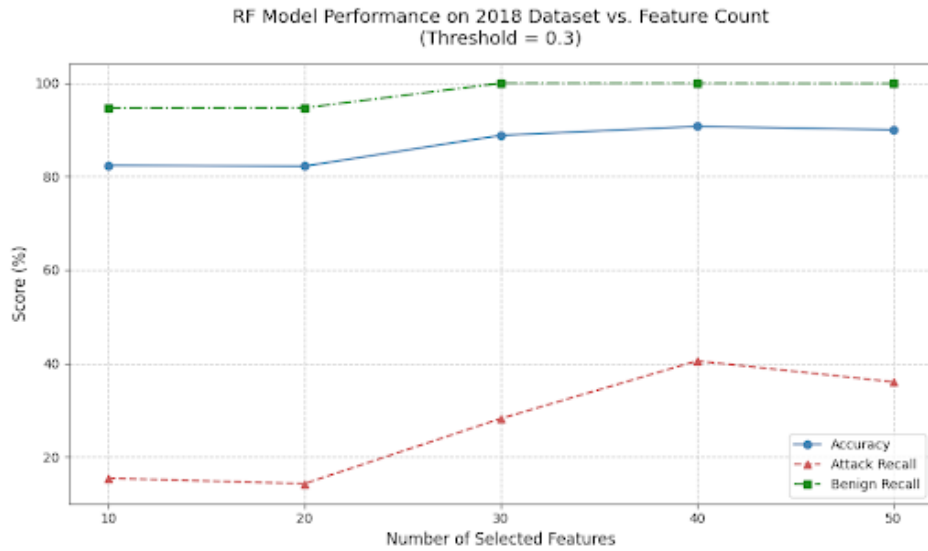


Figure 3.9: Performance Curve of Random Forest

Traffic Type	Recall (Th. = 0.5)	Recall (Th. = 0.3)
1. Benign	99.95	99.93
2. DoS Hulk	0.8789	0.9363
3. DoS SlowHTTPTest	1.0000	1.0000
4. DoS GoldenEye	0.4309	0.4713
5. DoS Slowloris	0.9844	0.9845
6. FTP BruteForce	1.0000	1.0000
7. SSH BruteForce	0.1443	0.1995
8. BruteForce Web	0.0000	0.0049
9. BruteForce XSS	0.0043	0.0043
10. SQL Injection	0.0345	0.0690
11. Botnet	0.0047	0.2819
12. DDoS LOIC HTTP	0.5018	0.5021
13. DDoS HOIC	0.0000	0.0000
14. DDoS LOIC UDP	0.0000	0.0000
Global Attack Recall	0.3834	0.4001

Table 3.11: Binary Classification Performance Report on CSE-CIC-IDS2018 (Random Forest/ 40 feature)

Figure 3.10 presents the Confusion Matrix and the KDE plot for the **Dos hulk** attack.

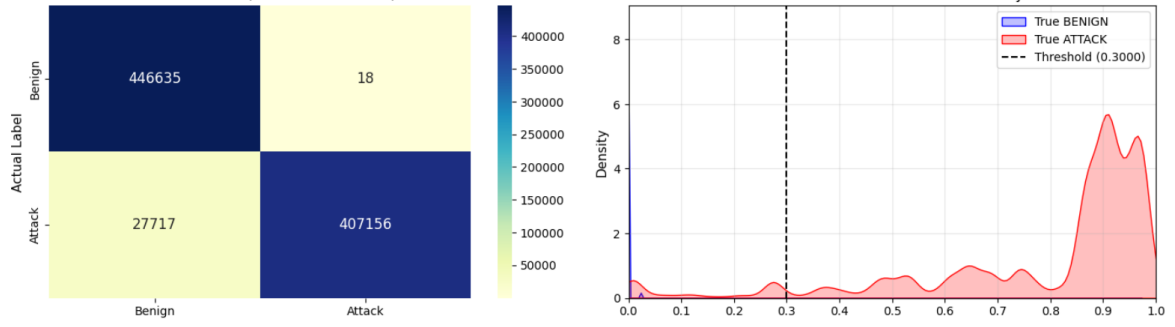


Figure 3.10: Confusion Matrix and KDE Probability Distribution for Dos hulk using RF (Threshold = 0.3)

3.6 Comparative Discussion

After presenting the results of all feature selection and reduction techniques for both binary and multiclass classification tasks, this section provides a comparative discussion of the obtained findings. The objective is to analyze the differences between the tested methods and identify which techniques provide the best balance between detection performance, feature number reduction, and generalization ability.

3.7 Multi-Classification Comparison

Figure 3.11 compares the recall obtained by the different feature selection techniques under internal validation on CIC-IDS2017. Overall, all feature selection methods achieved very high recall values across almost all attack classes, showing that the selected feature subsets preserved the most important information needed for classification. ANOVA, Lasso L1, Random Forest, Dual-Stage (DT+Lasso L1), and two-stage Hybrid (MI+ Lasso L1) produced nearly perfect results for Botnet, DDoS, Patator, and Portscan. For DoS and DoS Hulk, only very small differences can be observed between the methods, while the Web-Based class shows slightly more variation. However, the overall results remain highly stable, indicating that the evaluated feature selection techniques are effective in reducing the feature space without significantly affecting internal detection performance.

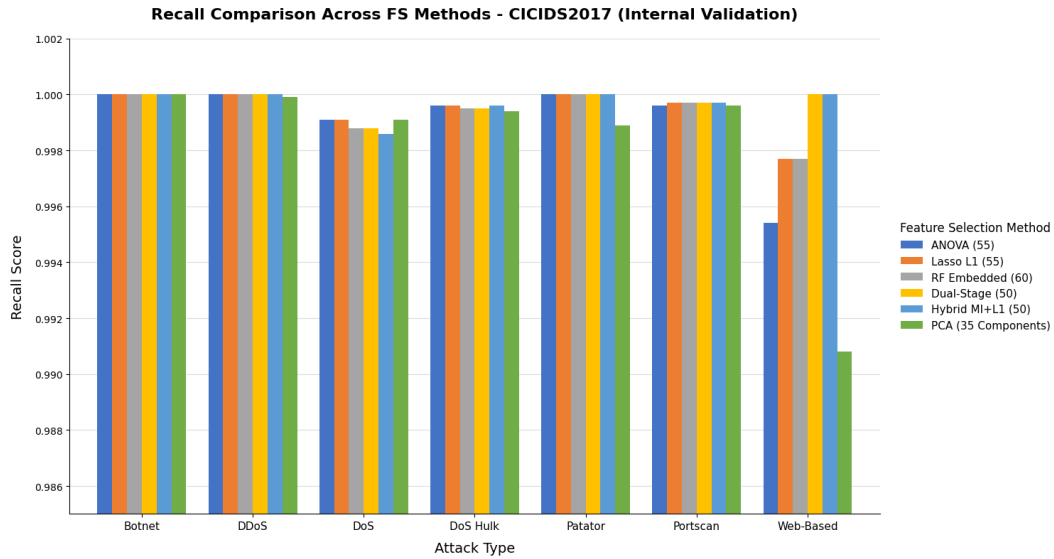


Figure 3.11: Recall Comparison Across FS Methods – CICIDS2017

Figure 3.12 presents the external evaluation results on CSE-CIC-IDS2018. Unlike the internal validation results, the performance varies clearly between the methods, showing the importance of evaluating generalization on an independent dataset. The best performance was obtained by the Dual-Stage (DT+Lasso L1) method, achieving the highest accuracy of 95.63% and macro recall of 84.68%. The Two-stage hybrid (MI/Lasso L1) method also showed strong results, with 94.47% accuracy and 80.81% macro recall. In comparison, the single feature selection methods achieved lower performance, especially ANOVA and Random Forest, while PCA obtained the weakest results. These findings indicate that hybrid feature selection methods provide better external generalization than individual feature reduction techniques.

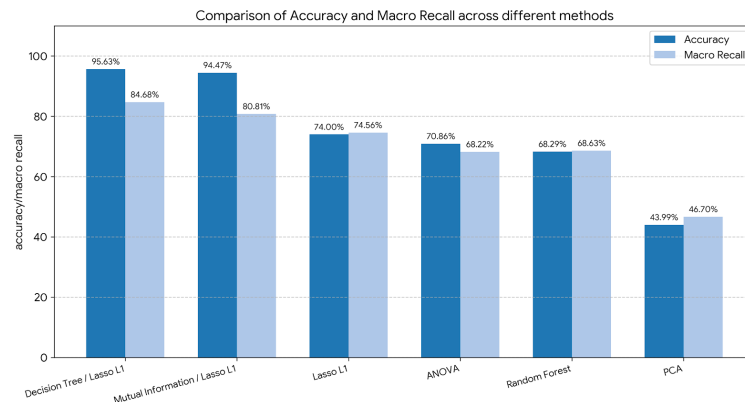


Figure 3.12: Multiclass Classification Comparison of Accuracy and Macro Recall Across Different Feature Selection Techniques

Figure 3.13 presents a detailed per-class recall comparison on CSE-CIC-IDS2018. The results show that the detection performance differs from one attack class to another. Most methods achieve strong recall for Botnet, DoS Hulk, and Patator, indicating good generalization for these attack types. However, lower recall values are observed for Web-Based attacks, showing that this class remains more difficult to detect across datasets. The hybrid methods, especially Dual-Stage (DT+Lasso L1) and Hybrid (MI+Lasso L1), generally provide more stable recall across several attack categories compared to simple methods.

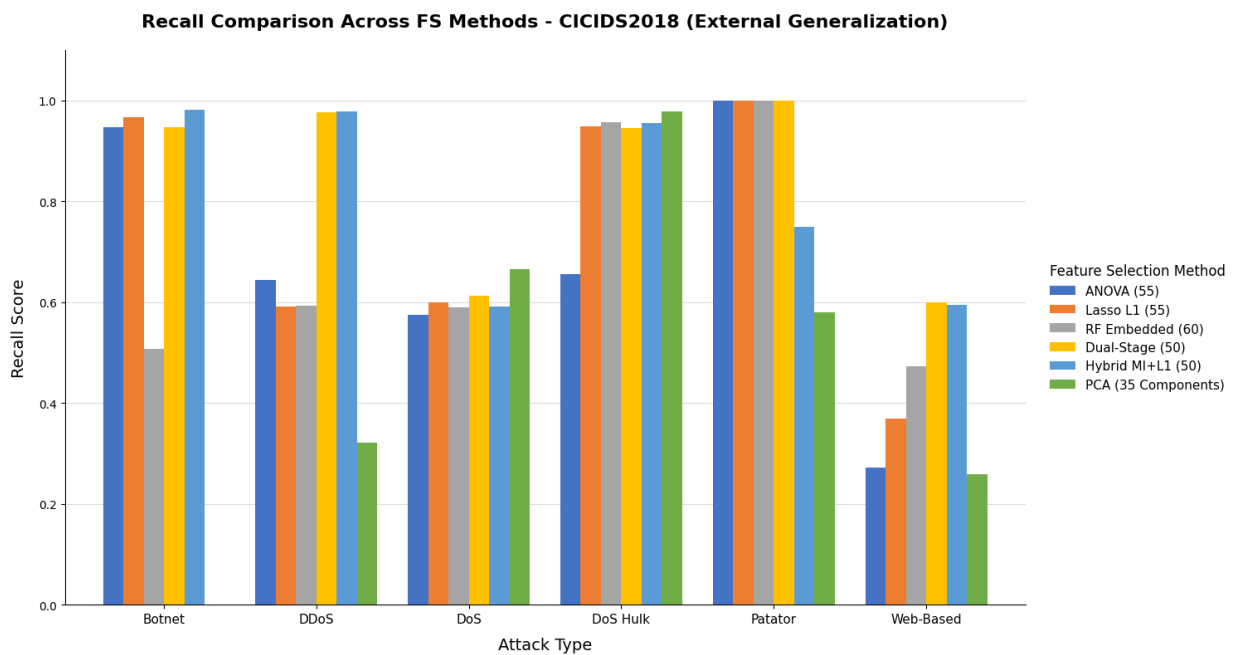


Figure 3.13: Per-Class Recall Comparison Across Feature Selection Techniques for Multi-class Classification-CICIDS2018

Table 3.12 presents the number of common features shared between each pair of feature selection techniques. The results show a high overlap between the selected subsets, indicating that most methods identify similar important features. The strongest intersection is observed between Random Forest and Lasso L1, with 52 common features, followed by several pairs sharing 48 to 50 features. The lowest overlap is observed between dual-stage(DT + Lasso) and MI + Lasso, with 43 common features. Overall, this matrix confirms that the selected methods are generally consistent in identifying relevant features. However, even small differences in the selected feature subsets can affect the final classification results, especially under external validation.

Method	DT + Lasso	Random Forest	Lasso L1	ANOVA	MI + Lasso
DT + Lasso (50)	-	48	49	44	43
Random Forest(60)	48	-	52	50	48
Lasso L1 (55)	49	52	-	48	48
ANOVA (55)	44	50	48	-	44
MI + Lasso (50)	43	48	48	44	-

Table 3.12: Pairwise Feature Intersection Matrix across Feature Selection Techniques.

3.8 Binary Classification Comparison

Figure 3.14 compares the global attack recall obtained on CSE-CIC-IDS2018 in the binary classification task. The results show that Mutual Information achieved the best performance, with a global attack recall of **70.09%**, clearly outperforming Random Forest, dual-stage (DT + Lasso L1), and the baseline model.

An important advantage of Mutual Information is that it achieved this result using only 10 selected features , while Random Forest required 40 selected features to obtain its best recall of **40.01%**. dual-stage (DT + Lasso L1) achieved **28.85%**, while the baseline model obtained only **23.65%**. Overall, these results show that Mutual Information provides the best trade-off between attack detection performance and feature reduction.

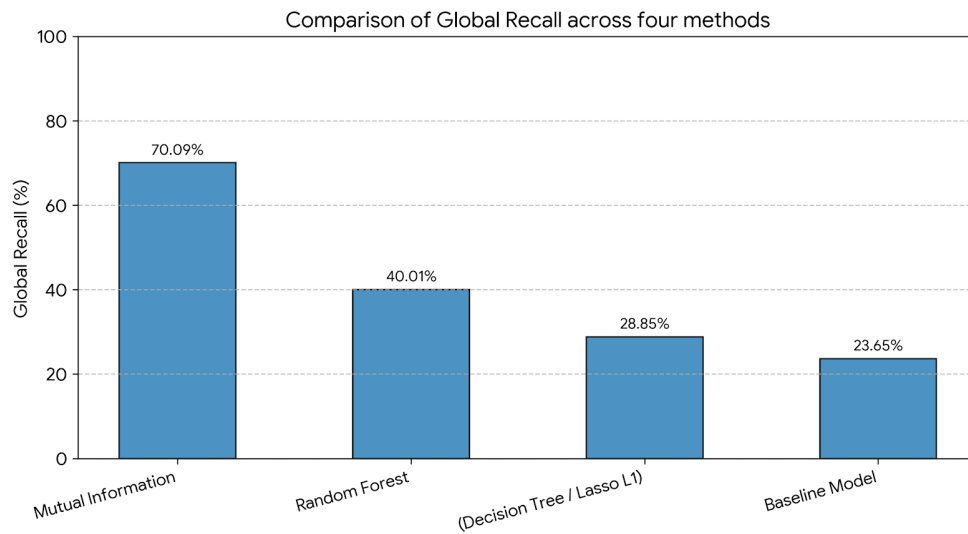


Figure 3.14: Comparison of Global Recall Across Features selection techniques.

The per-attack recall results reported in the above tables confirm that, compared with the dual-stage approach (DT + Lasso L1) and Random Forest, Mutual Information improves the detection of several attack types, particularly those that were weakly detected by the other methods. For instance, Mutual Information achieves higher recall for BruteForce Web, BruteForce XSS, SQL Injection, Botnet, and DDoS HOIC, which remain almost undetectable or poorly detected using dual-stage (DT + Lasso L1) and Random Forest as shown in Figure 3.15. In addition, it maintains strong recall for attacks such as DoS Hulk, DoS Slowloris, FTP BruteForce, and SSH Bruteforce. These results indicate that Mutual Information not only improves the overall attack recall, but also enhances the detection performance across a wider range of individual attack categories.

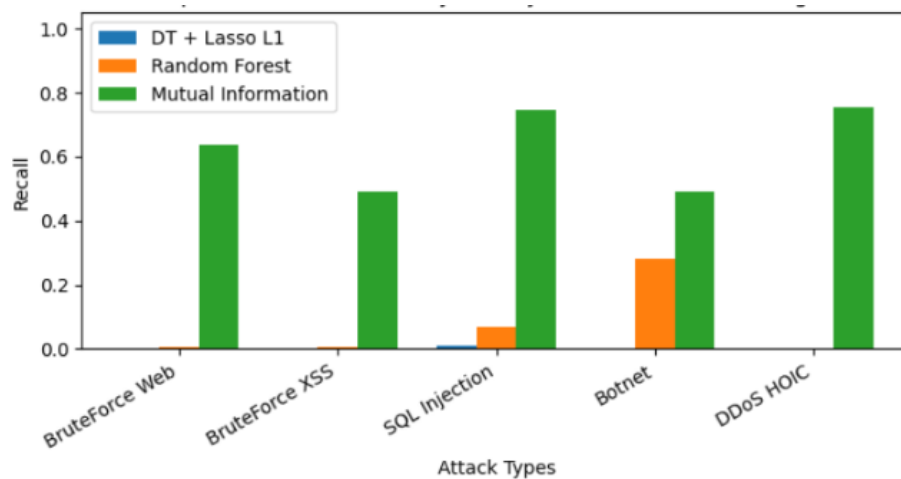


Figure 3.15: Recall Improvement of Weakly Detected Attacks Using Mutual Information

3.9 Conclusion

This chapter presented the methodology and experimental results of several feature selection and reduction techniques for IDS classification. The tested methods included ANOVA, Mutual Information, Lasso L1, Random Forest, Dual-Stage Embedded (DT+Lasso L1), Tow-stage Hybrid (MI + Lasso), and PCA.

The results showed that internal validation on CIC-IDS2017 produced high performance for most methods. However, external validation on CSE-CIC-IDS2018 revealed clear differences between the techniques and highlighted the importance of cross-dataset evaluation.

Overall, the findings confirm that feature selection can reduce dimensionality while improving detection performance and generalization. The best method depends on the classification task, with Dual-Stage Embedded selection being more effective for multiclass classification and Mutual Information being more suitable for binary attack detection.

General Conclusion

This research investigates the role of dimensionality reduction in enhancing ML-based IDS. Using CIC-IDS2017 and CIC-IDS2018 as benchmark datasets, the study analyzes the difficulties associated with high-dimensional feature spaces and assesses how different dimensionality reduction strategies can help simplify the data and maintain reliable detection performance.

In the experimental phase, several dimensionality reduction techniques were evaluated within a comprehensive ML-based IDS pipeline, including ANOVA, Mutual Information, Random Forest, Lasso L1 regularization, hybrid approaches, and PCA. In the multi-class classification task, the results showed that hybrid techniques provided the most effective performance on both CIC-IDS2017 and CIC-IDS2018 datasets. These methods were able to preserve the most relevant features for distinguishing between different attack categories while reducing the complexity of the feature space. In the binary classification task, Mutual Information technique achieved the best results, demonstrating perfect detection performance on CIC-IDS2017 and remarkable detection performance across several attack types in CIC-IDS2018. Overall, these findings confirm that dimensionality reduction can significantly support IDS models by improving detection efficiency, reducing unnecessary features, and maintaining strong classification performance in both multi-class and binary scenarios.

Future research directions could extend this study by exploring additional dimensionality reduction techniques and conducting a deeper investigation of feature reduction methods, particularly PCA and other related approaches. Moreover, further experiments should be carried out using a wider range of IDS datasets during both the training and testing phases in order to evaluate the stability and effectiveness of the selected techniques under different network conditions. Since this work mainly relies on XGBoost as the evaluation model, future studies could also consider other ML models to

provide a broader comparison and better assess the impact of dimensionality reduction on different classification algorithms.

Bibliography

- [1] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). NIST Special Publication 800-94, National Institute of Standards and Technology, Gaithersburg, MD, 2007.
- [2] K. Munivara Prasad, A. Rama Mohan Reddy, and K. Venugopal Rao. Dos and ddos attacks: Defense, detection and traceback mechanisms—a survey. *Global Journal of Computer Science and Technology*, 14(7), 2014.
- [3] Subodh Gangan. A review of man-in-the-middle attacks. *arXiv preprint arXiv:1504.02115*, 2015.
- [4] Ömer Aslan Aslan and Refik Samet. A comprehensive review on malware detection approaches. *IEEE Access*, 8:6249–6271, 2020.
- [5] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: A literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.
- [6] William G. J. Halfond, Jeremy Viegas, and Alessandro Orso. A classification of sql-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, pages 13–15. IEEE, 2006.
- [7] Dinei Florêncio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web*, pages 657–666. ACM, 2007.
- [8] Portnox. What is intrusion detection system? <https://www.portnox.com/cybersecurity-101/general-security/intrusion-detection-system/>, 2024. Accessed: 2026-07-01.

- [9] Splunk. Intrusion detection systems (ids): Definition, types, purpose. https://www.splunk.com/en_us/blog/learn/ids-intrusion-detection-systems.html, 2024. Accessed : 2026-07-01.
- [10] Stamus Networks. Ids detection types. <https://www.stamus-networks.com/ids-detection-types>, 2024. Accessed: 2026-07-01.
- [11] OWASP Foundation. Intrusion detection. https://owasp.org/www-community/controls/Intrusion_detection. Accessed : 2026-07-01.
- [12] DataGuard. How intrusion detection systems help identify cyber threats. <https://www.dataguard.com/blog/how-intrusion-detection-systems-help-identify-cyber-threats/>, 2024. Accessed: 2026-07-01.
- [13] Group8. Intrusion detection: Key challenges and their solutions. <https://group8.co/intrusion-detection-key-challenges-and-their-solutions.html>, 2024. Accessed: 2026-07-01.
- [14] Luís Dias and Miguel Correia. Big data analytics for intrusion detection: An overview. In *Security and Privacy From a Legal, Ethical, and Technical Perspective*, pages 325–349. IGI Global, 2020.
- [15] Pietro Spadaccino and Francesca Cuomo. Intrusion detection systems for iot: Opportunities and challenges offered by edge computing and machine learning. *arXiv preprint arXiv:2012.01174*, 2020.
- [16] Nicholas Kolokotronis, Sotirios Brotsis, Georgios Germanos, Costas Vassilakis, and Stavros Shiaele. On blockchain architectures for trust-based collaborative intrusion detection. *arXiv preprint arXiv:2109.03635*, 2021.
- [17] Mohammad Rabbani, Yu-Lun Wang, Reza Khoshkangini, Hamid Jelodar, Rui Zhao, and Peng Hu. A review on machine learning approaches for network anomaly detection. *Computers, Materials & Continua*, 68(1):797–824, 2021.
- [18] IBM. What is machine learning? <https://www.ibm.com/think/topics/machine-learning>, 2024. Accessed: 2026-07-01.

- [19] IBM. Five machine learning types to know. <https://www.ibm.com/think/topics/machine-learning-types>, 2024. Accessed: 2026-07-01.
- [20] GeeksforGeeks. Types of machine learning. <https://www.geeksforgeeks.org/machine-learning/types-of-machine-learning/>, 2026. Accessed: 2026-07-01.
- [21] Melad Mohammed Issa, Mohammad Aljanabi, and Hassan M. Muhaldeen. Systematic literature review on intrusion detection systems: Research trends, algorithms, methods, datasets, and limitations. *Journal of Intelligent Systems*, 33(1), 2024.
- [22] Asadullah Momand, Sana Ullah Jan, and Naeem Ramzan. A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy. *Journal of Sensors*, 2023, 2023.
- [23] IBM. User and entity behavior analytics with ibm qradar siem. <https://www.ibm.com/products/qradar-siem/user-entity-behavior-analytics>, 2026. Accessed: 2026-07-01.
- [24] Cisco. Cisco secure network analytics. <https://www.cisco.com/site/in/en/products/security/security-analytics/secure-network-analytics/index.html>, 2026. Accessed: 2026-07-01.
- [25] Microsoft. Next-generation protection overview. <https://learn.microsoft.com/en-us/defender-endpoint/next-generation-protection>, 2026. Accessed: 2026-07-01.
- [26] Patrik Goldschmidt and Daniela Chudá. Network intrusion datasets: A survey, limitations, and recommendations. *Computers & Security*, 156:104510, 2025.
- [27] Vijayalakshmi Shanmugam et al. Addressing class imbalance in intrusion detection: A comprehensive evaluation of machine learning approaches. *Electronics*, 14(1):69, 2024.

- [28] A comprehensive review of dimensionality reduction techniques for real time network intrusion detection with applications in cybersecurity. Technical report, Core.ac.uk, 2024.
- [29] Sabrine Ennaji, Fabio De Gaspari, Dorjan Hitaj, Alicia K/bidi, and Luigi Vincenzo Mancini. Adversarial challenges in network intrusion detection systems: Research insights and future prospects. *arXiv preprint*, 2025.
- [30] Microsoft Azure. Model interpretability - azure machine learning. Documentation technique, 2026.
- [31] Patrik Goldschmidt and Daniela Chudá. Network intrusion datasets: A survey, limitations, and recommendations. *Computers & Security*, 156:104510, 2025.
- [32] Abhishek Divekar, Meet Parekh, Vaibhav Savla, Rudra Mishra, and Mahesh Shirole. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. *arXiv preprint*, 2018.
- [33] Mohamed Amine Agalit et al. Optimization of intrusion detection with deep learning: A study based on the kdd cup 99 database. *International Journal of Safety and Security Engineering*, 14(4):1029–1038, 2024. Disponible en ligne le 30 août 2024.
- [34] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–6. IEEE, 2009.
- [35] Nour Moustafa and Jill Slay. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6. IEEE, 2015.
- [36] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In

Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), pages 108–116. SCITEPRESS, 2018.

- [37] Wissam Zoued and Ahlam Bouanane. Analyzing the impact of data augmentation for enhancing machine learning models in the context of intrusion detection systems. Master’s thesis, Abdelhafid Boussouf Mila University Centre, Institute of Mathematics and Computer Sciences, Department of Computer Sciences, Mila, Algeria, 2025. Specialty: Artificial Intelligence and its Applications.
- [38] Dehua Peng et al. Interpreting the curse of dimensionality from distance concentration and manifold effect. *arXiv preprint*, 2024. Dernière révision en mars 2025.
- [39] Dure Ali et al. A comprehensive review of high-dimensional data: Challenges and opportunities. *Applied Soft Computing*, 174:112789, 2025.
- [40] IEEE Xplore. A dual-strategy framework for cyber threat detection in imbalanced, high-dimensional data across heterogeneous networks. In *IEEE Transactions on ...* IEEE, 2025. DOI: 10.1109/... (à compléter).
- [41] Fred Hutch Data Science Lab. Model complexity and overfitting. Machine Learning Course Materials, 2025. Consulté le 1er juillet 2026.
- [42] Deepthi Ramesh et al. Interpreting black-box machine learning models for high dimensional datasets. *arXiv preprint*, 2022. Sous presse dans une revue en 2025.
- [43] IBM. What is feature selection? <https://www.ibm.com/think/topics/feature-selection>, 2024. Accessed: 2026-07-02.
- [44] DeepAI. Feature reduction. <https://deepai.org/machine-learning-glossary-and-terms/feature-reduction>. Accessed: 2026-07-02.
- [45] Yuepeng Chen, Weiping Ding, Qichong Hua, Jiaru Yang, Tianyi Zhou, and Fan Fu. A survey on feature selection techniques for biomedical data: Methods and applications. *Neurocomputing*, 670:132534, 2026.

- [46] Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014.
- [47] M.O. Arowolo, S.O. Abdulsalam, Y.K. Saheed, and M.D. Salawu. A feature selection based on one-way-anova for microarray data classification. ..., 2016. Consulté via Kwara State University.
- [48] GeeksforGeeks. Wrapper methods in feature selection. <https://www.geeksforgeeks.org/machine-learning/wrapper-methods-feature-selection/>, 2025. Accessed: 2026-07-02.
- [49] CORE. A systematic literature review: Recursive feature elimination algorithms. *CORE Repository*, 2024. Consulté le 1er juillet 2026.
- [50] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
- [51] Zhengze Zhou and Giles Hooker. Unbiased measurement of feature importance in tree-based methods. *ACM Transactions on Knowledge Discovery from Data*, 15(2):1–21, 2020.
- [52] IEEE Access. Comprehensive review of hybrid feature selection methods for microarray-based cancer detection. *IEEE Access*, 13:78739–78768, 2025.
- [53] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, 2nd edition, 2002.
- [54] Scikit-learn. Linear discriminant analysis. https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html. 2026 – 07 – 02.
- [55] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

- [56] GeeksforGeeks. Categorical data encoding techniques in machine learning. <https://www.geeksforgeeks.org/machine-learning/categorical-data-encoding-techniques-in-machine-learning/>, 2025. Accessed: 2026-07-02.
- [57] Scikit-learn. Importance of feature scaling. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html. Accessed : 2026 – 07 – 02.
- [58] Scikit-learn. Preprocessing data. <https://scikit-learn.org/stable/modules/preprocessing.html>. Accessed: 2026-07-02.
- [59] Scikit-learn. Quantiletransformer. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html>. Accessed: 2026-07-02.
- [60] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.
- [61] Google Developers. Classification: Accuracy, recall, precision, and related metrics. <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall>. Accessed: 2026-07-02.