الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and Popular Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

Abdelhafid Boussouf Mila University Centre
Institute of Mathematics and Computer Science
Department of Computer Science

Specialty: Artificial Intelligence and its Applications.

Master's Thesis

Applying Machine Learning and Deep Learning Techniques for Intrusion Detection Systems

Presented by:

- **Bellout Chaima**
- **Belattar Mohammed Lamine**

Supported by the jury:

- **President: Mr.SELMANE Samir** **Rank: M.A.A**
- **Examiner: Mr. DIB Abderrahim** **Rank: M.A.A**
- **Supervisor: Mr.BENCHEIKH LEHOCINE Madjed** **Rank: M.C.B**

Academic year: 2023/2024


بسم الله الرحمن الرحيم

Acknowledgement

praise be to Allah, who granted us success and facilitated the completion of this Master's thesis.

We extend our deepest gratitude to our supervisor, **Mr.BENCHIEKH EL HOCINE Madjed**, for his invaluable guidance, patience in addressing all our inquiries, and continuous support, which greatly contributed to the successful completion and presentation of this work in its current form.

Our thanks also go to the members of the discussion committee, **Mr.SELMANE Samir** and **Mr.DIB Abderrahim**, for their kind acceptance to review this work and for providing constructive feedback that will enhance our skills and future research endeavors.

We are deeply grateful to Dr.Faiza Medjek, Pr.Imed Romdhani, Phd.Vegim Bytyqi, and Phd.Sara Koko for their unwavering assistance and guidance at various stages of our academic journey. Their support and encouragement played a crucial role in helping us reach this milestone.

We would also like to express our appreciation to the Canadian Institute for Cybersecurity, headed by Pr. Ali A. Ghorbani, for their support and cooperation with us.

Heartfelt thanks are also extended to our families, who have been our unwavering support system throughout every stage of our lives, in times of success and challenge alike. Your constant encouragement and support have been our backbone.

We would also like to thank our professors, friends, and colleagues, who were companions and supporters throughout our academic journey.

Lastly, we ask Allah to accept this work sincerely for His sake and to grant us and all of you continued success and prosperity. May this work serve as a source of inspiration for future students, God willing

إهداء

لم تكن الرحلة قصيرة ولا الطريق محفوفًا بالتسهيلات، لكنني قطعتها، فالحمد لله الذي يسر البدايات وبلغنا النهايات بفضله وكرمه.

بكل حب أهدي ثمرة نجاحي وتخرجي إلى النور الذي أنار دربي والسراج الذي لا ينطفئ نوره.. إلى من أحمل اسمه بكل فخر.. لطالما عاهدته بهذا النجاح ها أنا وفيت وعدي وأهديته إليك والدي العزيز **يحي**.

إلى من علمتني الأخلاق قبل الحروف إلى الجسر الصاعد بي إلى الجنة.. إلى اليد الخفية التي أزالت عن طريقي الأشواك.. ومن تحملت كل لحظة ألم مررت بها وساندتني عند ضعفي.. إلى التي بذلت جهد أنفاسها من أجل أن أصبح في يوم من الأيام مهندسة.. ها أنا أفي بوعدي لكي والدتي الحبيبة **فاطمة**.
إلى أخوتي الغاليين على قلبي **هاجر** وأطفالها وإلى **ريان** وابنها **عابد الليث** وأخي **تاج الدين** وأخي **هيثم**..
أنتم دائماً مصدر الفخر والإلهام بالنسبة لي.

إلى جدتي الحبيبة وأمي الثانية.. التي كانت حنانها ودعاءها نبراسًا يضيء طريقي..
إلى روح جدي **عامر**.. الذي ستظل ذكراه تعيش في قلبي وتدفعني لتحقيق المزيد من النجاح..
وأخيرا.. أهدي هذا النجاح لنفسي الطموحة.. إذ ابتدأت بطموح وعزيمة وانتهت بنجاح.
لكم جميعًا.. أهدي ثمرة هذا الجهد...

الشيماء

إهداء

الحمد لله الذي بنعمته تتم الصالحات، وبفضله تكتمل الغايات. فلولا توفيقه، لما وصلتُ إلى ما أنا عليه اليوم. أهدي هذا النجاح المتواضع لكل من كان سندًا وعونًا في رحلتي العلمية والحياتية.

إلى والدي العزيز **علي**، الذي علمني الصبر والإصرار، وغرس فيّ قيم العلم والمعرفة. إلى من سأظل أحمل لقبه بفخر واعتزاز ما حييت، دمتَ لي قدوةً ومثالًا يُحتذى به.

إليكِ والدتي الحبيبة **بريزة**، يا من قال عنكِ الحبيب المصطفى: "الجنة تحت أقدام الأمهات." أهديكِ هذا التخرج بكل فخر، راجيًا من الله أن يطيل في عمركِ لأهديكِ المزيد من النجاحات التي هي ثمرة لتربيتكِ وتضحياتكِ، وسهركِ وتعبكِ، وكل ما قدمتِه لأجلي. أنتِ نبض حياتي، ودعاؤكِ هو سر كل نجاح أحققه.

إلى إخوتي **هارون**، **هشام**، و**أنيس**، وأخواتي **نبيلة** و**صبرينة**، أنتم نبع الدعم والقوة ووقود نجاحي. لا أملك إلا أن أقول لكم: شكرًا من القلب.

إلى أصدقائي الأعزاء **عبد الناصر**، **زكرياء**، **أسامة**، و**عبد الرحيم** الذين رافقوني في مختلف مراحل حياتي وشاركوا معي التحديات والنجاحات. كان دعمكم وتشجيعكم حافزًا لي للاستمرار.

إلى أستاذي الفاضل **الشيخ خالد زغواني**، الذي له فضل كبير عليّ، أقول لك: بارك الله فيك، ورفع مقامك، ونصرك.

وأخيرًا، إلى كل أقاربي وكل من كان له دور في حياتي، أهديكم هذا العمل المتواضع، راجيًا من الله أن يكون بداية لمزيد من النجاح والتقدم.

"سبحان ربك رب العزة عما يصفون وسلام على المرسلين، والحمد لله رب العالمين، وصلّ اللهم على حبيبنا ونبينا محمد، عليه أفضل الصلاة والتسليم".

محمد لمين

Abstract

Network security risks have significantly increased in recent years due to the rapid growth of digital technology, leading to the emergence of new and advanced threats. To keep pace with these developments, experts are compelled to adopt artificial intelligence (AI)-based solutions to ensure robust defense systems, thereby enhancing their ability to protect networks from escalating cyber threats.

This project leverages Machine Learning (ML) and Deep Learning (DL) techniques to enhance Intrusion Detection Systems (IDS), which are considered the frontline defense against cyber threats. Our primary objective was to develop and evaluate the performance of the trained ML/DL models using well-known metrics. As a foundation, we began with a descriptive and comparative analysis of various IDS-related datasets. Based on this analysis, we selected the CIC-IDS2017 dataset for our study. We then implemented a robust data preprocessing methodology, inspired by best practices established in related works.

The CIC-IDS2017 dataset was used to train our ML/DL models, achieving robust performance across most evaluation metrics on the test set. However, a critical challenge in IDS design is ensuring model resilience against unseen traffic. To address this, we extended our evaluation by validating our trained models on the CIC-IDS2018 dataset, which contains different attack vectors and network configurations. The models demonstrated strong generalization capabilities, maintaining high performance on this unseen data.

The results of our study demonstrated the significant potential of ML/DL techniques in enhancing network security.

Keywords: Intrusion detection systems, Machine Learning, Deep Learning, Pre-processing of network flows, IDS-Related datasets.

Résumé

Les risques liés à la sécurité des réseaux ont considérablement augmenté ces dernières années en raison de la croissance rapide de la technologie numérique, entraînant l'émergence de nouvelles menaces avancées. Pour suivre ces évolutions, les experts ont été contraints d'adopter des solutions basées sur l'intelligence artificielle (IA) afin de garantir des systèmes de défense robustes, renforçant ainsi leur capacité à protéger les réseaux contre les menaces cybernétiques croissantes.

Ce projet utilise des techniques d'apprentissage automatique (Machine Learning ou ML) et d'apprentissage profond (Deep Learning ou DL) pour améliorer les systèmes de détection d'intrusion (IDS), qui sont considérés comme la première ligne de défense contre les menaces cybernétiques. Notre objectif principal était de développer et d'évaluer la performance des modèles ML/DL entraînés en utilisant des métriques standard. Comme base, nous avons commencé par une analyse descriptive et comparative de divers ensembles de données liés aux IDS. Sur la base de cette analyse, nous avons sélectionné l'ensemble de données CIC-IDS2017 pour notre étude. Nous avons ensuite mis en œuvre une méthodologie de prétraitement des données robuste, inspirée des meilleures pratiques établies dans les travaux connexes.

L'ensemble de données CIC-IDS2017 a été utilisé pour entraîner nos modèles ML/DL, obtenant des performances solides sur la plupart des métriques d'évaluation sur le jeu de test. Cependant, un défi crucial dans la conception des IDS est d'assurer la résilience du modèle face à un trafic inconnu. Pour résoudre ce problème, nous avons élargi notre évaluation en validant nos modèles entraînés sur l'ensemble de données CIC-IDS2018, qui contient différents types d'attaques et configurations réseau. Les modèles ont démontré de fortes capacités de généralisation, maintenant une haute performance sur ces données inconnues.

Les résultats de notre étude ont démontré le potentiel significatif des techniques ML/DL pour renforcer la sécurité des réseaux.

Mots-clés : Systèmes de détection d'intrusions, Apprentissage automatique, Apprentissage profond, Prétraitement des flux réseau, Ensembles de données liés aux IDS.

الملخص

ازدادت مخاطر أمن الشبكات بشكل كبير في السنوات الأخيرة نتيجة للنمو السريع في التكنولوجيا الرقمية، ما أدى إلى ظهور تهديدات سيبرانية جديدة ومتطورة. ولمواكبة هذه التطورات، يضطر الخبراء إلى تبني حلول قائمة على الذكاء الاصطناعي لضمان أنظمة دفاع قوية، مما يحسن قدرتهم على حماية الشبكات من التهديدات السيبرانية المتصاعدة.

يعتمد هذا المشروع على تقنيات التعلم الآلي (ML) والتعلم العميق (DL) لتحسين أنظمة كشف التسلل(IDS) ، والتي تعتبر من خطوط الدفاع الأولى ضد التهديدات السيبرانية. يتمثل هدفنا الرئيسي في تطوير نماذج لتعلم الآلي والتعلم العميق وتقييم أدائها باستخدام مقاييس معيارية. بدأنا بتحليل وصفي ومقارن لمجموعات بيانات مختلفة متعلقة بأنظمة كشف التسلل. بناءً على هذا التحليل، اخترنا مجموعة بيانات "CIC-IDS2017" لدراستنا. ثم قمنا بتنفيذ منهجية قوية لمعالجة البيانات، مستندة إلى أفضل المنهجيات المتبعة في الدراسات السابقة في هذا المجال.

تم استخدام مجموعة بيانات "CIC-IDS2017" لتدريب نماذج التعلم الآلي والتعلم العميق المعتمدة في دراستنا، حيث حققت هذه النماذج أداءً قويًا عبر معظم مقاييس التقييم على مجموعة الاختبار. لكن التحدي الرئيسي في تصميم نظام كشف التسلل هو ضمان مرونة النموذج المدرب تجاه حركة المرور غير المعروفة مسبقًا. لمعالجة هذا، وسعنا تقييمنا من خلال التحقق من صحة نماذجنا المدربة على مجموعة بيانات "CIC-IDS2018"، والتي تحتوي على أنماط هجوم وتهيئات الشبكة مختلفة. أظهرت النماذج قدرات تعميم قوية، محافظة على أداء عالٍ لهذه البيانات الجديدة.

أظهرت نتائج دراستنا الإمكانات الكبيرة لتقنيات التعلم الآلي والتعلم العميق في تعزيز أمن الشبكات.

الكلمات المفتاحية: أنظمة كشف التسلل، التعلم الآلي، التعلم العميق، المعالجة المسبقة لتدفقات الشبكة، مجموعات البيانات المتعلقة بأنظمة كشف التسلل.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

ACCS Australian Center for Cyber Security.

Adam Adaptive Moment Estimation.

ADASYN Adaptive Synthetic Sampling.

AI Artificial Intelligence.

AWS Amazon Web Services.

CatBoost Categorical Boosting.

CIC Canadian Institute for Cybersecurity.

CNN Convolutional Neural Networks.

CSE Communications Security Foundation.

DDoS Distributed Denial of Service.

DL Deep Learning.

DoS Denial of Service.

DT Decision Tree.

ENN Edited Nearest Neighbors.

FN False Negatives.

FP False Positives.

GIGO Garbage in, Garbage out.

GRU Gated Recurrent Unit.

IDS Intrusion Detection Systems.

IoT Internet of Things.

LIST OF ABBREVIATION

LAN Local Area Network.

LSTM Long Short-Term Memory.

MIT Massachusetts Institute of Technology.

ML Machine Learning.

MitM Man in the Middle.

Nan Not a Number.

RMSprop Root Mean Square Propagation.

RNN Recurrent Neural Network.

SMOTE Synthetic Minority Oversampling Technique.

TN True Negatives.

TP True Positives.

TPR True Positive Rate.

XGBoost Extreme Gradient Boosting.

GENERAL INTRODUCTION

With the rapid development of digital technology, cyber threats have appeared, making the privacy of individuals and networks more sensitive due to the high flow of data and the fast-growing communication between electronic devices, increasing the risk of cyberattacks. Although traditional Intrusion Detection Systems (IDS) such as anti-virus systems, firewalls, and other security procedures are effective, they have difficulty keeping up with the complexities of modern and constantly evolving cyber threats.

IDSs play a major role in cybersecurity through their ability to monitor network traffic and system activities to identify malicious behaviors and policy violations. However, the traditional version of these systems faces big challenges, such as high false-positive rates and the inability to deal with advanced and unknown attacks. Therefore, experts are turning to Artificial Intelligence (AI) technologies to increase protection capabilities and improve response to new threats [4,5].

AI encompasses various learning techniques, including Machine Learning (ML) and Deep Learning (DL). These approaches are broadly categorized into three main types: supervised learning, which uses labeled data to predict outcomes; unsupervised learning, which identifies hidden patterns in unlabeled data; and reinforcement learning, which optimizes decision-making through reward-based feedback. AI enhances the management and security of networks by improving traffic analysis, automating threat detection, optimizing network performance, predicting potential issues, and enabling real-time decision-making to ensure network integrity [54].

In the context of enhancing network security, our study focuses on improving the efficiency of IDS through the application of supervised learning techniques for both ML and DL, since IDS-related datasets are typically labeled. This approach leverages the pre-classified nature of the data to develop more accurate and efficient detection models.

ML provides effective solutions to improve IDS by analyzing data patterns and learning how to differentiate between usual and unusual activities. Most research and experiments have shown that ML has provided more precise results and better flexibility and adaptability

to threats compared to traditional methods. These models are trained using datasets to detect known threats and pre-defined attack patterns. They also demonstrate the ability to generalize learning to similar patterns, which may help in improving the detection of certain deviations from normal network behavior [18].

DL models have gained significant attention in recent years due to their exceptional ability to process large volumes of data and automatically extract key features. The strength of DL models lies in their capacity to detect complex patterns in data and adapt to the evolving landscape of security threats. These properties significantly contribute to improving the effectiveness of IDS, leading to reduced false alarm rates and enhancing the immediate response to potential threats [55].

The application of ML and DL techniques to IDS is a major leap and advancement in the field of cybersecurity. This is because it improves the perfection of defense systems while expanding their range, which enhances their ability and efficiency in protecting networks from increasingly complex cyber threats.

This project leverages ML and DL techniques to improve IDS. To showcase the different steps of our research, we have organized this document into three main chapters as follows:

- The first chapter provides a comprehensive analysis of IDS, explaining their various types, working mechanisms, and role in enhancing cybersecurity. The chapter also reviews the importance of AI in supporting and advancing these systems to achieve a higher level of security.
- Chapter two presents a descriptive and comparative study of five IDS-related datasets. Several factors are highlighted, including strengths and limitations, the diversity of attack patterns, the balance of the data distribution, noise levels, the importance of features, and the extent to which these datasets accurately reflect modern network traffic.
- Chapter three addresses the practical aspect of our research in the field of IDS using ML/DL techniques, starting with data preprocessing, followed by training and evaluating six different models, along with an analysis of the results.

CHAPTER 1

Security and Intrusion Detection Systems

1.1 Introduction

With the rapid pace of technological advancement and the diversification of communication channels, coupled with the widespread adoption of digital systems and data transfer across networks, the risk of unauthorized access and data manipulation by malicious actors or competitors is increasing. Consequently, ensuring information security through advanced protection technologies and access control mechanisms has become essential for building a secure infrastructure capable of mitigating these vulnerabilities.

In this context, the implementation of IDS stands out as a critical defense mechanism. These systems provide continuous monitoring and surveillance of network activities, detecting any attempts to breach security policies or unauthorized intrusions [5].

This chapter examines the foundational concepts of IDS and discusses the challenges and limitations these systems encounter. Additionally, it presents proposed solutions to mitigate these challenges. Finally, the pivotal role of AI in enhancing cybersecurity is highlighted, particularly through its ability to autonomously detect and respond to threats by analyzing vast amounts of data and identifying abnormal patterns. AI also plays a crucial role in assessing risks, identifying vulnerabilities before they can be exploited, and improving the efficiency of threat detection and response mechanisms.

1.2 IDS

To gain a deeper understanding of IDS, it is essential to explore their definition, classifications, the types of security threats they address, their components and architecture, as well as the categories of attacks, challenges, and limitations that affect their effectiveness:

1.2.1 Definition

An IDS acts as a real-time anomaly detection mechanism. It continuously monitors network traffic and system activities, comparing them to predefined rules or behavioral patterns. When the system identifies deviations that indicate potential malicious activity, it generates an alert for further investigation. This enables the prompt implementation of appropriate countermeasures to mitigate security threats and protect the network infrastructure [4,5].

1.2.2 Types of Security Threats and Categories of Attacks

Understanding the diversity and evolution of security risks, as well as their categories, is crucial for developing and implementing effective defense mechanisms. Initially, based on the comprehension of the role of IDS, various types of security threats will be explored, followed by a detailed of the categories of attack methods.

1.2.2.1 Types of Security Threats

In the first step, we will provide a breakdown of some common security threats and discuss how IDS can help mitigate them:

Malware

It's including a variety of security threats, such as viruses, Trojans, ransomware, and spyware. These programs can steal sensitive data, corrupt files, disrupt processes, and compromise systems. To block these threats, signature-based IDS can detect known malware based on predefined patterns. Additionally, anomaly-based IDS can identify unusual behavior patterns that may indicate the presence of malware [6].

Phishing Attacks

Phishing emails or messages deceive users into clicking malicious links, downloading infected attachments, compromising personal information, or granting unauthorized access to systems. Although IDS cannot specifically stop phishing attempts targeting individual users, it can help identify suspicious network activity associated with phishing campaigns, such as attempts to access known phishing sites [6].

Denial-of-Service (DoS) Attacks

DoS attacks overwhelm a system with excessive traffic, rendering it unavailable to legitimate users. This can disrupt critical services and lead to financial losses. By monitoring traffic patterns, IDS can identify anomalies indicative of DoS attacks, such as sudden spikes in traffic volume or attempts to exploit known DoS vulnerabilities. Early detection enables the implementation of mitigation measures [6].

Man-in-the-Middle (MitM) Attacks

It's a type of cyberattack where an attacker secretly intercepts and relays communications between two parties, enabling them to eavesdrop, modify, or inject malicious data into the communication channel. The attacker positions themselves in the middle of the conversation, acting as a proxy without the parties' knowledge. This allows the attacker to monitor, capture,

and potentially manipulate exchanged data, compromising the confidentiality and integrity of the communication [6].

Social Engineering Attacks

It represents a sophisticated form of manipulation that exploits human psychology to deceive individuals into divulging sensitive information or granting unauthorized access to systems. These attacks often involve phishing emails, phone calls, or the impersonation of legitimate entities. While IDS may not be able to directly block social engineering attacks, they can provide a secondary layer of defense by detecting suspicious activities that may arise from a successful attack, such as attempts at unauthorized access or data exfiltration [6].

Insider Threats

These threats result from malicious or negligent actions by authorized users within an organization. Such individuals may have access to sensitive data and systems, posing a significant security risk. This is why anomaly-based IDS can be useful in identifying unusual activity from authorized users, which may indicate malicious intent. For example, this includes attempts to access unauthorized files or unusual data transfer activities [6].

1.2.2.2 Categories of Attacks in the Security Domain

In the next step, as shown in Table 1.1, we will present several illustrative examples of security attack categories to enhance understanding of the different types of threats:

Table 1.1: Examples of categories of attacks in the security domain [22].

Category of Attack	Description	Example
Reconnaissance Attacks	Gather information about a target system or network.	Ping Sweeps, Port Scans, Social Engineering
Denial-of-Service (DoS) Attacks	Overwhelm a system or network with traffic.	SYN Floods, Distributed DoS (DDoS) Attacks, Smurf Attacks
Privilege Escalation Attacks	Gain higher privileges on a system.	Exploiting software vulnerabilities, Password Spraying, Pass-the-Hash Attacks
Malware Attacks	Infect a system with malicious software.	Viruses, Worms, Trojans, Ransomware
Man-in-the-Middle (MitM) Attacks	Eavesdrop on communication and potentially alter data.	ARP Spoofing, SSL Stripping, Public Wi-Fi Eavesdropping
SQL Injection Attacks	Inject malicious SQL code to manipulate a database.	Tricking a login form into stealing user data.
Phishing Attacks	Trick users into revealing sensitive information.	Phishing emails impersonating legitimate entities.
Watering Hole Attacks	Compromise websites or resources to infect devices.	Hacking a website popular among gamers to inject malware.

1.2.3 Classification of IDS

IDS can be classified based on the techniques used to detect intrusions into a computer system. Here are the three main classifications:

Signature – based IDS

Signature-based IDS rely on a predetermined set of signatures, which are patterns identifying known attacks (see Figure 1.1**)**. These signatures may be based on data packets, system calls, or other activities. When the IDS detects a match between a signature and ongoing activity, it raises an alarm.

The IDS continuously monitors all incoming and outgoing network traffic, analyzing it against a comprehensive database of known cyber threats and attack patterns. If any activity is detected that matches the signatures or behaviors stored this knowledge base, the IDS immediately makes an alert, notifying security personnel about the potential threat [1,8].

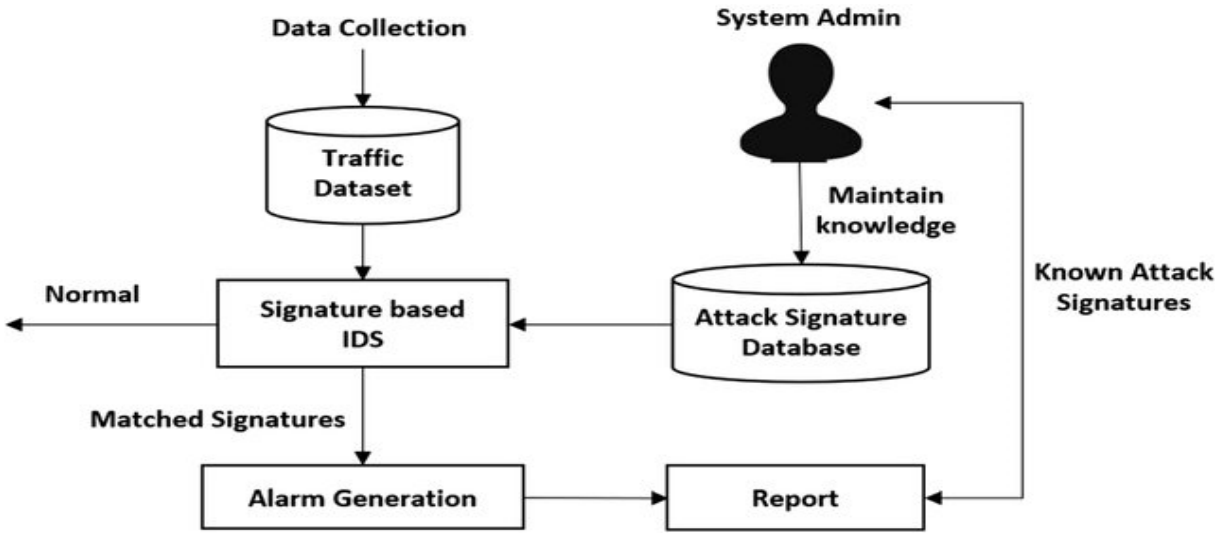


Figure 1.1: Signature-based IDS [2].

Anomaly – based IDS

Anomaly-based IDS monitors system activity for deviations from a baseline of normal behavior. This baseline can be confirmed by learning the patterns of network traffic, system calls, or other activity over time. When the IDS detects an anomaly, it raises an alarm. An Anomaly-based IDS keeps the system in constant adaptation to the network flow which is by nature constantly changing. It learns the normal traffic and flags any unusual activity for security personnel to examine. In this way it ensures its effectiveness over time [1,8]. Figure 1.2 helps to better understand:

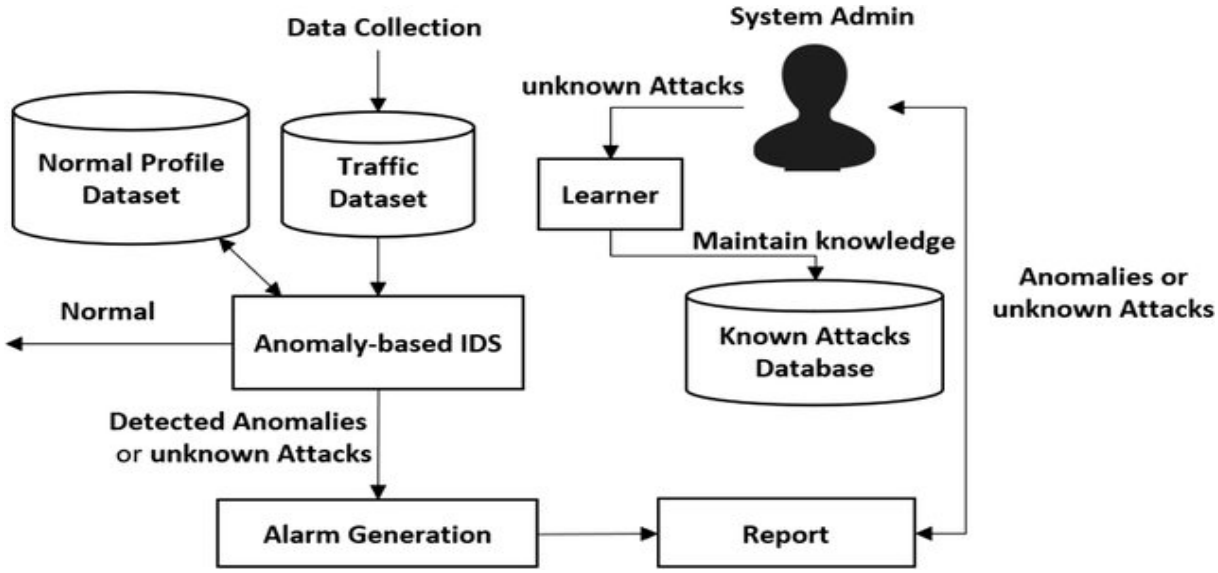


Figure 1.2: Anomaly – based IDS [2].

Hybrid IDS

Hybrid IDS combine signature-based and anomaly-based detection techniques. This approach can provide the benefits of both techniques, while reducing some of the drawbacks [8]. For example, a hybrid IDS can use signature-based detection to identify known attacks, and anomaly-based detection to identify zero-day attacks [1], as shown in Figure 1.3:

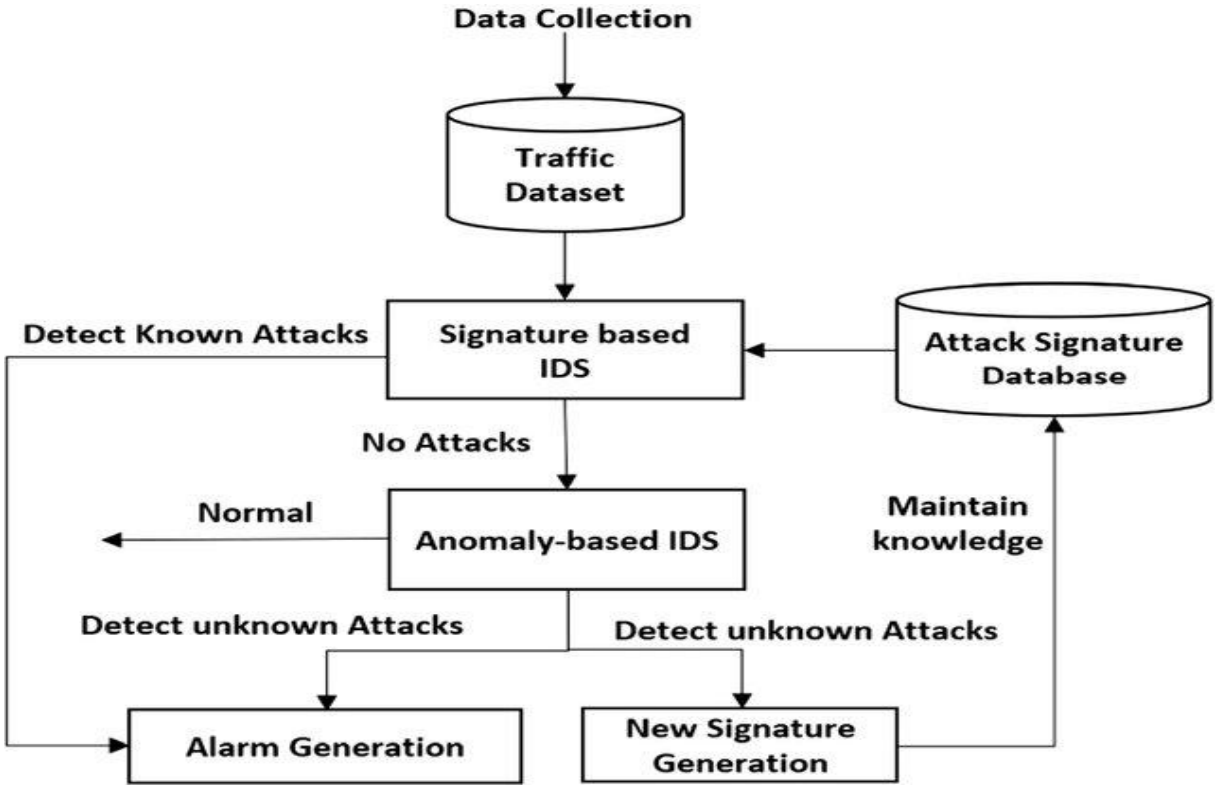


Figure 1.3: Hybrid IDS [2].

In Figure 1.4, a more comprehensive illustration of the main types of IDS is shown:

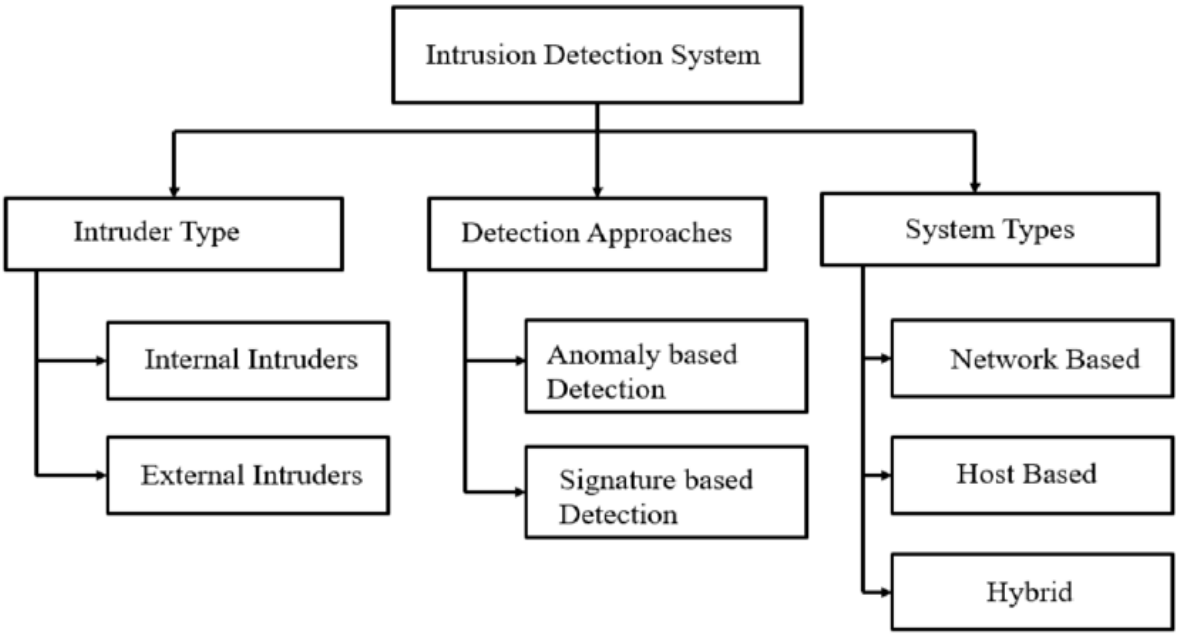


Figure 1.4: Types of IDS [10].

Advantages & Disadvantages of IDS techniques

The following Table 1.2 summarizes the advantages and disadvantages of different IDS techniques with a simple comparison between them to guide cyber security passionate in selecting the most suitable approach for their needs.

Table 1.2: Advantages & Disadvantages of IDS Techniques.

IDS Techniques	Advantages & Disadvantages
Signature IDS	Highly accurate for detecting known attacks.
	Requires regular updates to the signature database.
	Relatively quick and efficient.
	Cannot detect zero-day attacks (attacks that are unknown).
Anomaly IDS	May be more responsive to false positives than signature-based IDS.
	May require more processing power to monitor system activity.
	Can detect zero-day attacks.
	More flexible than signature-based IDS.
Hybrid IDS	Can be more complex to configure and maintain than signature-based or anomaly-based IDS.
	May still be responsive to false positives.
	Provides comprehensive protection against a wider range of attacks.
	Can be more precise than signature-based or anomaly-based IDS alone.

1.2.4 IDS Components and Architecture

An IDS works by continuously monitoring doubtful activity and making alerts when it detects potential threats. The following provides an overview of its key components and architectures, as shown in Figure 1.5:

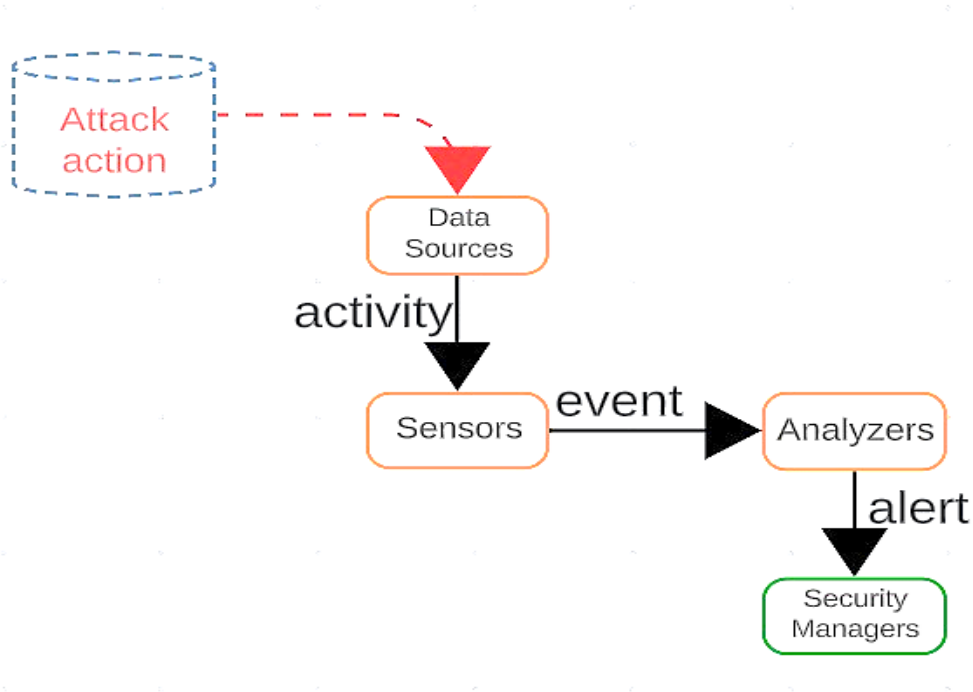


Figure 1.5: Components & Basic Architecture of an IDS.

- **Sensors:** These are the eyes and ears of the IDS, deployed strategically to collect data from various sources. They can be:
  - ✓ Network Traffic Sensors (Network-based IDS): Monitor network traffic for suspicious activity, such as port scans, unauthorized access attempts, or malware signatures.
  - ✓ File System Sensors (Host-based IDS): Monitor system activity on individual devices, including file access, system calls, and application logs, to detect malware or unauthorized modifications.
  - ✓ Agent-based Sensors: Lightweight software agents installed on devices that collect and report system activity data to a central IDS manager.
- **Analyzers:** These are the brains of the IDS, responsible for analyzing the data collected by the sensors. They use various techniques like signature matching, anomaly detection, and behavioral analysis to identify potential threats [9].
- **Consoles:** These are the user interfaces, which use security personnel to monitor the IDS activity, view alerts, investigate potential threats, and configure the system.

1.2.5 Challenges and Limitations of IDS

IDS are valuable security tools, but they are not without their challenges and limitations. Here's a breakdown of some key issues to consider:

1.2.5.1 False Alert and False Detection

- **False Alert (false positive):** These happens when the IDS raises an alert for harmless activity that mistakenly resembles a known attack signature or anomaly. False alert waste time and resources for security personnel investigating them, as mentioned in [11,12].
- **False Detection (false negative):** These happens when the IDS fails to detect a real security threat. This can happen due to limitations in signature coverage (missing signatures for new attacks) or the inability to identify subtle anomalies indicative of an attack [11,12].

1.2.5.2 Factors Contributing to False Alert and False Detection

At this point, the four main factors contributing to this problem have been highlighted:

- **Imperfect Signature Matching:** Signatures may not be specific enough, leading to false positives for benign activity with some similarities to malicious patterns.
- **Overly Sensitive Anomaly Detection:** Anomaly thresholds might be set too low, causing alerts for minor variations in normal activity.
- **Rapidly Evolving Threats:** Attackers constantly develop new techniques, and signature updates might not always keep pace, leading to false negatives.
- **Limited Visibility:** Network-based IDS might miss attacks happening within individual devices (host-based IDS can help address this).

1.2.5.3 Scalability and Performance Issues

The three primary factors contributing to this issue have been identified:

- **Large Data Volumes:** Modern networks generate massive amounts of data. Analyzing all this data for anomalies can be computationally expensive and resource-intensive, impacting system performance.
- **Large-Scale Deployments:** Deploying IDS across a vast network infrastructure can be challenging to manage and scale effectively, as mentioned in [13].

- **Alert Fatigue:** A high volume of alerts, even with a low false positive rate, can overwhelm security personnel, leading to alert fatigue and potentially missed critical threats [14].

1.2.5.4 Evasion Techniques by Attackers

Attackers use various evasion techniques to get around IDS in order not to be detected:

- **Polymorphism:** Attackers can modify malware code partially to avoid signature matching.
- **Encryption:** Encrypting malicious traffic can render it invisible to signature-based detection on network traffic.
- **Low and Slow Attacks:** Spreading out an attack over time and using minimal resources can avoid triggering anomaly detection.
- **Zero-day Attacks:** By definition, these new attacks lack corresponding signatures, making them undetectable by signature-based IDS [15].

Here are some proposed solutions to address these challenges and limitations of IDS:

- **Fine-Tuning IDS Configuration:** Proper configuration of signature matching criteria, anomaly thresholds, and filtering rules can help reduce false positives.
- **Security Awareness Training:** Educating users about social engineering tactics and phishing attempts can help prevent them from falling victim to attacks**.**
- **Layered Security Approach:** IDS is just one piece of the security puzzle. Combining it with firewalls, vulnerability management, and data encryption provides a more comprehensive defense.
- **Staying Informed about Threats:** Security teams need to stay updated on the latest attack methods and adjust their IDS configurations accordingly.

By acknowledging these limitations and implementing appropriate mitigation strategies, organizations can maximize the effectiveness of their IDS and enhance their overall security posture.

1.2.5.5 IDS Evaluation and Selection Criteria

Choosing the most suitable IDS requires careful evaluation. Here are key factors to consider:

Threat Detection Capabilities

IDS supports advanced threat detection capabilities to identify and mitigate cyber threats.

When evaluating a security solution, it is essential to understand its detection methods. Does it rely on signatures for known threats, anomaly-based detection for unusual activity, or a combination of both? More importantly, how well does it address the specific threats an organization faces? Additionally, the ability to integrate with threat intelligence feeds for real-time updates is essential to staying ahead of evolving threats, as mentioned in [16]. On the deployment side, considering the existing infrastructure is key. Does the solution support network-based, host-based, or cloud-based deployment, or maybe even a combination? Finally, easy integration with the current setup is necessary for efficient operation.

Evaluating a security solution requires considering several key factors, especially when it comes to scalability and performance, which are critical. It's important to assess whether the solution can handle the volume of data a network generates without impacting performance and if it's scalable to accommodate future growth. Management and usability are also important considerations. The interface should be examined to determine if it's easy to use for configuration, monitoring, and alert management, and if it offers reporting capabilities to analyze security trends. Finally, cost and support must be measured, taking into account licensing fees, maintenance costs, and training requirements for commercial solutions.

Commercial Vs Open-Source IDS

When selecting an IDS, organizations typically choose between commercial and open-source solutions. Commercial IDS often offer comprehensive, pre-configured features such as signature databases, and come with dedicated vendor support, making them easier to deploy and maintain, especially for users with limited cybersecurity expertise. However, they can be costly, particularly for large-scale environments. Open-source IDS, on the other hand, are freely available and offer high levels of customization, enabling users to tailor the system to specific needs. These systems usually require more advanced technical knowledge but benefit from active community support, which drives their continuous development. IDS have been successfully implemented across various sectors, such as financial institutions to detect fraud, healthcare providers to secure sensitive patient data, and government agencies to protect critical infrastructure from cyberattacks [17].

1.2.5.6 The Future of IDS (Emerging Technologies and Evolving Strategies)

The world of cybersecurity is constantly evolving, and IDS are keeping pace by leveraging new technologies and refining their approaches. Here's a brief look into what the future holds for IDS:

Machine Learning (ML)

ML algorithms are being progressively integrated into IDS for anomaly detection. ML can analyze vast amounts of data to identify perfect patterns that might signify attacks, improving the accuracy of anomaly-based detection and significantly reducing false positives and false negatives [18].

Big Data Analytics

The ability to analyze massive datasets in real-time is crucial for effective security. IDS are integrating big data analytics tools to handle the ever-growing volume of network traffic and system activity data, enabling them to detect emerging threats more quickly [19].

User and Entity Behavior Analytics (UEBA)

UEBA goes beyond traditional network traffic analysis. It focuses on user and entity behavior patterns, including login attempts, file access, and application usage. This allows IDS to identify suspicious activity that might indicate compromised accounts or insider threats [20].

1.2.5.7 How IDS are Progressing to Address New Challenges

Emerging trends in IDS include a focus on deception and active defense mechanisms to attract and disrupt attacks, deeper integration with the other systems for a large security context, and cloud-based IDS solutions to secure cloud environments and workloads. Additionally, adapting IDS for IoT (Internet of Things) device security needs and the explosion of data from connected devices will be crucial. However, the future of IDS depends on collaborative efforts—sharing threat intelligence, best practices, and continuous innovation—between security vendors, researchers, and the wider cybersecurity community to stay ahead of promoting threats [21].

1.3 The Importance of AI in Security

AI is rapidly transforming the security field, offering significant advantages over traditional methods. Here are some key reasons, why AI is becoming increasingly important in security:

1.3.1 AI-Powered Anomaly Detection: Unmasking Hidden Threats in Real-Time

The threat environment demands security solutions that are not only comprehensive but also adaptable. Traditional signature-based detection methods struggle to keep up with the ingenuity of attackers. This is where AI comes in, revolutionizing the way we identify and respond to security threats [23].

1.3.1.1 The Power of AI in Anomaly Detection

AI algorithms excel at processing large datasets to uncover hidden patterns, making them well-suited for security anomaly detection. By analyzing network traffic, logs, and user activity, AI can identify deviations that may indicate zero-day attacks, phishing scams through anomalous emails, or insider threats exhibiting suspicious behavior. This pattern recognition capability allows AI to detect threats without relying solely on predefined signatures [23].

1.3.1.2 Benefits of AI-Powered Anomaly Detection

AI enables proactive threat detection by identifying anomalies in real-time, preventing attacks quickly. It reduces false alarms by distinguishing true abnormalities from benign variations. This broader detection of threats, including zero-day threats, strengthens the security position. Additionally, AI's continuous learning provides enhanced threat intelligence, offering insights into attacker behaviors and emerging threats to inform security strategies [23].

1.3.1.3 The Future of IA in Security

While AI plays a crucial role in threat detection, human expertise remains irreplaceable. Security professionals need to interpret AI-generated alerts, investigate potential threats, and make informed decisions about mitigation strategies. The future of security lies in a collaborative approach where AI empowers human analysts to make faster and more accurate security decisions.

1.3.2 Automated Security Tasks and Improved Efficiency

AI can automate repetitive tasks like log analysis, incident response, and vulnerability scanning. This frees up security professionals to focus on more strategic tasks, such as investigation and threat hunting.

It also reinforces cybersecurity operations through automated log analysis to identify anomalies, enabling analysts to focus on investigating threats. It streamlines incident response by handling initial containment, evidence collection, and triage. AI-powered vulnerability scanners automate identifying system vulnerabilities to prioritize patching. Additionally, AI automates installing security patches, ensuring timely updates to reduce vulnerability windows while minimizing manual workloads [23].

1.4 Conclusion

In conclusion, this chapter has demonstrated the critical role of IDS in securing modern networks by covering key concepts, classifications, components, and the security threats they address. Despite their effectiveness, IDS continue to face challenges, including false positives, scalability issues, and sophisticated evasion techniques. However, the growing integration of AI is expected to significantly enhance IDS capabilities, particularly in anomaly detection and automation. As AI technologies advance, they are poised to play a pivotal role in improving the precision, scalability, and adaptability of IDS in the future.

In upcoming studies, special attention will be given to the datasets used to enhance IDS capabilities. The next chapter, titled "Descriptive and Comparative Study of IDS-related Datasets," will explore the sources, structures, and application domains of these datasets. By comparing their strengths and limitations, this analysis will provide a robust foundation for future research aimed at further advancing IDS performance.

CHAPTER 2

Descriptive and Comparative Study of IDS-related Datasets

2.1 Introduction

When talking about a dataset in general, it is not reduced to simple information; rather, it represents a compilation of interconnected data points. Data includes anything that can be recorded and stored electronically, such as numbers, text, images, videos, audio recordings, and so on. Talking about the dataset in the field of cybersecurity, which largely revolves around the development of effective IDS, the dataset concerns network traffic, which can be either normal or various scenarios of attack, and we mentioned some of these scenarios specifically in the first chapter. This data is used to train ML and DL models to detect and respond to various threats.

The next chapter presents a descriptive and comparative study of IDS-related Datasets, taking into account several factors, including strengths, limitations, attack diversity, data balance, noise levels, the importance of features, and the representativeness of modern network traffic. There is a wide range of datasets in this area; it is impractical to work with all of them. Therefore, five datasets (KDD Cup 99, NSL-KDD, UNSW-NB15, CIC-IDS2017, and CIC-IDS2018) are analyzed due to their importance, the novelty of some of them, and their ability to capture a wide range of network traffic patterns. Furthermore, numerous studies have been conducted on these datasets.

2.2 Datasets for Cybersecurity

The data set in cybersecurity includes packets or flow records captured from network traffic such as Wireshark and tcpdump over a specific period of time, which are collected through several methods, most notably package capture tools (Packet Sniffers). System logs such as login logs, application logs, operating system logs, and firewall logs are also collected, as they may contain useful information about suspicious activity. Virtual network environments are created to simulate network traffic and various attacks in a secure and controlled manner. Also, in some cases, data is collected from real business networks. Figure 2.1 shows the timeline of a network traffic-based dataset.

Observed phenomena such as packet sizes, port numbers, and used protocols are called "features" and are individual measurable characteristics that are stored in NetFlow logs. The data is labeled indicating whether the traffic is normal or malicious, and often identifies the type of attack. Therefore, the more recent and diverse the attack collection in a dataset, the more important it becomes.

Imbalanced datasets are a challenge for ML and DL models. This means that the number of

records in both the normal and malicious categories must be close to or proportional to each other because the models tend to be biased towards the more representative category. Noise in the data, whether caused by errors in data collection, missing or incomplete data, or irrelevant features, is another challenge as it leads to training models on inaccurate data [24].

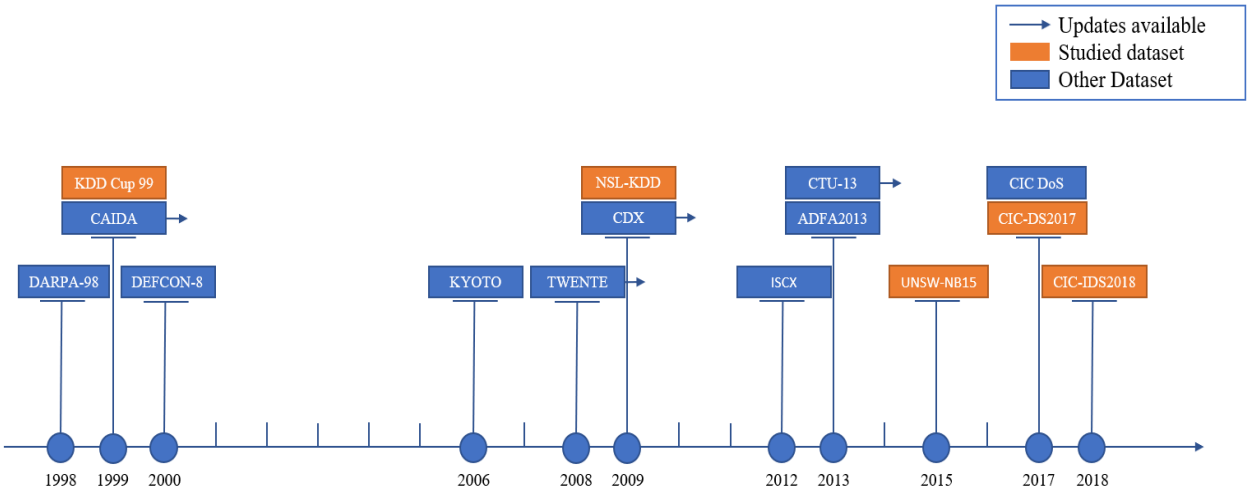


Figure 2.1: Timeline of network traffic-based dataset.

2.3 The KDD Cup 99 Dataset

Considered one of the first datasets in security, it was prepared in 1999 by Stolfo et al [25]. The tcpdump data served as the basis for creating this set, which was captured from the DARPA IDS program in 1998 within a simulated Local Area Network (LAN) by Massachusetts Institute of Technology (MIT) Lincoln Labs for about nine weeks. The dataset contains 42 features and five types of traffic that are classified as normal or attack-type (as shown in Table 2.1). There are more than 5 million connection records, and each record contains about 100 bytes of data. During the period of release of this dataset, many researches and studies were conducted on it, but with the speed of progress, defects were found in it, which made work on it almost non-existent in the current period.

The features in the KDD dataset are divided into four categories. Basic features that define the basic context for understanding a connection, such as the type of protocol and service used. We also find content-based features that identify attacks that exploit weaknesses in data formats or protocols. As well as time-based traffic features such as duration and number of bytes transferred. Finally, host-based traffic features, where the number of failed logins attempts from the host may also indicate an attack [26].

Table 2.1: Different types of traffic in the KDD Cup 99 dataset.

Traffic Types			Number of Instances		
Attacks	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_overflow	30	52	
		Loadmodule	9		
		Perl	3		
		Rootkit	10		
	R2L	Ftp_write	8	1126	
		Guess_passwd	53		
		Imap	12		
		Multihop	7		
		Phf	4		
		Spy	2		
		Warezclient	1020		
		Warezmaster	20		
Normal			972781		
Total			***4898431***		

2.4 The NSL-KDD Dataset

To stay on the same side of the development and modernization wheel, in 2009, Tavallaee et al. [27,28]. from the University of New Brunswick, the Canadian Institute for Cybersecurity (CIC), and others attempted to solve some of the limitations in the KDD Cup 99 dataset, and the result was the NSL-KDD dataset. This new dataset consists of two primary components: a training set and a test set. It also contains the same features and attack types found in the old dataset. Table 2.2 provides a detailed overview of the various types of traffic, including both normal and attack types, found in the NSL-KDD dataset.

There were improvements in the NSL-KDD dataset. Notably, there are no redundant or duplicate records in the training set. This dataset also implements a more balanced selection strategy compared the KDD Cup 99 dataset, prioritizing the inclusion of records from underrepresented difficult attack categories. Finally, the dataset maintains a reasonable size for both training and testing sets, enabling researchers to perform comprehensive experiments on the entire dataset without the need to randomly select a small portion [29].

Table 2.2: Different types of traffic in the NSL-KDD dataset.

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		
		Httptunnel	0			133		
Normal			67343			9711		
Total			***125973***			***22544***		

2.5 The UNSW-NB15 Dataset

In order to create a dataset that represents modern attack scenarios, which KDD lacks, the

Australian Center for Cyber Security (ACCS) took the lead and worked with other researchers around the world to build the UNSW-NB15 dataset in 2015. To achieve this goal, ACCS's IXIA PerfectStorm tool was used in the Cyber Range Lab to create a hybrid of real modern normal behaviors and synthetic attack activities. The dataset consists of 49 features extracted using tools such as Argus and bro-IDS. In addition, twelve algorithms written in C# were used to capture various aspects of the behavior of network packets. This dataset contains the attack subcategory within nine types of attacks (as shown in Table 2.3).

This dataset uses both packet-based features, flow-based features, and general-purpose features. Packet-based features provide detailed analysis of individual network packets, while flow-based features analyze connected sequences within a network flow. The dataset also includes general-purpose features, which are broader characteristics that do not fit neatly into the packet or flow categories. These may encompass overall network statistics, temporal patterns, or features that aggregate information from multiple sources. Some features are engineered to analyze sequences of multiple connection records, enabling the detection of advanced threats, such as slow scanning techniques used by attackers [30].

Table 2.3: Different types of traffic in the UNSW-NB15 dataset.

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***	

2.6 The CIC-IDS2017 Dataset

The CIC returned to present the best in collaboration with the University of New Brunswick by creating the CIC-IDS2017 dataset, which is considered to be very recent, the most popular, and readily available to the public. It is worth noting that it not only contains the most updated attack scenarios but also meets all the criteria for real-world attacks. Where the Bprofile system was used in them. One of the key strengths of this dataset is its incorporation of various network traffic patterns generated by protocols such as HTTP, HTTPS, FTP, SSH, and email in order to use them to infer the abstract behaviors of 25 users. In addition, network flows between the source and the destination are considered bidirectional. It involved benign data and 14 different types of attacks, which were collected over five consecutive days and stored in eight separate files (as shown in Table 2.4). The set also contains 79 features. Many new models and algorithms have been analyzed and developed due to the attraction of this dataset for researchers [27,31].

Table 2.4: Different types of traffic in the CIC-IDS2017 dataset.

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-Infilteration.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***	

2.6.1 Descriptions of Dataset

The CIC-IDS2017 traffic was created by a testbed architecture consisting of a victim network with four machines and an attacker network with fifteen machines during work hours from Monday to Friday. It can be seen from Table 2.4 that Monday was free of attacks, but during the remaining four days, a variety of attacks were conducted. The data from Tuesday, Wednesday, and Thursday mornings are dominated by a primary attack type each day but include multiple subtypes like "FTP-Patator" and "SSH-Patator" attacks, which are two variations of Patator attacks. Wednesday highlights different "DoS" attacks like Hulk and Slowloris, along with the "Heartbleed" attack. Thursday morning involves three types of web attacks: "XSS," "Brute Force," and "SQL Injection." This variation within a single attack category makes these days ideal for developing multi-class classification detection models. In contrast, the data from Thursday afternoon and Friday afternoon is more suited for binary classification. Thursday afternoon includes "BENIGN" traffic versus the "Infiltration" attack, while Friday afternoon presents "BENIGN" traffic with either a "DDoS" or "PortScan" attack [31,32].

Distribution of instances in figures 2.2 and 2.3 can help better understand, since we note that the "BENIGN" category has the highest number of instances, which is normal for traffic. For the attack scenarios: "DDoS," "DDoS Hulk," and "PortScan" have the largest number of instances where such high numbers reflect the prevalence of such attacks. In addition, "PortScan" is a common reconnaissance activity performed by attackers to search for vulnerabilities and penetration opportunities in networks. While the incidence of specific web attacks, such as "XSS," "Brute Force," and "SQL Injection," are relatively low.

Dataset suffers from a high imbalance, which is one of the shortcomings as clearly illustrated in Figures 2.2 and 2.4, where benign traffic contains 80.30% compared to attacks accounting for 19.70%. This leads to a great possibility that instances of a particular attack label, such as "Heartbleed" or "Web Attack-Sql Injection" may not be found in the training set, as well as the tendency of models towards the benign category. An important shortcoming is also its enormous size, which spans eight files and makes working with a large and fragmented data set stressful.

The "Heartbleed" attack type has only 11 recorded instances, which may indicate either the rarity of such attacks or the presence of incorrect or incomplete data, raising concerns about its reliability and accuracy [31].

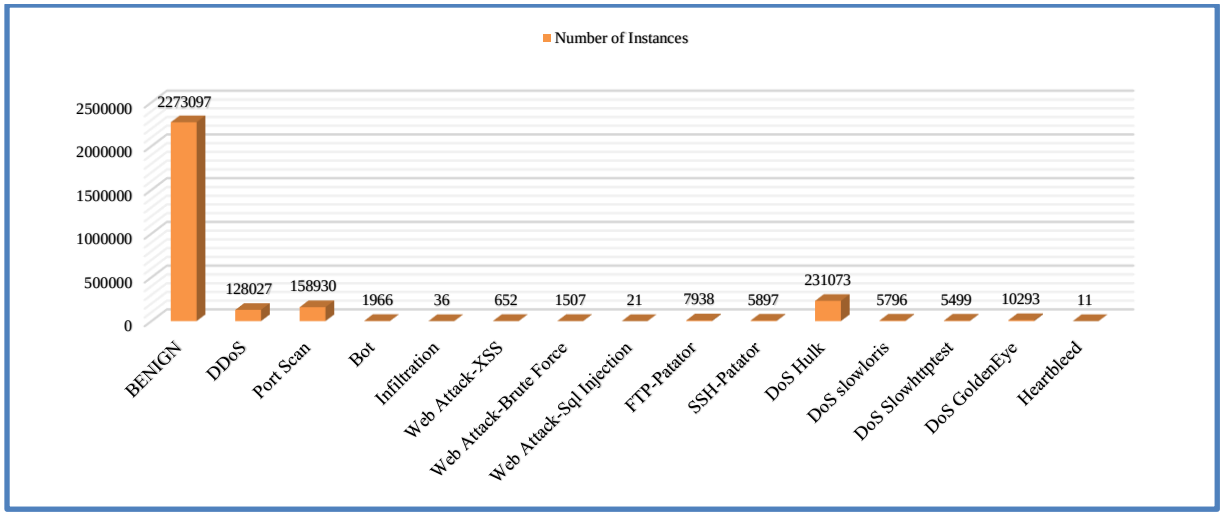


Figure 2.2: Number of instances for traffic types in the CIC-IDS2017 dataset.

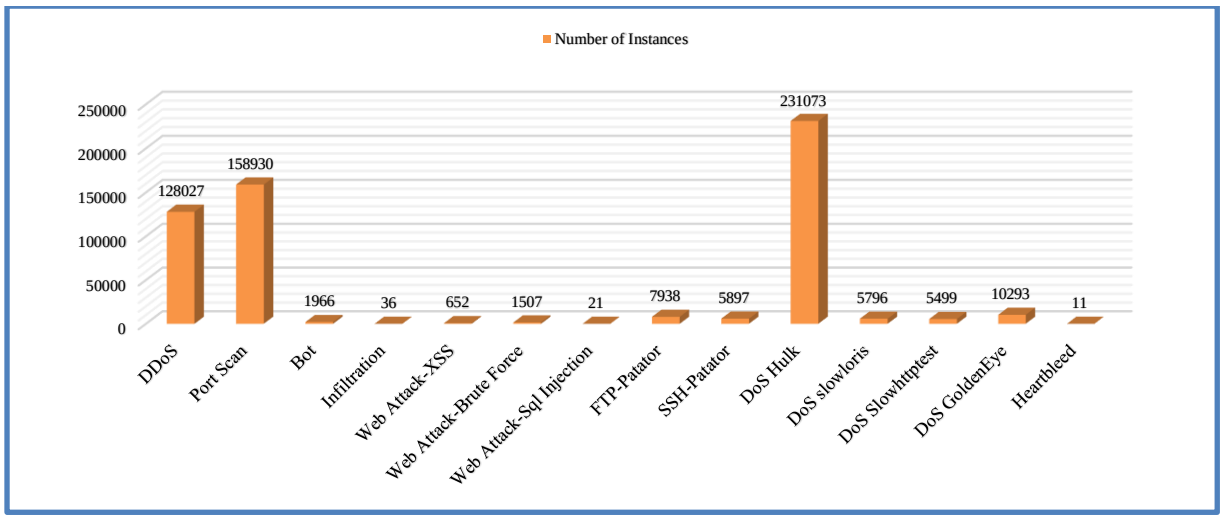


Figure 2.3: Number of instances for attack type in the CIC-IDS2017 dataset.

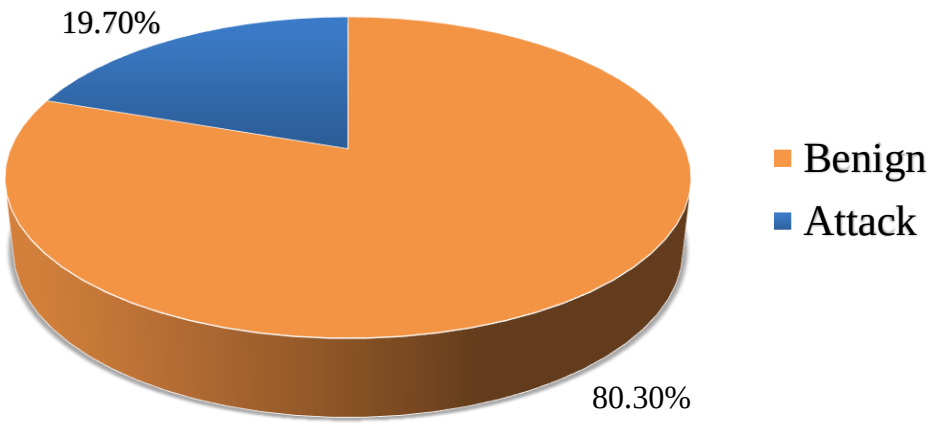


Figure 2.4: The CIC-IDS2017 dataset's distribution of benign and attack instances.

2.7 The CIC-IDS2018 Dataset

As part of the ongoing efforts to provide the research community with reliable and up-to-date data sources, we return to talking about CIC, which is known internationally for its distinguished datasets, as it added a new dataset to its series of releases in 2018 in cooperation with the Communications Security Foundation (CSE) using the AWS (Amazon Web Services) platform. The dataset, known as CIC-IDS2018, surpasses its predecessor "CIC-IDS2017" by providing a more complex and diverse network environment. Incorporates 84 features extracted using the CICFlowMeter-V3 tool, where these features are calculated separately for both the forward and backward directions. Recorded 14 different attack types in addition to benign traffic. This dataset is distributed across 10 separate files, as displayed in Table 2.5. Finally, it was released in both raw PCAP format, which enables the extraction of new features, and a pre-processed and ready-to-use CSV format [33,34].

Table 2.5: Different types of traffic in the CIC-IDS2018 dataset.

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

2.7.1 Descriptions of Dataset

The data was created using a realistic network environment within 10 days, from Wednesday 14/2/2018 to Friday 2/3/2018. Were used 50 attack and 420 victim machines along with 30 servers [33].

The dataset is characterized by a very low number of duplicates or uncertain data, and a large variety of attack traffic scenarios, as shown in Figures 2.5 and 2.6. However, it is important to note, as illustrated in Figure 2.7, that the dataset is imbalanced, as the number of instances for Benign is 83.07% and Attack is 16.93% [33,35].

Based on Tables 2.4 and 2.5, a substantial increase of two million attack types and eleven million traffic benign was recorded compared to the CIC-IDS2017 dataset.

The dataset from Figure 2.6 a significant variation in the number of instances across different attack types. We find certain types of attacks with high number of instances, such as "DDOS attack-HOIC," "DoS attacks-Hulk," and "DDoS attacks-LOIC-HTTP." Conversely, attack types like "Boot," "DoS attacks-SlowHTTPTest," "Infiltration," "FTP-BruteForce," and "SSH-Bruteforce" exhibited a moderate number of instances. As for the rest, they are small or almost non-existent. Where "SQL Injection" took the smallest value of 87 instance, which is negligible. This substantial variation underscores the diversity of this dataset.

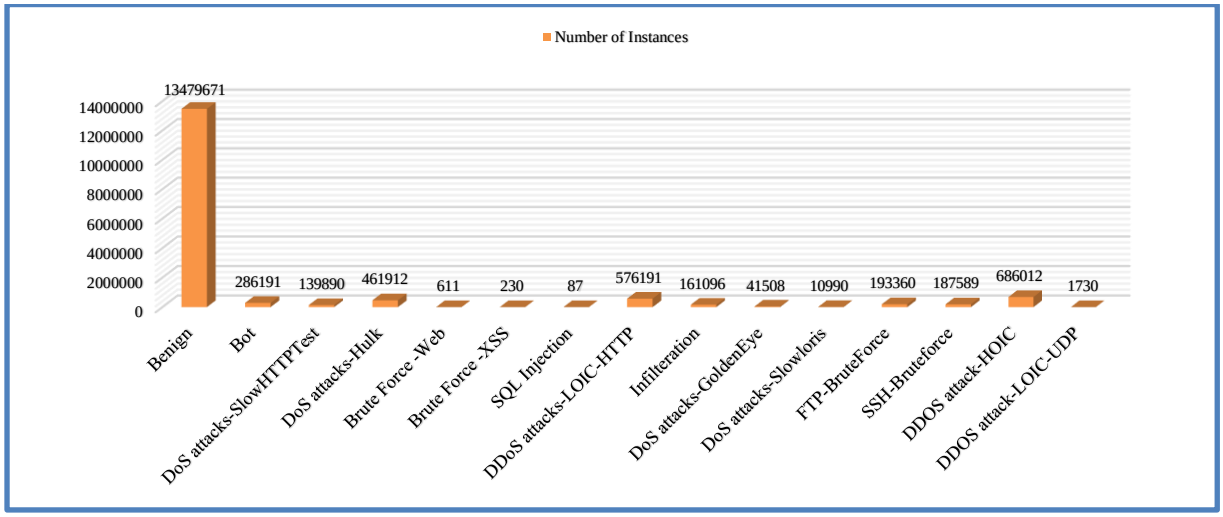


Figure 2.5: Number of instances for traffic types in the CIC-IDS2018 dataset.

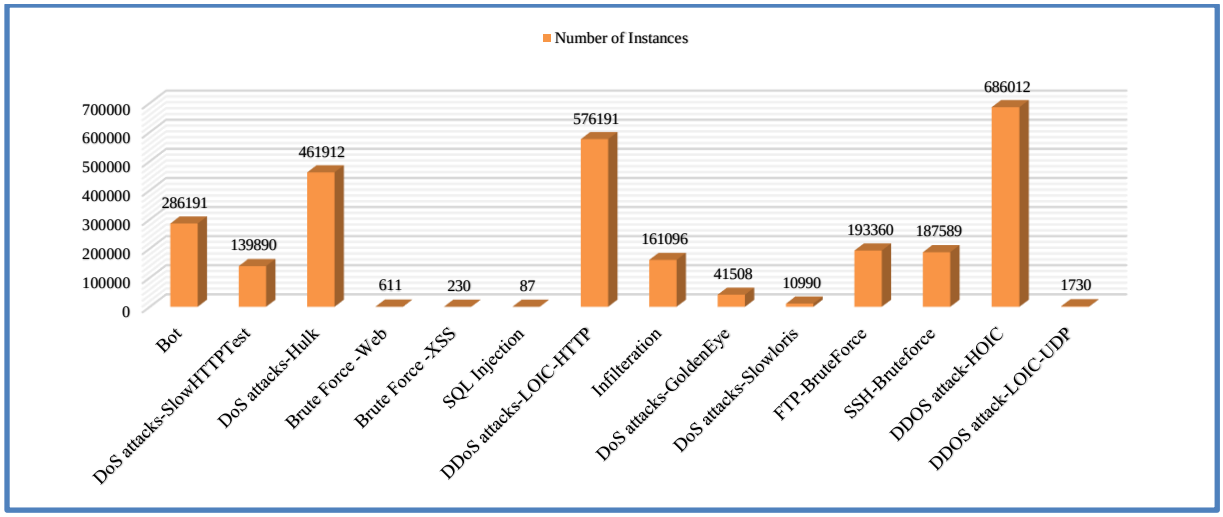


Figure 2.6: Number of instances for attack type in the CIC-IDS2018 dataset.

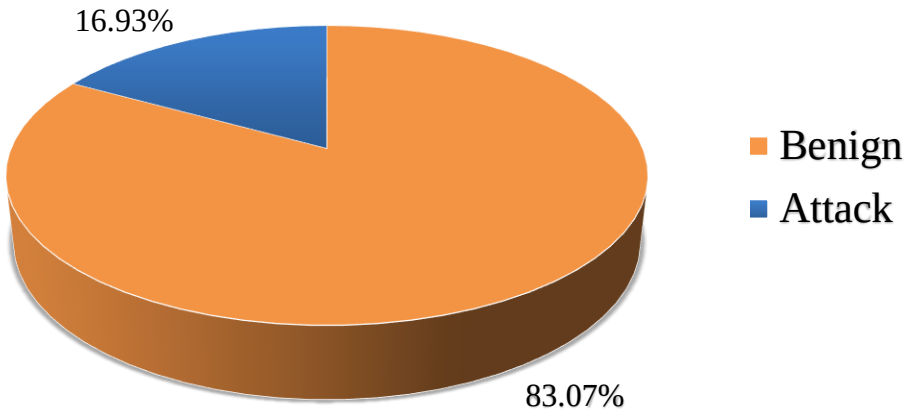


Figure 2.7: The CIC-IDS2018 dataset's distribution of benign and attack instances.

2.8 Comparison of Datasets

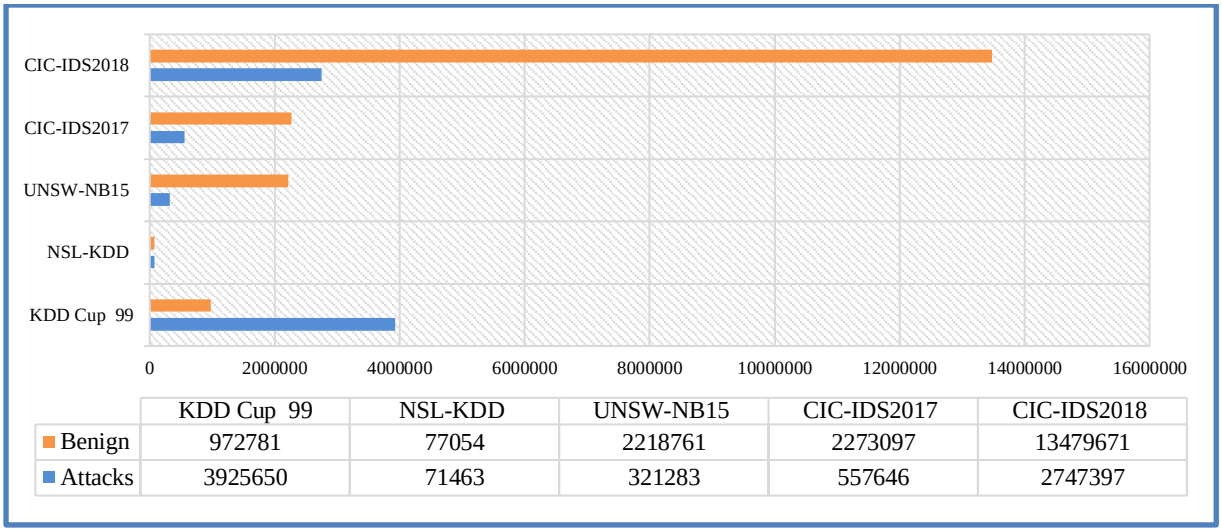
When selecting a dataset for cybersecurity research and development, several crucial factors must be considered. Foremost, the dataset should accurately reflect the current cyber threat environment, encompassing a comprehensive range of attack scenarios and realistic network traffic patterns. It is essential that the dataset maintain a balanced representation of both normal and malicious network activities. Data quality and reliability are paramount considerations, as are the dataset's accessibility and usability.

Based on the analysis in Table 2.6, which presents a comparison between the five datasets mentioned earlier, and figure 2.8 showing the number of benign and attack instances in each dataset. Both the NSL-KDD and KDD Cup 99 datasets, which were created over two decades ago, have revealed significant limitations that render them unsuitable for modern IDS research. Some key issues include imbalanced traffic type distributions, incomplete training sets that fail to capture the full diversity of attacks, this means the model may not learn to

recognize new or uncommon attack types not represented in the training data, a lack of thorough validation procedures, questionable data generation methods, and unrealistically low data rates, as real-world networks and systems typically generate and process large volumes of data at very high speeds [36]. These shortcomings have highlighted the need for more contemporary and representative datasets. The UNSW-NB15 dataset was developed to address these limitations, but evaluations have revealed its own set of challenges. Specifically, some features in the dataset introduce significant noise, yet their contribution to IDS performance is very low, In addition to class imbalances [36]. To overcome these limitations, researchers have turned to more recent datasets, such as CIC-IDS2017 and CIC-IDS2018, which have gained widespread recognition and adoption within the cybersecurity research community. These datasets are considered high-quality and reliable resources, accurately representing real-world network traffic and covering a broad range of modern cyberattacks. Furthermore, the features are exclusive and matchless in comparison with other datasets and are freely available for everyone to use [35]. However, it is important to point out that both datasets exhibit class imbalance, a common challenge in cybersecurity research.

Table 2.6: Comparison between IDS-related datasets.

Name	**KDD cup 99**	**NSL-KDD**	**UNSW-NB15**	**CIC-IDS2017**	**CIC-IDS2018**
year	1999	2009	2015	2017	2018
Separate train/test	No	Yes	No	No	No
Based On	Simulated	Based on KDD Cup 99	Realistic Network	Realistic Network	Realistic Network
Features	42	42	49	79	84
Balance of Classes	Unbalanced	More balanced	Unbalanced	Unbalanced	Unbalanced
Time Span	9 weeks of network traffic	Same as KDD Cup 99	1 month	5 days	10 days
NB of instances	4898431	Train (125973) Test (22544)	2540044	2830743	16227068
Data Quality	High level of redundancy	Redundancies removed	More realistic and diverse	Very high quality	Very high quality
Source	DARPA 1998 IDS	CIC and University of New Brunswick	ACCS	CIC and University of New Brunswick	CIC and CSE
NB Traffic	5	5	10	15	15
Representativeness	Not representative of modern traffic	Slightly more representative than KDD Cup 99	Represents modern traffic	Highly representative of modern traffic	Highly representative of modern traffic
Noise Level	High (contains errors and inconsistencies)	Lower noise than KDD Cup 99	High	Very low	Very low
Common Uses	ML/DL-Based IDS research				



	KDD Cup 99	NSL-KDD	UNSW-NB15	CIC-IDS2017	CIC-IDS2018
Benign	972781	77054	2218761	2273097	13479671
Attacks	3925650	71463	321283	557646	2747397

Figure 2.8: Number of benign and attack instances in each dataset.

2.9 Conclusion

In conclusion, selecting an appropriate dataset is a critical step for effective cybersecurity research and for developing robust IDS. Throughout this chapter, we have analyzed and compared several prominent datasets, including KDD Cup 99, NSL-KDD, UNSW-NB15, CIC-IDS2017, and CIC-IDS2018. While earlier datasets like KDD Cup 99 and NSL-KDD laid the foundation for network IDS research, they have become increasingly outdated and fail to represent modern network traffic patterns and attack scenarios accurately. The UNSW-NB15 dataset aimed to address these limitations but introduced its own set of challenges, such as noisy features and class imbalances. In recent years, the CIC-IDS2017 and CIC-IDS2018 datasets have emerged as the most comprehensive and representative resources for cybersecurity research. That is why we have chosen to focus our study in the next chapter on these two datasets, due to their strengths, while not neglecting their weaknesses, which require preprocessing to mitigate. Indeed, these datasets offer a strong foundation for leveraging advanced ML/DL algorithms in intrusion detection and network security analysis.

CHAPTER 3

Evaluating the Performance of ML and DL Models for IDS

3.1 Introduction

This chapter covers the practical side of our research in the field of IDS using ML and DL techniques. Here we will review the steps we followed in training and evaluating different models, starting from data preprocessing to the result analysis.

We start by preprocessing the CIC-IDS2017 dataset, followed by a detailed explanation of six models' development and training processes. The top five results for each model are highlighted based on validation metrics for both the training set and the CIC-IDS2018 dataset, the latter being a new dataset used for evaluating the trained models. The same preprocessing steps were applied to the CIC-IDS2018 dataset to ensure consistency in the processing across all datasets, enabling a fair and reliable comparison of results.

The main objective is to clarify the practical aspects of applying AI techniques in the field of cyber security and to provide insights into the effectiveness of these techniques in improving IDS.

3.2 Software and tools

This research was conducted using the Python programming language, optimally for ML/DL applications due to its extensive ecosystem of libraries such as TensorFlow, Keras, and Scikit-learn, which offer comprehensive support [53,54]. The development and execution of code were facilitated through diverse programming environments, including Jupyter Notebook, Google Colab, and Visual Studio Code, ensuring flexibility, ease of use, and seamless collaboration. Computational tasks were executed on two different machines: the first, labeled DESKTOP-9BM92I3, runs on Windows 10 Professional, with 8 GB of RAM and an Intel(R) Core (TM) i5-4200M processor clocked at 2.50 GHz. The second machine, DESKTOP-MAQMSA4, also operates on Windows 10 Professional, equipped with 8 GB of RAM and an Intel(R) Core (TM) i5-6300U processor running at 2.40 GHz. These configurations provided sufficient computational power to efficiently perform all necessary calculations and model training.

3.3 Data Pre-processing

The data preprocessing phase represents the beginning of our research journey, and due to its importance, we spent more than 60% of our time in this phase, which took around two months, based on the principle of "Garbage in, Garbage out (GIGO)," a concept that focuses on the importance of the quality of the inputs. More precisely, inputting inaccurate, incorrect, or invalid data (GI) will certainly lead to results that are not reliable, incorrect, invalid, or

useless (GO). The data preprocessing includes the following steps:

- Data Cleaning.
- Handling Imbalanced Data.
- Data Transformation.
- Feature Selection.
- Splitting the Dataset.

3.3.1 Data Cleaning

The data cleansing process is done by:

- Handle infinite values by deleting them or converting them to Nan (Not a Number) values to handle them in the same way as Nan values later.
- Dealing with Nan values can be done in many ways, including removing them (by removing columns or rows) or replacing Nan values with a constant value, calculating the sum of all values and dividing it by the total number of values in the column (Mean), sorting all values in ascending order in the column and selecting the middle value (Median), or the most frequent value in the column (Mode). Also, it's possible to use ML techniques to predict Nan values.
- Dealing with duplicate rows by removing them.
- In our feature analysis, we observed that certain features, such as Duration, Protocol, Type, Service, Flag, and others, must have positive values given their nature. To handle instances where negative values are present, we can consider the following approaches: replacing negative values with zero or one, as they are the closest positive values; using the Mode method to substitute each negative value with the most frequently occurring positive value within the feature; or removing the affected rows or columns altogether.

We tried to follow most of the previous research by deleting rows with Nan and infinite values. The number of Nan and infinite rows was 5734, as we noticed that they are very few and do not exceed 5% of the total data (2,830,743 samples). Therefore, deleting them does not affect data's interpretation, and it also gives credibility to the data, as some methods may provide values that may be incorrect. After that, we deleted the duplicate rows, which were 307,078 samples.

During the feature analysis, we identified 13 features containing negative values, accounting for a total of 2,130,167 entries. Given that these represent a significant portion of the dataset, removing them was not a viable option. Initially, we considered eliminating two specific

features: "Init_Win_bytes_forward" (with 911,012 entries) and "Init_Win_bytes_backward" (with 1,215,625 entries); this led to the removal of 99.8% of negative values. However, recognizing their importance for model training [37], we opted for an alternative solution.

Instead of deletion, we decided to convert the negative values to their absolute values for all affected features. This decision was based on our assumption that these entries hold meaningful information. For instance, in cases like time differences (e.g., 2-5 = -3), the absolute value accurately reflects the magnitude without altering the underlying significance. This approach ensures that the model can leverage important quantitative information without being adversely affected by the sign of the value.

By preserving these features in absolute form, we maintain the integrity of the dataset while addressing the issue of negative values. The forthcoming performance results of our models will serve to validate this hypothesis, demonstrating whether this method effectively retains the relevant information for accurate predictions.

3.3.2 Handling Imbalanced Data

Secondly, we noticed that one of the categories significantly surpasses the other in terms of number, and to address this imbalance there are many methods, of which we will mention some of them:

Oversampling:

A common method for balancing the dataset is the Oversampling technique, which aims to increase the number of cases belonging to the underrepresented class (as shown in Figures 3.1), which is done in two main ways:

- ✓ Doubling existing samples.
- ✓ Creating new artificial data points.

This can be done using techniques like SMOTE (Synthetic Minority Oversampling Technique), random sampling, Adaptive Synthetic Sampling (ADASYN), et cetera [38].

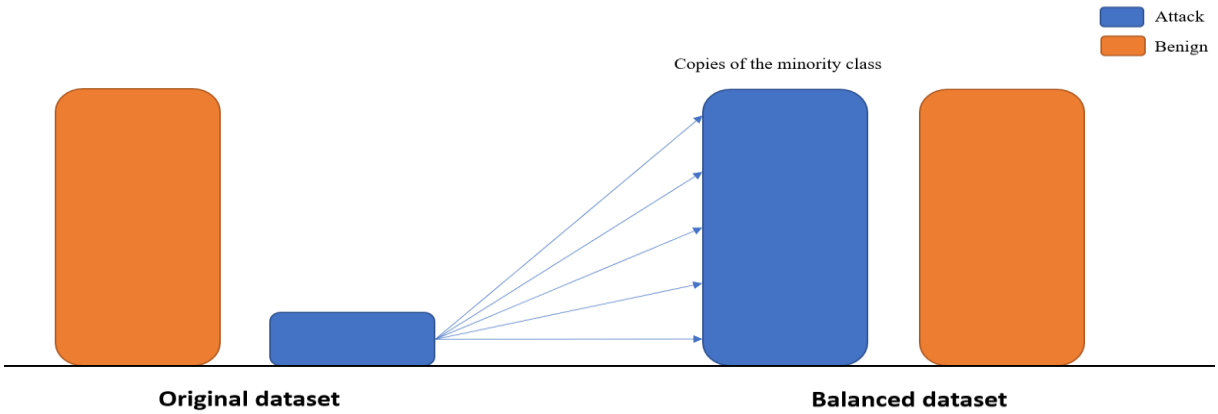


Figure 3.1: Balancing data using Oversampling technique.

Undersampling:

When dealing with a large dataset, researchers use undersampling. It is a technique based on the strategy of reducing the number of most representative samples, ensuring that the original data is accurately represented in the underrepresented group (as shown in Figure 3.2). This can be done using several techniques. One approach is the RandomUnderSampler technique, which reduces in a random way. Additionally, there are several more selective methods, such as Tomek Links, Edited Nearest Neighbors (ENN), Cluster-based Undersampling, et cetera [39].

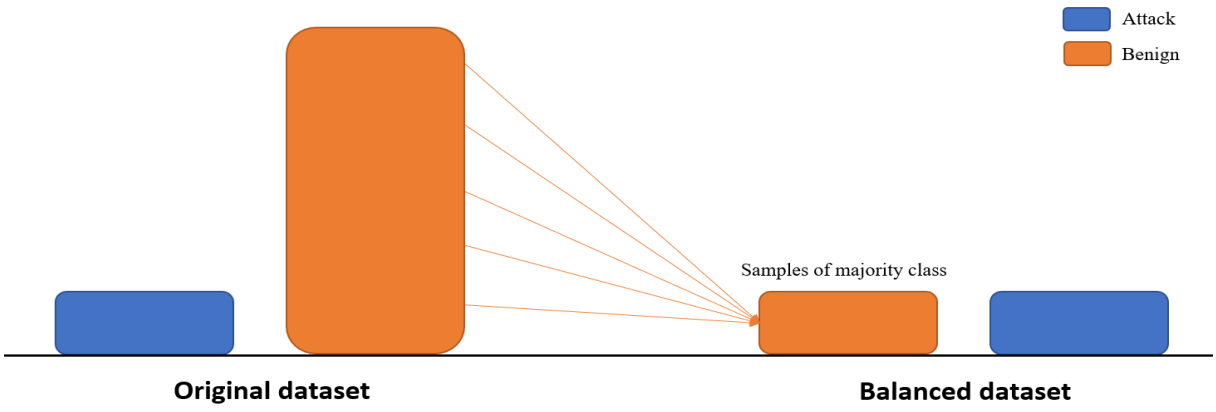


Figure 3.2: Balancing data using Undersampling technique.

Following an extensive review of relevant literature and a thorough analysis of the dataset [40], we decided to complete our study by balancing the data, where we first consolidated similar attack types (such as "DoS Hulk," "DoS GoldenEye," "DoS Slowloris," and "DoS Slowhttptest") under a single category ("DoS"). This step aimed to reduce the complexity of

the classification task. Next, we applied a threshold-based selection criterion, retaining only those classes with more than 10,000 instances (as seen in Figure 3.3).

The traffic type detection process was implemented in two scenarios: first, binary classification, where data was categorized as either attack or benign, and second, multi-class classification, where data was initially classified as benign or attack. The detected attacks were then further classified into three different attack families ("DoS," "DDoS," and "PortScan"). Data imbalance was addressed differently for each scenario, tailored to the specific classification approach used.

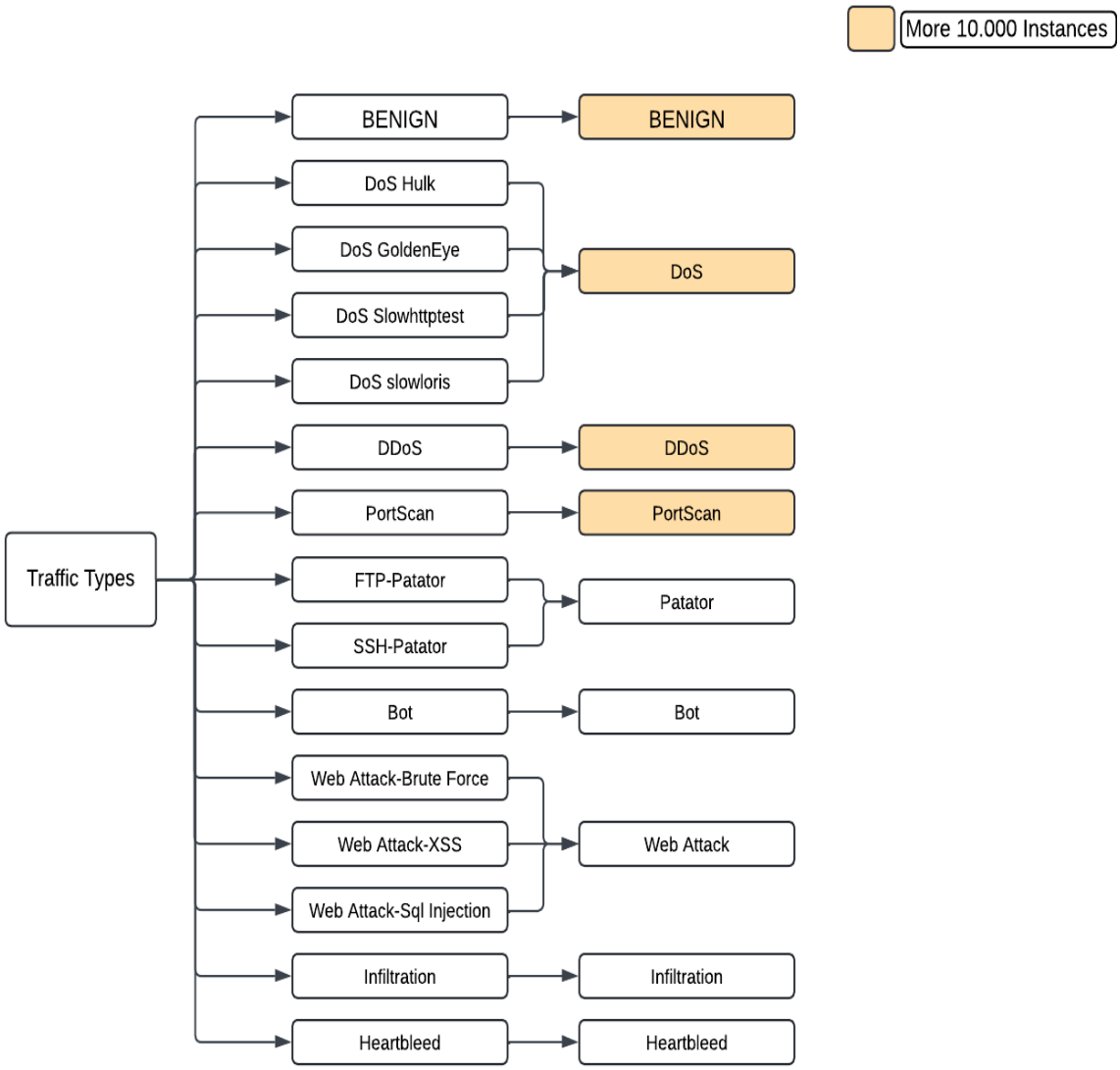


Figure 3.3: Renamed and selected classes.

Data Balancing in Binary Classification:

First scenario, we calculated the sum of the attack instance. After that, we noticed that the number of "Benign" instances is more than the number of the "Attack" instances, so we used the RandomUnderSampler algorithm to reduce the number of benign instances from 2,095,075 to 401,840 until it equals the sum of the attack. Figure 3.4 helps to better understand:

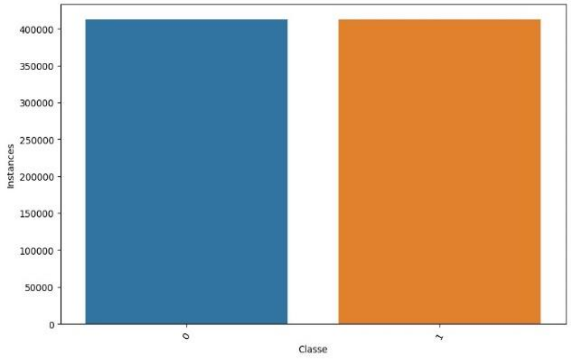


Figure 3.4: Distribution of Binary classification dataset.

Data Balancing in Multi-Class Classification:

In the second scenario, we also used RandomUnderSampler algorithm to reduce the number of benign instances to 200,000 while keeping the number of instances of attack types the same, as represented in figure 3.5:

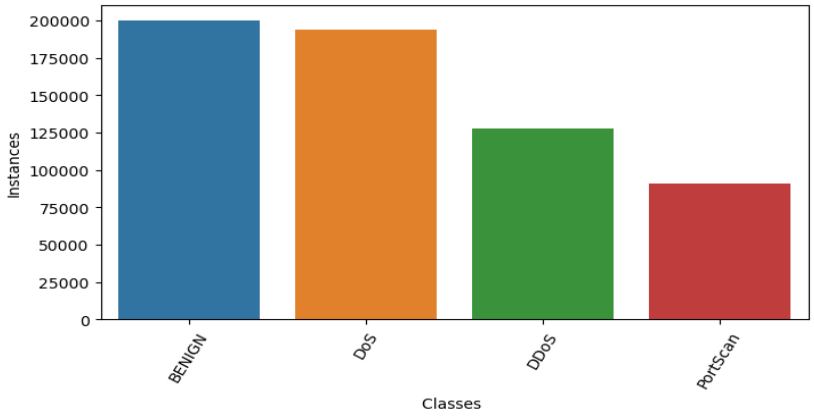


Figure 3.5: Distribution of Multi-Class Classification dataset.

3.3.3 Data Transformation

Since the data contains a huge variety of formats, we moved to the 3rd step, where we encode and normalize the data. At this step, we focus on:

3.3.3.1 Categorical Data Encoding

It is a technique used to convert categorical data into numbers. In this process, each unique class is assigned a unique number. This transformation makes the textual data suitable for ML and DL algorithms that require numeric inputs [41]. Table 3.1 gives an overview of LabelEncoder and OneHotEncoder.

In the context of binary classification, we assigned a value of 0 for benign traffic and 1 for attack traffic. For multi-class classification, we employed both OneHotEncoding and LabelEncoding methods, applying them to different types of models.

OneHotEncoding was utilized for DL models because it effectively handles unordered categories, particularly when dealing with a limited number of classes (in this case, 4). This method creates a binary column for each category, which aligns well with the input requirements of neural networks, allowing them to interpret each category as a distinct entity.

On the other hand, LabelEncoding was applied because it efficiently converted categorical values into numerical form. This method is particularly advantageous for ML models, such as tree-based algorithms, as it simplifies the data representation and facilitates easier processing and interpretation by the models.

By employing both methods, we aimed to leverage their respective strengths and assess their performance in our specific context, ensuring that we selected the most effective approach for our multi-class classification tasks.

Table 3.1: Comparison between LabelEncoder and OneHotEncoder.

Feature/Technique	**LabelEncoder**	**OneHotEncoder**
Description	Assigns a unique integer to each different category in a single column.	Converts each unique category to binary columns. For each entry, it puts 1 in the column corresponding to its category, while filling the remaining columns with 0.
Dim Input Type	1 dimensional array-like structure.	1-dimensional array-like structure.
Dim Output Type	1 dimensional integer array.	2-dimensional binary array.
Example Encoding Type	BENIGN >> 0 DoS >> 1 DDoS >> 2	BENIGN >> [0, 0, 1] DoS >> [0, 1, 0] DDoS >> [1, 0, 0]
Use case	Particularly useful for ordinal data where the order of categories matters (e.g., low, medium, high).	Useful for nominal data where the categories do not have an intrinsic order (e.g., types of attacks).
Advantages	-Simple and efficient for ordinal data. -Does not increase dimensionality.	-It prevents the model from making any category ordering.

3.3.3.2 Data Normalization

The model training process faces difficulties when using features with varying measures. This can lead to complex and very time-consuming training, with the potential for model failure. To address this issue, researchers use standardization techniques that aim for a uniform data format [40,42]. Here are some of the common data normalization methods:

- **Min-Max Normalization:**

Linear normalization is one of the most flexible and simple normalization techniques. It consists of creating a new reference rule for each data point, which ranges from 0 to 1 [43]. Here's the formula for linear normalization:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where X_{max} and X_{min} are the minimum and maximum values of the feature X.

- **Z-Score Normalization (Standardization):**

This method scales the data so that its mean (μ) is 0 and its standard deviation (σ) is 1, by subtracting the mean from each data point and then dividing by the standard deviation. The formula is:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

We opted for the Min-Max Normalization method to standardize the data range, which enhances model performance and ensures consistent scaling across all features.

3.3.4 Feature Selection

Our efforts continue to prepare the data to reach the pre-final step. Through our series of experiments, we considered feature selection as one of the most important steps due to its role in increasing accuracy, reducing overfitting, speeding up training, and avoiding noise. This process aims to identify the most important features in the dataset by removing duplicate, irrelevant, or redundant ones, etc. A variety of methods can be used to select appropriate features, including classification models in both ML and DL [44].

By removing highly correlated and constant features, we reduced the number of features to 41. Although these features performed well in the training phase, we faced challenges in generalizing the model to new data.

Therefore, we resorted to a different approach using the Random Forest technique, with

random_state on 42 and n_estimators on 1,000 [45]. The model was trained on binary classification, and then we set four different thresholds (0.02, the mean, 0.01, and 0.005) where the number of features selected (16, 23, 28, and 40, respectively per threshold). Using only 16 features shown in Figure 3.6, we achieved satisfactory results in model performance with binary classification. We continued to work with the same methodology and the same threshold, but this time on a trained model of multiple classification where the number of features selected (20, 30, 33, and 46, respectively per threshold). In the multi-classification process, we relied on the 30 features shown in Figure 3.7 because they also provided the best results, while the feature 'Fwd Header Length.1' was deleted due to its absence in the CIC-IDS2018 datasets. Table 3.2 helps for a better understanding:

Table 3.2: Selected features using random forest in both classifications.

thresholds	0.02	mean	0.01	0.005
Number of features selected in the binary classification	16	23	28	40
Number of features selected in the multi-class classification	20	30	33	46

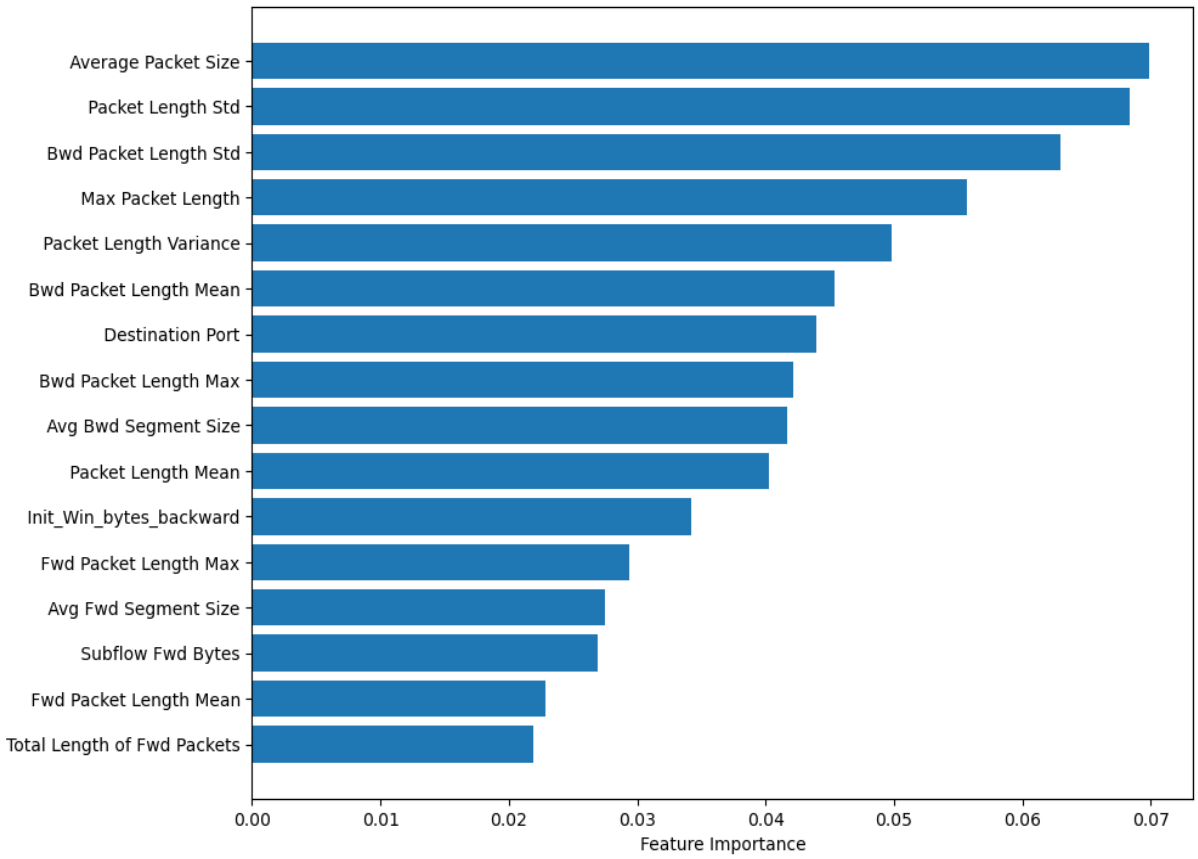


Figure 3.6: The 16 selected features using the random forest of binary classification.

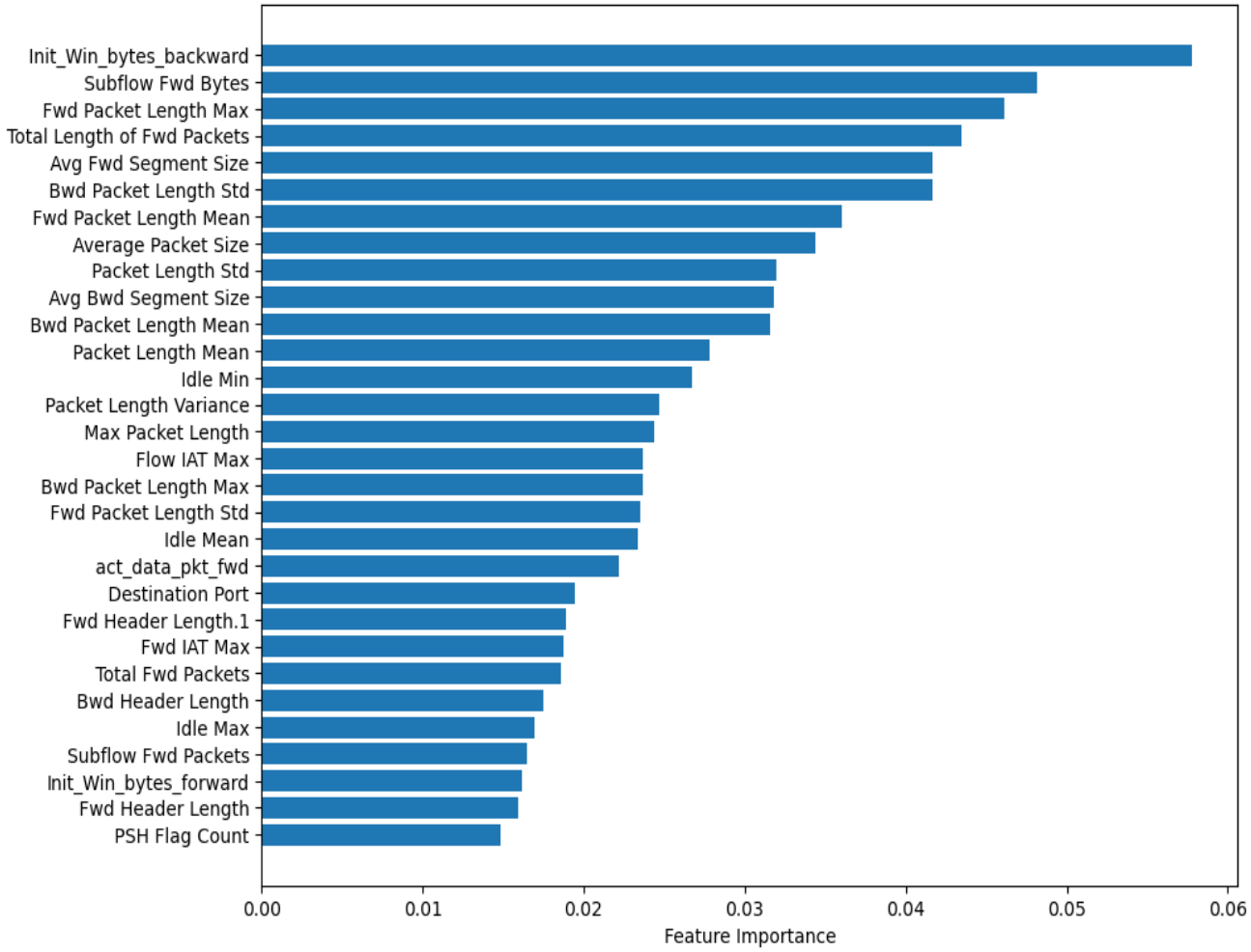


Figure 3.7: The 30 selected features using the random forest of multi-class Classification.

3.3.5 Splitting the Dataset

We finish the preprocessing of data by splitting the available dataset into two different sets, either training and validation and testing, or training and testing only, resulting in:

- ✓ Training set: Used to train the model.
- ✓ Validation set: Helps adjust the model.
- ✓ Test set: ensure that the model's performance is accurately evaluated on data on which it has not been trained [46].

Table 3.3 presents some common methods to split the dataset:

Table 3.3: Some known methods of data splitting.

Name	**Method of splitting**
Random splitting	Splitting data randomly
Stratified Splitting	Ensures equal distribution of classes with an unbalanced dataset.
Time Series Splitting	Dividing dataset while keeping the chronological order.
Shuffle Split	Splitting the database several times randomly, enabling different groups to be formed each time.

Most recent research relies on splitting the data into training, validation, and testing, such as (80%, 10%, 10%) or (75%, 10%, 15%), etc. As for our methodology, we adopted the stratification method suitable for the nature of our dataset, where we split the dataset to 70% for training, 10% for validation, and 20% for testing, which was adopted in the article [47].

3.4 Classification Models

After successfully completing the data processing, we are now on the threshold of a new stage. In this stage, we will analyze six different models: three in the ML field and three in the DL field. To enhance the credibility and effectiveness of these models, we generalize them to the CIC-IDS2018 dataset, taking into consideration binary and multi-class classification.

The term "prediction set" is used to refer to the dataset (CIC-IDS2018) used for evaluating our trained models. While a test set is typically a portion of the original dataset (CIC-IDS2017) held out during training, our prediction set is an entirely separate dataset.

To evaluate trained models, we applied various packet selection methodologies from the prediction set. Upon analyzing the comparative Table 3.4 between the test and prediction sets, we noticed that similar packets in the attack category belong to "DoS" types. Based on this observation, we used in the multi-class classification on 196,568 instances of "DoS" attacks and 200,000 instances of benign for the prediction set.

For binary classification, we expanded the test to cover both "DoS" and "DDoS" attacks, aiming to distinguish between benign and attacks for the prediction set, even when the model hasn't been extensively trained on specific attack types. To achieve this, we used a balanced sample of 159,397 instances each for benign and attack cases for the prediction set. This comprehensive approach to model testing aims to evaluate their performance under a variety of conditions, providing deeper insight into each model's capabilities in handling unseen traffic types and security threats.

Table 3.4: Traffic types used in both test set and prediction set.

Dataset	**CIC-IDS2017**	**CICIDS2018**
Traffic Types	'BENIGN'	'BENIGN'
	'DoS Hulk', 'DoS GoldenEye', 'DoS slowloris', 'DoS Slowhttptest'	'DoS attacks-Hulk', 'DoS attacks-GoldenEye', 'DoS attacks-Slowloris', 'DoS attacks-SlowHTTPTest'
	'DDoS'	'DDOS attack-HOIC', 'DDOS attack-LOIC-UDP'
	'PortScan'	/

3.4.1 Validation Metrics of ML & DL

To understand model forces, identify weaknesses, optimize, and compare, we use confusion matrices and performance evaluation measures. In this section, we will explain these measures and how to use them in a simple way.

3.4.1.1 Confusion matrix

In the context of AI models evaluation, a confusion matrix in tabular form is used as a metric to evaluate the classification accuracy. It allows monitoring the performance of the model and provides a summary of the model's predictions compared to the actual results [33]. The confusion matrix in Table 3.5 consists of four elements for binary classification:

- **TP (True Positives)**: Correctly predicted positive instances.
- **TN (True Negatives)**: Correctly predicted negative instances.
- **FP (False Positives)**: When the classifier identifies the true negative label as positive.
- **FN (False Negatives)**: When the classifier identifies the true positive label as negative.

There is a well-known understanding in IDS research that a correct classification of an attack event is defined as TP, while a correct classification of a benign event is defined as TN. Errors can occur in the classification process. When a benign event is classified as an attack, it is called FP. Conversely, if an attack event is classified as benign, it is called FN [48].

Table 3.5: Confusion Matrix.

		Predicted Class	
		+	-
Class	+	***TP***	***FN***
	-	***FP***	***TN***

3.4.1.2 Performance Evaluation Metrics

This part focuses on explaining the model performance evaluation metrics listed below, which are defined based on TP, TN, FP, and FN.

a. Accuracy is a metric that expresses the sum of correct predictions (TP & TN) made by the model over the total number of predictions. It is worth noting that this metric is most effective when the data is balanced and biased in the case of unbalanced data.

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

b. Precision is a metric that determines the division of the number of TP predictions by the sum of the positive predicted values. Where the precision takes a low value if the number of FP is large.

$$Precision = \frac{TP}{TP + FP}$$

c. Recall, Sensitivity, or TPR (True Positive Rate) is a metric that gauges the model's ability to identify all TP cases. Divide the TP by the sum of both TP and FN predictions to calculate recall. If the result is low, this indicates a large number of FN.

$$Recall = \frac{TP}{TP + FN}$$

d. F1-Score or F-Measure is a metric that provides a good balance of precision and recall by calculating the harmonic mean between them, which makes it valuable in many classification tasks.

$$F1 - Score = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

3.4.2 ML Classification Models

ML models are extremely effective in IDS due to their ability to analyze large amounts of network data in a short time and automatically detect potential security threats. These models can detect known attacks much faster than traditional systems. making them a powerful tool in enhancing cybersecurity.

3.4.2.1 Extreme Gradient Boosting Model

Extreme Gradient Boosting (XGBoost) is an algorithm that is highly flexible and efficient at processing large datasets. It combines the power of Decision Trees (DT) with the effectiveness of incremental boosting, making it highly scalable. The main advantage of XGBoost is its unique ability to optimize the objective function. It does this through a systematic process of loss minimization, with an exclusive focus on using decision trees as the

basis for classification. This combination of techniques gives XGBoost superior performance on a wide range of ML tasks.

The following Tables 3.6 and 3.7 show the main hyperparameters used in XGBoost models:

Table 3.6: Model settings used in XGBoost binary classification.

	learning_rate	**n_estimators**	**max_depth**	**scale_pos_weight**	**reg_alpha**	**reg_lambda**
Test_1	0.009	900	7	9	0.1	0.0
Test_2	0.009	900	6	9	0.0	0.0
Test_3	0.009	1000	5	7	0.0	0.0
Test_4	0.009	800	5	7	0.0	0.0
Test_5	0,008	800	5	7	0.0	0.0

Table 3.7: Model settings used in XGBoost multi-class classification.

	learning_rate	**n_estimators**	**max_depth**	**min_child_weight**	**subsample**	**scale_pos_weight**
Test_6	0.006	200	7	9	0.5	7
Test_7	0.008	260	7	22	0.5	7
Test_8	0.005	300	7	25	0.5	7
Test_9	0.006	250	7	9	0.2	7
Test_10	0.002	220	8	30	/	7

XGBoost testing results:

Tables 3.8 and 3.9 illustrate the XGBoost algorithm's effectiveness in both classification tasks for the test and prediction sets, complemented by confusion matrices displayed in Figures 3.8 to 3.11 for both classifications for test set.

Table 3.8: Results of XGBoost models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	99.3890	0.0224	0.9995	0.9882	0.9939	97.6798	0.9778	0.9757	0.9768
	Attack			0.9884	0.9995	0.9939		0.9758	0.9779	0.9768
Test_2	BENIGN	99.3551	0.0236	0.9995	0.9876	0.9935	97.7114	0.9784	0.9757	0.9771
	Attack			0.9877	0.9995	0.9936		0.9758	0.9785	0.9771
Test_3	BENIGN	99.2866	0.0245	0.9995	0.9862	0.9928	98.7063	0.9985	0.9756	0.9869
	Attack			0.9864	0.9995	0.9929		0.9762	0.9985	0.9872
Test_4	BENIGN	99.2151	0.0277	0.9995	0.9848	0.9921	98.1857	0.9986	0.9651	0.9815
	Attack			0.9850	0.9995	0.9922		0.9662	0.9987	0.9822
Test_5	BENIGN	99.2042	0.0296	0.9993	0.9847	0.9920	97.9679	0.9941	0.9651	0.9794
	Attack			0.9850	0.9993	0.9921		0.9660	0.9943	0.9800

Table 3.9: Results of XGBoost models in multi-class classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	99.8375	0.3447	0.9981	0.9973	0.9977	95.7805	0.9976	0.9296	0.9624
	DDoS			0.9998	0.9986	0.9992		/	/	/
	DoS			0.9979	0.9991	0.9985		0.9328	0.9865	0.9589
	PortScan			0.9980	0.9987	0.9984		/	/	/
Test_7	BENIGN	99.8343	0.1428	0.9980	0.9973	0.9976	95.4777	0.9967	0.9247	0.9594
	DDoS			0.9996	0.9986	0.9991		/	/	/
	DoS			0.9980	0.9990	0.9985		0.9285	0.9854	0.9561
	PortScan			0.9980	0.9987	0.9983		/	/	/
Test_8	BENIGN	99.8212	0.2544	0.9981	0.9972	0.9976	95.5251	0.9980	0.9244	0.9598
	DDoS			0.9992	0.9986	0.9989		/	/	/
	DoS			0.9978	0.9991	0.9985		0.9283	0.9866	0.9566
	PortScan			0.9980	0.9979	0.9979		/	/	/
Test_9	BENIGN	99.8196	0.2544	0.9981	0.9971	0.9976	95.5347	0.9980	0.9246	0.9599
	DDoS			0.9991	0.9986	0.9989		/	/	/
	DoS			0.9978	0.9992	0.9985		0.9284	0.9866	0.9566
	PortScan			0.9980	0.9979	0.9979		/	/	/
Test_10	BENIGN	99.8033	0.7909	0.9973	0.9973	0.9973	92.8484	0.9978	0.8711	0.9302
	DDoS			0.9993	0.9981	0.9987		/	/	/
	DoS			0.9976	0.9987	0.9982		0.8836	0.9868	0.9324
	PortScan			0.9988	0.9980	0.9984		/	/	/

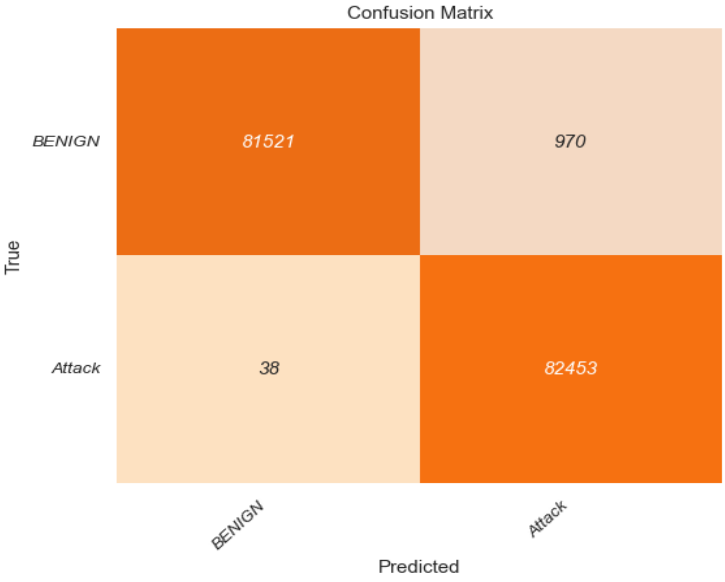


Figure 3.8: Confusion matrix for XGBoost Test_1.

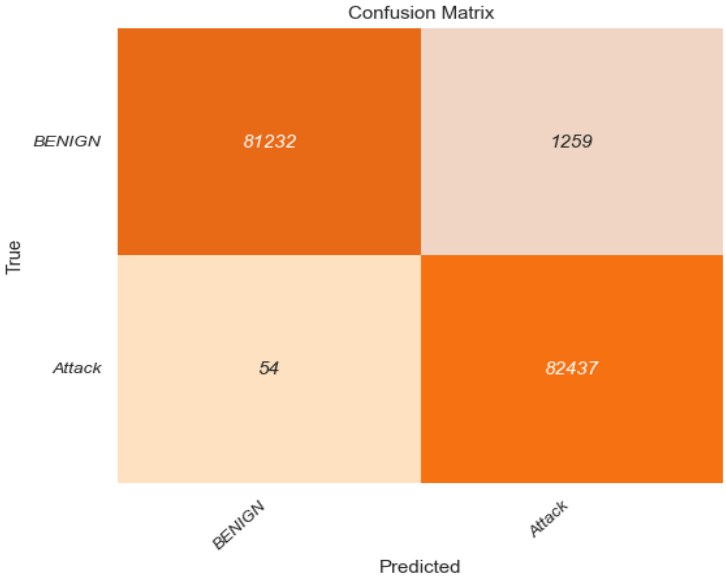


Figure 3.9: Confusion matrix for XGBoost Test_5.

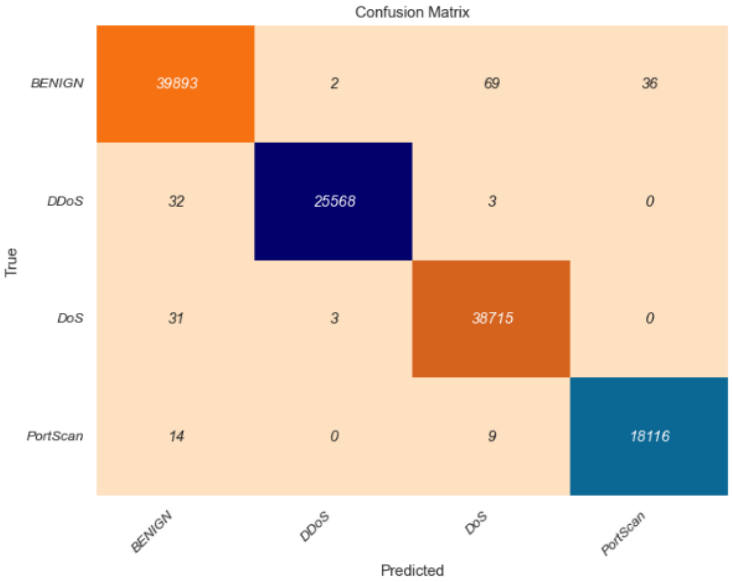


Figure 3.10: Confusion matrix for XGBoost Test_6.

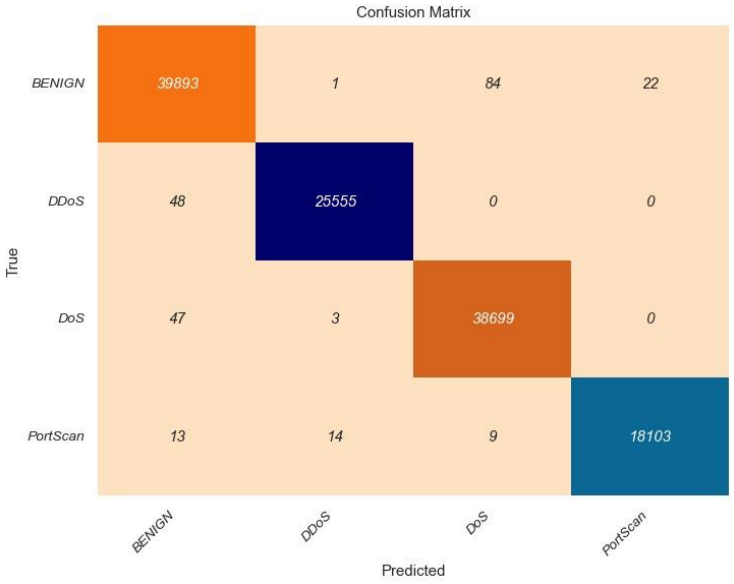


Figure 3.11: Confusion matrix for XGBoost Test_10.

3.4.2.2 Decision Tree Model

DT is a ML model that uses specific rules to make decisions. This makes it suitable for tackling a wide range of real-world problems. This technique excels in classification and regression tasks, and is considered one of the most reliable tools in the field of AI, as it is characterized by its ability to provide easily interpretable results, while requiring relatively limited resources for training and implementation.

Primary hyperparameters for DT models are displayed in the Tables 3.10 and 3.11:

Table 3.10: Model settings used in DT binary classification.

	criterion	splitter	max_depth	min_samples_split	min_samples_leaf
Test_1	entropy	random	48	28	3
Test_2			60	25	5
Test_3			120	90	7
Test_4			50	45	4
Test_5			120	120	9

Table 3.11: Model settings used in DT multi-class classification.

	criterion	splitter	max_depth	min_samples_split	min_samples_leaf
Test_6	entropy	best	75	20	7
Test_7			55	49	5
Test_8			50	45	4
Test_9			56	47	4
Test_10			40	35	7

DT testing results:

Tables 3.12 and 3.13 showcase the results achieved by the DT algorithm in tackling both classification tasks for test and prediction sets, while Figures 3.12 to 3.15 provide additional details, which contain confusion matrices for test set.

Table 3.12: Results of DT models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	99.4709	0.0264	0.9981	0.9913	0.9947	96.4693	0.9806	0.9482	0.9641
	Attack			0.9913	0.9982	0.9947		0.9498	0.9812	0.9653
Test_2	BENIGN	99.4624	0.0282	0.9979	0.9913	0.9946	97.3276	0.9806	0.9656	0.9731
	Attack			0.9914	0.9979	0.9946		0.9662	0.9809	0.9735
Test_3	BENIGN	99.4442	0.0253	0.9979	0.9910	0.9944	98.0316	0.9940	0.9664	0.9800
	Attack			0.9910	0.9979	0.9945		0.9673	0.9942	0.9806
Test_4	BENIGN	99.4357	0.0279	0.9979	0.9908	0.9943	96.9988	0.9936	0.9460	0.9693
	Attack			0.9909	0.9979	0.9944		0.9485	0.9939	0.9707
Test_5	BENIGN	99.4254	0.0248	0.9980	0.9905	0.9942	96.0508	0.9841	0.9362	0.9595
	Attack			0.9905	0.9980	0.9943		0.9391	0.9848	0.9614

Table 3.13: Results of DT models in multi-class classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	99.9445	0.0113	0.9996	0.9990	0.9993	95.9480	0.9850	0.9344	0.9590
	DDoS			0.9997	0.9998	0.9998		/	/	/
	DoS			0.9994	0.9996	0.9995		0.9383	0.9850	0.9611
	PortScan			0.9988	0.9997	0.9993		/	/	/
Test_7	BENIGN	99.9420	0.0101	0.9994	0.9990	0.9992	97.6213	0.9855	0.9676	0.9765
	DDoS			0.9999	0.9998	0.9999		/	/	/
	DoS			0.9994	0.9995	0.9995		0.9698	0.9850	0.9774
	PortScan			0.9987	0.9996	0.9991		/	/	/
Test_8	BENIGN	99.9412	0.0103	0.9994	0.9990	0.9992	97.6186	0.9855	0.9675	0.9764
	DDoS			0.9999	0.9998	0.9999		/	/	/
	DoS			0.9994	0.9995	0.9994		0.9697	0.9850	0.9773
	PortScan			0.9987	0.9996	0.9992		/	/	/
Test_9	BENIGN	99.9404	0.0101	0.9994	0.9990	0.9992	97.6186	0.9855	0.9675	0.9764
	DDoS			0.9999	0.9998	0.9999		/	/	/
	DoS			0.9994	0.9995	0.9994		0.9697	0.9850	0.9773
	PortScan			0.9987	0.9996	0.9991		/	/	/
Test_10	BENIGN	99.9380	0.0105	0.9994	0.9990	0.9992	95.9318	0.9850	0.9341	0.9589
	DDoS			0.9997	0.9998	0.9998		/	/	/
	DoS			0.9994	0.9994	0.9994		0.9383	0.9850	0.9611
	PortScan			0.9987	0.9996	0.9992		/	/	/

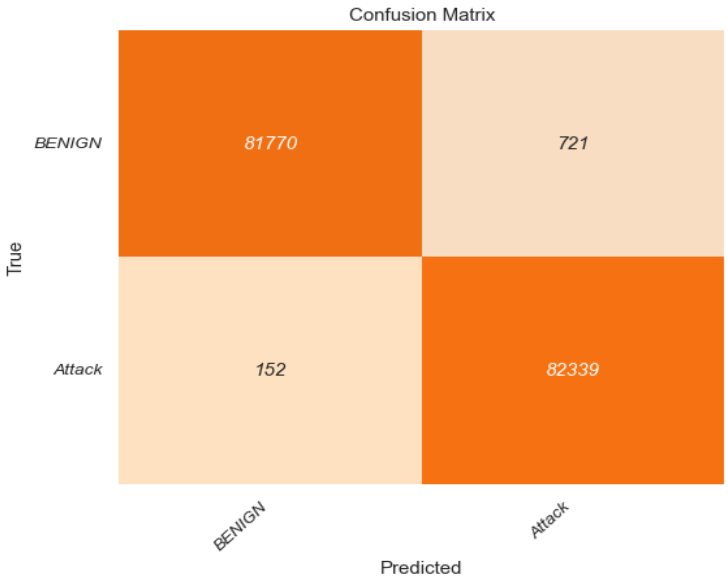


Figure 3.12: Confusion matrix for DT Test_1.

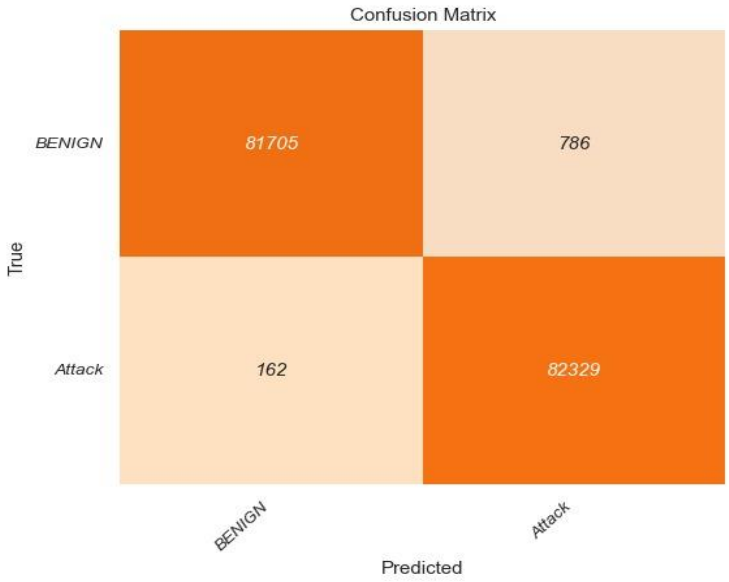


Figure 3.13: Confusion matrix for DT Test_5.

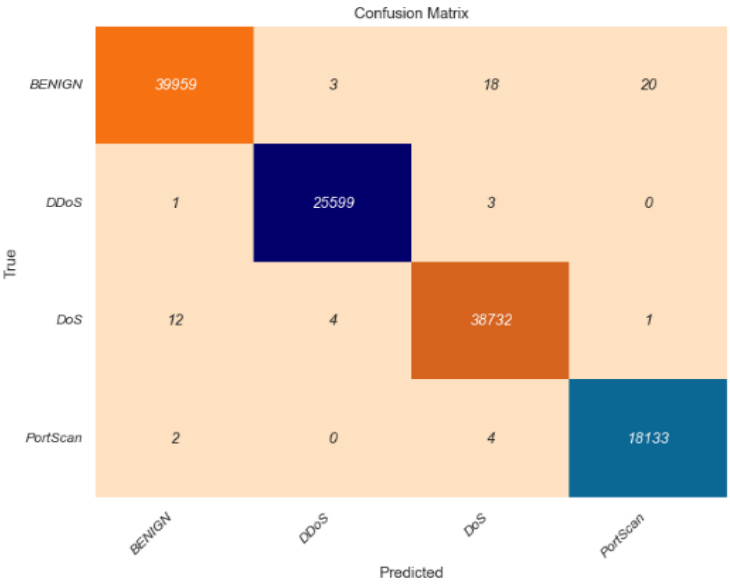


Figure 3.14: Confusion matrix for DT Test_6.

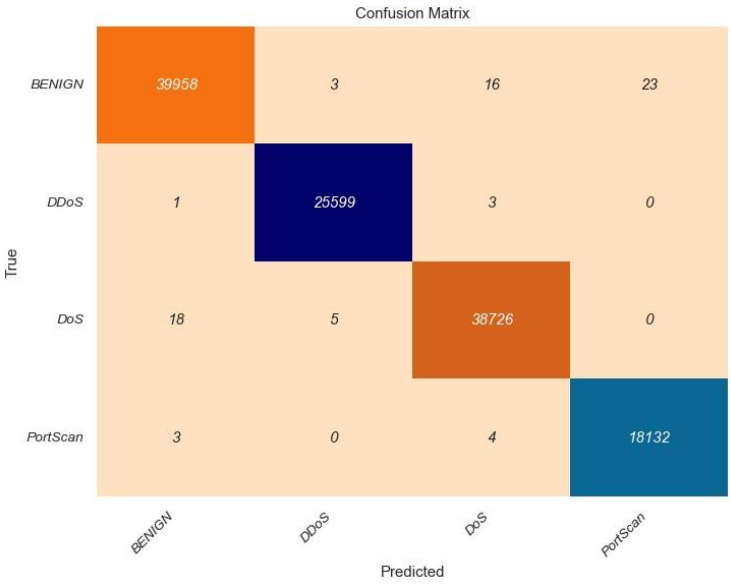


Figure 3.15: Confusion matrix for DT Test_10.

3.4.2.3 Categorical Boosting Model

Categorical Boosting (CatBoost) is an advanced gradient boosting algorithm developed by Yandex, specifically designed to efficiently process categorical features. This makes it an ideal tool for a variety of ML tasks, outperforming traditional methods. CatBoost shares with XGBoost and LightGBM the use of ensemble learning, where it combines predictions from several simple models to form a powerful predictive model.

The Tables 3.14 and 3.15 below provide detailed information on the key hyperparameters

employed in the development of CatBoost models:

Table 3.14: Model settings used in CatBoost binary classification.

	iterations	learning_rate	depth	l2_leaf_reg	border_count	bagging_temperature
Test_1	265	0.05	6	5	/	/
Test_2	265	0.03	7	14	128	/
Test_3	175	0.09	7	28	150	
Test_4	350	0.01	9	28	64	0,2
Test_5	250	0.01	9	34	64	/

Table 3.15: Model settings used in CatBoost multi-class classification.

	iterations	learning_rate	depth	l2_leaf_reg	early_stopping_rounds	border_count
Test_6	222	0.05	7	14	50	128
Test_7	175	0.09	7	28	50	150
Test_8	235	0.03	7	14	50	128
Test_9	265	0.03	6	3	50	128
Test_10	250	0.02	7	14	50	64

CatBoost testing results:

The performance of the CatBoost algorithm on the two classification tasks can be seen in Tables 3.16 and 3.17 for the test and prediction sets, while Figures 3.16 to 3.19 present confusion matrices for Test_1, Test_5, Test_6, and Test_10 for test set.

Table 3.16: Results of CatBoost models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	99.5084	0.0151	0.9987	0.9915	0.9951	98.4184	0.9925	0.9758	0.9840
	Attack			0.9916	0.9987	0.9951		0.9762	0.9926	0.9843
Test_2	BENIGN	99.4987	0.0159	0.9985	0.9914	0.9950	97.8660	0.9807	0.9766	0.9786
	Attack			0.9915	0.9985	0.9950		0.9767	0.9808	0.9787
Test_3	BENIGN	99.4636	0.0178	0.9981	0.9912	0.9946	98.2473	0.9941	0.9707	0.9823
	Attack			0.9912	0.9981	0.9947		0.9714	0.9942	0.9827
Test_4	BENIGN	99.4448	0.0184	0.9979	0.9910	0.9944	98.4301	0.9935	0.9749	0.9842
	Attack			0.9911	0.9979	0.9945		0.9754	0.9937	0.9844
Test_5	BENIGN	99.4018	0.0209	0.9978	0.9902	0.9940	98.2943	0.9935	0.9722	0.9828
	Attack			0.9903	0.9978	0.9940		0.9728	0.9937	0.9831

Table 3.17: Results of CatBoost models in multi-class classification.

Dataset		CIC-IDS2017					CIC-IDS2018			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	99.8922	0.0066	0.9992	0.9980	0.9986	91.8241	0.8802	0.9699	0.9229
	DDoS			0.9997	0.9993	0.9995		/	/	/
	DoS			0.9982	0.9996	0.9989		0.9658	0.8657	0.9130
	PortScan			0.9987	0.9988	0.9988		/	/	/
Test_7	BENIGN	99.8849	0.0060	0.9992	0.9977	0.9985	96.7400	0.9707	0.9645	0.9676
	DDoS			0.9998	0.9993	0.9996		/	/	/
	DoS			0.9979	0.9997	0.9988		0.9641	0.9704	0.9672
	PortScan			0.9987	0.9989	0.9988		/	/	/
Test_8	BENIGN	99.8359	0.0111	0.9988	0.9972	0.9980	96.4833	0.9994	0.9309	0.9639
	DDoS			0.9991	0.9990	0.9991		/	/	/
	DoS			0.9975	0.9995	0.9985		0.9344	0.9994	0.9658
	PortScan			0.9981	0.9976	0.9978		/	/	/
Test_9	BENIGN	99.8122	0.0109	0.9982	0.9966	0.9974	96.4919	0.9992	0.9312	0.9640
	DDoS			0.9995	0.9988	0.9992		/	/	/
	DoS			0.9972	0.9991	0.9981		0.9347	0.9992	0.9659
	PortScan			0.9979	0.9985	0.9982		/	/	/
Test_10	BENIGN	99.7681	0.0201	0.9981	0.9959	0.9970	96.2644	0.9995	0.9264	0.9616
	DDoS			0.9989	0.9986	0.9988		/	/	/
	DoS			0.9963	0.9989	0.9976		0.9305	0.9996	0.9638
	PortScan			0.9980	0.9975	0.9978		/	/	/

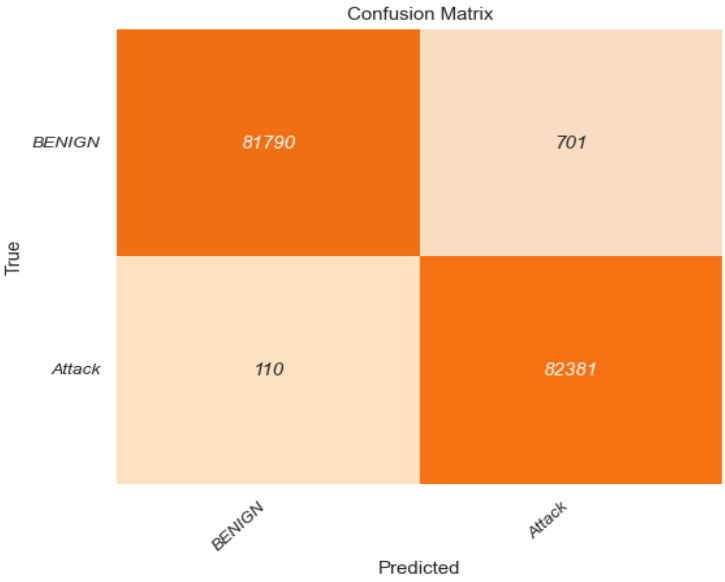


Figure 3.16: Confusion matrix for CatBoost Test_1.

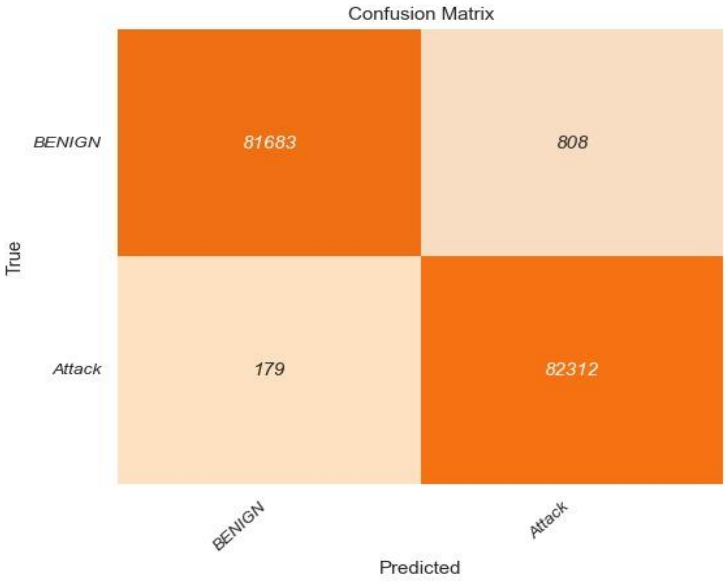


Figure 3.17: Confusion matrix for CatBoost Test_5.

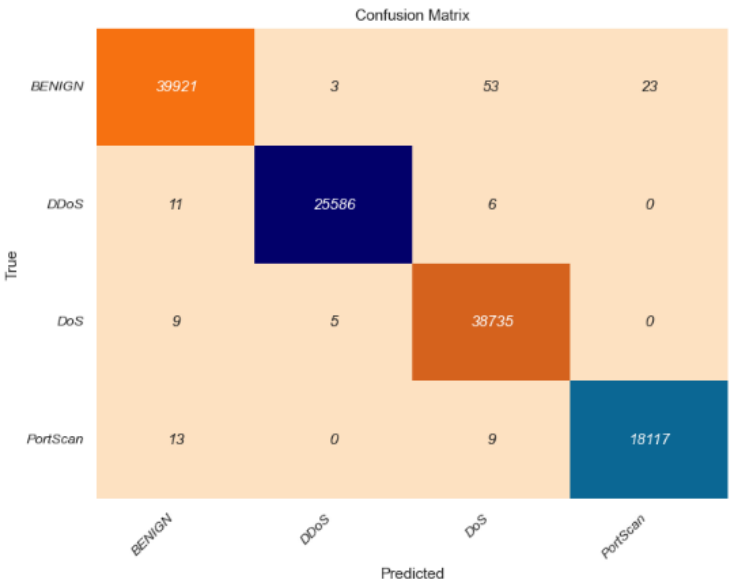


Figure 3.18: Confusion matrix for CatBoost Test_6.

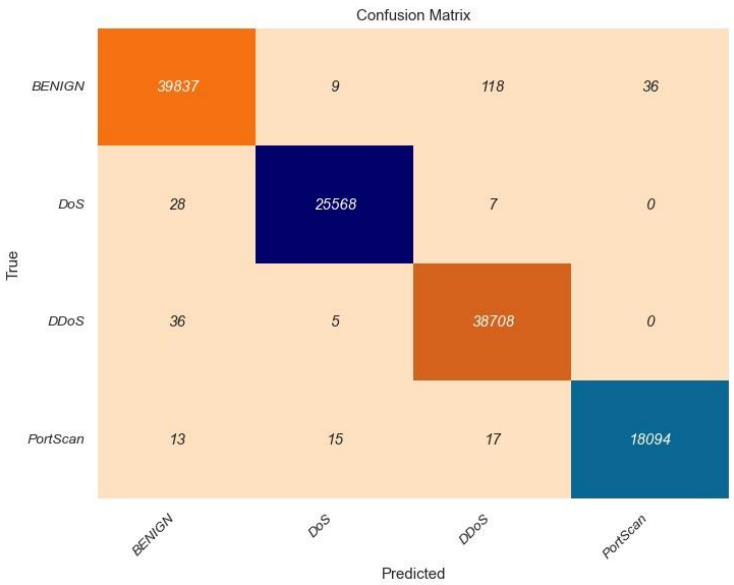


Figure 3.19: Confusion matrix for CatBoost Test_10.

Binary classification

This This comparative analysis explores a comprehensive evaluation study of the effectiveness of CatBoost, XGBoost, and DT models for binary classification on the test and prediction sets. Table 3.16 shows that CatBoost achieved high accuracy in the test and prediction sets, indicating its ability to reduce false positives and correctly identify the attack vector, as depicted by the confusion matrixes presented in Figures 3.16 and 3.17. As for XGBoost, it also gave us a good result, especially in terms of accuracy shown in Table 3.8, where there was more variability than CatBoost, especially on the prediction set. As for XGBoost's confusion matrix, it revealed few false negatives compared to CatBoost, especially in determining the attack vector represented in Figures 3.8 and 3.16. We achieved strong performance in the DT on the test set illustrated in Table 3.12, as it showed competitive performance on the test set and somewhat divergent results with the prediction set. This was confirmed by its confusion matrices, shown in Figures 3.12 and 3.13. These matrices showed a higher number of false negatives, especially in detecting attacks, indicating that the model is less reliable in more complex scenarios.

Overall, while all models perform well in binary classification, CatBoost consistently achieves the best balance between precision and recall, followed by XGBoost and DT.

Multi-Class Classification:

For multi-class classification, comparing the three models (CatBoost, XGBoost, and DT) revealed consistently high performance, with all models achieving excellent accuracy (more than 99%) on the test set. Classification performance was strong across traffic types on the test and prediction sets with some minor differences, with the DT model on the test set achieving the highest accuracy at 99.94%, slightly outperforming the CatBoost model accuracy at 99.89%, and the XGBoost model accuracy at 99.83% (as shown in Tables 3.9, 3.13, and 3.17). On the prediction set, CatBoost and DT outperformed XGBoost, in which CatBoost and DT achieved 96% and 97% accuracy, respectively. In return, XGBoost achieved an accuracy of 95%. Based on the confusion matrices in Figures 3.10 and 3.11 for XGBoost, Figures 3.14 and 3.15 for DT, and Figures 3.18 and 3.19 for CatBoost, we can see high positive rates with all models showing excellent accuracy, with the vast majority of predictions falling on the diagonal (correct classifications). The "BENIGN" class predictions were consistently accurate across all models. Additionally, the models achieved high accuracy in detecting attacks ("DDoS," "DoS," and "PortScan"), with minimal false predictions observed.

3.4.3 DL Classification Models

IDS uses DL models because of their exceptional ability to process vast amounts of data and automatically extract important features. These models excel at identifying complex patterns and adapting to changing threats, resulting in fewer false alarms and enhanced real-time detection capabilities. However, the long training time is a major challenge in using these models. To address this problem, optimization algorithms are used to improve the performance and accelerate the process, such as Momentum, RMSprop (Root Mean Square Propagation), and the Adam (Adaptive Moment Estimation) optimizer which combines Momentum and RMSprop algorithms, it become the preferred choice due to its high efficiency in most applications. As a result, we chose Adam in all our DL experiments. The key hyperparameters that distinguish Adam are the learning rate α (alpha) as well as the exponential decay rates β_1 and β_2 (beta1 and beta2). These parameters help to dynamically adapt the learning rate for each parameter in the model. α is usually set to a small value (such as 0.001), while β_1 and β_2 are set to values close to 1 (such as 0.9 and 0.999, respectively) for optimal performance. Despite our attempts to change these hyperparameters, we noticed a decline in performance. Therefore, we focused on making changes in the model structure, the number of epochs, and the batch size for each test, keeping Adam's hyperparameters the same.

When training DL models to handle different types of classification tasks, we used the sigmoid activation function for binary classification, as it outputs a probability score between 0 and 1 (like 0.25 and 0.75). Meanwhile, for multi-class classification, we employed the softmax activation function, which outputs a probability distribution across multiple classes (like 0.25, 0.25, 0.3, and 0.2), allowing the model to assign probabilities to each class.

3.4.3.1 Convolutional Neural Network

Convolutional Neural Networks (CNN) are DL models designed to emulate the human visual processing system. The architecture of a CNN consists of alternating layers of convolution and pooling. Where the convolutional layers are responsible for feature extraction, while pooling layers enhance the generalizability of these features. As for the flattening layer and the fully connected layers, the first transforms the data into a one-dimensional (1D) matrix, and the second connects all neurons for final learning and classification, as shown in Figure 3.20, CNNs operate on two-dimensional (2D) data, necessitating the transformation of input data into matrices [49].

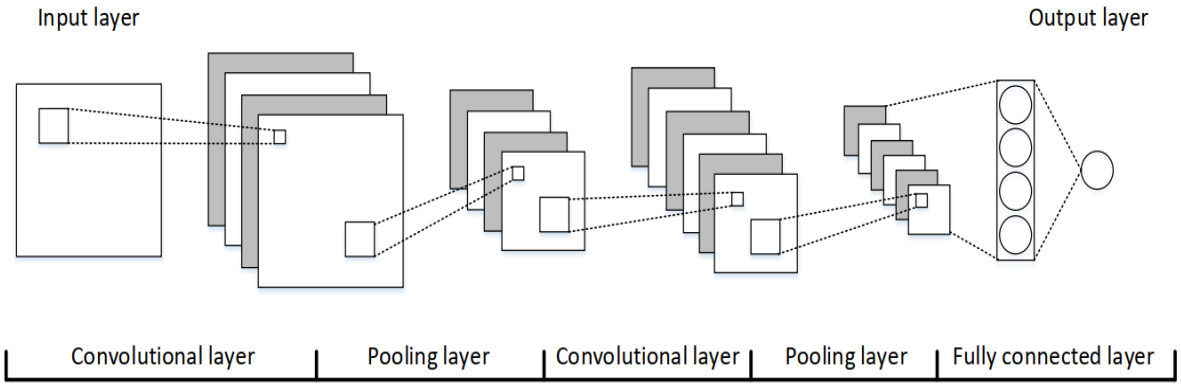


Figure 3.20: The structure of a CNN [49].

In this study, we implemented top five distinct CNN models to address the specified classification task: cnn_1, cnn_2, cnn_3, cnn_4, and LeNet5. The detailed structures of each model are presented in Tables 3.18 to 3.22, which show the type of each layer and the shape of its output. The first four models exhibit structural diversity, while the fifth model represents an implementation of the classic LeNet [50] architecture. This variety of structures allows for a comprehensive comparison and highlights the impact of different structural components on the accuracy of the classification and the efficiency of the model.

We used the same basic structures for both multi-class and binary classification, with only one modification in the last layer. In the case of binary classification, the last layer was changed to contain (None, 1).

Tables 3.23 and 3.24 show the details of the different tests, with each table displaying the structure used in each test, along with the specific values for the number of epochs and batch size used in the training process.

Table 3.18: Structure cnn_1.

Layer	Output Shape
Dense	(None, 81)
Reshape	(None, 9, 9, 1)
Conv2D	(None, 9, 9, 64)
MaxPooling2D	(None, 4, 4, 64)
Conv2D	(None, 4, 4, 128)
MaxPooling2D	(None, 2, 2, 128)
Conv2D	(None, 2, 2, 256)
MaxPooling2D	(None, 1, 1, 256)
Flatten	(None, 256)
Dropout (0.5)	(None, 256)
Dense	(None, 256)
Dense	(None, 4)

Table 3.19: Structure cnn_2.

Layer	Output Shape
Dense	(None, 81)
Reshape	(None, 9, 9, 1)
Conv2D	(None, 9, 9, 128)
BatchNormalization	(None, 9, 9, 128)
MaxPooling2D	(None, 4, 4, 128)
Conv2D	(None, 4, 4, 128)
BatchNormalization	(None, 4, 4, 128)
MaxPooling2D	(None, 2, 2, 128)
Conv2D	(None, 2, 2, 32)
MaxPooling2D	(None, 1, 1, 32)
Flatten	(None, 32)
Dropout (0.2)	(None, 32)
Dense	(None, 128)
Dense	(None, 4)

Table 3.20: Structure cnn_3

Layer	Output Shape
Dense	(None, 81)
Reshape	(None, 9, 9, 1)
Conv2D	(None, 9, 9, 128)
BatchNormalization	(None, 9, 9, 128)
MaxPooling2D	(None, 4, 4, 128)
Conv2D	(None, 2, 2, 128)
BatchNormalization	(None, 2, 2, 128)
MaxPooling2D	(None, 1, 1, 128)
GlobalAverage Pooling2D	(None, 128)
Dropout (0.2)	(None, 128)
Dense	(None, 64)
Dense	(None, 4)

Table 3.21: Structure cnn_4.

Layer	Output Shape
Dense	(None, 81)
Reshape	(None, 9, 9, 1)
Conv2D	(None, 7, 7, 128)
BatchNormalization	(None, 7, 7, 128)
MaxPooling2D	(None, 3, 3, 128)
Flatten	(None, 1152)
Dense	(None, 32)
Dense	(None, 32)
Dense	(None, 4)

Table 3.22: Structure LeNet5.

Layer	Output Shape
Dense	(None, 1024)
Reshape	(None, 32, 32, 1)
Conv2D	(None, 32, 32, 6)
AveragePooling2D	(None, 16, 16, 6)
Conv2D	(None, 12, 12, 16)
AveragePooling2D	(None, 6, 6, 16)
Flatten	(None, 576)
Dense	(None, 120)
Dense	(None, 84)
Dense	(None, 4)

Table 3.23: Model settings used in CNN binary classification.

	Test_1	Test_2	Test_3	Test_4	Test_5
Models	cnn_1	cnn_2	cnn_3	cnn_4	LeNet5
epochs	60	60	60	60	30
batch_size	64	64	64	128	128

Table 3.24: Model settings used in CNN multi-class classification.

	Test_6	Test_7	Test_8	Test_9	Test_10
Models	cnn_1	cnn_2	cnn_3	cnn_4	LeNet5
epochs	60	30	30	30	30
batch_size	64	512	64	32	32

CNN testing results:

The CNN model demonstrates through Tables 3.25 and 3.26 robust performance on the test set while facing some generalization challenges on the prediction set. These challenges are particularly evident in Test_5 and Test_9. Nevertheless, other results range from acceptable to good, indicating satisfactory overall model performance. Confusion matrices for the test set shown in Figures 3.21 to 3.24 reveal relatively few misclassifications between the "BENIGN" and "Attack" categories in binary classification. For multi-class classification, tests showcase excellent performance in categorizing the four classes, with minimal misclassifications between "BENIGN" and "PortScan." Accuracy and loss curves indicate rapid learning and good results on training data (as shown in Figures 3.25 to 3.28). However, fluctuations in validation curves indicate variations in model performance and require further investigation.

Table 3.25: Results of CNN models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	98.8350	0.0284	0.9939	0.9827	0.9883	89.5905	0.9793	0.8089	0.8860
	Attack			0.9829	0.9940	0.9884		0.8372	0.9829	0.9042
Test_2	BENIGN	99.0472	0.0256	0.9959	0.9850	0.9904	91.2863	0.9626	0.8591	0.9079
	Attack			0.9852	0.9959	0.9905		0.8728	0.9666	0.9173
Test_3	BENIGN	99.1726	0.0255	0.9955	0.9879	0.9917	93.2140	0.9797	0.8826	0.9286
	Attack			0.9880	0.9956	0.9918		0.8932	0.9817	0.9353
Test_4	BENIGN	98.9520	0.0275	0.9944	0.9846	0.9895	89.2652	0.9762	0.8049	0.8823
	Attack			0.9847	0.9945	0.9896		0.8340	0.9804	0.9013
Test_5	BENIGN	99.2733	0.0219	0.9970	0.9884	0.9927	70.2782	0.6313	0.9748	0.7663
	Attack			0.9885	0.9970	0.9928		0.9447	0.4308	0.5917

Table 3.26: Results of CNN models in multi-class classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	99.1771	0.0315	0.9992	0.9764	0.9877	84.2541	0.7820	0.9558	0.8602
	DDoS			0.9993	0.9995	0.9994		/	/	/
	DoS			0.9936	0.9991	0.9964		0.9479	0.7273	0.8231
	PortScan			0.9623	0.9989	0.9803		/	/	/
Test_7	BENIGN	98.9746	0.0357	0.9995	0.9701	0.9846	94.2151	0.9949	0.8904	0.9397
	DDoS			0.9979	0.9989	0.9984		/	/	/
	DoS			0.9884	0.9994	0.9939		0.9077	0.9949	0.9493
	PortScan			0.9622	0.9989	0.9802		/	/	/
Test_8	BENIGN	98.9493	0.0380	0.9990	0.9697	0.9841	90.6227	0.9060	0.9089	0.9074
	DDoS			0.9968	0.9990	0.9979		/	/	/
	DoS			0.9884	0.9994	0.9939		0.9127	0.9035	0.9081
	PortScan			0.9622	0.9987	0.9801		/	/	/
Test_9	BENIGN	99.1624	0.0333	0.9993	0.9759	0.9875	79.5608	0.9743	0.9309	0.9521
	DDoS			0.9993	0.9993	0.9993		/	/	/
	DoS			0.9930	0.9993	0.9961		0.9130	0.6579	0.7648
	PortScan			0.9624	0.9989	0.9803		/	/	/
Test_10	BENIGN	99.1795	0.0304	0.9990	0.9768	0.9877	93.4167	0.9491	0.9212	0.9349
	DDoS			0.9971	0.9993	0.9982		/	/	/
	DoS			0.9948	0.9990	0.9969		0.9373	0.9473	0.9423
	PortScan			0.9635	0.9989	0.9809		/	/	/

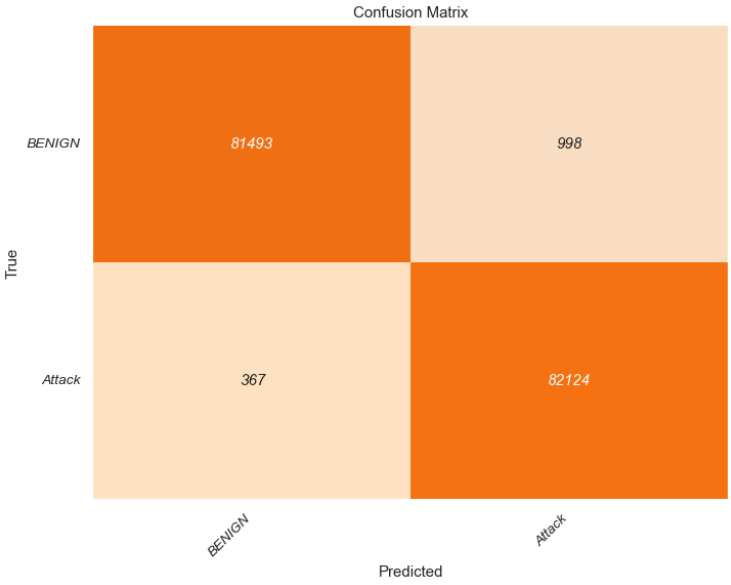


Figure 3.21: Confusion matrix for CNN Test_3.

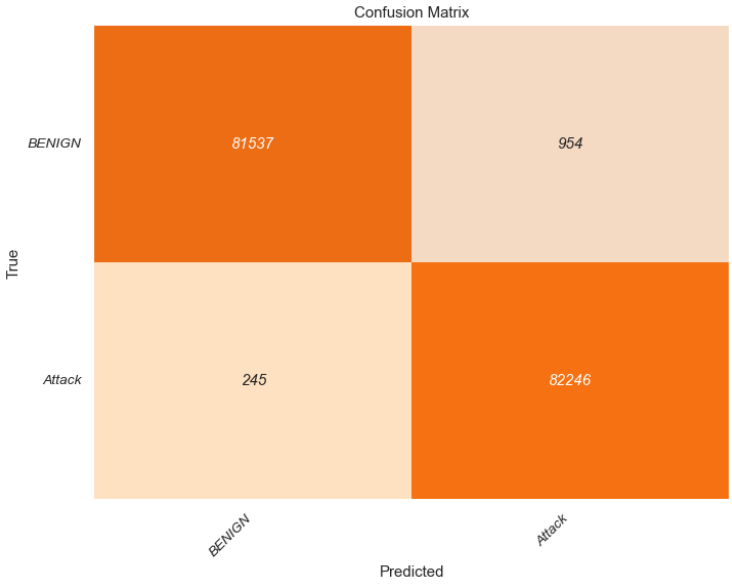


Figure 3.22: Confusion matrix for CNN Test_5.

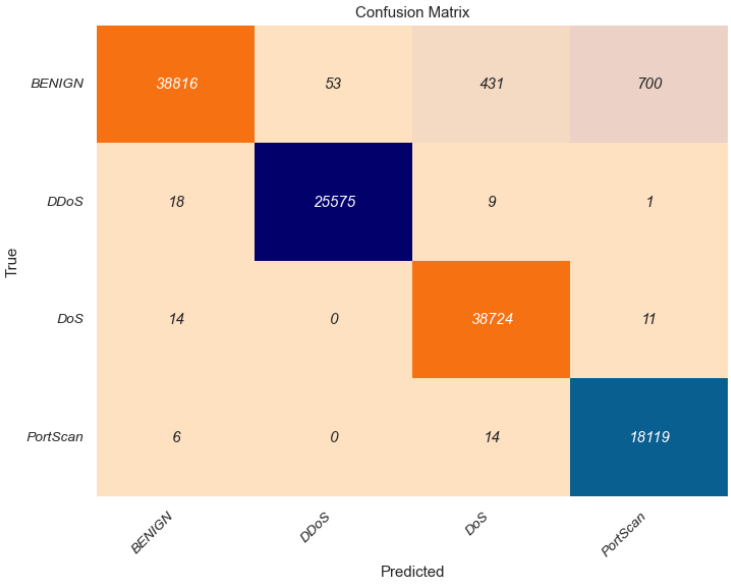


Figure 3.23: Confusion matrix for CNN Test_7.

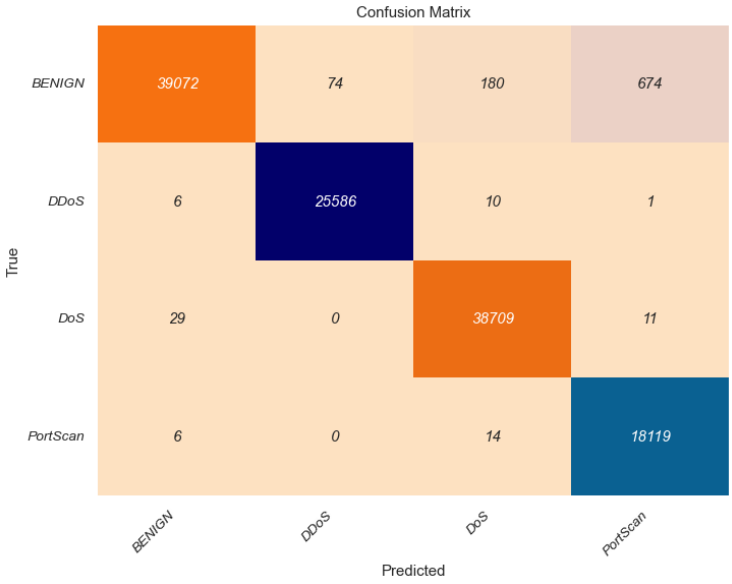


Figure 3.24: Confusion matrix for CNN Test_10.

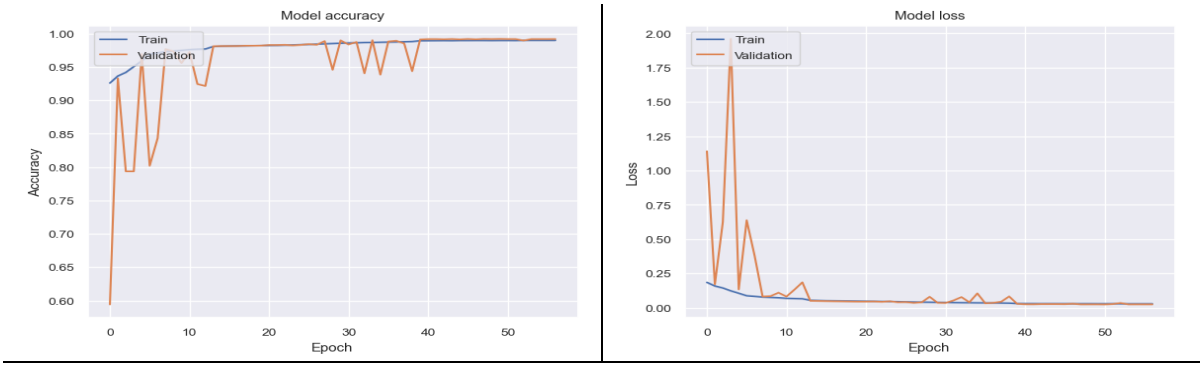


Figure 3.25: CNN Test_3 performance.

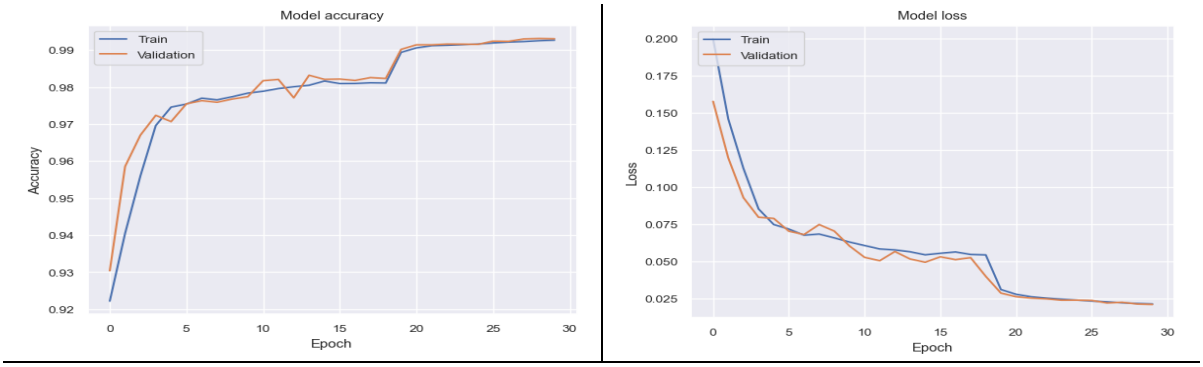


Figure 3.26: CNN Test_5 performance.

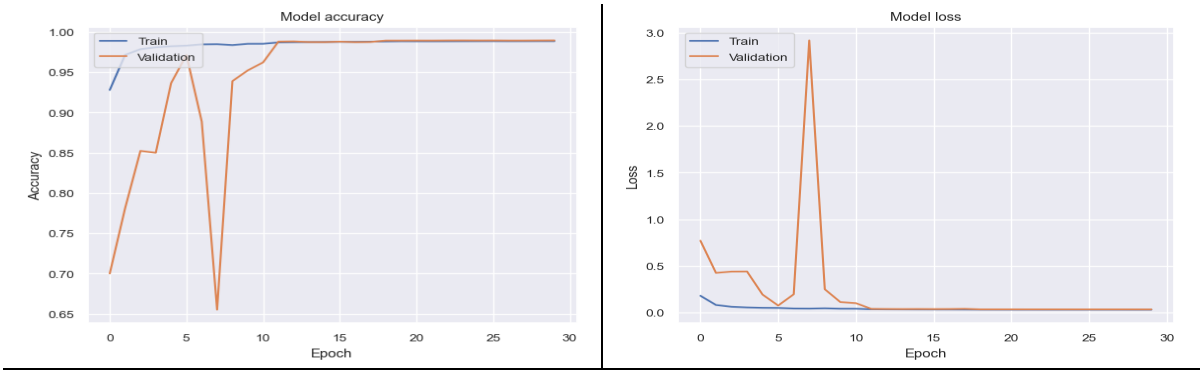


Figure 3.27: CNN Test_7 performance.

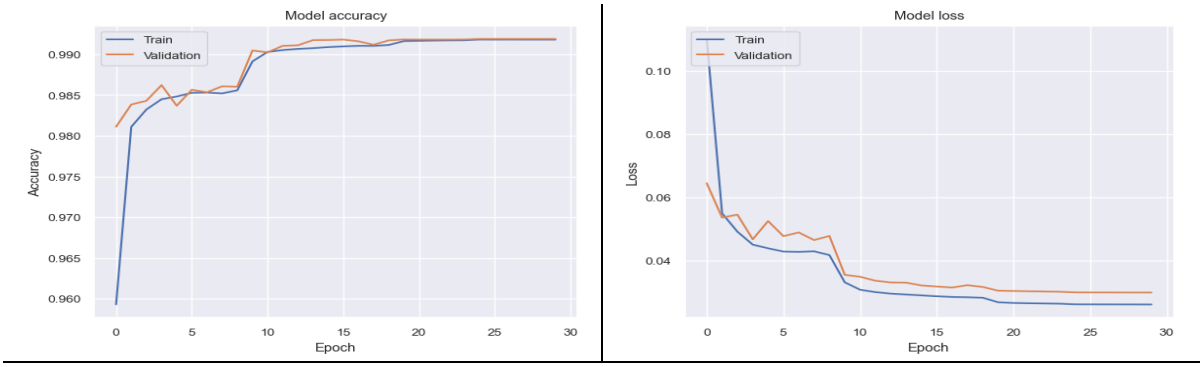


Figure 3.28: CNN Test_10 performance.

3.4.3.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) is an advanced type of Recurrent Neural Network (RNN) that excels at classification, analysis, and prediction of time-series data, particularly those with long-term dependencies. The core of LSTM is its unique cell structure, which utilizes three specialized gates to selectively process information (as shown in Figure 3.30). The forget gate removes unnecessary data from the network's memory, the input gate receives new data, and the output gate determines the current memory by combining short-term memory and long-term memory [34].

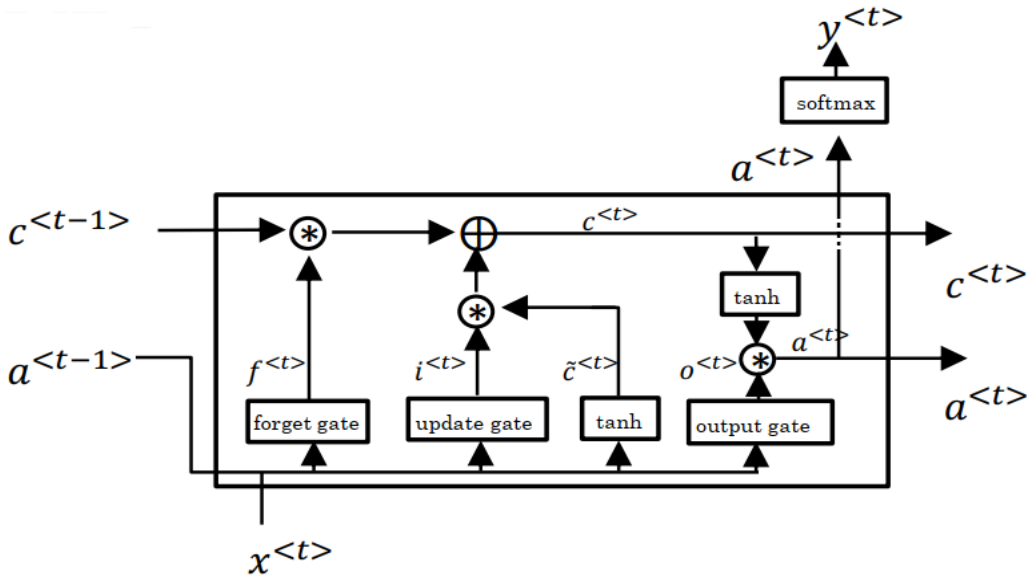


Figure 3.29: The structure of a LSTM [51].

For the LSTM tests, we employed top nine distinct structures in both classification tasks, as detailed in Tables 3.27 to 3.35. These structures incorporated different configurations LSTM layers, with some models utilizing Bidirectional layers and Dropout mechanisms to fine-tune generalization. Each experiment was associated with a specific structure, with the number of epochs and batch size for each test detailed in the Tables 3.36 and 3.37.

Table 3.27: Structure Lstm_1.

Layer	Output Shape
LSTM	(None, 1, 50)
LSTM	(None, 50)
Dense	(None, 1)

Table 3.28: Structure Lstm_2.

Layer	Output Shape
LSTM	(None, 1, 100)
LSTM	(None, 1, 50)
LSTM	(None, 50)
Dense	(None, 1)

Table 3.29: Structure Lstm_3.

Layer	Output Shape
Bidirectional	(None, 1, 100)
Bidirectional	(None, 100)
Dense	(None, 1)

Table 3.30: Structure Lstm_4.

Layer	Output Shape
Bidirectional	(None, 1, 200)
Bidirectional	(None, 1, 150)
Bidirectional	(None, 1, 100)
Bidirectional	(None, 1, 50)
Bidirectional	(None, 100)
Dense	(None, 1)

Table 3.31: Structure Lstm_5.

Layer	Output Shape
LSTM	(None, 50)
Dense	(None, 4)

Table 3.32: Structure Lstm_6.

Layer	Output Shape
LSTM	(None, 1, 100)
LSTM	(None, 100)
Dense	(None, 4)

Table 3.33: Structure Lstm_7.

Layer	Output Shape
LSTM	(None, 1, 150)
LSTM	(None, 150)
Dense	(None, 4)

Table 3.34: Structure Lstm_8.

Layer	Output Shape
Bidirectional	(None, 1, 200)
Dropout (0.5)	(None, 1, 200)
Bidirectional	(None, 200)
Dropout (0.5)	(None, 200)
Dense	(None, 4)

Table 3.35: Structure Lstm_9.

Layer	Output Shape
Bidirectional	(None, 1, 100)
Bidirectional	(None, 100)
Dense	(None, 4)

Table 3.36: Model settings used in LSTM binary classification.

	Test_1	**Test_2**	**Test_3**	**Test_4**	**Test_5**
Models	Lstm_1	Lstm_1	Lstm_2	Lstm_3	Lstm_4
epochs	30	60	50	50	50
batch_size	32	64	32	32	32

Table 3.37: Model settings used in LSTM multi-class classification.

	Test_6	**Test_7**	**Test_8**	**Test_9**	**Test_10**
Models	Lstm_5	Lstm_6	Lstm_7	Lstm_8	Lstm_9
epochs	100	30	50	50	60
batch_size	32	32	128	32	128

LSTM testing results:

Tables 3.38 and 3.39 analyses show exceptional classification performance for the LSTM model, achieving high accuracy in both binary and multi-class tasks across test and prediction sets. Confusion matrices for the test set confirm the model's effectiveness, showcasing its ability to accurately distinguish between different classes with minimal misclassifications (as shown in Figures 3.30 to 3.33). Training curves provide additional insights into the model's learning process, indicating rapid and stable learning progression. The strong alignment between training and validation performance further reflects the model's robust generalization capabilities (as shown in Figures 3.34 to 3.37).

Table 3.38: Results of LSTM models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	99.0363	0.0267	0.9945	0.9862	0.9903	92.3091	0.9782	0.8655	0.9184
	Attack			0.9863	0.9945	0.9904		0.8794	0.9807	0.9273
Test_2	BENIGN	99.1120	0.0245	0.9944	0.9878	0.9911	92.9183	0.9786	0.8775	0.9253
	Attack			0.9879	0.9944	0.9911		0.8890	0.9808	0.9327
Test_3	BENIGN	99.3690	0.0192	0.9974	0.9900	0.9937	95.0439	0.9606	0.9394	0.9499
	Attack			0.9900	0.9974	0.9937		0.9407	0.9615	0.9510
Test_4	BENIGN	99.2854	0.0206	0.9962	0.9895	0.9928	94.3020	0.9796	0.9049	0.9408
	Attack			0.9896	0.9962	0.9929		0.9116	0.9812	0.9451
Test_5	BENIGN	99.4866	0.0155	0.9985	0.9912	0.9948	95.1023	0.9290	0.9767	0.9523
	Attack			0.9913	0.9985	0.9949		0.9755	0.9253	0.9497

Table 3.39: Results of LSTM models in multi-class classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	98.9403	0.0372	0.9988	0.9695	0.9840	94.0292	0.9686	0.9127	0.9398
	DDoS			0.9927	0.9989	0.9958		/	/	/
	DoS			0.9889	0.9988	0.9938		0.9420	0.9684	0.9550
	PortScan			0.9673	0.9988	0.9828		/	/	/
Test_7	BENIGN	99.3159	0.0299	0.9987	0.9810	0.9897	92.8403	0.9223	0.9418	0.9320
	DDoS			0.9995	0.9990	0.9992		/	/	/
	DoS			0.9953	0.9990	0.9972		0.9457	0.9148	0.9300
	PortScan			0.9688	0.9989	0.9836		/	/	/
Test_8	BENIGN	99.2922	0.0269	0.9989	0.9802	0.9895	93.3509	0.9764	0.8898	0.9311
	DDoS			0.9996	0.9988	0.9992		/	/	/
	DoS			0.9939	0.9991	0.9965		0.9014	0.9780	0.9381
	PortScan			0.9690	0.9990	0.9838		/	/	/
Test_9	BENIGN	99.0212	0.0362	0.9988	0.9722	0.9853	94.4503	0.9999	0.8904	0.9420
	DDoS			0.9970	0.9988	0.9979		/	/	/
	DoS			0.9906	0.9991	0.9948		0.9116	0.9995	0.9535
	PortScan			0.9624	0.9985	0.9801		/	/	/
Test_10	BENIGN	99.1142	0.0340	0.9973	0.9764	0.9867	94.9705	0.9970	0.9033	0.9479
	DDoS			0.9988	0.9978	0.9983		/	/	/
	DoS			0.9908	0.9983	0.9945		0.9140	0.9969	0.9536
	PortScan			0.9690	0.9989	0.9837		/	/	/

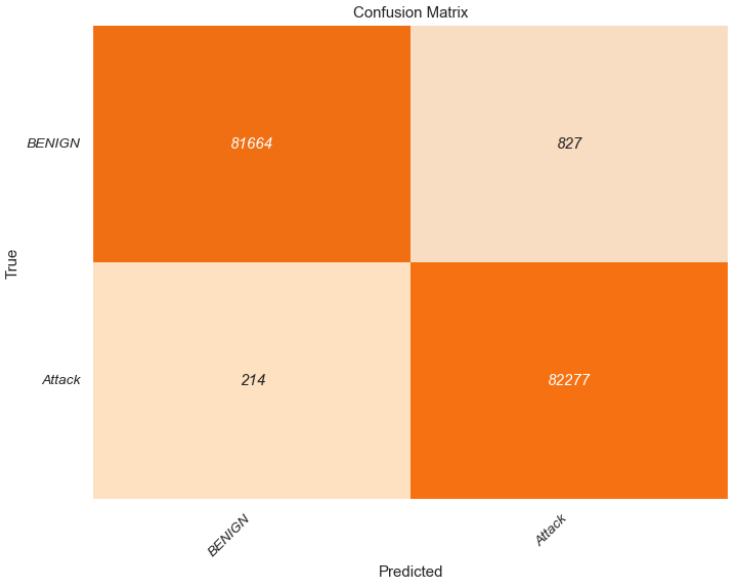


Figure 3.30: Confusion matrix for LSTM Test_3.

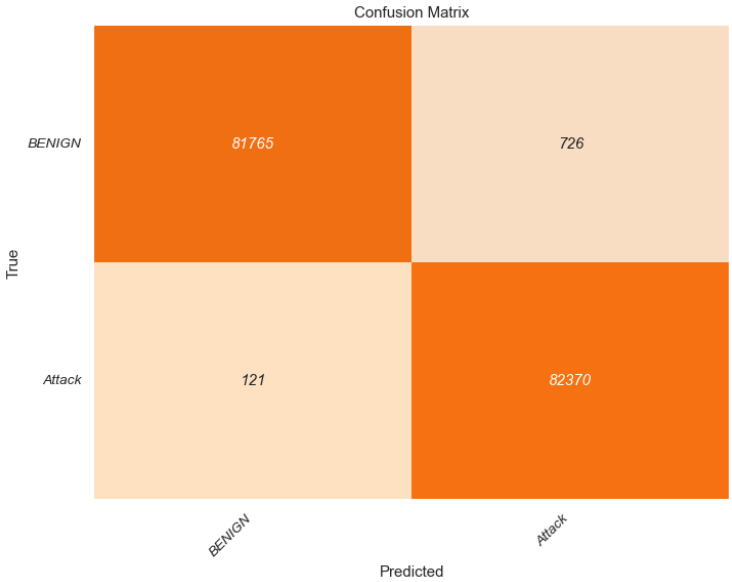


Figure 3.31: Confusion matrix for LSTM Test_5.

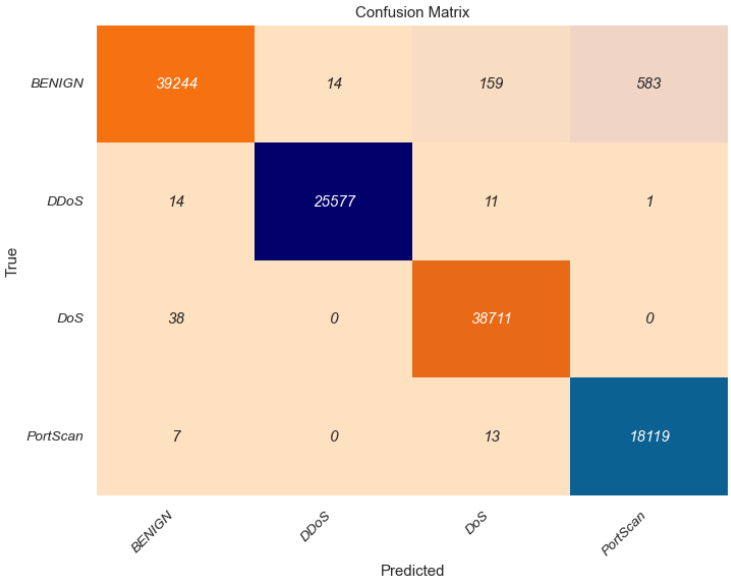


Figure 3.32: Confusion matrix for LSTM Test_7.

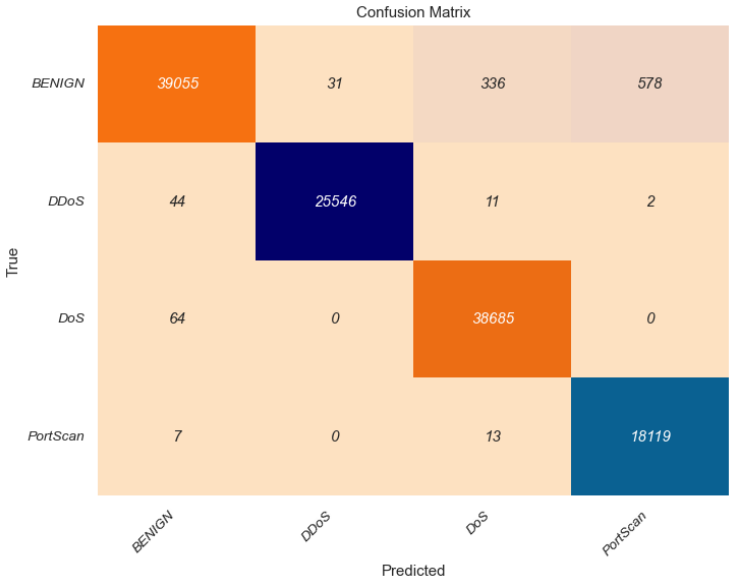


Figure 3.33: Confusion matrix for LSTM Test_10.

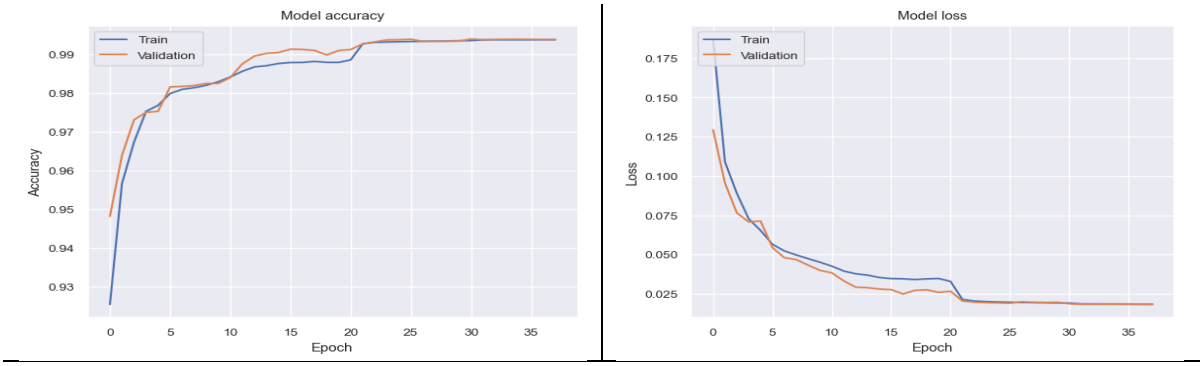


Figure 3.34: LSTM Test_3 performance.

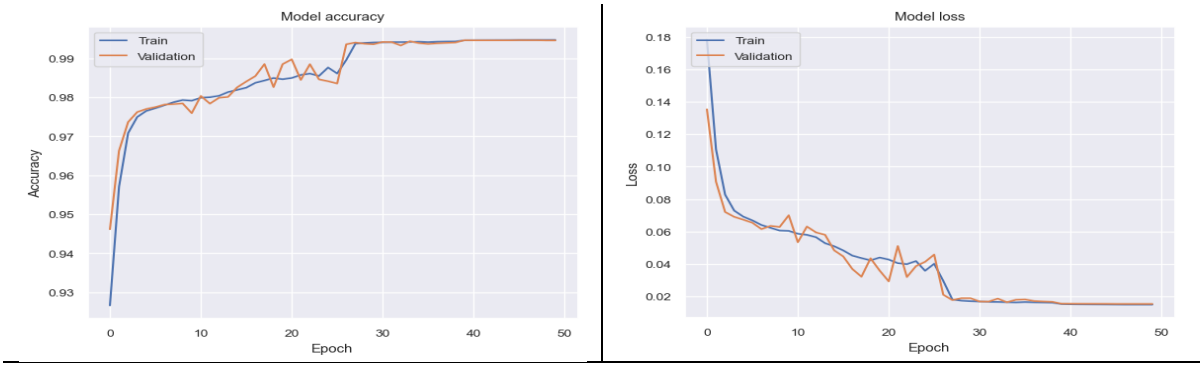


Figure 3.35: LSTM Test_5 performance.

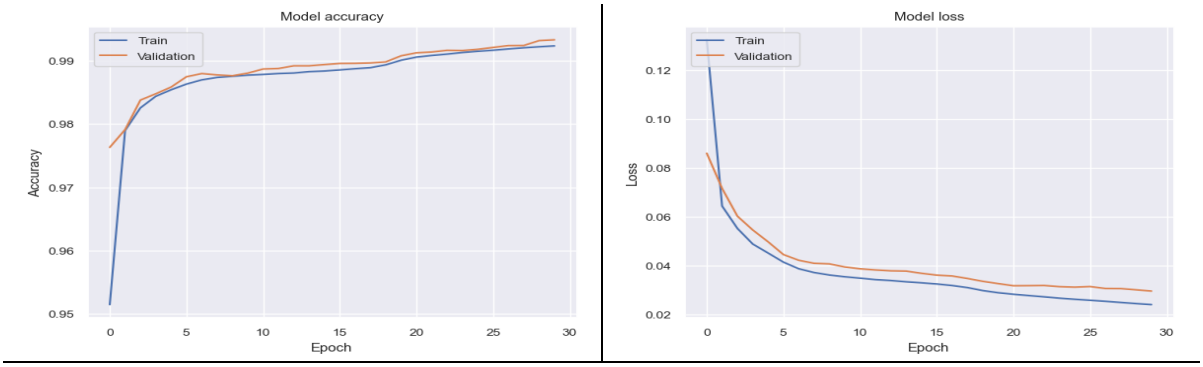


Figure 3.36: LSTM Test_7 performance.

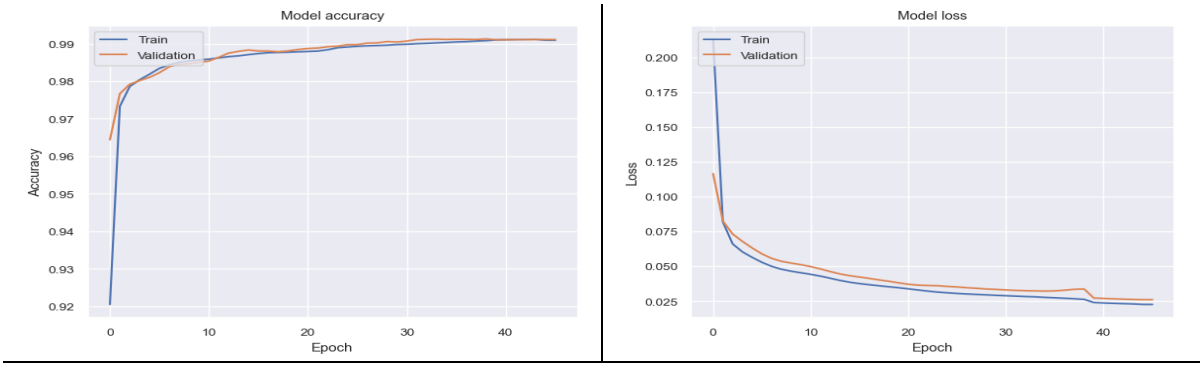


Figure 3.37: LSTM Test_10 performance.

3.4.3.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is an alternative to the LSTM architecture, proposed in 2014 as a way to simplify the complex LSTM structure while maintaining its ability to model sequential data effectively. The key simplification in GRUs is the merging of the forget gate and input gate into a single update gate. GRUs have two main gates: the reset gate, which determines how much of the previous hidden state should be forgotten or 'reset,' and the update gate, which controls how much of the new input should be used to update the hidden state (as shown in Figure 3.39). This streamlined gating mechanism allows GRUs to capture long-term dependencies in data while requiring fewer parameters than traditional LSTMs [52].

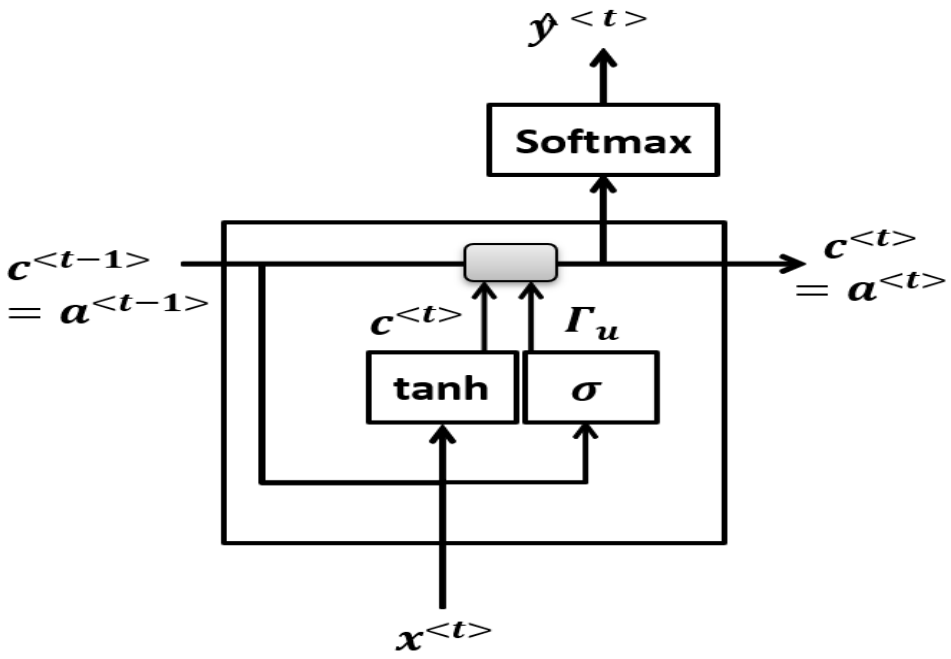


Figure 3.38: The structure of a GRU [51].

Given the results achieved using LSTM, we extended our tests to include GRU structures, which represent a recent advancement in RNN. At this stage, we implemented top 10 distinct GRU-based structures while maintaining consistency with our LSTM approach in using Bidirectional and Dropout layers (as noted in Tables 3.40 to 3.49). The structures were applied to various parameters, also outlined in Tables 3.50 and 3.51.

This GRU test not only complements our LSTM research but also offers valuable insights into the comparative strengths of these two prominent RNN types in the context of our classification tasks.

Table 3.40: Structure gru_1.

Layer	Output Shape
GRU	(None, 1, 100)
GRU	(None, 100)
Dense	(None, 1)

Table 3.41: Structure gru_2.

Layer	Output Shape
GRU	(None, 1, 200)
Dropout (0.5)	(None, 1, 200)
GRU	(None, 200)
Dropout (0.2)	(None, 200)
Dense	(None, 1)

Table 3.42: Structure gru_3.

Layer	Output Shape
GRU	(None, 1, 100)
GRU	(None, 1, 100)
GRU	(None, 100)
Dense	(None, 1)

Table 3.43: Structure gru_4.

Layer	Output Shape
Bidirectional	(None, 1, 100)
Bidirectional	(None, 100)
Dense	(None, 1)

Table 3.44: Structure gru_5.

Layer	Output Shape
Bidirectional	(None, 1, 200)
Bidirectional	(None, 1, 100)
Bidirectional	(None, 100)
Dense	(None, 1)

Table 3.45: Structure gru_6.

Layer	Output Shape
GRU	(None, 100)
Dense	(None, 4)

Table 3.46: Structure gru_7.

Layer	Output Shape
GRU	(None, 1, 100)
GRU	(None, 50)
Dense	(None, 4)

Table 3.47: Structure gru_8.

Layer	Output Shape
GRU	(None, 1, 25)
GRU	(None, 25)
Dense	(None, 4)

Table 3.48: Structure gru_9.

Layer	Output Shape
Bidirectional	(None, 1, 140)
Bidirectional	(None, 100)
Dense	(None, 4)

Table 3.49: Structure gru_10.

Layer	Output Shape
Bidirectional	(None, 1, 200)
Dropout	(None, 1, 200)
Bidirectional	(None, 200)
Dropout	(None, 200)
Dense	(None, 4)

Table 3.50: Model settings used in GRU binary classification.

	Test_1	Test_2	Test_3	Test_4	Test_5
Models	gru_1	gru_2	gru_3	gru_4	gru_5
epochs	60	60	60	70	60
batch_size	32	32	128	128	512

Table 3.51: Model settings used in GRU multi-class classification.

	Test_6	Test_7	Test_8	Test_9	Test_10
Models	gru_6	gru_7	gru_8	gru_9	gru_10
epochs	100	30	30	30	30
batch_size	32	32	32	64	64

GRU testing results:

The GRU model achieves satisfactory results in classification tasks on test and prediction sets (as shown in Tables 3.52 and 3.53), demonstrating efficient and remarkably fast learning capabilities. Confusion matrices for the test set reveal high accuracy in classification, as indicated by Figures 3.39 to 3.42, confirming the model's effectiveness in distinguishing between different categories. Learning curves exhibit significant stability across several experiments described in Figures 3.43 to 3.46, indicating good generalization capability and a low likelihood of overfitting. This stability enhances confidence in the model's performance and its applicability to diverse network traffic classification scenarios.

Table 3.52: Results of GRU models in binary classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_1	BENIGN	99.3211	0.0210	0.9973	0.9891	0.9932	93.5266	0.9568	0.9117	0.9337
	Attack			0.9892	0.9973	0.9932		0.9156	0.9589	0.9368
Test_2	BENIGN	99.2836	0.0218	0.9973	0.9884	0.9928	87.7496	0.8296	0.9501	0.8858
	Attack			0.9885	0.9973	0.9929		0.9417	0.8048	0.8679
Test_3	BENIGN	99.3811	0.0184	0.9970	0.9906	0.9938	93.4601	0.9657	0.9012	0.9323
	Attack			0.9907	0.9970	0.9938		0.9074	0.9680	0.9367
Test_4	BENIGN	99.3430	0.0200	0.9960	0.9908	0.9934	92.6326	0.9626	0.8871	0.9233
	Attack			0.9909	0.9960	0.9934		0.8953	0.9656	0.9291
Test_5	BENIGN	99.1066	0.0256	0.9941	0.9880	0.9910	91.7609	0.9707	0.8612	0.9127
	Attack			0.9880	0.9942	0.9911		0.8753	0.9740	0.9220

Table 3.53: Results of GRU models in multi-class classification.

Dataset		**CIC-IDS2017**					**CIC-IDS2018**			
Metrics		Accuracy	Loss	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Test_6	BENIGN	99.1950	0.0247	0.9979	0.9781	0.9879	91.0532	0.8945	0.9336	0.9137
	DDoS			0.9977	0.9988	0.9983		/	/	/
	DoS			0.9931	0.9984	0.9957		0.9490	0.8870	0.9170
	PortScan			0.9695	0.9988	0.9840		/	/	/
Test_7	BENIGN	99.1322	0.0318	0.9954	0.9787	0.9870	89.7152	0.8946	0.9033	0.8989
	DDoS			0.9993	0.9988	0.9990		/	/	/
	DoS			0.9953	0.9959	0.9956		0.9290	0.8909	0.9095
	PortScan			0.9637	0.9989	0.9810		/	/	/
Test_8	BENIGN	99.1191	0.0338	0.9987	0.9750	0.9867	95.4103	0.9912	0.9176	0.9530
	DDoS			0.9980	0.9990	0.9985		/	/	/
	DoS			0.9900	0.9991	0.9946		0.9375	0.9912	0.9636
	PortScan			0.9687	0.9989	0.9836		/	/	/
Test_9	BENIGN	99.2424	0.0322	0.9987	0.9786	0.9886	92.4772	0.9572	0.8951	0.9251
	DDoS			0.9996	0.9989	0.9992		/	/	/
	DoS			0.9929	0.9989	0.9959		0.9155	0.9550	0.9349
	PortScan			0.9686	0.9989	0.9835		/	/	/
Test_10	BENIGN	99.0775	0.0353	0.9989	0.9738	0.9862	94.3026	0.9970	0.8901	0.9406
	DDoS			0.9979	0.9984	0.9981		/	/	/
	DoS			0.9889	0.9996	0.9942		0.9052	0.9968	0.9488
	PortScan			0.9680	0.9983	0.9829		/	/	/

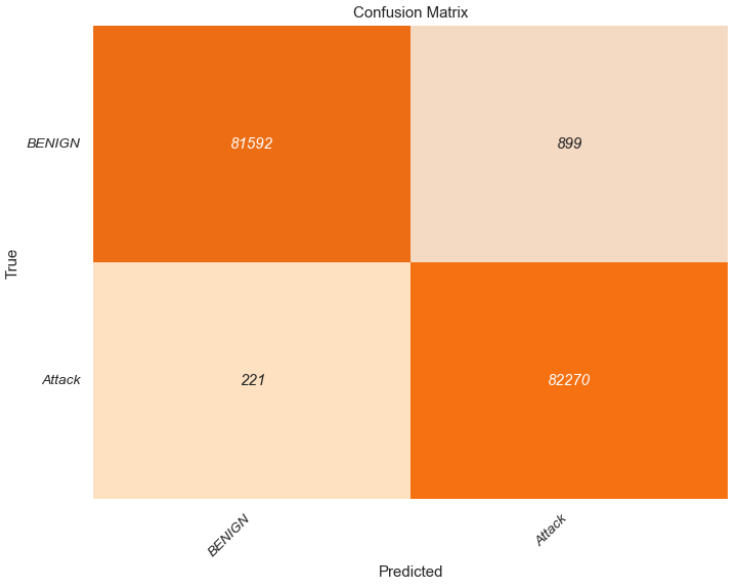


Figure 3.39: Confusion matrix for GRU Test_1.

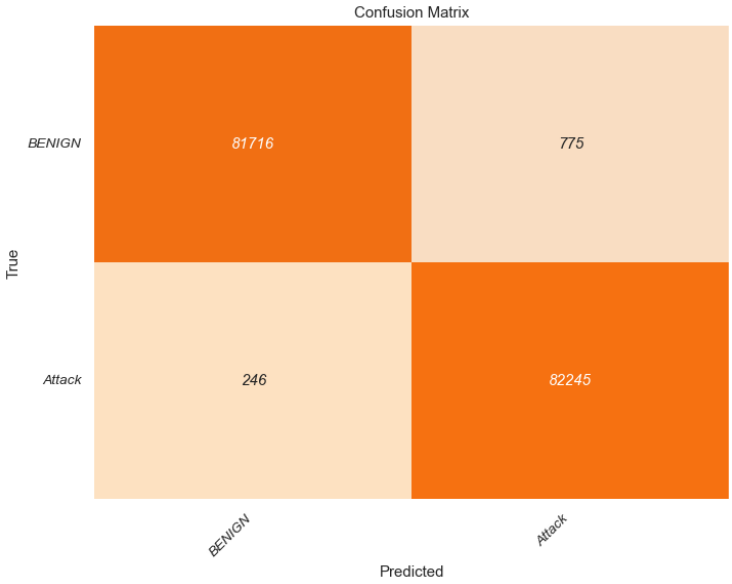


Figure 3.40: Confusion matrix for GRU Test_3.

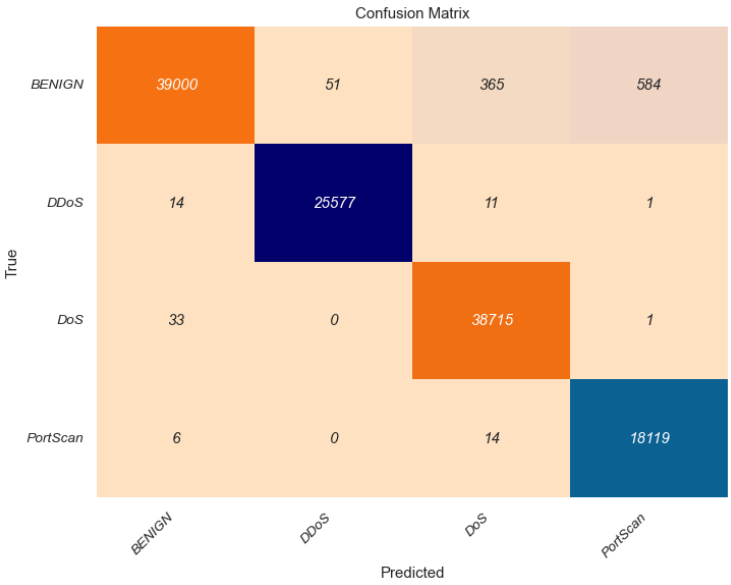


Figure 3.41: Confusion matrix for GRU Test_8.

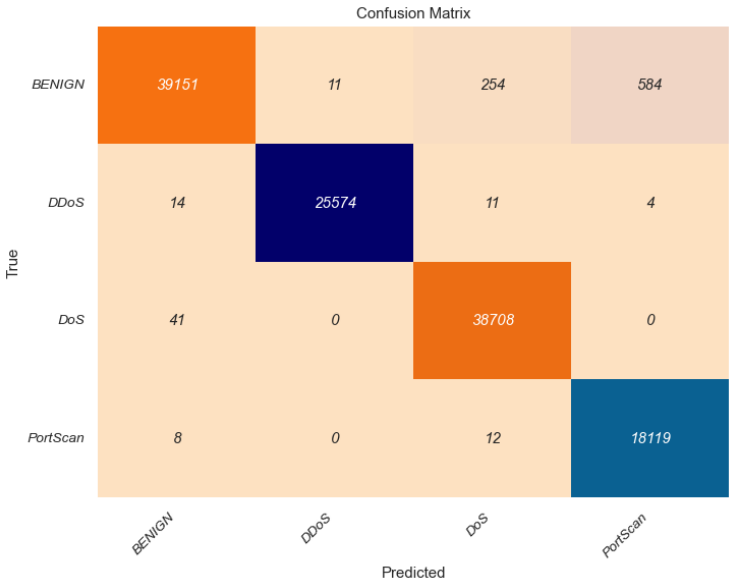


Figure 3.42: Confusion matrix for GRU Test_9.

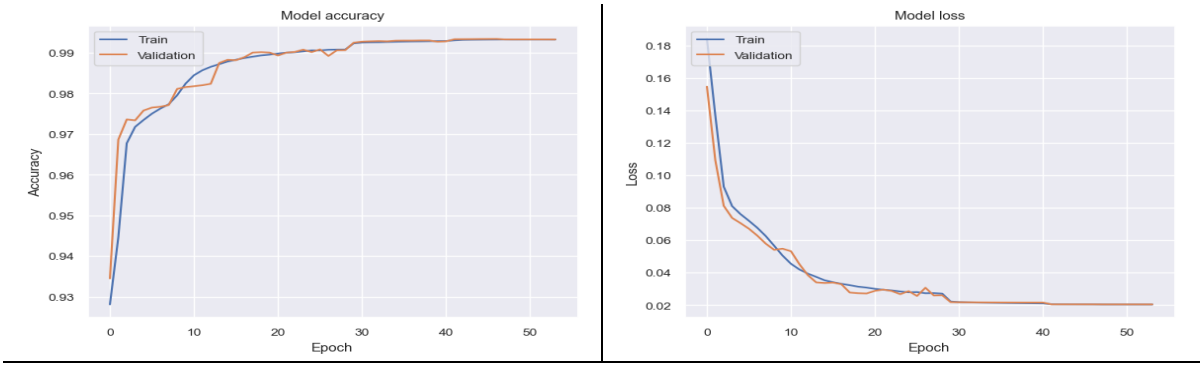


Figure 3.43: GRU Test_1 performance.

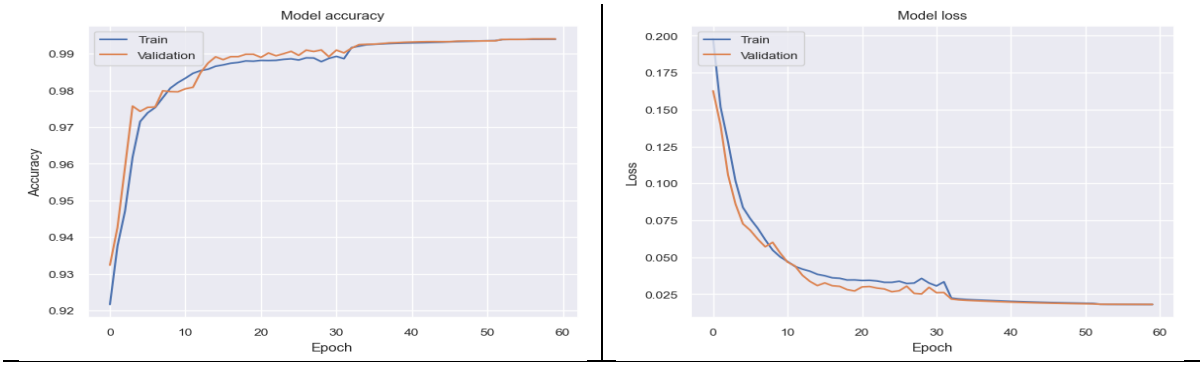


Figure 3.44: GRU Test_3 performance.

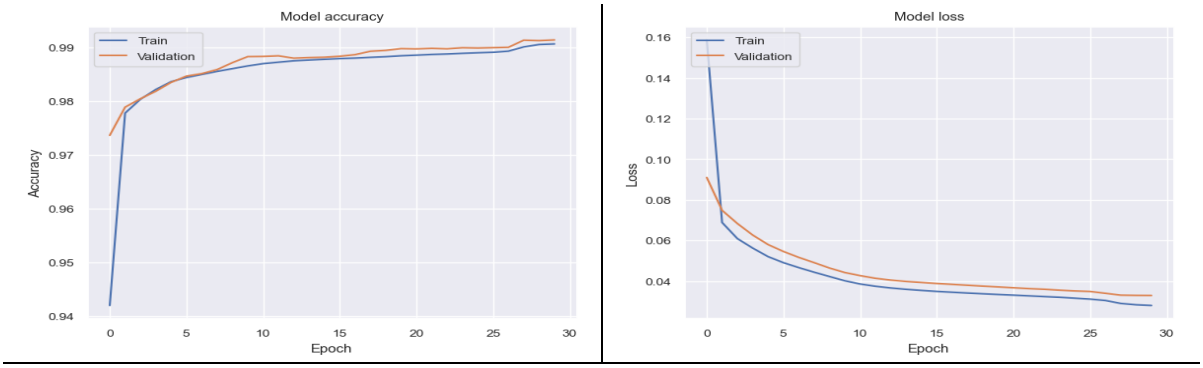


Figure 3.45: GRU Test_8 performance.

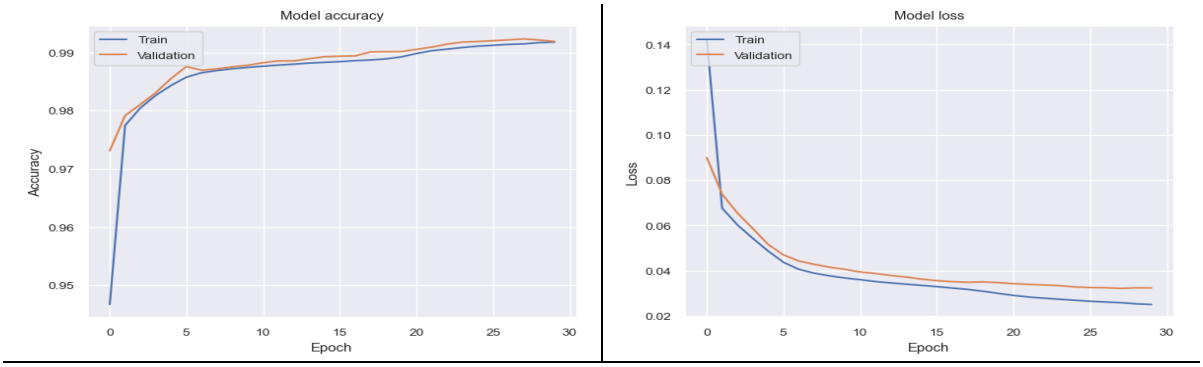


Figure 3.46: GRU Test_9 performance.

3.4.4 Comparison between ML and DL

Through comparative analysis Based on our study and the results obtained between ML and DL models, we can say that data preprocessing including cleaning, balancing, and feature selection allow any model to reach its full potential in training because it plays a crucial role in achieving an optimal and strong performance. This ultimately leads to higher accuracy and more credible predictions across different datasets and scenarios.

Furthermore, DL techniques may offer an additional advantage in handling complex and large-scale data. However, ML models can be more efficient and interpretable in certain

scenarios. It is also important to consider the training time for both types of models. Generally, ML models have shorter training times, making them more suitable for quick iterations and rapid prototyping, whereas DL models often require longer training periods due to their complex architectures and the need to optimize a large number of parameters.

Figures 3.47 and 3.48 present the best results obtained for the ML and DL models using accuracy metrics across binary and multi-class classification tasks. Overall, all models performed well on the test and prediction sets. This indicates that the models effectively handled both binary and multi-class classification problems for IDS.

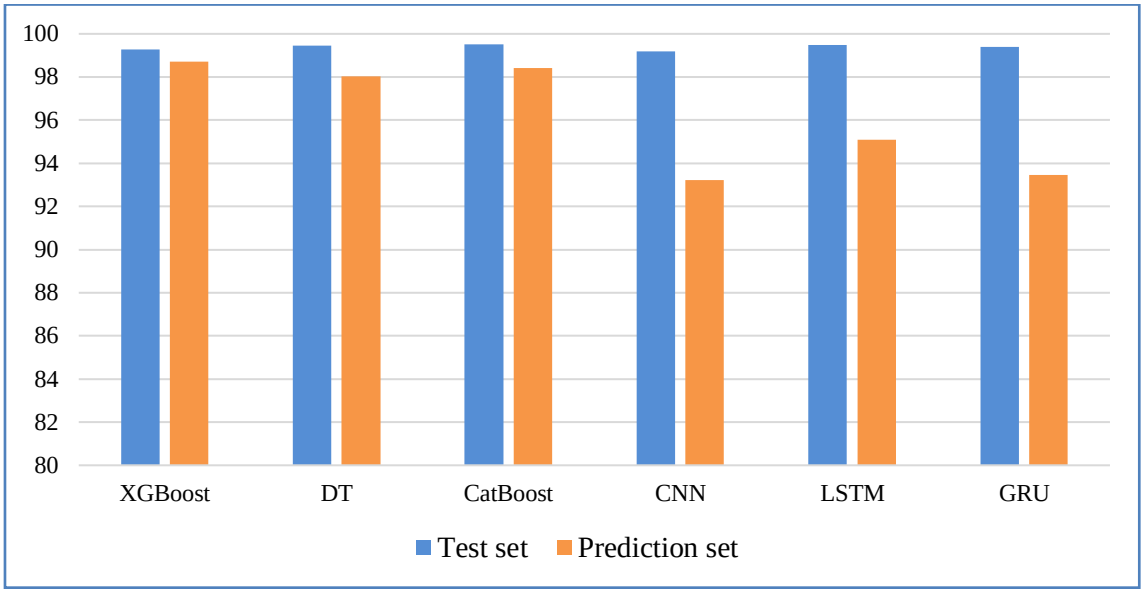


Figure 3.47: Best results in ML and DL models for binary classification.

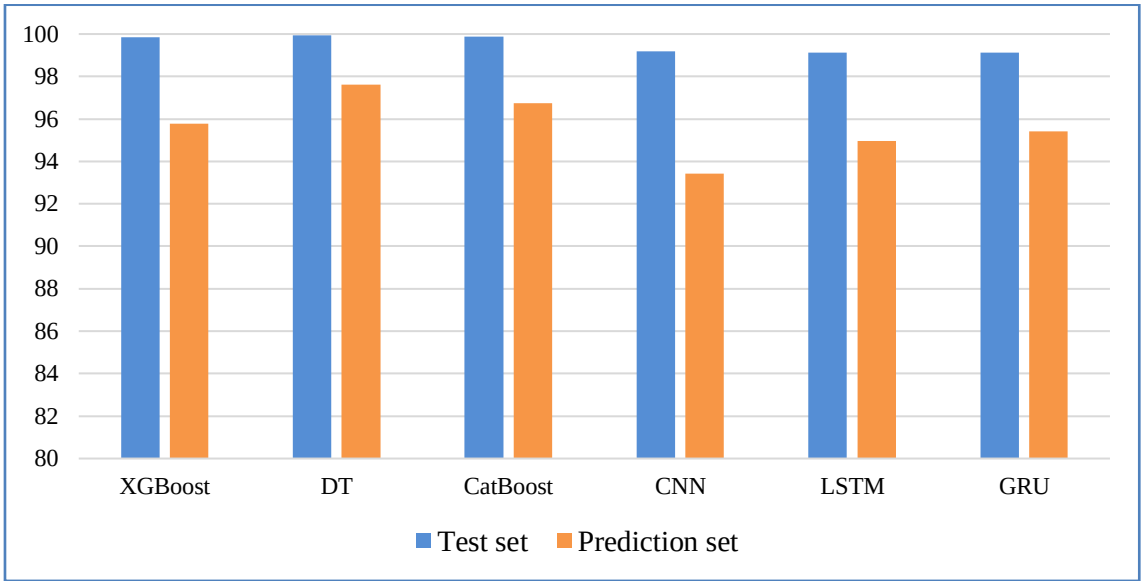


Figure 3.48: Best results in ML and DL models multi-class classification.

3.5 Conclusion

In conclusion, this chapter provided a comprehensive explanation of the practical implementation of ML and DL techniques in the field of IDS. By reviewing various models and evaluating their effectiveness, our study has proven their ability to achieve excellent results, reflecting the great potential of these techniques in enhancing the performance of IDS and providing more efficient solutions to confront cyber threats.

GENERAL CONCLUSION

At the conclusion of this study, we can draw several key insights and recommendations regarding the use of ML/DL models to enhance IDS. In this project, we conducted a systematic analysis of prominent IDS datasets and evaluated the performance of six different models. Through an in-depth examination of the training set and attentive evaluation of these models across the test set and the prediction set, our research highlights the significant potential of AI in enhancing the capabilities of detecting cyber threats.

The developed ML/DL models exhibited satisfactory performance across various evaluation metrics in both binary and multi-class classification tasks within the test set. In the context of binary classification, accuracy was selected as the primary metric due to the balanced nature of the dataset, where the distribution of attack and benign cases was equal at 50%. Under such balanced conditions, accuracy reliably reflects the model's performance without bias toward any specific class, since the models achieved high performance with an accuracy rate exceeding 99%. Besides, the confusion matrix analysis demonstrated the models' high efficiency in distinguishing between benign and attack cases, with only a few misclassifications recorded.

In multi-class classification, a broader range of metrics was used to evaluate model performance in order to ensure a fair assessment across all classes and account for variations in class sizes. While accuracy remained high and exceeded 99%, both the F1 score and recall also achieved satisfactory results, surpassing 99% for all classes. Additionally, the confusion matrix demonstrated the models' effective performance in distinguishing between different classes, with minimal misclassifications recorded.

Across both classification types, the models demonstrated an ability to distinguish between benign and attack types in the prediction set, achieving results greater than 90% across various metrics. This confirms the models' ability to generalize to unseen data.

Despite the promising results, it is essential to recognize that the field of cybersecurity is continuously evolving. We therefore recommend continued research and development, focusing on improving model adaptability to emerging and unknown threats. This includes employing anomaly detection techniques to identify abnormal behaviors that may indicate

new attacks. Furthermore, exploring semi-supervised techniques can help make better use of limited labeled data, enhancing detection accuracy in real-world scenarios. A promising avenue for future research could focus on the application of data augmentation techniques to artificially increase the size and diversity of IDS datasets. These techniques could help address issues such as data imbalance and improve the generalization capabilities of ML/DL models. Developing mechanisms to explain model decisions will also increase trust in their deployment in sensitive environments, while combining various ML/DL techniques could further improve performance.

Finally, we hope that this study contributes to advancing research and innovation in the field of IDS and enhances our ability to address future security challenges more efficiently and effectively.

BIBLIOGRAPHY

[1] K.K.Patel and B.V.Buddhadev, "An Architecture of Hybrid Intrusion Detection System," International Journal of Information & Network Security (IJINS), Vol.2, No.2, pp. 197~202 ,ISSN: 2089-3299, April 2013.

[2] H.Banguia, M.Geb, and B.Buhnovaa, "A hybrid machine learning model for intrusion detection in VANET," Computing 104(11)DOI: 10.1007/s00607-021-01001-0, March 2022.

[3] I.Naqvi, A.Chaudhary, and A.Kumar, "A Systematic Review of the Intrusion Detection Techniques in VANETS," 1Amity Institute of Information Technology, Amity University, Noida, U.P., India 2 DIT University, Dehradun, Uttarakhand, IndıaTEM Journal. Volume 11, Issue 2, pages 900-907, ISSN 2217-8309, DOI: 10.18421/TEM112-51, May 2022.

[4] "What is an Intrusion Detection System." https://www.barracuda.com/support/glossary/intrusion-detection-system (accessed August April 2024).

[5] "What is an Intrusion Detection System (IDS)." https://www.fortinet.com/resources/cyberglossary/intrusion-detection-system (accessed August April 2024).

[6] "Cyber Security Types and Threats Defined – Detailed Guide." https://collegevidya.com/blog/cyber-security-types-and-threats/ (accessed March 2024).

[7] A.Lazarevic, V.Kumar, and J.Srivastava, "Intrusion Detection :Asurvey," DOI: 10.1007/0-387-24230-9_2, In book: Managing Cyber Threats, January 2005.

[8] "Types of Intrusion Detection System." https://intellipaat.com/blog/intrusion-detection-system/ (accessed March 2024).

[9] B.Mouloud and K.Zakaria, "Détection d'intrusions à base des réseaux de neurones et algorithmes génétiques," End of studies memory, University of Abou Bakr Belkaid– Tlemcen Faculté des Sciences Département d'Informatique, 2017.

[10] B.Deore and S.Bhosale, "A decisive approach to intrusion detection system using machine learning model," WEENTECH Proceedings in Energy, DOI: ,10.32438/WPE.152021, International Conference on Recent Advances in Mechanical Engineering, Department of Mechanical Engineering, Delhi Technological University, New Delhi, India, March 2021.

[11] "Intrusion Detection." https://owasp.org/www-community/controls/Intrusion_Detection (accessed March 2024).

[12] "What limitations do intrusion detection and prevention systems have." https://www.linkedin.com/advice/3/what-limitations-do-intrusion-detection-prevention-8yl5e (accessed March 2024).

[13] "Understanding and fighting alert fatigue." https://www.atlassian.com/incident-management/on-call/alert-fatigue#The-risks-of-alert-fatigue (accessed April 2024).

[14] "The Hidden Risks of False Positives: How to Prevent Alert Fatigue in Your Organization." https://www.stamus-networks.com/blog/the-hidden-risks-of-false-positives-how-to-prevent-alert-fatigue-in-your-organization (accessed April 2024).

[15] "What is a Zero-day Attack- Definition and Explanation." https://me-en.kaspersky.com/resource-center/definitions/zero-day-exploit (accessed April 2024).

[16] "How can you effectively evaluate the effectiveness of your intrusion detection system." https://www.linkedin.com/advice/0/how-can-you-effectively-evaluate-effectiveness-qppec (accessed April 2024).

[17] F.Hok and P.Kortis, "Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/IPS) design for an IP networks," Conference: 2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA), DOI:10.1109/ICETA.2015.7558466, Nov 2015.

[18] "The Future of Intrusion Detection: Protecting What Matters." https://www.linkedin.com/pulse/future-intrusion-detection-protecting-what-matters-cogent-safety-iyzpf/ (accessed April 2024).

[19] M.Correia, "Big Data Analytics for Intrusion Detection," In book: Advances in Information Security, Privacy, and Ethics, DOI: 10.4018/978-1-5225-9611-0.ch014, January 2020.

[20] "What is user and entity behavior analytics (UEBA)." https://www.ibm.com/topics/ueba (accessed April 2024).

[21] M. Aljanabi, M.A.Ismail, and A.H.Ali, "Intrusion Detection Systems, Issues, Challenges, and Needs," International Journal of Computational Intelligence Systems In Press, Uncorrected Proof DOI: https://doi.org/10.2991/ijcis.d.210105.001; ISSN: 1875-6891; eISSN: 1875-6883, January 2021.

[22] "Types of Cyber Attacks." https://www.fortinet.com/resources/cyberglossary/types-of-cyber-attacks (accessed April 2024).

[23] J.P.Bharadiya, "AI-Driven Security: How Machine Learning Will Shape the Future of Cybersecurity and Web 3.0," American Journal of Neural Networks and Applications, 2023; 9(1): 1-7: 10.11648/j.ajnna.20230901.11ISSN: 2469-7400 (Print); ISSN: 2469-7419 (Online), June2023.

[24] M.A.Ferrag, L.Maglaras, S.Moschoyiannis, and H.Janicke, “Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study,” Journal of Information Security and Applications 50, December 2019.

[25] M.Tavallaee, E.Bagheri, W.Lu, and A.A.Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications, CISDA 2009.

[26] S.Choudhary, N.Kesswani, "Analysis of KDD-Cup’99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," Procedia Computer Science 167:1561-1573, DOI:10.1016/j.procs.2020.03.367, January 2020.

[27] A.Alshaibi, M.Al-Ani, A.Al-Azzawi, and A.Konev, "The Comparison of Cybersecurity Datasets," Data 7(2):22 DOI:10.3390/data7020022, January 2022.

[28] M.Ghurab, G.Gaphari, F.Alshami, and R.Alshamy, "A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System", April 2021.

[29] G.Meena and R.R.Choudhary, "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA," DOI:10.1109/COMPTELIX.2017.8004032, 2017 International Conference on Computer, Communications and Electronics (Comptelix), July 2017.

[30] N.Moustafa and J.Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," DOI:10.1109/MilCIS.2015.7348942, Military, Communications and Information Systems Conference (MilCIS), 2015At: Canberra, Australia, November 2015.

[31] R.Panigrahi and S.Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," International Journal of Engineering & Technology 7(3):479-482, January 2018.

[32] M.Lanvin, P.F.Gimenez, Y.Han, F.Majorczyk, L.Mé, et al, "Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes," 17th International Conference on Risks and Security of Internet and Systems (CRiSIS), Sousse, Tunisia. pp.18-33, ff10.1007/978-3-031-31108-6_2ff. ffhal-03775466f, Dec 2022.

[33] Ravikumar and Dharshini, "Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset," Thesis. Rochester Institute of Technology, 2021.

[34] A.A.Hagar and B.W.Gawali, "Deep Learning for Improving Attack Detection System Using CSE-CICIDS2018," NeuroQuantology 20(7):3064-3074, DOI:10.14704/nq.2022.20.7.NQ33385, August 2022.

[35] G.K.Baydogmus, Ö.Demir, and O.K.Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," IEEE Access PP(99):1-1, DOI:10.1109/ACCESS.2020.2973219, February 2020.

[36] S.Meftah, T.Rachidi, and N.Assem, "Network Based Intrusion Detection Using the UNSW-NB15 Dataset," School of Science and Engineering, Al Akhawayn University in Ifrane, Ifrane 53000, Morocco, September 2019.

[37] A.Šarčević, D.Pintar, M.Vranić, and A.Krajna, " Cybersecurity Knowledge Extraction Using XAI," Appl. Sci. 12(17), 8669, August 2022.

[38] "Oversampling—Handling Imbalanced Data." https://medium.com/@abdallahashraf90x/oversampling-for-better-machine-learning-with-imbalanced-data-68f9b5ac2696 (accessed August 2024).

[39] "Undersampling vs. Oversampling for Imbalanced Datasets." https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/ (accessed August 2024).

[40] K.Bakour and O.Ceviz, "Empirical Enhancement of Intrusion Detection Systems: A Comprehensive Approach with Genetic Algorithm-based Hyperparameter Tuning and Hybrid Feature Selection," Arabian Journal for Science and Engineering, DOI:10.1007/s13369-024-08949-z, April 2024.

[41] "One Hot Encoding vs. Label Encoding in Machine Learning." https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/ (accessed August 2024).

[42] "What is Normalization of Data in Database." https://www.simplilearn.com/automated-recruiting-in-companies-article (accessed August 2024).

[43] "Different Normalization methods." https://medium.com/@mkc940/different-normalization-methods-a1be71fe9f1 (accessed August 2024).

[44] "Feature Selection." https://www.heavy.ai/technical-glossary/feature-selection (accessed August 2024).

[45] A.A. Hagar and B.W. Gawali, " Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018," August 2022.

[46] "Five Methods for Data Splitting in Machine Learning." https://medium.com/@tubelwj/five-methods-for-data-splitting-in-machine-learning-27baa50908ed (accessed August 2024).

[47] V.Bytyqi and B.Rexha, "Machine Learning Boosted Trees Algorithms in Cybersecurity: A Comprehensive Review," March 2024.

[48] R. Abdulhammed, "Intrusion Detection: Embedded Software Machine Learning and Hardware Rules Based Co-Designs," p. 176, 2019.

[49] H.Liu and B.Lang, " Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," Applied Sciences 9(20):43969(20):4396, DOI:10.3390/app9204396, October 2019.

[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11), 2278-2324, November 1998.

[51] J.Chung, C.Gulcehre, K.Cho, and Y. Bengio, " Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," December 2014.

[52] "Gated Recurrent Unit Networks." https://www.geeksforgeeks.org/gated-recurrent-unit-networks/ (accessed August 2024).

[53] Python. What is Python? Executive Summary. url: https://www.python. org/doc/essays/blurb/.

[54] A.Madani, "Debugging Machine Learning Models with Python: Develop high-performance, low-bias, and explainable machine learning and deep learning models," Packt Publishing, ISBN 978-1-80020-858-2, 2023.

[55] A.M.Aleesa, B.B.Zaidan, A.A.Zaidan, and N.M.Sahar, " Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions," Springer-Verlag London Ltd., part of Springer Nature 2019, October 2019.

Table 1: List of features in the CIC-IDS2017 datasets.

NB	Feature Name	NB	Feature Name
1	Destination Port	41	Packet Length Mean
2	Flow Duration	42	Packet Length Std
3	Total Fwd Packets	43	Packet Length Variance
4	Total Backward Packets	44	FIN Flag Count
5	Total Length of Fwd Packets	45	SYN Flag Count
6	Total Length of Bwd Packets	46	RST Flag Count
7	Fwd Packet Length Max	47	PSH Flag Count
8	Fwd Packet Length Min	48	ACK Flag Count
9	Fwd Packet Length Mean	49	URG Flag Count
10	Fwd Packet Length Std	50	CWE Flag Count
11	Bwd Packet Length Max	51	ECE Flag Count
12	Bwd Packet Length Min	52	Down/Up Ratio
13	Bwd Packet Length Mean	53	Average Packet Size
14	Bwd Packet Length Std	54	Avg Fwd Segment Size
15	Flow Bytes/s	55	Avg Bwd Segment Size
16	Flow Packets/s	56	Fwd Header Length
17	Flow IAT Mean	57	Fwd Avg Bytes/Bulk
18	Flow IAT Std	58	Fwd Avg Packets/Bulk
19	Flow IAT Max	59	Fwd Avg Bulk Rate
20	Flow IAT Min	60	Bwd Avg Bytes/Bulk
21	Fwd IAT Total	61	Bwd Avg Packets/Bulk
22	Fwd IAT Mean	62	Bwd Avg Bulk Rate
23	Fwd IAT Std	63	Subflow Fwd Packets
24	Fwd IAT Max	64	Subflow Fwd Bytes
25	Fwd IAT Min	65	Subflow Bwd Packets
26	Bwd IAT Total	66	Subflow Bwd Bytes
27	Bwd IAT Mean	67	Init_Win_bytes_forward
28	Bwd IAT Std	68	Init_Win_bytes_backward
29	Bwd IAT Max	69	act_data_pkt_fwd
30	Bwd IAT Min	70	min_seg_size_forward
31	Fwd PSH Flags	71	Active Mean
32	Bwd PSH Flags	72	Active Std
33	Fwd URG Flags	73	Active Max
34	Bwd URG Flags	74	Active Min
35	Fwd Header Length	75	Idle Mean
36	Bwd Header Length	76	Idle Std
37	Fwd Packets/s	77	Idle Max
38	Bwd Packets/s	78	Idle Min
39	Min Packet Length	79	Label
40	Max Packet Length		

Table 2: List of features in the CIC-IDS2018 datasets.

NB	Feature Name	NB	Feature Name
1	Destination Port	43	Fwd Pkts/s
2	Protocol	44	Bwd Pkts/s
3	Timestamp	45	Pkt Len Min
4	Flow ID	46	Pkt Len Max
5	Src IP	47	Pkt Len Mean
6	Dst IP	48	Pkt Len Std
7	Src Port	49	Pkt Len Var
8	Flow Duration	50	FIN Flag Cnt
9	Tot Fwd Pkts	51	SYN Flag Cnt
10	Tot Bwd Pkts	52	RST Flag Cnt
11	TotLen Fwd Pkts	53	PSH Flag Cnt
12	TotLen Bwd Pkts	54	ACK Flag Cnt
13	Fwd Pkt Len Max	55	URG Flag Cnt
14	Fwd Pkt Len Min	56	CWE Flag Count
15	Fwd Pkt Len Mean	57	ECE Flag Cnt
16	Fwd Pkt Len Std	58	Down/Up Ratio
17	Bwd Pkt Len Max	59	Pkt Size Avg
18	Bwd Pkt Len Min	60	Fwd Seg Size Avg.1
19	Bwd Pkt Len Mean	61	Bwd Seg Size Avg
20	Bwd Pkt Len Std	62	Fwd Byts/b Avg
21	Flow Byts/s	63	Fwd Pkts/b Avg
22	Flow Pkts/s	64	Fwd Blk Rate Avg
23	Flow IAT Mean	65	Bwd Byts/b Avg
24	Flow IAT Std	66	Bwd Pkts/b Avg
25	Flow IAT Max	67	Bwd Blk Rate Avg
26	Flow IAT Min	68	Subflow Fwd Pkts
27	Fwd IAT Tot	69	Subflow Fwd Byts
28	Fwd IAT Mean	70	Subflow Bwd Pkts
29	Fwd IAT Std	71	Subflow Bwd Byts
30	Fwd IAT Max	72	Init Fwd Win Byts
31	Fwd IAT Min	73	Init Bwd Win Byts
32	Bwd IAT Tot	74	Fwd Act Data Pkts
33	Bwd IAT Mean	75	Fwd Seg Size Min
34	Bwd IAT Std	76	Active Mean
35	Bwd IAT Max	77	Active Std
36	Bwd IAT Min	78	Active Max
37	Fwd PSH Flags	79	Active Min
38	Bwd PSH Flags	80	Idle Mean
39	Fwd URG Flags	81	Idle Std
40	Bwd URG Flags	82	Idle Max
41	Fwd Header Len	83	Idle Min
42	Bwd Header Len	84	Label