Implementing Machine Learning to achieve dynamic Zero-Trust Intrusion Detection Systems (ZT-IDS) in 5G based IoT Networks

by Mohamed G. Nour

B.S. in Electrical Communications and Electronics Engineering, June 2003, University of Alexandria, Egypt
M.E. in Cybersecurity Policy & Compliance, May 2021, The George Washington University, USA

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

August 31, 2023

Praxis directed by

John M. Fossaceca
Professorial Lecturer of Engineering Management and Systems Engineering

Shahryar Sarkani
Adjunct Professor of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Mohamed G. Nour has passed the Final Examination for the degree of Doctor of Engineering as of July 29, 2023. This is the final and approved form of the Praxis.

Implementing Machine Learning to achieve dynamic Zero-Trust Intrusion Detection Systems (ZT-IDS) in 5G based IoT Networks

Mohamed G. Nour

Praxis Research Committee:

John M. Fossaceca, Professorial Lecturer in Engineering Management and Systems Engineering, Praxis Co-Director

Shahryar Sarkani, Adjunct Professor of Engineering Management and Systems Engineering, Praxis Co-Director

Muhammad Islam, Professorial Lecturer in Engineering Management and Systems Engineering, Committee Member

Dedication

I would like to dedicate this research to many people without whom this work might not have come to light. My father and my beloved late mother for their endless love, who have always urged me to pursue my graduate degrees. My great advisors, Dr. Fossaceca and Dr. Sarkani, for their support and guidance towards this endeavor. Special thanks to my beloved wife, Dr. Badran, who always encouraged and taught me that it's never too late to chase your passions. And finally, my two boys who put up with me while I pursued this.

Acknowledgements

The author wishes to acknowledge that I would only have pursued a doctorate with the opportunity provided by The NOKIA of America Corporation. I also want to acknowledge my great professors who lightened the way for me throughout my two graduate degrees at George Washington University for their support and guidance towards this achievement.

Abstract of Praxis

Implementing Machine Learning to achieve dynamic Zero-Trust Intrusion Detection Systems (ZT-IDS) in 5G based IoT Networks

The Internet of Things (IoT) pertains to devices that link to a network to gather and exchange data. With the increasing quantity of such devices, so do the associated security risks. With the growth of IoT and the increase in sophisticated intrusions, research is being conducted to enhance attack detection capabilities. However, IoT devices' limited energy capacity and scalability present significant challenges in addressing privacy and security concerns. Limited energy capacity can impact encryption by restricting the processing power available for implementing robust encryption algorithms and authentication mechanisms, which need processing power, increasing the risk of unauthorized access and compromising privacy.

Weaker security measures, such as shorter encryption keys or less frequent security updates, could make IoT devices more susceptible to attacks. Furthermore, it can make it challenging for IoT devices to receive regular firmware and software updates, including security patches. Outdated software leaves devices exposed to known vulnerabilities,

Limited energy capacity may also lead to less secure physical components, which can be more easily tampered with or compromised.

Furthermore, the widespread integration of IoT devices across diverse industries has led to an increase in cyberattacks targeting these devices and their networks. The lack of a zero-trust approach further exacerbates the security risks associated with IoT devices. Zero trust is a security design approach that requires strict authentication and

authorization measures for every access attempt, regardless of the source or location. Intruders can access IoT devices and networks without such measures, potentially compromising sensitive data and systems. Hence, by adopting a zero-trust strategy, the impact of IoT intrusions can be mitigated by reducing the attack surface and enhancing control over resource access.

IoT devices' storage and processing capabilities are often limited, making it difficult for them to support advanced Intrusion Detection Systems. Consequently, cyber attackers leverage these device vulnerabilities to gain access to other systems linked to the same network. To overcome this problem, many organizations opt for Edge layer data centers that process IoT user traffic closer to the end user, reducing latency. These smaller facilities are interconnected to a larger central data center at the backend network's core. Nevertheless, the Edge layer also encounters power and processing limitations that must be addressed.

5G represents the most recent advancement in cellular network technology, offering several benefits, with one of the key advantages being low latency. Low latency refers to the time data travels between endpoints. With 5G, latency can be reduced to below one millisecond, enabling near real-time communication between devices. However, this low latency also means that intrusion detection systems must be fast enough to keep up with the high-speed data transfer rates of 5G networks. If the intrusion detection system is too slow, it can become a bottleneck in the network, causing delays and reducing the overall data transfer speed. Therefore, intrusion detection systems must be designed to be fast and efficient enough to operate seamlessly in 5G networks and ensure that security risks are identified and addressed on time.

In order to address these challenges, a proposed solution involves implementing an ML Intrusion Detection System within a zero-trust framework. This system aims to enhance security for IoT devices connected to 5G networks through the Edge layer. (HaddadPajouh et al., 2020).

This system can detect and classify malicious network traffic in real time while operating in a power-constrained environment like a 5G Edge layer data center. The system monitors the traffic of end-user IoT devices before sending it to the 5G core network for further processing.

Several datasets representing multiple IoT industries have been used to train and test the system, which contains over 5 million records of simulated traffic and real one. The goal is to develop an effective intrusion detection system that can work within the constraints of the Edge layer environment while providing adequate security for IoT devices.

Seven supervised ML models are developed and evaluated to classify several types of attacks. These models include bagging, boosting and neural networks algorithms. The research concludes that a machine learning model can accurately detect and classify intrusions in real time.

The Decision Tree (DT-CART) and Extreme Gradient Boosting (XGB) classifiers outperforms the other models, with the combined performance accuracy metrics, lowest false alarm rate, and shortest classification time per record, making it suitable for processing-constrained environments like 5G IoT devices connected to Edge layers.

Table of Contents

List of Figures

List of Tables

List of Symbols

k	Number of folds in cross validation
f_m	function in the functional space W
m	number of trees in XGB
x	Raw original value in the feature column
$\bar{X}$	The mean of the values in the feature column
S	Standard deviation of the values in the feature columnf_m
W	The space of all decision tree

List of Acronyms

3GPP	The 3rd Generation Partnership Project
4G	4th Generation of telecommunications networks
5G	5th Generation of telecommunications networks
ADFA	Australian Defence Force Academy
AI	Artificial Intelligence
AIDS	Anomaly-based Intrusion Detection System
AUC-ROC	Area Under the Curve of the Receiver Operator Characteristic
BA	Bagging Algorithm
Botnet	Robot network
CART	Classification And Regression Tree
CIS	Continuous Intelligence Services
CoAP	Constrained Application Protocol
CSV	Comma-Separated Values
CV	Cross-Validation
D2D	Device-to-Device
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Trees
EC	Edge computing
EDA	Exploratory Data Analysis
eMBB	Enhanced Mobile Broadband
GaussianNB	Gaussian Naïve Bayes

GB	Gradient Boosting
HIDS	Host-based IDS
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICU	Intensive Care Unit
IDS	Intrusion Detection System
IIOT	Industrial IoT
IMT	International Mobile Telecommunications
IoT	Internet of Things
IP	Internet Protocol
ITU-R	International Telecommunications Union Radiocommunications Sector
i-ZTA	Intelligent-Zero Trust Architecture
KNN	K-nearest neighbor
M2M	Machine to Machine
MEC	Multi-access Edge Computing
MiTM	Man in The Middle
ML	Machine Learning
mMTC	Massive Machine Type Communications
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrow Band IoT
NIDS	Network-based IDS
NN	Neural Network
O-RAN	Open Radio Access Network

PA	Policy Administrator
PCA	Principal Component Analysis
PDP	Policy Decision Point
PE	Policy Engine
PEP	Policy Enforcement Point
PLC	Programmable logic controller
RF	Random Forest
RNN	Recurrent Neural Network
SBA	Service-Based Architecture
SDN	Software-Defined Networking
SEIT	School of Engineering and Information technology
SIDS	Signature-based Intrusion Detection System
SMF	Session Management Function
SSO	Standards Setting Organizations
SVC	Support Vector Classifier
SVM	Support Vector Machine
SYN	Synchronize
TCP	Transmission Control
Protocol	
UDP	User Datagram Protocol
UNSW	University of New South
Wales	
UPF	User Plane Functions
URLC	Ultra-Reliable and Low Latency Communications
USD	United States Dollars

XGBoost	eXtreme Gradient Boosting
ZT	Zero Trust

Chapter 1—Introduction

1.1 Background

The rapid urbanization of the world's population poses sustainability challenges, with 55% of the population currently living in cities, which is expected to rise to 68% by 2050 (United Nations, 2018). Cities are turning to "smart city" technology, using IoT (Internet of Things) to address challenges and to improve habitability (Kim et al., 2017).

However, these cities must address the cybersecurity risks that could potentially harm critical functions to ensure that the deployment of IoT is truly efficient and effective. Smart city IoT solutions include smart lighting, smart parking, smart law enforcement, and public safety solutions. Outdoor solutions would require distinct types of connectivity apart from traditional Wi-Fi. However, these cities must address the significant cyber security risks that could harm critical functions to ensure IoT deployment is truly secure.

Similarly, one of the main targets of the new era industrial revolution called Industry 4.0 is the digital transformation aiming towards connecting all assets and extracting valuable insights from the accumulated data (Gadre et al., 2020).

As per IDC's European Continuous Intelligence Services (CISs), 43% of European organizations identified IT transformation as one of the leading network issues for the upcoming two years, ranking second to security-related challenges. (Bakkers, 2020).

The Internet of Things (IoT) progress is propelled by the convergence of Machine to Machine (M2M) communications, the proliferation of connected devices and big data

analytics. With this, more sensors and machines are going to be connected, requiring ultra-reliable and low-latency communication capabilities.

These capabilities are crucial for applications such as remote medical surgery, and drone-based inspections of remote power lines, etc.

These use cases, such as massive machine-type communications in smart factories, require many connected devices, typically transmitting non-delay-sensitive data. Fire sensors in high-risk forest fire areas and vehicle and road safety collision avoidance require continuous high bandwidth and mobility coverage. The connectivity requirements of IoT devices for the above scenarios can only be adequately met solely through 5G coverage, as opposed to the present constraints of Wi-Fi coverage and bandwidth-limited technology.

5G networks can enable dynamic cellular communication, allowing users to communicate without relying on base stations or core networks. IoT Machine-to-machine (M2M) communication can also occur, allowing large numbers of devices to communicate with one another via base stations.

According to ITU-R Recommendation M.2083 (ITU-R, 2015) explained in Figure 1-1 below, 5G services can be used for three scenarios: “Massive Machine Type Communications (mMTC), Ultra-Reliable and Low Latency Communications (URLC), and Enhanced Mobile Broadband (eMBB)”. eMBB focuses on human-centric uses, including accessing multimedia content and data, while URLC is used for industrial manufacturing, medical surgery, transportation safety, etc. mMTC is used for transmitting low volumes of data between many devices, including sensors, and is a combination of M2M and the Internet of Things (IoT).

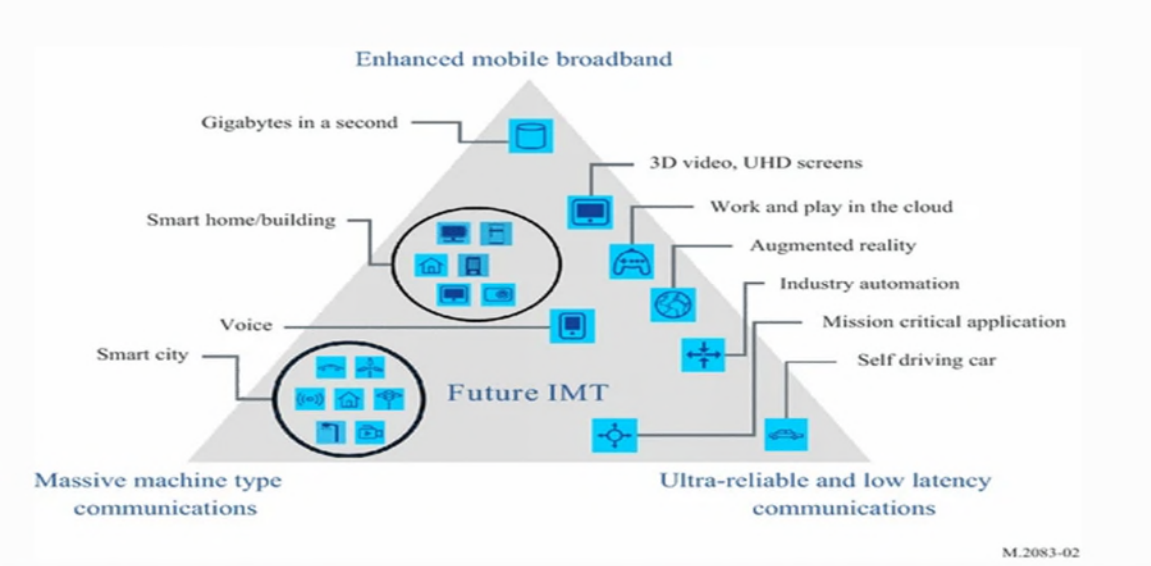


Figure 1-1. 5G Usage Scenarios (ITU-R, 2015)

Given the prospects of International Mobile Telecommunications (IMT) discussed earlier, the rise of IoT has brought about new challenges. These challenges include addressing low latency requirements, managing limited capacity of IoT devices, ensuring continuous services despite intermittent connectivity, and bolstering security measures. However, these challenges can only partially be resolved using a centralized cloud computing architecture. (Chiang et al., 2016)

Increasing the volume of data can lead to a decline in network performance. Additionally, with cloud-based services, computation is located far from end users. This process requires significant network resources for such volume of data transmission, leading to significant latency within the network.

This increased latency poses a critical issue for time sensitive IoT applications, which require very short response times. This issue is especially important since latency can affect safety and emergency response as illustrated in Figure 1-2 below.

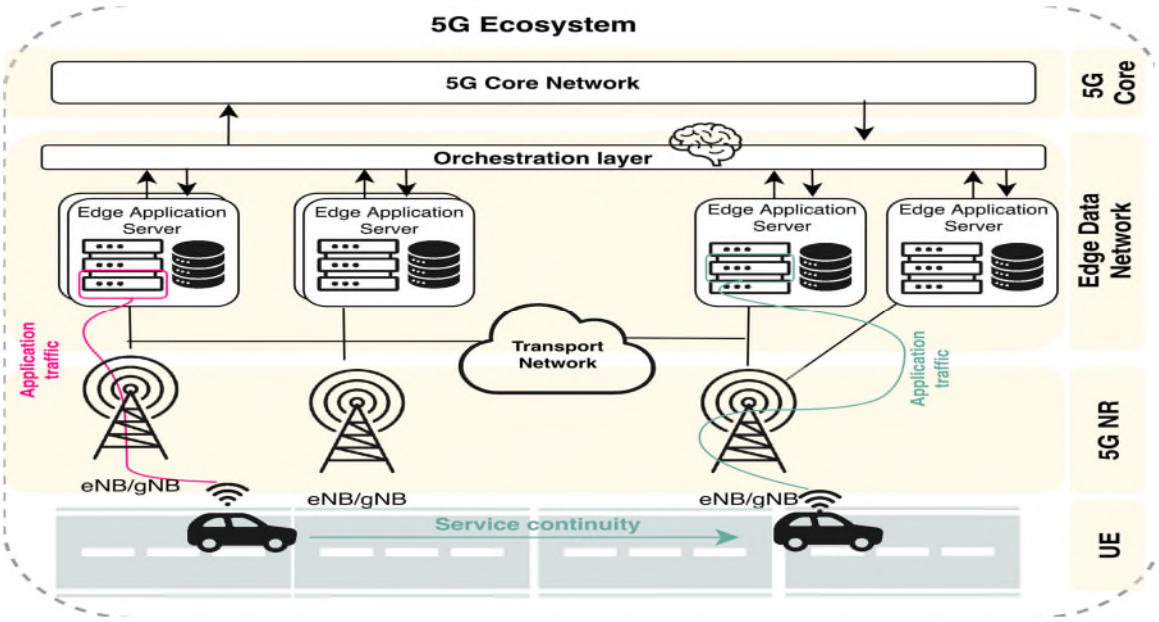


Figure 1-2. Edge reduces latency (Johann & Nina,2022)

Reducing transmission time by processing data in computation nodes located near the end user leads to reduction in power consumed which is a desirable outcome.

Therefore, it is crucial to offload energy-consuming application computations to the network's edge (Mao et al., 2017) to extend the longevity of IoT device batteries. Recent research has focused on effectively utilizing edge network capabilities to meet such requirements. (Stankovic, 2014)

Mobile Edge Computing (MEC) is a technique that enables the processing and storage of enormous volumes of data generated by various IoT devices at the network's edge, in proximity to the user, rather than sending it to the back end of the central cloud. (Montori et al., 2018).

Since the computation happens near the end user, traffic flow peaks can be reduced, and bandwidth requirements and latency problems can be mitigated (Khan et al., 2019) (Das et al., 2018).

Edge computing also provides advantages in data security and privacy (Hsu et al., 2018). Despite 5G cloud core service providers offering comprehensive security solutions, a leak could have severe consequences. (Sha et al., 2017).

Edge computing enables the implementation of the most suitable security measures near the end user, decreasing the chances of data leakage and reducing stored data. As a result, this approach minimizes security risks. (Tange et al., 2020). Figure 1-3 below illustrates Edge-based smart factory connectivity.

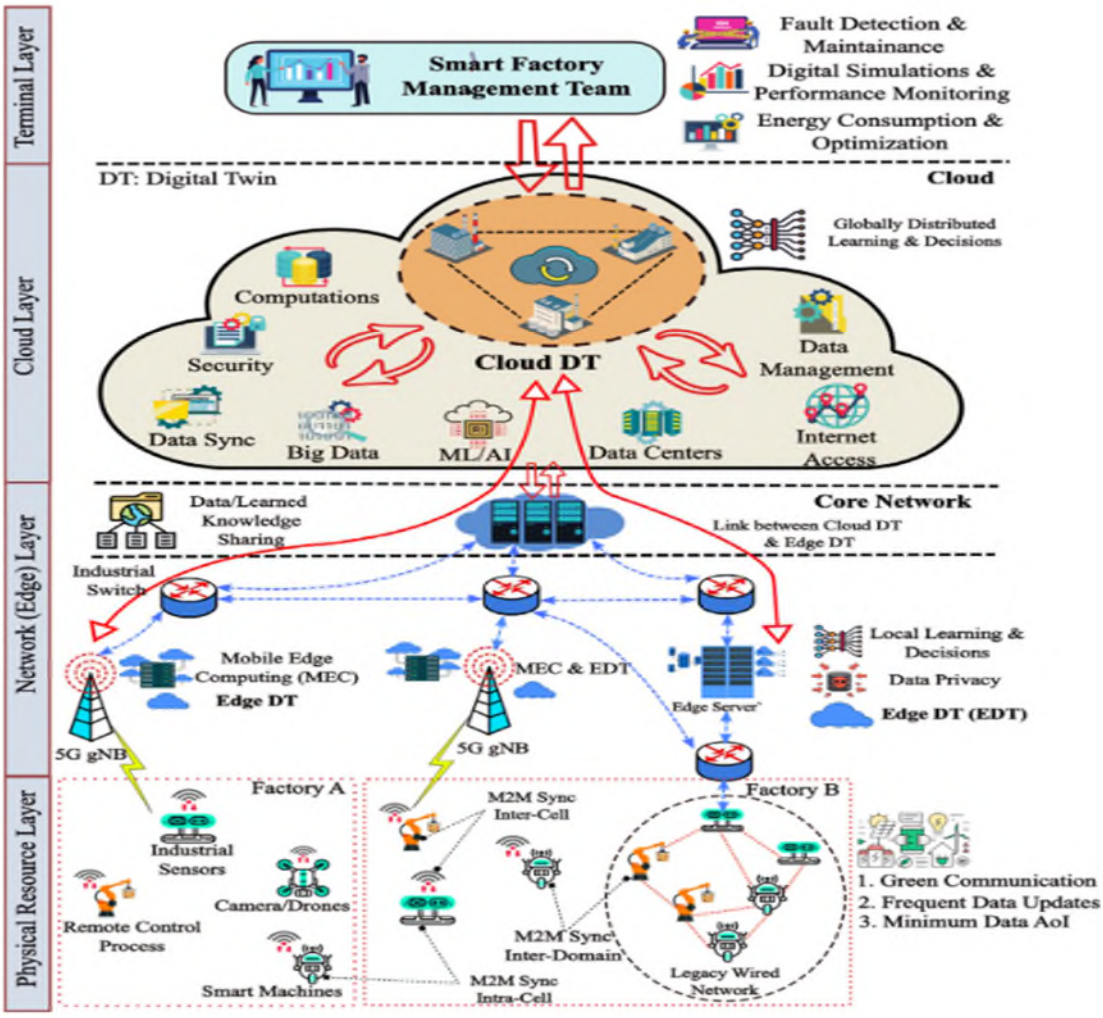


Figure 1-3. Edge-based architecture for smart industries (Zeb et al., 2022)

1.2 Research Motivation

Although traditional security measures such as authentication, encryption and access control are employed to secure IoT devices, they are inadequate, particularly given the lack of a “zero-trust” approach in most current deployments. Instead, security measures mostly rely on a “defense in depth” perimeter philosophy to protect assets (NIST SP 800-207, 2020).

The major motivation behind this research is recognition of the lack of machine learning-powered intrusion detection systems in existing 5G IoT network deployments.

This research focuses on applying Machine Learning-supervised algorithms for detecting IoT network attacks. This study identifies several types of IoT threats and explores the use of both shallow (e.g., decision tree (DT), random forest (RF)) and deep machine learning (e.g., neural network (NN), based intrusion detection systems (IDS) in the IoT environment.

The performance of these models is evaluated using four benchmark datasets that represent different IoT industries. Mainly healthcare, Industrial IoT (IIOT), and commercial IoT. Various performance metrics such as accuracy, area under the curve (AUC), F1-score, and time to perform classification time per record are used to assess the performance of the above models.

The datasets include contemporary attacks and normal network traffic. The goal is to find a machine learning classifier that is suitable to be used as an intrusion detection mechanism in a resource-constrained environment.

1.3 Problem Statement

The adoption of 5G has resulted in an increase in the number of potential targets of IoT cybersecurity attacks, as of 2017 alone, 90% of organizations using IoT devices suffered from cyber breaches. (Banafa, 2018), (Ghorbani et al., 2020)

5G IoT traffic is expected to be processed at the edge layer to reduce the load on the backend network and for faster processing, cutting down latencies for low latency use cases.

5G 3GPP standard supports up to 1 million IoT devices per 1 square kilometer. Whereas 4G supports 60,680 devices only considering the same coverage area (Kim, 2020).

1.4 Thesis Statement

A Classification model for detecting IoT malicious traffic is required by 5G providers across different IoT industries to protect the network and customers from IoT cyber-attacks.

As the utilization of IoT devices increases across various industries, safeguarding networks and customers against IoT cyber-attacks has emerged as a crucial concern for 5G providers. To address this concern, a classification model is necessary to identify malicious traffic in IoT devices. This model should be capable of detecting malicious network traffic patterns, alerting the security team to take necessary measures to prevent potential threats. In addition, the suggested framework goes further by incorporating an intelligent zero-trust framework to take action based on the classification results and block any traffic identified as malicious by the model. The model's utility extends across diverse IoT industries, from healthcare to industrial applications, to counter IoT cyber-attacks.

This classifier enables users to detect malicious traffic in their IoT devices and act appropriately to safeguard their networks and data. In addition to 5G providers, any IoT user can leverage the classifier to augment their security posture. Users can add an extra layer of protection by integrating the model into the security solutions provided by various IoT devices.

Furthermore, 5G and Edge cloud providers can transform this classification model into a security feature for the edge layer. By providing this service, providers can distinguish themselves in the market by offering a unique value proposition that enhances their customers' security posture. It can also appeal to new customers who prioritize security and seek dependable and secure 5G and Edge cloud providers.

1.5 Research Objectives

This research aims to develop a machine learning classifier capable of accurately detecting malicious IoT traffic from IoT users across various industrial domains, with the highest possible accuracy and a time to predict less than 0.5 millisecond per record. The machine learning classifier will enable the 5G network provider and IoT users, who may own their Edge data center, to implement an intelligent and light intrusion detection system based on their unique traffic patterns by productizing the classifier instead of relying on static policies.

1.6 Research Questions and Hypotheses

The following questions served as the basis of this research project:

RQ1: Which features are the most significant for detecting malicious IoT traffic across multiple domain datasets?

RQ2: Can a Machine Learning classification model be developed to detect IoT malicious traffic?

RQ3: Which ML classifiers are best for detecting malicious IoT traffic in 5G networks?

The following hypotheses served as the basis of this research project:

H1: Features state_number, ltime, tnbpdstip, mqtt_msgtype, tcp_time_delta, sphone_signal, seq, icmp, sttl and drate and are the most significant for detecting malicious IoT traffic across multiple domain datasets.

H2: An ML classification model will detect malicious IoT traffic with at least 95% accuracy.

H3: XGB, DT and RF are the best classification algorithms for detecting malicious IoT traffic in 5G networks.

1.7 Scope of Research

The scope of the research will be limited to publicly available datasets. A binary classification ML-based IDS with a smart zero-trust framework is proposed. The system provides security for IoT devices connected to 5G networks via the Edge layer. Four datasets representing malicious traffic scenarios from different IoT industries have been selected to be the scope of this research, along with seven different machine-learning models (DT-CART, RF, GaussianNB, SVC, RNN, Bagging, and XGBoost). The first dataset we used, the Bot-IoT dataset, captures commercial temperature, pressure, and humidity IoT sensor data and offers botnet attacks against these commercial IoT devices. The second dataset represents the IoT healthcare industry. Whereas the third dataset represents the industrial IoT (IIoT) telemetry use case. The fourth dataset is a real 5G

network with malicious nodes attacking the Edge server deployed in the 5G Multi-access Edge Computing Environment (MEC).

1.8 Research Limitations

The primary constraint of this study is the lack of some datasets representing an essential segment of the IoT industry: the mission-critical tactical datasets for military applications. The traffic features studied in the current datasets come from civil applications. In order to effectively implement the proposed classification model, the algorithm needs to be trained on different types of traffic representing many IoT industry domains.

However, testing on multiple datasets from different IoT industrial domains provides evidence that the model can be generalized to other types of traffic, such as military IoT applications.

1.9 Organization of Praxis

The remaining sections of the research are organized as follows: Chapter 2 reviews existing research methods for intrusion detection. Chapter 3 offers an overview of the system, methodology and experiment details, including the datasets used. Chapters 4 presents the experimental results and their discussion. Lastly, Chapter 5 concludes the research by summarizing the findings.

Chapter 2—Literature Review

2.1 Introduction

This chapter comprises five primary parts, all of which aim to offer an introduction and contextualization of the issue addressed by this research and some potential solutions found in the literature. Section 2.1 provides an overview of this chapter and its contents. Section 2.2 delves into the primary concern this praxis addresses: the increasing frequency and intensity of cyberattacks targeting IoT devices, particularly considering the expanding number of IoT devices and the impact of 5G technology on such growth. Section 2.3 provides an overview of the zero-trust architecture and links it to the research's topic. Section 2.4 explores the most prevalent types of attacks that target IoT devices and networks. Intrusion detection systems are also discussed, including machine learning-based intrusion detection techniques. Finally, Section 2.5 summarizes the literature review and highlights gaps in the literature, underscoring the necessity for this praxis to develop a real-time intrusion detection system for IoT devices in 5G networks.

2.2 IoT technology and cyberattacks

The reliance on digitalization and Internet-of-Things (IoT) has soared in recent years (Li et al., 2015), leading to a surge in security incidents, including but not limited to malware attacks (McIntosh et al., 2019), denial-of-service (DoS) attacks (Sun et al., 2018), zero-day attacks (Alazab et al., 2011), healthcare data breaches (Moganedi, 2018) among many others. A visualization of the global installed base of IoT-connected devices from 2015 to 2025, measured in billions, is depicted in Figure 2-1 (Radanliev et al., 2018).

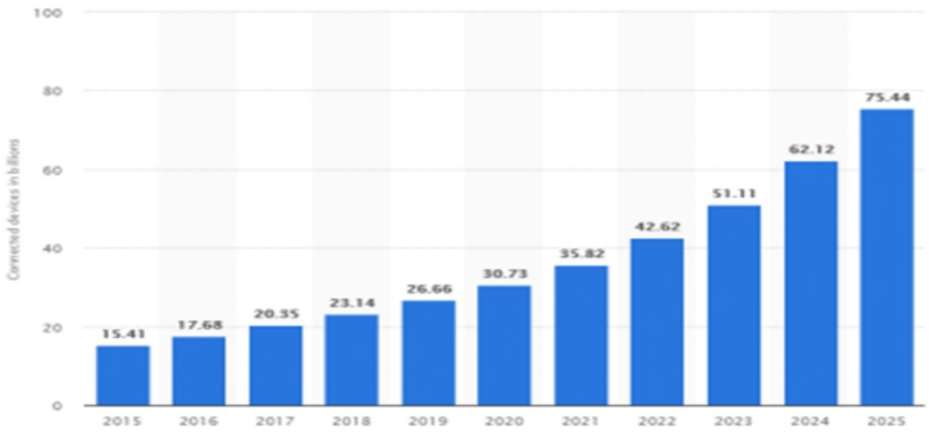


Figure 2-1. IoT worldwide devices (Radanliev et al., 2018)

With their limited memory and storage capacities, IoT devices are highly vulnerable to cyberattacks (Sharma, 2019). This vulnerability arises from the challenges of implementing cybersecurity measures on these devices. Moreover, the wide variety of IoT devices, their diverse types, and communication methods make it challenging to devise a single efficient, interoperable, and scalable solution that can work for all IoT devices (Roman et al., 2013).

Typically, attacks begin by targeting IoT devices and then spread horizontally or vertically to other network components within the same network. As per (Fischer, 2014), the cost to the global economy from cybercrime is 400 billion USD. According to (Zarour et al., 2020), healthcare data breaches impacted a staggering 249.09 million individuals between 2005 and 2019. Of this total, a significant 157.40 million individuals were affected within the past five years alone. Compared to other countries, the USA incurred the highest cost from data breaches, particularly in the healthcare sector. While a typical data breach results in an $8.19 million loss, the average cost of a healthcare data breach involving around 25,575 records in the USA soared to $15 million (HIPAA Journal, 2022).

Remarkably, the healthcare industry saw the most significant rise in the cost of a breached record. rising by 19.4% (Wikina, 2014) (Ponemon, 2015). While Data breach costs surged 13% from 2020 to 2022 (IBM, 2022)

The Internet of Medical Things (IOMT) has had a substantial impact on the healthcare industry, as they store sensitive data to provide quick access and better patient care.

The use of wireless communication for data exchange also introduces potential risks of privacy violations, as such exchanges can expose the underlying system to multiple attacks. Consequently, ensuring security is a challenge in current research that that if not adequately addressed could potentially limit the deployment of IoT.

3GPP (3rd Generation Partnership Project) is an engineering consortium responsible for formulating technical specifications. by collaborating with hundreds of different entities. After the technical specifications are developed, the seven regional Standards Setting Organizations (SSOs) within the 3GPP partnership transform them into standards, as clarified in Figure 2-2 below.

Figure 2-2. Regional 3GPP Standard Organizations (Lorenzo,2017)

3GPP establishes specifications for comprehensive end-to-end cellular systems, including user equipment, core network, radio access and the services around them.

5G represents a generational shift, thanks to its defined technical specifications, including sub-1ms latency and downlink speeds exceeding 1 Gbps. This ultra-low latency is crucial to many mission-critical use cases. The 5G network is also efficient and appealing due to its ability to support over 60,000 connections and its large bi-directional bandwidth (Ravalli, 2016). With a focus on reducing power consumption by up to 90% in devices and network centers, 5G aims to establish itself as a greener technology (Reshma, 2013), (Akhil, 2015).

Industry initiatives working on 5G have identified eight critical requirements shown in Table 2-1 below. Thus, for all the reasons mentioned above, it is clear that 5G is the future technology for IoT connectivity.

Table 2-1. 5G key specification table (Sathiya et al., 2015)

PARAMETER	PERFORMANCE
Network capacity	10000 times the capacity of the current network
Peak data rate	20Gbps downlink and 10Gbps uplink
Cell edge data rate	100 Mbps
Latency	< 1 ms
Bandwidth	Possibly 1-2 GHZ
Connection density	1 million connected devices per square km (0.38 sq. miles)
5G mobility	500km/h high speed
Spectral efficiency	30bits/Hz downlink and 15 bits/Hz uplink

IoT applications can be endless. Akpakwu et al. (2018) provided a summary of the differences in requirements for the most important applications. These requirements can be served well by the 5G technology as per the specifications discussed earlier in Table 2-1.

The adoption of 5G has increased the potential targets of IoT cybersecurity attacks (Ghorbani et al., 2020). As of 2017 alone, 90% of organizations using IoT devices

suffered from cyber breaches (Banafa, 2018). A general 5G cellular network architecture is shown in Figure 2-3.

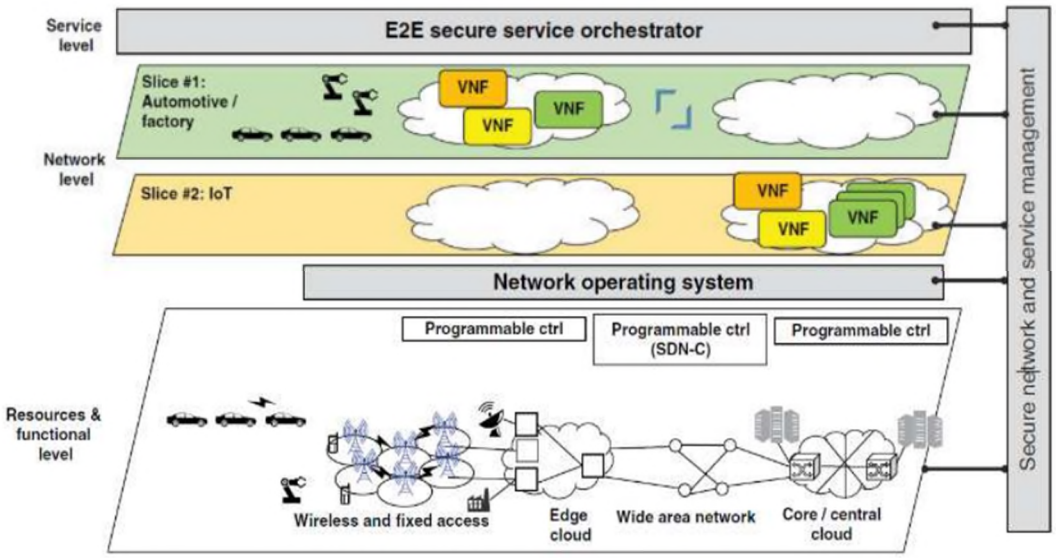


Figure 2-3. End to End 5G cellular network architecture (Noohani et al., 2020)

Edge computing, also known as multi-access edge computing (MEC), involves moving data processing to the edge of communication networks. By bringing computing near the end user, MEC eliminates the need for data to travel to the cloud for processing. This reduces latency for end-users and reallocates network resources.

In 5G systems, the focus is on delivering enhanced network performance. Each development phase of 5G, as defined in a 3GPP (3rd Generation Partnership Project) release, targets one of these enhancements. 5G Release 16, delivers URLLC through technical enhancements in new radio and core technologies, as well as leveraging MEC and its orchestrated services. The edge can host several services that cater to the diverse needs of various industry verticals, such as health, finance, and transportation. The architecture depicted above displays the Edge cloud functioning as a layer that processes

traffic prior to transmitting it to the core or central cloud. Additionally, the introduction of logical network slices, a concept unique to 5G technology, is evident.

To cater to various needs, the Network slicing architecture offers multiple autonomous service-level agreements. A slicing network can simultaneously offer several different services. The 3GPP specifications are thoroughly explained in (3GPP, 2018) and by (Bianchi et al., 2016).

The focus of this research is on the IoT slice illustrated in Figure 2-4 below.

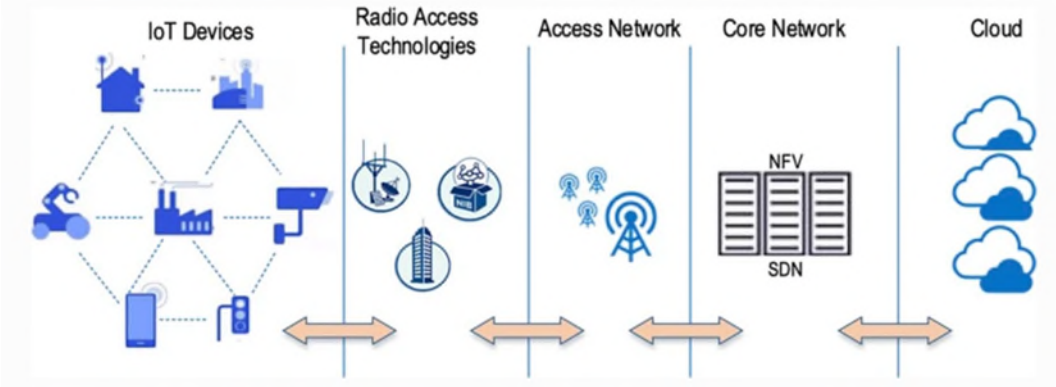


Figure 2-4. 5G IoT slice (Li et al., 2022)

A detailed analysis of various existing and emerging technologies is presented in Table 2-2, which summarizes their 5G KPIs (Silva et al., 2017).

Table 2-2 fulfill all the requirements. Whereas Table 2-3 provides an overview of various application protocols relevant to the IoT, each offering distinct performance features. (Silva et al., 2017).

Table 2-2. KPIs analysis for IoT transmission types (Silva et al., 2017)

KPI	ZigBee	BLE	Wi-Fi	SigFox	LoRa	EC-GSM-IoT	eMTC	NB-IoT
Extended Coverage	x	x	x	✓	✓	✓	✓	✓
Reliability	x	✓	✓	x	x	✓	✓	✓
Low Deployment Cost	✓	✓	✓	✓	✓	✓	✓	✓
Long Battery-Life	✓	✓	✓	✓	✓	✓	✓	✓
Low Latency	x	✓	✓	x	x	✓	✓	✓
Network Scalability	x	x	✓	x	x	✓	✓	✓
Mobility Support	x	x	x	x	x	✓	✓	✓
SLA support	x	x	x	x	x	✓	✓	✓
Support for Roaming	x	x	x	x	x	✓	✓	✓
Dedicated Spectrum	x	x	x	x	x	✓	✓	✓

Table 2-3. Summary IoT application layer protocols (Silva et al., 2017)

Applicable Protocol	Supported Architectures	Applicable Transport	Lossy nature	Available Compute Resources	Security Requirement	Developer
CoAP	Request/Response	UDP/IP	Very excellent	10Ks/RAM Flash	Medium – not compulsory	IETF
MQTT	Publish/Subscribe	TCP/IP	Fair enough	10Ks/RAM Flash	Medium – not compulsory	IBM (OASIS)
MQTT-SN	Publish/Subscribe	UDP/IP	Fair enough	10Ks/RAM Flash	Medium – not compulsory	
AMQP	Publish/Subscribe	TCP/IP	Fair enough	-	Very High - compulsory	John O'Hara (JP Morgan)
XMPP	Publish/Subscribe Request/Response	TCP/IP	Fair enough	10Ks/RAM Flash	Very High - compulsory	Jabber Open Source
Continua HDP	Publish/Subscribe Request/Response	UDP/IP	Fair enough	10Ks/RAM Flash	-	Continua Health Alliance
DDS	Publish/Subscribe	UDP/IP	Fair enough	100Ks/RAM Flash	Very High – not compulsory	OMG

As we shift the focus towards 5G security, it becomes apparent that significant efforts are required to guarantee the security of the 5G network system (Panwar et al., 2016).

Ahmed et al. (2018) proposes potential techniques to mitigate security challenges and standardize efforts for 4G and previous generations.

Meanwhile, Roman et al. (2018) investigated security attacks targeting mobile networks.

Rupprecht et al. (2018) presented considerable research on future mobile network security with the aim of comprehensively understanding mobile network security and identifying research challenges. However, the integration of large-scale Internet of Things and modern technology concepts will result in more diverse security challenges in 5G. Unfortunately, MEC offers performance and cost-efficiency benefits, but they also have security weaknesses (He et al., 2018).

Table 2-4 illustrates the evolution of security across different cellular network generations. (Noohani et al., 2020).

Table 2-4. Progression of Security from1G to 4G (Noohani et al., 2020)

Network	Security Mechanisms	Security Challenges
1G	No explicit security and privacy measures.	Eavesdropping, call interception, and no privacy mechanisms.
2G	Authentication, anonymity and encryption-based protection.	Fake base station, radio link security, one way authentication, and spamming.
3G	Adopted the 2G security, secure access to network, introduced Authentication and Key Agreement (AKA) and two way authentication.	IP traffic security vulnerabilities, encryption keys security, roaming security.
4G	Introduced new encryption (EPS-AKA) and trust mechanisms, encryption keys security, non-3G Partnership Project (3GPP) access security, and integrity protection.	Increased IP traffic induced security, e.g. DoS attacks, data integrity, Base Transceiver Stations (BTS) security, and eavesdroping on long term keys. Not suitable for security of new services and devices, e.g. massive IoT, foreseen in 5G.

While the 5G standard includes many new security features (Arfaoui et al., 2018), the security system depicted in Figure 2-5 does not address specific security risks and corresponding solutions (Ijaz et al., 2019). However, there are various existing security measures that have been improved upon or newly explicitly developed for 5G. The LTE security requirements form the cornerstone and are essential safety guidelines for futuristic wireless networks. (He et al., 2018).

Recognizing the key supporting technologies in 5G, like MEC, can easily highlight security issues in the system.

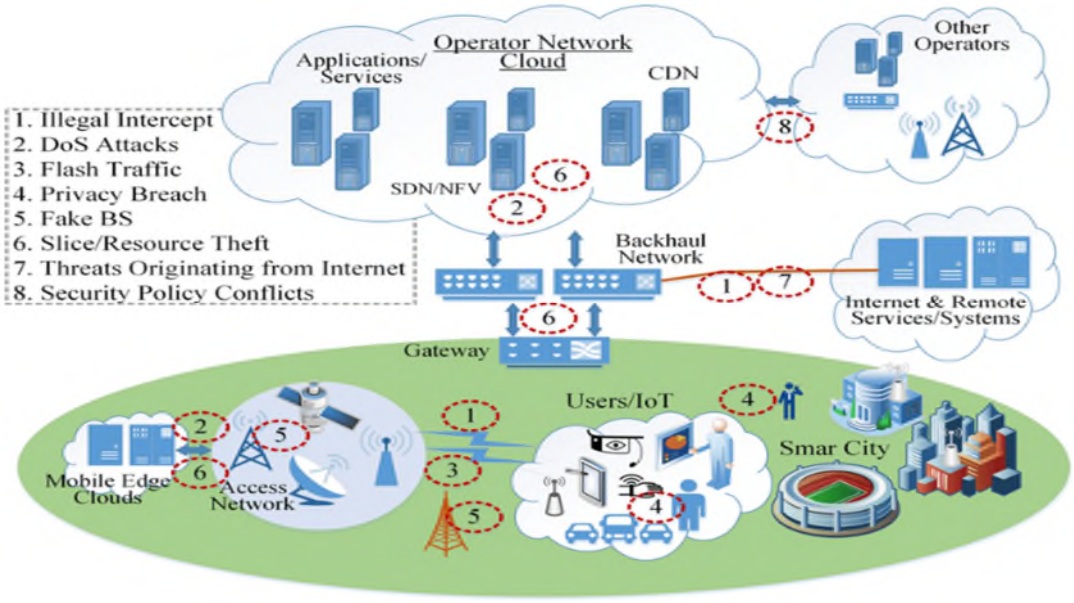


Figure 2-5. The security threat landscape in 5G networks (Ijaz et al., 2019)

In addition to the 5G standard's traditional perimeter-based security techniques, such as enhanced security encryption keys and new authentication methods, a systematic literature review on 5G security by (Sahni, 2022) could be a valuable reference point. Deploying security functions can offer several opportunities to strengthen network

security. Basic firewall application software can be considered the first step toward software-defined and virtualized network security.

However, other security techniques focus on AI (Artificial Intelligence) models. Monitoring numerous devices and analyzing big data will require an automated intelligent system that employs innovative AI algorithms and techniques, making cybersecurity one of the most promising application areas for AI.

Due to the distributed nature of MEC systems, shown in Figure 2-5 above. MEC systems –also called Edge cloud- often have limited resources at each location. As a result, task requests may need to wait for sufficient resources to become available in a single MEC system unless there is a collaboration between edge entities.

While connecting MEC systems in different domains using standard networking is easy, they still suffer from latency. The transport network is critical for ensuring end-to-end Service Level Agreements (SLAs).

As illustrated in Figure 2-6 below, MEC plays a crucial role in deploying 5G networks.

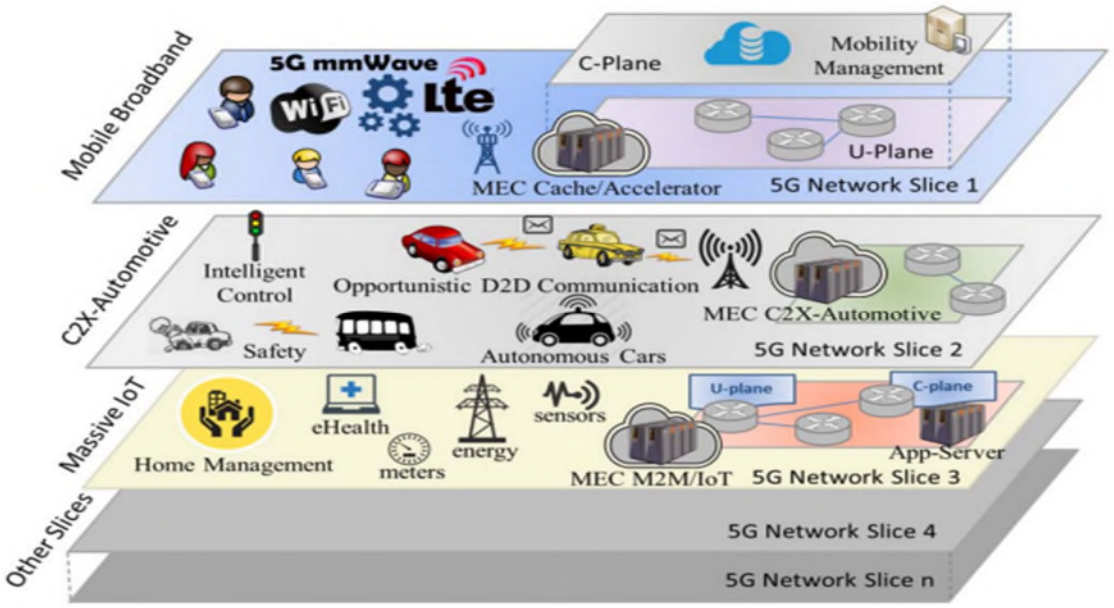


Figure 2-6. Network slicing and the role of MEC (Taleb et al., 2017)

2.3 Zero Trust Architecture (ZTA)

Traditional security frameworks for networks need to improve when it comes to providing security in complex and dynamic environments, such as the next-generation networks that are agile, mobile, and heterogeneous. These frameworks assume a perimeter of trust that is protected against unauthorized access.

However, identifying the network perimeter is challenging, and assumptions of trust among network entities and services can lead to serious vulnerabilities that can be exploited as illustrated in Figure 2-5 above.

According to NIST (National Institute of Standards and Technology) “the concept of Zero Trust (ZT) refers to a constantly evolving set of cybersecurity approaches that shift defenses away from fixed, network-based perimeters and prioritize users, assets, and resources instead”. (NIST SP 800-207, 2020)

The ZT concept was published in 2020 after 5G standardization was published. The focus of zero trust is to protect resources such as assets, services, workflows, and network accounts. On the other hand, network segments are no longer regarded as the primary determinant of the resource's security posture.

In ZTA, assets are not inherently trusted based on location, like being wired locally or connected wirelessly to a 5G network.

Authentication and authorization, including for subjects and devices, are separate functions before accessing network resources.

The abstract access model depicted in Figure 2-7 below demonstrates that a subject's traffic must pass a policy decision point (PDP) and a policy enforcement point (PEP) to gain access to an enterprise resource.

In this research, the subject is the IoT device, and the enterprise resources are the 5G network resources. Before the PDP/PEP grants access, the system must verify the subject's authenticity and validate the request.

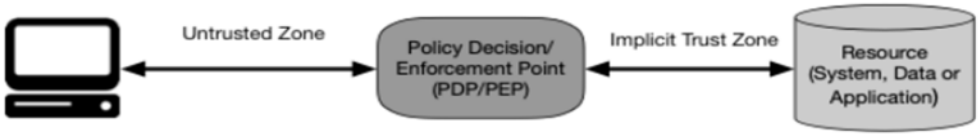


Figure 2-7. ZTA (NIST SP 800-207, 2020)

ZTA deployment consists of logical components that can be hosted on a cloud-based service. Figure 2-8 shows the basic relationship between those components.

In this research, we are proposing ZTA to be deployed in a cloud-based service which is the MEC layer connected to the IoT devices.

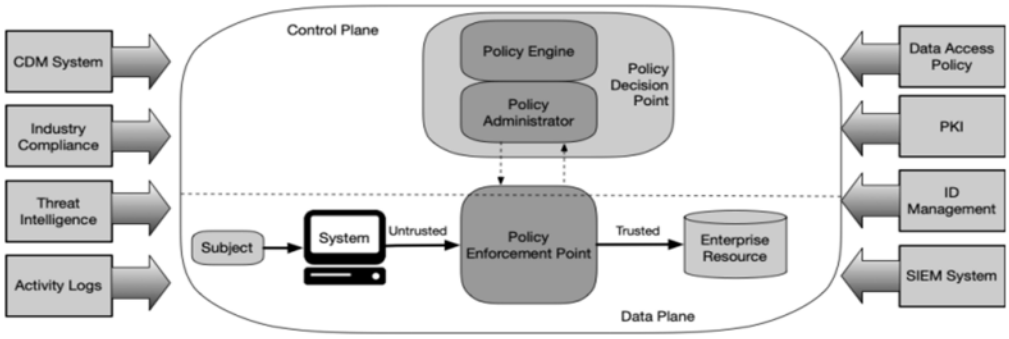


Figure 2-8. ZT core components (NIST SP 800-207, 2020)

Figure 2-8 divides the policy decision point (PDP) into the policy administrator and the policy engine. The policy engine (PE) is responsible for granting access to a resource, considering network policies and inputs from other sources. (NIST SP, 800-207)

This research proposes to base the decision on artificial intelligence classification results rather than a static policy. The policy administrator (PA) creates and terminates communication paths between a subject and a resource. The PA works closely with the policy engine (PE) and relies on its decision to determine whether to permit or deny a session. (NIST SP, 800-207)

The policy enforcement point (PEP) monitors and terminates connections between a user and an organization's resource. It collaborates with the policy administrator (PA) to forward requests and receive policy changes. PEP may be a single logical element or divided into components. Alternatively, it could be a standalone portal component.

The logical components communicate with each other via a control plane. And data is communicated via a data plane. The trust zone, where the network resource is located, lies beyond the PEP.

The 3GPP has developed 5G security frameworks (Cao et al., 2019). However, despite these carefully crafted protocols, the static nature of traditional security frameworks can still allow for lateral movements in the network perimeter. As a solution (ZTA) can be employed. (Ramezanpour et al., 2021)

The adoption ZT approach in security has been widespread in privately owned networks, and it can mitigate many security related issues in 5G-IoT. (Li, 2020; Bhattacharjya, 2020; Dhar & Bose, 2020).

Moreover, existing Security solutions must be improved to secure the ever-expanding array of IoT devices and applications. In response, the ZT security approach can counter many security problems like identity authentication and security monitoring of IoT environments. For that, (Li et al., 2022) suggested below architecture illustrated in Figure 2-9.

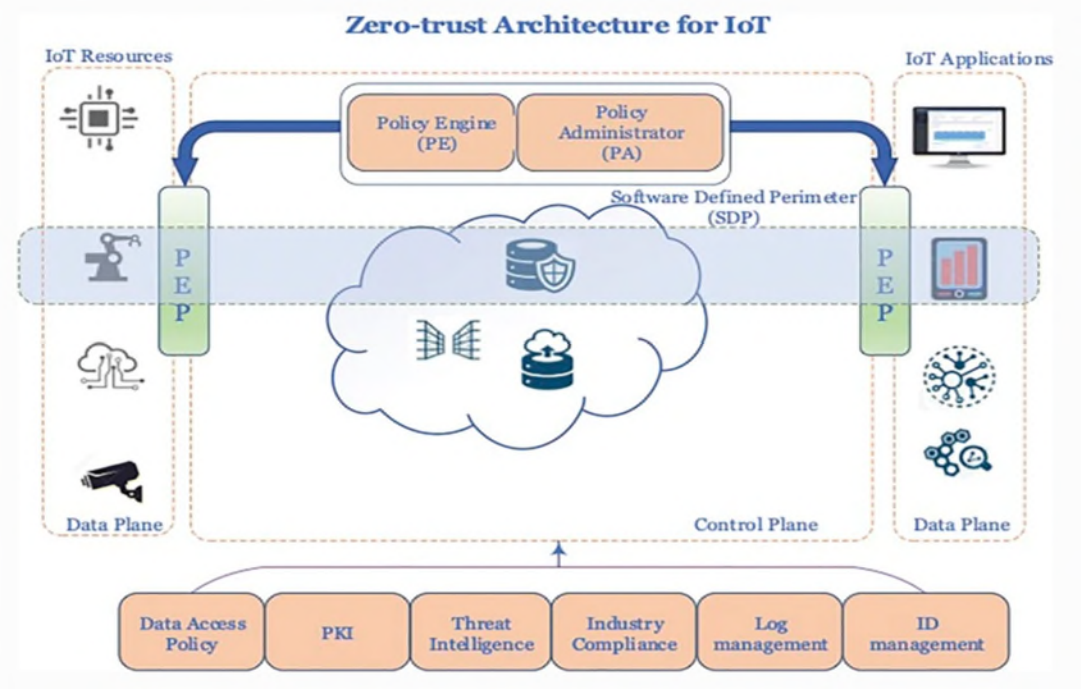


Figure 2-9. Zero Trust Security Architecture in IoT (Li et al., 2022)

Ramezanpour et al. (2021) introduced a new idea to enhance the previously discussed ZTA concept in IoT. They proposed an intelligent zero-trust architecture as a framework incorporating (AI) algorithms. Figure 2-10 explains their i-ZTA concept.

Comparing Figure 2-9 with Figure 2-10, we can notice that i-ZTA enhanced the traditional PE model and added intelligence to it by utilizing artificial intelligence to make decisions instead of relying on static policies. UE in Figure 2-10 can either represent User Equipment or an IoT sensor.

Baker and Waldron (2020) argue that ZTA offers an effective means for public and private sector leaders and policymakers to better manage cybersecurity risks associated with vulnerable equipment in global 5G networks.

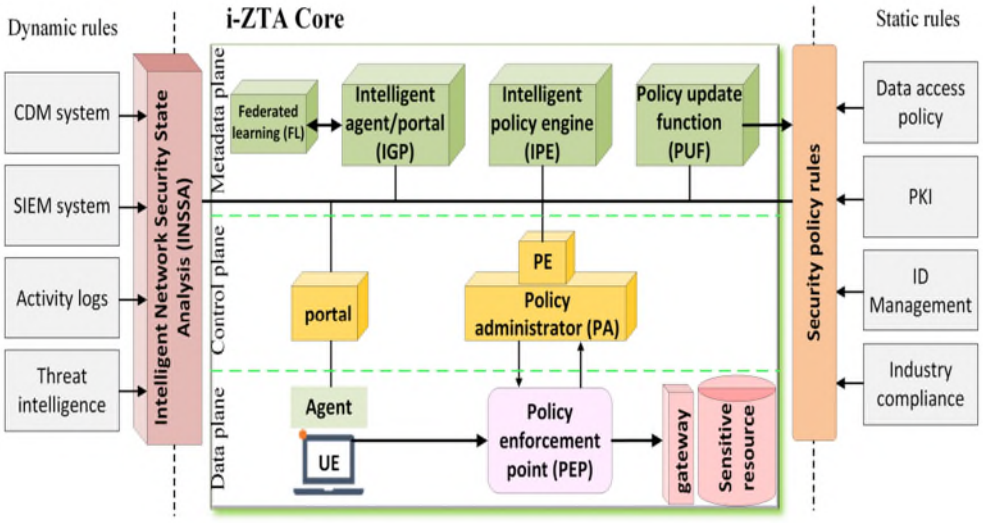


Figure 2-10. i-ZTA components with static rules (Ramezanpour et al., 2021)

Adopting a Zero Trust approach is the more effective strategy to achieve confidentiality, integrity, and availability. Fortunately, it appears that some branches of the United States Department of Defense understand the threat and are aware of some ways to mitigate it. (Baker et al., 2020).

However, the zero-trust model faces some challenges, as to Li et al. (2022):

1. Defining and enforcing security policies for an IoT system with millions of connected devices is complex.

2. 5G-IoT involves network slicing along with MEC, which challenges network service providers in establishing a hybrid policy definition.

3. The ZT approach requires intermediate monitoring applications which could lead to latency issues,

To avoid these limitations, the ZT model should be integrated with a Machine Learning module so the fastest trained ML models can automate the decisions. This

research explores this area by combining ZT and ML in the proposed end-to-end framework.

2.4 IoT types of attacks and Intrusion detection techniques

Cybersecurity attacks on IoT devices can result in data breaches, theft of personal information, and even physical harm to users. They are also vulnerable to a wide range of cyber security attacks due to their often-limited computing power, lack of powerful built-in security features, and the substantial number of devices in use. Distributed Denial of Service (DDoS) Attacks are some of the most common types of cyber security attacks on IoT devices.

The distinction between DoS (denial of service) and DDoS attacks also lies in the number of devices employed in the attack. A DoS attack typically stems from a single source, often a server or computer system. On the other hand, a DDoS attack involves multiple resources, potentially spanning across the globe and distributed across several systems. (Ghorbani et al., 2020; Shariati et al., 2015).

DDoS attacks overwhelm the victim with traffic, making it impossible to use. IoT devices are vulnerable to DDoS attacks because they often have limited processing power and bandwidth. Attackers can use compromised device networks as botnets to target IoT devices.

One example of a DDoS attack on IoT devices is the Mirai botnet attack in 2016, which exploited the security vulnerabilities of IoT devices to create a massive botnet that targeted and shut down major internet services. The Mirai botnet compromised over 600,000 IoT devices. It took down popular websites such as Twitter, Spotify, and Reddit (Antonakakis et al., 2017).

Kolias et al. (2017) explained that Mirai infected many IoT devices with weak security measures and default login credentials, allowing the botnet to launch large-scale DDoS attacks. The study suggests that IoT devices should be designed with stronger security measures, and users should be encouraged to change default passwords to prevent similar attacks.

In DDoS attacks, there are typically two types of victims: the primary victim, which is the intended target that becomes overwhelmed by the flood of incoming traffic, and the secondary victim (Pedreira et al., 2021), which is the compromised zombie device used to launch the attack. (Deshmukh et al., 2015).

In 2019, D. Sattar et al. proposed employing slice isolation as a measure to mitigate DDoS attacks in 5G networks.

In another study, M. A. Javed et al. (2019) investigated 5G artifacts, which encompassed DDoS attacks.

Overall, these studies highlight the prevalence of vulnerabilities in IoT devices and the significant risks associated with DDoS attacks. They have proposed various techniques and strategies to mitigate these attacks, including machine-learning algorithms, cloud computing models, and network-layer models. These findings underscore the importance of implementing robust security measures in IoT networks to safeguard against potential cyber threats.

Malware attacks involve installing malicious software on an IoT device, which can then control the device remotely. Malware can be delivered via email, social media, or other means and can be challenging to detect and remove once installed.

Gaurav et al. (2022) conducted a study investigating the application of ML to detect IoT malware attacks. Their findings indicated that ML can identify malware attacks and proposed that future research should enhance the accuracy and efficiency of these algorithms.

Another famous type of attack is the Man-in-the-Middle (MitM) attack. In his type, the attacker intercepts communications between an IoT device and its intended recipient, allowing attackers to steal or modify data. These attacks can be particularly damaging in IoT contexts, allowing attackers to control or manipulate physical devices like smart home appliances.

Khraisat et al. (2019) categorized IDS systems into two main categories: A Signature-based Intrusion Detection System (SIDS) and an Anomaly-based Intrusion Detection System (AIDS).

An Anomaly-based IDS (AIDS) establishes a baseline by learning patterns and behaviors of normal traffic. It then flags any traffic that deviates outside a predetermined tolerance level from the baseline as malicious. It only identifies traffic as malicious without specifying the type of attack since its primary function is to flag traffic that deviates from the normal baseline. (Zarpelão et al., 2017).

Although helpful in detecting novel zero-day attacks, this system requires continuous baseline training to reduce errors. On the other hand, Signature-based IDS (SIDS) compares incoming traffic to known attack patterns or signatures and flags any matches as malicious cyberattacks. As SIDS has prior knowledge of different attack types, it can categorize them based on their unique signatures, unlike AIDS, which can

only classify traffic as either normal or malicious. (Jyothsna et al., 2011). A summary of comparisons is provided in Table 2-5.

Table 2-5. Comparison of IDS technology types (Khraisat et al., 2019)

		Advantages	Disadvantages	Data source
Technology	HIDS	• HIDS can check end-to-end encrypted communications behaviour. • No extra hardware required. • Detects intrusions by checking hosts file system, system calls or network events. • Every packet is reassembled • Looks at the entire item, not streams only	• Delays in reporting attacks • Consumes host resources • Needs to be installed on each host. • It can monitor attacks only on the machine where it is installed.	• Audits records, log files, Application Program Interface (API), rule patterns, system calls.
	NIDS	•Detects attacks by checking network packets. •Not required to install on each host. •Can check various hosts at the same period. •Capable of detecting the broadest ranges of network protocols	•Challenge is to identify attacks from encrypted traffic. •Dedicated hardware is required. •It supports only identification of network attacks. •Difficult to analysis high-speed network. •The most serious threat is the insider attack.	•Simple Network Management Protocol (SNMP) •Network packets (TCP/UDP/ICMP), •Management Information Base (MIB) •Router NetFlow records

Introducing machine learning in intrusion detection enhanced it in several ways, such as: Increased accuracy: ML algorithms learn from extensive data to identify anomalies, enabling IDS to detect even minor deviations from normal traffic that traditional rule-based systems may miss. Real-time detection: Machine learning models can operate in real-time, enabling IDS to identify and respond to threats as they happen quickly. Adaptability: Machine learning models can adjust to changing network conditions and attack patterns, allowing IDS to remain current with the latest threats. Reduced false positives: By using machine learning algorithms to analyze network traffic, IDS can minimize false positives, reducing security analysts' workload. Improved threat intelligence: ML can assist IDS in identifying previously unknown threats and vulnerabilities by analyzing threat intelligence feeds. As a result, ML has made IDS more

effective and efficient in detecting and responding to cyber threats, enhancing the overall security of networks and systems.

Machine learning models can be classified as Supervised or Unsupervised. In this praxis, we will be using the supervised models only. Supervised learning entails defining specific targets from a given set of input data. It involves using ML regression and classification techniques. (Sarker et al., 2019)

Classification techniques such as Decision Tree (Quinlan et al., 1993), K-nearest neighbors (Aha et al., 1991), Navies Bayes (Langley,1995), adaptive boosting (Freund et al., 1996), logistic regression (Cessie, 1992) and support vector machines (Keerthi et al., 2001) are well-known and effective for this purpose. Recently, XGBoost and CatBoost have been proposed as new classification techniques that can effectively build data-driven predictive models. (Leevy et al., 2020).

Apart from classification, regression techniques aid in predicting continuous or numeric values, like the number of phishing attacks within a specific period. Regression analyses can be utilized to identify the underlying factors behind some types of crimes like cyber fraud. Popular regression techniques encompass linear regression (Han et al., 1999).

Ensemble learning combines various simple models, generating multiple decision trees to address specific security tasks. (Breiman, 2001).

In a study by Al-Juboori et al. (2023), the authors utilized multiple ML algorithms such as decision tree (DT) and boosting algorithms. Performance assessment was conducted using various classification metrics, including accuracy and f1-score. Across

all tested datasets, the research revealed that all algorithms achieved a detection performance of over 99% for the MiTM attack, as indicated by all metrics.

Similarly, Agarwal et al. (2021) conducted a study to detect suspicious network activities accurately and efficiently. They evaluated three classification ML algorithms: K-nearest neighbor (KNN), Support Vector Machine (SVM) and Naïve Bayes (NB).

The primary objective was to identify the best algorithm to classify malicious traffic effectively while minimizing processing time. Classification performance reports generated using Recall, Precision and F1-score were employed. However, it is important to note that the study did not utilize any performance metric that measures how fast the models can arrive at a specific classification result.

In order to address the above issue, this research will prioritize the "time to predict per record" or the inference time as a crucial factor when selecting a suitable machine learning classifier. The selection criteria will not solely rely on the accuracy, as other factors such as efficiency and classification speed will also be considered.

In Deo's (2015) work, various emerging aspects are explored. One such aspect is the discovery of new features or predictors. Another key finding is that simple machine learning algorithms may be just as effective as complex algorithms. While machine learning algorithms excel at classifying large, multidimensional datasets, domain-specific human expertise remains valuable in finding aspects that may not be discovered by the classification algorithms.

Lastly, due to the varying strengths and weaknesses of different classifier algorithms, an ensemble approach involving a combination of algorithms may produce

superior results compared to using a single algorithm (Deo, 2015; Margolin et al., 2013). Table 2-6 below summarizes many researched datasets.

From the above table the author concluded that several IoT and other attack datasets were not used to create the models. Additionally, many studies did not consider hyperparameter tuning, and there was no comparison between shallow machine learning and deep learning models. Moreover, complex deep-learning models received less attention, and training and validation were not widely observed. This praxis addresses all the above-mentioned concerns.

Finally, as per Ferrag et al. (2022), the datasets mentioned in Table 2-6 are not suitable for Industrial IoT research, and some were not simulated in the real world. Ferrag et al. (2022) summarized different datasets which can be used in industrial IoT attack research.

On top, the authors introduced their own dataset as well based on realistic traffic as per Table 2-7 below.

Table 2-6. Summary of datasets used earlier in IoT IDS research (Islam et al., 2021)

Method	Dataset	Contribution	Limitation
Bi-LSTM	UNSW-NB15	IDS classifier detected normal or attack types with 95% accuracy.	It failed to identify various types of attack. Besides, parameters were not optimized.
CNN	System Call Graph	It used CNN based model to detect Botnet using system call graphs with an accuracy of 97% and an F-measure of 98.33%.	No experiment was done for other malicious lines on IoT devices.

Method	Dataset	Contribution	Limitation
RNN	NSL-KDD	Fog computing-based IDS having multi-layered deep RNN with higher detection rate (DoS: 98.27%, Probe: 97.35%, U2R: 64.93%, R2L: 77.25%).	It explored a single dataset only without explaining the hyper-parameters tuning.
BGRU+MLP, GRU+MLP, BLSTM+MLP, LSTM+MLP, GRU, LSTM, MLP	KDD99, NSL-KDD	Four types of attacks such as DOS, Probe, U2R and R2L had been detected using multiple models with high accuracy where BGRU+MPL performs well achieving 99.24% accuracy.	More generic attacks were detected exploring a single dataset only.
LSTM, GRU Bi-LSTM, BLS	NSL-KDD	Three types of RNN and BLS models were applied to detect intrusions where BLS outperforms with 84.14% accuracy and 84.68% F-measures.	A single simple network dataset was considered only. No hyper-parameter tuning was done.
DF, DJ, DNN, DBN, LSTM, GRU	NSL-KDD, KDD99, CICIDS	An empirical extensive study on NIDS as multiclass classification using four DL models and two shallow ML models with multiple evaluation matrices where DBN outperforms with an accuracy of 96.9%.	No IoT attack datasets are studied.
J48, SVM, NB, NB Tree, MLP, RF, RF Tree, ANN and RNN-IDS	NSL-KDD	NIDS using the feed-forward nature of Random Neural Networks (RNN-IDS) was proposed and compared with multiple ML algorithms where RNN-IDS accuracy reached up to 95.2%.	Only ML models and a single dataset are considered for performance comparison. No hyper-parameter tuning.
RF, CNN, Bi-LSTM, CNN-BiLSTM, AlexNet, LeNet-5	NSL-KDD, UNSW-15	Hybrid sampling and a deep hierarchical network constructed on CNN and BiLSTM were proposed with an accuracy of 83.58%.	No performance comparison with related studies. Multiple models were tested with two similar types of datasets.
Bi-RNN, RNN, GRNN	10% KDD	IDS was designed and evaluated using the Bi-RNN model which outperformed RNN and GRNN with a 99.04% detection rate.	Only 10% of the full KDD dataset was used, failed to analyze the performance using a large amount of data, and to classify multi-class attack data.

Method	Dataset	Contribution	Limitation
LR, SVM, DT, RF, ANN	DS2OS	Data-analysis based IDS model was explored where RF model achieved higher accuracy (99.4%) than other ML models, which could detect multi-class attacks more accurately.	With a single dataset, only ML models were implemented. The efficiency of the RF model for a large volume of data was not examined.
TCN, LSTM, SVM	DS2OS	A semi-supervised HS-TCN model outperformed the other two models with 98.15% accuracy. Besides, a balanced version of the dataset has also been evaluated along with the original one.	Lack of efficiency optimization of semi-supervised models and failed to identify multi-class attack types.
LR, ANN	DS2OS	The dataset was experimented with a data-analysis based technique in a two-fold way, using LR and ANN models. In both cases, LR and ANN detected attacks with equal accuracy (99.4%, 99.99%), respectively.	Indistinct analysis of achieved result and no parameter optimization.
RaNN	DS2OS	Achieved 99.20% attack detection accuracy with less prediction time using an advanced and lightweight scheme of ANN, the RaNN model.	Only a single dataset was used. No statistical comparison of attack data for experimented SVM and DT models.
DNN	DS2OS	Seven types of attacks were classified with 98.26% accuracy using an optimized DNN model.	Complex DL models had not analyzed.

.

Table 2-7. Available datasets for IoT security research (Ferrag et al., 2022)

Dataset*	Year	Description	Features	ML Techniques	Testbed	IoT/IIoT Devices	Threats	Learning Approach		Traffic	
								Centralized	FL	IoT	IIoT
N-BaIoT [9]	2018	Built using 9 IoT devices for legitimate traffic and two botnets (BASHLITE and Mirai) for 10 attack types.	23	LOF, On Class SVM, and IF.	2 Layers	9 Types	10 Attacks	✓	✗	✓	✗
Bot-IoT [10]	2019	Built from IoT legitimate traffic as well as malicious traffic generated by botnets on IoT-specific networks.	46	RNN, SVM, and LSTM.	VM	Simulated	8 Attacks	✓	✗	✓	✗
MQTTset [11]	2020	Built from MQTT protocol traffic and a variety of attack streams associated with IoT devices that leverage it.	33	NN, DT, RF, NB, MP, and GB.	2 Layers	8 Types	5 Attacks	✓	✗	✓	✗
Federated TON_IoT [3]	2020	Consists of three different data types, namely: IoT service telemetry, OSs logs, and network traffic.	31	N/A	3 Layers	Simulated	9 Attacks	✗	✗	✓	✗
X-IIoTID [12]	2021	Consists of device agnostic data used in the context of ML/DL based IDS for both IoT and IIoT systems.	59	DT, NB, SVM, KNN, LR, DNN, and GRU.	3 Layers	N/A	18 Attacks	✓	✗	✓	✓
WUSTL-IIOT-2021 [2]	2021	Created using legitimate and malicious data generated by various IIoT and industrial devices to mimic an actual industrial application.	41	LR, KNN, SVM, NB, RF, DT, and ANN.	4 Layers	5 Types	4 Attacks	✓	✗	✓	✓
Ours	/	Cyber security dataset of IoT and IIoT applications, basedon realistic testbed, for evaluating machine learning-based intrusion detection systems	61	DT, RF, SVM, KNN, and DNN	7 Layers	+10 Types	14 Attacks	✓	✓	✓	✓

*There are other cyber security datasets used by security researchers for evaluating artificial intelligence-based security systems in different IoT and IIoT applications, such as ISCX, DARPA-2009, CICDS2017, UNSW-NB15, KDD99, and NSL-KDD [13]. However, these datasets are not simulated in a real-world IoT/IIoT environment.

2.5 Summary

The research community has made progress in developing ML models for detecting intrusions in IoT networks. However, much of the literature has focused on a limited set of algorithms and datasets that do not adequately represent various IoT domains. Additionally, hyperparameter tuning and comparisons between shallow and deep machine learning models across various IoT domains have been overlooked. Furthermore, the research community has often focused solely on model performance metrics like accuracy without considering the broader applicability and usage of the model.

The literature review revealed that several authors acknowledged the insufficiency of certain datasets and recommended the creation of new ones, particularly for emerging IoT domains such as the industrial IoT attack sector.

In addition to the limitations of the dataset discussed in Table 2-6, there are several other gaps that were identified.

Firstly, there is a lack of a complete framework on how to implement the proposed models in real-life scenarios.

Secondly, while most research has focused on IP network attacks, modern attacks are increasingly targeting the application layer.

Thirdly, the NIST Zero-Trust concept has not been adequately utilized in the design of ML-based IDS.

Fourthly, there is a lack of 5G intrusion detection solutions for IoT networks, and fifthly, there is a dearth of real testbeds for 5G research.

Finally, little research has been conducted on the performance of various ML models on multi-domain IoT datasets, and the impact of inside threats against the edge layer or the model itself has not been thoroughly studied.

This study aims to enhance the existing knowledge on the design of effective intrusion detection algorithms by proposing a novel approach to multi-domain IoT intrusion detection. This approach involves utilizing multi-domain IoT datasets, including one from a real 5G network that simulates attacks on the edge layer. Others include attacks against the IoT application layer. Additionally, a smart zero-trust module is proposed. The proposed module can be integrated with the ML models to utilize their benefits.

Apart from traditional performance metrics, prediction time per record will also be tested. Such a metric will show which models are suitable for power-constrained environments, such as a 5G IoT network connected to an Edge layer node.

The proposed application of such an experiment is proposed to realize the concept of a dynamic intrusion detection system to be placed at the 5G Edge layer, allowing for an intelligent filtering of traffic based on unique patterns instead of relying on current static policies. Thus, protecting 5G core networks from malicious traffic beforehand and applying the intelligent zero trust concept discussed earlier in the literature.

Chapter 3—Methodology

3.1 Introduction

This chapter includes seven primary parts, all of which aim to discuss the methodology followed in this research to build the binary classification intrusion detection system and test its performance for the sole application of placing it inside an edge resource-constrained processing environment of a 5G network achieving the zero-trust concept.

Section 3.1 provides an overview of this chapter and its contents. Section 3.2 explains the proposed architecture framework to develop a network intrusion detection based on the zero-trust concept, utilizing ML algorithms in 5G deployments. Section 3.3 discusses the datasets collected and their description. Section 3.4 discusses the data pre-processing steps. Section 3.5 explores the features selection and importance methods used in this research. Section 3.6 of this chapter summarizes the seven machine learning models examined in this research. These models include Decision Tree (DT), Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Recurrent Neural Network (RNN), Gaussian Naive Bayes (GNB), Bagging Algorithm (BA), and Support Vector Classification (SVC). Those models were developed using the Python programming language.

Section 3.7 covers the ML model development steps. Finally, Section 3.8 discusses the seven supervised ML models' evaluation and performance testing. It also discusses the approach for selecting the best model suitable for 5G-connected IoT devices, satisfying this praxis's goal of suitability to be placed in a resource-constrained environment.

3.2 Proposed architectural framework

To implement a Zero-Trust strategy for a given system, various layers must be considered. These layers are depicted in Figure 3-1. This research is focusing on the Zero-Trust analytics layer.

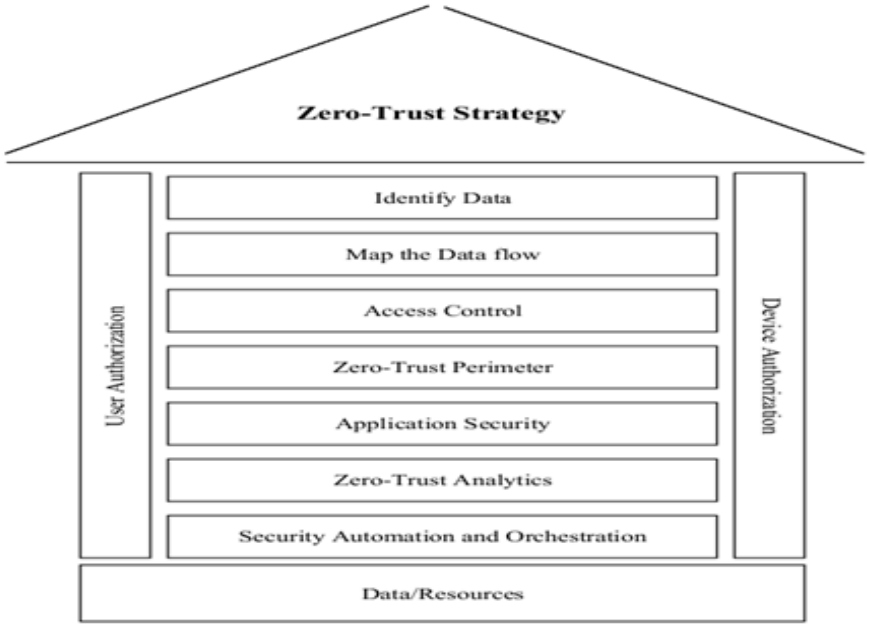


Figure 3-1. Zero Trust Strategy layers (Mehraj et al., 2020)

NIST (NIST SP 800-207, 2020) defines ZTA as the principle of "never trust, always verify." Applying this concept to IoT networks. This means that IoT devices must undergo authorization, authentication, and continuous validation of their security posture before accessing applications and data.

This architecture eliminates the differentiation between internal and external networks, eliminating the idea of implicit trust for networked IoT devices.

In this research, we refer to the subject or source shown in Figure 3-2 as the IoT device. The target resource refers to the internet or an application in a private network that provides IoT with a specific service, while the 5G network serves as the support infrastructure.

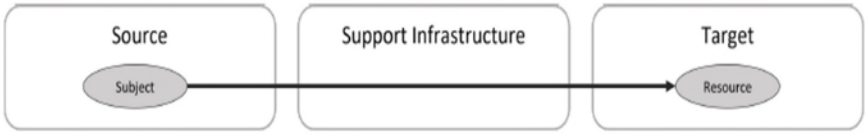


Figure 3-2. Zero Trust Access (Federici et al., 2022)

The functionalities of PDP, PE, PA, and PEP have been explained in Figure 2-7 and Figure 2-8 in chapter 2. In this research, a NIDS Smart Zero-Trust Module is proposed to be placed at the Edge layer. The module layers are mapped to the NIST Zero-Trust framework NIST SP 800-207 (2020), as illustrated in Figure 3-3 below.

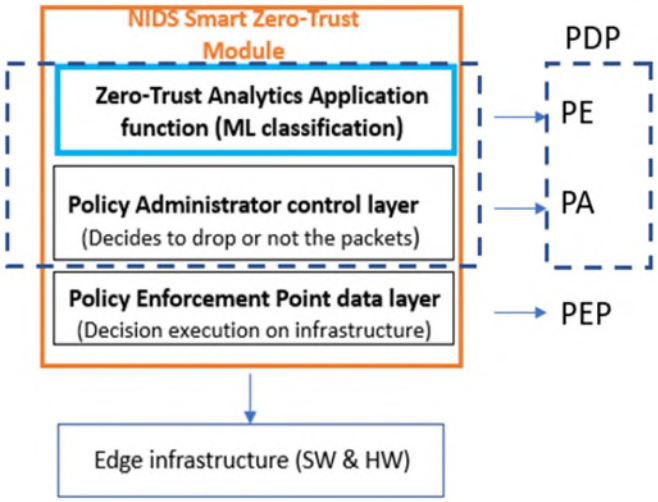


Figure 3-3. Proposed NIDS Smart Zero Trust Module

The above module shown in Figure 3-3 is proposed to be deployed in two main 5G network deployment scenarios. The first scenario assumes that the end-user has its own 5G private network, and hence the edge data center will process only a single IoT industry type of traffic.

Private 5G deployments refer to the deployment of 5G network infrastructure owned, operated, and used by a single organization or entity. This contrasts with public 5G networks deployed by telecommunications carriers and available for public use.

Private 5G deployments offer diverse applications, such as enhancing connectivity and automation in manufacturing, enabling remote operations in industries like mining, oil, and gas, and providing high-speed connectivity for large events. Their key advantage lies in customization to meet specific organizational requirements, providing heightened network security and control compared to public 5G networks. Figures 3-4 and 3-5 show the proposed smart ZT module in both scenarios.

The role of The Policy Enforcement Point data layer (PE), as explained in NIST SP, 800-207, is realized by the Zero-Trust Analytics function. This is achieved primarily through implementing machine learning-supervised classification methods to determine whether incoming traffic is malicious or benign. The Policy Administrator (PA) control layer then uses this classification to decide whether to drop the packet. The PA layer is closely linked to the PE and relies on its decision to determine whether to allow or deny a session. The Policy Enforcement Point data layer (PEP) executes the decisions the layers above make. Its main function is to manage, monitor, and terminate connections between IoT devices and requested resources if the traffic is classified as malicious.

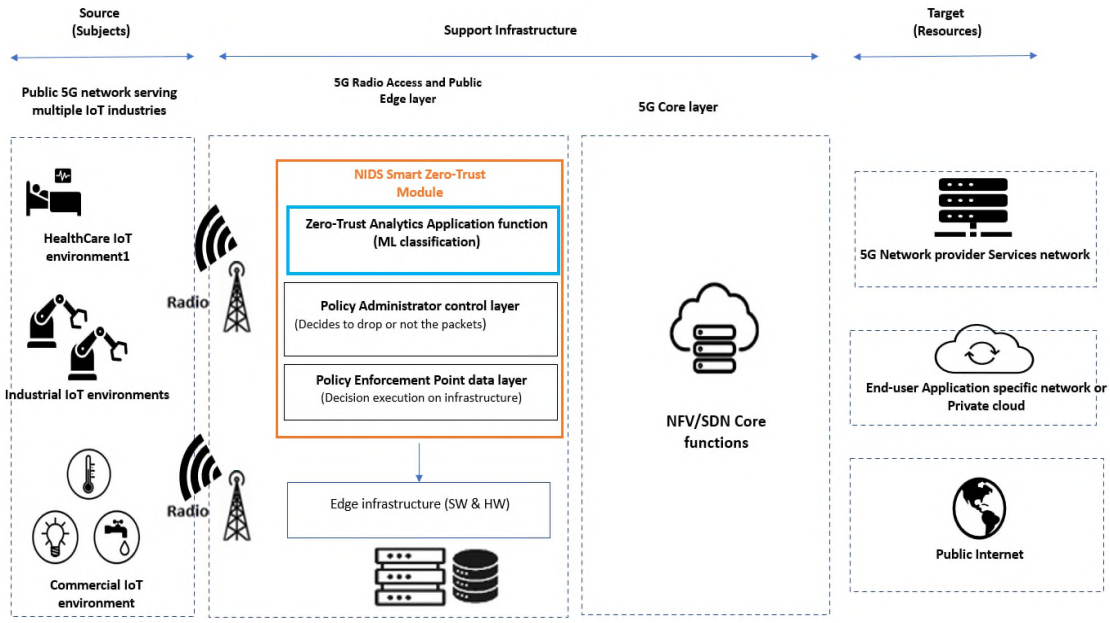


Figure 3-4. Proposed NIDS Smart ZT Module in public 5G IoT deployments

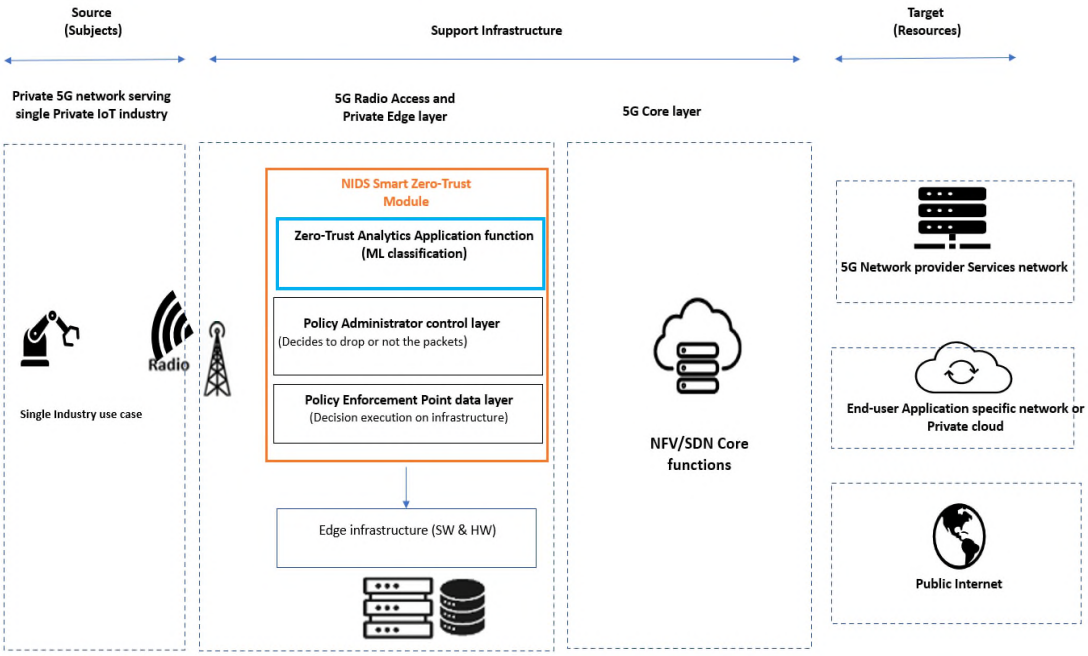


Figure 3-5. Proposed NIDS Smart ZT Module in private 5G IoT deployments

The PEP function is achieved by controlling the software and hardware of the Edge infrastructure. It is worthy to note that without the smart Zero-Trust module the classification function alone will not be able to drop the malicious traffic and it will need human intervention for that to happen.

A strategic methodology involving several elementary steps is followed to detect attacks on an IoT network. Figure 3-6 provides a visual representation of this methodology, which begins with collecting a dataset that is thoroughly observed and analyzed for its features. Then preprocessing the dataset is necessary to ensure it is suitable for learning algorithms. The feature engineering step applies feature selection and dimensionality reduction techniques. The processed dataset is then divided into

training and testing sets with a ratio of 70:30, respectively.

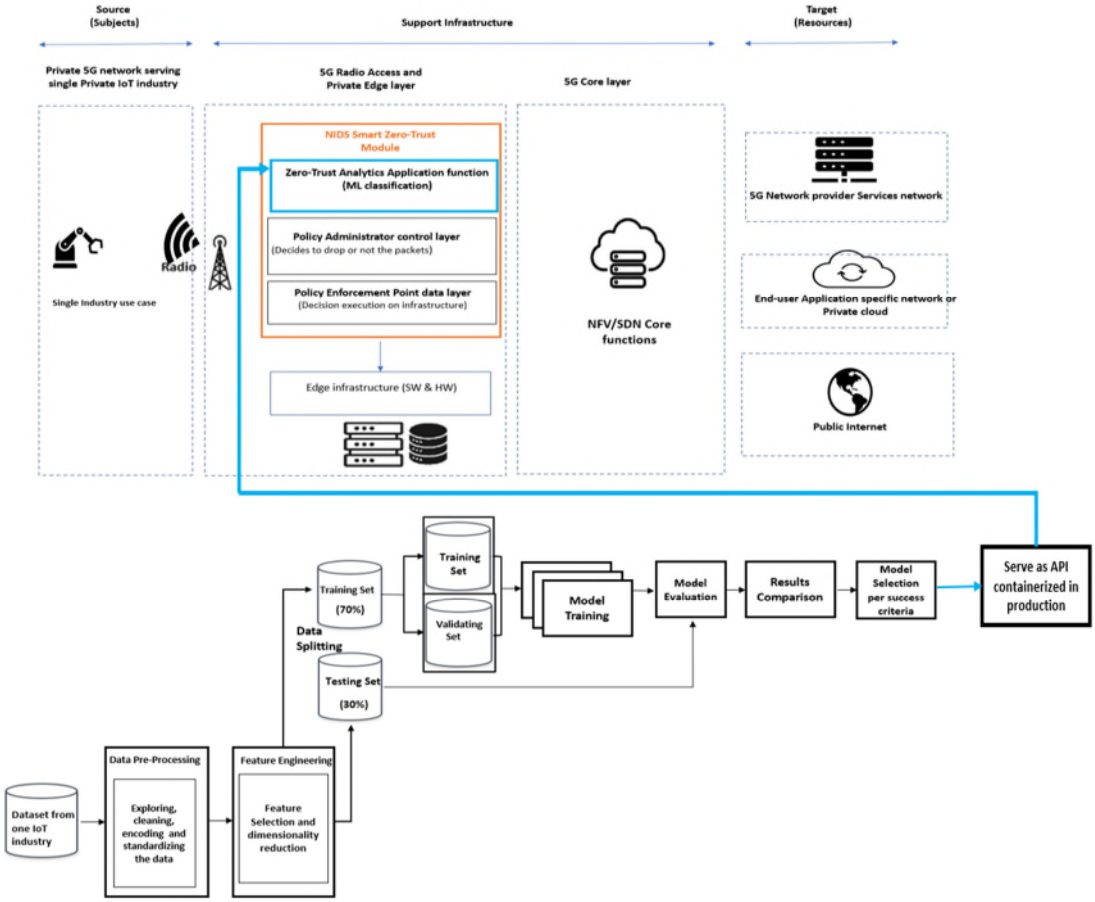


Figure 3-6. Proposed framework including the methodology to develop an ML model for private 5G

The training set serves the purpose of training the ML algorithms, while the validating set acts as a subset of the data employed to evaluate the performance of a trained model during the cross-validation process.

In cross-validation, the data is divided into multiple subsets, with one used for testing and the rest for training. The validating set plays a crucial role in evaluating the model's performance during training and fine-tuning the hyperparameters of a specific model.

By incorporating a validation set, researchers can prevent overfitting and ensure the model can generalize effectively to new data. Once the final models are ready, they are evaluated using the test set based on the chosen performance metrics. More details about these steps will be discussed in subsequent sections of this chapter.

The methodology outlined in Figure 3-6 is better suited for private 5G deployments with a single end-user from a specific IoT industry type. In this case, the feature sets are relatively stable and don't change significantly over time. However, for public 5G deployments, there may be various IoT devices under the same radio access cell connected to the same edge layer, which means that any type of IoT traffic could be classified. Therefore, a different methodology is required, and each IoT industry domain must have its own feature set. To develop a machine learning model suitable for different IoT traffic types, we need to add additional training datasets to represent each IoT industry type as illustrated in Figure 3-7. Additionally, we need to perform feature fusion before training the final model. More details will be shared in subsequent sections about this step.

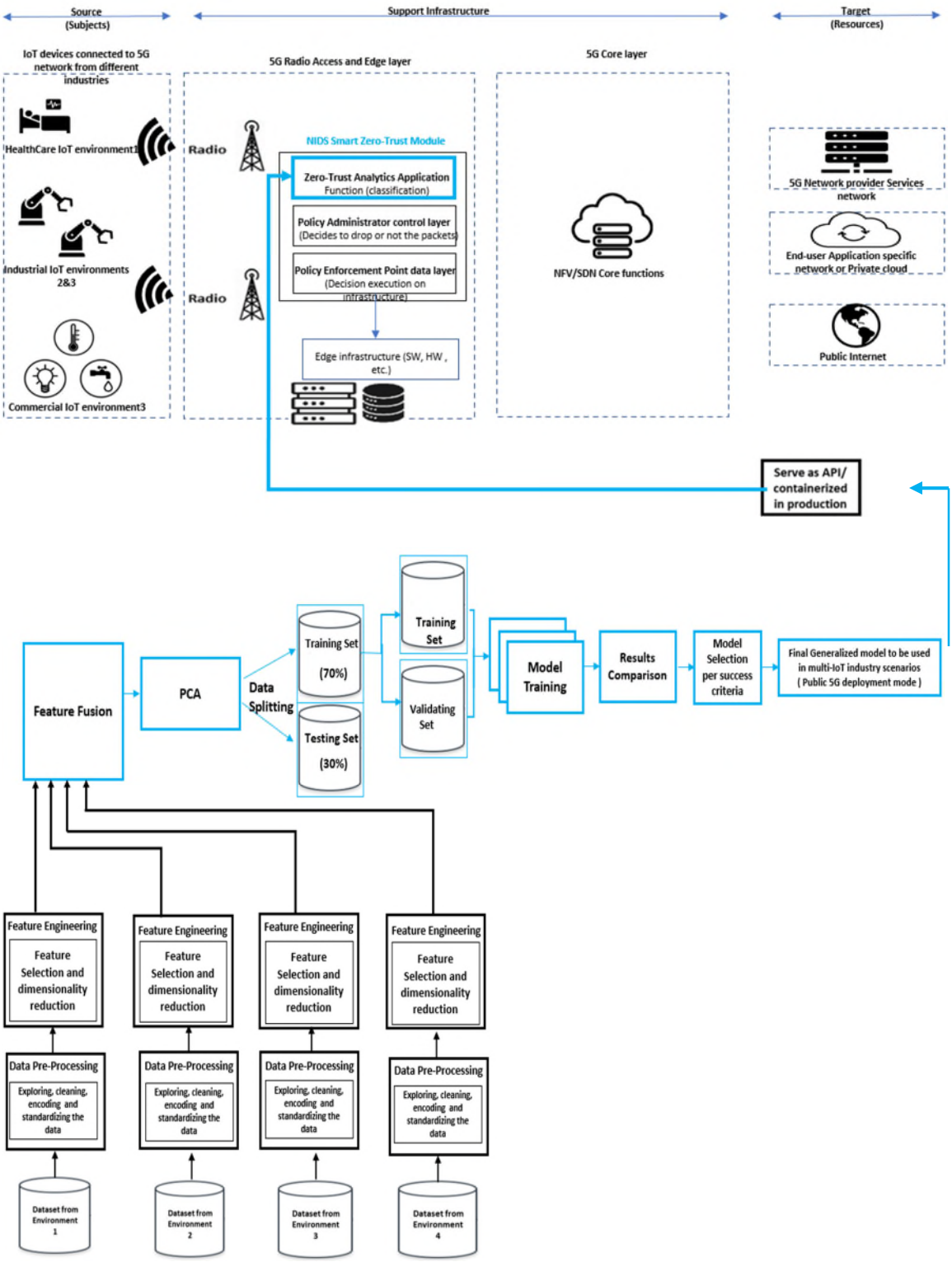


Figure 3-7. Proposed framework including the methodology to develop an ML model for public 5G

To conclude section 3.2, we introduce this architecture for network intrusion detection based on machine learning and the Zero-Trust concept. This architecture is proposed for both private and public 5G deployment scenarios. Figures 3-6 and 3-7 provide visual representations of the proposed architecture.

3.3 Datasets

This research has used four datasets, each representing a different IoT industry. We carefully selected them to ensure their relevance to this research and their representation of different real-world scenarios.

3.3.1 BoT-IoT dataset

The first dataset we used, the Bot-IoT dataset, captures commercial temperature, pressure, and humidity IoT sensor data and offers Botnet attacks against these commercial IoT devices. The BoT-IoT dataset was produced in the Cyber Range Lab of the University of New South Wales (UNSW), a public research university located in Sydney, Australia. In 2019, Koroniotis et al. (2019) constructed a practical IoT network and collected network traffic, including both normal and various forms of malicious attacks, to create this dataset. The BoT-IoT dataset includes four primary types of cyberattacks, with a total of ten subtypes. The initial attack type involves Denial of Service (DoS) with three subcategories based on the type of Internet protocol utilized in the attack: User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Hypertext Transfer Protocol (HTTP). The second attack type is Distributed Denial of Service (DDoS), with three subcategories, based on the type of Internet protocol used in the attack: UDP, TCP, and HTTP. The third attack type is Reconnaissance, with two subcategories, based on the nature of the attack: Operating System (OS), Fingerprinting,

and Service Scan. The last type of attack incorporated into this dataset is Theft. IoT-specific protocol on the application layer (Message Queuing Telemetry Transport - MQTT) is considered in this dataset. (Koroniotis et al., 2019).

The BoT-IoT dataset comprises over 3.5 million network traffic records initially obtained in CSV format. Python and the Pandas library are utilized to import the CSV data files and store each file's data in a data frame object. The data frame objects are merged using Pandas to create a large file comprising all the records.

The BoT-IoT dataset consists of 46 different columns. Each row of the dataset is flagged as either normal or malicious network traffic. This flag is a feature called attack, and it is a binary label, where 0 is used to represent normal traffic, and 1 is used to represent attack traffic. The full list of features and their descriptions can be found in Table A-1 in the Appendix (Koroniotis et al., 2019).

3.3.2 IoT HealthCare dataset

The second dataset we used represents the IoT healthcare industry. In 2019, The IoT HealthCare security dataset was produced by Faisal et al. (2019). The dataset was created from intensive care unit (ICU) related patient and environmental IoT devices. This dataset includes attacks on major application-layer protocols, Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT).

MQTT is a layer over TCP, whereas CoAP works over UDP. This dataset includes four primary types of cyberattacks. MQTT distributed denial-of-service (DOS), MQTT Authentication Bypass Attack, MQTT Packet Crafting Attack, and COAP Replay Attack. The IoT Healthcare dataset comprises over 160,000 network traffic records initially obtained in CSV format. Python and the Pandas library are utilized to import the

CSV data files and store the data in a data frame object. The HealthCare security dataset consists of 50 different columns. Each row of the dataset is flagged as either normal or malicious network traffic. This flag is a feature called attack, and it is a binary label where 0 is used to represent normal traffic, and 1 is used to represent attack traffic. A few examples of the independent variables in this dataset, which are the features that serve as inputs into an ML model, are the length of the TCP header in bytes, the size of the receive window for the TCP connection, the length of the MQTT message payload in bytes, the content of the MQTT message payload and the type of the MQTT message. The full list of features and their descriptions can be found in Table A-2 in the Appendix. (Vaccari et al., 2020)

Figure 3-8 below illustrates the Protocol stack of IoT, and Figure 3-9 shows its interaction within the overall 5G system.

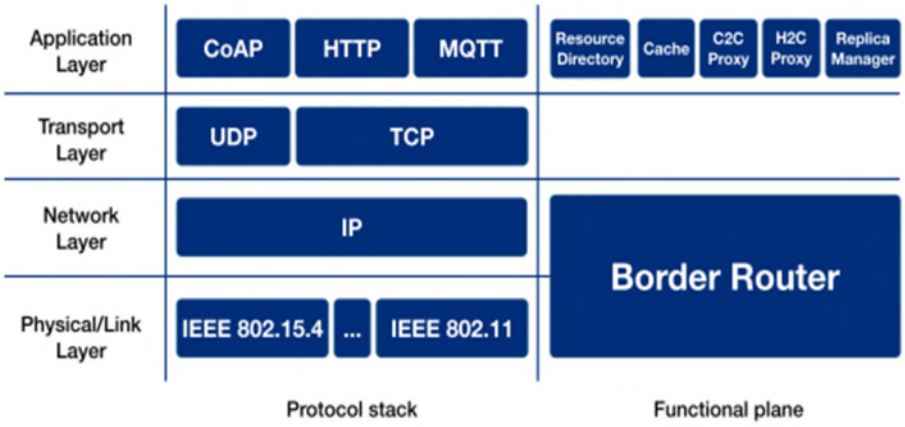


Figure 3-8. Protocol stack in IoT (Cirani et al., 2018)

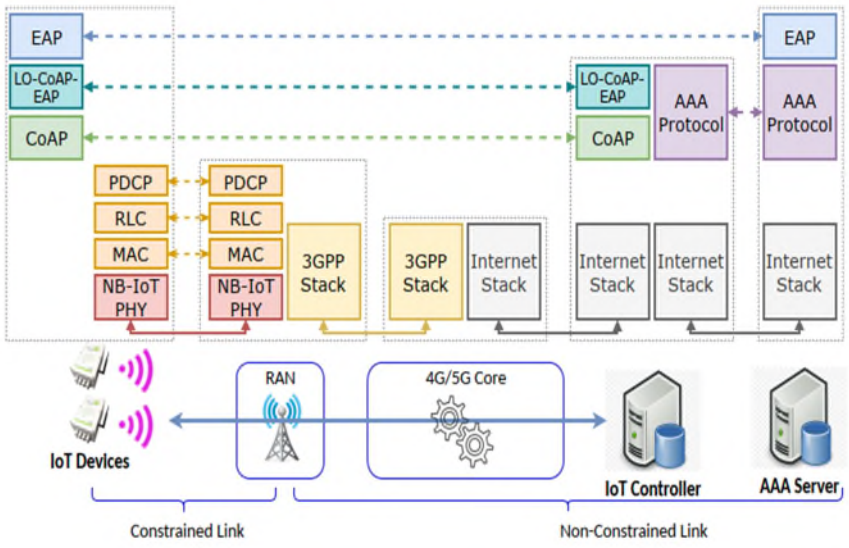


Figure 3-9. CoAP IoT Application protocol on top of 5G lower layers (Sanchez-Gomez et al., 2020)

3.3.3 TON-IOT dataset

This third dataset was created by Nour, M. (2021). It includes heterogeneous data sources collected from telemetry and network data of both commercial IoT and Industrial IoT devices. The dataset was collected from a Cyber Range real network, (SEIT), and (UNSW-ADFA). Attack types include Denial of service (DOS), ransomware, backdoor, injection attack, (MITM) attack, and password cracking attack. It is the only dataset with both IoT service telemetry packet capturing. Modbus protocol for industrial Programmable logic controller (PLC) sensors is also represented. In 1979, Modbus was initially introduced as a data communications protocol to be used with their programmable logic controllers (PLCs). Since then, Modbus has evolved into a widely accepted communication protocol and is now extensively used for linking industrial electronic devices. (Parian et al., 2020)

The TON-IOT dataset comprises over 401,000 network traffic records initially obtained in CSV format. Python and the Pandas library are utilized to import the CSV data files and store the data in a data frame object. The dataset consists of 20 different columns. Each row of the dataset is flagged as either normal or malicious network traffic. This flag is a feature called attack, and it is a binary label where 0 is used to represent normal traffic, and 1 is used to represent attack traffic. A few examples of the independent variables in this dataset, which are the features that serve as inputs into an ML model, are the read coil which is a function code in the Modbus communication protocol that is used to read the current state of one or more coils in a remote device. Coils in Modbus terminology refer to digital outputs that can be either on or off. Examples of coils include relays, LEDs, or any other digital output device. Another example feature is read input register is a function code in the Modbus communication protocol that is used to read the current value of one or more input registers in a remote device. Input registers in Modbus terminology refer to analog inputs that can have a range of values, such as temperature sensors, pressure sensors, or other types of sensors. The full list of features and their descriptions can be found in Table A-3 in the Appendix (Alsaedi et al., 2020).

3.3.4 5G-NIDD dataset

The fourth dataset used in this praxis was introduced in December 2022. The 5G-NIDD dataset was produced in the Centre for Wireless Communications, University of Oulu, Finland, by Sehan et al. (2022)

5G-NIDD is a comprehensive dataset derived from an actual 5G test network. The real network consists of two radio access base stations, each having an attacker node and

several benign 5G users. The malicious node attacks the server deployed in the 5G Mobile Edge Computing Environment (MEC) so acting as an insider threat.

Figure 3-10 below illustrates the network architecture. The 5G-NIDD dataset comprises over 1.2 million network traffic records initially obtained in CSV format. Python and the Pandas library are utilized to import the CSV data files and store the data in a data frame object.

We included this specific dataset in this research because it represents a scenario of an insider threat attack. In this scenario, attackers are authenticated to connect to the 5G radio network but are issuing attacks against parts of the network, specifically the Edge server. Our Zero-Trust module will not consider the attacker's perimeter as an allowed perimeter and will classify the traffic as malicious, dropping the packets eventually.

Another reason for including this dataset is that the attacker targets the edge layer. In the framework proposed in this praxis, the trained model will eventually be placed in the edge layer. Therefore, if DOS attacks against the Edge layer are successful, the edge layer functionality, along with the trained model, will be impacted. Our goal is to ensure that the model can detect such attacks, and the Zero-Trust module will drop the packets, protecting the entire framework. The 5G-NIDD dataset consists of 94 different columns. Each row of the dataset is flagged as either normal or malicious network traffic. This flag is a feature called attack, and it is a binary label where 0 is used to represent normal traffic, and 1 is used to represent attack traffic. A few examples of the independent variables in this dataset, which are the features that serve as inputs into an ML model, are TCP Round-Trip Time, which is the time it takes for a packet of data to travel from a

client to a server, and back again. Source hops refer to the number of routers or network devices a packet must pass through to reach its destination from its source. Each time a packet is forwarded from one network device to another, it is said to have passed through a hop. Mean packet size refers to the average size of the packets that are transmitted in a particular network session or over a certain period. It is often used as a performance metric to measure the efficiency of a network's data transmission. The full list of features and their descriptions can be found in Table A-4 in the Appendix (Samarakoon et al., 2022).

By utilizing these datasets, we aim to develop a suitable ML NIDS model for each IoT industry type to be used in private 5G deployment scenarios. Additionally, we aim to develop a generalized ML model capable of classifying malicious traffic regardless of the IoT industry type, making it more suitable for public 5G deployment scenarios.

The effectiveness of algorithms used for problem-solving is dependent on the quality of the data used to train them. Various methods are applied to datasets to prepare them for further processing. These methods include exploring, cleaning, encoding, normalizing, and reducing data.

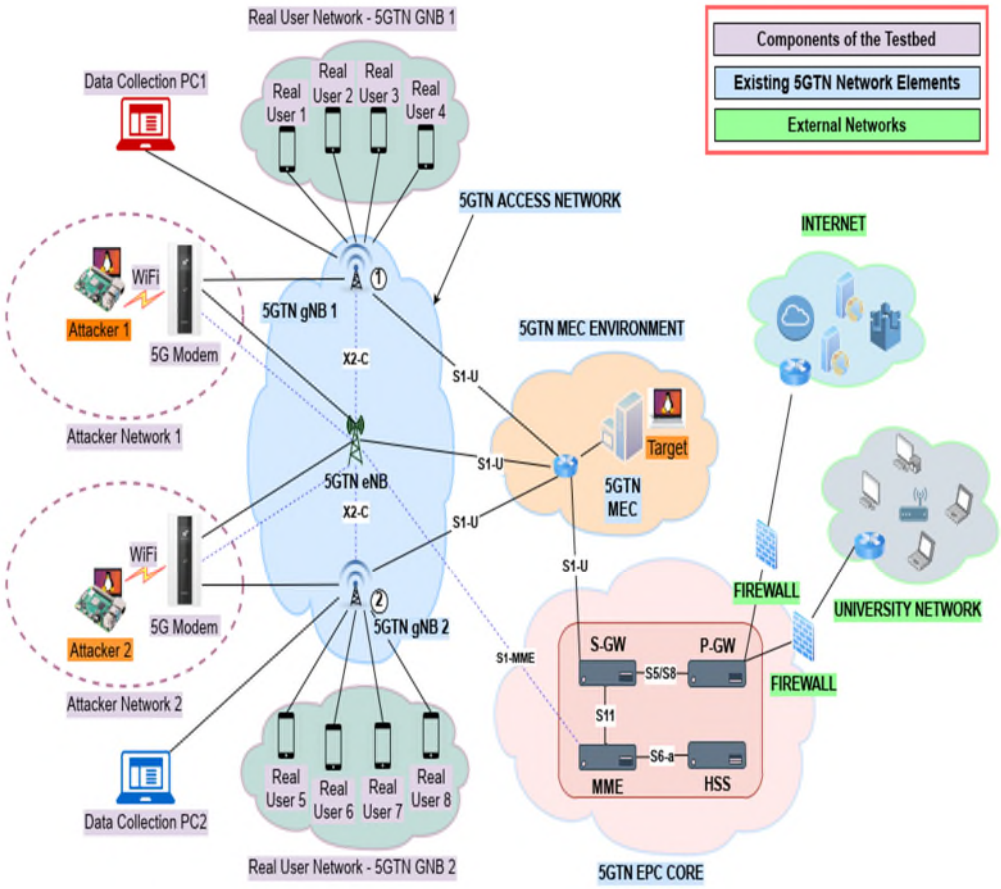


Figure 3-10. Network architecture of the 5G testbed (Sehan et al., 2022)

3.4 Data pre-processing

One of the crucial steps in machine learning (ML) model development is exploratory data analysis (EDA). EDA is a valuable technique that involves visualizing the data to extract meaningful insights from datasets. EDA helps understand the dataset's features, their distribution, scale, and any patterns or relationships between variables that may not be immediately noticeable.

During EDA, various statistical measures are examined. Empty or missing data points are checked, and appropriate actions are taken, such as dropping them. The EDA process also involves visualizing the distribution of unique features in the dataset to

compare their scales and identify any unique features that do not generalize to the rest of the dataset.

To prepare the data for further processing, categorical values, if found, are converted to numeric ones. This can be achieved through two fundamental processes: numericalization and normalization. Label encoding is employed to transform categorical variables such as protocol types (e.g., MQTT or COAP) into numeric values.

By performing EDA and pre-processing, ML practitioners can gain valuable insights into the dataset, which can be used to develop accurate and robust ML models.

Since the purpose of this praxis is to do binary classification and not multi-class modelling. The attack column is the binary flag showing whether a given transaction is malicious or normal traffic; the category has a subcategory type, and hence we are not focusing on researching sub-attacks. This column was always dropped. Similarly, the attack tool will be dropped. For all the datasets below steps have been implemented:

- Initial exploration.
- Identification of the features that are not relevant to the analysis example is the Source IP as in the context of intrusion detection, most attackers now hide behind proxies or virtual private networks, making this feature useless.
- Checking and fixing null Values.
- Counting unique values each column has. This is called high cardinality check. High cardinality columns refer to columns with many distinct values. In the case of numerical columns, high cardinality means that there are many unique values within that column. Columns with high cardinality, which have a large number of unique values, can pose challenges for machine learning models as it increases computational

complexity and could lead to overfitting, as the model may start to learn specific patterns in the training data that are not relevant or generalizable to new, unseen data. Therefore, reducing the cardinality of high cardinality features is often recommended, either by combining similar values into groups or dropping the feature altogether if it does not provide much predictive power.

This praxis uses a high cardinality threshold of 15 and label encoding to fix cardinality columns with a threshold lower than 15. Label encoding converts categorical variables into the numerical format by assigning a unique integer value to each category or label. This can be useful for machine learning algorithms that expect input data to be in numerical format.

Removing redundant features by identifying highly correlated features and removing the one that is least correlated with the target variable. Removing features that are highly correlated with the target variable can improve machine learning model performance in various ways. Firstly, it helps reduce overfitting by preventing the model from becoming too closely tied to the training data and unable to generalize well to new, unseen data. Secondly, it reduces multicollinearity, which can impact the performance of some ML algorithms by making it difficult to determine the true relationship between features and the target variable. Finally, it can improve the interpretability of the model by removing features that are not truly predictive but may instead be proxies for other underlying factors. By reducing redundant features, ML models can achieve better performance, stability, and reliability.

A correlation heatmap is created to uncover any correlations between different features. The Pearson correlation coefficient (PCC) has values in the range from -1 to +1.

PCC is a helpful metric to determine the strength of their relationship.

A higher value indicates a more significant security feature for the model, whereas a lower value indicates that the model's output is not dependent on that feature. The value of 0 implies no correlation between the feature and the model's output.

In summary, the correlation-based feature selection approach utilizing the Pearson correlation coefficient is a useful technique for developing an effective learning-based security model. The model can be optimized to perform accurately and efficiently by identifying and selecting significant features.

After that Normalization techniques are applied. Normalization holds significant importance during the data pre-processing phase in machine learning. Normalization aims to standardize the data and bring it to a common range, making it easier to process and compare the data. Normalization is useful when the features in the dataset are measured in different units or have different scales. By scaling the features to a common range, we can help ensure that all features are treated equally by ML algorithms that are sensitive to the scale of the features. Each column of data is transformed through standardization, which involves adjusting the values so that they have a mean of zero (mean-centered) and a standard deviation of 1. This is achieved by using the following z-score equation to convert each value into a standardized value:

$$zdata = \frac{x - \overline{\mathrm{X}}}{S}$$

Equation 3.1: Zero Score standardization equation

where zdata is the standardized value, x is the raw value in the feature column, $\bar{X}$ the mean of the values in the feature column, and S is the standard deviation of the feature column. Since the values for each feature may be measured on different scales, auto-

scaling is performed to make the interval-scale data behave like ratio-scale data. (Bro & Smilde, 2003).

Auto-scaling normalizes the variance for all features to unit variability to remove the differences in variance that may arise due to different units of measurement. Python provides a function called StandardScaler (); applying this function to the dataset applies the z-score standardization equation to the values of the features in the dataset and scales them so that they are all on the same scale.

3.5 Feature Selection

Feature selection is a crucial preliminary step in preparing data for ML models. Due to the sheer volume of data, processing and storage demands increase substantially. This approach significantly reduces training times and enhances the model's efficiency and effectiveness by focusing on the most relevant information for making accurate predictions.

Feature selection efficiently eliminates irrelevant data, resulting in significant computational time savings. It achieves this by identifying essential features from the original set of features, thus saving time in processing non-useful features. (Cai et al., 2018).

Various techniques, including the filter, wrapper, and embedded methods, are available for feature selection. In this praxis, the filter method is chosen to preserve data generality, considering that the data will be utilized to train diverse ML models. The filter method remains unaffected by the characteristics or parameters of any specific model. (Stańczyk, 2015).

ANOVA reduction refers to using F-tests (F-statistic) to evaluate the differences between the means of the groups formed by each feature with respect to the target variable. The F-statistic measures the ratio of the variance between the groups to the variance within the groups. With an overall target of feature reduction.

ANOVA F-score is selected due to the presence of numerical data and the linearity observed in the four selected datasets. ANOVA compares the variance between groups with the variance within groups (Dissanayake et.,2021). A higher F-score indicates a greater difference between the means of the groups. ANOVA F-score is used as the first step to identify the most important features that distinguish different classes or groups. (Brownlee et al., 2019)

However, in this praxis, large datasets are being examined, so it is challenging to reduce the dimension of the data. So, to enhance the process and select a further smaller subset of features, we used Logistic Regression as a feature importance method. (Huda et al., 2017)

Feature importance pertains to the significance or contribution of individual features in a machine-learning task. The primary objective of feature importance analysis is to identify which features have the most substantial effect on the model's predictions and to determine the most critical features for a specific problem.

In logistic regression, a common technique for assessing feature importance is by utilizing the model's coefficients. Logistic regression models the connection between the predictor variables and the target variable using a logistic function, and the coefficients quantify the shift in the log odds of the target variable for every change in the corresponding predictor variable. The magnitude of these coefficients can serve as a basis

for ranking the features according to their relevance to the prediction task.

This method involves the following steps:

- Training a logistic regression model using a set of predictor variables and a binary target variable.
- Obtaining the estimated coefficients of the logistic regression model.
- Ranking the predictor variables based on the magnitude of their corresponding coefficients. Larger coefficients indicate that the corresponding predictor variable has a stronger influence on the target variable and vice versa.
- Selecting the top-ranked predictor variables as the most important features for the prediction task.

The last step in this section is dimensionality reduction. Principal Component Analysis (PCA) is a statistical method used to reduce the dimensionality of a dataset. It identifies the principal components of the data by projecting the input features onto a new set of orthogonal axes that capture the most important information in the data.

The primary objective of PCA is to decrease the number of features while preserving as much relevant information as feasible. By accomplishing this, algorithms can experience enhanced performance due to reduced computation time. and can also aid in data visualization. PCA can also handle large datasets like the ones processed in this praxis, and it is computationally efficient.

PCA transforms the provided dataset into a small sub-dimensional subspace. The resulting set is referred to as Principal Components (PC). The first component accounts for the most significant variance; subsequent components

cover progressively decreasing amounts of variance.

3.6 Machine Learning Models

In this praxis, seven different ML models were examined, Decision Tree (DT-CART), Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Recurrent Neural Network (RNN), Gaussian Naive Bayes (GNB), Bagging Algorithm (BA), and Support Vector Classification (SVC). Each of these models will be discussed in the following section**.**

3.6.1 Classification And Regression Decision Tree (DT-CART)

The first ML model investigated in this praxis is Decision Tree (DT). DT resembles a tree, composed of branches and leaf nodes representing all possible outcomes of a given scenario. A leaf node is assigned a specific label, while each node represents a test condition based on variable (x) and a split point. CART, the C4.5 flavor used in the python library sklearn, is used in this praxis.

The C4.5 (CART) is capable of classifying samples with continuous or discrete attributes (Munther et al., 2014). The DT algorithm forms the basis for various other essential tree-based methods, such as Random Forest (RF).

This praxis uses CART implementation in python which uses Gini impurity as a split criterion to decide which feature to split at each step of the tree-building process, as recommended in the current literature (Wu et al., 2008).

The Gini Index quantifies the likelihood of a randomly chosen instance being misclassified. As per (Wu et al., 2008) C4.5 rulesets are formed from the original (unpruned) model.

The Gini impurity considers the distribution of class labels among samples and

achieves its maximum value when all classes are equally represented. The value is 0 when there is only one class present and increases to its maximum value when all classes are equally represented. The information gain ratio metric determines tree splits, and the largest gains are chosen as the initial node (Sudrajat et al., 2017).

3.6.2 Random Forest (RF)

Random Forest (RF) is composed of multiple DTs (Resende et al., 2018). RF reduces the variance of each prediction by utilizing a voting technique. This approach of combining multiple DTs to create a stronger model is known as ensemble learning. RF uses Bootstrap Aggregating, also called Bagging, to construct multiple DTs with reduced variance of the constructed trees. (Alam & Vuong, 2013)

Like DT-CART, Gini impurity is also used as the criteria for splitting. Figure 3-11 illustrates the RF trees.

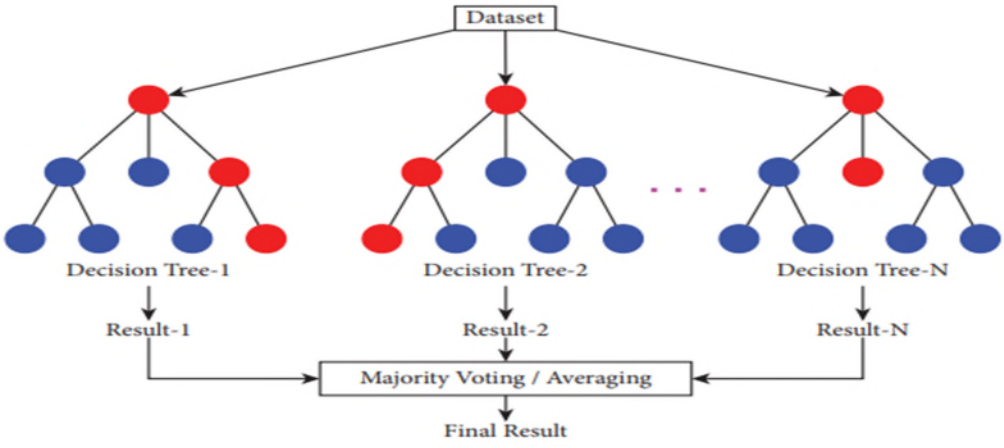


Figure 3-11. RF process (Khan et al., 2021)

3.6.3 eXtreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting is an advanced version of the Gradient Boosting (GB) algorithm designed to be scalable and efficient, making it a popular choice for machine learning practitioners across various industries.

At its core, XGBoost is an ensemble ML model based on decision trees. It generates a sequence of trees, with each subsequent tree aiming to rectify the errors made by its predecessor. This iterative process enables XGBoost to achieve remarkable prediction accuracy, even when dealing with intricate and challenging datasets. (Chen et al., 2016)

XGBoost uses an iterative method to combine multiple "weak" learning models into a "strong" learner. Figure 3-12 explains the XGBoost process. (Han et al., 2019).

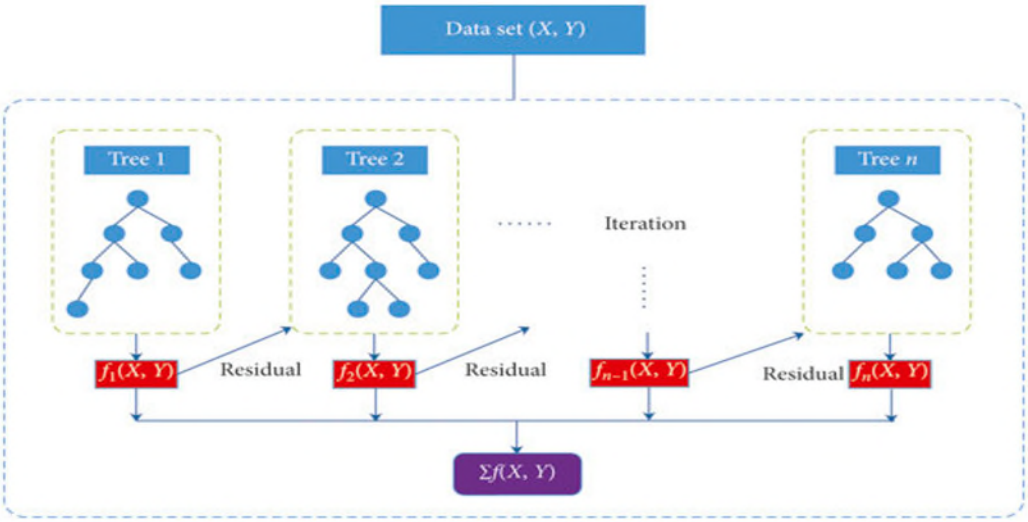


Figure 3-12. XGBoost process (Han et al., 2019)

One of the key advantages of XGBoost is its speed and scalability. It is designed to be highly parallelizable, leveraging multiple CPUs to accelerate model training and

prediction. This makes XGBoost an ideal choice for large-scale datasets where traditional machine-learning algorithms may struggle to perform efficiently. (Alshari et al., 2021)

Another important feature of XGBoost is its ability to handle missing data. XGBoost includes a built-in mechanism for handling missing values, allowing it to continue making accurate predictions even when data is missing. (Aydin & Ozturk ,2021)

XGBoost also includes a wide range of hyperparameters to optimize the model, such as the depth and width of the trees, as well as parameters related to the learning process, such as the learning rate and the regularization strength.

This praxis will use XGBoost for binary classification. as its speed and scalability make it an ideal choice for large datasets in resource-constrained environments like 5G Edge layers.

3.6.4 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are categorized as deep learning models specifically designed to manage sequential data like speech recognition. RNNs can also process information from previous time steps, making them ideal for tasks that require context or temporal dependencies. (Hochreiter & Schmidhuber,1997)

At its core, an RNN consists of a group of recurrent nodes connected cyclically. Each node receives input from the previous time step. This allows the network to maintain an internal state that can carry information across multiple time steps. (Graves, 2013).

RNNs possess a significant capability to learn that extends over long data sequences. Because the network can maintain an internal state across multiple time steps, it can capture dependencies between distant elements in a sequence. This allows RNNs to

make accurate predictions even if time lag exists between input and output. (Chung et al., 2014)

This praxis uses one of the most used libraries for building and training RNNs in Python, which is TensorFlow. RNN illustration can be found in Figure 3-13 below.

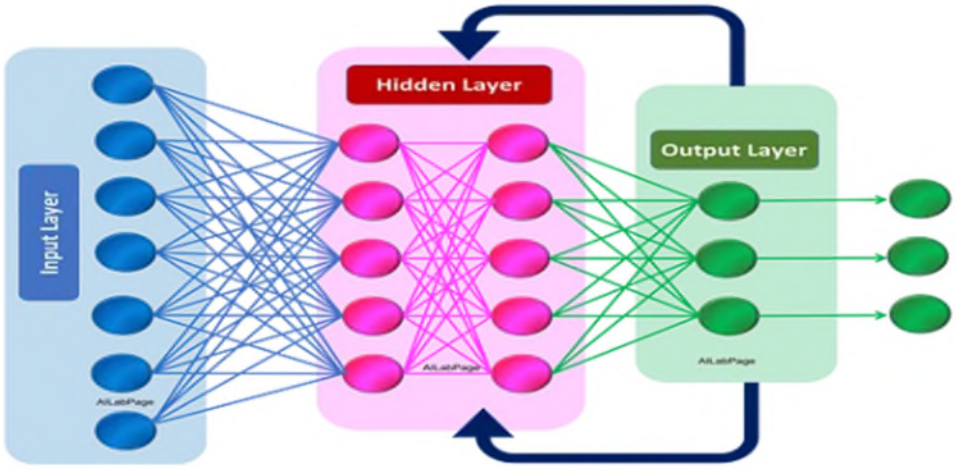


Figure 3-13. Recurrent neural network (Boufeloussen, 2020)

3.6.5 Gaussian Naive Bayes (GNB)

Naïve Bayes (NB) is a widely used tool in various fields, including malware classification, and functions based on the premise of independent features. GNB assumes that the features are normally distributed. GNB uses the mean and variance of each feature in each class to estimate the conditional probabilities needed for classification. (Mukherjee & Sharma, 2012)

In contrast, other variations of NB, such as the Multinomial NB, are designed for discrete data, such as text.

The NB classifier's strength lies in its simplicity, as it generates a table of class probabilities and conditional feature probability estimates during training. Although feature independence may not hold in real-life scenarios, the NB classifier still performs

well and is competitive with other types of classifiers (Al-Aidaroos et al., 2010). The NB classifier is advantageous in that it does not require a large training dataset, trains quickly, requires minimal computational resources, and is robust to missing data. However, the NB classifier may exhibit bias if highly correlated features are present. This ML model is based upon Bayes' theorem. (Jiawei Han, 2011)

3.6.6 Bagging Algorithm (BA)

Bootstrap Aggregating or Bagging involves training multiple base models on different subsets of the training data and then combining their predictions through averaging to reach a final decision. Thus, it is categorized as an ensemble method. Figure 3-14 illustrates the concept.

The main idea behind bagging is to reduce the prediction variance by introducing diversity among the base models. (Breiman, 1996)

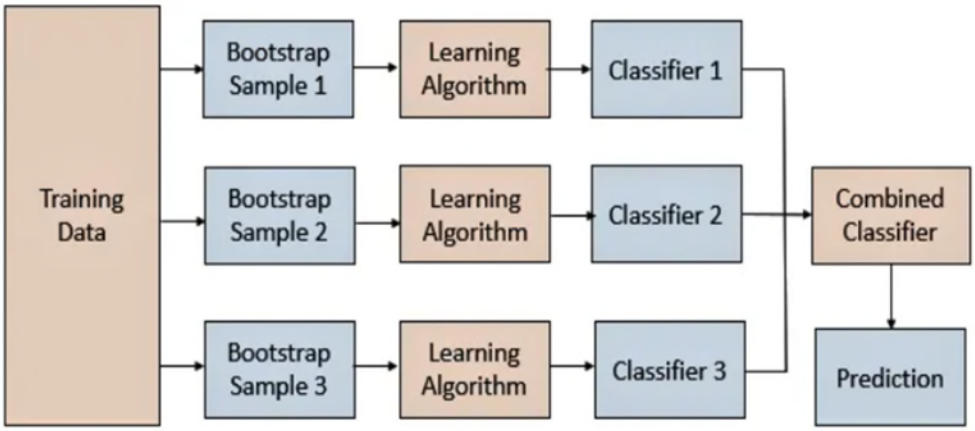


Figure 3-14. Bagging process (CFI, 2022)

Other bagging algorithms include AdaBoost and Gradient Boosting, which use decision trees and other base models.

3.6.7 Support Vector Classification (SVC)

Support Vector Classification (SVC) is a classification variant of Support Vector Machines (SVM) frequently employed for classification tasks. (J et al., 2011)

The primary objective of SVC is to determine the optimal hyperplane that effectively separates the data into distinct classes. SVC should maximize the margin between the classes using mathematical functions called kernels. Kernels can take various forms, such as linear, nonlinear, sigmoid, etc. (Bishop et al., 2006). In this praxis, a linear kernel is used.

Figure 3-15 illustrates the SVM hyperplane. (Papadonikolakis et al., 2012)

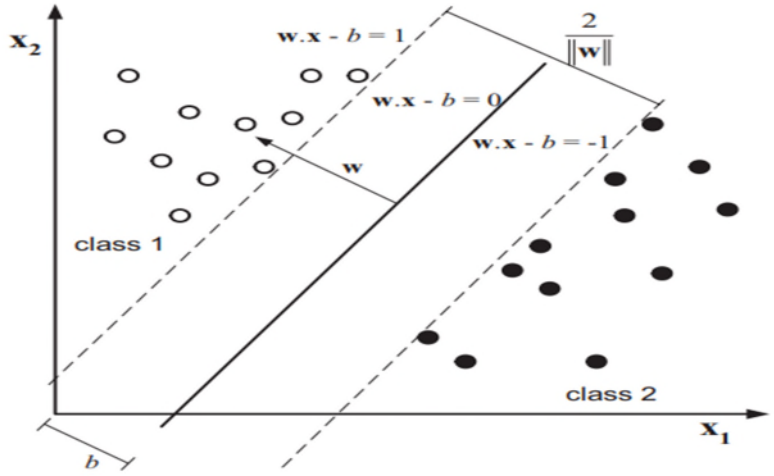


Figure 3-15. SVM separating hyperplane (Papadonikolakis et al., 2012)

While both linear and sigmoid kernels can be used for SVM, they have distinct characteristics and are suitable for different data types. The linear kernel is preferred for linearly separable data, while the sigmoid kernel can handle nonlinear relationships between features. SVC has been popular in many real-world applications, including image recognition, text classification, and bioinformatics.

3.7 ML model development steps

The following section details constructing the seven ML models studied in this praxis. The initial stage involves conducting hyperparameter tuning for each of the models. Once complete, validation and testing are conducted for each model.

3.7.1 Hyperparameter tuning

Hyperparameter tuning is a critical step in machine learning model development to optimize the model's performance. The values of these hyperparameters can significantly impact the accuracy. Hyperparameter tuning process experiments with various combinations of values for the hyperparameters to determine the optimal settings that result in the best performance. (Bergstra & Bengio, 2012)

In this praxis, the GridSearchCV () function in Python's Sklearn library was used for hyperparameter tuning. (Sklearn, n.d)

The grid search algorithm is an exhaustive and computationally intensive method, taking a considerable amount of time to execute. Figure 3-16 below illustrates a grid search space. (Liashchynsky & Liashchynskyi, 2019)

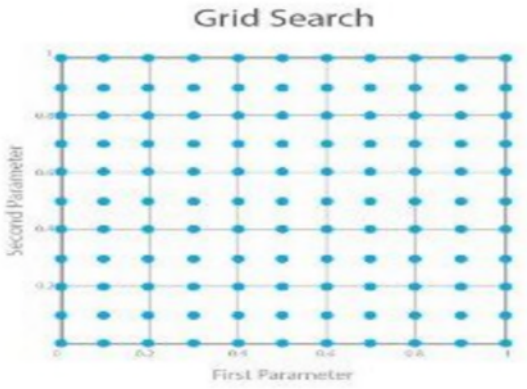


Figure 3-16. Grid search space (Liashchynsky & Liashchynskyi, 2019)

The GridSearchCV () function is a cross-validation method that tests the performance of an ML model using each combination of the values of the

hyperparameters provided to it. The parameter grid is a dictionary that maps each hyperparameter to a list of possible values to test. The function evaluates each combination of hyperparameters using cross-validation. (Sklearn, n.d)

Table 3-1 below lists the various hyperparameters and values tested for each of the seven ML models investigated in this praxis.

Table 3-1. Hyperparameters for tested models

Hyperparameter being tuned	Search space	Explanation
Extreme Gradient Boosting (XGB)		
n_estimators	10,1000,1	Number of boosting rounds or decision trees to be built
max_depth	2,20,1	Search space for max_depth from 2 to 20 with step of 1
learning_rate	0.01 ,1	It controls the step size or shrinkage rate used during each boosting round. The learning_rate parameter determines the contribution of each tree to the final prediction and plays a crucial role in preventing overfitting.
reg_alpha	0,10	Parameter controls L1 regularization on the weights of the model. The purpose of L1 regularization is to encourage the model to learn sparse weight vectors where only a subset of the features has non-zero weights. This can help prevent overfitting by reducing the model's complexity and increasing its generalization performance.
reg_lambda	0,10	It controls L2 regularization on the weights of the model. The purpose of L2 regularization is to encourage the model to learn small weight vectors where all the features have non-zero weights
min_child_weight	1,10,1	It is used to control overfitting. It represents the minimum sum of instance weight needed in a child, or node, for that child to be created during the tree-building process. Increasing min_child_weight can help prevent overfitting by

		requiring more samples to be present in a node for it to be split, leading to a more conservative tree. However, setting min_child_weight too high can result in underfitting as the algorithm may not be able to learn enough from the data."
Random Forest (RF)		
n_estimators	10,1000,1	Search for number of trees in the forest in a space from 10 to 1000 with steps of 1. The goal is to find the value of n_estimators that results in the best performance on a validation set.
min_samples_split	2,20,1	Min number of samples to split a node
min_sample_leaf	1,20,1	Min number of samples required to be at a leaf node. When splitting a node, the algorithm checks if the number of samples at the current node is greater than or equal to min_sample_leaf. If it is less than min_sample_leaf, the node is not split, and it becomes a leaf node.
max_features	sqrt ,log2,none	If set to Sqrt or logs the algorithm will consider a random subset of Sqrt or Log2 of the total number of features at each node. If none then all features will be considered The hyperparameter search will consider three possible values: 'sqrt', 'log2', and None and the goal is to find the value of max_features that results in the best performance on a validation set.
max_depth	2,20,1	Search space for max_depth from 2 to 20 with step of 1
criterion	Gini, entropy	Method of measuring impurity or degree of randomness

RNN		
units	64	Number of neurons in the hidden layer, we used 1 hidden layer for simplicity and computation time in constrained edge environments
activation	relu	Activation function in the hidden layers
solver/optimizer	adam	Optimization algorithm to train the model (adam is adaptive moment estimation)

Decision Tree (DT-CART)		
min_samples_split	2,20,1	Min number of samples to split a node
min_sample_leaf	1,20,1	Min number of samples required to be at a leaf node. When splitting a node, the algorithm checks if the number of samples at the current node is greater than or equal to min_sample_leaf. If it is less than min_sample_leaf, the node is not split, and it becomes a leaf node.
max_features	sqrt, log2, none	If set to Sqrt or logs the algorithm will consider a random subset of Sqrt or Log2 of the total number of features at each node. If none then all features will be considered The hyperparameter search will consider three possible values: 'sqrt', 'log2', and None and the goal is to find the value of max_features that results in the best performance on a validation set.
Ccp_alpha	Cost complexity	controls the complexity of the tree by adding a cost complexity penalty to the impurity reduction criterion. It is used to prevent overfitting by pruning the tree and reducing its size.
Bagging (Decision trees)		
n_estimators	10,1000,1	Search for number of trees in the forest in a space from 10 to 1000 with steps of 1. The goal is to find the value of n_estimators that results in the best performance on a validation set.
max_samples	0.1,1	Hyperparameter controls the maximum proportion of the training data to be used for each bootstrap sample, For example, if max_samples is set to 0.5, then each bootstrap sample will randomly select 50% of the training data with replacement to train the base estimator.
max_features	0.1,1	It Controls the maximum number of features for splitting a node in a decision tree

bootstrap	True, False	TRUE means that bootstrap sampling will be used to create multiple subsets of the data during training. The goal is to reduce the variance of the model and improve its accuracy. FALSE when the dataset is small or when the model is already well-regularized and does not require additional variance reduction. Both values are tested to get the best performing model.
Gaussian NB		
var_smoothing	-10,1	It controls the amount of smoothing applied to the data to prevent zero probabilities. the parameter will be sampled from a logarithmic distribution between 10^-10 and 10^1 during hyperparameter tuning. This range includes very small values close to zero (i.e., a lot of smoothing) and larger values that will have a less smoothing effect on the data.
SVC		
C	0,10	It controls the regularization strength applied to the SVM algorithm. SVMs aim to find the optimal decision hyperplane that separates the different classes of data points with the largest margin. The hyperparameter C controls the tradeoff between achieving a low training error and a low testing error. A high value of C will result in a smaller-margin hyperplane that correctly classifies more of the training data but may be more prone to overfitting. Conversely, a low value of C will result in a larger-margin hyperplane that may generalize better to new data but may also misclassify more of the training data.
kernel	linear, poly, sigmoid, rbf	kernel function used
gamma	scale, auto	It controls the kernel coefficient for the Radial Basis Function "RBF" kernel.

3.7.2 Model training

Model training and testing are essential steps in developing ML models. The main objective is to enable the models to construct a relation between the input data and corresponding labels to make accurate predictions. The goal is to classify new, unseen data by leveraging the patterns and insights gained from the training phase.

Typically, a dataset is split into two parts, the training set and the testing set with 70% and 30% respectively. The training set is used to develop the model, while the testing set is used to assess the model's performance. The next step is to train and run the ML models on the features identified as significant during the feature selection step. In addition to training and testing, cross-validation is applied.

In this praxis,

3-fold cross-validation is employed to assess the model's performance. Cross-validation enables us to estimate the model's performance on unseen data more precisely and ensures that it does not overfit the training data.

3.8 Performance testing and model evaluation

3.8.1 Performance metrics

Various performance metrics have been collected to evaluate the effectiveness of the seven models explored and come up with the most suitable one for each of the four datasets examined. These metrics include accuracy, recall, precision, Area Under the Curve (AUC), F1-score, cross-validation score-train, cross-validation score-test, F1-train, F1-test and inference time.

Accuracy measures the percentage of correctly classified instances out of all classified instances. Precision assesses the quality of classifications and measures the

percentage of true positive classifications out of all records classified as positive. Recall measures the percentage of true positive classifications from all positive records in the dataset. The F1 score is a harmonic mean of the model's precision and recall values and measures the model's classification quality. The AUC-ROC value quantifies the accuracy of each model in distinguishing between different classes effectively. Lastly, inference time in milliseconds is the model's average time to classify one record.

In addition, to ensure robustness and avoid overfitting, three-fold CV process was done during the model development process.

This enables a more reliable model assessment by reducing the likelihood of being biased toward a particular subset of the data.

We created these two additional metrics so we can use the cross-validation score-train to estimate the model's performance during training and the cross-validation score-test to evaluate the final performance of the model on new, unseen data.

Suppose the cross-validation score train is significantly higher than the cross-validation score test. In that case, we may have an overfitting model. This can indicate that the model is too complex and may benefit from simplification or regularization techniques. Comparing both can provide insights into the model's generalization performance.

If the cross-validation score test is significantly higher than the cross-validation score train, we may have an underfitting model. This can indicate that the model is too simple and may benefit from increasing its complexity or adding more features.

Ideally, the cross-validation score train and cross-validation score test should be close in value, which means that the model generalizes well to new, unseen data without overfitting or underfitting the training data.

Similarly, if the F1-Train score is significantly higher than the F1-Test score, we may have an overfitting model. On the other hand, if the F1-Test score is significantly lower than the F1-Train score, we may have an underfitting situation.

Therefore, comparing F1-Test and F1-Train scores can help us determine whether the model needs to be further optimized or modified to improve its performance. Equations used to generate these metrics for each model can be found in Table A-17 in the Appendix.

Brief explanations of the used metrics can be found below:

-True Positive (TP): The model predicted the attack traffic as an attack.

-True Negative (TN): The model predicted the normal traffic as normal.

-False Positive (FP): The model didn't predict the normal traffic as an attack.

-False Negative (FN): The model didn't predict the attack traffic as normal.

-The ROC curve is a plot of the true positive rate (TPR) on the y-axis and false positive rate (FPR) on the x-axis.

-The TPR is the ratio between TP and (TP + FN).

-The FPR is the ratio between (FP) and (FP + TN).

The cross-validation score is calculated as the average of the accuracy metric of the three folds. In this praxis, the cross_val_score () function was used from the scikit-learn python library.

3.8.2 Model selection step

This praxis aims at deploying the chosen model in 5G implementations, which is why we include the inference time per record in milliseconds as a crucial metric for the ultimate model selection. Under the 5G standards outlined in Table 2.1 discussed in Chapter 2, the latency parameter must be less than one millisecond for end-to-end round-trip delay (Arunachalam et al., 2018).

Therefore, our winning model inference duration must not exceed 0.5 milliseconds to avoid any adverse effects on the overall system in either download or upload.

In 5G deployments, we have two deployment modes, as depicted in Figures 3-6 and 3-7. The first is the private mode, where the features are unlikely to change. One classifier per IoT industry can be recommended based on the success criteria of performance metrics and inference duration, as explained in the above section.

The second one is the public mode, which comprises different feature sets due to various IoT industries connecting to the same radio link, and traffic processing happens in the same edge layer. Consequently, combined traffic will be processed by the same proposed smart zero trust module.

For the public 5G deployment scenarios, we develop a model based on the combined features from the four datasets. For that purpose, feature fusion technique will be used. Feature fusion is a technique used to combine multiple types of features, enhancing a model's performance. The idea is to integrate complementary information from different feature sources to enhance the overall representation of the data.

As per (Li et al., 2018), data fusion techniques are employed across three distinct levels concerning the processing stage of fusion. The first level, known as data-level fusion or low-level fusion, combines multiple sources to generate refined data.

At the second level, known as feature-level fusion or intermediate-level fusion, the focus is combining various data features. The primary objective is to extract a limited set of crucial features for analysis. This process is facilitated through the application of feature reduction methods, which serve to reduce computational and memory requirements. Our current study employs this specific type of fusion.

The third level is decision-level fusion which is known as high-level fusion, which encompasses the fusion of decisions derived from multiple detectors. Each detector performs essential detection, and the resulting inferences are then consolidated into the final accurate decision by utilizing decision fusion techniques.

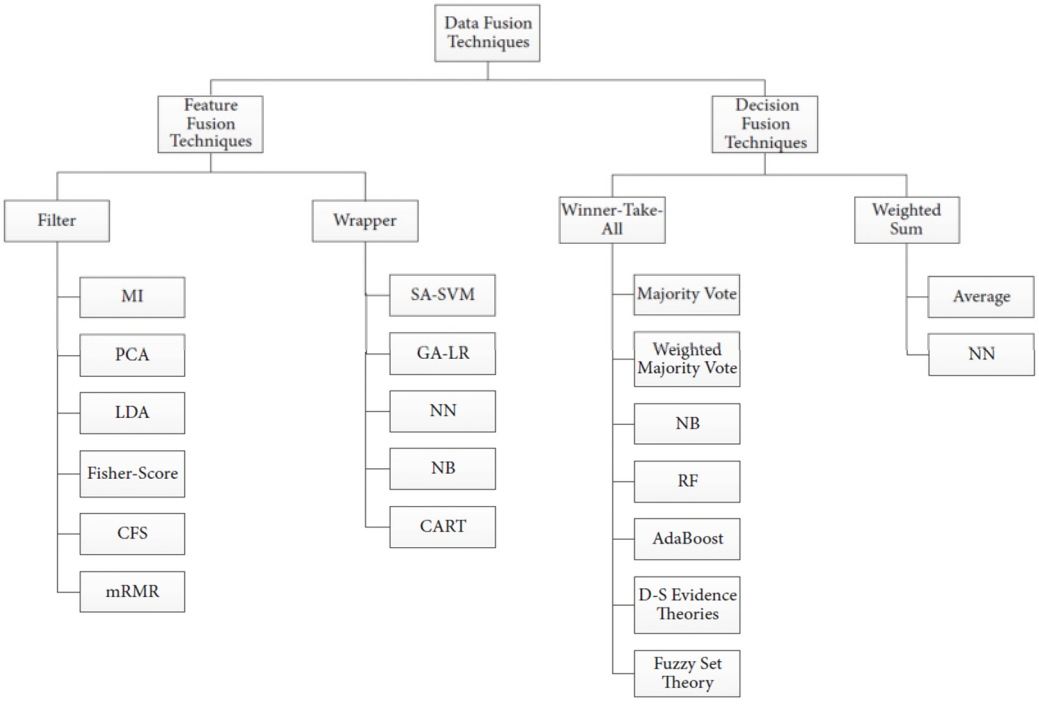


Figure 3-17. Categories of fusion techniques (Li et al., 2018)

In this praxis, PCA method is used for dimensionality reduction, the seven models will be trained and evaluated, as illustrated earlier in Figure 3-7 of this chapter. Finally, the recommended models for deployment will be selected as illustrated in Figure 3-18 below.

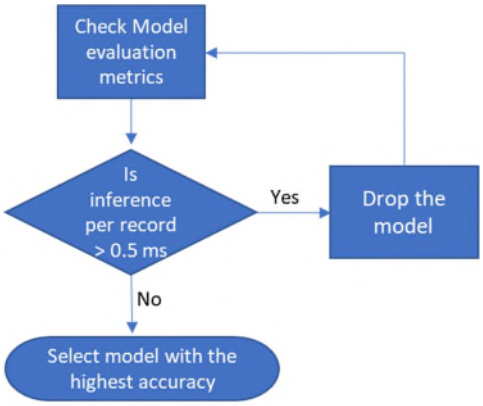


Figure 3-18. Methodology for selecting public 5G deployment models

3.8.3 Model interpretation step

This research uses the Shapley method to interpret the final model classification results. Shapley values are model agnostic and very useful in interpreting a model's individual predictions. To calculate the Shapley value for a feature, Shapley considers all permutations of the features and determines each feature's marginal contribution when it is added to a subset of the features. Shapley's method compares the predictions made with and without the feature. The Shapley value for a given feature is the average marginal contribution across all permutations. (Fryer no., 2021)

For a specific prediction, the Shapley values indicate how much each feature contributed to the final prediction. Positive Shapley values positively impact the classification result, whereas negative values indicate a negative influence.

Moreover, Shapley values effectively handle highly correlated features. They assign highly correlated features a low Shapley value due to a lack of novel contribution to the model. This is because adding a feature highly correlated to other features considered by Shapley values would only provide a small to negligible marginal contribution to the model's performance.

Chapter 4—Results

4.1 Introduction

This research aims to develop a machine learning-based Intrusion Detection System with a smart zero-trust framework to provide security for IoT devices connected to 5G networks via the Edge layer. This system can detect, and binary classify the malicious network traffic while operating in a power-constrained environment like a 5G Edge layer data center. Seven ML models were investigated in this praxis, including Decision Tree (DT-CART), Random Forest (RF), eXtreme Gradient Boosting (XGB), Recurrent Neural Network (RNN), Gaussian Naive Bayes (GNB), Bagging Algorithm (BA), and Support Vector Classification (SVC).

Apart from this introductory section, this chapter includes four primary parts, all of which aim to discuss the results of each methodology step followed in this research for each dataset examined. Section 4.2 discusses the results of the data preprocessing steps, such as EDA, standardization, and feature selection. Section 4.3 will discuss the results of building and testing the performance of the seven ML models examined in this praxis to use them in the private 5G deployment scenarios.

Finally, section 4.4 will discuss the results of building and testing the performance of a suitable model for public 5G deployments utilizing the feature fusion technique. Overall, the chapter provides a comprehensive overview of the various model performance metrics and how they compare between the models.

4.2 Data pre-processing

The BoT-IoT dataset contains over 72 million records of network traffic, which together occupy more than 16 GB of storage space in multiple CSV files in the Microsoft Excel format. To manage the dataset's large size, Koroniotis et al. (2019) extracted a 5% sample, resulting in a subset of more than 3.5 million network traffic records stored in four CSV files totaling 1.07 GB in size. In this praxis we sampled 13% to have a more representable sample.

Increasing the sample size of data used for machine learning classification can improve the accuracy and generalizability of the classification model and reduce sampling bias. By providing more data for the model to learn from, it can identify more patterns and relationships between features, making it more likely to perform well on new, unseen data, thus increasing generalizability. Increasing the sample size also ensures that the classification model is more representative of the studied population, which helps

reduce the risk of sampling bias and mitigate overfitting. Table A-18 in the Appendix illustrates the sample size for all the datasets used in this praxis.

After that, EDA is performed to describe this dataset, as illustrated in Table 4-1 below.

We can see that the BoT-IoT dataset contains 3,668,522 rows and 46 columns. The dataset contains numerical and categorical data: forty-five features and three target variables (*attack, Category*, and *subcategory*). The target variable is the binary feature *attack*. Whereas Category and subcategory are labels that could be used in multi-class classification. Since this praxis focuses on binary classification, we did not perform categorical data encoding for the several attack types.

Upon examining the dataset, we found no null or missing values. Each column in the dataset represents a descriptive attribute of a network transaction. All transactions in the dataset have complete attribute values.

As per Table 4-1, all features are of five different data types: float32, float64, int32, int8, and object. Twenty of the features are of the float32 type, six are of the float64 type, five are of the int32 type, six are of the int8 type, and nine of the features are of the object type.float64 type is a 64-bit floating point number, float32 type is a 32-bit floating point number, int32 type is a 32-bit integer, int8 type is an 8-bit integer, and finally, the object type is a string of alphanumeric characters.

Table 4-1. BoT-IoT dataset description

```
Data columns (total 46 columns):
 #    Column                                 Dtype
---   ------                                 -----
 0    pkseqid                                float32
 1    stime                                  float64
 2    flgs                                   object
 3    flgs_number                            int8
 4    proto                                  object
 5    proto_number                           int8
 6    saddr                                  object
 7    sport                                  object
 8    daddr                                  object
 9    dport                                  object
 10   pkts                                   float32
 11   bytes                                  int32
 12   state                                  object
 13   state_number                           int8
 14   ltime                                  float64
 15   seq                                    float32
 16   dur                                    float32
 17   mean                                   float32
 18   stddev                                 float32
 19   sum                                    float32
 20   min                                    float32
 21   max                                    float32
 22   spkts                                  float32
 23   dpkts                                  float32
 24   sbytes                                 int32
 25   dbytes                                 int32
 26   rate                                   float32
 27   srate                                  float32
 28   drate                                  float32
 29   tnbpsrcip                              int32
 30   tnbpdstip                              int32
 31   tnp_psrcip                             float32
 32   tnp_pdstip                             float32
 33   tnp_perproto                           float32
 34   tnp_per_dport                          float32
 35   ar_p_proto_p_srcip                     float64
 36   ar_p_proto_p_dstip                     float64
 37   n_in_conn_p_dstip                      int8
 38   n_in_conn_p_srcip                      int8
 39   ar_p_proto_p_sport                     float64
 40   ar_p_proto_p_dport                     float64
 41   pkts_p_state_p_protocol_p_destip       float32
 42   pkts_p_state_p_protocol_p_srcip        float32
 43   attack                                 int8
 44   category                               object
 45   subcategory                            object
dtypes: float32(20), float64(6), int32(5), int8(6), object(9)
```

Table 4-2 below illustrates the null values, null percentages, data type, number of unique values per column, and a sample from the values.

Table 4-2. BoT-IoT null values analysis

	feat	nan	nan_p	dtype	unique	sample
0	ar_p_proto_p_dport	0	0	float64	42237	[0.0, 0.00154555, 0.00154939, 0.00155324]
1	ar_p_proto_p_dstip	0	0	float64	39186	[0.0, 0.00191092, 0.00202493, 0.00205182]
2	ar_p_proto_p_sport	0	0	float64	136207	[0.0, 0.00203635, 0.00222217, 0.00249818]
3	ar_p_proto_p_srcip	0	0	float64	46289	[0.0, 0.00203635, 0.00222217, 0.00249818]
4	attack	0	0	int8	2	[0, 1]
5	bytes	0	0	int32	1633	[60, 62, 65, 66]
6	dbytes	0	0	int32	472	[0, 42, 60, 66]
7	dpkts	0	0	float32	62	[0.0, 1.0, 2.0, 3.0]
8	drate	0	0	float32	20714	[0.0, 0.000643, 0.000835, 0.000836]
9	dur	0	0	float32	612509	[0.0, 1e-06, 2e-06, 3e-06]
10	flgs_number	0	0	int8	9	[1, 2, 3, 4]
11	ltime	0	0	float64	383624	[1526344227.07794, 1526344302.81496, 1526344328.93349,
12	max	0	0	float32	594525	[0.0, 1e-06, 2e-06, 3e-06]
13	mean	0	0	float32	507089	[0.0, 1e-06, 2e-06, 3e-06]
14	min	0	0	float32	271147	[0.0, 1e-06, 2e-06, 3e-06]
15	n_in_conn_p_dstip	0	0	int8	100	[1, 2, 3, 4]
16	n_in_conn_p_srcip	0	0	int8	100	[1, 2, 3, 4]
17	pkts	0	0	float32	123	[1.0, 2.0, 3.0, 4.0]
18	pkts_p_state_p_protocol_p_destip	0	0	float32	1595	[1.0, 2.0, 3.0, 4.0]
19	pkts_p_state_p_protocol_p_srcip	0	0	float32	1526	[1.0, 2.0, 3.0, 4.0]
20	rate	0	0	float32	139677	[0.0, 0.000773, 0.000775, 0.000777]
21	sbytes	0	0	int32	1052	[42, 60, 62, 65]
22	seq	0	0	float32	262212	[1.0, 2.0, 3.0, 4.0]
23	spkts	0	0	float32	91	[1.0, 2.0, 3.0, 4.0]
24	srate	0	0	float32	119709	[0.0, 0.000643, 0.000773, 0.000775]
25	state_number	0	0	int8	11	[1, 2, 3, 4]
26	stddev	0	0	float32	421379	[0.0, 1e-06, 2e-06, 3e-06]
27	stime	0	0	float64	392259	[1526344031.66514, 1526344031.66515, 1526344031.66516,
28	sum	0	0	float32	934972	[0.0, 1e-06, 2e-06, 3e-06]
29	tnbpdstip	0	0	int32	7631	[60, 70, 102, 120]
30	tnbpsrcip	0	0	int32	8639	[60, 70, 107, 120]
31	tnp_pdstip	0	0	float32	1587	[1.0, 2.0, 3.0, 4.0]
32	tnp_per_dport	0	0	float32	1582	[1.0, 2.0, 3.0, 4.0]
33	tnp_perproto	0	0	float32	1560	[1.0, 2.0, 3.0, 4.0]
34	tnp_psrcip	0	0	float32	1522	[1.0, 2.0, 3.0, 4.0]

The models used in this praxis can process solely numerical data, which renders all the previously mentioned data types appropriate, except the object type. Object-type features necessitate mapping to numerical values to enable the ML models to assimilate information from them.

Then we checked the statistics for each numerical feature in the dataset as per Table 4-3 below. One of the key observations from the descriptive statistics presented in this table is that the features in this dataset exhibit varying scales, as evident from the differences in means and standard deviations across different features. For instance, a

stark difference in feature values can be observed when comparing the minimum, maximum, mean, and standard deviation and the 25th, 50th, and 75th percentiles values of many features, highlighting the scale disparity between them.

Table 4-3. Statistics for BoT-IoT Dataset

	feature	count	mean	std	min	25%	50%	75%	max
0	ar_p_proto_p_dport	3.66852e+06	538.52	15698.2	0	0.245773	0.394306	0.576971	2e+06
1	ar_p_proto_p_dstip	3.66852e+06	285.183	4096.94	0	0.243668	0.398629	0.579639	1e+06
2	ar_p_proto_p_sport	3.66852e+06	456.495	14329.2	0	0.231481	0.378591	0.572555	3e+06
3	ar_p_proto_p_srcip	3.66852e+06	332.744	8466.03	0	0.235995	0.390089	0.572558	2.71429e+06
4	attack	3.66852e+06	0.99987	0.0114021	0	1	1	1	1
5	bytes	3.66852e+06	869.05	112267	60	420	600	770	7.18333e+07
6	dbytes	3.66852e+06	102.658	49449	0	0	0	0	3.41817e+07
7	dpkts	3.66852e+06	0.411817	49.6493	0	0	0	0	35029
8	drate	3.66852e+06	0.445505	60.2922	0	0	0	0	58823.5
9	dur	3.66852e+06	20.3348	21.4633	0	12.5626	15.5085	27.0999	2771.49
10	flgs_number	3.66852e+06	1.49021	0.865244	1	1	1	2	9
11	ltime	3.66852e+06	1.52805e+09	258151	1.52634e+09	1.52808e+09	1.5281e+09	1.5281e+09	1.52938e+09
12	max	3.66852e+06	3.02001	1.84877	0	0.280607	4.00911	4.29358	5
13	mean	3.66852e+06	2.23106	1.51708	0	0.181967	2.69013	3.5652	4.98188
14	min	3.66852e+06	1.01754	1.47896	0	0	0	2.15114	4.98047
15	n_in_conn_p_dstip	3.66852e+06	92.4517	18.1764	1	100	100	100	100
16	n_in_conn_p_srcip	3.66852e+06	82.5385	24.3974	1	69	100	100	100
17	pkts	3.66852e+06	7.72596	115.578	1	5	7	9	70057
18	pkts_p_state_p_protocol_p_destip	3.66852e+06	642.29	452.992	1	324	600	828	112544
19	pkts_p_state_p_protocol_p_srcip	3.66852e+06	585.999	431.728	1	294	500	800	117939
20	rate	3.66852e+06	244.941	7919.79	0	0.191727	0.30584	0.493037	1e+06
21	sbytes	3.66852e+06	766.392	71929.8	42	420	600	770	3.7747e+07
22	seq	3.66852e+06	121321	75696.9	1	54876	117769	184930	262212
23	spkts	3.66852e+06	7.31414	77.2525	1	5	6	8	35029
24	srate	3.66852e+06	2.95511	724.532	0	0.15597	0.283783	0.488201	1e+06
25	state_number	3.66852e+06	3.13439	1.18697	1	3	4	4	11
26	stddev	3.66852e+06	0.88715	0.801025	0	0.030019	0.793896	1.7453	2.49676
27	stime	3.66852e+06	1.52805e+09	258150	1.52634e+09	1.52808e+09	1.5281e+09	1.5281e+09	1.52938e+09
28	sum	3.66852e+06	7.72163	7.5917	0	0.344598	8.26996	11.7104	1913.19
29	tnbpdstip	3.66852e+06	56845.6	415300	60	38960	54000	75460	2.20391e+08
30	tnbpsrcip	3.66852e+06	51016.2	442813	60	32010	47520	66000	2.20391e+08
31	tnp_pdstip	3.66852e+06	687.203	584.603	1	438	616	836	225521
32	tnp_per_dport	3.66852e+06	736.907	652.29	1	500	700	920	244425
33	tnp_perproto	3.66852e+06	753.565	1435.61	1	502	700	924	228373
34	tnp_psrcip	3.66852e+06	619.706	583.363	1	350	566	800	225519

For that, it is necessary to perform data standardization on the entire dataset. This is important to ensure that the features with higher value scales do not cause skewness in the outcome in their favor, thereby avoiding any disproportionate weightage in the analysis.

After that we inspected the features carefully. We found that certain features, such as (saddr, sport, daddr, flg, state, dport, proto, proto_number), serve as identifiers for network flow and aid in labeling. However, they are being removed as they can uniquely identify a flow. Attack, Category and Subcategory attributes are not useful for this binary

classification task. Additionally, pkseqid, which assigns a Sequence ID for each packet in a flow, is also irrelevant to this analysis. Similarly, the statistics of the remaining datasets can be found in Tables A-5, A-6 and A-7 in the Appendix.

Next step in data-preprocessing was to look at the correlation between the different features of the dataset. We checked for features that are highly correlated with each other based on the Pearson correlation coefficient matrix. This praxis considers a threshold of 90%. Then the least correlated with the target variable was removed. Redundant features removed were: (*pkts, tnp_pdstip, tnp_perproto, tnp_psrcip, srate and stime*). Figure 4-1 below shows the feature correlation with the target variable.

Next, we made a check to see which features are highly correlated with the target variable. In this praxis, a threshold of 60% for high correlation with target variable was set.

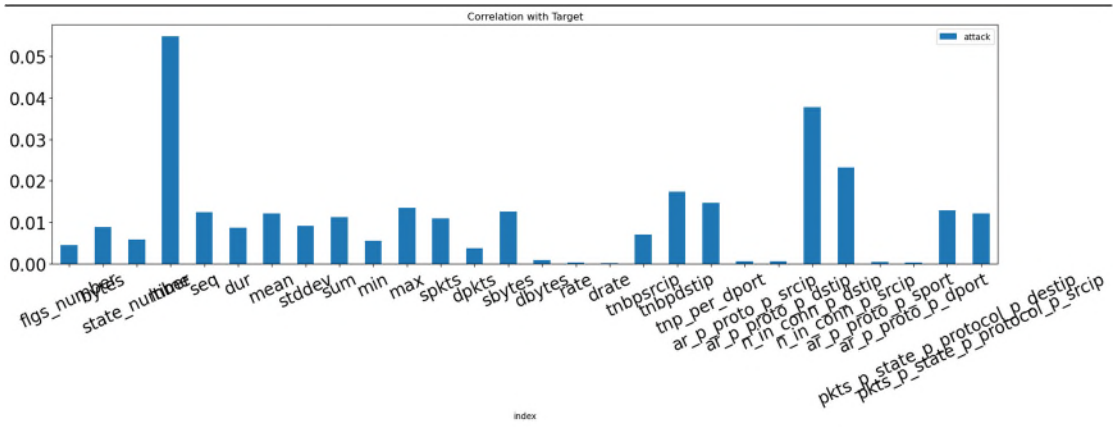


Figure 4-1. Feature correlation with target variable

Removing features that are highly correlated with the target variable can help improve the performance of a machine-learning model in several ways:

1. Reduce overfitting: When a feature is highly correlated with the target variable, the feature has strong predictive power over the target variable. When the feature

is too closely tied to the target variable, the model may overfit the training data and will not generalize well to new, unseen data. Removing highly correlated features with the target variable can help reduce the risk of overfitting and improve the model's ability to generalize.

2. Reduce multicollinearity: When two or more features are highly correlated with each other and the target variable, it can lead to multicollinearity, which can cause problems for some machine learning algorithms. Multicollinearity can make it difficult for the algorithm to determine the true relationship between the features and the target variable. It can result in unstable and unreliable model performance. Removing one or more highly correlated features can help reduce multicollinearity and improve the model's stability and reliability. In this praxis a threshold of 90% is used and the redundant feature kept was the one having lower correlation coefficient with the target variable.

3. Improve interpretability: In some cases, highly correlated features with the target variable may not be predictive but may instead be proxies for other underlying factors. Removing these features can help improve the interpretability of the model, making it easier to understand which features are truly driving the model's predictions.

Similarly, the feature correlation figures of the remaining datasets can be found in Figures A-1, A-2 and A-3 in the Appendix.

4.2.1 Feature Selection for first dataset BoT-IoT

After removing the above redundant features, we used the ANOVA F-value to determine the top features and their relationship with the target variable. Table 4-5 illustrates all features with their corresponding F and p scores.

The F-score value measures the difference in means between groups, which measures how well a particular feature separates the data into distinct classes. A higher F-score indicates that the feature is more relevant in distinguishing between classes and should be considered for inclusion in the classification model.

On the other hand, the p-value indicates the probability of observing the F-score value by chance alone, assuming the null hypothesis is true. In this praxis, 0.05 is the threshold used for the p-value, below which we consider the feature statistically significant and relevant for classification.

Therefore, a lower p-value indicates a stronger statistical significance of the F-score, implying that the feature is more important for classification.

Table 4-5. ANOVA analysis for BoT-IoT Dataset

	feature	f_value	p_value
17	tnbpsrcip	327646.610347	0.000000e+00
18	tnbpdstip	260036.945876	0.000000e+00
11	spkts	253140.890494	0.000000e+00
13	sbytes	202218.967300	0.000000e+00
1	bytes	168599.540797	0.000000e+00
19	tnp_per_dport	138115.048344	0.000000e+00
8	sum	122527.451508	0.000000e+00
12	dpkts	80285.787996	0.000000e+00
14	dbytes	77047.450482	0.000000e+00
26	pkts_p_state_p_protocol_p_destip	43954.752945	0.000000e+00
5	dur	40879.084816	0.000000e+00
27	pkts_p_state_p_protocol_p_srcip	30306.061614	0.000000e+00
3	ltime	20833.310634	0.000000e+00
22	n_in_conn_p_dstip	10249.996057	0.000000e+00
23	n_in_conn_p_srcip	3096.576458	0.000000e+00
4	seq	1051.126207	1.493827e-230
10	max	848.544810	1.582286e-186
6	mean	621.167599	4.278229e-137
7	stddev	530.936153	1.801596e-117
0	flgs_number	111.837275	3.882110e-26
9	min	107.336539	3.759451e-25
21	ar_p_proto_p_dstip	47.695787	4.978406e-12
2	state_number	24.042408	9.424113e-07
16	drate	14.700286	1.260294e-04
15	rate	5.153509	2.319951e-02
20	ar_p_proto_p_srcip	0.592722	4.413684e-01
25	ar_p_proto_p_dport	0.537731	4.633746e-01
24	ar_p_proto_p_sport	0.335043	5.627047e-01

In summary, the F-score value in ANOVA measure the relevance of each feature in separating data into classes. In contrast, the p-value measures the statistical significance of the F-score. Together, they help identify the most relevant and statistically significant features for classification as illustrated in Table 4-5 above.

Similarly, the ANOVA analysis tables of the remaining datasets can be found in Table A-8, A-9 and A-10 in the Appendix.

As explained in section 3.5 in Chapter 3. For the large data set, it is challenging to discard the co-related feature set and reduce the dimension of the data set. So, to enhance the process and select a further smaller subset of features, we used Logistic Regression as a feature importance method. Logistics regression coefficients were calculated per each feature as illustrated in Table 4-6 below and first twenty features were selected for further processing.

Table 4-6. Logistics regression coefficients for feature selection

	feature	importance
0	tnp_per_dport	0.186
1	ltime	0.1204
2	tnbpsrcip	0.0999
3	pkts_p_state_p_protocol_p_destip	0.0661
4	seq	0.0587
5	n_in_conn_p_dstip	0.0522
6	tnbpdstip	0.0493
7	spkts	0.0482
8	min	0.0415
9	flgs_number	0.0405
10	mean	0.0402
11	n_in_conn_p_srcip	0.0313
12	max	0.0312
13	dpkts	0.0294
14	ar_p_proto_p_dport	0.0241
15	state_number	0.0202
16	ar_p_proto_p_sport	0.016
17	sbytes	0.0152
18	bytes	0.0116
19	sum	0.0053
20	dbytes	0.0042
21	rate	0.0034
22	pkts_p_state_p_protocol_p_srcip	0.0031
23	stddev	0.0009
24	ar_p_proto_p_dstip	0.0004
25	ar_p_proto_p_srcip	0.0004
26	dur	0.0002
27	drate	0.0001

Similarly, the Logistics regression coefficients for feature importance selection of the remaining datasets, can be found in Tables A-11, A-12 and A-13 in the Appendix.

Figure 4-2 illustrates the correlation heatmap for the features. As we can see there are no highly correlated features anymore with 90% or more correlation coefficient.

Similarly, the correlation heatmaps for the remaining datasets, can be found in Figures A-4, A-5 and A-6 in the Appendix.

The BoT-IoT dataset exhibits a wide range of scales of values among its features, thus necessitating z-score standardization to standardize all numerical features to a mean of zero and unit standard deviation and variance. The dataset underwent standardization, as elaborated in sections 3.4 and 4.2.1.1.

A histogram of the features, as shown in Figure 4-3, reveals non-uniform and skewed distributions. Consequently, the standardization process resulted in the standardization of all features' mean to zero, as shown in Figure 4-5.

Incorporating standardization for all numerical features in the dataset eliminates the potential for bias in ML models from features with larger scales. This process guarantees equal treatment for all features. It avoids a situation where a feature's impact on model performance is exaggerated merely because its values have a larger scale than other features in the dataset. Instead, the standardized features directly affect the target variable, promoting fair and unbiased analysis.

Similarly, the features' histograms before standardization, for the remaining datasets, can be found in Figures A-7, A-8 and A-9 in the Appendix.

As explained in section 3.5 in Chapter 3, PCA is used in this praxis to reduce the dimensionality while retaining as much information as possible. PCA projects the high-dimensional data onto a 3D plot, making visualizing the relationships between the data points easier. This can improve the performance of algorithms by reducing computation time. PCA can also handle large datasets like the ones processed in this praxis, and it is computationally efficient. PCA projects the input features onto a new set of orthogonal axes that captures the most important information in the data, Figure 4-4 illustrates this concept. The below Principal Components are newly created variables obtained by linearly combining or mixing the original variables.

These combinations are designed so that the resulting principal components are uncorrelated, and most of the information contained in the original variables is compressed or condensed into the first components. The goal of PCA is to capture the maximum amount of information possible in the first component, followed by the maximum remaining information in the second component, and so on, resulting in a scree plot that demonstrates this distribution of information across the components.

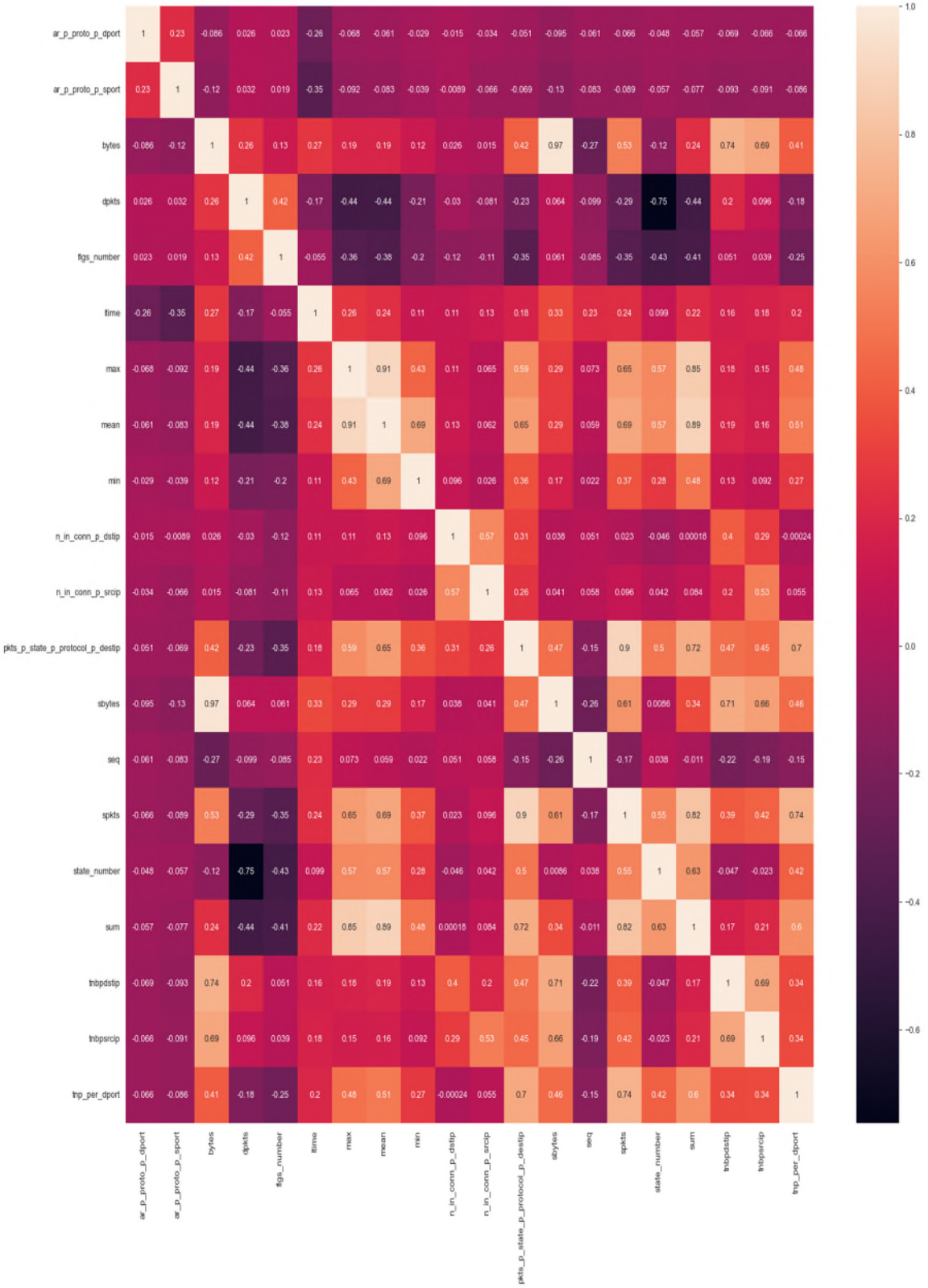

ar_p_proto_p_dport
ar_p_proto_p_sport
bytes
dpkts
flgs_number
ftime
max
mean
min
n_in_conn_p_dstip
n_in_conn_p_srcip
pkts_p_state_p_protocol_p_destip
sbytes
seq
spkts
state_number
sum
tnbpdstip
tnbpsrcip
tnp_per_dport
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Figure 4-2. Correlation matrix for BoT-IoT dataset

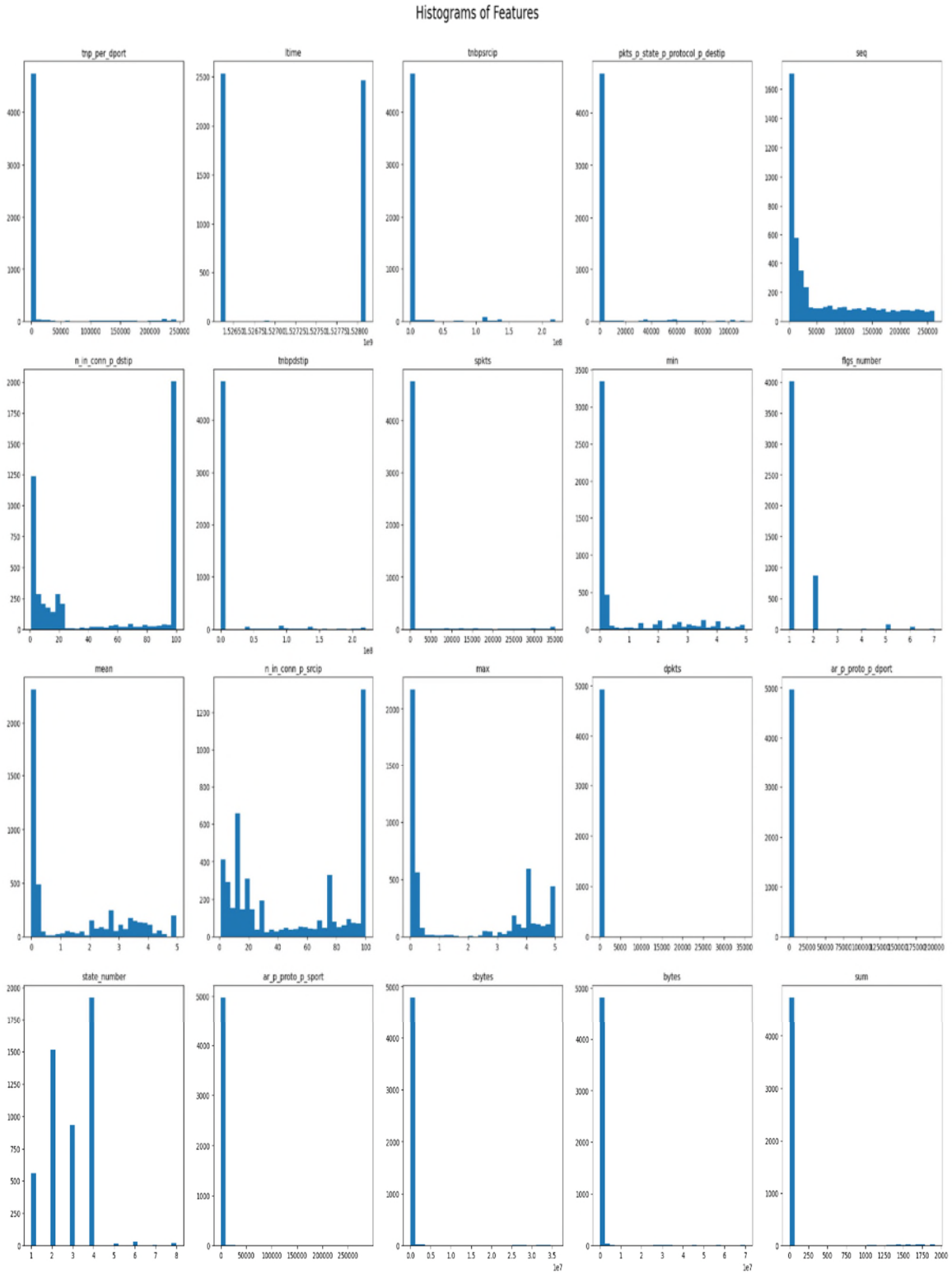


Figure 4-3. Histogram of Bot-IoT dataset features before standardization

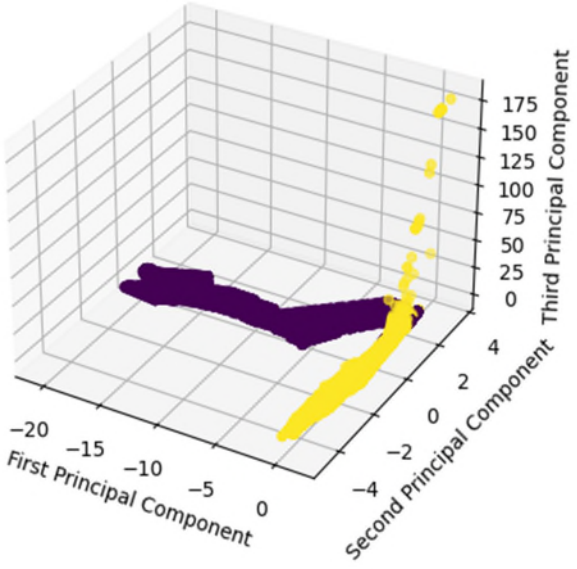


Figure 4-4. The three principal components of BoT-IoT dataset

Similarly, the diagrams of the three principal components, for the remaining datasets, can be found in Figures A-10, A-11 and A-12 in the Appendix.

We can see that all features now have a distribution near the normal distribution pattern, as shown in Figure 4-5 below.

Figure 4-6 shows the boxplot after performing PCA. Similarly, the features' histograms after PCA and standardization, for the remaining datasets, can be found in Figures A-13, A-14 and A-15 in the Appendix.

Similarly, the features' boxplots after PCA and standardization, for the remaining datasets, can be found in Figures A-16, A-17 and A-18 in the Appendix.

Out of the twenty previously selected features, Figure 4-7 below illustrates that only 5 features can explain 80% of target variable. And 12 features can explain 100% of the target variable. Those 12 features are listed in Figure 4-7 from left to right.

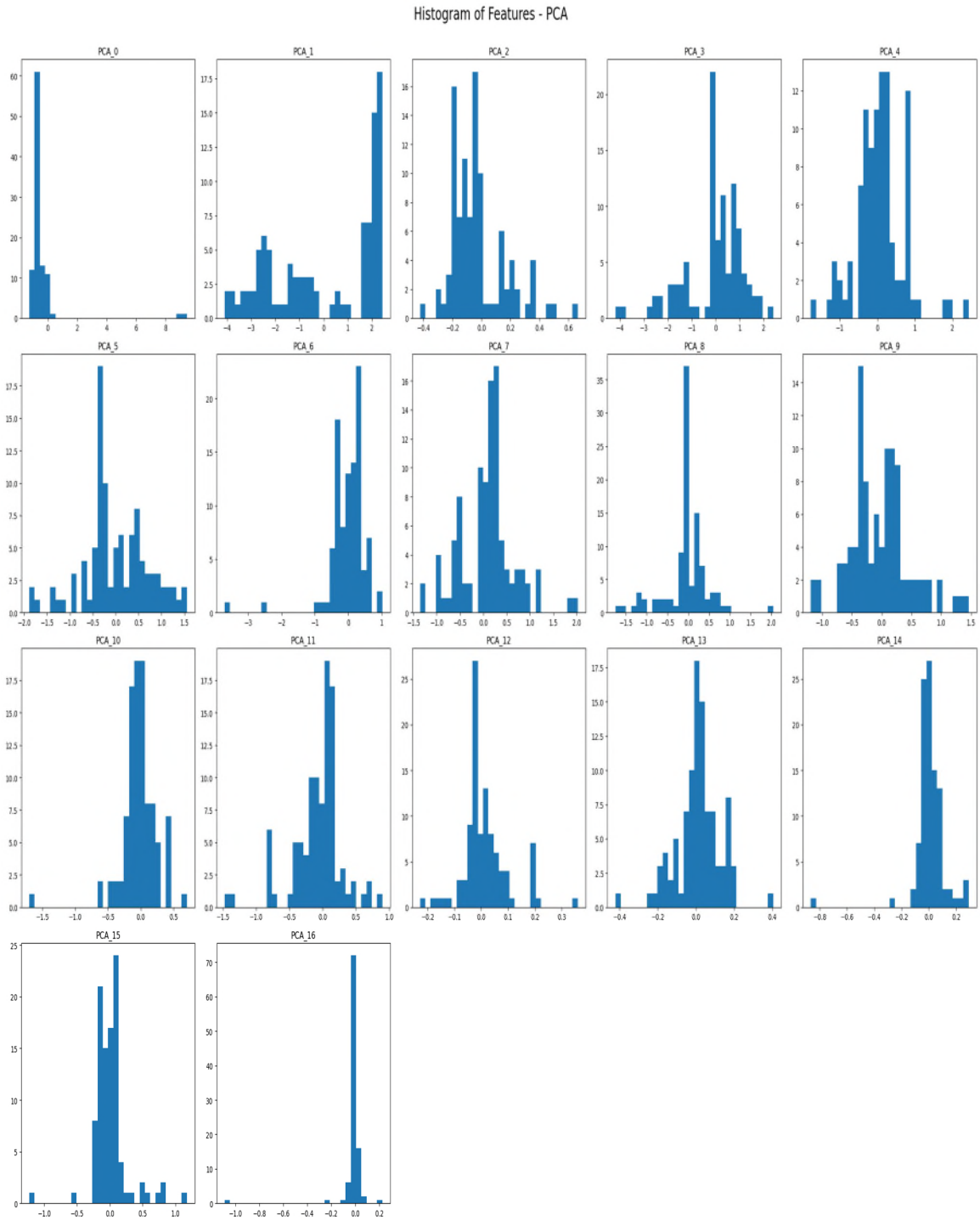


Figure 4-5. Histogram of Bot-IoT dataset features after PCA and standardization

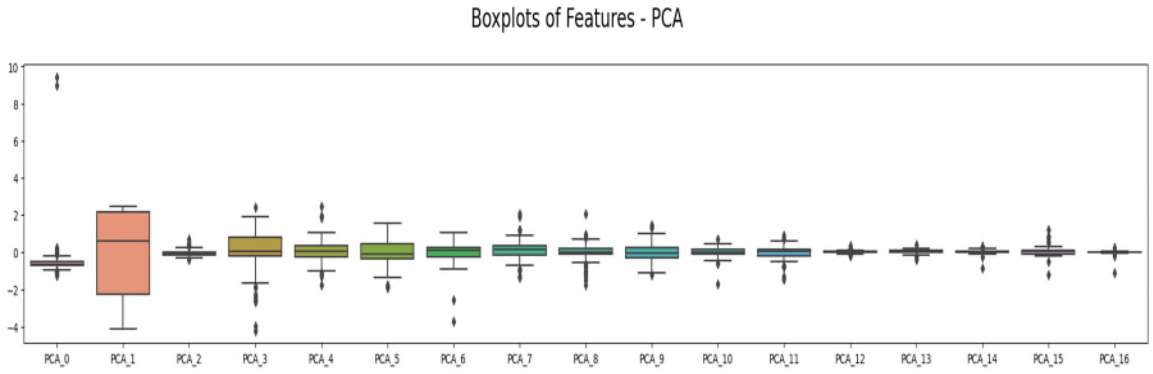


Figure 4-6. Boxplots of Bot-IoT dataset features after PCA

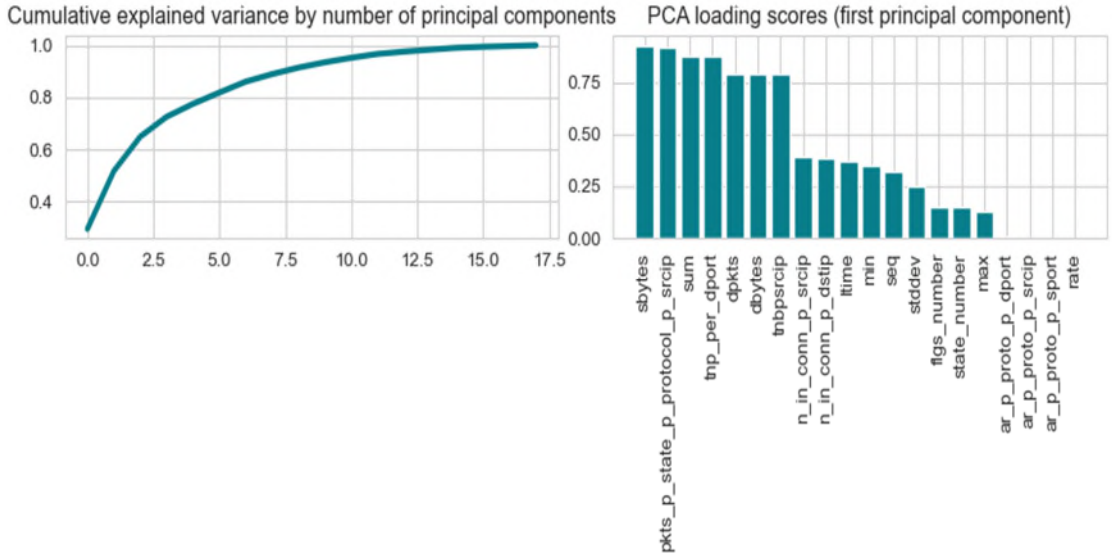


Figure 4-7. Explained variance by number of principal components

Similarly, the explained variance by number of principal components diagrams, for the remaining datasets, can be found in Figures A-19, A-20 and A-21 in the Appendix.

4.3 Model Development for first dataset BoT-IoT

This section discusses the results of model hyperparameter tuning and model performance evaluation of the seven ML models examined in this praxis to use them in the private 5G deployment scenarios for the first dataset.

4.3.1 Model Hyperparameter Tuning

This section discusses the results of model hyperparameter tuning and model performance evaluation of the seven ML models examined in this praxis to use them in the private 5G deployment scenarios for the first dataset. The hyperparameter tuning process is a crucial aspect of model preparation in this study, as explained in section 3.7.1. It involves selecting optimal values for the various hyperparameters associated with each ML model. Table 4-7 displays the results of this process, with the selected value for each hyperparameter listed in the third column, following an examination of the range of values tested for each hyperparameter listed in the second column.

The n_estimators hyperparameter of Random Forest (RF) specifies the number of Decision Tree (DT) weak learners utilized to build the strong model. The process of hyperparameter tuning for the RF model revealed that the optimal performance was achieved by employing 407 DTs. Additionally, the hyperparameters that were investigated for the RF model included max_features, max_depth of the DTs, and the criterion used for splitting each node in a DT. The selected values for these hyperparameters were sqrt, 11, and gini, respectively. min_samples_split was 7 whereas min_sample_leaf is 3.

In the realm of ensemble machine learning (ML) models, XGBoost shares a resemblance to RF in terms of its architecture. To optimize the performance of XGBoost, it is crucial to adjust its n_estimators hyperparameter, which represents the number of weak decision trees used in the model. Following experimentation, the optimal number of trees for XGBoost was determined to be 539, with each tree having a maximum depth of 12. Additionally, the hyperparameters learning_rate, reg_alpha, reg_lambda, and

min_child_weight were examined, and their respective values were selected as 0.7432, 4.0946, 4.27090, and 4.

Table 4-7. Hyperparameter tuning results for the seven examined models for BoT-IoT dataset

Hyperparameter being tuned	Search space	Final value
XGB		
n_estimators	10,1000,1	539
max_depth	2,20,1	12
learning_rate	0.01 ,1	0.7432
reg_alpha	0,10	4.0946
reg_lambda	0,10	4.2709
min_child_weight	1,10,1	4
Gaussian NB		
var_smoothing	-10,1	0.000257693
Decision Tree		
min_samples_split	2,20,1	2
min_sample_leaf	1,20,1	1
max_features	sqrt, log2, none	none
Ccp_alpha	0, x	3.65141E-05
RNN		
units	64	
activation	relu	
solver/optimizer	adam	
Bagging (Decision trees)		
n_estimators	10,1000,1	536
max_samples	0.1,1	0.457088607
max_features	0.1,1	0.916658909
bootstrap	True, False	TRUE
SVC		
C	0,10	7.127249271
gamma	scale,auto	auto
kernel	linear, poly, sigmoid, rbf	linear
Random Forest		
n_estimators	10,1000,1	407
min_samples_split	2,20,1	7
min_sample_leaf	1,20,1	3
max_features	sqrt ,log2,none	sqrt
max_depth	2,20,1	11
criterion	Gini, entropy	Gini

The cost complexity parameter, also known as Ccp_alpha (α), is a tuning parameter that controls the amount of pruning to be performed. The algorithm works by computing a sequence of subtrees of increasing complexity, with each subtree corresponding to a different alpha value. The algorithm then selects the subtree with the

lowest cost complexity, corresponding to the optimal balance between accuracy and complexity.

The algorithm stops trying new alpha values when it finds a subtree with the lowest cost complexity. The algorithm uses validation sets to estimate the generalization performance of each subtree and select the best value of alpha that minimizes the validation error.

Possible alpha values along with their respective accuracies for DT model were plotted in Figure 4-8 below.

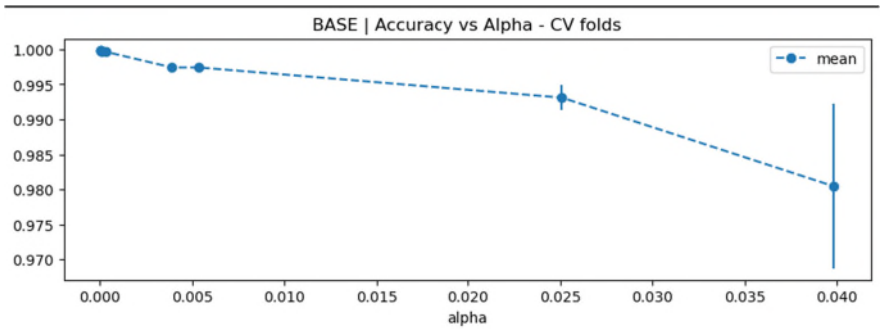


Figure 4-8. Possible tested alpha values versus the accuracy for pruned DT model

These findings highlight the importance of fine-tuning hyperparameters in XGBoost to attain the best possible results. The remaining results and their explanations can be found in Table 3.1 and Table and 4.7.

Similarly, the hyperparameter tuning results, for the remaining datasets, can be found in Tables A-14, A-15 and A-16 in the Appendix.

4.3.2 Model performance results

After tuning the hyperparameters of each of the ML models investigated in this praxis, the models are trained with the chosen hyperparameters. Each of the ML models

was trained on the training dataset, and their performance was measured using the validation dataset. The selected features per dataset can be found in Table 4-17.

The confusion matrix illustrated in Figure 4-9 below is a table used to evaluate a classification model's performance by comparing the predicted class labels with the true class labels of a test data set. Similarly, the confusion tables for the remaining datasets can be found in Tables A-22, A-23, and A-24 in the Appendix.

Table 4-8 compares the performance of seven machine learning (ML) classifiers for the BoT-IoT dataset. The results indicate that the random forest (RF) classifier outperformed all other classifiers, achieving a 99.991% accuracy rate, 100% area under the curve, and zero false positive rate (FPR). In contrast, the CART decision tree was the fastest, with an inference time of 0.000817768 milliseconds per record and an accuracy of 99.9551%.

Notably, Gaussian Naive Bayes (GNB) obtained the lowest accuracy of 99.7309% and the highest FPR of 0.169591. Additionally, the recurrent neural network (RNN) exhibited the slowest prediction time per record, with 1.0523856 milliseconds above the 0.5 milliseconds latency threshold considered suitable for 5G applications discussed in previous chapters. The red color shows the models which are not fit to be used in 5G deployments, whereas the green color denotes the recommended ones.

Given the high occurrence of false positives in intrusion detection systems, we excluded GNB from our final selection table. Similarly, we removed RNN due to its unsuitable inference time for 5G private use cases. However, RNN may be helpful for many other applications outside our 5G focus. Removed models are highlighted in red in below tables.

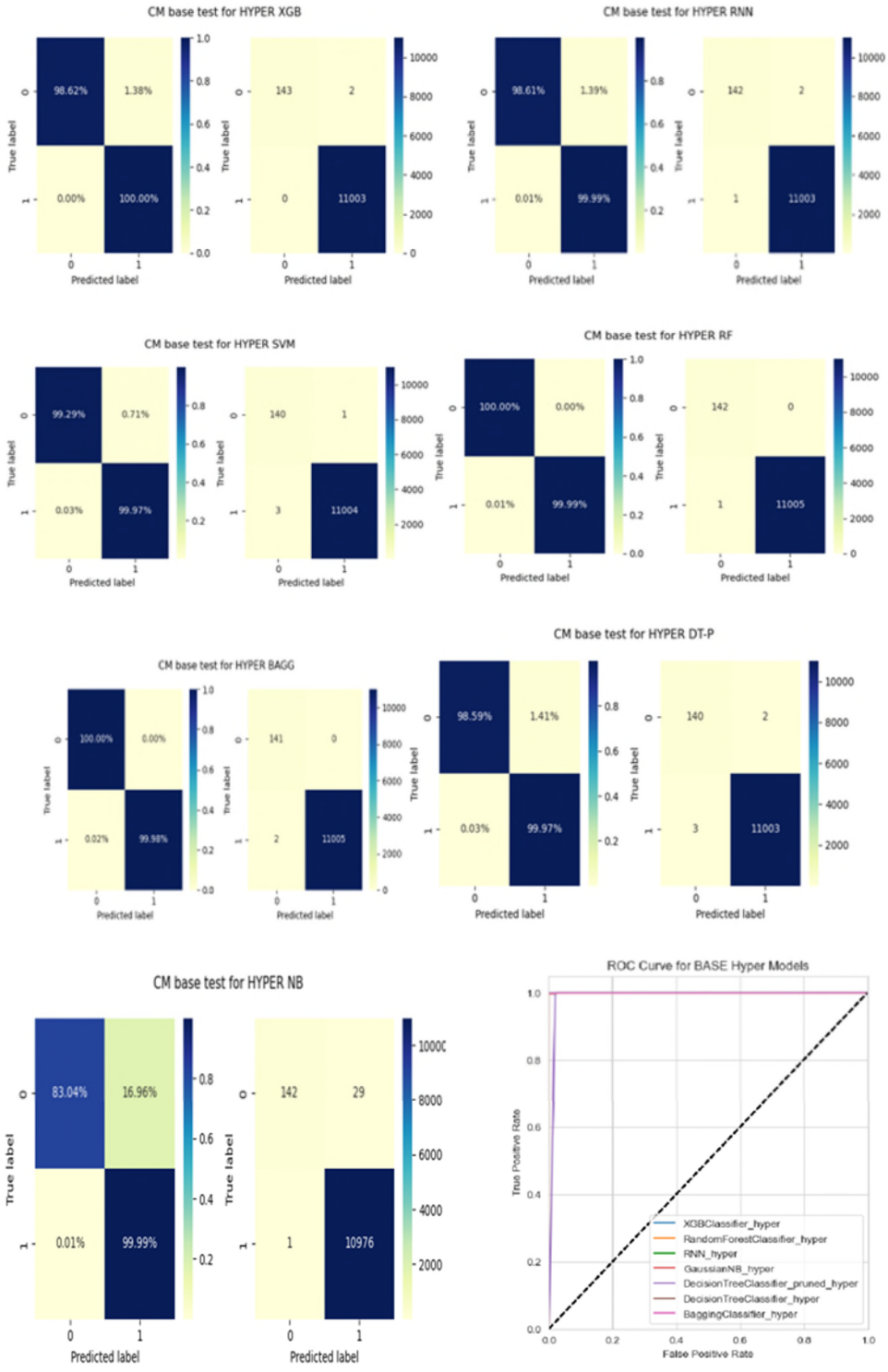


Figure 4-9. Confusion tables and AUC-ROC graph for all trained models of Bot-IoT dataset

Table 4-8. Model performance results for BoT-IoT dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier (CART)	0.999551	0.99955	0.999551	0.999981	0.999551	0.989464	0.998	0.997333	0.999727	0.014085	0.5505	0.000817768	0.0166
GaussianNB	0.997309	0.997735	0.997309	0.997488	0.997427	0.999935	0.998	0.994667	0.999909	0.169591	0.014	0.008886413	0.007
XGBClassifier	0.999821	0.999823	0.999821	0.999786	0.999821	1	0.992667	0.986	1	0.013793	12.3118	0.013356878	1.5374
RandomForestClassifier	0.99991	0.99991	0.99991	0.999961	0.99991	1	1	0.994667	0.999909	0	26.5121	0.227966467	2.3425
BaggingClassifier	0.999821	0.999821	0.999821	0.999961	0.99982	0.999999	0.997333	0.993333	0.999818	0	30.4503	0.318629682	1.7458
SVM (SVC)	0.999641	0.99964	0.999641	0.999766	0.99964		0.994	0.996667	0.999727	0.007092	0.6395	0.359599861	0.024
RNN	0.999731	0.999732	0.999731	0.999688	0.999731	0.999994	0.989333	0.996667	0.999909	0.013889	17.3321	1.052385675	3.6029

Table 4-9 displays the remaining models suitable for the 5G private use case. The accuracies of all models exceed the 99.99% threshold, leading us to select the fastest models in prediction time. The CART decision tree, with an inference time of 0.000817 milliseconds, is the fastest classifier, followed by the XGB classifier with 0.01335 milliseconds. At the same time, XGB is having higher accuracy of %99.9821 compared to CART's accuracy of %99.9551. This implies that these two classifiers are highly effective in detecting attacker attempts without compromising the assigned 0.5-millisecond latency in a 5G transport pipe, with an accuracy level of over 99.99%.

It is worth noting that RF achieved the highest accuracy of %99.991, but it comes third with an inference time of 0.227 milliseconds was slower than CART and XGB. Since all of them have accuracies exceeding 99.9%, the first two algorithms were considered as the most relevant for the sake of this praxis purpose, as they were the ones having the shortest classification time.

Table 4-9. Selected models that fit private 5G scenarios for Bot-IoT dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier (CART)	0.999551	0.99955	0.999551	0.999981	0.999551	0.989464	0.998	0.997333	0.999727	0.014085	0.5505	0.000817768	0.0166
XGBClassifier	0.999821	0.999823	0.999821	0.999786	0.999821	1	0.992667	0.986	1	0.013793	12.3118	0.013356878	1.5374
RandomForestClassifier	0.99991	0.99991	0.99991	0.999961	0.99991	1	1	0.994667	0.999909	0	26.5121	0.227966467	2.3425
BaggingClassifier	0.999821	0.999821	0.999821	0.999961	0.99982	0.999999	0.997333	0.993333	0.999818	0	30.4503	0.318629682	1.7458
SVM (SVC)	0.999641	0.99964	0.999641	0.999766	0.99964		0.994	0.996667	0.999727	0.007092	0.6395	0.359599861	0.024

Employing a comparable methodology for the remaining datasets, it becomes evident that the CART decision tree model consistently delivers excellent performance across diverse datasets and features along with the XGB classifier. Both are consistently demonstrating the shortest prediction time for attack detection. The outcomes for the other three datasets are illustrated in Tables 4-10, 4-11, 4-12, 4-13, 4-14, and 4-15.

Table 4-10. Model performance results for IoT Healthcare dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier (CART)	0.999864112	0.999864144	0.999864	0.999797	0.999864	1	0.993333	0.998	1	0.000237	0.038	0.010599171	0.008
XGBClassifier	0.998777008	0.998777096	0.998777	0.999392	0.998777	0.999997	0.998	0.999333	0.998405	0.000947	0.9964	0.084793369	0.2591
BaggingClassifier	0.999864112	0.999864155	0.999864	1	0.999864	1	0.998	0.998	0.999681	0	5.6839	1.606304387	0.9358
SVM (SVC)	0.998369344	0.998369629	0.998369	0.998327	0.998369		0.993333	0.988667	0.997767	0.001184	0.4861	2.080617296	0.0161
RandomForestClassifier	0.999456448	0.999457141	0.999456	0.999747	0.999456	0.999999	0.999333	0.998667	0.998725	0	8.758	3.410283316	2.8764
RNN	0.998369344	0.998369629	0.998369	0.998378	0.998369	0.999523	0.985333	0.985333	0.997767	0.001184	9.9204	13.27069223	4.4859
GaussianNB	0.970512298	0.970613504	0.970512	0.972318	0.970535	0.978315	0.971333	0.966667	0.959647	0.021256	0.023	0.068894612	0

Table 4-11. Selected models that fit private 5G scenarios for IoT Healthcare dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier (CART)	0.999864112	0.999864144	0.999864	0.999797	0.999864	1	0.993333	0.998	1	0.000237	0.038	0.010599171	0.008
XGBClassifier	0.998777008	0.998777096	0.998777	0.999392	0.998777	0.999997	0.998	0.999333	0.998405	0.000947	0.9964	0.084793369	0.2591

Table 4-12. Model performance results for ToN-IoT dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
RNN	0.71695219	0.713682233	0.716852186	0.659743848	0.703331929	0.787492904	0.665333	0.708	0.697305	0.27594	7.1727	4.937686821	2.4885
XGBClassifier	0.69464331	0.722456064	0.694643314	0.781459981	0.698502017	0.813065163	0.654667	0.706667	0.581164	0.188012	11.7886	0.533507513	0.9247
BaggingClassifier	0.69342879	0.72333571	0.693428791	0.829092095	0.697252665	0.808088057	0.678667	0.696667	0.578987	0.185195	81.2724	12.00790788	3.605
RandomForestClassifier	0.68019688	0.718299033	0.680196881	0.736371863	0.683767674	0.806905647	0.673333	0.716	0.563235	0.183442	12.3482	2.170926832	1.4404
DecisionTreeClassifier(CART)	0.67802352	0.725893142	0.678023523	0.764089832	0.680801302	0.797258386	0.684	0.672667	0.558533	0.168056	0.4861	0.012465129	0.019
GaussianNB	0.64107645	0.634484448	0.641076451	0.611183701	0.636553844	0.673354602	0.574667	0.624	0.542061	0.30695	0.007	0.029816309	0.003
SVM (SVC)	0.4672718	0.694520581	0.467271797	0.465785784	0.385476329		0.650667	0.706667	0.418851	0.130797	1.9718	5.551968368	0.083

Table 4-13. Selected models that fit private 5G scenarios for ToN-IoT dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier(CART)	0.67802352	0.725893142	0.678023523	0.764089832	0.680801302	0.797258386	0.684	0.672667	0.558533	0.168056	0.4861	0.012465129	0.019

Table 4-14. Model performance results for 5G-NIDD dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier(CART)	0.998080978	0.998080892	0.998080978	0.998718101	0.998080879	0.99952043	0.966	0.969333333	0.998227	0.002144	1.29	0.021054536	0.009
GaussianNB	0.891480388	0.906088703	0.891480388	0.859402802	0.887527651	0.98046889	0.8886667	0.868	0.850596	0.008435	0.0484	0.103874528	0.0102
XGBClassifier_hyper	0.999304091	0.999304202	0.999304091	0.999895659	0.999304042	0.99999568	0.9873333	0.986	0.999166	0.000483	28.1244	0.261043351	0.5187
RandomForestClassifier	0.998418389	0.998419602	0.998418389	0.998926782	0.998418053	0.99995618	0.984	0.988	0.997917	0.000805	38.1765	0.65630937	0.8213
SVM (SVC)	0.713686208	0.763187895	0.713686208	0.607656637	0.670585812		0.9893333	0.987333333	0.68619	0.118212	4.2188	2.375955062	0.0241
BaggingClassifier	0.998840152	0.998840805	0.998840152	0.999619902	0.998839971	0.99997561	0.9933333	0.992666667	0.998471	0.00059	450.3285	8.855077351	5.5727
RNN	0.998186419	0.998186413	0.998186419	0.995990309	0.998186266	0.99996642	0.968	0.972666661	0.998193	0.001823	54.2563	11.66191021	5.3999

Table 4-15. Selected models that fit private 5G scenarios for 5G-NIDD dataset

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms	T_cv
DecisionTreeClassifier(CART)	0.998080978	0.998080892	0.998080978	0.998718101	0.998080879	0.99952043	0.966	0.969333333	0.998227	0.002144	1.29	0.021054536	0.009
XGBClassifier_hyper	0.999304091	0.999304202	0.999304091	0.999895659	0.999304042	0.99999568	0.9873333	0.986	0.999166	0.000483	28.1244	0.261043351	0.5187

To evaluate potential overfitting issues that can occur in machine learning models, we conducted an analysis of the F1 scores for both the training and testing datasets. F1 scores reflect both precision and accuracy. If the F1 score for the training set is significantly higher than that of the testing set, it suggests that the model is overfitting the training data and may not generalize well to new data. Conversely, if the F1 score for the testing set is much lower than that of the training set, the model may be underfitting the training data and needs adjustments or greater complexity.

Additionally, we assessed the cross-validation scores for both the training and testing datasets. If the cross-validation score for the training set is substantially higher than that of the testing set, it may indicate overfitting to the training data. In contrast, if the cross-validation score for the testing set is significantly lower than that of the training set, the model may be underfitting the training data and failing to learn critical data patterns. Analysis of the F1 and CV scores mentioned above indicates no evidence of overfitting in the final developed models, as shown in the above results.

4.4 Feature fusion results and the final model for public 5G deployment

This section will discuss the results of building and testing the performance of a suitable model for public 5G deployments utilizing the feature fusion technique. The objective is to develop a model which can be applied universally across all industries. As explained in section 3.8.2, feature fusion is a technique used in machine learning to combine multiple sources or types of feature information, enhancing a model's performance. Table 4-16 shows the structure of the Combined dataset, with 711,647 records.

Table 4-16. Combined dataset

Dataset	Size	Sample	Ratio
BoT-IoT	3668522	476907	0.13
IoT HealthCare	188694	24530	0.13
ToN-IoT	401119	52145	0.13
5G-NIDD	1215890	158065	0.13
Combined	5,474,225	711,649.25	0.13

Due to the dataset's different sizes, null values appeared after combining all four datasets, as illustrated in Figure 4-10 below. We solved this problem by filling those null values with the average. The combined dataset has a total of 362 features.

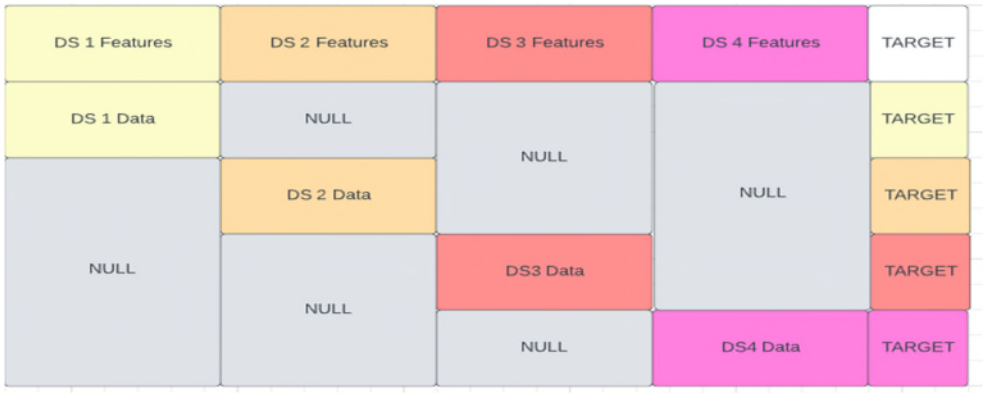


Figure 4-10. The record structure of the combined four datasets

The feature correlation diagram with the target variable can be found in Figure A-25 in the Appendix. Like all previous datasets, and as per our methodology in this praxis, redundant features were removed and ANOVA F-value was used to determine the top features and their relationship with the target variable, followed by logistics regression coefficients for feature selection. The features were reduced after implementing above steps from 362 features to only 78 features. We used a threshold of first 20 features based on below coefficients. The results can be seen in Table 4-17. In addition, Figure A-26 in the Appendix illustrates the correlation heatmap for the features. There are no highly correlated features with 90% or more correlation coefficient.

As Figure 3-7 explained in the methodology chapter, PCA for dimensionality reduction was applied to the final set of features after feature fusion was implemented.

Figures A-27 in the Appendix illustrate the histogram of the combined dataset features after PCA, and standardization were implemented.

Table 4-17. Logistics regression coefficients for feature selection for the combined dataset

	feature	importance
0	0_tcp_dstport	0.1987
1	0_tcp_srcport	0.1793
2	0_tcp_hdr_len	0.1374
3	0_mqtt_msgtype	0.0793
4	0_frame_time_relative	0.0724
5	0_tcp_checksum	0.0484
6	0_tcp_time_delta	0.044
7	0_ip_ttl	0.0267
8	0_mqtt_topic_nan	0.0254
9	0_tcp_flags_push	0.0248
10	0_tcp_flags	0.022
11	0_mqtt_msg	0.0183
12	0_tcp_window_size_value	0.0181
13	0_mqtt_topic	0.018
14	0_mqtt_len	0.0163
15	0_tcp_ack	0.0101
16	0_mqtt_qos	0.0098
17	0_tcp_flags_fin	0.0082
18	0_tcp_flags_syn	0.0075
19	0_tcp_payload	0.0058

	feature	importance
0	2_con	0.0975
1	2_ef	0.0912
2	2_seq	0.09
3	2_dttl	0.077
4	2_fin	0.0617
5	2_rst	0.0572
6	2_ackdat	0.0567
7	2_smeanpktsz	0.0534
8	2__e_	0.0417
9	2_icmp	0.0394
10	2_tcp	0.0375
11	2_srcwin	0.0372
12	2_sttl	0.0316
13	2_srcbytes	0.0295
14	2_offset	0.0288
15	2__e_s_	0.0238
16	2_int	0.0223
17	2_start	0.0168
18	2_cs0	0.0147
19	2_ploss	0.0139

	feature	importance
0	1_bytes	0.5017
1	1_dbytes	0.4117
2	1_dpkts	0.0359
3	1_pkts	0.0191
4	1_tnp_perproto	0.0112
5	1_tnp_pdstip	0.0068
6	1_tnbpsrcip	0.0059
7	1_tnbpdstip	0.0017
8	1_n_in_conn_p_dstip	0.0014
9	1_state_number	0.0008
10	1_max	0.0006
11	1_tnp_psrcip	0.0005
12	1_stddev	0.0005
13	1_min	0.0004
14	1_dur	0.0004
15	1_drate	0.0003
16	1_n_in_conn_p_srcip	0.0003
17	1_ar_p_proto_p_srcip	0.0001
18	1_ar_p_proto_p_dstip	0.0001
19	1_rate	0.0001

	feature	importance
0	4_ts	0.6465
1	4_light_status_nan	0.2483
2	4_sphone_signal	0.0518
3	4_door_state	0.0349
4	4_latitude	0.0062
5	4_pressure	0.0035
6	4_temp_condition	0.0022
7	4_humidity	0.0021
8	4_fridge_temperature	0.001
9	4_light_status	0.0008
10	4_thermostat_status	0.0008
11	4_fc3_read_holding_register	0.0005
12	4_motion_status	0.0005
13	4_current_temperature	0.0004
14	4_temperature	0.0002
15	4_fc1_read_input_register	0.0001
16	4_fc2_read_discrete_value	0.0001
17	4_fc4_read_coil	0

We can notice that features were further reduced to 68 only. At the same time, Figure A-28 illustrates the three principal components of the combined dataset. Figure 4-11 below explains the variance by the number of principal components for the combined dataset.

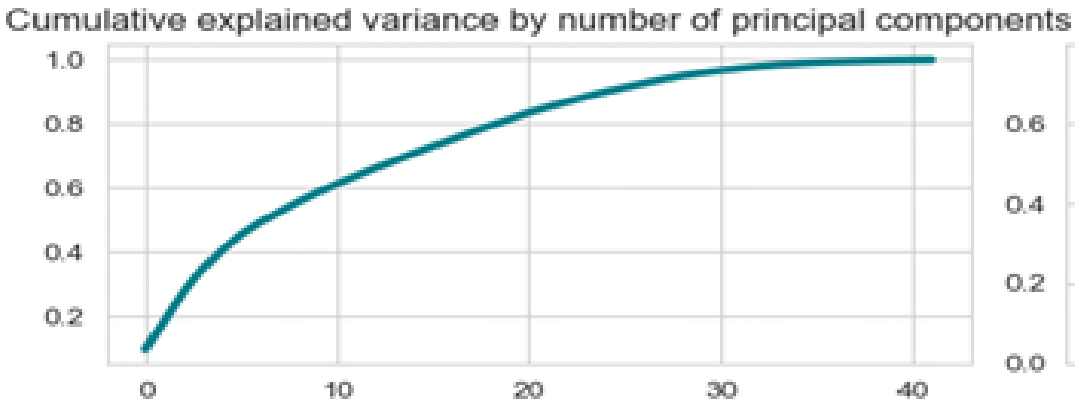


Figure 4-11. Cumulative Explained variance for the combined dataset

In this study, a total of 68 features were initially selected, and Figure 4-11 demonstrates that only 20 of these features can account for 80% of the target variable. Meanwhile, all 30 features can explain 100% of the target variable. After feature selection, models were trained, and hyperparameter tuning was conducted using a similar approach to previous datasets. The outcomes of this process are presented in Table 4-18, while the confusion tables and AUC-ROC graph for the combined dataset can be found in Figure A-29 in the Appendix.

Table 4-19 provides a comparative analysis of the performance of seven ML classifiers on the combined dataset. The results reveal that the bagging classifier, SVC, and RNN models have an excessively high inference time beyond the 5G target threshold of 0.5 milliseconds. Therefore, these three classifiers were eliminated from further consideration.

Table 4-18. Hyperparameter tuning results for the seven examined models for the combined dataset

Hyperparameter being tuned	Search space	Final value
XGB		
n_estimators	10,1000,1	178
max_depth	2,20,1	16
learning_rate	0.01 ,1	0.6960988
reg_alpha	0,10	1.993026823
reg_lambda	0,10	6.466198556
min_child_weight	1,10,1	1
Gaussian NB		
var_smoothing	-10,1	0.077835269
Decision Tree		
min_samples_split	2,20,1	2
min_sample_leaf	1,20,1	1
max_features	sqrt, log2, none	none
Ccp_alpha	0, x	0.008695629
Bagging (Decision trees)		
n_estimators	10,1000,1	203
max_samples	0.1,1	0.860484841
max_features	0.1,1	0.196691202
bootstrap	True, False	TRUE
SVC		
C	0,10	7.163317696
gamma	scale,auto	scale
kernel	linear, poly, sigmoid, rbf	linear
Random Forest		
n_estimators	10,1000,1	134
min_samples_split	2,20,1	18
min_sample_leaf	1,20,1	19
max_features	sqrt ,log2,none	log2
max_depth	2,20,1	11
criterion	Gini, entropy	gini
RNN		
units	64	
activation	relu	
solver/optimizer	adam	

Table 4-20 further shows that the Gaussian Naive Bayes (GNB) classifier has the highest false positive rate (FPR) of 0.19 among all classifiers, resulting in a relatively low accuracy rate of 92.466%. Consequently, the GNB classifier was also disqualified.

Table 4-19. Model performance results for the combined dataset

Model name	Accura	Precisi	Recall	F1_trai	F1_test	AUC	cv_trai	cv_test	TPR	FPR	T_train	inferen	T_cv
DT (CART)	0.989263	0.989217	0.989263	0.976539	0.989182	0.993071	0.97	0.985333	0.990507	0.018007	9.903	0.015457	0.0658
RF	0.998955	0.998958	0.998955	0.999217	0.998956	0.999993	0.986	0.989333	0.999703	0.005212	11.7121	0.262771	0.2702
GaussianNB	0.924662	0.921057	0.924662	0.799544	0.92169	0.971624	0.882667	0.886	0.942032	0.196382	0.0805	0.339213	0.007
XGBClassifier	0.99964	0.99964	0.99964	0.999909	0.99964	0.999997	0.991333	0.991333	0.999745	0.000951	68.8563	0.376872	1.5379
BaggingClassifier	0.999748	0.999748	0.999748	1	0.999748	0.999999	0.992	0.998	0.999873	0.000951	607.8034	11.27891	5.0163
SVM (SVC)	0.864205	0.877037	0.864205	0.420351	0.81512		0.844667	0.932667	0.862563	0.041929	13.4064	11.71002	0.1238
RNN	0.998415	0.998418	0.998415	0.998289	0.998416	0.999676	0.954	0.958667	0.999405	0.007106	49.4475	15.29902	6.1143

Table 4-20. Model performance results for the combined dataset after disqualifying the slowest models

Model Name	Accuracy	Precision	Recall	F1_Train	F1_Test	AUC	CV_Train	CV_Test	TPR	FPR	T_Train	Inference time (ms)
DT (CART)	0.989263	0.989217	0.989263	0.976539	0.989182	0.993071	0.97	0.985333	0.990507	0.018007	9.903	0.0154571
RF	0.998955	0.998958	0.998955	0.999217	0.998956	0.999993	0.986	0.989333	0.999703	0.005212	11.7121	0.26277073
GaussianNB	0.924662	0.921057	0.924662	0.799544	0.92169	0.971624	0.882667	0.886	0.942032	0.196382	0.0805	0.33921312
XGB	0.99964	0.99964	0.99964	0.999909	0.99964	0.999997	0.991333	0.991333	0.999745	0.000951	68.8563	0.37687224

On the other hand, Table 4-21 illustrates that the XGBoost (XGB) model achieved the highest accuracy of 99.9639% and the lowest FPR of 0.000951. Random Forest (RF) and CART models displayed the second and third-best accuracies, respectively. These three models are suitable for deployment in 5G public scenarios.

Table 4-21. Model performance results for the combined dataset after disqualifying the slowest models

Model name	Accuracy	Precision	Recall	F1_train	F1_test	AUC	cv_train	cv_test	TPR	FPR	T_train	inference time_ms
DT(CART)	0.98926	0.989217	0.989	0.97654	0.98918	0.993071	0.97	0.98533	0.9905	0.01801	9.903	0.0154571
RF	0.99896	0.998958	0.999	0.99922	0.99896	0.999993	0.986	0.98933	0.9997	0.00521	11.712	0.2627707
XGB	0.99964	0.99964	1	0.99991	0.99964	0.999997	0.99133	0.99133	0.9997	0.00095	68.856	0.3768722

It is worthy to note that DT-CART was the fastest in classification with inference time of 0.01547, whereas RF and XGB displayed the second and third-shortest inference time.

Figure 4-12 shows a visualization of the pruned version of the decision tree classifier (CART) of the combined dataset. Whereas Figure 4-13 shows a visualization of the XGB classifier of the combined dataset. In Figure 4-12, we can see that we have only five leaf nodes after pruning. Samples represent the number of samples before the split. Value shows the benign and attack values.

The Gini Index or Impurity measures the probability of a random instance being misclassified when chosen randomly. A node is pure when its gini =0. The class represents the majority class in that node. At the same time, darker color means more pure nodes.

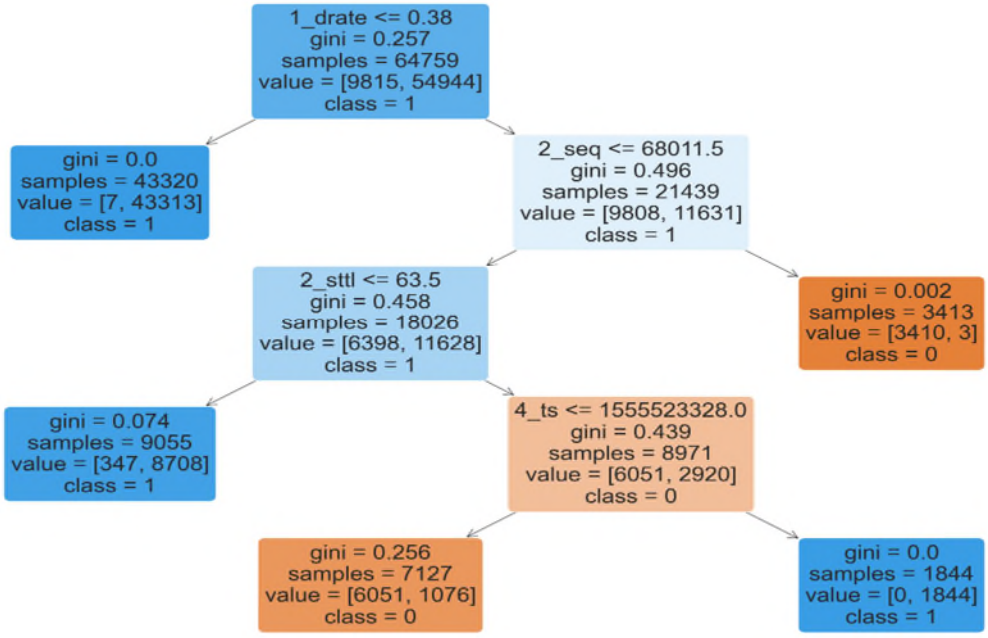


Figure 4-12. Visualization of a pruned Decision Tree classifier (CART) of the combined dataset

Chapter 5-Discussion and Conclusions

5.1 Discussion

This chapter includes three parts that aim to conclude this research. Section 5.1 reviews the research questions presented in Chapter 1 and summarizes the corresponding answers. Section 5.2 illustrates the contribution of this praxis to the body of knowledge. Whereas Section 5.3 recommends further future research directions.

5.1.1 Hypothesis 1

The initial research question in this study focuses on the features essential for the ML models to binary classify malicious IoT traffic.

RQ1: Which features are the most significant for detecting malicious IoT traffic across multiple domain datasets?

The first Hypothesis discussed in Chapter 1 assumes that out of the 362 features studied in this praxis, ten key features have the strongest influence on the model classification results. Those are found in H1 below, and their descriptions can be found in the Appendix.

H1: Features state_number, ltime, tnbpdstip, mqtt_msgtype, tcp_time_delta, sphone_signal, seq, icmp, sttl and drate and are the most significant for detecting malicious IoT traffic across multiple domain datasets.

As explained in section 3.8.3, this research uses the Shapley method to interpret the final model classification results. For a specific prediction, the Shapley values indicate how much each feature contributed to the final prediction.

A high value of a specific feature is colored in red, whereas a low value is colored in blue. A positive impact on the model means the feature is associated with a higher probability of belonging to normal traffic. In contrast, a negative effect on the model implies that the feature is associated with a lower probability of belonging to the normal traffic class. If the SHAP value is much closer to zero, we can say that the data point contributes very little to predictions.

Shapley visualizations shown below in Figures 5-1 illustrate how the Decision Tree model for the Bot-IoT is making its classification decisions. Similarly, Figures A-30, A-31, A-32, and A-33, which can be found in the Appendix, explain Shapley's interpretation of other dataset’s models.

Figure 5-1 shows that when the values of the *state_number* feature are high, it drives the model's prediction to be normal. The *state_number* feature in TCP packet flows often represents the numerical value assigned to indicate a TCP connection's specific state at a given time.

When the values of *ltime and tnbpdstip* features tend to be low, it drives the model's prediction to be attack. The "record last time" or "ltime" indicates explicitly the timestamp of the most recent packet in the TCP connection. It helps track the activity and timing of packets within the connection.

The "total number of bytes in destination IP" indicates the sum of the payload size (in bytes) of all the packets sent to a specific destination IP address within the TCP connection. By analyzing both features in TCP packet flows, a network administrator or security analyst can understand the timing and sequence of packet exchanges, measure

latency or delays, detect anomalies in packet transmission, and gain information about the volume of data transmitted to a particular IP address.

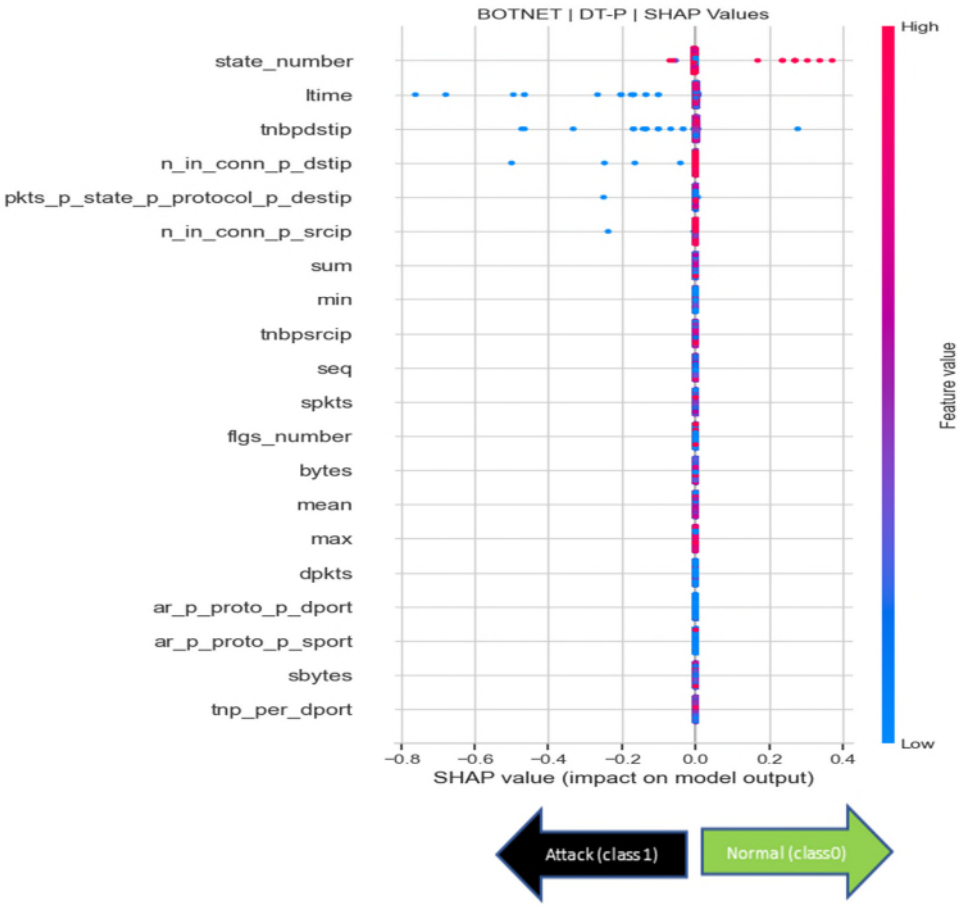


Figure 5-1. Shapley values interpretation of DT model for BOT-IoT dataset

5.1.2 Hypothesis 2

The second research question of this praxis tries to explore if machine learning models can be developed to implement classification tasks of IoT malicious traffic.

RQ2: Can a Machine Learning classification model be developed to detect IoT malicious traffic?

The second Hypothesis discussed in Chapter 1 assumes that an intrusion detection system based on machine learning can detect and classify malicious traffic in IoT networks with 95% accuracy or more.

H2: An ML classification model will detect malicious IoT traffic with at least 95% accuracy.

Table 5-1 below summarizes the results of the model performance metrics for all models examined in this praxis for the combined dataset representing the public 5G deployment scenario.

Table 5-1. Performance of all examined classifiers for the combined dataset

Model name	Accuracy	Precision	Recall	inference time_ms
BaggingClassifier	**0.999747793**	0.999747823	0.999747793	11.27890654
XGBClassifier	**0.999639705**	0.999639654	0.999639705	**0.376872241**
RF Classifier	**0.998955143**	0.998957799	0.998955143	**0.262770728**
RNN	**0.9984147**	0.998418453	0.9984147	15.29901763
DecisionTreeClassifier (CART)	**0.989263196**	0.989216571	0.989263196	**0.015457102**
GaussianNB	0.924662223	0.921056616	0.924662223	**0.339213121**
SVM (SVC)	0.864204648	0.877036551	0.864204648	11.71001915

As Table 5-1 illustrates, all seven models examined in this praxis for the combined dataset had a classification accuracy of more than 95.00% except SVM and Gaussian NB, which have 86% and 92% accuracy, respectively. We can also notice that RNN accuracy is 0.9984, which is lower than the Bagging classifier, XGB classifier, and RF Classifier, which have accuracies of 0.9997, 0.9996, and 0.9989, respectively.

Furthermore, all models examined in the combined dataset had a classification time of less than 0.5 milliseconds except RNN, Bagging, and SVM highlighted in red in the above table.

5.1.3 Hypothesis 3

The third research question of this praxis investigates the ML classifiers that are most suitable for deployment at the Edge layer of 5G networks, intending to achieve a classification time of less than 0.5 milliseconds and the highest possible accuracy.

RQ3: Which ML classifiers are best for detecting malicious IoT traffic in 5G networks?

The third Hypothesis discussed in Chapter 1 assumes that XGB, DT and RF are the best classification algorithms for detecting malicious IoT traffic in 5G networks.

H3: XGB, DT and RF are the best classification algorithms for detecting malicious IoT traffic in 5G networks.

Table 5-2 below summarizes the best suitable models for 5G deployments based on their accuracies and classification time in milliseconds. It's worth noting that they are all ensemble models based on decision trees. We can notice that all their respective classification time is below this praxis's threshold of 0.5 milliseconds, and all their accuracies are above 95% except for the ToN-IoT telemetry dataset.

DT (CART) showed persistent performance of being the fastest model among different types of datasets to reach a classification decision.

The observed DT-CART accuracy for classifying malicious traffic in the TON-IoT dataset is 67.8%, which is comparatively lower than the accuracy achieved by the RNN approach at 71.69% for the same dataset, as illustrated in Table 5-2 and Table 5-3.

Furthermore, in the TON-IoT dataset, the RNN exhibits a classification time of 4.9 milliseconds, whereas the DT-CART achieves a significantly shorter classification

time of 0.012 milliseconds. Notably, the TON-IoT dataset represents the sole dataset encompassing telemetry data.

Consequently, these findings suggest that the RNN model is more suitable for detecting malicious traffic based on telemetry data. However, its performance is unsuitable for cases where IoT devices are connected via a 5G network, given that its classification time exceeds the 0.5 milliseconds threshold employed in this study.

Therefore, for such scenarios, it is recommended to employ the DT-CART model. Conversely, the RNN model can be utilized if an alternative form of connectivity, such as Wi-Fi, is utilized.

Additionally, it is noteworthy that the accuracy results obtained by the RNN model, as presented in Table 5-3, fall within the same range as those achieved by ensemble models displayed in Table 5-2. However, irrespective of the dataset type, all classification times associated with the RNN model exceed those of the other models.

Throughout this praxis, the analysis of seven ML models revealed that DT-CART and XGB consistently emerged as the top-performing models outperforms the other models. They had the highest performance metrics, lowest false alarm rate, and shortest classification time per record highlighted in green in the below table, making it suitable for processing-constrained environments like 5G IoT devices connected to Edge layers. This is illustrated in Tables 9-11, 9-13, 9-15, and 5-2.

Table 5-2. Best suitable models for 5G deployments

	Best suitable models for 5G deployments	5G deployment mode	Accuracy	Classification time (ms)
BoT-IoT dataset	DT-CART	Private	0.99955	0.0008178
	XGB	Private	0.99982	0.0133569

	RF	Private	0.99991	0.2279665
HealthCare IoT dataset	**DT-CART**	Private	0.99987	**0.0105992**
	XGB	Private	0.99878	0.0847934
TON-IoT dataset	**DT-CART**	Private	**0.67803**	**0.0124651**
5G-NIDD dataset	**DT-CART**	Private	0.99808	**0.0210545**
	XGB	Private	0.99931	0.2610434
Combined dataset	**DT-CART**	Public	0.989263	**0.0154571**
	RF	Public	0.998955	0.2627707
	XGB	Public	0.999639	0.3768723

Table 5-3 below illustrates the neural network model examined in this praxis which is the RNN. As can be seen, all its classification time highlighted in red below, is well above the 0.5-millisecond threshold making it not suitable to be deployed on resource-constrained 5G Edge environments. Therefore, the results of this praxis confirm the validity of the three-hypothesis discussed in Chapter 1.

Table 5-3. Performance of the NN examined model

	NN examined model	Accuracy	Classification time (ms)
BoT-IoT dataset	RNN	0.999731	**1.523856**
HealthCare IoT dataset	RNN	0.998369	**13.27069**
TON-IoT dataset	RNN	**0.71695**	**4.93768**
5G-NIDD dataset	RNN	0.998184	**11.66191**
Combined dataset	RNN	0.998415	**15.29902**

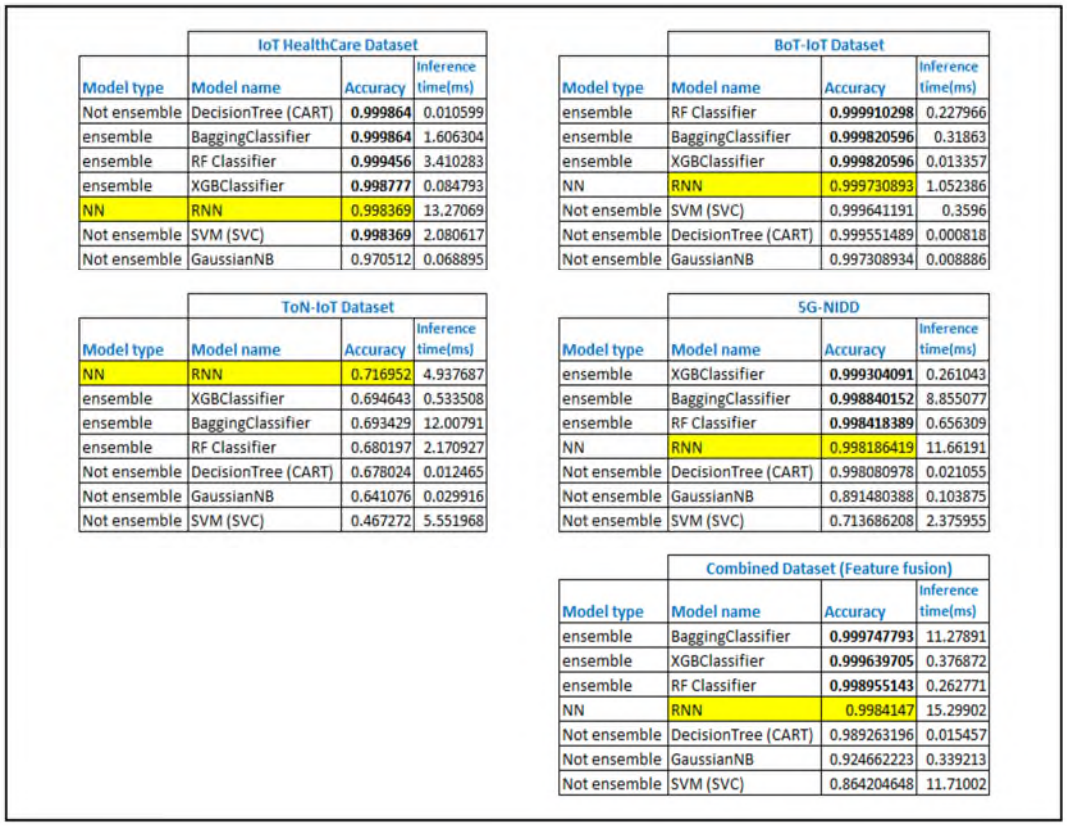
Finally, Figure 5-2 below illustrates that RNN accuracy was lower than all ensemble models in classifying data of the IoT-HealthCare dataset. This dataset has the

smallest size and simplest features to classify; hence ensemble models excelled in classifying it.

For the TON-IoT dataset, RNN has the most accuracy, with 71.69%, as this dataset has telemetry data and the most difficult features to relate to the output variable. In such kind of complex datasets, RNN excelled in classifying the data.

Figure 5-2 illustrates that RNN accuracies for the other datasets, BoT-IoT,5G-NIDD, and the Combined dataset, were slightly lower than XGB, Bagging, and RF.

It is worth noting that, as per Table 3-1, we used one hidden layer for RNN simplicity, which still impacted the classification time. So, if we want to increase the accuracy of RNN, one of the methods is to increase the number of hidden layers, but that will likely increase the classification time, which is not desirable for this praxis goal of finding the best models to use in 5G networks.



IoT HealthCare Dataset

Model type	Model name	Accuracy	Inference time(ms)
Not ensemble	DecisionTree (CART)	**0.999864**	0.010599
ensemble	BaggingClassifier	**0.999864**	1.606304
ensemble	RF Classifier	**0.999456**	3.410283
ensemble	XGBClassifier	**0.998777**	0.084793
NN	RNN	0.998369	13.27069
Not ensemble	SVM (SVC)	**0.998369**	2.080617
Not ensemble	GaussianNB	0.970512	0.068895

BoT-IoT Dataset

Model type	Model name	Accuracy	Inference time(ms)
ensemble	RF Classifier	**0.999910298**	0.227966
ensemble	BaggingClassifier	**0.999820596**	0.31863
ensemble	XGBClassifier	**0.999820596**	0.013357
NN	RNN	0.999730893	1.052386
Not ensemble	SVM (SVC)	0.999641191	0.3596
Not ensemble	DecisionTree (CART)	0.999551489	0.000818
Not ensemble	GaussianNB	0.997308934	0.008886

ToN-IoT Dataset

Model type	Model name	Accuracy	Inference time(ms)
NN	RNN	0.716952	4.937687
ensemble	XGBClassifier	0.694643	0.533508
ensemble	BaggingClassifier	0.693429	12.00791
ensemble	RF Classifier	0.680197	2.170927
Not ensemble	DecisionTree (CART)	0.678024	0.012465
Not ensemble	GaussianNB	0.641076	0.029916
Not ensemble	SVM (SVC)	0.467272	5.551968

5G-NIDD

Model type	Model name	Accuracy	Inference time(ms)
ensemble	XGBClassifier	**0.999304091**	0.261043
ensemble	BaggingClassifier	**0.998840152**	8.855077
ensemble	RF Classifier	**0.998418389**	0.656309
NN	RNN	0.998186419	11.66191
Not ensemble	DecisionTree (CART)	0.998080978	0.021055
Not ensemble	GaussianNB	0.891480388	0.103875
Not ensemble	SVM (SVC)	0.713686208	2.375955

Combined Dataset (Feature fusion)

Model type	Model name	Accuracy	Inference time(ms)
ensemble	BaggingClassifier	**0.999747793**	11.27891
ensemble	XGBClassifier	**0.999639705**	0.376872
ensemble	RF Classifier	**0.998955143**	0.262771
NN	RNN	0.9984147	15.29902
Not ensemble	DecisionTree (CART)	0.989263196	0.015457
Not ensemble	GaussianNB	0.924662223	0.339213
Not ensemble	SVM (SVC)	0.864204648	11.71002

Figure 5-2. RNN accuracy compared to other models.

5.2 Contributions to Body of Knowledge

To address the identified limitations in existing research outlined in the literature review conducted in Chapter 2, this praxis addresses the need for comprehensive multi-domain IoT intrusion detection by carefully selecting four datasets encompassing network attacks, application layer attacks, and insider attacks specifically targeting the 5G edge layer. This dataset selection contributes to developing a robust and encompassing multi-domain IoT intrusion detection model.

This research's primary finding is the identification of XGB (Extreme Gradient Boosting), DT (Decision Trees), and RF (Random Forest) as the most suitable classification algorithms for detecting malicious IoT traffic in 5G networks.

Additionally, this praxis recognizes the insufficient incorporation of the NIST Zero-Trust concept in the design of machine-learning-based IDS. To bridge this gap, the praxis integrates the Zero-Trust methodology into the proposed framework, presenting a proposal for a novel and holistic dynamic monitoring using an ML-based zero-trust framework.

Furthermore, this praxis goes beyond traditional machine learning performance metrics to evaluate the suitability of models for power-constrained environments, such as a 5G IoT network connected to an Edge layer node. It introduces classification time per record as an additional metric. This metric provides insights into the efficiency and practicality of the models, considering the time taken for classification on a per-record basis.

In conclusion, this praxis addresses the limitations identified in the existing research by carefully selecting diverse datasets, identifying DT, XGB and RF as most suitable for detecting IoT malicious traffic in 5G networks and incorporating classification time per record as a relevant performance metric. These contributions enhance the understanding and applicability of multi-domain IoT intrusion detection, particularly in the context of 5G networks and the Edge layer.

5.3 Recommendations for Future Research

In conclusion, this praxis has provided valuable insights into the binary classification of malicious traffic across various IoT industries. However, there are several avenues for future

research that can expand and enhance the findings of this research.

Firstly, extending the scope of this praxis to include multi-class classification attacks would be a fruitful direction. Investigating the detection and classification of multiple types of attacks would provide a more comprehensive understanding of IoT network security.

Additionally, exploring the potential of hybrid models by combining different machine-learning techniques could lead to improved accuracy and speed in IoT intrusion detection systems. Integrating decision tree-based models like DT-CART with boosting techniques such as XGBoost could leverage the strengths of each model and potentially enhance the overall performance of intrusion detection systems.

Moreover, to further advance the field of 5G research, future investigations into the use of federated learning on multiple edge data centers distributed among different cities would be valuable.

Researching other modules of the Zero-Trust model, like the Policy Administrator and the Policy Enforcement Point, would enhance the literature on this area.

By pursuing these avenues for future research, researchers can contribute to the continuous development and improvement of intrusion detection techniques in IoT networks. These directions will help address the challenges and complexities inherent in securing IoT systems, particularly in the context of evolving technologies such as 5G and distributed edge computing.

References

3GPP, 2018 “System architecture for the 5g system; stage 2”, release-15, v. 15.1.0. In TS 23.501.

Agarwal, A., Sharma, P., Alshehri, M., Mohamed, A. A., & Alfarraj, O. (2021). Classification model for accuracy and intrusion detection using machine learning

Deo, R. C. (2015). Machine learning in medicine. Circulation, 132(20), 1920-1930.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. Machine learning, 6, 37-66.

Ahmad, I., Kumar, T., Liyanage, M., Okwuibe, J., Ylianttila, M., & Gurtov, A. (2018). Overview of 5G security challenges and solutions. IEEE Communications Standards Magazine, 2(1), 36-43.

Ahmad, Ijaz & Shahabuddin, Shahriar & Kumar, Tanesh & Okwuibe, Jude & Gurtov, Andrei & Ylianttila, Mika. (2019). Security for 5G and Beyond. IEEE Communications Surveys & Tutorials. PP. 10.1109/COMST.2019.2916180.

Akhil Gupta, (Student Member, IEEE), Rakesh Kumar Jha, (Senior Member, IEEE), "A Survey of 5G Network: Architecture and Emerging Technologies", IEEE Access, Volume 3,2015.

Akpakwu, Silva, B. J., Hancke, G. P., & Abu-Mahfouz, A. M. (2018). A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. IEEE Access, 6, 3619–3647. https://doi.org/10.1109/ACCESS.2017.2779844

Alam, M. S., & Vuong, S. T. (2013, August). Random forest classification for detecting android malware. In 2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing (pp. 663-669). IEEE.

Alazab, M., Venkatraman, S., Watters, P. A., & Alazab, M. (2011). Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures. AusDM, 11, 171-182.

Al-Juboori, S. A. M., Hazzaa, F., Jabbar, Z. S., Salih, S., & Gheni, H. M. (2023). Man-in-the-middle and denial of service attacks detection using machine learning algorithms. Bulletin of Electrical Engineering and Informatics, 12(1), 418-426.

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. Ieee Access, 8, 165130-165150.

Alshari, H., Saleh, A. Y., & Odabaş, A. (2021). Comparison of gradient boosting decision tree algorithms for CPU performance. Journal of Institue Of Science and Technology, 37(1), 157-168.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., & Zhou,Y. (2017). Understanding the mirai botnet. In 26th {USENIX} security symposium ({USENIX} Security 17) (pp. 1093-1110).

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., ... & Zhou, Y. (2017). Understanding the mirai botnet. In 26th {USENIX} security symposium ({USENIX} Security 17) (pp. 1093-1110).

Arunachalam, S., Kumar, S., Kshatriya, H., & Patil, M. (2018). Analyzing 5G: prospects of future technological advancements in mobile. IOSR Journal of Engineering (IOSRJEN), 1, 2278-8719.

Aydin, Z. E., & Ozturk, Z. K. (2021). Performance analysis of XGBoost classifier with missing data. Manchester Journal of Artificial Intelligence and Applied Sciences (MJAIAS), 2(02), 2021.

Baker, J. & Waldron, K. (2020).5G AND ZERO TRUST NETWORKS. https://www.jstor.org/stable/resrep27016

Bakkers, J. H. (2020). Digital Transformation requires network transformation. IDC Analyze the Future. https://www.orange-business.com/sites/default/files/digital-transformation-requires-network-transformation_may2020.pdf

Banafa, A. (2018). Secure and smart internet of things (IoT): Using blockchain and AI. River Publishers.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

Bhattacharjya, S. (2020). A novel zerotrust framework to secure iot communications. Ph.D. thesis, University of Kansas.

Bianchi, G., Biton, E., Blefari-Melazzi, N., Borges, I., Chiaraviglio, L., Ramos, P.D., Eardley, P., Fontes, F., Mcgrath, M.J., Natarianni, L., Niculescu, D., Parada, C., Popovici, M., Riccobene, V., Salsano, S., Sayadi, B., Thomson, J., Tselios, C., & Tsolis, G.K. (2016). Superfluidity: a flexible functional architecture for 5G networks. Transactions on Emerging Telecommunications Technologies, 27, 1178 - 1186.

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Boufeloussen, O. (2020, September 5). Simple explanation of recurrent neural network (RNN). Medium. Retrieved March 30, 2023, from https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363 Al-Aidaroos, K. M., Bakar, A. A., & Othman, Z. (2010, March).

Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

Breiman. (1996). Bagging predictors. Machine Learning, 24(2), 123–140. https://doi.org/10.1007/BF00058655

Bro, R., & Smilde, A. K. (2003). Centering and scaling in component analysis. Journal of chemometrics, 17(1), 16-33.

Brownlee, J. (2019). How to choose a feature selection method for machine learning. Machine Learning Mastery, 10.

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. Neurocomputing, 300, 70-79.

Cessie, S. L., & Houwelingen, J. V. (1992). Ridge estimators in logistic regression. Journal of the Royal Statistical Society Series C: Applied Statistics, 41(1), 191-201.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Chiang, M., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. IEEE Internet of things journal, 3(6), 854-864.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Cirani, S., Ferrari, G., Mancin, M., & Picone, M. (2018). Virtual replication of iot hubs in the cloud: A flexible approach to smart object management. Journal of Sensor and Actuator Networks, 7(2), 16.

D. Rupprecht and A. Dabrowski and T. Holz and E. Weippl and C. Ppper, "On Security Research Towards Future Mobile Network Generations," IEEE Communications Surveys Tutorials, vol. 20, no. 3, pp. 2518–2542, 2018.

D.Sattar and A. Matrawy (2019).Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices.2019 IEEE Conference on Communications and Network Security (CNS), Washington DC, DC, USA, 2019, pp. 82-90. doi: 10.1109/CNS.2019.8802852

Das, A., Patterson, S., & Wittie, M. (2018, December). Edgebench: Benchmarking edge computing platforms. In 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion) (pp. 175-180). IEEE.

Deshmukh, R. V., & Devadkar, K. K. (2015). Understanding DDoS attack & its effect in cloud environment. Procedia Computer Science, 49, 202-210.

Dhar, S., & Bose, I. (2020). Journal of Organizational Computing and Electronic Commerce, 1–17.

Dissanayake, K., & Md Johar, M. G. (2021). Comparative study on heart disease prediction using feature selection techniques on classification algorithms. Applied Computational Intelligence and Soft Computing, 2021, 1-17.

Faisal H., Syed A., Ghalib A. Shah, Ivan P., Ubaid U., Farrukh S., Nuno M., Eftim Z. (2021). IoT Healthcare Security Dataset. IEEE Dataport. https://dx.doi.org/10.21227/9w13-2t13

Federici, F., Martintoni, D., & Senni, V. (2023). A Zero-Trust Architecture for Remote Access in Industrial IoT Infrastructures. Electronics, 12(3), 566.

Ferrag, Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. IEEE Access, 10, 40281–40306. https://doi.org/10.1109/ACCESS.2022.3165809

Fischer, E. A. (2014). Federal laws relating to cybersecurity: Overview of major issues, current laws, and proposed legislation.

Freund, Y., & Schapire, R. E. (1996, July). Experiments with a new boosting algorithm. In icml (Vol. 96, pp. 148-156).

Fryer, D., Strumke, I., & Nguyen, H. (2021). Shapley values for feature selection: The good, the bad, and the axioms. *IEEE Access*, 9, 144352-144360. https://doi.org/10.1109/access.2021.3119110

G. Arfaoui and P. Bisson and R. Blom and R. Borgaonkar and H. Englund and E. Flix and F. Klaedtke and P. K. Nakarmi and M. Nslund and P. OHanlon and J. Papay and J. Suomalainen and M. Surridge and J. P. Wary and A. Zahariev, “A Security

Architecture for 5G Networks," IEEE Access, vol. 6, no. , pp. 22 466–22 479, 2018.

Gadre, M., & Deoskar, A. (2020). Industry 4.0–digital transformation, challenges, and benefits. International Journal of Future Generation Communication and Networking, 13(2), 139-149.

Gaurav, A., Gupta, B. B., & Panigrahi, P. K. (2022). A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system. Enterprise Information Systems, pp. 1–25.

Ghorbani, H., Mohammadzadeh, M. S., & Ahmadzadegan, M. H. (2020). DDoS Attacks on the IoT Network with the Emergence of 5G. In 2020 International Conference on Technology and Entrepreneurship-Virtual (ICTE-V) (pp. 1-5). IEEE.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

H. Ghorbani, M. Izadyar, H. A. Deilami, M. Ahmadzadegan. (2018)

H. Kim, H. Chang, J. Suh, and T. Shon. (2016) A study on device security in IoT convergence.International Conference on Industrial Engineering, Management Science and Application (ICIMSA), pages 1– 4

Haarman, B. C. B., Riemersma-Van der Lek, R. F., Nolen, W. A., Mendes, R., Drexhage, H. A., & Burger, H. (2015). Feature-expression heat maps–A new visual method to explore complex associations between two variable sets. Journal of biomedical informatics, 53, 156-161.

HaddadPajouh, H., Khayami, R., Dehghantanha, A., Choo, K. K. R., & Parizi, R. M. (2020). AI4SAFE-IoT: An AI-powered secure architecture for edge layer of Internet of things. Neural Computing and Applications, 32, 16119-16133.

Han, J., Pei, J., & Kamber, M. (1999). Data mining: concepts and techniques. 2011.

Han, Y., Wu, J., Zhai, B., Pan, Y., Huang, G., Wu, L., & Zeng, W. (2019). Coupling a bat algorithm with xgboost to estimate reference evapotranspiration in the arid and semiarid regions of china. Advances in Meteorology, 2019, (1-16.)

He, L., Yan, Z., & Atiquzzaman, M. (2018). LTE/LTE-A network security data collection and analysis for security measurement: a survey. IEEE Access, 6, 4220-4242.

He, Qi & Ju, Yunxia & Wang, Jianguo & Zhao, Gang & Qin, Haiyong & Zhao, Kai & Zhou, Yilan & Li, Min & Dong, Qi. (2018). Network Slicing to Enable Resilience and High Availability in 5G Mobile Telecommunications. MATEC Web of Conferences. 246. 03028. 10.1051/matecconf/201824603028.

HIPAA Journal. (2022, December 5). 2019 cost of a data breach study reveals increase in U.S. Healthcare Data Breach Costs. HIPAA Journal. Retrieved February 27, 2023, from https://www.hipaajournal.com/2019-cost-of-a-data-breach-study-healthcare-data-breach-costs/

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hsu, R. H., Lee, J., Quek, T. Q., & Chen, J. C. (2018). Reconfigurable security: Edge-computing-based framework for IoT. IEEE Network, 32(5), 92-99.

Huda, S., Abawajy, J., Abdollahian, M., Islam, R., & Yearwood, J. (2017). A fast malware feature selection approach using a hybrid of multi-linear and stepwise binary logistic regression. Concurrency and Computation: Practice and Experience, 29(23), e3912.

IBM (2022) IBM 1 Cost of Data Breach Report. (n.d.). Retrieved February 27, 2023, from https://www.ibm.com/account/reg/us-en/signup?formid=urx-51643&utm_content=SRCWW&p1=Search&p4=43700072379140729&p5=p&&msclkid=ae3bc5e735f910481fcba61a14d982a7&gclid=ae3bc5e735f910481fcba61a14d982a7&gclsrc=3p.ds

Investigating NFC technology from the perspective of security, analysis of attacks, and existing risk.2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), pages 1083– 1087. IEEE, 2015

Islam, N., Farhin, F., Sultana, I., Kaiser, M. S., Rahman, M. S., Mahmud, M., & Cho, G. H. (2021). Towards machine learning based intrusion detection in IoT networks. Comput. Mater. Contin, 69(2), 1801-1821.

ITU-R Recommendations M. 2083. (2015) M.2083;IMT Vision - framework and overall objectives of the future development of IMT for 2020 and beyond. https://www.itu.int/rec/R-REC-M.2083

J. Cao, M. Ma, H. Li, R. Ma, Y. Sun, P. Yu, and L. Xiong, "A survey on security aspects for 3GPP 5G networks," IEEE communications surveys& tutorials, vol. 22, no. 1, pp. 170–195, 2019

J. Cao, M. Ma, H. Li, R. Ma, Y. Sun, P. Yu, and L. Xiong, "A survey on security aspects for 3GPP 5G networks," IEEE communications surveys& tutorials, vol. 22, no. 1, pp. 170–195, 2019

J. Prados-Garzon and T. Taleb, "Asynchronous Time-Sensitive Networking for 5G Backhauling", *IEEE Network*, vol. 35, no. 2, pp. 144–51, Mar./Apr. 2021.

Jiawei Han. (2011). Data Mining: Concepts and Techniques. Elsevier Science. https://doi.org/10.1016/C2009-0-61819-5

Johann, M. & Nina, S. (2022, May 3). Edge reduces network latency. 5G Technology World. Retrieved February 20, 2023, from https://www.5gtechnologyworld.com/orchestration-at-the-edge-reduces-network-latency

Jyothsna, V. V. R. P. V., Prasad, R., & Prasad, K. M. (2011). A review of anomaly based intrusion detection systems. International Journal of Computer Applications, 28(7), 26-35.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. Neural computation, 13(3), 637-649.

Khan, M. Y., Qayoom, A., Nizami, M. S., Siddiqui, M. S., Wasi, S., & Raazi, S. M. K. U. R. (2021). Automated prediction of Good Dictionary EXamples (GDEX): a comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques. Complexity, 2021, 1-18.

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., & Ahmed, A. (2019). Edge computing: A survey. Future Generation Computer Systems, 97, 219-235.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1), 1-22.

Kim, C. (2020, February 13). 5G and massive IOT: Legacy technologies will bridge the gap for now. Omdia. https://omdia.tech.informa.com/OM004695/5G-and-Massive-IoT-legacy-technologies-will-bridge-the-gap-for-now

Kim, T. H., Ramos, C., & Mohammed, S. (2017). Smart city and IoT. Future Generation Computer Systems, 76, 159-162.

Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. Computer, 50(7), 80-84.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems, 100*, 779-796.

Langley, P. (1995). John, GH: Estimating continuous distributions in bayesian classifiers. In Proc. Uncertainty in Artificial Intelligence.

Leevy, J. L., Hancock, J., Zuech, R., & Khoshgoftaar, T. M. (2020, October). Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI) (pp. 190-197). IEEE.

Li, G., Yan, Z., Fu, Y., & Chen, H. (2018). Data fusion for network intrusion detection: a review. Security and Communication Networks, 2018.

Li, S. (2020). EAI Endorsed Transactions on Internet of Things, 5(20).

Li, S., Iqbal, M. & Saxena, N. Future Industry Internet of Things with Zero-trust Security. Inf Syst Front (2022). https://doi.org/10.1007/s10796-021-10199-5

Li, S., Iqbal, M. & Saxena, N. Future Industry Internet of Things with Zero-trust Security. Inf Syst Front (2022). https://doi.org/10.1007/s10796-021-10199-5

Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. Information systems frontiers, 17, 243-259.

Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059.

Lorenzo C. (2017)– 3GPP starting with the basics: Qualcomm. Wireless Technology & Innovation. (n.d.). Retrieved February 27, 2023, from https://www.qualcomm.com/news/onq/2017/08/understanding-3gpp-starting-basics

M. A. Ferrag, L. Maglaras, A. Argyriou, D. Kosmanos, and H. Janicke, “Security for 4G and 5G Cellular Networks: A survey of existing authentication and privacy-preserving schemes,” Journal of Network and Computer Applications, vol. 101, pp. 55 – 82, 2018. Online available: https://bit.ly/2T7JgJN

M. A. Javed & S. khan Niazi (2019). 5G Security Artifacts (DoS / DDoS and Authentication, International Conference on Communication Technologies (ComTech).

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. IEEE communications surveys & tutorials, 19(4), 2322-2358.

Margolin, A. A., Bilal, E., Huang, E., Norman, T. C., Ottestad, L., Mecham, B. H., ... & Børresen-Dale, A. L. (2013). Systematic analysis of challenge-driven improvements in molecular prognostic models for breast cancer. Science translational medicine, 5(181), approach. *PeerJ Computer Science*, *7*, e437.

Massive DDOS occurrence investigation in future IoT devices. International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pages 695–698, Nov 2018.

McIntosh, T., Jang-Jaccard, J., Watters, P., & Susnjak, T. (2019). The inadequacy of entropy-based ransomware detection. In Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26 (pp. 181-189). Springer International Publishing.

Mehraj, S., & Banday, M. T. (2020, January). Establishing a zero trust strategy in cloud computing environment. In 2020 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). IEEE.

Moganedi, S. (2018, June). Undetectable data breach in iot: Healthcare data at risk. In 17th European Conference on Cyber Warfare and Security (Vol. 2, p. 296).

Montori, F., Bedogni, L., Di Felice, M., & Bononi, L. (2018). Machine-to-machine wireless communication technologies for the Internet of Things: Taxonomy, comparison and open issues. Pervasive and Mobile Computing, 50, 56-81.

Mukherjee, S., & Sharma, N. (2012). Intrusion detection using naive Bayes classifier with feature reduction. Procedia Technology, 4, 119-128.

Munther, A., Alalousi, A., Nizam, S., Othman, R. R., & Anbar, M. (2014, August). Network traffic classification—a comparative study of two common decision tree methods: C4. 5 and random forest. In 2014 2nd International Conference on Electronic Design (ICED) (pp. 210-214). IEEE.

Naive Bayes variants in classification learning. In 2010 international conference on information retrieval & knowledge management (CAMP) (pp. 276-281). IEEE.

NIST SP 800-207. (2020). (2020, August 11). *Zero trust architecture*. CSRC. Retrieved March 25, 2023, from https://csrc.nist.gov/publications/detail/sp/800-207/final

Noohani, M. Z., & Magsi, K. U. (2020). A review of 5G technology: Architecture, security and wide applications. International Research Journal of Engineering and Technology (IRJET), 7(5), 34.

Nour, M. (2021). TheTON_IoT Datasets | UNSW Research. Retrieved March 29, 2023, from https://research.unsw.edu.au/projects/toniot-datasets

Panwar, N., Sharma, S., & Singh, A. K. (2016). A survey on 5G: The next generation of mobile communication. *Physical Communication*, *18*, 64-84.

Papadonikolakis, & Bouganis, C. (2012). Novel Cascade FPGA Accelerator for Support Vector Machines Classification. IEEE Transaction on Neural Networks and Learning Systems, 23(7), 1040–1052. https://doi.org/10.1109/TNNLS.2012.2196446

Parian, C., Guldimann, T., & Bhatia, S. (2020). Fooling the master: exploiting weaknesses in the modbus protocol. Procedia Computer Science, 171, 2453-2458.

Pedreira, V., Barros, D., & Pinto, P. (2021). A review of attacks, vulnerabilities, and defenses in industry 4.0 with new challenges on data sovereignty ahead. Sensors, 21(15), 5189.

Ponemon (2015) Cost of Data Breach Study: Global Analysis. from: https://www.ponemon.org/local/upload/file/2015%20Global%20CODB%20FINAL%203%20copy.pdf

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In Machine Learning: ECML-93: European Conference on Machine Learning Vienna, Austria, April 5–7, 1993 Proceedings 6 (pp. 1-20). Springer Berlin Heidelberg.

Radanliev, P., De Roure, D., Cannady, S., Montalvo, R. M., Nicolescu, R., & Huth, M. (2018). Economic impact of IoT cyber risk-analysing past and present to predict the future developments in IoT risk analysis and IoT cyber insurance.

Ramezanpour, & Jagannath, J. (2021). Intelligent Zero Trust Architecture for 5G/6G Networks: Principles, Challenges, and the Role of Machine Learning in the context of O-RAN. https://doi.org/10.48550/arxiv.2105.01478

Ravalli Kolli, Shweta Malle, Shreya Shetty, Sunanda Dixit, “Review on 5G wireless technology”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 12, December2016.

Resende, P. A. A., & Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR), 51(3), 1-36.

Reshma S. Sapakal , Ms. Sonali S. Kadam, “5G Mobile technology” in International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 2, February2013.

Roman, R., Lopez, J., & Mambo, M. (2018). Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, 78, 680-698.

Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. Computer networks, 57(10), 2266-2279.

S. Mahdi Shariati, A. Abouzarjomehri, M. Ahmadzadegan. (2015)

Sahni, & Kaur, A. (2022). A Systematic Literature Review on 5G Security. https://doi.org/10.48550/arxiv.2212.03299

Samarakoon, S., Siriwardhana, Y., Porambage, P., Liyanage, M., Chang, S. Y., Kim, J., ... & Ylianttila, M. (2022). 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network. *arXiv e-prints*, arXiv-2212.

Sanchez-Gomez, J., Garcia-Carrillo, D., Marin-Perez, R., & Skarmeta, A. F. (2020). Secure authentication and credential establishment in narrowband IoT and 5G. Sensors, 20(3), 882.

Sarker, I. H., Kayes, A. S. M., & Watters, P. (2019). Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage. Journal of Big Data, 6(1), 1-28.

Sathiya M, R. Gowthami, G. Karpagam3, B. Saranya, U. Suganya, “Cellular And Network Architecture For 5g Wireless Communication Networks In Mobile

Technology", International Journal of Technical Research and Applications, Volume 3, Issue 2 (Mar-Apr2015).

Seh, A. H., Zarour, M., Alenezi, M., Sarkar, A. K., Agrawal, A., Kumar, R., & Ahmad Khan, R. (2020, May). Healthcare data breaches: insights and implications. In Healthcare (Vol. 8, No. 2, p. 133). MDPI.

Sehan, S., Siriwardhana, Y., Porambage, P., Liyanage, M., Chang, S. Y., Kim, J., ... & Ylianttila, M. (2022). 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network. IEEE Dataport. https://dx.doi.org/10.21227/xtep-hv36

Sha, K., Errabelly, R., Wei, W., Yang, T. A., & Wang, Z. (2017, May). Edgesec: Design of an edge layer security service to enhance iot security. In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC) (pp. 81-88). IEEE.

Sharma, Satija, S., & Bhushan, B. (2019). Unifying Blockchian and IoT:Security Requirements, Challenges, Applications and Future Trends. 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2019-, 341–346. https://doi.org/10.1109/ICCCIS48478.2019.8974552

Silva, B. J., Hancke, G. P., & Abu-Mahfouz, A. M. (2017). A survey on 5G networks for the Internet of Things: Communication technologies and challenges. IEEE access, 6, 3619-3647.

Sklearn model selection (n.d.). Retrieved March 30, 2023, from https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Stańczyk, U. (2015). Feature evaluation by filter, wrapper, and embedded approaches. Feature selection for data and pattern recognition, 29-44.

Stankovic, J. A. (2014). Research directions for the internet of things. IEEE internet of things journal, 1(1), 3-9.

Sudrajat, R., Irianingsih, I., & Krisnawan, D. (2017). Analysis of data mining classification by comparison of C4. 5 and ID algorithms. In IOP Conference Series: Materials Science and Engineering (Vol. 166, No. 1, p. 012031). IOP Publishing.

Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. Y., & Xiang, Y. (2018). Data-driven cybersecurity incident prediction: A survey. IEEE communications surveys & tutorials, 21(2), 1744-1772.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., & Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, *19*(3), 1657-1681.

Tange, K., De Donno, M., Fafoutis, X., & Dragoni, N. (2020). A systematic survey of industrial Internet of Things security: Requirements and fog computing opportunities. IEEE Communications Surveys & Tutorials, 22(4), 2489-2520.

Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. (2015). Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, pages 1–7.

United Nations. (2018, May 16). 68% of the world population projected to live in urban areas by 2050, says UN. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

Vaccari I, Chiola G, Aiello M, Mongelli M, Cambiaso E. MQTTset, a New Dataset for Machine Learning Techniques on MQTT. Sensors (Basel). 2020 Nov 18;20(22):6578. doi: 10.3390/s20226578. PMID: 33217936; PMCID: PMC7698741.

Wikina, S. B. (2014). What caused the breach? An examination of use of information technology and health data breaches. Perspectives in health information management, 11

Wu, Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1), 1–37. https://doi.org/10.1007/s10115-007-01142

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, 84, 25-37.

Zeb, S., Mahmood, A., Hassan, S. A., Piran, M. J., Gidlund, M., & Guizani, M. (2022). Industrial digital twins at the nexus of nextG wireless networks and computational intelligence: A survey. Journal of Network and Computer Applications, 103309.

Appendix A

Table A-1. BoT-IoT Dataset Feature list

Feature	Description
Dpkts	Destination-to-source packet count
Sbytes	Source-to-destination byte count
Dbytes	Destination-to-source byte count
Rate	Total packets per second in transaction
Srate	Source-to-destination packets per second
Drate	Destination-to-source packets per second
TnBPSrcIP	Total Number of bytes per source IP
TnBPDstIP	Total Number of bytes per Destination IP
TnP_PSrcIP	Total Number of packets per source IP
TnP_PDstIP	Total Number of packets per Destination IP
TnP_PerProto	Total Number of packets per protocol
TnP_Per_Dport	Total Number of packets per dport
AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP (calculated by pkts/dur)
AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP
AR_P_Proto_P_Sport	Average rate per protocol per sport
AR_P_Proto_P_Dport	Average rate per protocol per dport
Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP
Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP
Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Category	Traffic category
Subcategory	Traffic subcategory
RKSeqID	Row Identifier
Stime	Record start time
Flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
Saddr	Source IP address
Sport	Source port number
Daddr	Destination IP address
Dport	Destination port number
Pkts	Total count of packets in transaction
Bytes	Total number of bytes in transaction
State	Transaction state
state_number	Numerical representation of feature state
ltime	Record last time
Seq	Argus sequence number
Dur	Record total duration
Mean	Average duration of aggregated records
Stddev	Standard deviation of aggregated records
Sum	Total duration of aggregated records
Min	Minimum duration of aggregated records
Max	Maximum duration of aggregated records
Spkts	Source-to-destination packet count

Table A-2. IoT HealthCare Dataset Feature list

Feature	Description
tcp.flags	TCP flags
tcp.time.delta	Time TCP stream
tcp.len	TCP Segment Len
mqtt.conack.flags	Acknowledge Flags
mqtt.conack.val	Return Code
mqtt.conflag.passwd	Password Flag
mqtt.conflag.qos	QoS Level
mqtt.conflag.reserved	(Reserved)
mqtt.conflag.retain	Will Retain
mqtt.conflag.willflag	Will Flag
mqtt.conflags	Connect Flags
mqtt.dupflag	DUP Flag
mqtt.hdrflags	Header Flags
mqtt.kalive	Keep Alive
mqtt.len	Msg Len
mqtt.msg	Message
mqtt.msgtype	Message Type
mqtt.qos	QoS Level
mqtt.retain	Retain
mqtt.ver	Version
mqtt.willmsg_len	Will Message Length
mqtt.topic.len	Topic length
mqtt.topic	Topic
frame.time.delta	Time delta
frame.time.relative	Relative time
frame.len	Frame length
ip.src	Source IP
ip.dst	Destination IP
tcp.srcport	TCP source port
tcp.dstport	TCP destination port
tcp.ack	TCP acknowledge
tcp.connection.fin	TCP connection finish
tcp.connection.rst	TCP connection reset
tcp.connection.sack	TCP connection source acknowledge
tcp.connection.syn	TCP connection sync
tcp.flags.ack	tcp flag acknowledge
tcp.flags.fin	TCP flags finish
tcp.flags.push	TCP flags push
tcp.flags.reset	TCP flags reset
tcp.flags.syn	TCP flags sync
tcp.flags.urg	TCP flags urgent
tcp.hdr.len	TCP header length
tcp.payload	TCP payload
tcp.pdu.size	TCP PDU size
tcp.window.size.value	TCP window size value
tcp.checksum	TCP checksum
ip.proto	IP protocol
ip.ttl	IP time to live TTL
mqtt.clientid.len	MQTT Client Id length
mqtt.clientid	MQTT Client Id
class	sensor monitoring type
label	Class: 0 for Normal traffic, 1 for Attack

Table A-3. TON-IOT Dataset Feature list

Feature	Description
ts	time stamp of sensor reading data
fridge_temperature	temperature measurement of fridge sensor
label	0' indicates normal, '1' indicates attack
latitude	latitude value of GPS tracker sensor
longitude	longitude value of GPS tracker sensor
FC1_Read_Input_Register	Modbus function code responsible of reading input register
FC2_Read_Discrete_Value	Modbus function code responsible of reading discrete value
FC3_Read_Holding_Register	Modbus function code responsible of reading holding register
FC4_Read_Coil	Modbus function code responsible of readinga coil
motion_status	status of motion sensor, '1' is On and '0' is off
current_temperature	current reading of thermostat temperature
thermostat_status	status of thermostat sensor is either on or off
temperature	temperature reading of weather sensor
pressure	pressure reading of weather sensor
humidity	humidity reading of weather sensor
date	date of logging sensor telemetry data
time	time of logging sensor telemetry data
temp_condition	temperature condition of fridge sensor
type	type of attack subcategory
door_state	State of the door sensor open or closed
sphone_signal	state of receiving the door signal on a phone where the signal is true or false
light_status	status of light sensor on or off

Table A-4. 5G-NIDD Dataset Feature list

Feature	Description
acc	accepted connection
ackdat	TCP connection setup time, the time between the SYN_ACK and the ACK packets
af11	DSCP values which is related to QoS mapping
af12	DSCP values which is related to QoS mapping
af41	DSCP values which is related to QoS mapping
arp	DSCP values which is related to QoS mapping
con	DSCP values which is related to QoS mapping
cs0	DSCP values which is related to QoS mapping
cs4	DSCP values which is related to QoS mapping
cs6	DSCP values which is related to QoS mapping
cs7	DSCP values which is related to QoS mapping
dhops	stimate of number of IP hops from destination to this point
dmeanpktsz	Mean of the flow packet size transmitted by the destination
dstbytes	dst -> src transaction bytes
dstgap	destination bytes missing in the data stream
dstload	destination bits per second
dstloss	destination pkts retransmitted or dropped
dstpkts	dst -> src packet count
dstrate	destination pkts per second
dsttcpbase	destination TCP base sequence number
dstwin	destination TCP window advertisement
dtos	destination TOS byte value
dttl	dst -> src TTL value
dur	record total duration
dvid	destination VLAN identifier
e_	flgs' feature that was obtained for each flow
eco	Echo Request
ef	Expedited Forwarding." It is a type of Quality of Service (QoS) mechanism that can be used to prioritize network traffic and ensure that certain types of data are transmitted with minimal delay and jitter
fin	Finish flag in TCP
icmp	icmp ping protocol
int	interpacket
ipv6_icmp	icmp ping type is IPV6
label	Normal or attack label
Attack type	Attack type
Attack tool	Attack tool
llc	encapsulation type
lldp	Link Layer Discovery Protocol, open standard protocol used for discovering information about directly connected network devices.
load	bits per second
loss	pkts retransmitted or dropped
max	maximum duration of aggregated records
mean	average duration of aggregated records
min	minimum duration of aggregated records
nrs	Neighbor Discovery Router Solicit
offset	percrecord byte offset in file or streament pkts retransmitted or dropped
ploss	percent pkts retransmitted or dropped
rate	pkts per second
req	sequence number
rsp	Response packet flag
rst	Reset , is a TCP control flag that is used to terminate an existing TCP connection abruptly
runtime	total active flow run time
sctp	SCTP stands for Stream Control Transmission Protocol. It is a transport layer protocol that is used to transmit data between two endpoints over an IP network.
seq	sequence number
shops	estimate of number of IP hops from source to this point
shutdown	shutdown" refers to the FIN (finish) flag that is sent by a TCP endpoint to indicate that it has finished sending data and is ready to close the connection
smeanpktsz	Mean of the flow packet size transmitted by the src
srcbytes	src -> dst transaction bytes
srcgap	source bytes missing in the data stream
srcload	source bits per second
srcloss	source pkts retransmitted or dropped
srcpkts	src -> dst packet count
srcrate	source pkts per second
srctcpbase	source TCP base sequence number
srcwin	source TCP window advertisement
start	cause code. Valid values are Start, Status, Stop, Close,Error
status	cause code. Valid values are Start, Status, Stop, Close,Error
stos	source TOS byte value
sttl	src -> dst TTL value
sum	total accumulated durations of aggregated records
svid	source VLAN identifier.
synack	TCP connection setup time, the time between the SYN and the SYN_ACK packets
tcp	TCP connection
tcprtt	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'
totbytes	total bytes count
totpkts	total packets count
tst	Time Stamp Request
udp	UDP connection
urp	Unreachable port

Table A-5. Statistics for IoT HealthCare dataset

	feature	count	mean	std	min	25%	50%	75%	max
0	frame_len	188694	164.857	329.238	54	70	78	85	1766
1	frame_time_delta	188694	0.0778461	0.400658	0	1.2e-05	4.2e-05	0.000143	44.4363
2	frame_time_relative	188694	1374.25	1806.57	0	37.6878	431.028	2075.03	6611.04
3	ip_proto	188694	6	0	6	6	6	6	6
4	ip_ttl	188694	72.3823	21.5919	64	64	64	64	128
5	label	188694	0.424635	0.494289	0	0	0	1	1
6	mqtt_clientid_len	188694	0.26482	10.1294	0	0	0	0	680
7	mqtt_conack_val	188694	0	0	0	0	0	0	0
8	mqtt_conflag_passwd	188694	0	0	0	0	0	0	0
9	mqtt_conflag_qos	188694	0	0	0	0	0	0	0
10	mqtt_conflag_reserved	188694	0	0	0	0	0	0	0
11	mqtt_conflag_retain	188694	0	0	0	0	0	0	0
12	mqtt_conflag_willflag	188694	0	0	0	0	0	0	0
13	mqtt_conflags	188694	0.0209122	0.203439	0	0	0	0	2
14	mqtt_dupflag	188694	0.0301122	0.170897	0	0	0	0	1
15	mqtt_hdrflags	188694	51.9202	43.11	0	48	48	50	224
16	mqtt_kalive	188694	353.17	4783.29	0	0	0	0	65535
17	mqtt_len	188694	27.8958	55.8652	0	2	8	14	692
18	mqtt_msgtype	188694	3.20827	2.69477	0	3	3	3	14
19	mqtt_qos	188694	0.172766	0.378046	0	0	0	0	1
20	mqtt_retain	188694	0.00063595	0.0252101	0	0	0	0	1
21	mqtt_topic_len	188694	11.6233	21.1278	0	0	3	8	66
22	mqtt_ver	188694	0.0418243	0.406877	0	0	0	0	4
23	mqtt_willmsg_len	188694	0	0	0	0	0	0	0
24	tcp_ack	188694	66682.9	226161	0	49	175	9029	1.71291e+06
25	tcp_connection_fin	188694	0.00874962	0.0931295	0	0	0	0	1
26	tcp_connection_rst	188694	0.00865422	0.0926249	0	0	0	0	1
27	tcp_connection_sack	188694	0.0102865	0.1009	0	0	0	0	1
28	tcp_connection_syn	188694	0.0102865	0.1009	0	0	0	0	1
29	tcp_dstport	188694	12653	20297.4	1883	1883	1883	1883	65471
30	tcp_flags	188694	22.0999	4.08141	2	24	24	24	25
31	tcp_flags_ack	188694	0.981277	0.135547	0	1	1	1	1
32	tcp_flags_fin	188694	0.00874962	0.0931295	0	0	0	0	1
33	tcp_flags_push	188694	0.789373	0.407755	0	1	1	1	1
34	tcp_flags_reset	188694	0.00865422	0.0926249	0	0	0	0	1
35	tcp_flags_syn	188694	0.020573	0.14195	0	0	0	0	1
36	tcp_flags_urg	188694	0	0	0	0	0	0	0
37	tcp_hdr_len	188694	27.7509	6.0059	20	20	32	32	44
38	tcp_len	188694	101.774	331.692	0	4	10	18	1698
39	tcp_pdu_size	188694	29.6975	56.3666	0	4	10	16	695
40	tcp_srcport	188694	35293.6	19333.1	1883	33387	39979	51684	65471
41	tcp_time_delta	188694	1.87043	6.48436	0	3.3e-05	0.504396	2.00098	60.0373
42	tcp_window_size_value	188694	2338.46	7052.41	0	512	512	4087	65535

Table A-6. Statistics for TON-IOT dataset

	feature	count	mean	std	min	25%	50%	75%	max
0	current_temperature	401119	28.4848	0.870655	25	28.4427	28.4427	28.4427	34.9962
1	door_state_closed	401119	0.133596	0.340218	0	0	0	0	1
2	door_state_nan	401119	0.851448	0.355647	0	1	1	1	1
3	door_state_open	401119	0.0149557	0.121376	0	0	0	0	1
4	fc1_read_input_register	401119	32489.2	6740.84	0	32450	32450	32450	65527
5	fc2_read_discrete_value	401119	32716.9	6786.73	0	32708	32708	32708	65535
6	fc3_read_holding_register	401119	32094.7	6754.3	0	32035	32035	32035	65525
7	fc4_read_coil	401119	32740.8	6733.42	0	32728	32728	32728	65534
8	fridge_temperature	401119	6.76454	1.40525	1	6.7	6.7	6.7	14
9	humidity	401119	46.6961	11.0068	0.278928	46.3436	46.3436	46.3436	99.876
10	label	401119	0.389209	0.487571	0	0	0	1	1
11	latitude	401119	10.7155	32.2581	0	4.51408	4.51408	4.51408	549.382
12	light_status_nan	401119	0.851695	0.355403	0	1	1	1	1
13	light_status_off	401119	0.092586	0.289852	0	0	0	0	1
14	light_status_on	401119	0.0557191	0.229379	0	0	0	0	1
15	longitude	401119	20.6569	32.3876	10	14.4219	14.4219	14.4219	555.134
16	motion_status_0	401119	0.092586	0.289852	0	0	0	0	1
17	motion_status_1	401119	0.0557191	0.229379	0	0	0	0	1
18	motion_status_nan	401119	0.851695	0.355403	0	1	1	1	1
19	pressure	401119	1.0139	1.0726	-12.0654	1.035	1.035	1.035	12.4876
20	sphone_signal_0	401119	0.0550884	0.228153	0	0	0	0	1
21	sphone_signal_1	401119	0.00620763	0.0785437	0	0	0	0	1
22	sphone_signal_false	401119	0.0785079	0.268969	0	0	0	0	1
23	sphone_signal_nan	401119	0.851448	0.355647	0	1	1	1	1
24	sphone_signal_true	401119	0.00874803	0.093121	0	0	0	0	1
25	temp_condition_high	401119	0.0836659	0.276887	0	0	0	0	1
26	temp_condition_low	401119	0.065776	0.24789	0	0	0	0	1
27	temp_condition_nan	401119	0.850558	0.356524	0	1	1	1	1
28	temperature	401119	35.7152	3.2082	20.527	35.7736	35.7736	35.7736	50
29	thermostat_status_0	401119	0.0155689	0.123801	0	0	0	0	1
30	thermostat_status_1	401119	0.115998	0.320223	0	0	0	0	1
31	thermostat_status_nan	401119	0.868433	0.33802	0	1	1	1	1
32	ts	401119	1.55497e+09	1.08741e+06	1.55406e+09	1.55406e+09	1.55418e+09	1.55624e+09	1.55653e+09

Table A-7. Statistics for 5G-NIDD dataset

	feature	count	mean	std	min	25%	50%	75%	max
0	39	1.21589e+06	8.22443e-07	0.000906886	0	0	0	0	1
1	4	1.21589e+06	6.08608e-05	0.0078011	0	0	0	0	1
2	52	1.21589e+06	0.000118432	0.010882	0	0	0	0	1
3	54	1.21589e+06	8.22443e-07	0.000906886	0	0	0	0	1
4	_*_	1.21589e+06	0.00403984	0.0634312	0	0	0	0	1
5	_*_f_	1.21589e+06	0.0918685	0.28884	0	0	0	0	1
6	_*_v_	1.21589e+06	1.97386e-05	0.00444278	0	0	0	0	1
7	_e_	1.21589e+06	0.885742	0.318124	0	1	1	1	1
8	_e_&_	1.21589e+06	3.78324e-05	0.00615069	0	0	0	0	1
9	_e_*_	1.21589e+06	0.000130768	0.0114347	0	0	0	0	1
10	_e_d_	1.21589e+06	0.00569377	0.075242	0	0	0	0	1
11	_e_f_	1.21589e+06	0.0022828	0.0477187	0	0	0	0	1
12	_e_g_	1.21589e+06	7.2375e-05	0.00850704	0	0	0	0	1
13	_e_i_	1.21589e+06	0.000555149	0.0235551	0	0	0	0	1
14	_e_r_	1.21589e+06	0.00135456	0.0367795	0	0	0	0	1
15	_e_s_	1.21589e+06	0.00755085	0.086567	0	0	0	0	1
16	_eu_	1.21589e+06	0.000648085	0.0254493	0	0	0	0	1
17	acc	1.21589e+06	0.000912089	0.030187	0	0	0	0	1
18	ackdat	1.21589e+06	0.00408815	0.0105128	0	0	0	0	0.266729
19	af11	1.21589e+06	0.000648987	0.0254654	0	0	0	0	1
20	af12	1.21589e+06	0.000112675	0.0106142	0	0	0	0	1
21	af41	1.21589e+06	0.000423558	0.0205762	0	0	0	0	1
22	arp	1.21589e+06	3.37202e-05	0.00580681	0	0	0	0	1
23	con	1.21589e+06	0.108706	0.31127	0	0	0	0	1
24	cs0	1.21589e+06	0.994511	0.0738841	0	1	1	1	1
25	cs4	1.21589e+06	7.64872e-05	0.00874516	0	0	0	0	1
26	cs6	1.21589e+06	0.000526363	0.0229366	0	0	0	0	1
27	cs7	1.21589e+06	0.00043425	0.0208341	0	0	0	0	1
28	dhops	1.21589e+06	5.00543	1.0871	0	5	5	5	50
29	dhops_is_nan	1.21589e+06	0.775619	0.417174	0	1	1	1	1
30	dmeanpktsz	1.21589e+06	61.8925	215.362	0	0	0	0	1478
31	dstbytes	1.21589e+06	1114.2	16768.2	0	0	0	0	4.55168e+06
32	dstgap	1.21589e+06	0.263775	66.9387	0	0	0	0	57387
33	dstgap_is_nan	1.21589e+06	0.770809	0.420313	0	1	1	1	1
34	dstload	1.21589e+06	4.78999e+06	5.8471e+08	0	0	0	0	2.63e+11
35	dstloss	1.21589e+06	0.00964479	0.155889	0	0	0	0	13
36	dstpkts	1.21589e+06	1.44413	12.4658	0	0	0	0	3151
37	dstrate	1.21589e+06	499.966	56571.1	0	0	0	0	2.6e+07
38	dsttcpbase	1.21589e+06	4.10539e+08	9.97769e+08	33089	5.49704e+06	5.49704e+06	5.49704e+06	4.29496e+09
39	dsttcpbase_is_nan	1.21589e+06	0.810799	0.391668	0	1	1	1	1
40	dstwin	1.21589e+06	65568.4	77141.7	0	64896	64896	64896	1.6777e+07
41	dstwin_is_nan	1.21589e+06	0.854363	0.352742	0	1	1	1	1
42	dtos	1.21589e+06	0.591827	10.0258	0	0	0	0	186
43	dtos_is_nan	1.21589e+06	0.775619	0.417174	0	1	1	1	1
44	dttl	1.21589e+06	60.4189	13.5656	37	59	59	59	255
45	dttl_is_nan	1.21589e+06	0.775619	0.417174	0	1	1	1	1
46	dur	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
47	dvid	1.21589e+06	0.998348	0.0406148	0	1	1	1	1
48	e_	1.21589e+06	4.11221e-06	0.00202786	0	0	0	0	1
49	eco	1.21589e+06	0.0237044	0.152127	0	0	0	0	1
50	ef	1.21589e+06	0.0029098	0.0538641	0	0	0	0	1
51	fin	1.21589e+06	0.048673	0.215184	0	0	0	0	1
52	icmp	1.21589e+06	0.0241739	0.154207	0	0	0	0	1
53	int	1.21589e+06	0.270375	0.444154	0	0	0	1	1
54	ipv6_icmp	1.21589e+06	1.64489e-06	0.00128253	0	0	0	0	1
55	label	1.21589e+06	0.607089	0.488398	0	0	1	1	1
56	llc	1.21589e+06	4.77017e-05	0.00690648	0	0	0	0	1
57	lldp	1.21589e+06	9.45809e-05	0.00972482	0	0	0	0	1
58	load	1.21589e+06	4.92898e+06	5.94578e+08	0	0	0	130.519	2.63e+11
59	loss	1.21589e+06	0.0228746	0.247945	0	0	0	0	28
60	max	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
61	mean	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
62	min	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
63	nan	1.21589e+06	0.000176003	0.0132654	0	0	0	0	1
64	nrs	1.21589e+06	1.64489e-06	0.00128253	0	0	0	0	1
65	offset	1.21589e+06	1.23794e+07	1.06968e+07	128	3.05303e+06	9.89799e+06	1.91238e+07	3.96922e+07
66	ploss	1.21589e+06	0.349491	3.66395	0	0	0	0	62.5
67	rate	1.21589e+06	934.132	85405.9	0	0	0	0.388612	2.8e+07
68	req	1.21589e+06	0.484665	0.499765	0	0	0	1	1
69	rsp	1.21589e+06	6.00383e-05	0.00774821	0	0	0	0	1
70	rst	1.21589e+06	0.0622342	0.241581	0	0	0	0	1
71	runtime	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
72	sctp	1.21589e+06	0.00355871	0.0595487	0	0	0	0	1
73	seq	1.21589e+06	40343.3	37758.6	1	8861	27223	63316	137210
74	shops	1.21589e+06	2.25688	3.56402	0	1	1	1	28
75	shops_is_nan	1.21589e+06	0.000176003	0.0132654	0	0	0	0	1
76	shutdown	1.21589e+06	0.000577355	0.0240213	0	0	0	0	1
77	smeanpktsz	1.21589e+06	74.0125	144.951	0	42	42	67	1467
78	srcbytes	1.21589e+06	2532.68	24354.7	0	42	74	84	1.34348e+06
79	srcgap	1.21589e+06	0.0119838	5.03488	0	0	0	0	3064
80	srcgap_is_nan	1.21589e+06	0.770809	0.420313	0	1	1	1	1
81	srcload	1.21589e+06	139417	1.30816e+07	0	0	0	130.493	6.072e+09
82	srcloss	1.21589e+06	0.013238	0.176643	0	0	0	0	15
83	srcpkts	1.21589e+06	3.70082	18.4122	0	1	1	2	986
84	srcrate	1.21589e+06	230.011	22368.3	0	0	0	0.388341	5e+06
85	srctcpbase	1.21589e+06	4.81912e+08	1.03656e+09	123073	1.73698e+07	1.73698e+07	1.73698e+07	4.29497e+09
86	srctcpbase_is_nan	1.21589e+06	0.770809	0.420313	0	1	1	1	1
87	srcwin	1.21589e+06	233868	2.25128e+06	0	64256	64256	64256	3.35539e+07
88	srcwin_is_nan	1.21589e+06	0.800623	0.399532	0	1	1	1	1
89	start	1.21589e+06	0.480137	0.489926	0	0	1	1	1
90	status	1.21589e+06	0.599285	0.490043	0	0	1	1	1
91	stos	1.21589e+06	0.86423	12.5179	0	0	0	0	224
92	stos_is_nan	1.21589e+06	0.000176003	0.0132654	0	0	0	0	1
93	sttl	1.21589e+06	81.4512	55.9223	36	63	63	63	255
94	sttl_is_nan	1.21589e+06	0.000176003	0.0132654	0	0	0	0	1
95	sum	1.21589e+06	1.36484	1.68541	0	0	0	2.5804	19.9206
96	svid	1.21589e+06	0.985772	0.292146	0	1	1	1	1
97	synack	1.21589e+06	0.000581607	0.0130498	0	0	0	0	1.02468
98	tcp	1.21589e+06	0.229191	0.420313	0	0	0	0	1
99	tcprtt	1.21589e+06	0.00466976	0.0174359	0	0	0	0	1.05124
100	totbytes	1.21589e+06	3646.88	30102.5	42	42	84	84	4.65626e+06
101	totpkts	1.21589e+06	5.14495	24.8201	1	1	2	2	3997
102	tst	1.21589e+06	2.30284e-05	0.00479874	0	0	0	0	1
103	udp	1.21589e+06	0.742699	0.437147	0	0	1	1	1
104	urp	1.21589e+06	0.00054644	0.025417	0	0	0	0	1

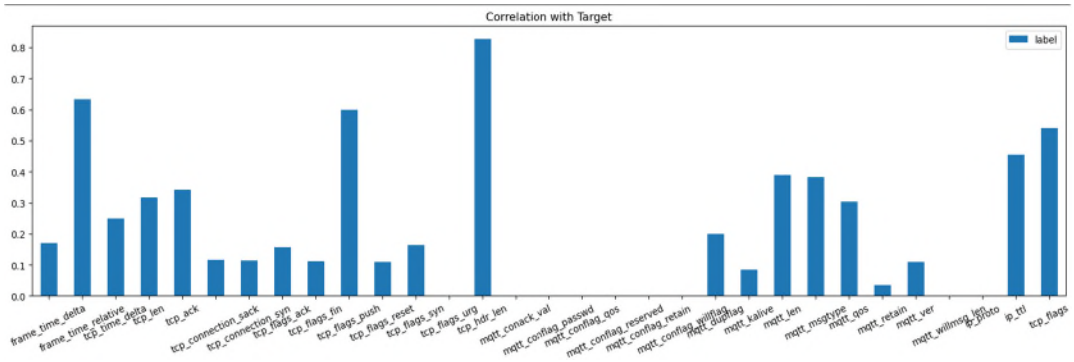


Figure A-1. Feature correlation with target variable for IoT HealthCare dataset

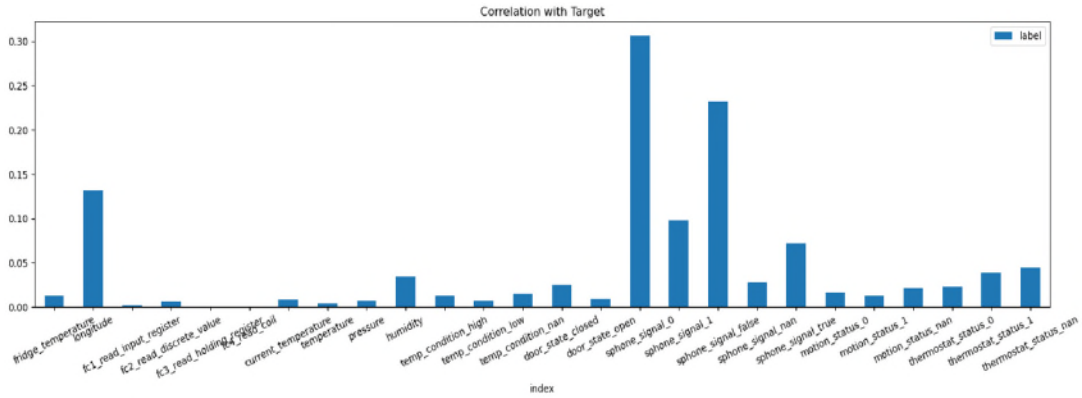


Figure A-2. Feature correlation with target variable for TON-IOT dataset

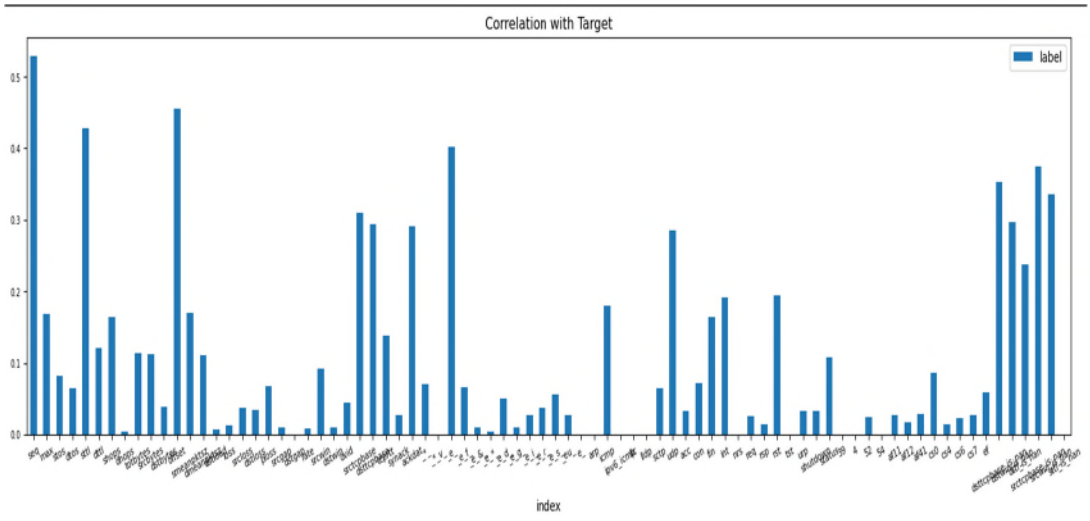


Figure A-3. Feature correlation with target variable for 5G-NIDD dataset

Table A-8. ANOVA analysis for IoT HealthCare Dataset

	feature	f_value	p_value
8	tcp_flags_push	106851.588761	0.000000e+00
28	tcp_flags	78450.239341	0.000000e+00
27	ip_ttl	48421.300672	0.000000e+00
20	mqtt_len	35378.970983	0.000000e+00
21	mqtt_msgtype	32128.881130	0.000000e+00
3	tcp_ack	24408.553818	0.000000e+00
2	tcp_len	21304.343734	0.000000e+00
22	mqtt_qos	19795.654304	0.000000e+00
1	tcp_time_delta	12134.909436	0.000000e+00
18	mqtt_dupflag	8286.454752	0.000000e+00
10	tcp_flags_syn	5527.733446	0.000000e+00
6	tcp_flags_ack	5007.855528	0.000000e+00
0	frame_time_delta	4652.018655	0.000000e+00
5	tcp_connection_syn	2695.251703	0.000000e+00
4	tcp_connection_sack	2695.251703	0.000000e+00
7	tcp_flags_fin	2284.088794	0.000000e+00
9	tcp_flags_reset	2258.668478	0.000000e+00
24	mqtt_ver	2176.921349	0.000000e+00
19	mqtt_kalive	1396.208668	1.836870e-304
23	mqtt_retain	162.838061	2.812125e-37
11	tcp_flags_urg	NaN	NaN
12	mqtt_conack_val	NaN	NaN
13	mqtt_conflag_passwd	NaN	NaN
14	mqtt_conflag_qos	NaN	NaN
15	mqtt_conflag_reserved	NaN	NaN
16	mqtt_conflag_retain	NaN	NaN
17	mqtt_conflag_willflag	NaN	NaN
25	mqtt_willmsg_len	NaN	NaN
26	ip_proto	NaN	NaN

Table A-9. ANOVA analysis for TON-IOT Dataset

	feature	f_value	p_value
15	sphone_signal_0	40394.398616	0.000000e+00
17	sphone_signal_false	23026.282697	0.000000e+00
1	longitude	6793.507521	0.000000e+00
16	sphone_signal_1	3970.910155	0.000000e+00
19	sphone_signal_true	2268.485415	0.000000e+00
25	thermostat_status_nan	703.528877	6.958669e-155
24	thermostat_status_1	489.517910	2.107576e-108
9	humidity	394.943940	7.655949e-88
23	thermostat_status_0	229.997425	6.170800e-52
12	temp_condition_nan	214.812793	1.260398e-48
18	sphone_signal_nan	161.348540	5.836440e-37
8	pressure	157.839667	3.408015e-36
22	motion_status_nan	148.069117	4.645532e-34
13	door_state_closed	139.374673	3.692260e-32
10	temp_condition_high	105.276544	1.069764e-24
7	temperature	104.865717	1.316157e-24
20	motion_status_0	97.250341	6.146203e-23
11	temp_condition_low	92.497742	6.776972e-22
21	motion_status_1	40.757693	1.725109e-10
4	fc3_read_holding_register	26.352162	2.846326e-07
14	door_state_open	17.133447	3.485038e-05
0	fridge_temperature	10.058375	1.516683e-03
3	fc2_read_discrete_value	6.481138	1.090289e-02
6	current_temperature	3.032701	8.160283e-02
2	fc1_read_input_register	1.732618	1.880777e-01
5	fc4_read_coil	0.091090	7.627960e-01

Table A-10. ANOVA analysis for 5G-NIDD Dataset

feature	f_value	p_value
seq	469849.6	0
offset	319314.4	0
sttl	271783.7	0
_e_	227726.2	0
srctcpbase_is_nan	203833.9	0
dsttcpbase_is_nan	174928.3	0
srcwin_is_nan	157667.9	0
srctcpbase	129168.8	0
dsttcpbase	118644.6	0
dstwin_is_nan	115841.2	0
ackdat	114843.3	0
udp	110850.7	0
dttl_is_nan	70001.61	0
rst	50539.54	0
int	48708.61	0
tcprtt	48169.83	0
icmp	42247.65	0
smeanpktsz	38473.45	0
fin	34926.33	0
shops	34676.73	0
max	33113.46	0
dttl	20906.78	0
srcbytes	18141.39	0
totbytes	17039.93	0
dmeanpktsz	14488.06	0
status	13781.76	0
cs0	10458.1	0
srcwin	9945.389	0
stos	9021.126	0
_*_	7668.38	0

Table A-11. Logistics regression coefficients for feature importance selection for IoT HealthCare dataset

	feature	importance
0	tcp_time_delta	0.3989
1	ip_ttl	0.1796
2	mqtt_msgtype	0.1402
3	mqtt_len	0.112
4	tcp_flags_push	0.0311
5	tcp_ack	0.0251
6	tcp_flags	0.0243
7	tcp_flags_fin	0.0211
8	mqtt_dupflag	0.0196
9	tcp_len	0.0189
10	mqtt_kalive	0.0114
11	frame_time_delta	0.0062
12	mqtt_ver	0.0049
13	tcp_connection_sack	0.0021
14	mqtt_qos	0.0017
15	tcp_flags_syn	0.0017
16	tcp_flags_reset	0.0006
17	tcp_flags_ack	0.0003
18	tcp_connection_syn	0.0002
19	mqtt_retain	0.0001
20	ip_proto	0
21	mqtt_willmsg_len	0
22	mqtt_conack_val	0
23	mqtt_conflag_passwd	0
24	mqtt_conflag_willflag	0
25	mqtt_conflag_retain	0
26	mqtt_conflag_reserved	0
27	tcp_flags_urg	0
28	mqtt_conflag_qos	0

Table A-12. Logistics regression coefficients for feature importance selection for TON-IOT dataset

	feature	importance
0	sphone_signal_0	0.3371
1	sphone_signal_false	0.3044
2	sphone_signal_1	0.1166
3	sphone_signal_true	0.1052
4	longitude	0.0464
5	sphone_signal_nan	0.0157
6	door_state_closed	0.0146
7	humidity	0.0077
8	temp_condition_nan	0.0071
9	motion_status_nan	0.0067
10	pressure	0.0059
11	door_state_open	0.0053
12	motion_status_0	0.0052
13	temp_condition_high	0.0049
14	temp_condition_low	0.0048
15	motion_status_1	0.0037
16	thermostat_status_0	0.0032
17	current_temperature	0.0013
18	fc2_read_discrete_value	0.0011
19	temperature	0.0008
20	thermostat_status_1	0.0007
21	thermostat_status_nan	0.0005
22	fc3_read_holding_register	0.0004
23	fc1_read_input_register	0.0004
24	fridge_temperature	0.0002
25	fc4_read_coil	0.0001

Table A-13. Logistics regression coefficients for feature importance selection for 5G-NIDD dataset

	feature	importance
0	srcbytes	0.3509
1	dttl	0.1208
2	smeanpktsz	0.0833
3	sttl	0.0752
4	seq	0.0673
5	dsttcpbase_is_nan	0.0433
6	dttl_is_nan	0.032
7	ackdat	0.0284
8	offset	0.028
9	udp	0.0226
10	cs0	0.0211
11	totbytes	0.0189
12	srctcpbase_is_nan	0.017
13	icmp	0.0167
14	fin	0.0146
15	srcwin_is_nan	0.0131
16	srcwin	0.0128
17	shops	0.0083
18	int	0.0055
19	_e_	0.0054
20	rst	0.0046
21	tcprtt	0.0022
22	status	0.0018
23	dmeanpktsz	0.0018
24	dstwin_is_nan	0.0016
25	_*_	0.0007
26	stos	0.0007
27	srctcpbase	0.0007
28	max	0.0004
29	dsttcpbase	0.0003

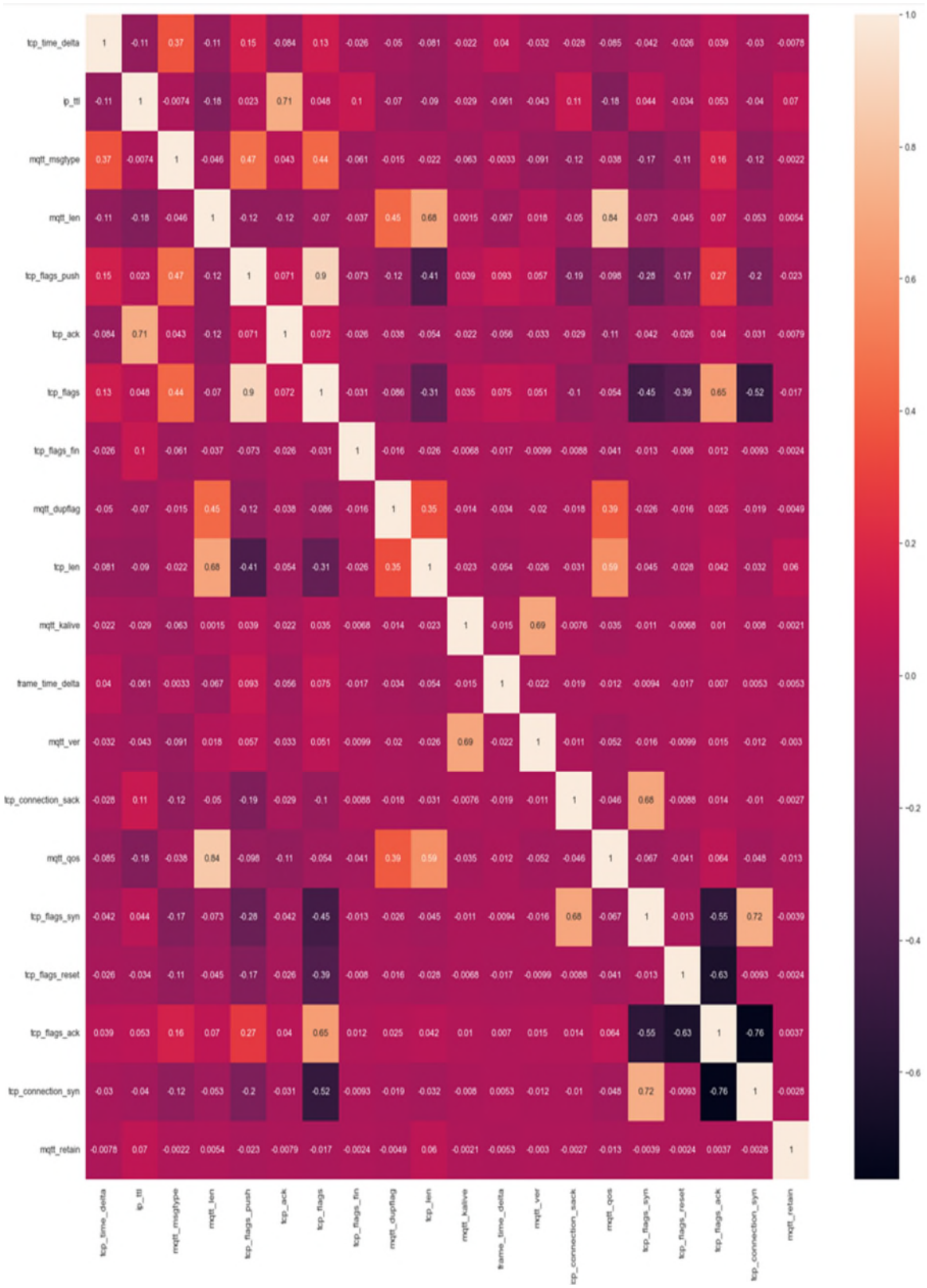


Figure A-4. Correlation matrix for IoT HealthCare dataset

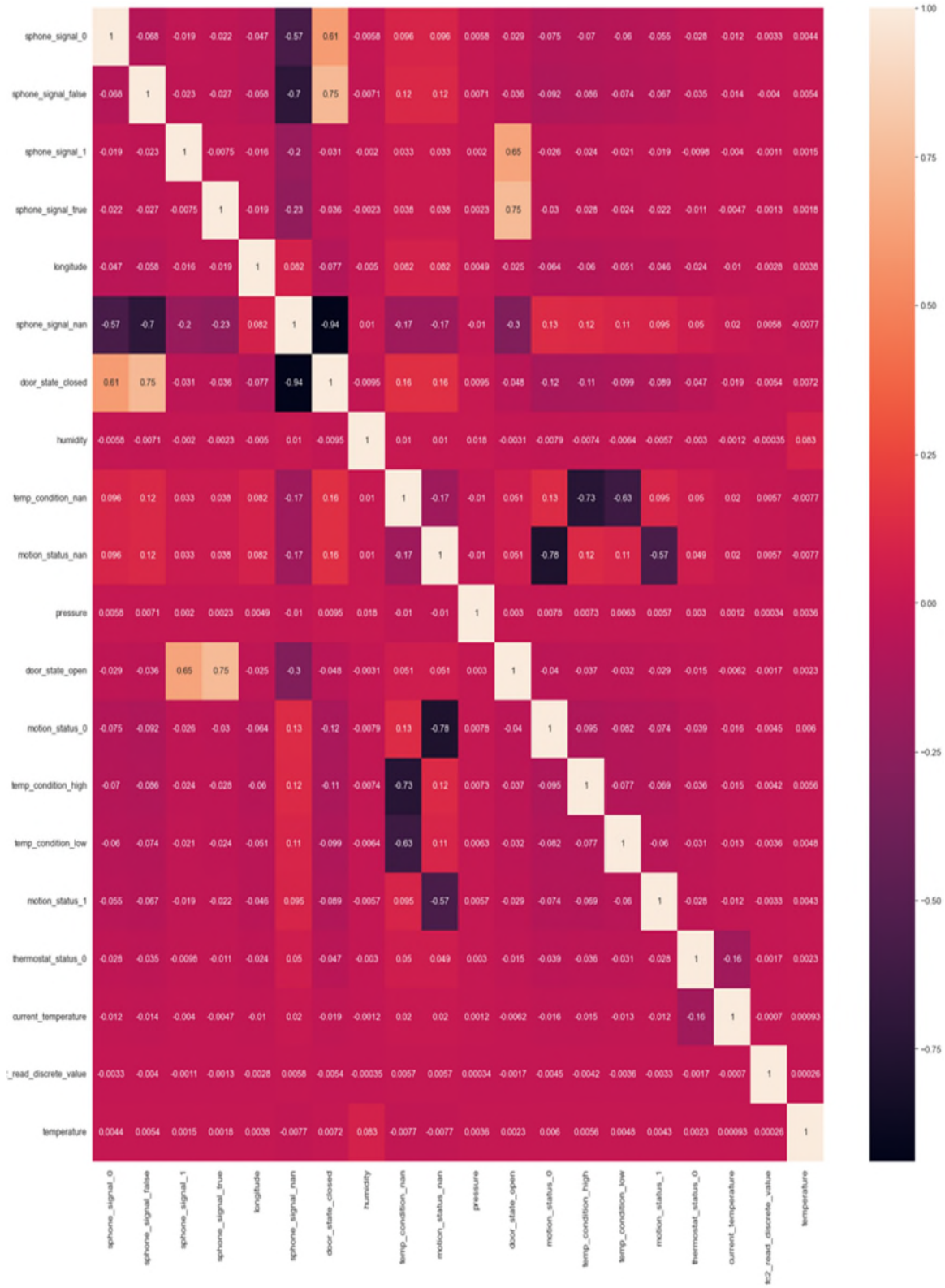


Figure A-5. Correlation matrix for TON-IOT dataset

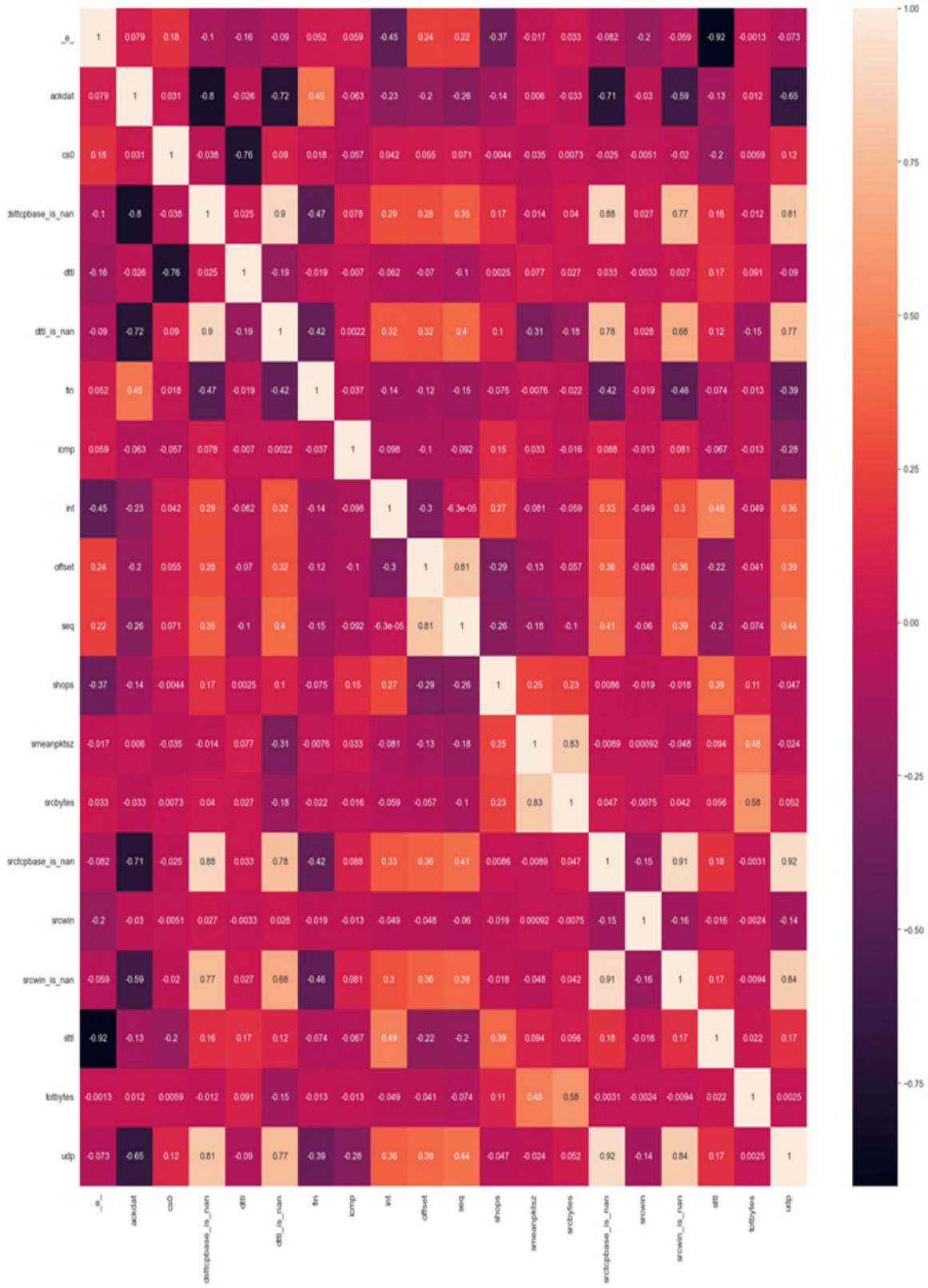


Figure A-6. Correlation matrix for 5G-NIDD dataset

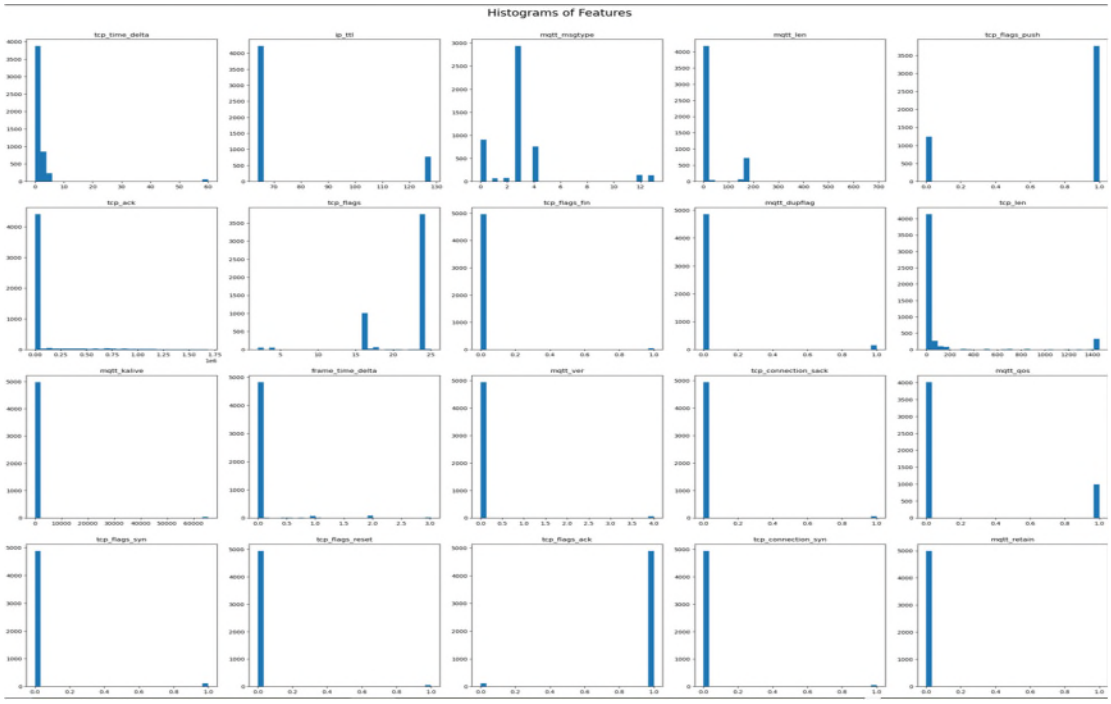


Figure A-7. Histogram of IoT HealthCare dataset features before standardization

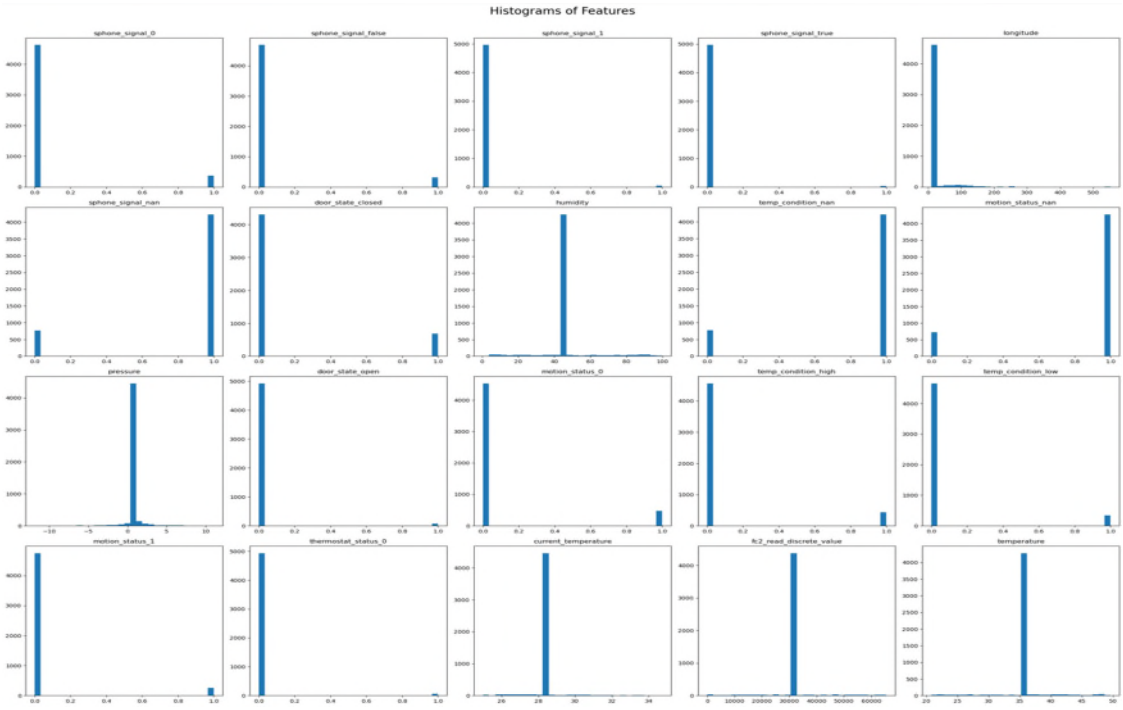


Figure A-8. Histogram of TON-IOT dataset features before standardization

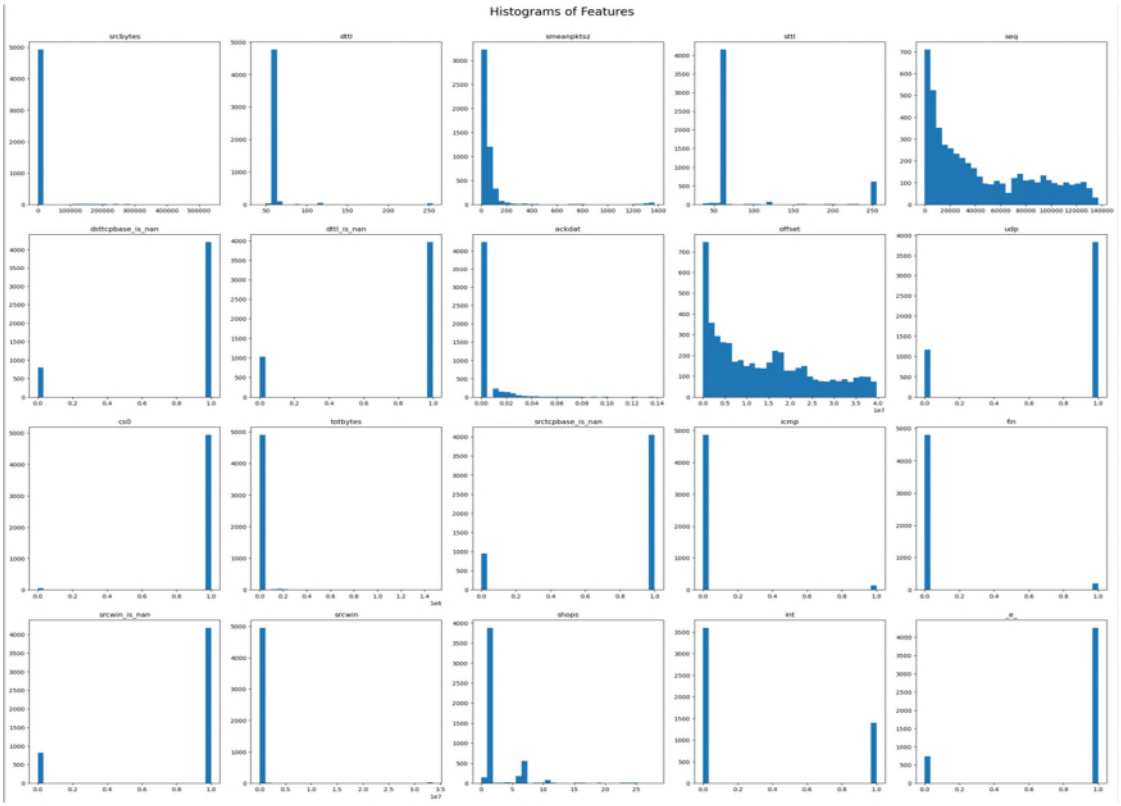


Figure A-9. Histogram of 5G-NIDD dataset features before standardization

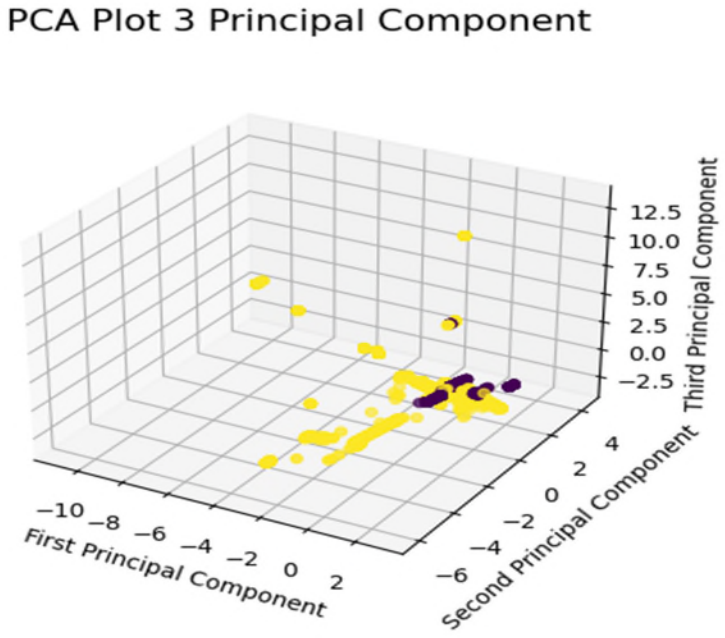


Figure A-10. The three principal components of IoT HealthCare dataset

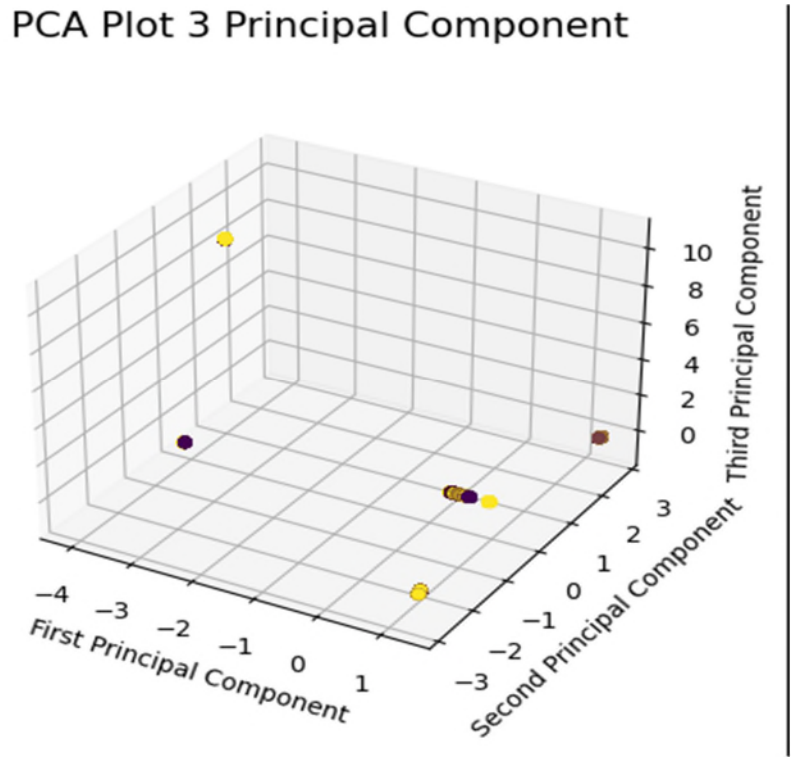


Figure A-11. The three principal components of TON-IOT dataset

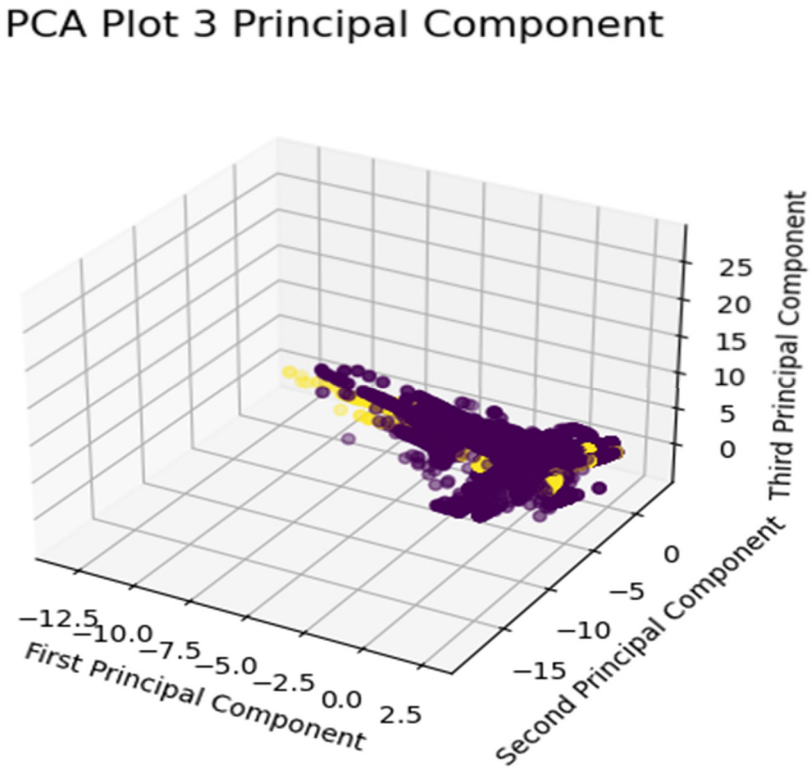


Figure A-12. The three principal components of 5G-NIDD dataset

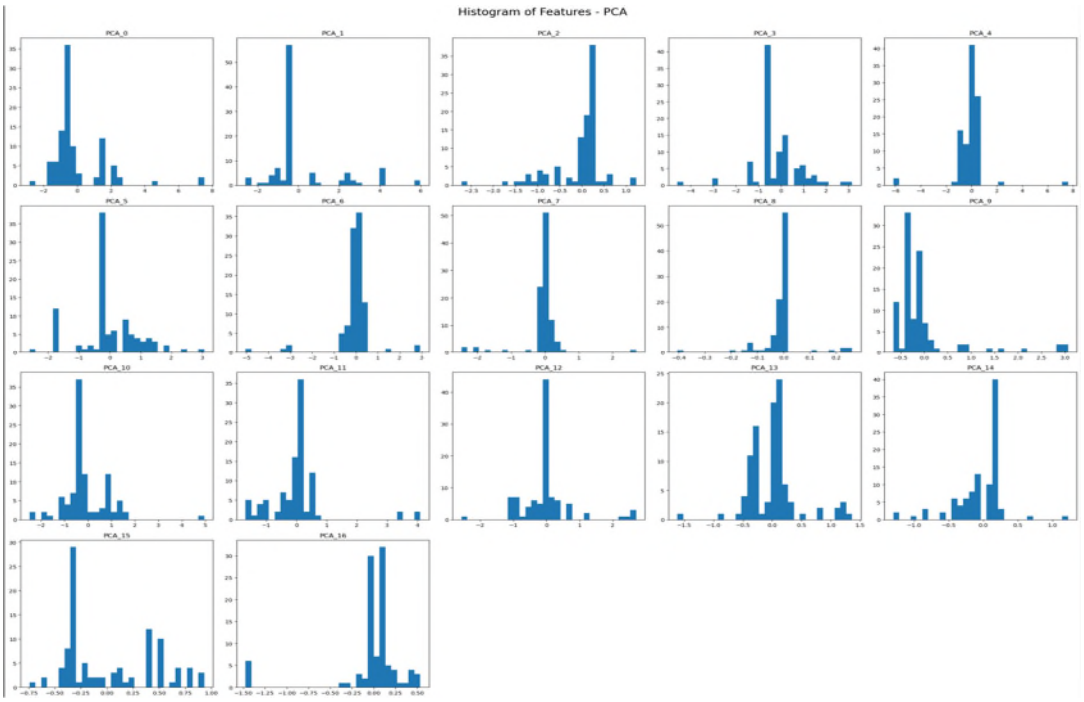


Figure A-13. Histogram of IoT HealthCare dataset features after PCA and standardization

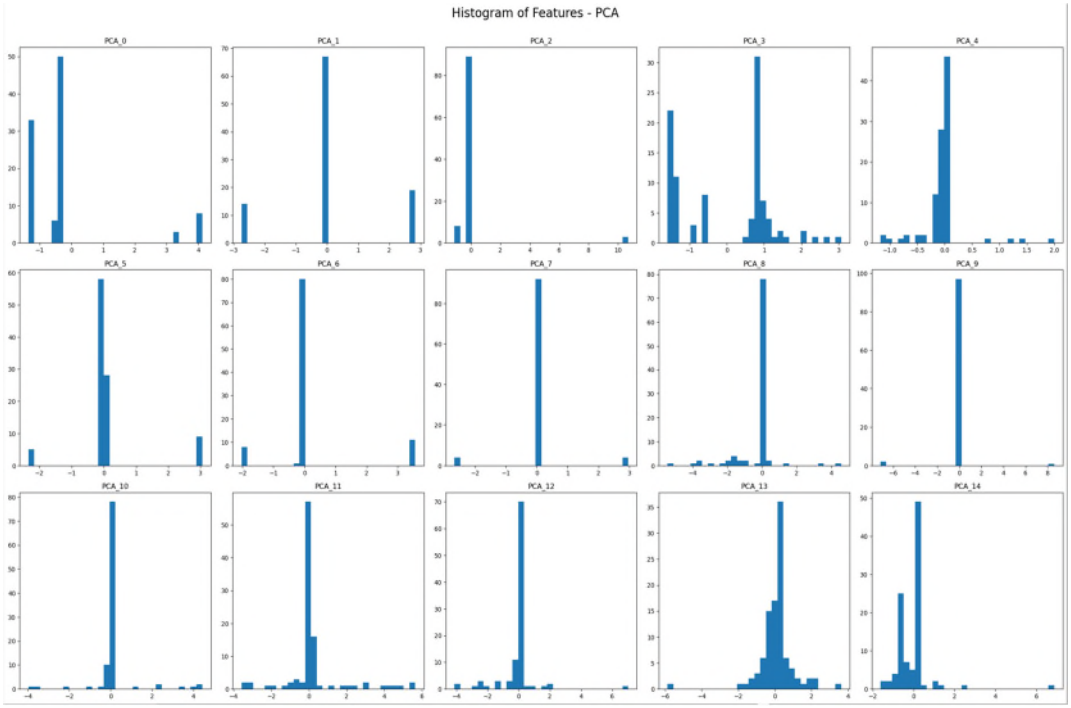


Figure A-14. Histogram of TON-IOT dataset features after PCA and standardization

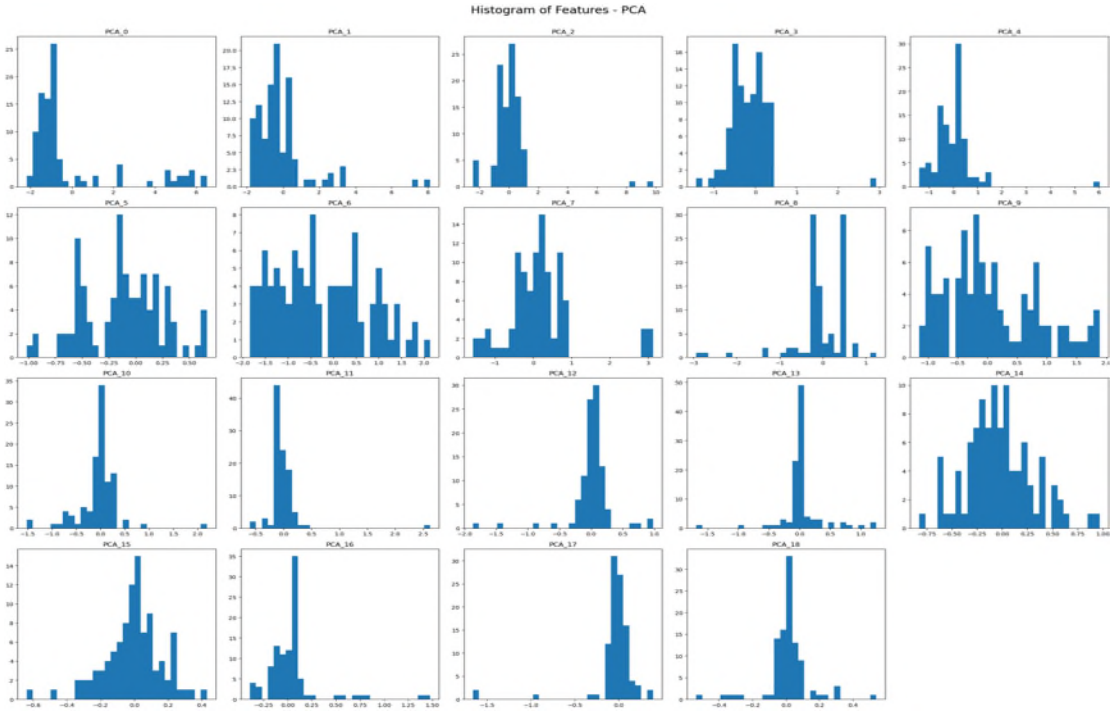


Figure A-15. Histogram of 5G-NIDD dataset features after PCA and standardization

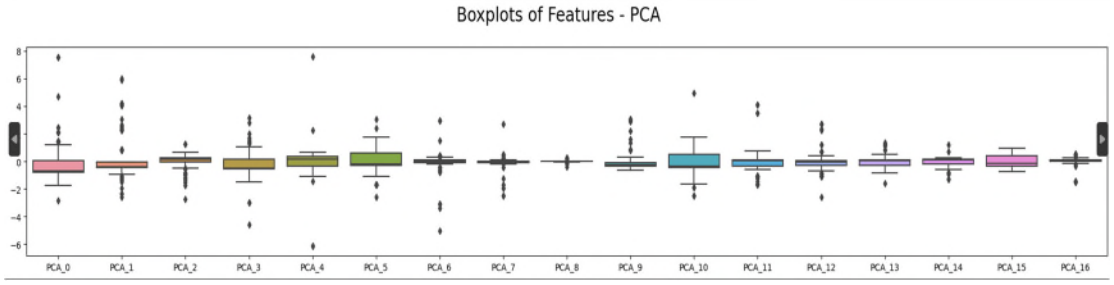


Figure A-16. Boxplots of IoT HealthCare dataset features after PCA and standardization

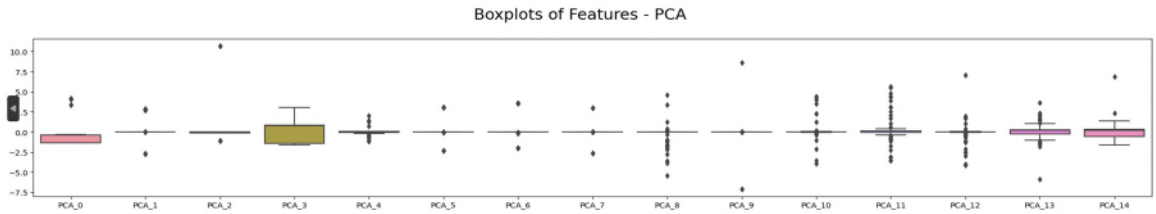


Figure A-17. Boxplots of TON-IOT dataset features after PCA and standardization

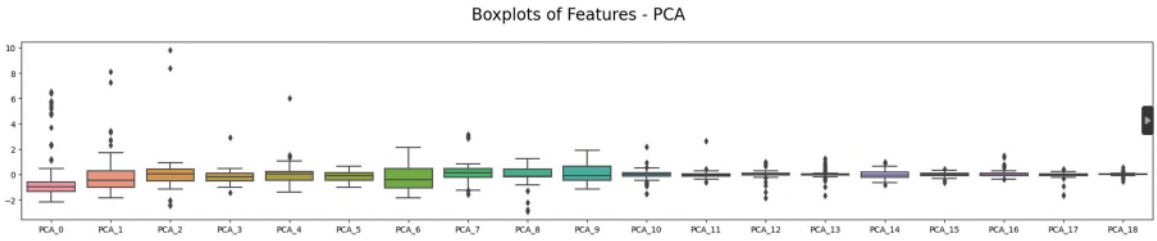


Figure A-18. Boxplots of 5G-NIDD dataset features after PCA and standardization

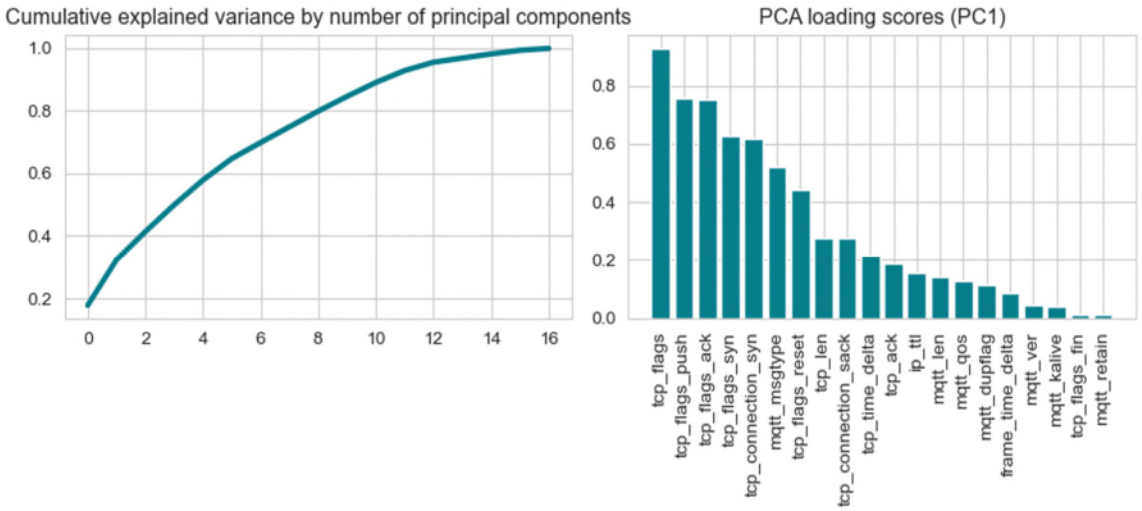


Figure A-19. Explained variance by number of principal components for IoT HealthCare dataset

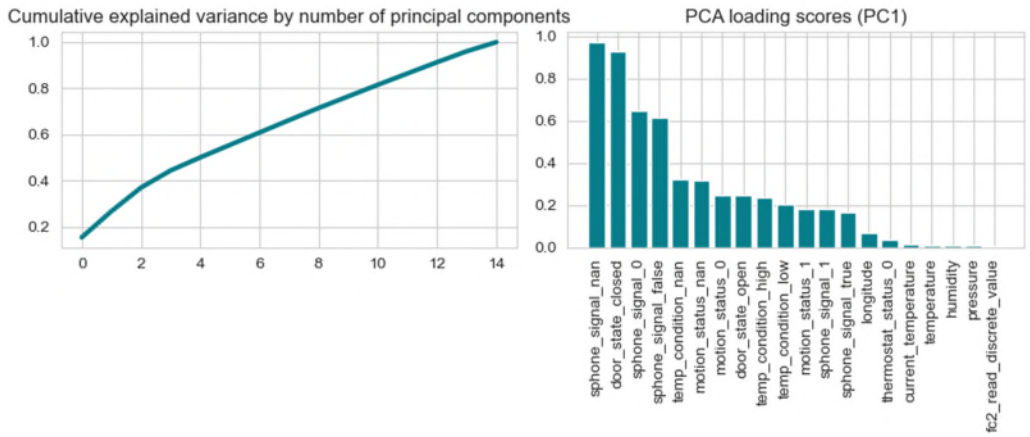


Figure A-20. Explained variance by number of principal components for TON-IOT dataset

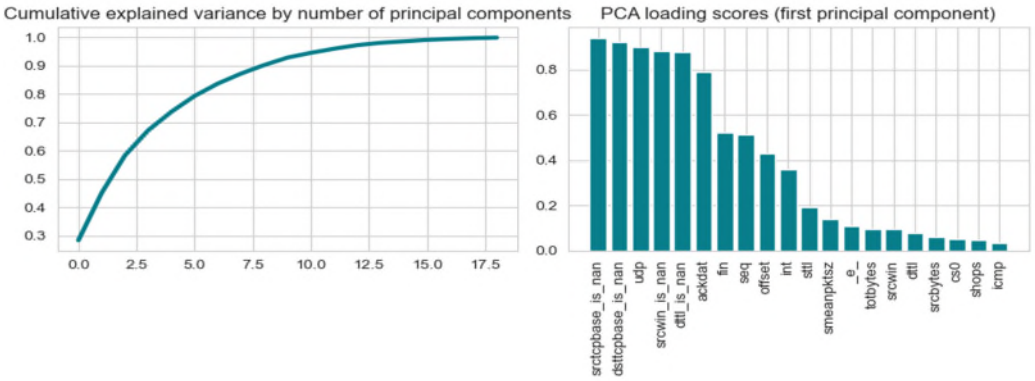


Figure A-21. Explained variance by number of principal components for 5G-NIDD dataset

Table A-14. Hyperparameter tuning results for the IoT Healthcare dataset

Hyperparameter being tuned	Search space	Final value
XGB		
n_estimators	10,1000,	166
max_depth	2,20,1	6
learning_rate	0.01 ,1	0.428258606
reg_alpha	0,10	1.358233947
reg_lambda	0,10	0.406473145
min_child_weight	1,10,1	9
Gaussian NB		
var_smoothing	-10,1	0.037085434
Decision Tree		
min_samples_split	2,20,1	2
min_sample_leaf	1,20,1	1
max_features	sqrt, log2, none	none
Ccp_alpha	0, x	0.045837652
RNN		
units	64	
activation	relu	
solver/optimizer	adam	
Bagging (Decision trees)		
n_estimators	10,1000, 1	115
max_samples	0.1,1	0.837435225
max_features	0.1,1	0.585181059
bootstrap	True, False	FALSE
SVC		
C	0,10	3.242170297
gamma	scale,auto	scale
kernel	linear, poly, sigmoid, rbf	sigmoid
Random Forest		
n_estimators	10,1000,	529
min_samples_split	2,20,1	13
min_sample_leaf	1,20,1	5
max_features	sqrt	sqrt
max_depth	2,20,1	9
criterion	Gini,	Gini

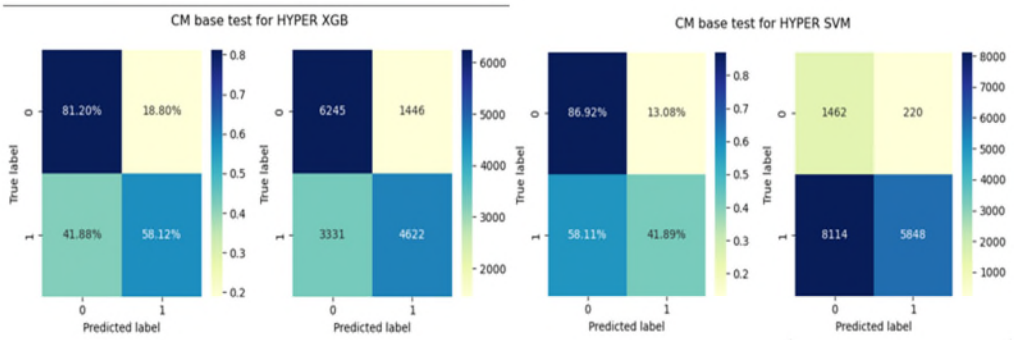
Table A-15. Hyperparameter tuning results for the ToN-IoT dataset

Hyperparameter being tuned	Search space	Final value
XGB		
n_estimators	10,1000,1	876
max_depth	2,20,1	14
learning_rate	0.01 ,1	0.404038534
reg_alpha	0,10	8.165058318
reg_lambda	0,10	1.149420753
min_child_weight	1,10,1	9
Gaussian NB		
var_smoothing	-10,1	0.000306336
Decision Tree		
min_samples_split	2,20,1	2
min_sample_leaf	1,20,1	1
max_features	sqrt, log2, none	none
Ccp_alpha	0, x	
RNN		
units	64	
activation	relu	
solver/optimizer	adam	
Bagging (Decision trees)		
n_estimators	10,1000,1	965
max_samples	0.1,1	0.6052957
max_features	0.1,1	0.946660838
bootstrap	True, False	TRUE
SVC		
C	0,10	2.370542226
gamma	scale,auto	scale
kernel	linear, poly, sigmoid, rbf	poly
Random Forest		
n_estimators	10,1000,1	713
min_samples_split	2,20,1	15
min_sample_leaf	1,20,1	18
max_features	sqrt ,log2,none	sqrt
max_depth	2,20,1	19
criterion	Gini, entropy	Gini

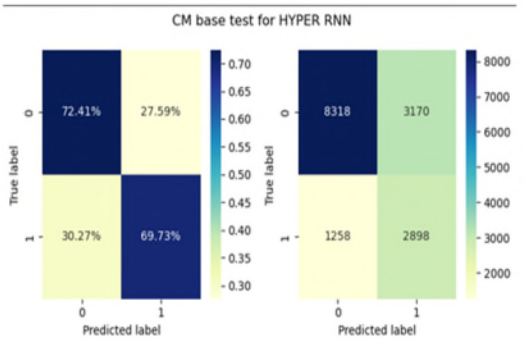
Table A-16. Hyperparameter tuning results for the 5G-NIDD dataset

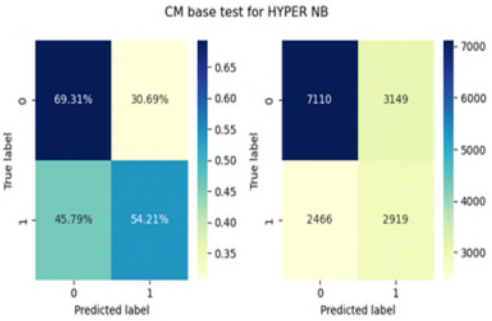
Hyperparameter being tuned	Search space	Final value
XGB		
n_estimators	10,1000,	201
max_depth	2,20,1	15
learning_rate	0.01 ,1	0.224078179
reg_alpha	0,10	1.797951455
reg_lambda	0,10	8.191522184
min_child_weight	1,10,1	2
Gaussian NB		
var_smoothing	-10,1	0.000563505
Decision Tree		
min_samples_split	2,20,1	2
min_sample_leaf	1,20,1	1
max_features	sqrt, log2, none	none
Ccp_alpha	0, x	0.005259345
RNN		
units	64	
activation	relu	
solver/optimizer	adam	
Bagging (Decision trees)		
n_estimators	10,1000, 1	989
max_samples	0.1,1	0.293271086
max_features	0.1,1	0.570849948
bootstrap	True, False	FALSE
SVC		
C	0,10	6.799511089
gamma	scale,auto	scale
kernel	linear, poly, sigmoid, rbf	linear
Random Forest		
n_estimators	10,1000,	114
min_samples_split	2,20,1	9
min_sample_leaf	1,20,1	8
max_features	sqrt	log2
max_depth	2,20,1	15
criterion	Gini,	Gini

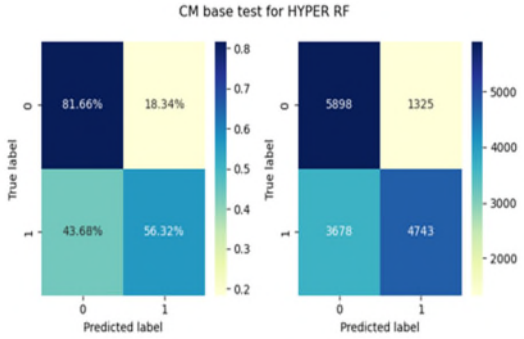
Figure A-22. Confusion tables and AUC-ROC graph for all trained models of IoT HealthCare dataset

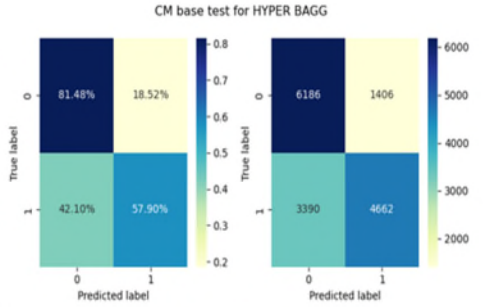

CM base test for HYPER XGB
81.20%
18.80%
41.88%
58.12%
6245
1446
3331
4622
True label
Predicted label

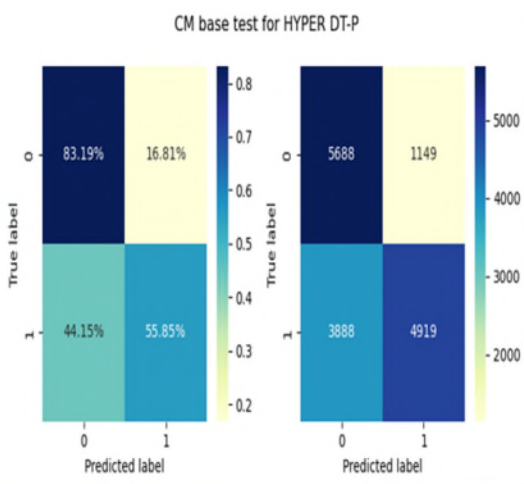
CM base test for HYPER SVM
86.92%
13.08%
58.11%
41.89%
1462
220
8114
5848
True label
Predicted label


CM base test for HYPER RNN
72.41%
27.59%
30.27%
69.73%
8318
3170
1258
2898
True label
Predicted label


CM base test for HYPER NB
69.31%
30.69%
45.79%
54.21%
7110
3149
2466
2919
True label
Predicted label


CM base test for HYPER RF
81.66%
18.34%
43.68%
56.32%
5898
1325
3678
4743
True label
Predicted label


CM base test for HYPER BAGG
81.48%
18.52%
42.10%
57.90%
6186
1406
3390
4662
True label
Predicted label


CM base test for HYPER DT-P
83.19%
16.81%
44.15%
55.85%
5688
1149
3888
4919
True label
Predicted label

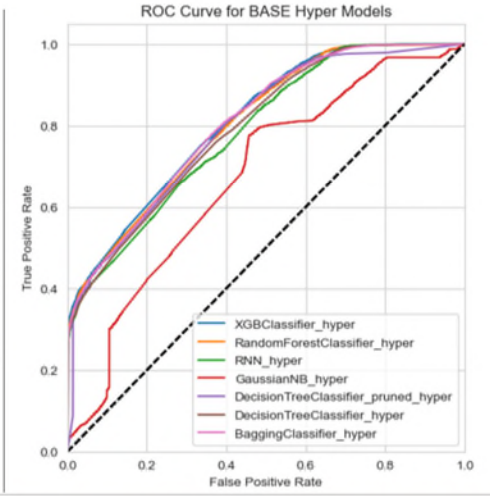
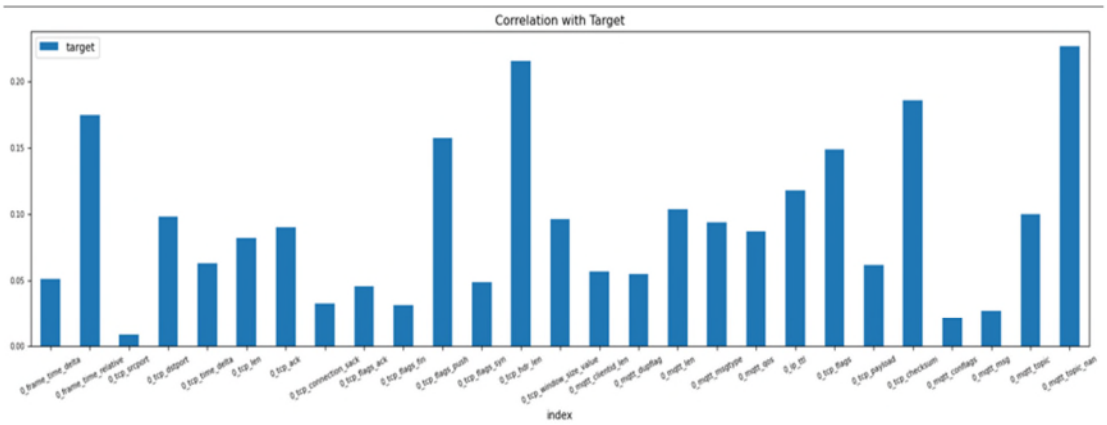
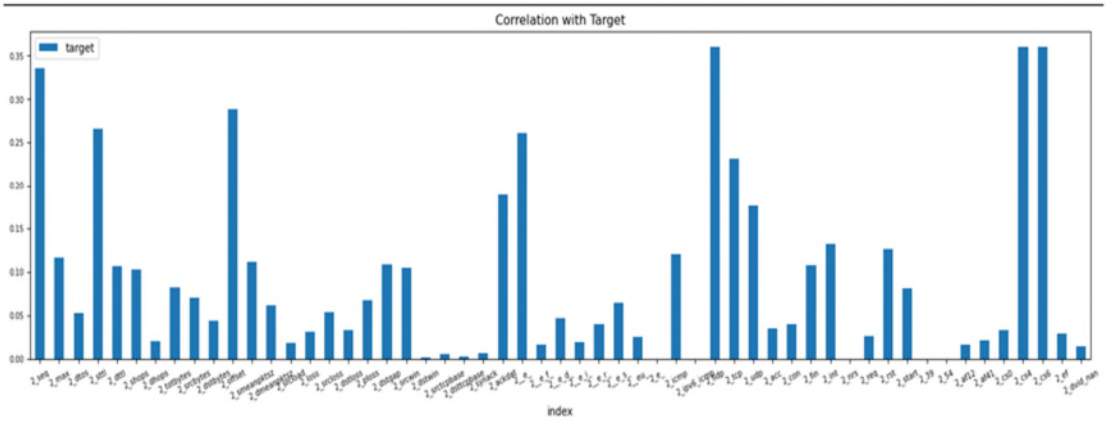

ROC Curve for BASE Hyper Models
True Positive Rate
False Positive Rate
XGBClassifier_hyper
RandomForestClassifier_hyper
RNN_hyper
GaussianNB_hyper
DecisionTreeClassifier_pruned_hyper
DecisionTreeClassifier_hyper
BaggingClassifier_hyper

Figure A-23. Confusion tables and AUC-ROC for all trained models of ToN-IoT dataset

Figure A-24. Confusion tables and AUC-ROC graph for all trained models of 5G-NIDD dataset


Correlation with Target
target
index


Correlation with Target
target
index

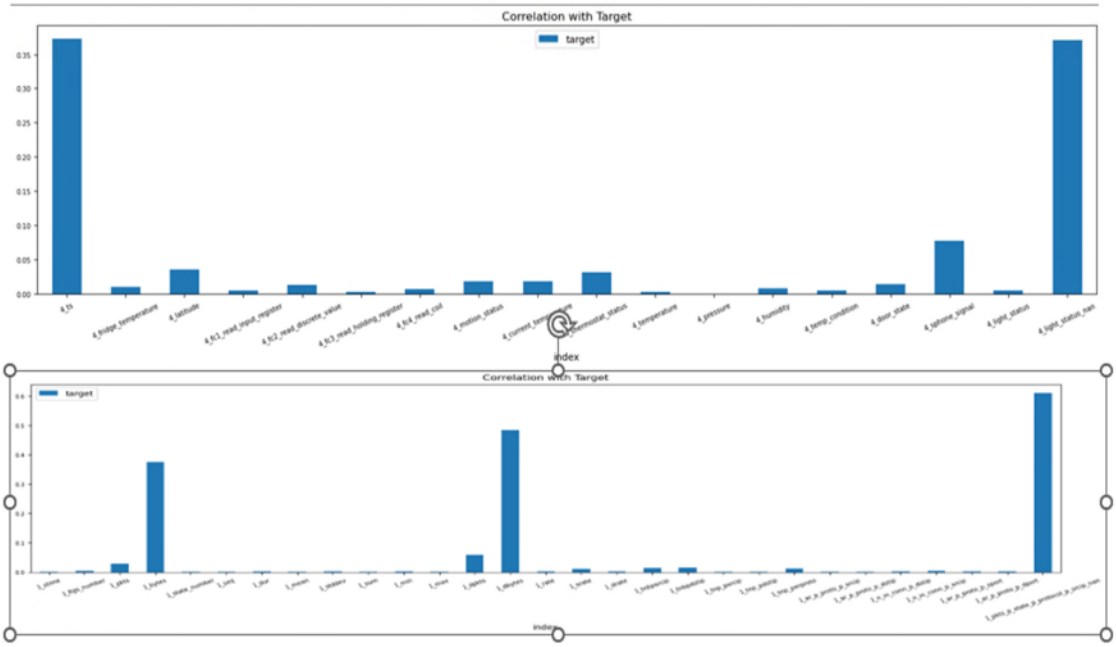

Correlation with Target
target
index
Correlation with Target
target

Figure A-25. Feature correlation with target variable for the combined dataset

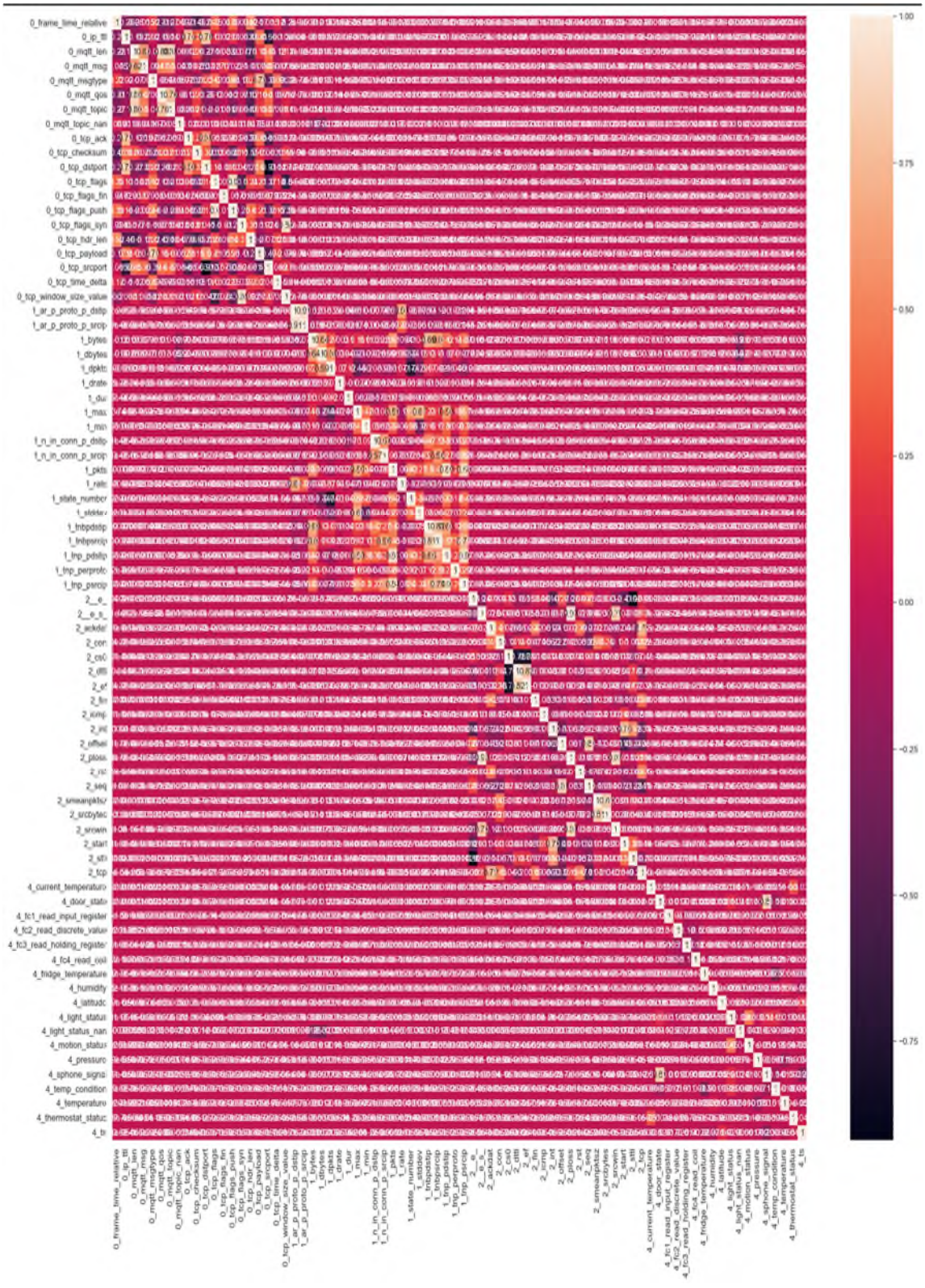


Figure A-26. Correlation matrix for the combined dataset

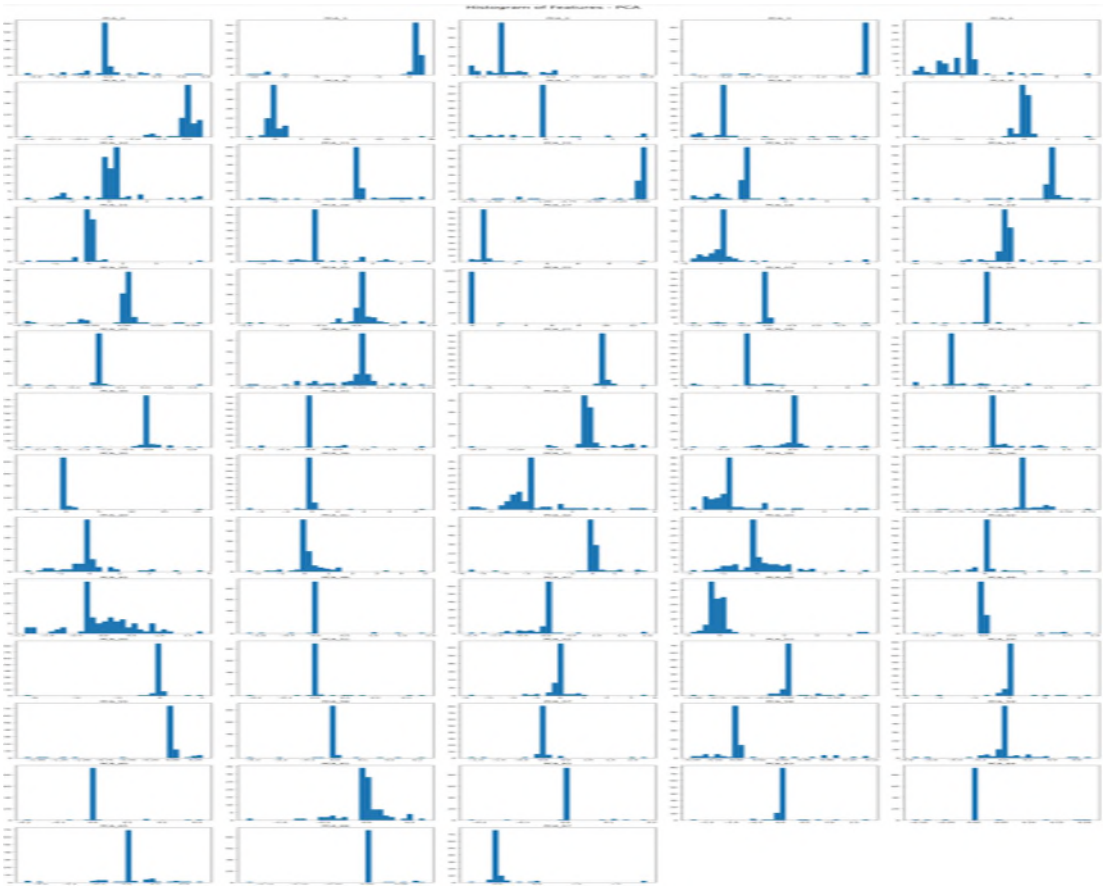


Figure A-27. Histogram of the combined dataset features after applying standardization and PCA techniques

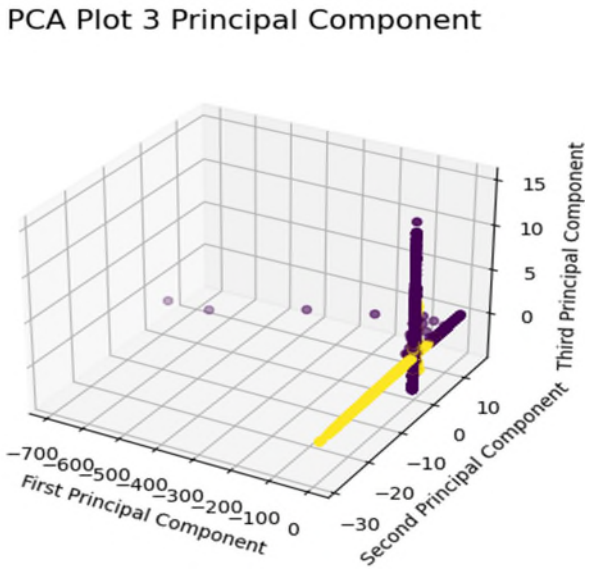


Figure A-28. The three principal components of the combined dataset

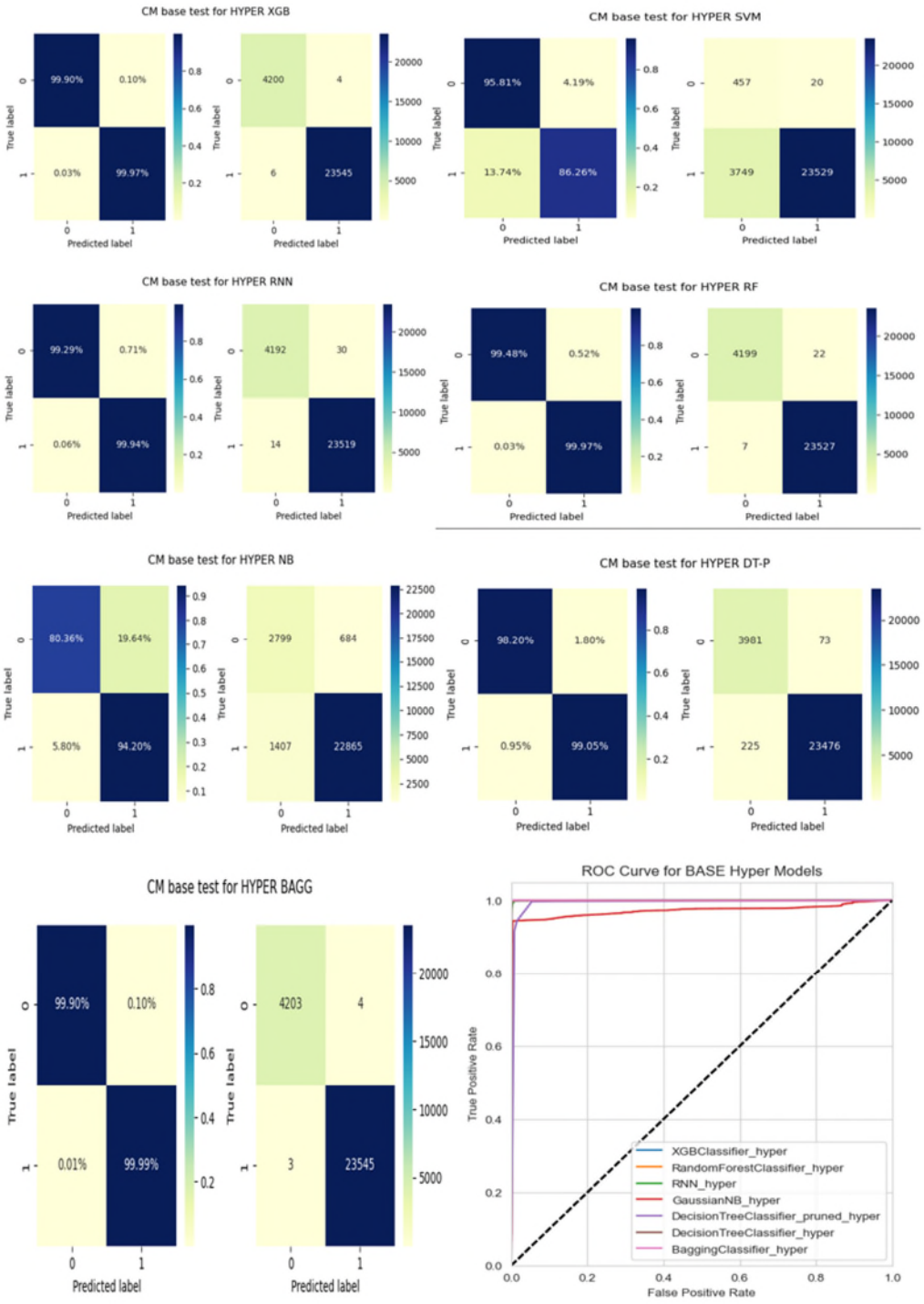


Figure A-29. Confusion tables and AUC-ROC graph for all trained models of the combined dataset

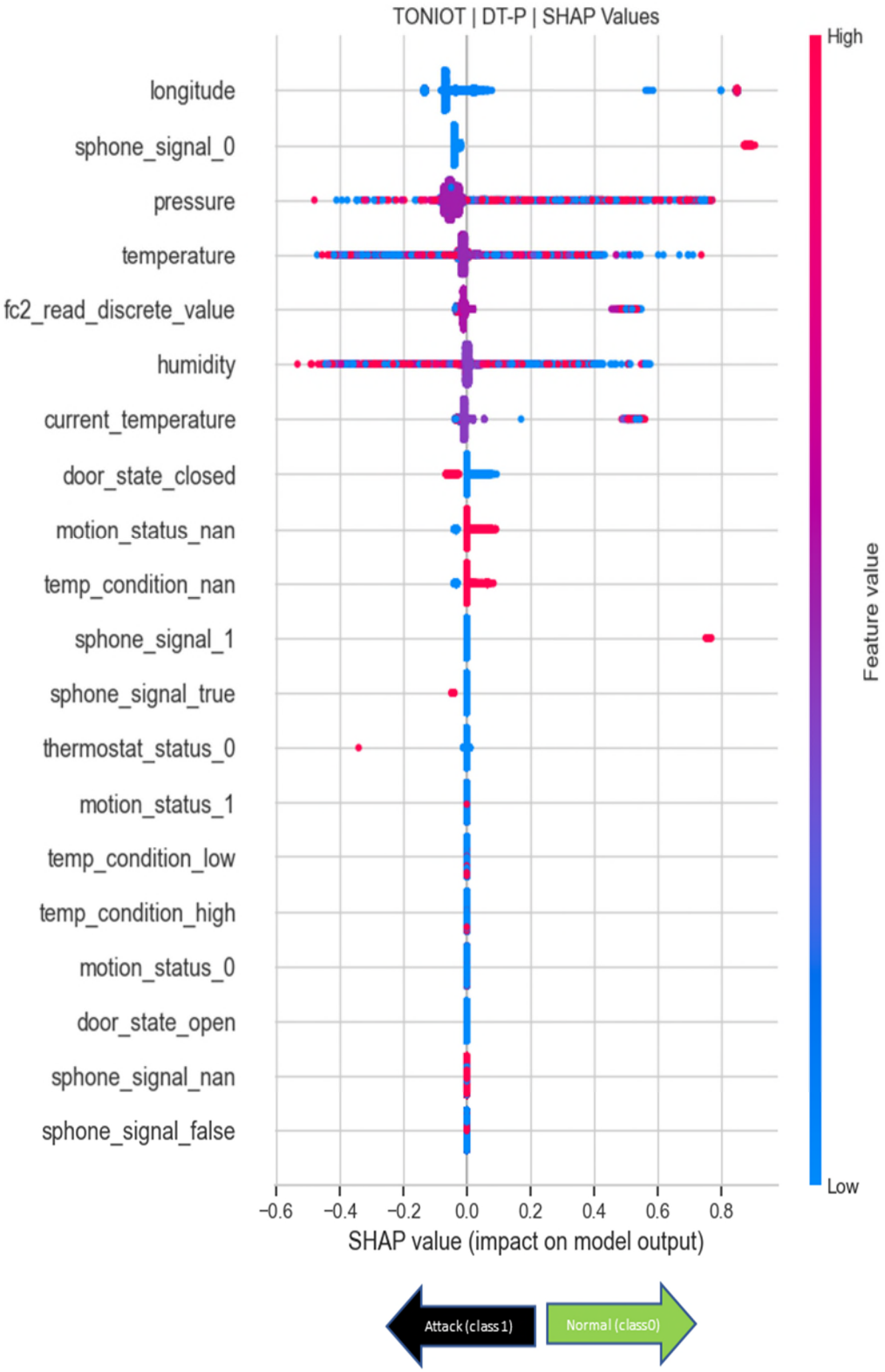


Figure A-30. Shapley values interpretation of DT model for ToN-IoT dataset

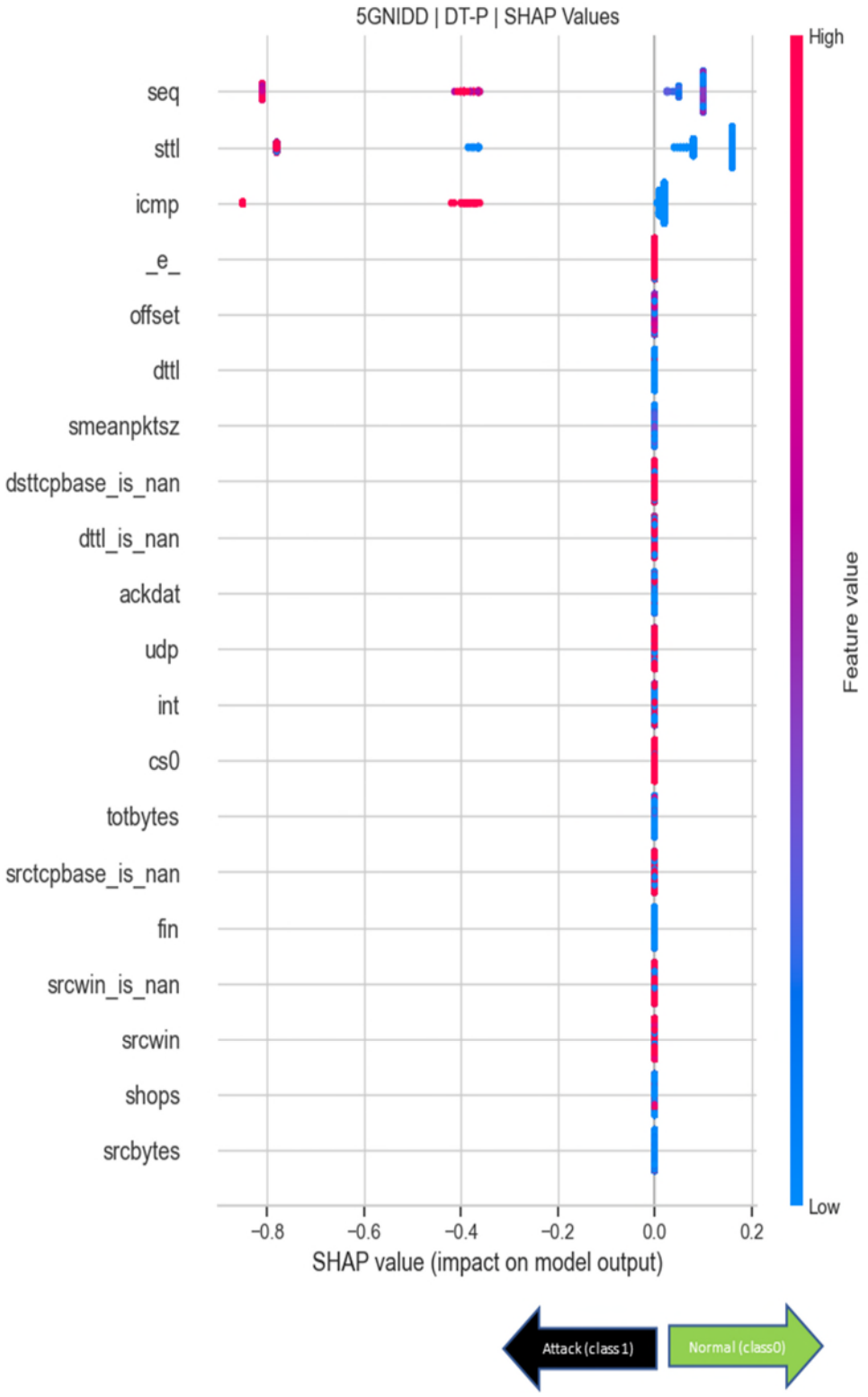


Figure A-31. Shapley values interpretation of DT model for 5GNIDD dataset

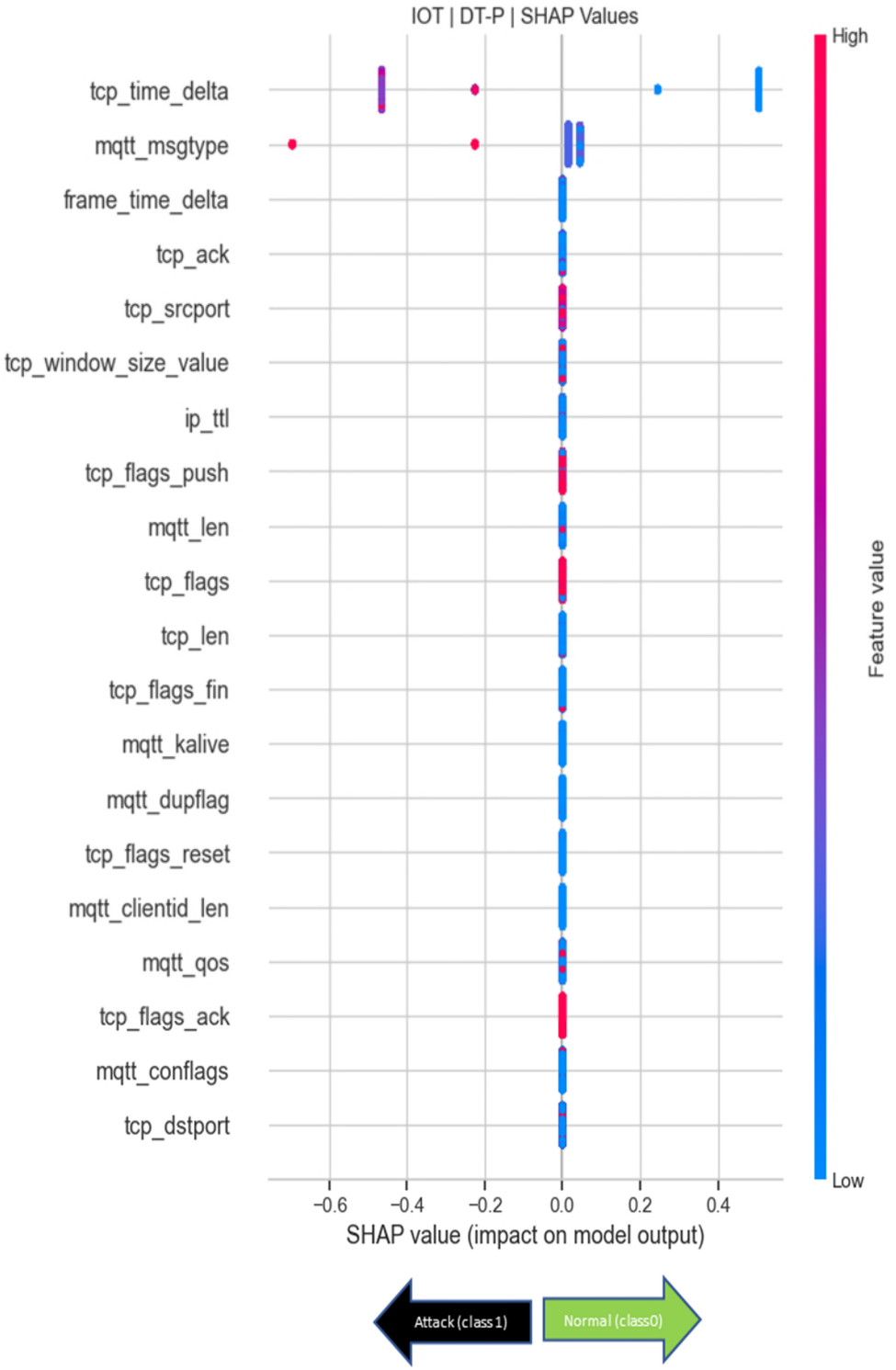

IOT | DT-P | SHAP Values
tcp_time_delta
mqtt_msgtype
frame_time_delta
tcp_ack
tcp_srcport
tcp_window_size_value
ip_ttl
tcp_flags_push
mqtt_len
tcp_flags
tcp_len
tcp_flags_fin
mqtt_kalive
mqtt_dupflag
tcp_flags_reset
mqtt_clientid_len
mqtt_qos
tcp_flags_ack
mqtt_conflags
tcp_dstport
−0.6
−0.4
−0.2
0.0
0.2
0.4
SHAP value (impact on model output)
High
Feature value
Low
Attack (class 1)
Normal (class0)

Figure A-32. Shapley values interpretation of DT model for HealthCare IoT dataset

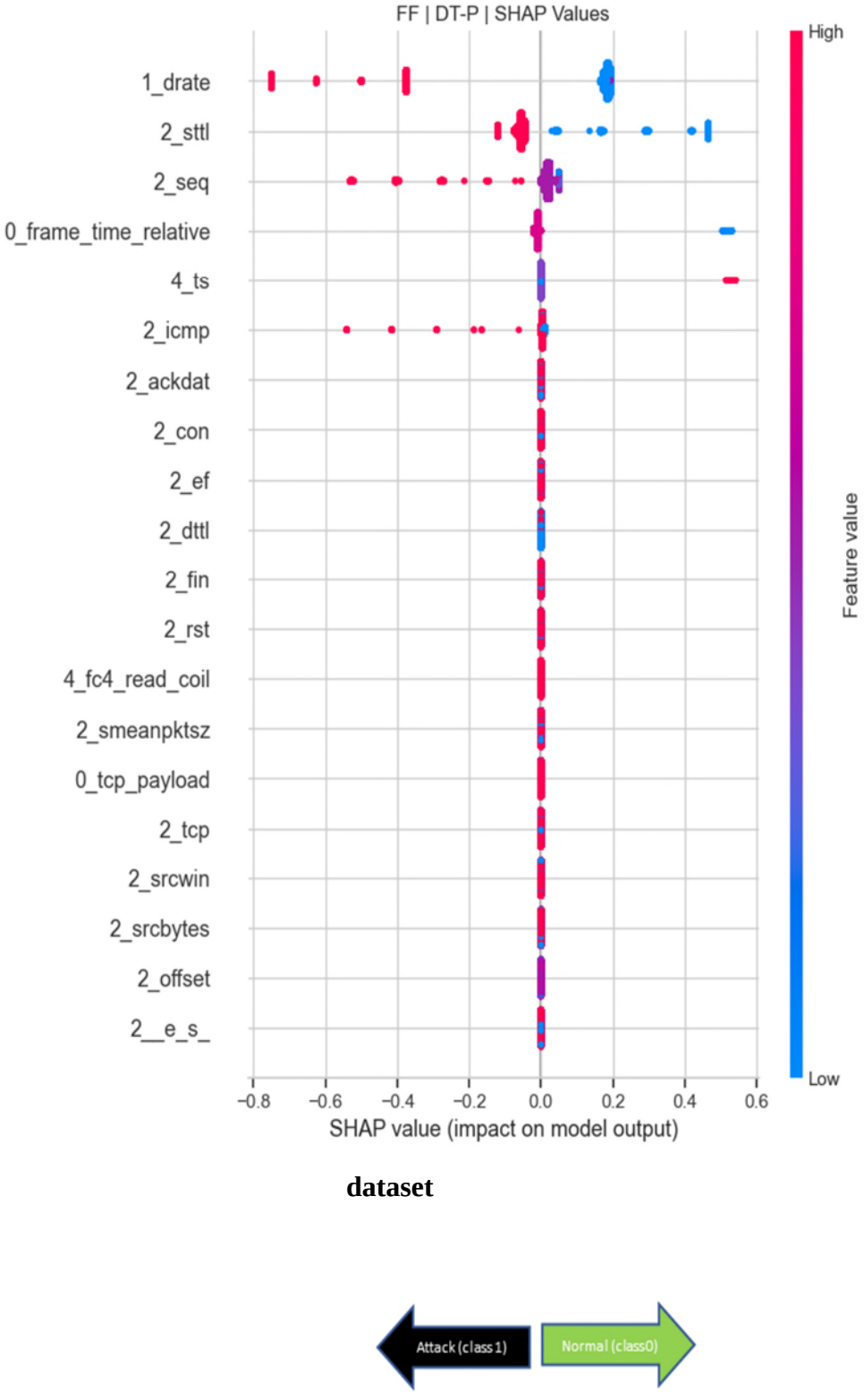


Figure A-33. Shapley values interpretation of DT model for the combined feature fused dataset

Table A-17. Model performance metrics

Metric	Equation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$ X 100
Precision	$\frac{TP}{TP + FP}$ X 100
Recall	$\frac{TN}{TN + FN}$ X 100
F1 - Score	2 X $\frac{Precision * Recall}{Precision + Recall}$
True Positive Rate (TPR)	$\frac{TP}{TP + FN}$ X 100
False Positive Rate (FPR)	$\frac{FP}{FP + TN}$ X 100

Table A-18. Datasets sampling size

Dataset	Size	Sample	Ratio
BoT-IoT	3668522	476907	0.13
IoT HealthCare	188694	24530	0.13
ToN-IoT	401119	52145	0.13
5G-NIDD	1215890	158065	0.13