7-2023

Cybersecurity Attacks Detection For MQTT-IoT Networks Using Machine Learning Ensemble Techniques

Sahar Mohamed Bukhari Abdelbasit
sa1364@rit.edu

Cybersecurity Attacks Detection For MQTT-IoT Networks Using Machine Learning Ensemble Techniques

By

Sahar Mohamed Bukhari Abdelbasit

A Thesis submitted

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Science

in

Computing Security

Supervised by

Dr. Wesam Almobaideen

Professor of Computing Security and Networking

Department of Electrical Engineering and Computing

Rochester Institute of Technology-Dubai Campus

United Arab Emirates

July 2023

RIT

Master of Science in

Computing Security

Thesis Approval

Cybersecurity Attacks Detection For MQTT-IoT Networks

Using Machine Learning Ensemble Techniques

Student Name: Sahar Mohamed Bukhari AbdelBasit

Dr. Wesam Almobaideen _____

Professor (Thesis Adviser)

Dept. of Electrical Engineering and Computing

Dr. Huda Saadeh _____

Assistant Professor (Committee Member)

Dept. of Electrical Engineering and Computing

Dr. Kevser Akpinar _____

Assistant Professor (Committee Member)

Dept. of Electrical Engineering and Computing

Acknowledgments

All gratitude belongs to the Mighty Allah, the benevolent and merciful, who granted me the capability to complete this thesis.

The successful completion of this thesis was made possible due to the invaluable guidance and support of numerous individuals, to whom I extend my sincere appreciation.

Firstly, I sincerely appreciate my academic supervisor Dr. Wesam Almobaideen for his expert advice, valuable feedback, and encouragement throughout the thesis. The successful completion of this thesis owes much to his expert guidance.

I want to sincerely thank the thesis committee members for their important advice and support during this journey. Thanks to Professor Kevser Akpinar and Professor Huda Saadeh for their continued support and assistance.

Finally, I want to convey my deepest appreciation to my family members, whose unwavering support and encouragement have been pivotal throughout my academic pursuits. They have been an invaluable source of inspiration and my biggest motivation.

Abstract

The Internet of Things (IoT) is one of the technical advancements that is progressing swiftly. which promises to be revolutionary soon. IoT systems are convenient due to its device centralized and computerized control. This technology allows various physical devices, home applications, vehicles, appliances, etc., to be interconnected and exposed to the Internet. On the other hand, it entails the fundamental need to protect the network from adversarial and unwanted alterations. Machine-to-machine protocols like Message Queuing Telemetry Transport (MQTT) are typically used by IoT devices to communicate. Numerous techniques to attack networks employ the lightweight messaging protocol known as MQTT (Message Queuing Telemetry Transport). Due to its heterogeneous nature and the lack of security approaches, the publish-subscribe strategy utilized by the MQTT protocol increases the number of potential network attacks. This thesis presents a novel approach to detecting cybersecurity breaches in MQTT-IoT networks using machine learning techniques.

We suggest a detection system to address the issue of cybersecurity threats in MQTT-IoT networks. Our method involves cleaning the data to pull out relevant features, training the ensemble machine learning models on these features, and then using these models to find anomaly behavior that could indicate a cyberattack.

We implemented our plan by using Machine Learning Ensemble techniques and Feature selection. To test our system, we ran many experiments using MQTT-IoT-IDS2020, a dataset that included both normal MQTT-IoT network activity and simulated attacks of different types.

Our experimental findings indicate that our detection system, grounded in machine learning, can identify cybersecurity threats on MQTT-IoT networks with notable accuracy, precision, F1-score, and recall.

The obtained results for binary and multiclass classification indicate that the proposed system can bring a remarkable layer of security. We show how Machine Learning Ensemble Techniques applied to small low-cost devices are an efficient and versatile combination characterized by a bright future ahead.

This thesis advances the application of machine learning methodologies in cybersecurity and contributes to the enhancement of security protocols within MQTT-IoT networks.

Contents

List of Figures

Listing of Tables

Listing of Acronyms and Abbreviations

Abbreviations	Meaning
AdaBoost	Adaptive Boosting
ACM	Association for Computing Machinery
AIDS	Artificial Intelligence and Data Science
AMD	Advanced Micro Devices
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
AUC	Area Under the Curve
API	Application Programming Interface
CART	Communication Access Realtime Translation
CoAP	Constrained Application Protocol
CIA	Confidentiality, Integrity, and Availability
CNN	convolutional neural network
CPU	Central Processing Unit
CV	Cross Validation
DCNN	Deep convolutional neural networks
DDOS	Distributed denial of service
DL	Deep Learning
DNN	Deep Neural Network
ELM	Extreme Learning Machine
FAR	False Acceptance Ratio
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GBM	Gradient boosted machines
GSM	A global system of mobile communication
GUI	Graphical User Interface
HIDS	Host-Based Intrusion Detection System
IBM	International Business Machines
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
KNN	K-Nearest Neighbour
LR	Logistic Regression
LSTM	long short-term memory networks
MITM	Man in the middle
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport

NB	Naïve Bayes
NIDS	Network Intrusion Detection System
OASIS	The Organization for the Advancement of Structured Information Standards
QoS	Quality of Service
PCA	Principal Component Analysis
PUBACK	Publish Acknowledgment
PUBCOMP	Publish Complete
PUBREC	Publish Received
PUBREL	Publish Release
PCAP	Packet Capture
PWN	Pwning
RBF	Radial Basis Function
RFID	Radio Frequency Identification
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SDN	Software-Defined Networking
SMOTE	Synthetic Minority Over-sampling
SQL	Structured Query Language
SSH	Secure Shell
SVM	Support Vector machine
TCP	Transport Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
VLC	VideoLAN Client
WEKA	Waikato Environment for Knowledge Analysis
WAN	Wide area network
WSN	Wireless Sensor Network
XGBOOST	Extreme Gradient Boosting
XMPP	Extensible Messaging and Presence Protocol

Chapter 1

In this chapter, we provide an introduction to the research topic and outline the objectives and significance of the study. We present this research's problem statement and motivation, emphasizing the need for an effective cybersecurity framework for MQTT-based IoT networks. The chapter's conclusion gives a summary of the contributions and thesis structure, setting the stage for the subsequent chapters that delve into the detailed analysis, methodology, experiments, and findings.

1.1. Introduction

Platforms for the Internet of Things (IoT) have revolutionized both our means of communication and our daily lives by enabling remote connections between IoT devices across a network architecture. This infrastructure collects data across several domains without involving humans or computers [1]. Our daily lives have changed significantly because more IoT devices are used in healthcare, industries, and smart homes [2]. Subsequently, the Government and people have faced increased cybersecurity threats and privacy violations [3].

CoAP, MQTT, AMQP, and XMPP application layer protocols enable IoT nodes to share data securely and reliably [4]. The MQTT protocol has been deployed in smart homes [5], agricultural IoT [6], industrial applications [7], and more. It is the most used publish-subscribe protocol because it has low bandwidth, memory, and packet loss. [8].

IBM launched the MQTT protocol as a message push protocol [9]. It is a lightweight client-server message transmission protocol perfect for connecting machines and the Internet of Things (IoT). It was developed to reliably transmit a message under conditions of low network bandwidth and lengthy network latency. In addition, MQTT uses a publish/subscribe communication paradigm. In this configuration, Subscribers are the recipients of messages generated by publishers. Subscribers and publishers can only communicate indirectly through the broker who serves as the server. The role of this component is to accept messages from publishers and then distribute them to the respective subscribers. [10], [11].

Because machine learning (ML) and artificial intelligence (AI) algorithms are getting better and better so quickly, it is now possible to monitor networks and detect incoming cyber-attacks [12]. Machine Learning (ML) and Deep Learning (DL) models are now leading technologies, with potent capabilities for overcoming situations where typical IDS are inadequate [13].

Instead of explicitly programming devices, Machine learning is a type of artificial intelligence that uses algorithms to teach machines how to do things and facilitates their ability

to learn from experience[14]. As a result, Machine Learning methods successfully utilized many areas of Information and Communication Technology (ICT), especially those related to cyber-security [15].

Because of limited resources, low power, and connection of IoT devices, traditional network intrusion detection systems (NIDS) may be less effective in IoT systems. Intrusion detection is defined by the National Institute of Standards and Technology (NIST) as monitoring activities on a computer system or network, analyzing these actions for signs of breaches, and reporting the malicious data to the network administrator [16].

Intrusion Detection Systems (IDSs) operate within two principal categories. The Signature-Based Detection method functions by comparing data observed by the IDS to predefined intrusion patterns, underscoring its robust reliability and efficacy. This approach has gained significant traction owing to its implementation in widely used tools such as Snort [17] and Suricata [18]. However, this method has one notable limitation; it is only equipped to identify known threats cataloged in a database.

Contrastingly, Anomaly Detection operates by establishing a baseline of the system's typical behavior and subsequently identifying any deviations within the surveilled data. This method's capacity to detect novel threats is noteworthy, yet it is frequently associated with an excessive generation of false positives.

Much research has been focused on Anomaly-Based IDSs in the last twenty years. Their propensity for identifying unknown threats is particularly crucial given the current landscape, where intrusion attempts are increasing in frequency and becoming progressively diverse in nature.

Problem Statement

IoT platforms generate valuable data that must be safely communicated and analyzed [19]. Additionally, the lack of security in IoT devices has highlighted the need to build a more effective and secure infrastructure by guarding against numerous security flaws, threats, and cyberattacks in IoT networks [20].

MQTT (Message Queuing Telemetry Transport) [10] has gained significant popularity in IoT (Internet of Things) networks due to its lightweight and efficient messaging protocol. Nevertheless, the extensive embrace of the MQTT protocol has concurrently escalated apprehensions regarding the security integrity of IoT systems. Networks utilizing MQTT are prone to a multitude of cybersecurity threats, such as unsanctioned access, data infringements, and denial-of-service (DoS) assaults. These menacing activities can potentially jeopardize the availability, confidentiality, and integrity of IoT devices, thereby resulting in serious repercussions.

In addition, every MQTT-IoT application relies heavily on MQTT brokers. However, because of their transparency, they are susceptible to cyber-attacks. Every MQTT-IoT

application relies heavily on MQTT brokers. However, their transparency makes them easy to cyber-attack [22]. Furthermore, we are communicating sensitive data/information over MQTT-IoT, which raises the need to secure and protect this information.

Identifying cybersecurity breaches in MQTT-IoT networks is a considerable hurdle. Existing approaches for attack detection often rely on rule-based systems or signature-based methods, which have limitations of scalability, and handling of complex attack scenarios [17], [23].

Our proposed research holds significant importance for the cybersecurity of MQTT-IoT networks. By harnessing the power of ensemble techniques, detection accuracy and resilience against various cyber-attacks can be significantly enhanced. Additionally, the findings of this research can benefit industries, organizations, and policymakers involved in securing IoT infrastructures against evolving cyber threats.

1.2. Thesis Importance and Objective

This thesis aims to design a robust cybersecurity framework to threat detection and mitigation in MQTT-focused IoT networks. The specific goals of this research are as follows:

1. Investigate the cybersecurity challenges and vulnerabilities in MQTT-IoT networks: This study examines the existing condition of MQTT-IoT networks and pinpoint the likely security risks and weaknesses these networks might face.
2. This thesis intends to create a dependable and precise detection system using machine learning ensemble methods. The goal is to recognize and categorize different cybersecurity breaches in MQTT-IoT networks—the system is designed to detect these attack patterns, ensuring proactive security measures.
3. Evaluate the proposed solution's performance: The developed detection system's performance is assessed through extensive experimentation and evaluation. This research will analyze the system's accuracy, efficiency, scalability, and resilience against cyber threats.

There is an enormous amount of reviewed literature about applying machine-learning algorithms to cyber-attack detection, and it has been proven to be an efficient and reliable approach to the problem. However, fewer studies have been conducted regarding machine-learning-based common attacks such as brute force attacks.

1.3. Contributions

By focusing on detecting and mitigating cybersecurity attacks, this thesis will contribute to the following:

1. Enhancing the security posture of MQTT-based IoT networks: The proposed detection system enables network administrators and security professionals to determine dangers and take appropriate action promptly. Protecting the privacy and integrity of IoT devices as well as the broader network architecture, the system works to prevent illicit access, compromises of data, and system failures by proactively spotting threats..
2. Pushing the boundaries of machine learning-driven cybersecurity: This study adds value to the domain of machine learning within cybersecurity by investigating the potency of ensemble methods in detecting attacks on MQTT-IoT networks. The findings and methodologies developed in this thesis will contribute to the growing body of knowledge in cybersecurity and provide insights into building robust defense mechanisms against evolving cyber threats.
3. Fostering trust and adoption of MQTT-IoT networks: As security concerns remain a significant barrier to the widespread adoption of MQTT-IoT networks, the outcomes of this research will provide valuable insights into addressing these concerns. This thesis aims to enhance MQTT-based IoT networks' overall security and dependability, promoting their broader adoption in sectors such as smart homes, healthcare, transportation, and industrial automation. This enhancement is achieved through the introduction of an effective detection strategy.
4. We undertake a comprehensive analysis encompassing both binary and multiclassification. By employing a dual approach, we ensure a thorough examination of the security aspects within the MQTT protocol. The binary classification distinguishes between benign and attack classes, effectively identifying malicious activities. Simultaneously, the multiclassification extends this analysis to categorize and differentiate various types of MQTT attacks, providing deeper insights into the diverse attack vectors present in the protocol. Our work aims to enhance the overall understanding and mitigation of MQTT-based threats through this combined approach, facilitating more robust security measures."

1.4. Thesis Organization

The structure of the remaining parts of the thesis is outlined below:

Chapter 2: This chapter delves into the essence of IoT, highlighting the driving technologies behind its emergence and laying out its typical applications and components. A review of security solutions for IoT, as well as associated challenges, is provided. Basic principles of computer security relevant to this thesis are discussed. The chapter also reviews previous research related to the topic, differentiating our approach from existing studies.

Chapter 3: This chapter centers on the MQTT Protocol, including its headers, messages, and overarching architecture. A comprehensive discussion on MQTT security and associated attacks is also provided in this segment.

Chapter 4: This chapter delves into Machine Learning, exploring its concepts and various applications.

Chapter 5: This chapter delves into a comprehensive literature review. Additionally, the advantages and constraints of the suggested systems are thoroughly examined.

Chapter 6: This chapter provides a synopsis of the dataset utilized in our experiments and detailed descriptions of the dataset preprocessing and feature selection procedures. In addition, the Machine Learning models used in our experiment are explained. Ensemble Techniques are explained in this chapter, followed by the methodology and experimental setup.

Chapter 7: The experiment result and analysis of Binary and Multiclass classification are presented.

Chapter 8: We concluded our thesis in this chapter, followed by future work.

Chapter 2

Concepts and background

This particular chapter offers a comprehensive exploration of the foundational principles of the Internet of Things (IoT). The definition of IoT, prospective applications, and the architectural framework, including key elements and protocols used within, are all clarified.

2.1 Introduction to IoT

The term "Internet of Things," or simply "IoT," describes the expansive web of tangible objects worldwide, interconnected via the Internet, collectively collecting and sharing information. It fosters a transformation wherein Internet-capable devices morph into an interconnected ecosystem wherein digital data is ubiquitously accessible at all times [24]. IoT devices encompass tangible items ranging from tiny to large-scale machinery, and they can communicate effortlessly with one another over the Internet without the need for human involvement [25].

2.1.1 IoT Characteristics

The Internet of Things, or IoT, combines different hardware and software technologies into one system. These solutions leverage information technology, encompassing hardware and software capable of storing, retrieving, and processing data [25].

The Internet is the leading way these devices communicate, using technologies like RFID and WSNs. These technologies use sensors to keep an eye on the surroundings. These devices have limited processing power, memory, storage, and battery life [26].

There are a few crucial features of the IoT:

Interconnectivity: In the realm of IoT, it signifies that any object can establish an internet connection and interact with other devices.

Things-related services: It can offer services related to the devices while protecting privacy and keeping things consistent.

Heterogeneity: This implies that even while IoT devices may have different underlying hardware and networks, they still possess the ability to communicate with one another.

Dynamic changes: Devices in the IoT realm can frequently shift states, including sleep and wake patterns, connectivity status, and contextual factors like location and velocity.

Enormous scale: The number of devices that require management and intercommunication will far outnumber the number of devices that are currently online.

Safety: While utilizing IoT, safety should always be a priority. This encompasses both the protection of our personal information and our physical safety. It emphasises how crucial it is to protect networks, and the data that flows between them.

Connectivity: This allows devices to access the network and be compatible with each other. Getting on a network is called accessibility, while the ability to share and understand data is called compatibility.

2.1.2 IoT Technologies

The Internet of Things (IoT) is a crucial bridge linking various products with the digital realm. The web of interconnected devices continues to grow, fueled by technological progress in sensors, smartphones, cloud technology, and communication capacities. It constitutes a network comprising diverse physical entities, including vehicles, machinery, household appliances, and beyond. These entities leverage multiple technologies to facilitate data exchange over the Internet [25]. Table 2.1 explains the technologies that underpin the IoT concept.

Table 2. 1: The Technologies of IoT

IoT Technologies	Examples
Communication and Networking Infrastructure	GSM and WIFI are often utilized in IoT setups for connectivity and data exchange.
Identification	WSN, RFID
Hardware and Software	Smart devices with enhanced inter-device communication

Technologies for Networking and Communication:

Tools such as Bluetooth and ZigBee facilitate device connectivity. It is crucial that communications between these interconnected devices are fortified with robust security measures to provide users with assured confidence in their network's safety and reliability.

Device Identification Technologies:

It is crucial to identify each connected device within an IoT ecosystem. To achieve this unique identification, technologies like RFID and WSN are employed.

Hardware and Software Technologies:

Intelligent devices that foster enhanced inter-device communication will pave the way for systems with greater intelligence and autonomy, promoting the swift deployment of IoT applications.

2.1.3 IoT Applications

The Internet of Things (IoT) incentivizes many applications to augment daily human activities [27]. Diverse applications are realized through various sensors, intelligent devices, servers, and more, as seen in Figure 2.3 enumerates a range of applications incorporating IoT concepts and platforms.

The concept of **a smart home** is one such application. It encapsulates an array of intelligent devices such as smart locks, baby monitors, and fire detectors installed within a household, communicating over wireless channels. These home appliances can be accessed remotely via a home gateway**.**

Innovative healthcare is another notable application of IoT, which facilitates the gathering, transmitting, and preserving of a patient's physiological data. For example, medical sensors can record a patient's heartbeat and transmit this data to a hospital's server for evaluation and tracking.

Within the domain of **intelligent transportation**, a vast system of smart vehicles has the capability to communicate with one another, with infrastructure elements, and with individuals on foot through wireless channels. These automobiles can gauge real-time traffic situations, adjust speed, and share information to ensure efficient and secure driving experiences.

Environmental systems provide distant management of factors like temperature, humidity, water levels, soil wetness, and specific climate conditions to optimize production standards and reduce economic setbacks. In advanced agriculture, sensors can be attached to animals, offering insights into their behavior and ensuring their well-being.

The **industrial** sector has embraced IoT, giving rise to the smart industry. The main goal of IIoT is to improve oversight of the manufacturing process, data, and emerging challenges, guaranteeing the end products' efficacy and dependability.

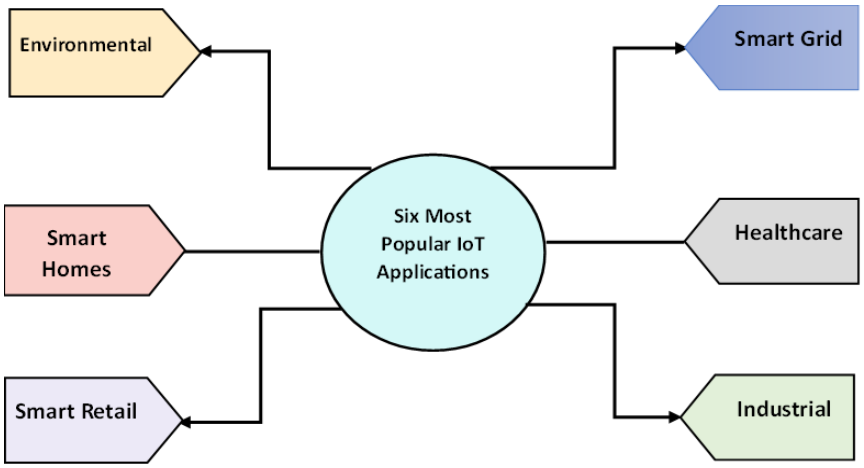

Figure 2. 1: IoT Applications

Smart retail: This application enables monitoring product storage facilities or during transit. Sensors can be affixed to retail items to keep tabs on their status. Numerous smart shopping systems have been developed to offer sophisticated customer services and attract a more extensive customer base.

A smart grid: This is a common use of IoT, made to measure, watch over, and control how much electricity is used. It helps users handle their electricity use trustworthy, saves energy, and lowers the chance of problems with the power grid.

2.1.4 IoT Architecture

The Internet of Things (IoT) is built on a variety of technology foundational layers that make up its structural structure. It delineates the interrelation of diverse technologies and the communicability, modularity, and configuration of IoT implementations in assorted scenarios. As depicted in Figure 2.2, this architecture is generally stratified into layers, which facilitate system administrators in evaluating, observing, and preserving the system's integrity. The IoT architecture essentially entails a four-stage process where data is centralized from devices linked to sensors, traverses a network, and is subsequently directed to the cloud for processing, analysis, and storage [28].

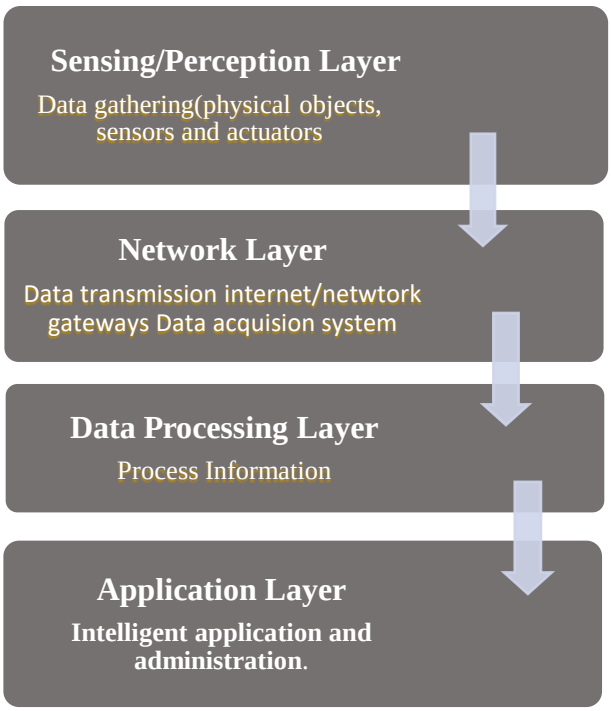

Figure 2. 2: IoT Architecture

The Perception/Sensing Layer

The physical layer is the foundation of the internet of things. This layer encompasses many IoT devices like sensors, each equipped to gather, process, and relay information through the network. These devices can be linked via wired or wireless connections and utilize intelligent technologies to collect data.

The Network Layer

This Layer is a critical component as it encompasses various communication technologies enabling IoT devices' connectivity. The Network Layer also includes Data Acquiring Systems (DAS) and Internet/Network gateways, which collect, aggregate, and convert analog data into digital data.

The Processing Layer

The data undergoes a series of transformations and computations within this layer to extract meaningful insights. Moreover, it is an intermediary stage bridging the gap between data collection and application layers. This involves readying the data for subsequent utilization by software applications, which oversee, control, and implement subsequent actions guided by the interpreted data.

The Application Layer

This Layer directly interacts with IoT users through essential facilities. This tier receives refined data from the Network Layer and accommodates a diverse spectrum of applications, encompassing smart homes, intelligent retail, and advanced power grids.

2.1.5 IoT Challenges

Even though IoT devices manage and produce a substantial amount of data, they are typically inexpensive. Consequently, these devices tend to possess limited computational power, storage capacity, and memory resources. Similarly, the software implemented in the IoT devices is based on open-source solutions or obsolete software and sometimes even with faulty software. In a worst-case scenario, we would have one IoT device plugged into a network with software with no security patches or updates, without knowing which ports are open and visible from the internet, and with no access control or credentials. IoT devices, when connected to public networks, can become susceptible to attacks in the absence of adequate security measures. Due to this vulnerability, private networks may experience unforeseen disruptions that jeopardize service accessibility, the security of data, and user safety.

2.2 Computer Security Fundamentals

Computer security fundamentals are the core principles and practices that aim to protect the various hardware and software components, known as assets, within a computing environment. These fundamentals are centered around ensuring the confidentiality, integrity, and availability of information for authorized users [29].

Confidentiality: Confidentiality focuses on maintaining the secrecy and privacy of assets, allowing only authorized individuals to access them. This involves implementing measures to control physical and technical access, classifying data, enforcing policies to keep workspaces clean and secure, establishing confidentiality agreements, implementing strong password policies, defining guidelines for employee IT use, and providing training to detect and prevent social engineering attacks [29].

Integrity: Integrity ensures that assets can only be modified by authorized users and aims to preserve the accuracy and unaltered state of information. Unauthorized modifications, whether deliberate or accidental, can compromise the integrity of data. To maintain integrity, organizations employ preventive measures to protect against fraudulent changes, both in physical and digital documents [30].

Availability: Availability is concerned with providing authorized users with timely access to information and services as needed [30]. To ensure availability, organizations implement backup procedures, adopt business continuity management (BCM) practices, and establish disaster recovery systems that allow for the duplication of essential services and applications,

even in the face of accidents, natural disasters, or intentional sabotage. Table 2.2 summarizes the main objectives or goals of computer security.

Table 2. 2: Goals of Computer Security

Objective	Description
Confidentiality	Ensuring that information is kept confidential and accessible only to authorized individuals or systems. This includes implementing access controls, data encryption, and secure communication channels.
Integrity	Maintaining information's precision, coherence, and dependability across its entire lifecycle. This entails preventing illicit modifications, detecting and mitigating data corruption, and maintaining the integrity of systems and processes.
Availability	Guarantee that data and amenities are available to and usable by approved users at the appropriate time. This includes implementing redundancy and backup strategies, disaster recovery plans, and mitigating denial-of-service (DoS) attacks.

2.2.1 Security in IoT

IoT integrates different technologies, which means it inherits each individual technology's security vulnerabilities [31]. Furthermore, the sheer scale of IoT connectivity, with billions of devices expected to be interconnected, means that a vast amount of data will be exposed to the Internet. This increased exposure creates a fertile ground for security attacks, including eavesdropping and data tampering. As a consequence, user privacy becomes increasingly at risk.

IoT devices are vulnerable to cyberattacks for three primary reasons:

- **Complexity and Heterogeneity:** The IoT ecosystem consists of a vast array of interconnected devices with different architectures, operating systems, and communication protocols. This complexity and heterogeneity make ensuring uniform security across all devices challenging. Each device may have unique vulnerabilities, and patching or updating them becomes a complex task, leaving vulnerabilities unaddressed [32].

- **The limited computational power of IoT devices** can pose challenges in implementing robust security measures, executing complex applications, causing delays in response time, and hindering firmware and software updates. Considering these limitations while creating and implementing IoT systems is crucial to ensure optimal performance and top-notch security [33].

- **Firmware and Software Updates:** IoT devices with limited computational power may face challenges in performing firmware and software updates. Updating the device's firmware or applying security patches requires sufficient computational resources to handle the update process. If the device's computational power is insufficient, it may not be able to handle these updates effectively, leaving it vulnerable to security vulnerabilities or lacking the latest features and improvements.

2.2.2 Security Attacks in IoT Ecosystem

Due to its interconnectedness and the wide variety of devices it includes, the IoT ecosystem is vulnerable to several security assaults. In the IoT environment, some frequent security assaults include: [34]

Physical Attacks:

In Physical attacks, attackers may manipulate or steal devices to gain control, extract sensitive data, or disrupt device functionality. Physical attacks can be carried out through techniques such as device tampering and physical theft,

Network Attacks:

Network attacks target the communication infrastructure of IoT devices [35]. Examples include:

- Attacks involving a Man-in-the-Middle (MitM): A malevolent actor expropriates and manipulates the conversation, permits them to eavesdrop, modify data, or impersonate trusted entities.
- Attacks using distributed denial of service (DDoS), which compromise, forming a botnet, flooding a target network or system with excessive traffic, and rendering it inaccessible to legitimate users.

Firmware Attacks:

Firmware attacks focus on exploiting vulnerabilities in the firmware [36].

Examples include:

- Exploiting Firmware Vulnerabilities: Attackers leverage security vulnerabilities in the firmware to gain unauthorized access, execute arbitrary code, or tamper with device behavior.
- Supply Chain Attacks: Malicious actors compromise the manufacturing or distribution process of IoT devices, injecting malicious firmware or components into devices. This allows them to gain unauthorized access or control over the devices later on.

Encryption Attacks:

Encryption attacks aim to bypass or exploit weaknesses in encryption mechanisms used for securing data in IoT devices [37]. Examples include:

- Cryptanalysis: Attackers employ cryptographic analysis techniques to break encryption algorithms or discover vulnerabilities, allowing them to decipher encrypted data.

- Side-Channel Attacks: Attackers exploit information leakage from the physical implementation of encryption, such as power consumption or electromagnetic emissions, to deduce encryption keys or sensitive information.

2.2.3 Impact of Cyber Attacks on IoT

Cyber-attacks on IoT can have far-reaching consequences that can impact various aspects of individuals' lives, organizations, and even critical infrastructure. Some notable consequences of cyber-attacks on IoT include:

- **Data Breaches:**

  Data breaches in IoT occur when sensitive information collected or transmitted by IoT devices is accessed or exposed without authorization. Research shows that attackers can easily obtain passwords, credit card information, or other confidential information from IoT devices through techniques such as brute force attacks and malware injection [38]. In a corporate setting, cyber-attacks on IoT devices such as industrial sensors can be used to steal intellectual property or sensitive business data [39].

- **Physical Harm:**

  In certain IoT deployments, cyber-attacks can have physical consequences. For instance, in critical sectors like energy, transportation, or healthcare, attacks on IoT systems can disrupt essential services, leading to transportation disruptions or compromised patient safety [40].

- **Disruption of Services:**

  Cyber-attacks can disrupt the normal operation of IoT devices and services. This can lead to service outages, rendering devices temporarily or permanently unusable. For example, DDoS attacks targeting IoT devices can overload networks or cloud infrastructures, causing service disruptions for both individuals and organizations [41].

- **Reputation Damage:**

  Cyber-attacks on IoT can result in reputation damage for organizations. Breaches and vulnerabilities in IoT devices can erode brand trust, decrease customer confidence, strain business relationships, lead to legal consequences, and provide a competitive advantage to more secure competitors [42].

- **Financial Losses:**

  Cyber-attacks on IoT can result in significant financial losses for both individuals and organizations. In cases of ransomware attacks, victims may be extorted to pay a ransom to regain control of their compromised devices or to prevent the release of sensitive data. Furthermore, the costs associated with incident response, recovery, and potential legal consequences can be substantial [42].

2.2.4 Cyber-Attacks Mitigation

Mitigating cyber-attacks requires a comprehensive approach that combines proactive measures, ongoing monitoring, and effective incident response [43]. The strategies discussed below can help secure IoT devices and prevent cyber-attacks.

- **Secure Communication:**

  One of the most effective ways to mitigate cyber-attacks on IoT devices is to use secure communication protocols. This includes implementing encryption, authentication, and access control mechanisms [44]. Secure communication ensures that data is encrypted and transmitted safely, making it harder for attackers to intercept and steal sensitive information.

- **Device Authentication:**

  Institute comprehensive security measures across the entire spectrum of the IoT ecosystem. This entails the establishment of robust device authentication protocols, implementing data encryption protocols for both transit and storage, regular and punctual deployment of security updates and patches, and adopting secure coding practices throughout the developmental phase of IoT devices [45].

- **Regular Software Updates:**

  Maintain the currency of all IoT devices, firmware, and software by consistently integrating the most recent security updates and patches. Regularly assess and implement patches provided by vendors to rectify acknowledged vulnerabilities.

- **Network Segmentation:**

  Segregate IoT devices into discrete sections of the network, isolating them from critical systems and sensitive data. This helps contain potential attacks and limit the impact of compromised devices [46].

2.3 Intrusion Detection Systems (IDS)

IDS for IoT refers to security systems designed to monitor IoT networks for any suspicious activities or breaches. These systems analyze and identify potential threats to the IoT infrastructure, enabling appropriate defensive measures to be taken to protect the network's integrity, confidentiality, and availability [47]. Figure 2.3 illustrates the two broad classifications under which intrusion detection systems (IDS) can be categorized [48].

Host-based Intrusion Detection System (HIDS):

It is implemented on specific devices or hosts within a network. It vigilantly observes and assesses the internal operations, in tandem with the network traffic that are routed through its network ports.

Network-Based IDS (NIDS):

NIDS, or Network-based Intrusion Detection System, constitutes a security mechanism strategically situated within a network to oversee incoming and outgoing traffic to all devices encompassed by the network. Its purpose is to identify potentially unauthorized or malicious actions through the scrutiny of network traffic and the surveillance of numerous hosts concurrently.

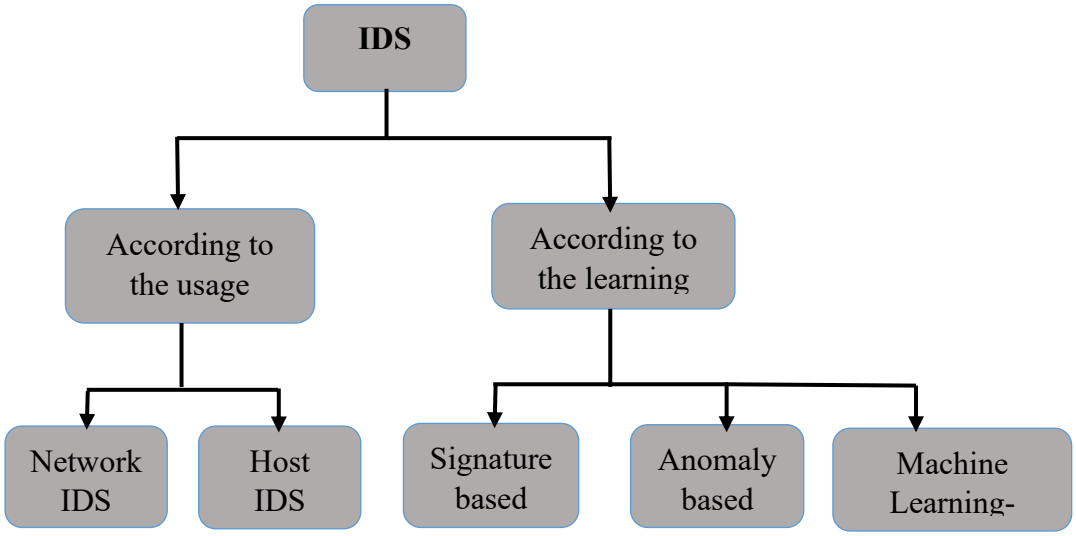

Figure 2. 3: Categorization of Intrusion Detection Systems (IDS)

It can be categorized according to their learning methods into the following types [49]

- **Anomaly detection**
  Anomaly-based IDS: These systems establish a standard of acceptable behavior for the system or network, monitor for activities that significantly deviate from this normal baseline. These deviations are considered potential threats.

- **Signature-based detection**
  These systems detect known threats using predefined rules or patterns (signatures). They compare these patterns against observed events to identify potential security breaches.

- **Machine learning IDS**
  They are used to anticipate and recognize malicious activities. These systems undergo training using a network traffic dataset, enabling them to categorize incoming traffic as either typical or malicious, drawing insights from the patterns acquired during training.

Consequently, IDSs designed for IoT devices must prioritize efficiency and minimal resource utilization. Furthermore, these IDSs need to possess scalability to accommodate expansive IoT networks composed of a multitude of devices [50].

Chapter 3

In this chapter, we explore the MQTT, a lightweight and efficient communication protocol designed for the Internet of Things (IoT). We discuss MQTT's simplicity, openness, and high bandwidth efficiency, making it an ideal choice for constrained environments. Additionally, we delve into MQTT's topic-based architecture, its components (publishers, subscribers, and brokers), and how messages are routed based on topic interests. Furthermore, we touch upon the importance of MQTT security and the potential vulnerabilities it may face, along with an overview of common MQTT attacks in IoT environments. Understanding MQTT and its security considerations is essential for ensuring robust and secure communication in IoT systems.

3.1 MQTT Protocol

The MQTT (Message Queuing Telemetry Transport) protocol, known for its lightweight nature, is a highly favorable. The publish/subscribe communication pattern used by this open standard protocol, approved by OASIS [51], is particularly well suited for machine-to-machine (M2M) communication. Owing to its functionality over TCP, the MQTT protocol showcases exemplary reliability, ensuring an organized, lossless, and bidirectional mode of communication.

It operates on the theory of topics, which are essentially hierarchically organized categories under which users can publish messages. Subsequent to this, any additional client who has a subscription to the specific topic will receive these messages, establishing an organized and targeted communication framework.

The three nodes comprising the MQTT physical structure are publishers, subscribers, and brokers, as shown in Figure 3.1. Publishers are nodes that send messages, subscribers are nodes that receive messages, and brokers act as middlemen to coordinate message delivery from publishers to subscribers [52].

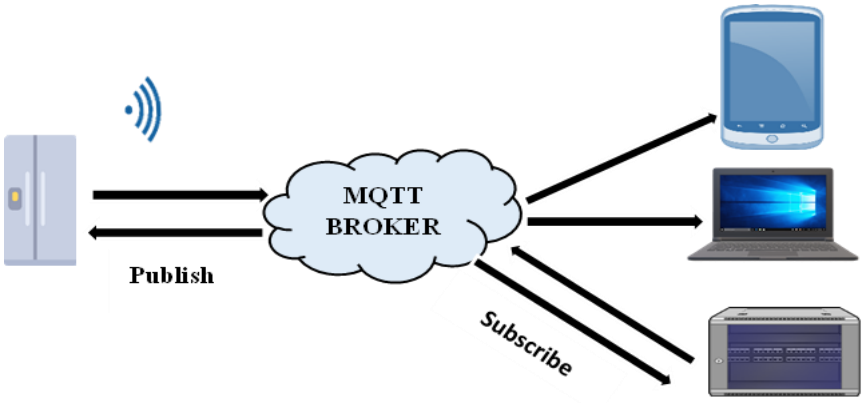

Figure 3. 1: MQTT Architecture

The architecture of the MQTT protocol is intentionally crafted to be straightforward and resource efficient. This design facilitates seamless communication between clients and the central broker, which is responsible for disseminating messages to all relevant subscribers. Incorporating topics and quality-of-service levels enhances the versatility and dependability of client communication.

QoS Level 0 (At most once): Often referred to as "fire and forget," this level offers the least assurance. Messages are dispatched at most once, possibly not at all, and no confirmation or overhead is involved. While it is the swiftest transmission mode, it lacks a delivery guarantee.

QoS Level 1 (At least once): The message is transmitted at least once through a single PUBLISH message exchange. Resending the PUBLISH message is an option for the sender if no acknowledgment (PUBACK) is received. However, this approach can lead to the duplication of messages.

QoS Level 2 (Exactly once): This level ensures the message's exact once-only delivery by representing the highest QoS tier. It entails a four-step handshake process between sender and receiver, making it highly suitable for applications demanding message delivery assurance. However, this level imposes the most substantial overhead. The handshake components encompass PUBLISH, PUBREC, PUBREL, and PUBCOMP messages.

The selection of the QoS level depends upon the particular specifications of the IoT application. If speed is more crucial than reliability, a lower level might be used. If assurance of delivery is the most crucial factor, then a higher level would be suitable. Figure 3.3. shows the three modes of QoS that can be defined in the PUBLISH messages exchanged between client and broker [52]:

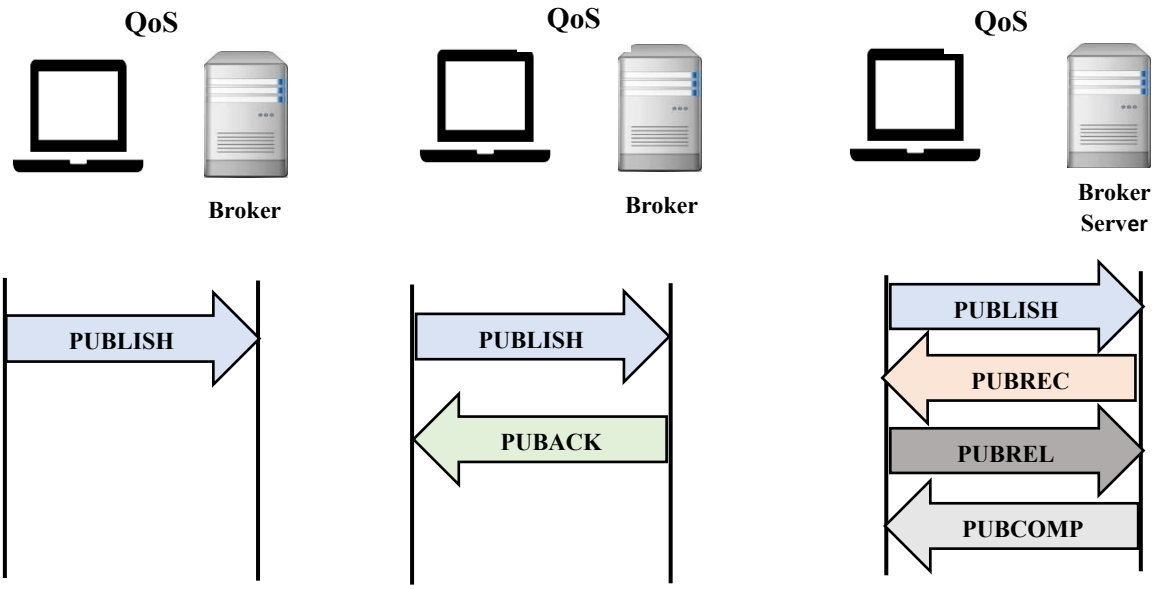

Figure 3. 2: MQTT Quality of Service (QoS) levels

3.3 MQTT Messages

MQTT communication messages are the main mechanism for transmitting data between clients and the broker. There are several MQTT messages, each with a specific purpose and format. The different types of MQTT messages provide the necessary functionality for efficient and reliable communication between clients and the broker [53].

There are several types of MQTT messages, including:

Connect: This is the first message sent.
Connack: This is the response from the broker to the client acknowledging the Connect request. It contains a return code indicating whether the connection was accepted or rejected.
Publish: This message serves the purpose of conveying application messages either from the broker to the client or the other way around within the MQTT communication framework.
Puback: This message is the acknowledgment from the broker to the client for a Publish message received at QoS level 1.
Pubrec: The sender is the recipient of this message to acknowledge a Publish message received at QoS level 2.
Pubrel: This message is sent from the sender to the receiver to ensure that the Publish message at QoS level 2 was received.
Pubcomp: The sender is the recipient of this message to confirm the Pubrel message.
Subscribe: This message is used by the client to register interest in one or more topics from the broker.
Suback: This is the acknowledgment from the broker to the client indicating that the subscription to a specific topic was successful.
Unsubscribe: This message is dispatched by the client with the intent of retracting its subscription from one or multiple topics within the MQTT system..
Ping request (Pingreq): This message is used by the client to verify that the network connection is alive.

Pingresp: This is the response from the broker to the Pingreq message.
Disconnect: This message is conveyed by the client to signal its desire for disconnection. Upon transmitting this message, the client is required to conclude the network connection.

.3.4 MQTT header

The MQTT header [53] is the first byte of each MQTT message allowing the broker to properly handle and distribute the message to subscribers. It is the fixed portion of an MQTT message, which contains information about the message type, topic, quality-of-service (QoS), and message flags. The MQTT header is followed by the variable-length payload, which contains the actual data being transmitted. The MQTT header includes the following fields:

Message Type: This is a 4-bit field that specifies the type of message being transmitted, such as a publish message, subscribe message, or acknowledge message.

Duplicate Delivery Flag: This is a 1-bit field that indicates whether the message being transmitted is a duplicate of a previously transmitted message. It is only used for QoS level 1 and 2 messages.

Quality-of-Service (QoS) Level: This attribute, occupying a 2-bit field, denotes the standard of assurance for the delivery of a particular message. It outlines the degree to which message delivery is guaranteed, offering three distinct levels.

Retain Flag: This is a 1-bit flag that indicates whether the broker should retain the message for later delivery to new subscribers.

Topic Length: This field specifies the length of the topic field in bytes.

Message Identifier: This field is used to identify messages and track their delivery. It is only present in messages with QoS levels greater than 0.

3.5 MQTT Security

MQTT is designed to be efficient and lightweight, and it is a popular IoT communication protocol used in smart homes and industrial IoT systems. However, as with any communication protocol, it can be vulnerable to various types of attacks [54]. Securing MQTT communications and various security measures are essential to prevent security threats and attacks. Some common MQTT attacks in IoT include:

Eavesdropping

An Eavesdropping attack in MQTT-IoT contexts signifies the illicit interception of data being transferred between a client device (such as a sensor or an IoT device) and a broker (the entity managing MQTT communications). The aggressor taps into the communication channel,

decoding the messages exchanged between the client and broker. This action might potentially lead to the acquisition of confidential information, including user credentials or sensor data.

Man-in-the-Middle (MitM) Attack

It targets circumstances where an evildoer intercepts and changes the transmission taking place between a client device (such as a sensor) and a broker.

Spoofing

A Spoofing attack in MQTT-IoT communications refers to a scenario where an attacker impersonates a legitimate client device (i.e., a sensor or an IoT device) or a broker (i.e., a server that manages MQTT communications) in order to manipulate the communication between the two. The attacker sends messages to the broker or the client that appear to be from a trusted source but are actually from the attacker.

Denial of Service (DoS) Attack

In MQTT-IoT communications, a DoS attack refers to an attack where an attacker disrupts the normal functioning of a broker (i.e., a server that manages MQTT communications) by overwhelming it with a large volume of requests, resulting in a complete or partial disruption of service. The assailant can induce widespread disruption in the functionality of IoT systems.

Injection Attack

An injection attack in MQTT-IoT communications refers to an attack where an attacker manipulates the data being transmitted between a client device (i.e., a sensor or an IoT device) and a broker (i.e., a server that manages MQTT communications) by injecting malicious payloads into the communication channel, causing devices to malfunction or behave unexpectedly. In an IoT environment, the attacker can alter sensor readings to cause damage or disrupt the operation of the IoT system, or they can inject malicious payloads into the communication channel to compromise the security of the system.

Sniffing

A sniffing attack in MQTT-IoT communications refers to an attack where an attacker intercepts MQTT traffic and listens to the communication between a client device (i.e., a sensor or an IoT device) and a broker (i.e., a server that manages MQTT communications). This type of attack aims to eavesdrop on the communication channel and obtain confidential information, such as passwords, usernames, or other sensitive data.

MQTT brute-force attack (MQTT BF)

This attack pertains to the methodical trial of several login and password combinations to illicitly gain access to an MQTT broker. Primarily utilized in IoT devices, MQTT is a streamlined messaging protocol developed to perform efficiently in networks characterized by low bandwidth, high latency, or unreliable connections [55].

Chapter 4

In this chapter, we delve into the diverse spectrum of machine learning algorithms and covers their wide range of applications where machine learning is making significant strides and revolutionizing industries. Additionally, we dive into ensemble techniques, their types, advantages, and disadvantages.

4.1 Machine Learning

Machine learning (ML), enhances computer systems to evolve and adapt over time by learning from data, without the need for explicit programming [56].

There are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. These systems are increasingly used in various applications, from recommendation systems to autonomous vehicles, cybersecurity, and beyond [57]:

- **Supervised Learning**

An algorithm learns from labeled training data using the machine learning technique of supervised learning in order to produce predictions or decisions. In supervised learning, the input characteristics (variables) are represented by target values or labels in the training data. It develops to match the input features to the appropriate output labels by generalizing patterns and relationships present in the training data [58]. Figure 4.1 depicts the supervised learning algorithm system.

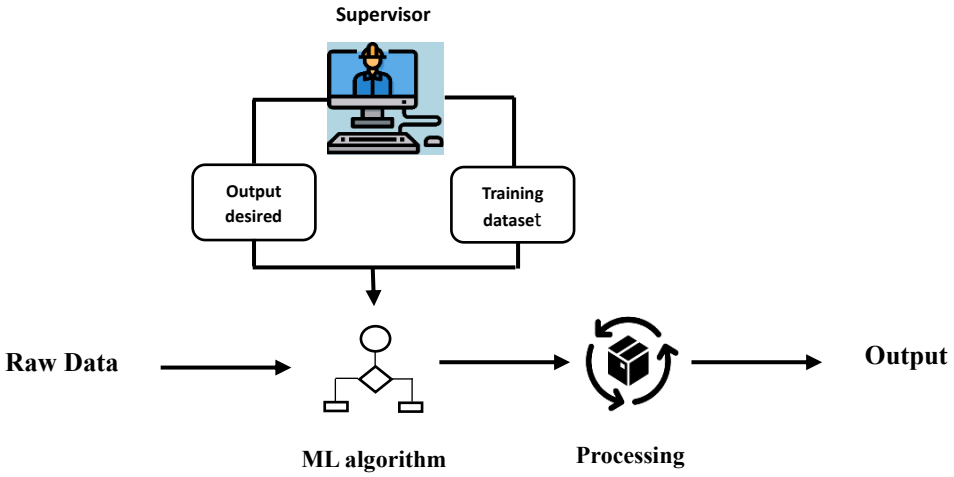

Figure 4. 1: Supervised learning algorithm

Two further categories of supervised learning exist:

- **Classification:** predicts categorical or discrete output labels. The algorithm learns to categorize new instances into established groupings or categories from labeled training data. It can classify email spam detection (classifying emails as spam or non-spam), image recognition (classifying images into different categories), and sentiment analysis (classifying text as positive, negative, or neutral sentiment).

- **Regression:** Regression attempts to predict output values that are either continuous or numerical. Training data with labels are used to train the algorithm to estimate or approximate a numerical value based on the input features. Regression tasks involve making predictions, such as the number of rooms, or forecasting stock prices based on historical data and market indicators.

One of the advantages of supervised learning is that it can achieve high accuracy in predictions, provided that the dataset is large and the features are relevant. However, a significant challenge in supervised learning is the need for labeled data, which can be expensive and time-consuming. Supervised learning models may overfit the training data, meaning they may perform poorly on new, unseen data.

- **Unsupervised Learning**

With this category [58] an algorithm can learn from unlabeled data to find patterns, structures, or correlations without using explicit input-output pairs. The algorithm identifies meaningful patterns or groups without prior knowledge or labels, as shown in Figure 4.2.

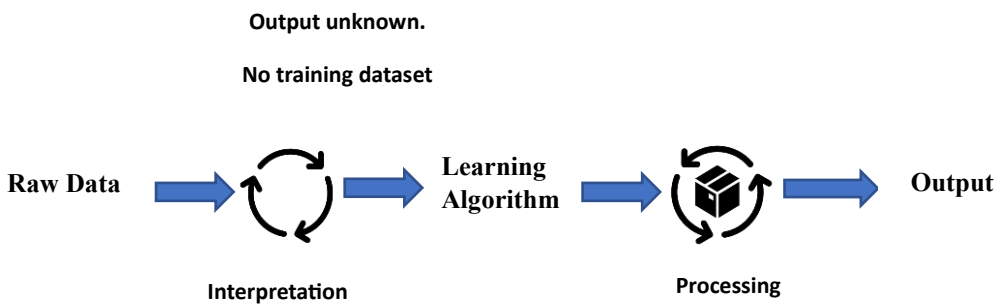

Figure 4. 2: Unsupervised Learning

Unsupervised learning can be further classified into two types:

- **Clustering:**
  Clustering involves segregating a dataset into distinct groups or clusters, wherein the data points within each cluster share a notable degree of similarity or proximity.

- **Dimensionality Reduction:**
  By minimizing a dataset's input features or variables, dimensionality reduction attempts to preserve the most important data. This involves transforming high-dimensional data into a representation with fewer dimensions.

Anomaly detection, data compression, and data visualization are just some areas where unsupervised learning is practical.

Unsupervised learning offers the advantage of uncovering hidden patterns or relationships in data that may go unnoticed when relying solely on labeled data. Furthermore, unsupervised learning proves helpful when labeled data is unavailable or the cost of labeling the data is excessively high.

Nevertheless, assessing the efficacy of an unsupervised learning algorithm poses challenges, given the absence of labeled output data for comparison against the projected outcomes.

- **Reinforcement Learning**

An agent learns how to interact with its environment using reinforcement learning [59] in order to maximize cumulative compensation. In this method, decisions are made based on the condition of the environment, which reacts by sending a reward signal. It aims to develop a policy that connects states to actions. The agent continually improves its policy through iterative interactions and learning from input, making better decisions in the environment.

As shown in Figure 4.3, Reinforcement learning can be broken down into:

• **Agent:** refers to the entity that interacts with the environment. The environment condition affects the agent's behavior, and it responds to the feedback by adjusting its actions accordingly.

• **Environment:** The agent engages with its surroundings, which is an external system. Depending on the agent's behaviors, the environment responds by sending a reward signal.

• **Reward Function:** The reward function transforms Each state-action pair into a reward indication that reflects the efficacy of the agent's activities.

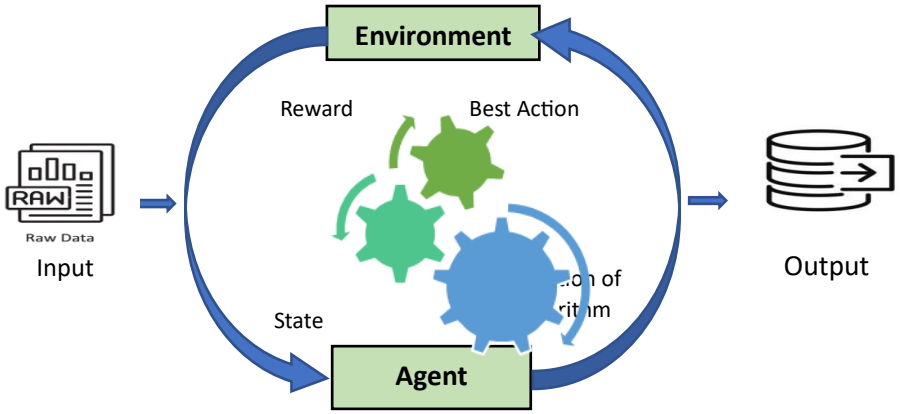

Figure 4. 3: Reinforcement learning algorithm

Reinforcement learning algorithms :

- **Value-Based Methods:**

  In these techniques, the agent determines the worth of the states or state-action pairings. The objective is to arrive at an ideal value function that maximizes long-term cumulative benefits.

- **Policy-Based Methods:**
  In this category, the agent directly optimizes the policy function, dictating actions in each state. Finding that optimizes the accumulation of rewards over a given timeframe.

Reinforcement learning has diverse applications in robotics, personalized medicine, natural language processing, resource management, recommendation systems, and financial trading. It can learn to make decisions in complex and dynamic environments, where the optimal policy may be unknown or change over time.

However, a major challenge in reinforcement learning is the need for extensive exploration of the environment, which can be time-consuming and computationally expensive.

Table 4.1 summarizes the differences between the categories, algorithms, and their explanations.

Table 4. 1: Classification of Machine Learning Algorithms [58]

Category	Algorithms	Explanation
Supervised	DT, LR, SVM, and KNN.	These algorithms learn from labeled data. The algorithm is given an input and produces the right result in response.
Unsupervised	K-Means, and PCA.	are trained using unlabeled data. Without output labels, the algorithm attempts to find hidden patterns in the data.
Reinforcement	Q-Learning, and Deep Q-Network.	They are used when the data is not labeled, and the only feedback is based on the model actions taken.

4.2 Machine Learning Applications

Machine learning has become a ubiquitous technology that is being used in various applications across different fields. Machine learning has found its way into almost every industry, from image and speech recognition to fraud detection and autonomous vehicles[60]. Machine learning applications continuously grow, and researchers are exploring new and innovative ways to incorporate this technology into their work [61].

There are numerous uses for machine learning [62] :

Speech and Image Recognition: can recognize items in images or recognize speech.

Natural Language Processing (NLP): Human language is interpreted and understood by machine learning algorithms enabling tasks like sentiment analysis and translation**.**

Predictive Modeling: Machine learning algorithms can make predictions based on historical data, such as predicting customer behavior or stock prices.

Anomaly Detection: Machine learning algorithms can identify unusual or anomalous behavior in data, such as detecting fraudulent activity or network intrusions.

Robotics: Machine learning trains robots to perform various tasks, such as object recognition, navigation, or manipulation.

Health care: Machine learning is used for medical applications, such as analyzing medical data and diagnosing diseases (Obermeyer and Emanuel, 2016) or developing personalized treatment plans.

Energy management: Machine learning is used for optimizing energy usage in various applications, such as in smart homes.

Environmental monitoring: Machine learning is used for analyzing and predicting environmental data, such as in climate modeling.

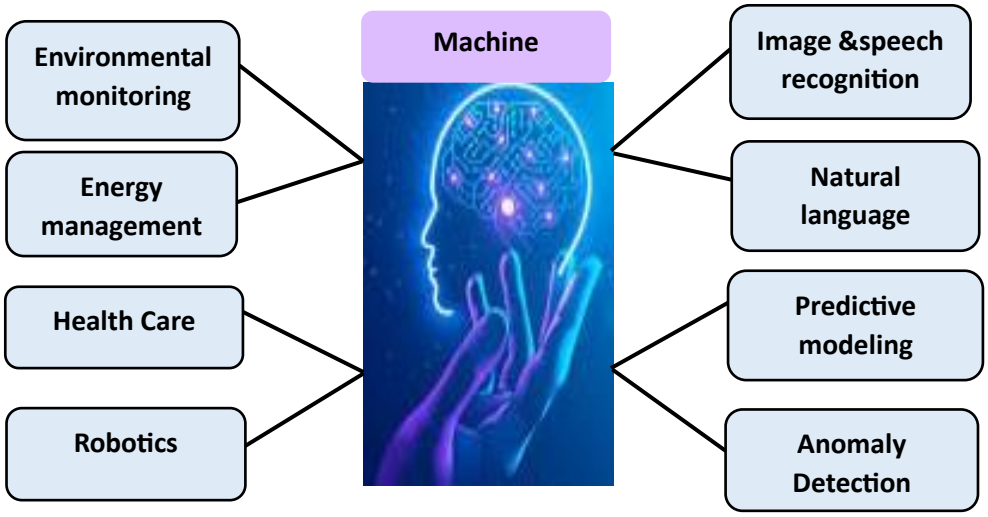

Figure 4. 4: Machine Learning Applications

4.3 Machine Learning Models

An overview of the machine learning algorithms used in this study is given in this section, each presenting advantages and limitations. Machine learning models [63] are algorithms that learn from data and then apply what they have learned to make informed decisions or predictions.

4.3.1 Logistic Regression

Logistic regression, a renowned statistical methodology, facilitates the elucidation of the relationship between a dichotomous dependent variable (y) and its associated predictor variables (x). By using the values of the predictor variables, it primarily aims to estimate the likelihood of the binary outcome variable. [64]. Figure 4.5 shows the Logistic Regression model.

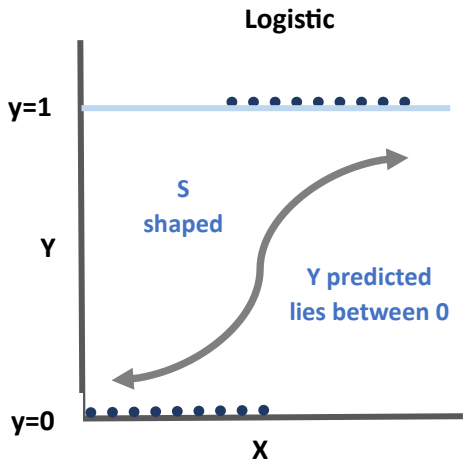

Figure 4. 5: Logistic Regression

Logistic regression is easy to implement and interpret. It can resist overfitting, especially in low-dimension datasets. It can provide output probabilities, making it useful in scenarios requiring probabilistic assessment. However, Logistic regression assumes linearity, which might not always hold true. It requires a large sample size for reliable predictions and may struggle with numerous categorical features.

4.3.2 Decision Trees

It uses a structure resembling a tree to create predictions and choices. It is a well-liked algorithm as a result of its simplicity [65].

A Decision Tree splits the data based on features, starting at a root node and creating branches for each outcome. This process repeats on each branch (or "child node") until specific criteria are met, forming a "leaf" or end node with a prediction value, as shown in Figure 4.6. The main types of decision trees are:

Classification trees: are employed when dealing with response variables that are of a categorical or qualitative nature**.** The leaf represents a class.
Regression trees: Used when the response variable is numeric or quantitative. The leaf represents a value.

Decision Trees are highly transparent, interpretable models that require minimal data preprocessing and can manage missing values effectively. As non-parametric models, they make no assumptions about data distribution, accommodating non-linear relationships. They are also useful for feature selection, with top nodes often representing the most significant features.

However, Decision Trees do come with some drawbacks. They can be prone to overfitting, especially when dealing with complex, noisy datasets, leading to overly complex

models that do not generalize well. This issue can be addressed to an extent by pruning, but it requires careful tuning.

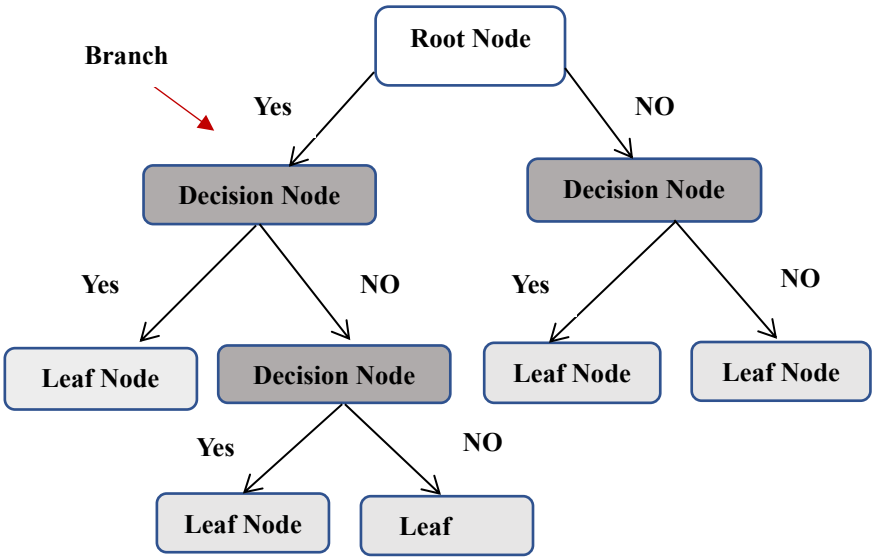

Figure 4. 6: Decision Tree

4.3.3 K-nearest Neighbors (KNN)

This algorithm functions by identifying the 'k' most proximate data points within the training dataset relative to a new, unseen data point. The label of this new point is then chosen depending on the labels of its closest neighbors. [66].

The KNN algorithm exhibits proficiency in handling binary and multiclass classification tasks. It predicts the class predominating among the 'k' nearest neighbors. In multiclass classification scenarios, it leverages distinct strategies to combine the neighbors' labels, such as majority voting or distance-weighted voting, in relation to the new data point. Figure 4.7 shows how KNN classifies new data points.

One of the defining strengths of the KNN algorithm is its simplicity coupled with versatility. It does not necessitate any preliminary assumptions regarding the data distribution or the functional form correlating the features and the target variable. Furthermore, it can manage nonlinear and nonparametric relationships between features and the target variable. However, KNN also has some limitations, such as the problem of dimensionality, which refers to the increased sparsity and the sensitivity to the choice of K and the distance metric.

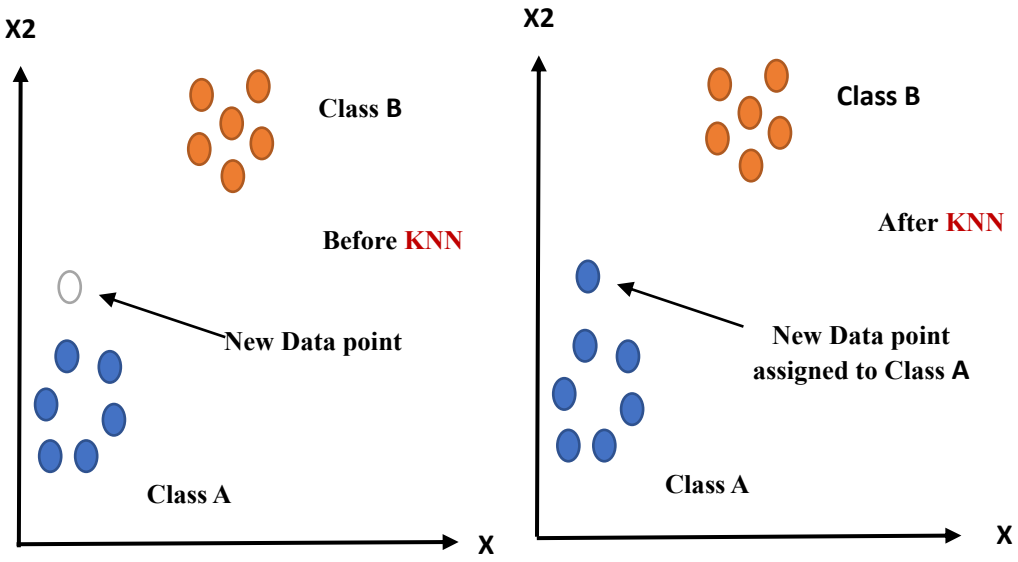

Figure 4. 7: KNN Classifier.

4.3.4 Adaboost

A popular machine learning technique called AdaBoost, or adaptive boosting, is made for classification and regression applications [67]. AdaBoost's fundamental concept revolves around fitting a series of weak learners, models that slightly surpass random guesswork such as small decision trees, to consistently adjusted data sets. These weak learners are subsequently combined to form a final prediction rule.

The data is altered by giving each training sample a particular weight during each boosting iteration. It is possible to train a weak learner in the first step using the original data because these weights are initially evenly distributed and set at 1/N. The learning process is then applied to the weighted data in subsequent iterations after the sample weights are individually changed. The predictions from all of the weak learners are then merged to create the final prediction, as shown in Figure 4.8, using a weighted majority vote (or sum for regression). The weights of each weak learner are determined during the training process, with higher weights given to the more accurate learners.

One of the main advantages of AdaBoost is that it is a fast algorithm and less prone to overfitting. However, noisy data and outliers in the data can negatively impact the algorithm's performance, so preprocessing is crucial.

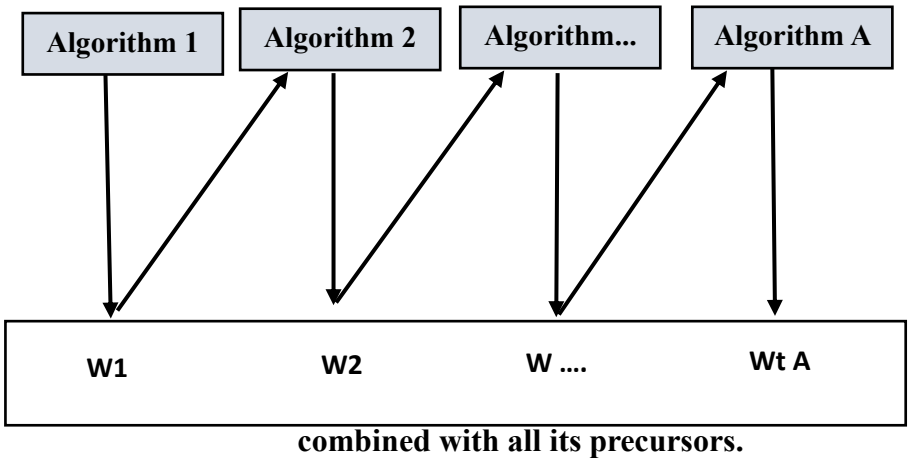

Figure 4. 8: AdaBoost Model

4.3.5 XGBoost

XGBoost, an acronym for Extreme Gradient Boosting, is a robust and favored machine learning algorithm [68]. The algorithm belongs to the gradient boosting methods category, constructing a robust predictive model via an ensemble of weaker prediction models, commonly in the form of decision trees, as depicted in Figure 4.9. The "Extreme" in XGBoost offers several notable advantages:

- **Speed and Performance**: XGBoost delivers superior efficiency, particularly when handling large datasets.

- **Core Algorithm is Parallelizable**: XGBoost employs parallel processing, making it significantly faster compared to other algorithms. Additionally, it can handle sparse data and missing values.

- **Integrated Cross-Validation:** Cross-validation is a feature of XGBoost that enables users to determine the ideal number of boosting iterations in a single run by performing it at each stage of the boosting process.

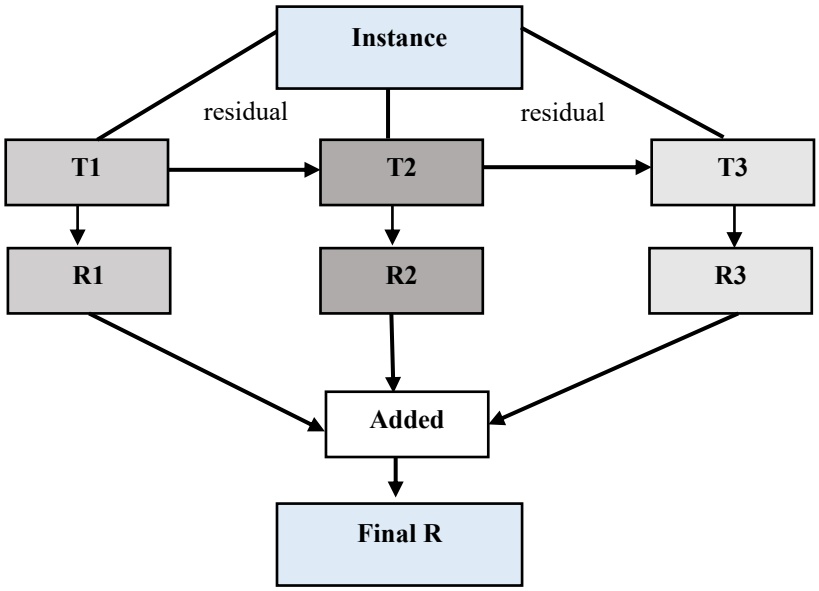

Figure 4. 9: XGBoost Model

4.4 Machine Learning Ensemble Techniques

Machine learning ensemble techniques combine multiple individual models to create a more powerful, accurate, and robust model [69]. The ensemble model can perform better overall by integrating the benefits of each model and minimizing its drawbacks. There are several ensemble techniques and algorithms. Some of the most popular ones include [70]:

4.4.1 Bagging (Bootstrap Aggregating)

Bagging, also known as bootstrap aggregating, seeks to minimize variance and prevent overfitting. This method creates a large number of bootstrap samples (random samples with replacement) and trains a different base model on each of them. For classification, the final ensemble prediction is obtained using a majority vote, as shown in Figure 4.10. Some popular Bagging Algorithms:

Random Forest: This algorithm constructs multiple decision trees and merges their outcomes. For the construction of each tree, a subset of the dataset is utilized, along with a random selection of feature values.

Extra Trees: Similar to Random Forest, but it builds more randomized decision trees by selecting random split points for each feature.

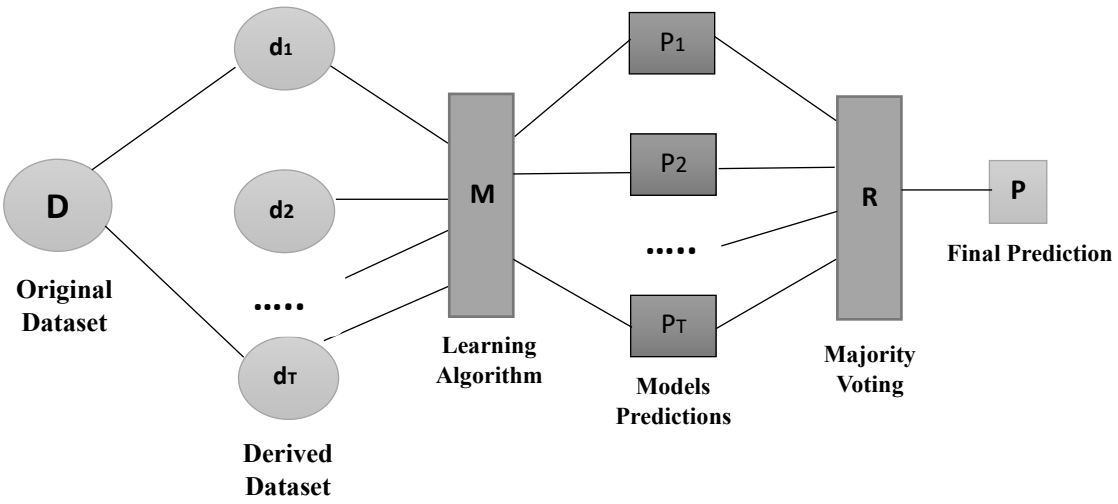

Figure 4. 10: Bagging (Bootstrap Aggregation) scheme

Advantages of Bagging include [71]:

Reduces overfitting: Bagging minimizes overfitting by generating numerous base models. The averaging of their predictions can smooth out the decision boundaries, yielding a more generalized model.

Improves stability: Bagging models are resilient to noise and outliers, delivering more consistent and reliable predictions. The ensemble model becomes less sensitive to outliers because of the averaging process and the diversity introduced by bootstrapping. This diversity allows different aspects of the data to be captured, enhancing the model's stability.

Parallelizable: Each base model in bagging can be trained independently, which suits parallel or distributed computing well. This parallel training process expedites the overall training time, particularly for large datasets or complex base models.

However, bagging has some drawbacks:

Computationally expensive: Training numerous base models can be time-consuming, particularly for complex models or large datasets. The model aggregation process, although less resource-intensive than model training, also adds to the overall computational cost.

Memory requirements: As bagging necessitates multiple base models, it may require more memory to store the individual models and their predictions. This can be problematic with limited resources or during model deployment in production.

4.4.2 Stacking (Stacked Generalization)

Stacking, alternatively known as Stacked Generalization, constitutes an ensemble strategy that incorporates the training of numerous foundational models on the identical dataset [72]. Subsequently, these models' predictions are utilized as input attributes for a more sophisticated meta-learner. The meta-learner is designed to ascertain the best way to combine the predictions made by the foundational models, thereby generating the final output, as demonstrated in Figure 4.11. KNN, DT, and neural networks, are frequently deployed as foundational learners in stacking scenarios.

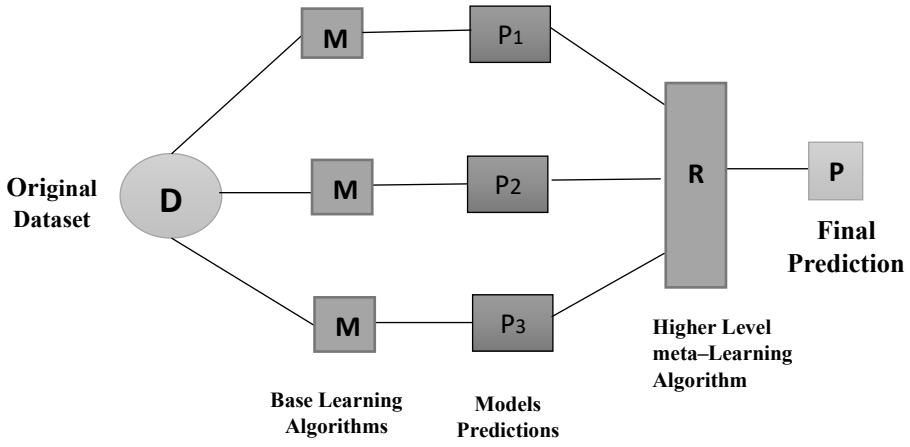

Figure 4. 11: The Stacking Scheme

Stacking's advantages include leveraging model diversity by integrating predictions from various base models, hence improving the overall performance. Furthermore, stacking allows for customizability, permitting the utilization of different base models and meta-models, which can be beneficial in addressing specific problems. Stacking's flexibility manifests in several ways:

- **Diverse model complexities:** Stacking allows for the usage of base models with varying complexity levels, helping balance bias and variance and leading to enhanced performance.
- **Customizable meta-model:** The user is free to choose the meta-model that merges the predictions of base models, optimizing ensemble performance.
- **Meta-model feature engineering:** Stacking allows for the incorporation of additional features to improve the meta-model's and overall ensemble's predictive prowess.
- **Customization of cross-validation strategy:** Stacking utilizes cross-validation to generate out-of-sample base model predictions, and this strategy can be customized to suit specific problems and dataset characteristics.

Conversely, stacking also comes with drawbacks. It is more complex and computationally demanding due to its multi-layered structure. The training process is also longer since it involves training both base models and the meta-model. This can be particularly taxing if the base models are inherently complex. The typically used cross-validation strategy can also add to the training time.

4.4.3 Boosting

Boosting represents another ensemble method striving to minimize both bias and variance by integrating the outcomes of several weak learners in successive order, as depicted in Figure 4.12. Every learner following the initial one endeavors to rectify the inaccuracies of its predecessor leading to an improved overall model [73]. Popular Boosting Algorithms:

- **AdaBoost (Adaptive Boosting):** The first boosting algorithm, which combines multiple weak classifiers by assigning different weights to each based on their accuracy.

- **The Gradient Boosting Machine (GBM):** leverages the gradient descent method to gradually reduce the loss function. It achieves this by sequentially training each weak learner on the residuals, or prediction errors, resulting from the prior learner.

- **XGBoost (eXtreme Gradient Boosting):** An optimized implementation of GBM, which provides better performance and is more scalable.

- **LightGBM:** A variation of gradient boosting that uses a histogram-based algorithm for faster training and improved performance on large datasets.

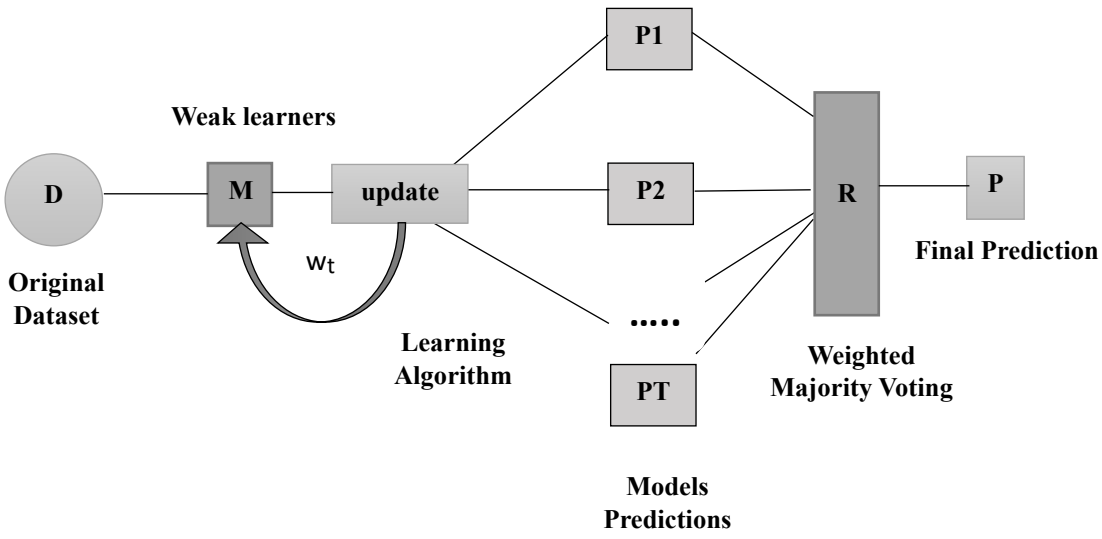

Figure 4. 12: The Boosting Scheme

Boosting Advantages:

- **Bias and Variance Reduction:** Boosting combines weak base models to lower both bias and variance, often leading to improved accuracy.
- **Adaptability:** Boosting adjusts to different subsets of data depending on difficulty, iteratively learning from its mistakes to improve performance.
- **High Predictive Performance:** Boosting, through algorithms like AdaBoost, Gradient Boosting Machines (GBMs), and XGBoost, often yields superior performance across diverse problems.
- **Noise and Outlier Resilience:** Boosting can handle noisy data and outliers fairly well, given its use of weak learners and control mechanisms like the learning rate parameter.
- **Versatility with Mixed Data:** Boosting can handle various data types, such as continuous, categorical, and ordinal features, making it a flexible technique.

Boosting Disadvantages:

- **Computational Cost:** Boosting can be computationally expensive compared to other models or ensemble techniques.

- **Sensitivity to Noise:** In certain circumstances, boosting can be sensitive to noise, focusing on fitting the noise rather than the true underlying pattern. Table 4.2 gives a summarized comparison between Bagging, Stacking, and Boosting ensemble techniques.

Table 4. 2: Comparison between Ensemble Techniques

Common Types of Ensemble Techniques	
Bagging	➢ Reduce variance and increase accuracy. ➢ Often used with Decision Trees. ➢ Overcome outliers or noisy Data.
Stacking	➢ Used to ensemble group of strong learners. ➢ Involves training a "meta learner" algorithm to learn the optimal combination of the base learners.
Boosting	➢ Flexible (can be used with any loss function). ➢ Reduce variance and increase accuracy. ➢ Not robust against outliers or noisy Data.

Chapter 5

In this chapter, we carry out a comprehensive examination of the existing literature related to our research topic. We delve into various studies, research papers, and academic sources that discuss the attacks detection in MQTT-IoT networks. In addition, we explore the strengths and limitations of different approaches, including ensemble techniques, for addressing cybersecurity challenges in this context.

5.1 Literature Review

In today's world, IoT has seen substantial growth, with a vast array of devices now connected to the web [74]. Nevertheless, this enhanced integration has also escalated the risk of cyber threats [75]. In response, various studies proposed intrusion detection systems (IDS), utilizing different algorithms, such as CNN, LSTM, DT, SVM, and ensemble learning [76],[77],[78],[79]. Even though applying machine learning to protect the Internet of Things (IoT) has been the subject of numerous studies in recent years, more efforts have yet to be made to ensure the safety of the MQTT protocol when used on the Internet of Things [80]. Table 5.1 summarizes the information covered in this section.

When writing their research, the authors in [81] used machine learning to spot security holes in an MQTT network. First, the authors created a new IoT-MQTT dataset named MQTT-IOTIDS2020 and then identified MQTT-based attacks. The system includes three main components: feature extraction, selection, and classification. The weighted average recall and precision were 93% and 97%, respectively. Yet, the authors failed to offer adequate details about data preprocessing and feature extraction.

The IDS suggested in [76] combined a multi-objective optimization approach for reducing data dimensions with a deep learning strategy that employs various models to detect DDoS attacks such as LSTM and CNN. The suggested approach was tested on the recent CISIDS2017 dataset for DDoS attacks and attained a 99.03% accuracy rate. While the proposed IDS appears promising, there are some limitations to consider:

- Lack of Comparative Analysis: The comparative analysis is limited to a few specific algorithms. A more detailed comparison would aid in better understanding how the proposed method performs relative to other approaches.
- Limited Scope: While the proposed method addressed DDoS attacks in IoT networks, it does not consider other cyber-attacks that may threaten these networks. A more

comprehensive approach that considers multiple cyber-attack types would be more beneficial.

Authors in [77] suggested that the Deep Residual CNN model protects IoT systems against Botnet attacks. As a result, LSTM, CNN, and RNN models were implemented on Real traffic data and the "N-BaIoT Dataset" to test the method. The findings indicated that the Deep Residual DCNN achieved the top training accuracy of 88.67%, trailed by LSTM and then CNN with RNN.

However, the study has only evaluated the proposed system's performance against the N-BaIoT dataset and real traffic data from the Mirai and BASHLITE botnets, which may limit the generalizability of the results to other IoT datasets and botnets. In addition, the research did not address the proposed system's scalability for expansive IoT networks. Moreover, the study needs to give information on the computational resources required to implement the proposed system on low-end IoT devices, which may have limited processing capabilities.

In [78], the authors introduced ARTEMIS IDS, which discerns the system's standard operation and sends warnings during anomalies. The system used a dataset from a simulated IoT network and evaluated its performance against various attack scenarios. The study found that ARTEMIS achieved a low false positive rate. With a good accuracy of 99.98% when utilizing One-Class SVM, the ARTEMIS IDS employed various ML techniques to identify fraudulent MQTT messages. However, the study used a dataset collected from a simulated IoT network. Therefore, it is unclear how well the system will perform in real-world IoT networks with different network topologies, traffic patterns, and IoT devices.

Authors [79] proposed an ensemble classification model using an automatic model selection method and implemented three classifiers, DT, RF, and Gradient boosting, in their study. The authors calculated the efficiency scores for NSL-KDD, UNSW-NB15, BoTNeTIoT, and BoTIoT datasets to choose the top three models. The model obtained a high F score of 99% even though their study looked at zero-day attacks as a binary classification problem. Moreover, the study did not compare their system with others which may limit the ability to evaluate the system's performance against different approaches.

Authors in [82] suggested feature clustering instead of classification in the UNSW-NB15 Data set. The dataset was pre-processed into an executable form for the algorithm, and then the accuracy results were evaluated. Using the entire features, the authors used supervised Machine Learning (ML) approaches to train Random Forest (RF), Support Vector Machine, and Artificial Neural Networks (ANNs) on the clusters. For binary and multi-class classification, RF attained accuracy rates of 98.67% and 97.37%, respectively. However, they still need to provide a comprehensive discussion on how this approach affects the performance.

Deep Learning (DL) based Network IDS was proposed in [83] to detect MQTT intrusions. The authors have implemented CNN, RNN, and LSTM using the MQTT-IoT-IDS2020 Dataset. The Aggressive scan, UDP-scan, and MQTT brute-force attacks were evaluated using weighted average evaluation metrics. On average, the DL-based Network IDS detected MQTT attacks with 97.09% accuracy and an F1 score of 98.33%. While the proposed

system achieved high accuracy and F1 score, other performance metrics were not discussed in detail.

The NSL-KDD Dataset was utilized in the solution of Dhanabal and Shantharajah's [84]. The study examined the Dataset using the C4.5 algorithm (J48), SVM, and Naive Bayes algorithms and found that the J48 analysis yielded the most accurate findings. In addition, the research used data mining techniques and the WEKA tool to analyze the dataset and find out which protocols were more vulnerable. The study, however, relies solely in a synthetic data which may not represent real-world intrusion attempts accurately.

Authors in [85] assessed five models: NB, ANN, XGB, DT, and KNN algorithms on the MiTM Dataset (an open-source Dataset from Dataset (joseaveleira.es). All the models gave accuracy above 95 percent. The extreme Gradient boosting algorithm showed a high accuracy rate, although it took much more training time than other algorithms.

Even though their paper is a conference proceeding, it is relatively short and provides limited details about the methodology and experimental setup. Furthermore, the study was conducted on a small-scale testbed and the study only focused on detecting one type of attack on a specific IoT communication protocol (MQTT.

In the study [86], DT and SVM were utilized in constructing a hybrid model to identify intrusions this study. The KDD99 intrusion detection dataset was mined for nine features that were deemed particularly significant and pertinent. Their study demonstrated an accuracy of 99%, while also having a low false alarm rate (FAR), coming in at 0.9%. The authors claimed their hybrid system achieved better accuracy and detection rates than individual DT and SVM models. However, the paper needs to provide its limitations or drawbacks.

In the study [23] the authors presented an IDS that used supervised learning to monitor potential signals of intrusion. To increase accuracy and decrease false positives, the suggested strategy employed an ensemble of support vector machines (SVM) and Nave Bayes classifiers. They used a true historical log dataset that had been standardized and pre-processed. The suggested system achieved accuracy and precision of 95%, and the classifier's efficiency improved after adding session-based characteristics. In addition, the suggested technique was compared with other techniques and showed significant reductions in false positives.

However, using this dataset could limit the applicability of the results to other datasets. Secondly, the proposed method demands a significant amount of computational time. This is especially true when employing the SVM and hybrid classifier, which might render it unsuitable for systems requiring real-time detection. Lastly, the method is dependent on event logs. Such logs might not encompass all indications of intrusions, leaving out potential threats like those originating at the network layer.

Another study [87] examined the effectiveness of numerous traditional machine learning methods by applying them to several ID-based datasets and analyzing the results. The authors conducted a study comparing machine learning techniques for intrusion detection across multiple datasets, namely UNSW-NB15, CIDDS-001, and NSL-KDD. After standardizing these datasets, they employed SVM, KNN, and DT. DT was superior to other

classifiers since it had a detection prediction accuracy that ranged between 99 and 100% for all datasets.

However, some potential limitations may include the fact that the experiments were conducted on specific datasets and attack scenarios, which may only be representative of some possible cyberattacks or network configurations. Additionally, the study's evaluation metrics and methods may only be appropriate for some use cases or scenarios.

The study in [88] introduced a lightweight system which used an SVM-based classifier to detect unwanted data injections by malicious actors. Recognizing the unique constraints of IoT networks—like limited computational power, restricted memory, and tight energy capacities—the IDS was meticulously designed. To assess the SVM's performance, the authors employed a Poisson process for generating training and test samples. Their results highlighted the SVM algorithm's efficacy when complemented by two or three straightforward features, achieving impressive classification accuracy and swift detection times. Notably, when compared to other machine-learning-based IDSs, the proposed SVM-centric approach excelled. This underscores its potential as a rapid response tool against DoS attacks on IoT networks.

Even though the study achieved good results regarding classification accuracy and detection time, it has some limitations. First, it could not find intrusions without slowing down traffic flow. Second, the proposed system was tested only in a simulated environment using a limited number of attack scenarios. Third, the authors used a Poisson process for traffic modeling, which may not accurately reflect the traffic patterns of real IoT networks. Finally, the study only considers DoS attacks.

A machine learning (ML)-driven botnet attack detection system was introduced in [89]. The framework reduced the requirement for handling resources by implementing an applicable feature extraction technique. The proposed model incorporated three, achieving a detection accuracy of 99%.

Although the suggested method was evaluated against various models, including NB and J48, more comparisons with other machine learning and rule-based methods could better understand the proposed approach's strengths and limitations. Moreover, the efficacy and execution of the proposed framework hinge on the quality of the chosen features. As a result, the selected features might be best suited for detecting only certain kinds of attacks.

In [90] the stacked ensemble learning technique utilized DT, LR, and gradient boosting as base models. The research sought to enhance the effectiveness of IDSs by merging the advantages of multiple models, aiming for superior IDS. When 23 key characteristics were extracted from the CICIDS-2018 , the proposed system achieved a score of 97.9% on the F-measure and had an accuracy of detection of 98.8%.

Limitations of the study encompass the notion that the efficacy of the models is contingent upon the dataset they are trained with. The CSE-CIC-IDS2018 dataset, in particular, may only represent a subset of potential attack scenarios. In addition, the proposed model used only one voting method for ensemble learning, and other methods were not compared.

The concept of anomaly detection systems for cloud computing was presented in [91]. The support vector machine (SVM) was chosen as the main ML method due to its diverse kernels. The results of experiments showed an accuracy of 96.24 percent and reduces the false alarms number.

However, the results are only based on one model and one dataset, which may not be generalizable to other algorithms or techniques. Using more diverse datasets would make the results more generalizable. The study also did not compare the proposed system with existing intrusion detection systems, making it difficult to determine how well it performs relative to other systems.

In [92], the efficacy of a contemporary IDS with a hybrid approach for multi-agent systems was assessed. When utilizing the NSL-KDD dataset to implement a Deep Neural Network (DNN), the proposed system exhibited a 98% success rate in anomaly detection and a 97% accuracy in distinguishing between various attack types.

On the other hand, there are several limitations of this research. Firstly, the proposed system was not tested in a real-world IoT environment, so its effectiveness in detecting real-world IoT attacks is yet to be demonstrated. Secondly, the system relied heavily on deep learning algorithms, which can be resource-intensive and unsuitable for low-power IoT devices. Thirdly, the proposed approach used a complex multi-agent system architecture, which may require more work to implement and maintain. Fourthly, although the suggested system employed the NSL-KDD dataset for testing, this dataset might only capture a fraction of the various attacks possible in actual IoT scenarios. In conclusion, the study recognized that the proposed system may fall short in accurately detecting infrequent attacks. This indicates a need for continued research to enhance the system's precision and efficacy.

In another study [93], a method was introduced for identifying DDOS attacks using DT and KNN classifiers. The efficacy of the technique was evaluated on two datasets, NSLKDD and KDDCup99. With a detection accuracy of 99.51 percent, KNN exceeded DT with an error rate of 1.5 percent.

Yet, the authors offered limited details about the pre-processing methods employed, which clouds understanding of their influence on the model's outcomes. Furthermore, the research narrowed its lens to just two machine learning algorithms, DT and K-Nearest Neighbor, without delving into other potential machine learning algorithms or probing into deep learning structures.

In [94], the authors investigated the application of the classifiers RF, AdaBoost, GBM, Highly Randomized Trees, Classification and Regression Trees, and MLP for DoS attacks. The research concentrated on both ensemble and individual classifiers. The authors utilized different datasets for testing the classifiers, namely CIDDS-001, UNSW-NB15, and NSLKDD. The objective of the research was to encourage scholars to craft IDSs using ensemble learning and to offer suitable techniques for the statistical evaluation of classifier efficacy. In addition, the study demonstrated the versatility of a classifier by showing that the XGB classifier performs well for both classification and regression trees.

However, the study only evaluated the performance of classifiers on Raspberry Pi, which may differ from other IoT hardware platforms. In addition, the study used few evaluation metrics to assess the performance of the classifiers, which will not capture the performance of the classifiers fully. Other metrics may provide a more comprehensive evaluation. Moreover , the study only examined a limited number of ensemble methods, and other ensemble methods may perform better on the datasets used.

In [95], the authors introduced an IDS tailored for IoT-MQTT networks, utilizing Elite Machine Learning (EML) algorithms. They assessed the proposed approach using a lightweight MQTT protocol on a testbed and the SEN-MQTTSET dataset. Out of the seven evaluated ML algorithms, they aimed to identify the optimal model for intrusion detection based on performance indicators, achieving an accuracy exceeding 99%.

However, the dataset used in the study may not represent all possible attack scenarios in a real-world IoT-MQTT network. Additionally, the study might not have fully captured the real-world challenges of an IoT-MQTT environment. Also, they only compared their system to existing ones and did not look at the latest methods or systems.

In [96], researchers introduced an IDS for SDN on flow data using the NSL-KDD dataset. This method utilized a five-level system that combines KNN, ELM, and H-ELM to spot DoS, R2L, U2R, and other unknown attacks and got a top accuracy of 84.29%. Plus, their system was fast in terms of computing and worked even when network traffic was encrypted.

However, it may not be effective in detecting attacks embedded in the packet's payload and may require further improvements to detect unknown attacks with a lower false alarm rate. Additionally, the proposed IDS was designed based on six flow features, which may only capture some of the necessary information required for effective intrusion detection.

The goal of the study discussed in [97] aimed to identify DDoS attacks. They wanted to find out which algorithm worked best. So, they compared Random Forest (RF), decision tree (C5.0), naive Bayes (NB), and support vector machines (SVM) using a normalized CICIDS2017 dataset. The results revealed that RF and C5.0 achieved average accuracies of 86.80% and 96.45%, respectively, with a success probability of 99%, while SVM exhibited 75 percent of FPR.

The study provided a comprehensive evaluation of pre-processing techniques, feature selection methods, and different classifiers. However, the researchers did not consider other types of attacks, such as network intrusion attacks or malware attacks.

In [98], on the KDDcup99 dataset, using a vote-based ensemble learning technique, the researchers assessed different algorithms. The Bayesian network used was more effective with small datasets, while the random tree performed better with more extensive sample data.

However, the study should have discussed the proposed model's computational complexity and training time, which could be a concern when dealing with large datasets.

Table 5. 1: Summary of the Literature Review

Ref	Models Trained	Dataset used	Attacks	MQTT	Simulation /Real Testbed	Ensemble Technique	Feature Selection	Evaluation Metrics
[81]	RF, SVM, NB, LG, KNN, and DT	MQTT-IOTIDS2020	Scan_A UDP Scan Brute force	Yes	MQTT Traffic	No	Yes	Recall, Precision F1-score
[76]	CNN LSTM	CISIDS2017	DDoS	No	Simulation	No	Yes	Accuracy
[77]	CNN, LSTM, and RNN	N-BaIoT	Botnet	No	Real Traffic	No	No	Accuracy
[78]	SVM	IoT network	Binary classification	Yes	Simulated IoT	No	Yes	Accuracy
[79]	DT, RF, Gradient boosting	NSL-KDD, UNSW-NB15, BoTNeT IoT, and BoTIoT	Zero-day attacks	No	Network Traffic	Yes	Yes	F1score-Roc curves
[82]	RF, SVM, and ANN	UNSW-NB15	DoS	Yes	Simulation	No	Yes	Accuracy
[83]	DNN, CNN, and LSTM	MQTT-IoT-IDS2020	Scan_A, UDP scan, Brute force	Yes	Simulation	No	Yes	Accuracy F1 score
[84]	J48, SVM, NB	NSL-KDD	DoS, R2L U2R, Probe	No	Simulation	No	Yes	Accuracy
[85]	NB, ANN, XGB, DT and KNN	MiTM dataset	MiTM	Yes	Real Testbed	No	Yes	Accuracy, precision recall,F1 score

[86]	DT, SVM	KDD99	Dos, probe, u2r r21	No	Simulation	Yes	Yes	Accuracy FAR
[23]	SVM, NB	Historica-l log	Unknown attacks	No	Real Testbed	Yes	Yes	Accuracy Precision
[87]	SVM, KNN, and DT	UNSW-NB15, CIDDS-001, and NSL-KDD	DDoS, Brute force Exploit SQL injection	No	Simulation	No	Yes	Accuracy Recall Precision F1score
[88]	SVM	Generated sample traffic	DoS	No	Simulation	No	Yes	Accuracy
[89]	DT, NB ANN	N-BaIoT	Botnet	No	Simulation	No	Yes	Accuracy
[90]	DT, LR GB	CIC-IDS 2018	Brute force, SQl, DoS	No	Simulation	Yes	Yes	Accuracy Precision Recall F1 score
[91]	SVM	NSL-KDD	Not mentioned	No	Simulation	No	Yes	Accuracy Roc curve
[92]	DNN	NSL-KDD	DoS, Probe, R2L and U2R	No	Simulation	No	Yes	Accuracy Precision Recall The 1 score
[93]	DT, KNN	NSLKDD, KDDCup 99	DDoS	No	Simulation	No	Yes	Accuracy
[94]	RF, AdaBoost, GBoost, XGboost	CIDDS-001, UNSW-NB15, NSLKDD	DoS	No	Simulation	Yes	Yes	Accuracy Precision Recall The 1 score

[95]	LR, KNN, RF, NB, SVM, GB, DT	SEN-MQTTSET	DoS	Yes	Real Testbed	Yes	Yes	Accuracy, F1 score, Roc curve
[96]	KNN, ELM, H-ELM	NSL-KDD	DoS, R2L, U2R, unknown attacks	No	Flow-based	No	Yes	Accuracy Precision Recall, F1 score, FAR
[97]	RF, DT, NB, SVM	CICIDS2017	DDoS	No	Flow-based	No	Yes	Accuracy
[98]	Bayesian network	KDDcup 99	Probe, DoS, U2R, R2L	No	Simulation	Yes	Yes	Accuracy, P,Recall,F1 score

Many studies which have been conducted utilized different techniques. However, there is a lack of need to be more research on detecting cyber-attacks specific to the MQTT protocol. While fewer studies have proposed machine learning algorithms for detecting MQTT intrusions, they have not been extensively evaluated in real-world scenarios, and their scalability and generalizability are unknown. Looking ahead, researchers should aim to create more robust IDSs capable of identifying a range of cyber-attacks.

Conversely, the research discussed in this section has demonstrated encouraging outcomes in identifying various cyber-attacks on IoT networks. However, most studies have only focused on detecting one type of attack or evaluating the performance of a specific set of algorithms on a limited dataset. Few studies have looked at the performance of IDS on different network topologies, traffic patterns, and IoT devices.

Chapter 6

In this chapter, we delve into the MQTT-IoT-IDS2020 dataset, offering an overview of its contents and features. We also shed light on the data preparation steps undertaken to make it ready for utilization in the machine learning models. In addition, we explain the methodology followed to accomplish our thesis goal. We then provide an overview of the experimental setup employed in our research. We outline the specific hardware and software setups employed. Lastly, we explain the evaluation metrics chosen to gauge the performance outcomes achieved through the application of ensemble machine learning methods.

6.1 MQTT-IoT-IDS2020 Dataset Overview

The MQTT-IoT-IDS2020 dataset[81] is a cybersecurity dataset that was created in 2020 for the purpose of doing IDS research.The MQTT protocol is the primary topic of this dataset, which was created with the intention of assisting academics and developers in improving the safety of Internet of Things systems. The dataset was created by capturing and analyzing actual IoT network traffic, ensuring its authenticity and relevance to real-world scenarios. The dataset includes both typical traffic and a variety of cyber-attack traffic, such as:

An aggressive scan (Scan A): This type of attack is a variant of network scanning where an attacker uses a tool or script to scan a network in an aggressive manner, probing many ports across multiple systems in a short time span. These scans can potentially disrupt network services or overwhelm systems due to the volume of traffic generated. The purpose is often to identify open, vulnerable ports that can be exploited later [99].

A scan of the User Datagram Protocol (UDP) (Scan sU): This refers to a type of UDP scan. In terms of network protection, this method is employed to pinpoint open UDP (User Datagram Protocol) ports on a designated system [99].

A Sparta SSH brute-force attack (Sparta): This refers to a brute-force attack on the SSH (Secure Shell) protocol using a tool like Sparta. SSH [100] is a protocol used to establish a safe connection over an insecure network. It is typically used for logging in remotely, executing commands, and accessing command-line interfaces on distant machines. It systematically attempts all combinations of passwords (or using a list of common passwords) until the correct one is found. Tools like Sparta can automate this process [101], [102].

An MQTT brute-force attack (MQTT BF): This refers to a brute-force attack on the MQTT protocol. An MQTT_BF attack would typically involve an attacker trying to guess a weak password, for MQTT messages, thereby gaining control over the communication and data flow [103].

The dataset, which includes these attack scenarios, provides a platform for creating and testing machine learning models and intrusion detection systems. This ensures they can recognize and categorize cyber assaults aimed at MQTT-based IoT networks effectively.

The main goal of the dataset is to make it possible for developers and researchers to design, train, and evaluate machine learning models that are able to accurately recognize and categorize various kinds of cyberattacks that are launched against IoT networks. Researchers and developers can leverage the MQTT-IoT-IDS2020 dataset to study the characteristics of MQTT-based network traffic, identify attack patterns, and design innovative intrusion detection algorithms. The dataset includes both processed features and raw capture files in the popular pcap format, providing flexibility for different analysis approaches.

While IOT-23 Dataset [104] exists, a recent and widely used dataset for IoT security research, it does not include MQTT. The MQTT-IoT-IDS2020 dataset specifically targets the MQTT protocol, which is widely deployed in IoT platforms. By focusing on MQTT-based traffic, we can gain insights into the unique security challenges and attack patterns specific to this protocol. Our study adds value by focusing on a specific protocol and addressing the unique security challenges associated with MQTT-based IoT networks.

6.2 Description of the Dataset

In this section, we give a thorough overview of the dataset that was produced by simulating 12 MQTT sensors.

The dataset, which can be found in [81], encompasses five recorded scenarios: normal operation and four distinct attack scenarios. The attacks performed include an aggressive scan (Scan A), a User Datagram Protocol (UDP) scan (Scan sU), a Sparta SSH brute-force attack (Sparta), and an MQTT brute-force attack (MQTT BF). The dataset was carefully assembled by utilizing the tcpdump tool [105], which specializes in capturing and monitoring Ethernet traffic. Once this traffic was recorded, it was then stored in the widely-accepted pcap file format for further analysis and use.

During the research process, a variety of specialized tools were incorporated to ensure comprehensive data capture and simulation. Virtual machines were employed to emulate different network devices, offering a digital representation of physical computers. The renowned security auditing tool, Nmap, was utilized to perform scanning attacks, aiming to uncover potential network vulnerabilities. Additionally, the versatile open-source media player, VLC (VideoLAN Client), was used to mimic the data streams typically associated with IoT devices, especially simulating camera feeds. Lastly, for a more targeted approach, the tool MQTT-PWN [106] was harnessed. This specific tool was designed to execute brute-force attacks, attempting to break into systems by testing a multitude of credential combinations.

In the examined network setup [107], 12 MQTT sensors, a central broker, a specific machine that replicates camera feeds, and a system that acts as the attacker make up the

structure. Under standard conditions, these sensors actively send messages through the "Publish" command of MQTT. These messages are characterized by their random nature, both in content and length, which varies among the sensors. This variance is designed to imitate the diversity found in real-world IoT deployments. Parallelly, a simulation of a camera feed is run using the VLC media player, which transmits data through a UDP stream. To further achieve authenticity and imitate real-world network conditions, the emulators within the setup have been programmed to drop packets at specified rates: 0.2%, 1%, and 0.13%. Notably, while the distinct scenarios were being recorded, the operations of the network continued undisturbed. To cater to different research needs, the dataset is offered in two formats: the unaltered capture format, known as .pcap files, and another that comprises processed features. These features span across packet-based, unidirectional, and bidirectional metrics.

Five pcap files make up the dataset in [107], including normal.pcap, sparta.pcap, scan_A.pcap, mqtt_bruteforce.pcap, and scan_sU.pcap. Each file is a recording of a different scenario, including normal operation, a Sparta SSH brute-force attack, an aggressive scan, a MQTT brute-force attack, and a UDP scan. The background regular operations are included in the assault pcap files.The MQTT-IoT-IDS2020 dataset features offer valuable information for analyzing the network traffic and developing intrusion detection systems in the IoT domain.

- Packet flow analysis in MQTT-IoT-IDS2020 involves examining the characteristics of individual packets within the network traffic. These features encompass flags, length, MQTT message parameters, and more.
- Unidirectional flow analysis in MQTT-IoT-IDS2020 focuses on capturing and analyzing the traffic in one direction only, either from the source device to the destination device or vice versa.
- Bidirectional flow analysis considers the traffic in both directions simultaneously. It involves capturing and analyzing the communication between the source and destination devices in a bidirectional manner. In the case of two-way traffic flows, some features possess a pair of values: one representing the outgoing (or forward) flow and the other for the incoming (or backward) flow.

Table 6.1 and Table 6.2. display the features description and distribution of cases, respectively.

Table 6. 1: Dataset Feature Description[108]

Feature	Data Type	Description	Packet	Uniflow	Biflow
ip_src	Text	Source IP Address	yes	yes	yes
ip_dest	Text	Destination IP Address	yes	yes	yes
protocol	Text	Last layer protocol	yes		
ttl	Integer	Time to live	yes		
ip_len	Integer	Packet Length	yes		
ip_ flag_df	Binary	Don't fragment IP flag	yes		
ip_flag_mf	Binary	More fragments IP flag	yes		
ip_flag_rb	Binary	Reserved IP flag	yes		
prt_src	Integer	Source Port	yes	yes	yes
prt_dst	Integer	Destination Port	yes	yes	yes
proto	Integer	Transport Layer protocol (TCP/UDP)		yes	yes
tcp_flag_res	Binary	Reserved TCP flag	yes		
tcp_flag_ns	Binary	Nonce sum TCP flag	yes		
tcp_flag_cwr	Binary	Congestion Window Reduced TCP flag	yes		
tcp_flag_ecn	Binary	ECN Echo TCP flag	yes		
tcp_flag_urg	Binary	Urgent TCP flag	yes		
tcp_flag_ack	Binary	Acknowledgement TCP flag	yes		
tcp_flag_push	Binary	Push TCP flag	yes		
Tcp_flag_reset	Binary	Reset TCP flag	yes		
tcp_flag_syn	Binary	Synchronization TCP flag	yes		
Tcp_flag_fin	Binary	Finish TCP flag	yes		
num_pkts	Integer	Number of Packets in the flow		yes	Fwd & bwd
Mean_iat	Decimal	Average inter arrival time		yes	Fwd & bwd
std_iat	Decimal	Standard deviation of inter arrival time		yes	Fwd & bwd
min_iat	Decimal	Minimum inter arrival time		yes	Fwd & bwd
max_iat	Decimal	Maximum inter arrival time		yes	Fwd & bwd
num_bytes	Integer	Number of bytes		yes	Fwd & bwd
num_psh_flags	Integer	Number of push flag		yes	Fwd & bwd
num_rst_flags	Integer	Number of reset flag		yes	Fwd & bwd
num_urg_flags	Integer	Number of urgent flag		yes	Fwd & bwd
mean_pkt_len	Decimal	Average packet length		yes	Fwd & bwd
std_pkt_len	Decimal	Standard deviation packet length		yes	Fwd & bwd
min_pkt_len	Decimal	Minimum packet length		yes	Fwd & bwd
max_pkt_len	Decimal	Maximum packet length		yes	Fwd & bwd
mqtt_messagetype	Integer	MQTT message type	yes		
mqtt_messagelength	Binary	MQTT message length	yes		
mqtt_flag_uname	Binary	User Name MQTT Flag	yes		
mqtt_flag_passwd	Binary	Password MQTT flag	yes		
mqtt_flag_retain	Binary	Will retain MQTT flag	yes		
mqtt_flag_qos	Integer	Will QoS MQTT flag	yes		
mqtt_flag_willflag	Binary	Will flag MQTT flag	yes		
mqtt_flag_clean	Binary	Clean MQTT flag	yes		
mqtt_flag_reserved	Binary	Reserved MQTT flag	yes		
is_attack	Binary	1 if the instance represents an attack, 0 legitimate.	no	no	no

Table 6. 2: MQTT-IoT-IDS2020 feature statistics distribution [108]

	Classes	Total Instances	Normal	Attack
Biflow	Biflow_mqtt_bruteforce Biflow_normal Biflow_scan_A Biflow_scan_sU Biflow_sparta	16,696 86,008 25,693 39,664 91,318	2152 86,008 5786 17,230 77,202	14,544 X 19,907 22,434 14,116
Uniflow	Uniflow_mqtt_bruteforce Uniflow_normal Uniflow_scan_A Uniflow_scan_sU Uniflow_sparta	33,079 171,836 51,358 25,845 182,407	4205 171,836 11,561 34,409 154,175	28,874 X 39,797 22,436 28,232
Packet-flow	mqtt_bruteforce Normal scan_A Biflow_scan_sU Biflow_sparta	91,056,230 1,056,230 111,392 233,255 130,876,584	70,980,732 1,056,230 70,768 210,819 90,980,732	19,895,852 X 40,624 22,436 39,895,852

In our research, we specifically focus on utilizing the unidirectional flow analysis from the MQTT-IoT-IDS2020 dataset. Unidirectional flow analysis allows us to examine the traffic in one direction only, providing valuable insights into the network traffic from the source device to the destination device or vice versa. This analysis enables us to identify specific features and attributes that are indicative of normal operation or potential attacks within the MQTT-IoT network.

6.3 Dataset Pre-processing

This part addresses the methods used to prepare the dataset before using it to perform machine learning tasks. Data cleansing, feature selection, normalization, and controlling class imbalance are among the pre-processing stages. By performing these pre-processing steps, the dataset is optimized for training and evaluating machine learning models, ensuring accurate and reliable results in the subsequent stages of the study.

Data preprocessing is essential in preparing the data for machine learning models [109]. The first step in the preprocessing involves combining the five files for the uniflow network level by implementing a Python script. The resulting combined CSV file contained binary and multi-class label attributes.

The preprocessing involved cleaning the data by removing repeated values and handling missing ones. To avoid the bias or undue influence of specific features, the "ip_src" (source IP address) and "ip_dst" (destination IP address) features were dropped. By excluding these attributes, the analysis and modeling process can focus more accurately on other relevant features and their contributions to the dataset.

To address the class imbalance, the oversampling technique SMOTE (Synthetic Minority Oversampling Technique) [110] is employed. It generates synthetic data points by interpolating between existing instances of the minority class and creates additional instances of the minority class. However, to avoid introducing bias into the model, we shuffle the dataset to randomize the order of the instances before applying SMOTE.

To evaluate how effectively the model works, the dataset was partitioned into training and testing sets at an 80:20 distribution ratio. All features except the target attribute were then normalized to ensure they were on the same scale, which is crucial for optimizing machine learning models. This process preserves the relative differences in values while ensuring that all features have comparable scales.

A key factor in obtaining the best outcomes is the choice of data characteristics. We used the **SelectKBest** method for feature selection. This method selects the top **k** features=10 in our case that are most impactful in predicting the target variable. These curated features are then saved into separate data structures for both binary and multiclass classifications.

As shown in Figures 6.1 and 6.2, respectively, the feature important scores are calculated for both binary and multiclass classification. The features were ranked according to their scores, and the top 10 features were chosen to be used in the model.

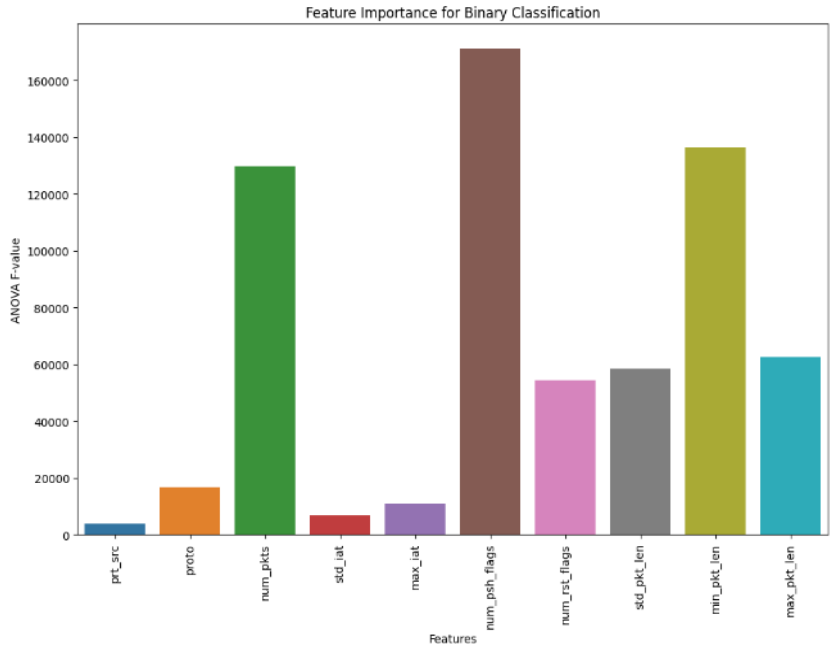

Figure 6. 1: Binary Classification Top 10 Feature Selection

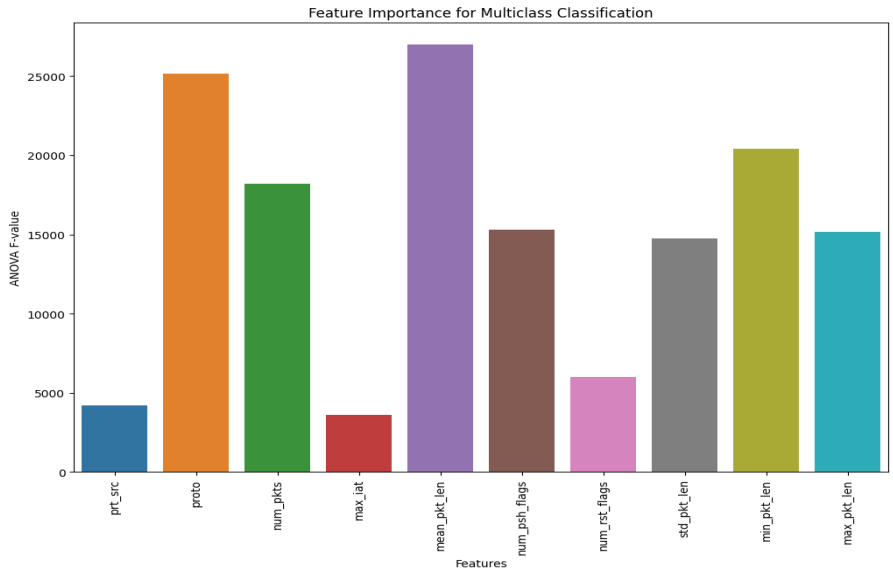

Figure 6. 2: Multiclass Classification Top 10 Feature Selection

6.4 Methodology

In our study, we implement Bagging, Boosting, and Stacking Ensemble techniques and figure out the effectiveness of these powerful methods in detecting the attacks found in the MQTT-IoT-IDS2020 dataset. Figure 6.3 demonstrates our workflow to accomplish our proposed IDS. Our approach for this project can be pointed out in these steps:

- Use a most recent real-world Dataset," MQTT-IoT-IDS2020," in an IoT environment to determine anomalies in combination with various ML models. The Dataset was pre-processed, and the most significant features were extracted.
- All intrusion detection algorithms are implemented and validated using Python and Sklearn libraries.
- Training the models by using ensemble learning algorithms.
- Evaluation analysis of the models using the standard machine learning classification metrics was considered to measure the efficiency of these models and predict the best-performing model.

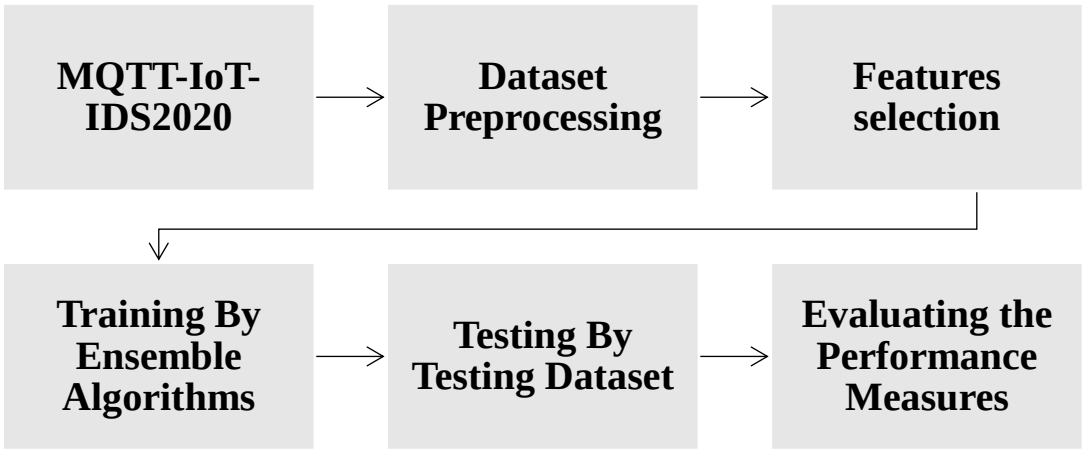

Figure 6. 3: Methodology Workflow

Bagging, Boosting, and Stacking ensemble techniques are implemented with 5-fold cross-validation to get a reliable performance estimation. Cross-validation provides a means to assess the robustness of ensemble techniques by evaluating their performance on different subsets of the data. This evaluation helps identify whether the ensemble's performance remains consistent or varies significantly across different data partitions, thus providing insights into the ensemble's stability and reliability.

6.5 Experimental Setup

We conducted the training and validation of the selected Ensemble models on a laptop running a 64-bit Windows 11 operating system and 16 GB RAM. The computer runs on an AMD Ryzen 7 5700U processor featuring Radeon Graphics and has a speed of 1.80 GHz.

Every experiment is carried out using the Python programming language (version 3.9.16) via several popular machine learning libraries, especially Scikit-learn, found in Google Colaboratory, which is also used to derive the performance and statistical results.

6.6 Performance evaluation metrics

In this section, we elucidate the criteria for evaluation that were used to determine whether our findings were efficient. when implementing ensemble techniques in machine learning.

When evaluating the classification on a test dataset compared to the training dataset, we can identify four possible outcomes.

- True Positives (TP), which represent the packets correctly identified as malicious.
- True Negatives (TN) denote the packets that have been accurately flagged as benign.

- False Positives (FP) occur when benign packets are mistakenly tagged as malicious.
- False Negatives (FN) refer to the instances where malicious packets are wrongly identified as benign.

In our study, we employed a range of metrics to evaluate the efficiency of the classifier, namely Precision, Recall, F1-score, and Accuracy [111]. Ideally, we aimed to optimize every metric, keeping in mind that each has a value scale ranging from 0 to 1. A metric closer to 1 indicates a better performance of the classifier, whereas a value closer to 0 suggests room for improvement. By striving for the upper end of this scale, our goal was to ensure the classifier's maximum accuracy and efficiency in its predictions. These include.

- **Precision(P)** represents the percent of correctly identified malicious samples (equation 1).
- **Recall (R)** refers to the fraction of all malicious samples that were correctly identified (equation 2).
- **The F1 score** represents the balance between precision and recall by taking their harmonic mean. It is particularly valuable when the distribution of classes is uneven or imbalanced, ensuring that neither precision nor recall is disproportionately favored (equation 3).
- **The accuracy** metric indicates the proportion of samples that were accurately classified in equation (4).

These formulas can be used to calculate these metrics:

$$\boldsymbol{P} = \frac{TP}{TP + FP} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots .. (1)$$

$$\boldsymbol{R} = \frac{TP}{TP + FN} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

$$\boldsymbol{F1 - score} = 2 * \frac{P.R}{P + R} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots .. (3)$$

$$\boldsymbol{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots . (4)$$

- The **Confusion matrix** [112], a powerful tool, is also utilized for visualizing machine learning models' performance. In binary classification (Figure 6.4) it consists of four entries: True Positives (correctly detected attacks), True Negatives (correctly identified legitimate flows), False Positives (legitimate flows mistaken as attacks), and False Negatives (attacks mistaken as legitimate flows).

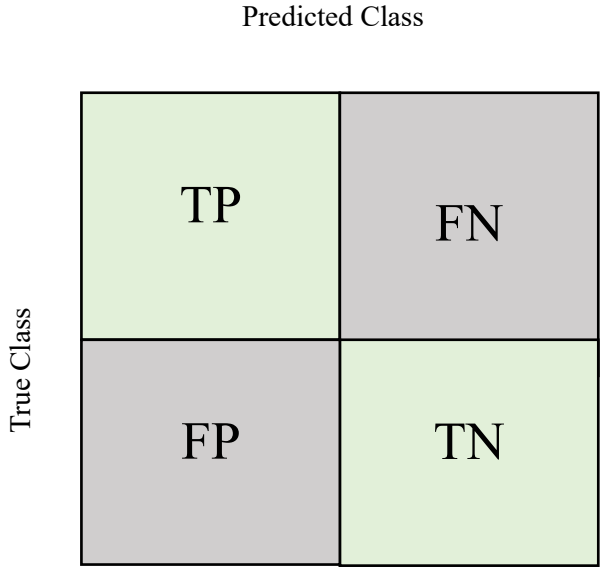

Figure 6. 4: Binary Classification Confusion Matrix

In a multiclass classification (Figure 6.5), the matrix has dimensions corresponding to the number of classes. The matrix's columns represent projected class occurrences, whereas each row represents actual class occurrences. The values show how many instances were rightly classified for every specific class and also point out instances that got classified incorrectly.

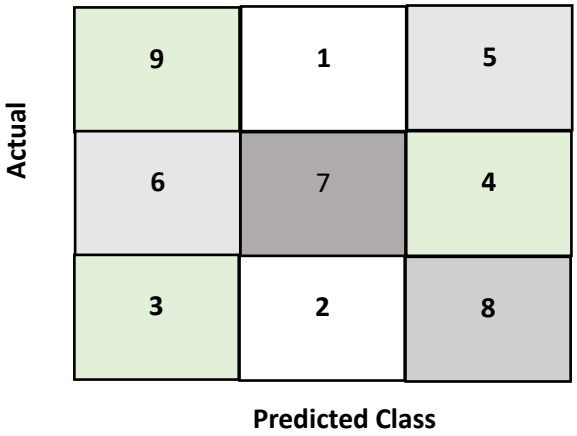

Figure 6. 5: Multiclass Classification Confusion Matrix

- The **Characteristic curve of the receiver operating (ROC) Curve** [113] is also used as it offers a glimpse of the model's effectiveness regardless of the threshold.
- The model's performance utilizing the ROC curve was thoroughly evaluated using the AUC metric [114]. It signifies the likelihood that the model will correctly prioritize a random positive sample over a random negative one. A higher AUC value reflects a more effective model, showcasing its enhanced capability to differentiate between positive and negative samples.

Chapter 7

Results and Analysis

In this section, we showcase the findings derived from our experimental evaluations, focusing on both binary and multiclass classifications based on uniflow features. We highlight the efficacy of the detection solution grounded in machine learning methodologies and also explore the effectiveness of the ensemble techniques in classifying the attacks. Through extensive testing and in-depth evaluation, our goal is to delve deeper into the capabilities, constraints, and overall effectiveness of the approach we have proposed. These results and analysis contribute to the advancement of cybersecurity in MQTT-IoT networks and provide valuable insights for future research and system enhancements.

7.1 Bagging Binary Classification

For binary classification, our dataset has been used to detect the MQTT-IoT attacks mentioned earlier. We utilized the Bagging Classifier, with a Decision tree with a maximum depth of 20 as the base estimator. The Bagging technique works by creating several subsets of the original data, training a base learner (150 Decision Trees) for each, and then combining the outputs.

We chose Decision Trees (DT) as our base model due to their transparent and interpretable decision-making process. Their robustness to outliers, scalability to large datasets, and quick training times make them ideal. Furthermore, their compatibility with ensemble techniques enhances model performance while mitigating overfitting concerns.

The Bagging Classifier has been trained on the dataset and then tested on unseen data. In terms of how outputs are combined, the ensemble model typically utilizes majority voting for our binary classification problem. The predicted class that receives the most votes is chosen as the final output, with votes for the predicted class coming from each base model.

We employed a k-fold cross-validation approach, specifically with five folds. The accuracies observed across the folds provide insight into the technique's consistency and efficiency on various dataset subsets.

We handled class imbalance using SMOTE to generate synthetic samples and employed a RandomizedSearchCV for hyperparameter tuning, which is an excellent choice when the

hyperparameter space is large. It selects random combinations of the parameters for a given number of iterations.

Bagging Binary Results and Analysis:

Table 7.1, Figure 7.1, and Figure 7.2 show the results of the bagging ensemble from the classification report, which gives an overall picture of the model's efficiency. We are getting precision, recall, and F1 scores of approximately 1.0 for both classes. The ideal balance between false positives and false negatives is demonstrated by these reliable results. The accuracy for the 'attack' class is 100%, and for the 'benign' class is 100%. The model's accuracy is approximately 100%, showing how well it differentiates between the two groups. Across the five folds, it consistently achieved a training accuracy of approximately 99.99%.

Testing accuracy, which is more indicative of the model's generalization capability to unseen data, remained consistently high across the folds, with an average of 99.96%. Only a slight variation was noted in the fourth fold, where it marginally dropped to 99.95%, suggesting the robustness of our model.

Out of 75218 instances of class Benign, the model correctly predicted 75216, misclassifying only 2. Similarly, out of 75257 instances of the class Attack, the model correctly predicted 75248, with only nine misclassifications. These results corroborate the high precision and recall values observed in the classification report.

The AUC score is 1.0, which is excellent and indicates a strong model's ability to discriminate across classes.

Table 7. 1: Binary bagging Classification Results using 5-fold Cross Validation

Class	Accuracy	Precision	Recall	F1 score	Support
Attack	99.96%	100%	100%	100%	75257
Benign	99.94%	100%	100%	100%	75218

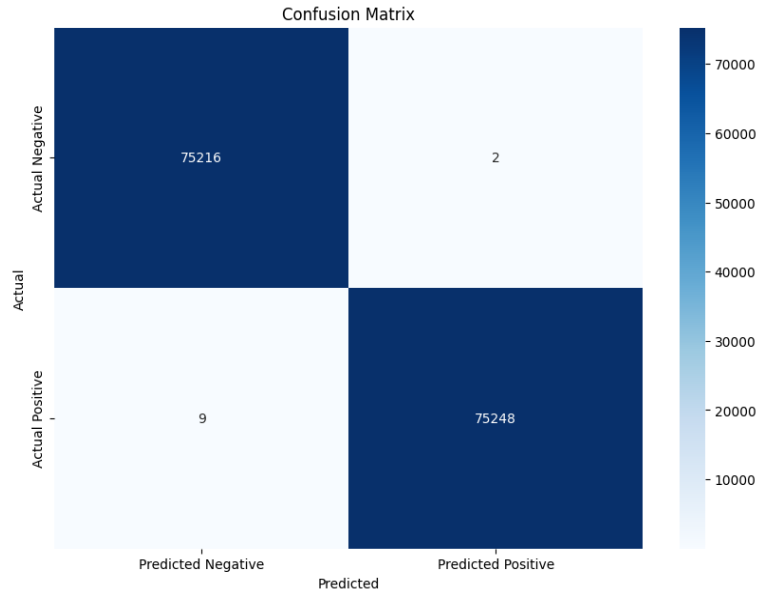

Figure 7. 1: Bagging Binary Confusion Matrix

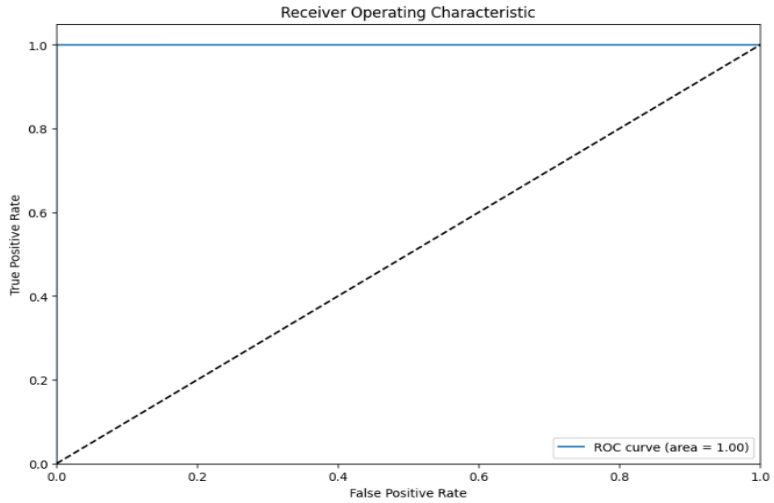

Figure 7. 2: Bagging Binary ROC Curve

7.2 Bagging Multiclassification

For Multiclassification Bagging, the central process involved using Decision Trees as the base estimator within a Bagging ensemble.

Given the presence of a class imbalance in our dataset, we deployed SMOTE ensuring our model was not biased towards the majority class and could learn useful patterns from all classes equally.

The base classifier used was a Decision Tree Classifier with an optimal depth of 20 and a minimum sample split of 4. The base classifier was then used in a Bagging Classifier, training it with 196 estimators obtained from RandomSearch, and sampling 80% of instances and features during the process. After training the classifier, accuracy was calculated on both the training and testing datasets. Additionally, KFold cross-validation with five splits was performed to validate the robustness of the model. A classification report was generated, providing insights into the evaluation metrics.

Bagging Multiclassification Results and Analysis

For multi-classification, we classify instances into one of five classes: Benign, MQTT_BF, Scan_A, Scan_sU, and Sparta. The outcomes from the classification report are displayed in Table 7.2, Figures 7.3, and 7.4.Table 7. 2: Multi-Classification Bagging Results using 5-fold Cross Validation.

Class	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Support**
Benign	93%	95%	94%	94%	36482
MQTT_BF	95%	96%	96%	96%	36444
Scan_A	97%	96%	97%	96%	36455
Scan_sU	94%	94%	94%	94%	36541
Sparta	95%	95%	96%	95%	36485

After running the model, the training accuracy achieved was a commendable 99.36%, while the testing accuracy stood at 95.20%. The confusion matrix, as shown in Figure 7.3, revealed that the classifier was able to tell the classes apart quite well, with the majority of the instances correctly classified. The cross-validated classification report further affirmed the model's performance, showcasing an overall accuracy of 95% across classes. Individual class accuracies were: Benign at 93.66%, MQTT_BF at 95.47%, Scan_A at 97.01%, Scan_sU at 94.17%, and Sparta at 95.61%. The area under the ROC curve (AUC) shown in Figure 7.4 for each class was impressive, with most classes scoring above 0.95, indicating the classifier's strong ability to distinguish between positive and negative instances for each class.

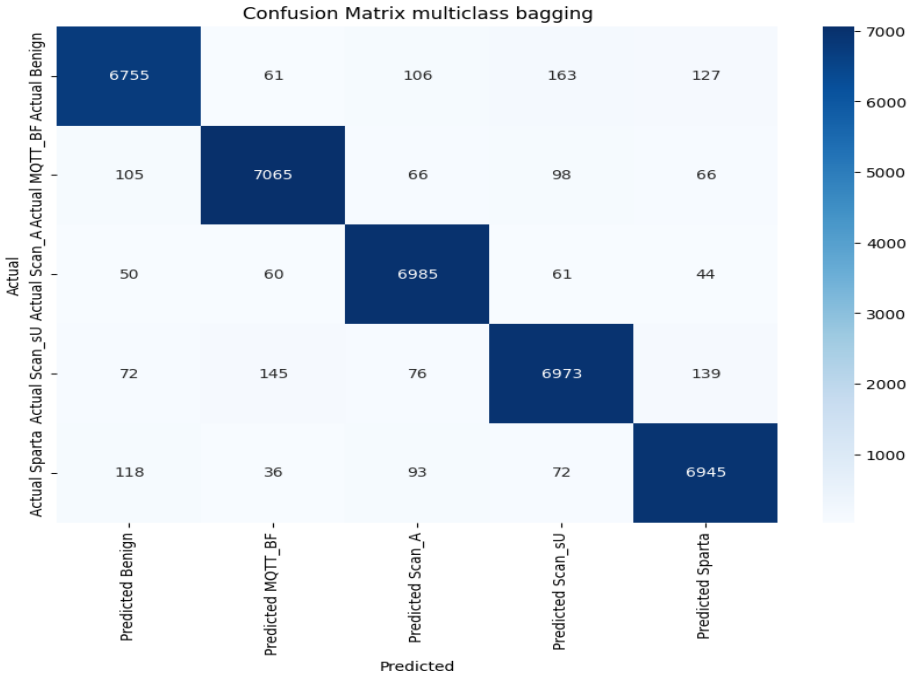

Figure 7. 3: Multiclassification Bagging Confusion Matrix

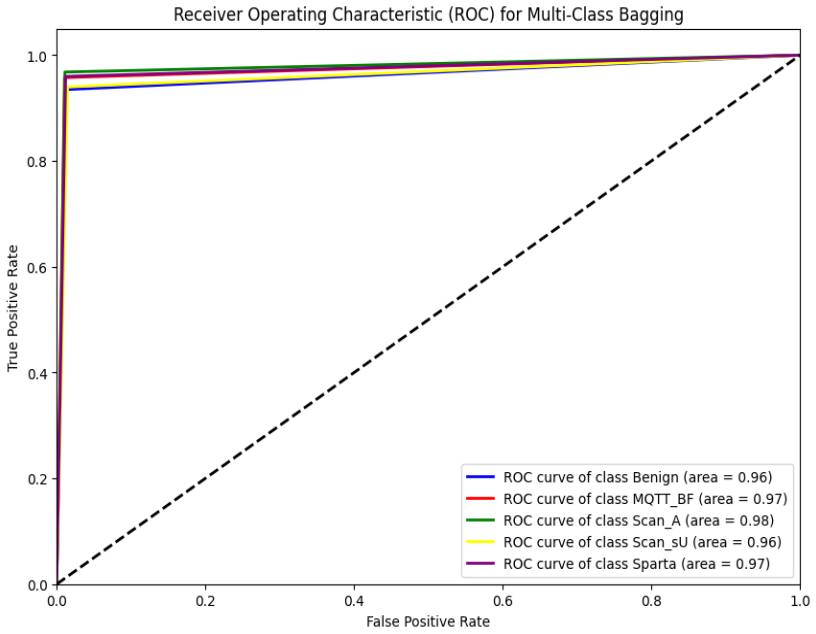

Figure 7. 4: Bagging Multiclassification ROC curves

7.3 Boosting Binary Classification

For Binary classification, we implemented an ensemble learning model using the Adaptive Boosting (AdaBoost) technique with a base estimator being a Decision Tree Classifier. The utilization of AdaBoost with a Decision Tree (DT) offers a powerful combination for classification tasks. AdaBoost transforms a collection of weak learners, often represented by shallow decision trees, into a single strong learner, effectively enhancing overall prediction performance. Despite DT's tendency to overfit, AdaBoost curtails this risk by aggregating outcomes from several DTs, thus improving generalization to unseen data. Additionally, this method is both flexible, as it can handle numerical and categorical data, and efficient, due to the computational simplicity of decision trees.

We employed Random Search for hyperparameter tuning, which effectively explored a broader range of parameter values and identified the best parameters for our AdaBoost model: a decision tree depth of 10, 150 estimators, and a learning rate of 0.1. To ensure robust estimation of the model's performance and to avoid overfitting, we incorporated 5-fold cross-validation into the Random Search process. This approach divided the dataset into five subsets and iteratively used four for training and one for validation.

We used SMOTE, a powerful technique for creating synthetic samples from the minority class, to address difficulties with class imbalance in our data.

The performance of the implemented AdaBoost model was assessed, along with the model's overall accuracy, for each class ('benign' and 'attack').

Boosting Binary Classification Results and Analysis

In this section, we present the binary classification outcomes derived from the boosting ensemble, employing AdaBoost with Decision Tree (DT) as the foundational estimator.

As shown in Table 7.3, for the 'benign' class, our model yielded a precision of 1.0, a recall of 1.0, and an F1-score of 1.0. These figures indicate a high success rate in accurately identifying and capturing most of the 'benign' instances. Meanwhile, for the 'attack' class, our model achieved a precision of 1.0, a recall of 1.0, and an F1-score of 1.0. The higher precision for 'attack' instances demonstrates that our model is cautious and effective in minimizing false positives when predicting an 'attack'. The accuracy for the 'benign' class was recorded at 99.96%, while the 'attack' class had an accuracy of 99.94%, signaling the overall competence of our model in making correct classifications.

Table 7. 3: Binary Classification Boosting Results using 5-fold Cross Validation

Class	Accuracy	Precision	Recall	F1 score	Support
Attack	99.94%	100%	100%	100%	75237
Benign	99.96%	100%	100%	100%	75237

As shown in Figure 7.5, the model correctly classified 75207 'benign' and 75193 'attack' instances, showcasing its robust performance. However, it also incorrectly classified 30 'benign' and 44 'attack' instances.

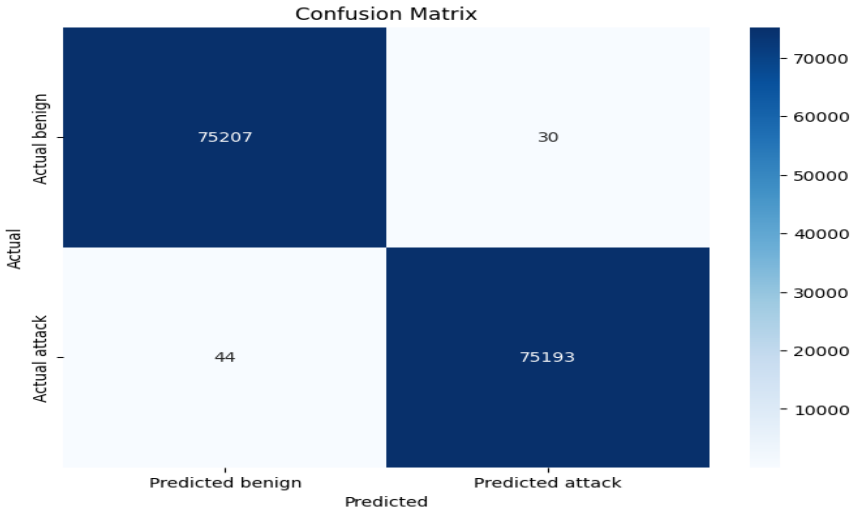

Figure 7. 5: Binary boosting Classification Confusion Matrix

Figure 7.6 illustrates further study of our model's performance using the ROC curve and associated Area Under the Curve (AUC) score. Our AdaBoost model achieved an AUC score of 1.0, a significant achievement that demonstrates an excellent capacity to distinguish between 'benign' and 'attack' instances.

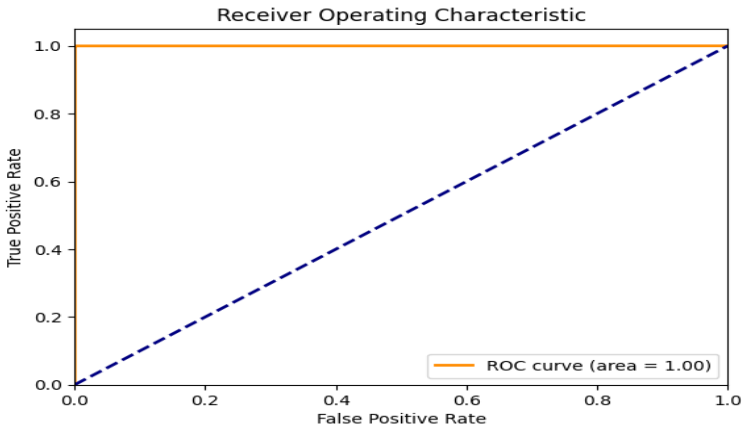

Figure 7. 6: Binary Boosting ROC Curve

7.4 Boosting Multiclass Classification

This section describes the model's performance for multiclass classification and provides examples of how our boosting ensemble strategy works.

As mentioned earlier, a key aspect of boosting is that it places more weight on instances that were previously misclassified, focusing on the harder-to-classify instances in successive iterations, ultimately enhancing the model's performance.

We focused on the MQTT-IoT-IDS2020 dataset, using the top 10 multi-class features. The data underwent a standardization process using the StandardScaler to enhance the model's convergence speed and performance. Using SMOTE to handle the imbalance, we employed the XGBoost classifier, a gradient boosting algorithm, and tuned its hyperparameters using RandomizedSearchCV with a Stratified K-Fold cross-validation scheme which helped ensure the selected parameters generalize well to unseen data.

Boosting Multiclass Classification Results and Analysis

The multiclassification outcomes for the boosting ensemble using XGBoost with DT as the base estimator are shown in this section.

As shown in Table 7.4, the training accuracy achieved was an impressive 99.65%, while the testing accuracy stood at 96.93%. A comprehensive cross-validated classification report provided further insights: the model showcased all metrics as 97% across all classes, signifying high reliability. The individual accuracies for the classes 'Benign,' 'MQTT_BF,' 'Scan_A,' 'Scan_sU,' and 'Sparta' were 96.18%, 97.17%, 98.08%, 96.46%, and 96.93% respectively.

The confusion matrix in Figure 7.7 provided a visual demonstration underscoring the model's proficient classification capability. For instance, the class 'Scan_A' witnessed the highest true positives (35,756 instances correctly identified) with minimal misclassifications, such as 149 mistaken as 'Benign' and 198 as 'MQTT_BF.'

The ROC curves in Figure 7.8 for each class further highlighted the model's capacity to discriminate between the classes, with AUC values accentuating its efficiency. The Area Under the Curve (AUC) values obtained effectively demonstrated the model's prowess in distinguishing between different traffic types. Benign, MQTT_BF, Scan_sU, and Sparta classes achieved an AUC of 0.98, and the Scan_A class stood out with a near-perfect score of 0.99. These results highlight the model's robust ability to accurately detect various attack patterns and benign traffic, signifying its effectiveness in intrusion detection for IoT contexts.

Table 7. 4: Multi-Classification Boosting Results using 5-fold Cross Validation

Class	Accuracy	Precision	Recall	F1 score	Support
Benign	96.18%	97%	96%	96%	36482
MQTT_BF	97.17%	97%	97%	97%	36444
Scan_A	98.08%	98%	98%	98%	36455
Scan_sU	96.46%	97%	96%	96%	36541
Sparta	96.93%	97%	97%	97%	36485

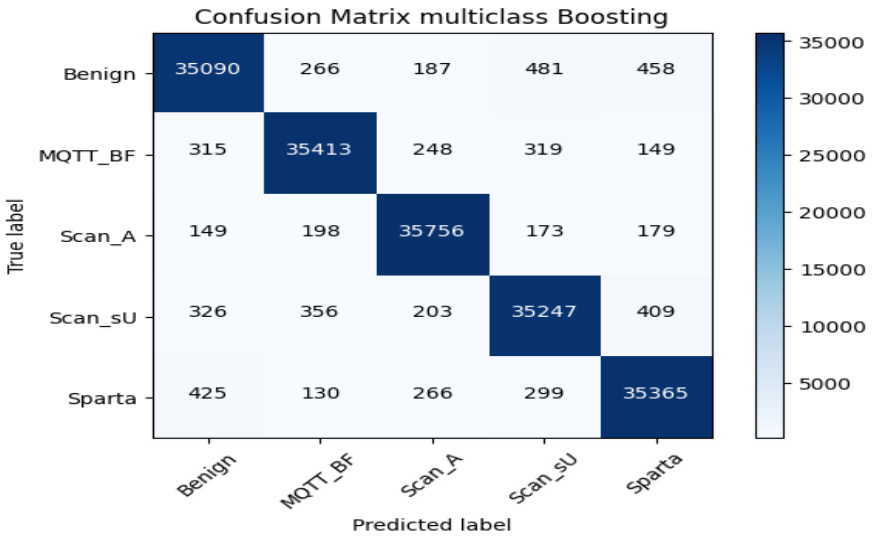

Figure 7. 7: Multiclassification Boosting Confusion Matrix

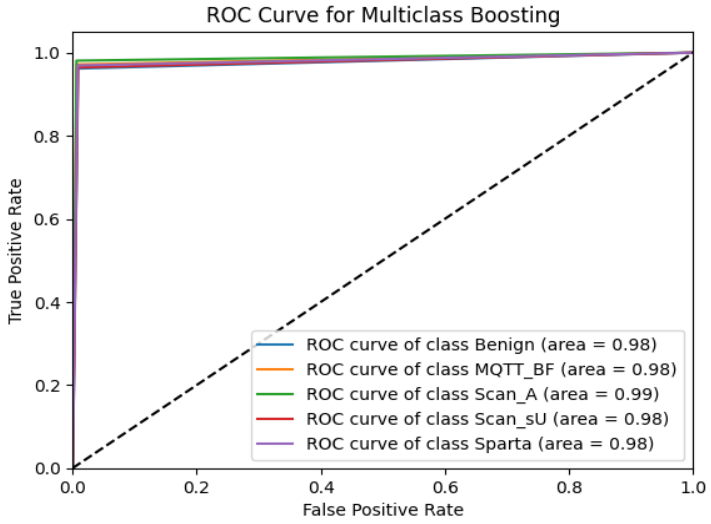

Figure 7. 8: Multiclassification Boosting ROC Curves

7.5 Stacking Binary Classification

In this section, we explore the utilization of a stacking ensemble approach for binary classification, specifically focused on distinguishing between the benign and attack classes.

The ensemble comprises three base classifiers: K-Nearest Neighbors (KNN), XGBoost, and Decision Tree (DT), which contribute their individual predictions to a final meta learner, Logistic Regression (LR). By combining the strengths of these diverse classifiers, the stacking ensemble aims to enhance the overall classification performance, effectively handling the complexities and imbalances present in the dataset. We used the SMOTE to correct the class disparity. After then, the data was divided into training and testing sets using StratifiedShuffleSplit.

We incorporated 5-fold cross-validation (cv=5) to the Randomized Search procedure to more thoroughly evaluate the model's generalizability and reduce the risk of overfitting. The dataset is then partitioned into five subsets, or "folds". The model is then trained and validated five times, using various combinations of one validation fold and four training folds. The model's capacity to generalize to new data is then estimated using an average of the performance measures throughout the five iterations. The integration of Randomized Search with 5-fold cross-validation thus ensured an efficient, robust, and rigorous approach to model tuning and validation, enabling us to develop a high-performing stacked ensemble model for the binary classification task at hand. Finally, we leveraged an array of evaluative metrics, encompassing accuracy, classification report, confusion matrix, and the Receiver Operating Characteristic (ROC) curve, to assess the efficacy of our results.

Stacking Binary Classification Results and Analysis

The binary classification results for the stacking ensemble model built with the three base classifiers Decision Tree (DT), K-Nearest Neighbors (KNN), and XGBoost are presented in this section. These classifiers were chosen for their complementary strengths in terms of model bias and variance. Logistic Regression (LR) was used as the final estimator, with the aim of making the most out of the diverse predictions of the base classifiers.

The model exhibited an impressive performance with a training accuracy of 0.9999 and a testing accuracy of 0.9996. The classification report provided deeper insights, indicating an almost perfect precision, recall, and F1-score for both the 'attack' and 'benign' classes, as shown in Table 7.5. Both classes saw an accuracy rate of 1.00, indicating a high level of correct predictions.

For the 'benign' class, out of 45,122 instances, 45,099 were correctly predicted, with only 22 misclassifications. Similarly, for the 'attack' class, 45,106 out of 45,121 instances were predicted accurately, with just 16 instances misclassified.

Table 7. 5: Binary Classification Stacking Results using 5-fold Cross Validation

Class	Accuracy	Precision	Recall	F1 score	Support
Attack	100%	100%	100%	100%	45121
Benign	100%	100%	100%	100%	45122

Figure 7.9 shows the binary confusion matrix providing further detailed insights into our model's performance. For the 'benign' class, out of 45,122 instances, 45,099 were correctly predicted, with only 22 misclassifications. Similarly, for the 'attack' class, 45,106 out of 45,121 instances were predicted accurately, with just 16 instances misclassified.

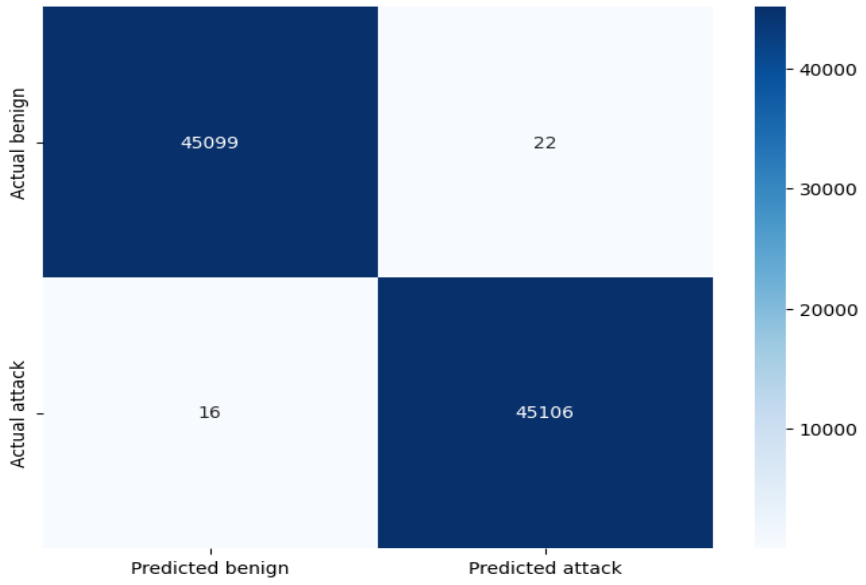

Figure 7. 9: Binary Classification Stacking Confusion Matrix

A ROC AUC score of 1.0, demonstrating its outstanding capacity to differentiate between the two classes, also served to support the ensemble's performance.

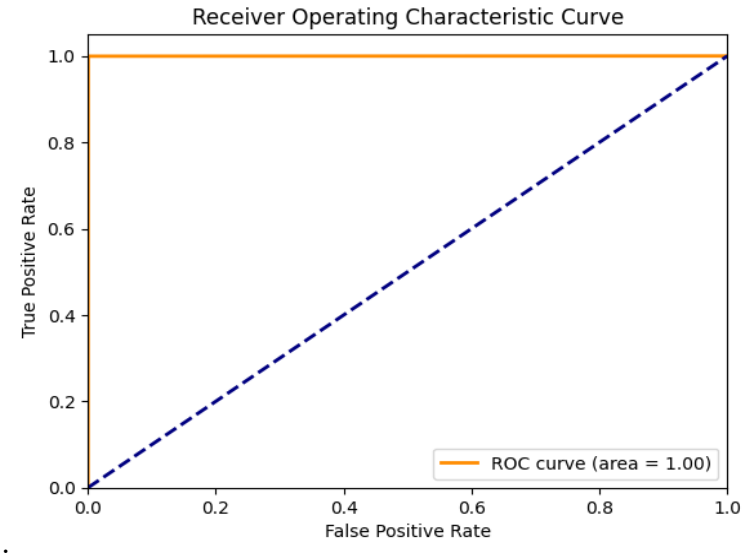

Figure 7. 10: Binary Stacking ROC Curve

It is clearly shown from these results that the stacking ensemble model, with its tuned base classifiers and final estimator, displays excellent performance on the test data. It is especially effective in identifying 'MQTT-attack' instances, which was our primary focus.

7.6 Stacking Ensemble Multiclass Classification

In this section, we utilize a stacking ensemble for multiclassification of the MQTT-IoT-IDS2020 dataset, focusing on distinguishing benign data from four types of attacks: 'MQTT_BF,' 'Scan_A,' 'Scan_sU,' and 'Sparta.' The ensemble combines the predictions of three base classifiers, Decision Tree, K-Nearest Neighbors, and XGBoost, using Logistic Regression as a final meta-learner.

To rectify the pronounced class imbalance , we utilized SMOTE to achieve equilibrium. The hyperparameters are refined by Randomized Search to lessen the risk of overfitting and to offer an accurate estimation of the model's capacity for generalization.

Stacking Multiclassification Classification Results and Analysis

The multiclassification results for the stacking ensemble, which combines the predictions of the three basic classifiers Decision Tree, K-Nearest Neighbors, and XGBoost, are presented in this section. The last meta-learner used was Logistic Regression.

Our results, shown in Table 7.6, indicated that the stacking ensemble approach was effective for this multiclassification problem. The cross-validated classification report highlights that the model consistently maintains a high precision, recall, and F1-score of above 0.97 across all

categories: Benign, MQTT_BF, Scan_A, Scan_sU, and Sparta. These metrics suggest that the model not only correctly predicts positive observations but also successfully captures most of the relevant results. The overarching accuracy of approximately 97.44% underscores the model's reliable performance. The presented confusion matrix in Figure 7.11 reinforces this, showing that the majority of predictions fall on the diagonal, implying correct classifications. Delving deeper into individual class accuracies reveals impressive scores, with Scan_A leading at 98.29% and the other categories not far behind. The AUC values, as shown in Figure 7.12, predominantly hovering around the 0.98 to 0.99 range, further solidify the model's efficiency. These scores indicate the classifier's high true positive rate and its ability to minimize false positives. Top of Form

Table 7. 6: Multi-Classification Stacking results using 5-fold Cross Validation

Class	Accuracy	Precision	Recall	F1 score	Support
Benign	96.92%	97%	97%	97%	29200
MQTT_BF	97.93%	98%	98%	98%	29139
Scan_A	98.29%	98%	98%	98%	29250
Scan_sU	97.31%	97%	97%	97%	29264
Sparta	97.45%	97%	97%	97%	29072

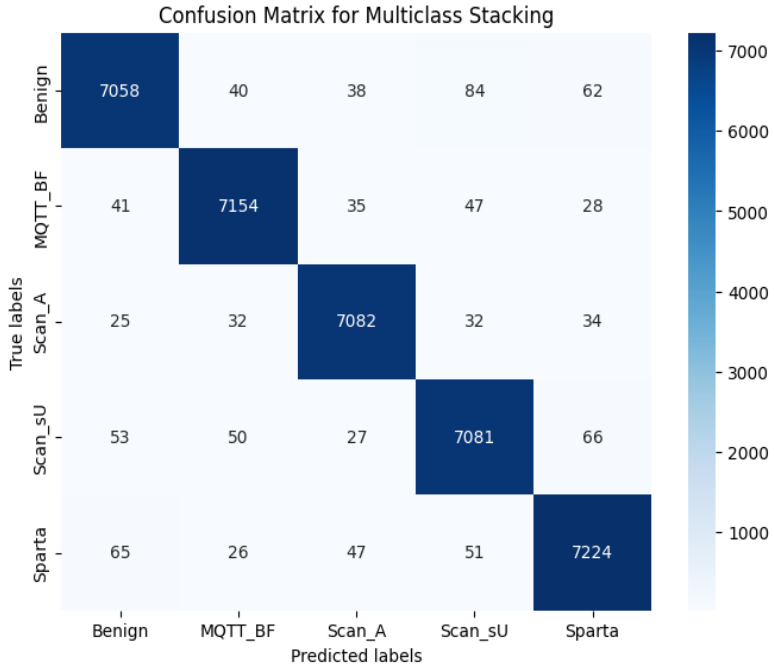

Figure 7. 11: Multiclassification Stacking Confusion Matrix

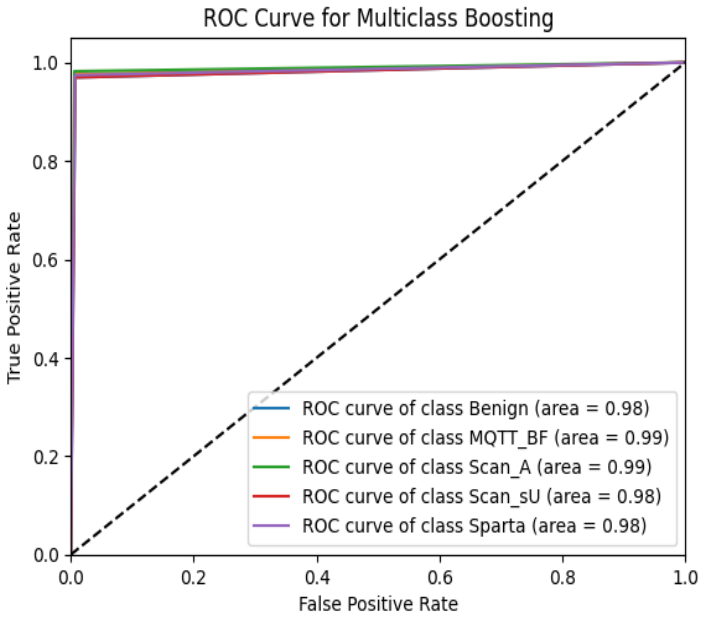

Figure 7. 12: Multiclassification Stacking ROC Curves

The results clearly show and underscore the efficacy of the stacking ensemble approach combined with SMOTE for multiclassification problems on imbalanced MQTT-IoT-IDS. They demonstrate that such an approach can deliver robust performance across all classes and handle the nuances and complexities of distinguishing between benign.

Chapter 8

In this final chapter, we consolidate the major findings of our research, emphasizing their implications within the broader context of the security of the MQTT-IoT field. We posed at the outset of this study, addressing each with the insights garnered through our research. In light of these conclusions, we identify areas that warrant further exploration and suggest potential directions for future work. Our objective is not only to draw a conclusive end to this particular research journey but also to inspire further investigations and advancements in this significant area.

8.1 Conclusion

In this thesis, we assessed how well ensemble machine-learning approaches performed when used to detect cybersecurity attacks for MQTT-IOT networks. The primary focus was on the three popular ensemble methods Bagging, Boosting, and Stacking. This study has successfully demonstrated the effectiveness of these ensemble methods in addressing both binary and multiclassification problems for the MQTT-IoT-IDS2020 dataset.

The chosen models addressed binary and multiclassification problems, demonstrating significant distinguishing between 'benign' and 'attack' instances and among various types of attacks such as MQTT_BF, Scan_A, Scan_sU, and Sparta.

Our binary classification models achieved impressive results, with AUC scores reaching up to 1.0, indicating an excellent ability to distinguish between classes. Moreover, we observed commendable accuracy, precision, recall, and F1 scores across both the 'benign' and 'attack' categories, suggesting a balanced and efficient performance.

The results of multiclassification were equally remarkable, with consistently high metrics across all classes and AUC scores extending to a perfect 1.0, reflecting flawless classification for some classes.

The techniques applied in the study were proficient in dealing with the class imbalance inherent in the MQTT-IoT-IDS2020 dataset. The use of SMOTE for synthetic sample generation and class weights adjustment was instrumental in achieving this balance and ultimately enhancing the performance of our models. The study results have reinforced the value of carefully tuning hyperparameters to optimize model performance. For this reason, RandomizedSearchCV has shown to be a time-effective method in the context of a large hyperparameter space. Moreover, the confusion matrices provided detailed insights into correct and incorrect classifications, assisting us in understanding the areas where the models excelled or needed improvement.

8.2 Future Work

While the results were largely positive, they also highlighted areas that could benefit from further refinement. Future work could consider more sophisticated oversampling techniques or alternative ways of handling class imbalance. Furthermore, given the model's success in predicting certain attack types, deeper investigation into the feature importance of these classifications might provide valuable insights.

The models could potentially benefit from further hyperparameter tuning. More computationally intensive methods like GridSearchCV could be employed to find the optimal parameters, provided sufficient computational resources are available. We could also consider exploring other base models for the ensemble methods to see if they provide better results.

Further validation of the models on additional datasets would be beneficial to ensure that the models generalize well across different scenarios and types of attacks. Lastly, the exploration of real-time implementation of these models would be a significant advancement in MQTT-IoT security.

This study underscores the promising capabilities of ensemble methods which can open the door for more secure and dependable IoT settings with further development and research.

References

[1] K. Fizza *et al.*, "QoE in IoT: a vision, survey and future directions," *Discover Internet of Things*, vol. 1, no. 1, Dec. 2021, doi: 10.1007/s43926-021-00006-7.

[2] N. K. Velayudhan, P. Pradeep, S. N. Rao, A. R. Devidas, and M. V. Ramesh, "IoT-Enabled Water Distribution Systems—A Comparative Technological Review," *IEEE Access*, vol. 10, pp. 101042–101070, Sep. 2022, doi: 10.1109/access.2022.3208142.

[3] G. V. Arbex, K. G. Machado, M. Nogueira, D. M. Batista, and R. Hirata, "IoT DDoS Detection Based on Stream Learning," in *Proceedings of the 2021 12th International Conference on Network of the Future, NoF 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/NoF52522.2021.9609940.

[4] E. Al-Masri *et al.*, "Investigating Messaging Protocols for the Internet of Things (IoT)," *IEEE Access*, vol. 8, pp. 94880–94911, 2020, doi: 10.1109/ACCESS.2020.2993363.

[5] Sri Venkateshwara College of Engineering. Department of Electronics and Communication Engineering, Institute of Electrical and Electronics Engineers. Bangalore Section, IEEE Computer Society, and Institute of Electrical and Electronics Engineers, *RTEICT-2017 : 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology : proceedings : 19-20 May 2017*.

[6] L. García, L. Parra, J. M. Jimenez, J. Lloret, and P. Lorenz, "IoT-based smart irrigation systems: An overview on the recent trends on sensors and iot systems for irrigation in precision agriculture," *Sensors (Switzerland)*, vol. 20, no. 4. MDPI AG, Feb. 02, 2020. doi: 10.3390/s20041042.

[7] R. A. Atmoko and D. Yang, "Online Monitoring & Controlling Industrial Arm Robot Using MQTT Protocol," in *Proceedings of the 2018 International Conference on Robotics, Biomimetics, and Intelligent Computational Systems, Robionetics 2018*, Institute of Electrical and Electronics Engineers Inc., Mar. 2019, pp. 12–16. doi: 10.1109/ROBIONETICS.2018.8674672.

[8] E. Di Paolo, E. Bassetti, and A. Spognardi, "Security assessment of common open source MQTT brokers and clients Automatic Spreading Factor allocation in LoRaWAN multi-gateway environment View project SeismoCloud View project Security assessment of common open source MQTT brokers and clients," 2021. [Online]. Available: https://mosquitto.org/documentation/dynamic-security

[9] H. C. Hwang, J. S. Park, and J. G. Shon, "Design and Implementation of a Reliable Message Transmission System Based on MQTT Protocol in IoT," *Wirel Pers Commun*, vol. 91, no. 4, pp. 1765–1777, Dec. 2016, doi: 10.1007/s11277-016-3398-2.

[10] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S-A Publish/Subscribe Protocol For Wireless Sensor Networks."

[11] D. Soni and A. Makwana, "A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT) MP-Index View project Analysis and Survey on String Matching Algorithms for Ontology Matching View project A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)," 2017. [Online]. Available: https://www.researchgate.net/publication/316018571

[12] H. Karimipour, A. Dehghantanha, R. M. Parizi, K. K. R. Choo, and H. Leung, "A Deep and Scalable Unsupervised Machine Learning System for Cyber-Attack Detection in Large-Scale

Smart Grids," *IEEE Access*, vol. 7, pp. 80778–80788, 2019, doi: 10.1109/ACCESS.2019.2920326.

[13] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning", doi: 10.1007/s12525-021-00475-2/Published.

[14] E. Horvitz and D. Mulligan, "Data, privacy, and the greater good," *Science (1979)*, vol. 349, no. 6245, pp. 253–255, Jul. 2015, doi: 10.1126/science.aac4520.

[15] M. Mingjianntang, "Advanced Sciences and Technologies for Security Applications Deep Learning Applications for Cyber Security." [Online]. Available: http://www.springer.com/series/5540

[16] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences (Switzerland)*, vol. 9, no. 20. MDPI AG, Oct. 01, 2019. doi: 10.3390/app9204396.

[17] V. Kumar and O. Prakash Sangwan, "Signature Based Intrusion Detection System Using SNORT," 2012. [Online]. Available: https://www.researchgate.net/publication/274952404

[18] N. V Sharma, Kavita, G. Aggarwal, and S. Sharma, "Performance Study of Snort and Suricata for Intrusion Detection System," *IOP Conf Ser Mater Sci Eng*, vol. 1099, no. 1, p. 012009, Mar. 2021, doi: 10.1088/1757-899x/1099/1/012009.

[19] H. Kwon, J. E. Fischer, M. Flintham, and J. Colley, "The Connected Shower," *Proc ACM Interact Mob Wearable Ubiquitous Technol*, vol. 2, no. 4, pp. 1–22, Dec. 2018, doi: 10.1145/3287054.

[20] SCAD Institute of Technology, IEEE Electron Devices Society, and Institute of Electrical and Electronics Engineers, *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC 2017) : 10-11, February 2017*.

[21] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Cybersecurity Threats, Countermeasures and Mitigation Techniques on the IoT: Future Research Directions," *Electronics (Switzerland)*, vol. 11, no. 20. MDPI, Oct. 01, 2022. doi: 10.3390/electronics11203330.

[22] S. V, V. A, and S. Pattar, "MQTT based Secure Transport Layer Communication for Mutual Authentication in IoT Network," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 60–66, Jun. 2022, doi: 10.1016/j.gltp.2022.04.015.

[23] P. Pokharel, R. Pokhrel, and S. Sigdel, "Intrusion detection system based on hybrid classifier and user profile enhancement techniques," in *2020 International Workshop on Big Data and Information Security, IWBIS 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 137–143. doi: 10.1109/IWBIS50925.2020.9255578.

[24] A. Khanna and S. Kaur, "Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture," *Computers and Electronics in Agriculture*, vol. 157. Elsevier B.V., pp. 218–231, Feb. 01, 2019. doi: 10.1016/j.compag.2018.12.039.

[25] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review," *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0268-2.

[26] Z. Yan, P. Zhang, and A. v. Vasilakos, "A survey on trust management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120–134, 2014, doi: 10.1016/j.jnca.2014.01.014.

[27] S. Patel, C. Salazar, K. K. Patel, S. M. Patel, and P. G. Scholar, “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges,” *International Journal of Engineering Science and Computing*, 2016, doi: 10.4010/2016.1482.

[28] A. el Hakim, “Internet of Things (IoT) System Architecture and Technologies, White Paper. Internet of Things (IoT) System Architecture and Technologies”, doi: 10.13140/RG.2.2.17046.19521.

[29] M. Cabric, “Confidentiality, Integrity, and Availability,” in *Corporate Security Management*, Elsevier, 2015, pp. 185–200. doi: 10.1016/b978-0-12-802934-3.00011-1.

[30] K. Y. Chai and M. F. Zolkipli, “Review on Confidentiality, Integrity and Availability in Information Security,” *Journal of ICT In Education*, vol. 8, no. 2, pp. 34–42, Jul. 2021, doi: 10.37134/jictie.vol8.2.4.2021.

[31] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, “Internet of Things: Security vulnerabilities and challenges,” in *Proceedings - IEEE Symposium on Computers and Communications*, Institute of Electrical and Electronics Engineers Inc., Feb. 2016, pp. 180–187. doi: 10.1109/ISCC.2015.7405513.

[32] Q. Jing, A. v. Vasilakos, J. Wan, J. Lu, and D. Qiu, “Security of the Internet of Things: perspectives and challenges,” *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, Oct. 2014, doi: 10.1007/s11276-014-0761-7.

[33] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, “Internet of Things: Security vulnerabilities and challenges,” in *Proceedings - IEEE Symposium on Computers and Communications*, Institute of Electrical and Electronics Engineers Inc., Feb. 2016, pp. 180–187. doi: 10.1109/ISCC.2015.7405513.

[34] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, “A survey on security in internet of things with a focus on the impact of emerging technologies,” *Internet of Things (Netherlands)*, vol. 19. Elsevier B.V., Aug. 01, 2022. doi: 10.1016/j.iot.2022.100564.

[35] “11-networkattacks-notes”.

[36] M. Bettayeb, Q. Nasir, and M. A. Talib, “Firmware update attacks and security for IoT devices survey,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Mar. 2019. doi: 10.1145/3333165.3333169.

[37] N. Gadkar and A. Deshmukh, “Classifications on IoT Attacks.” [Online]. Available: https://iotanalytics.com/inter

[38] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, “Cybersecurity Threats, Countermeasures and Mitigation Techniques on the IoT: Future Research Directions,” *Electronics (Switzerland)*, vol. 11, no. 20. MDPI, Oct. 01, 2022. doi: 10.3390/electronics11203330.

[39] Y. Cherdantseva *et al.*, “A review of cyber security risk assessment methods for SCADA systems,” *Computers and Security*, vol. 56. Elsevier Ltd, pp. 1–27, Feb. 01, 2016. doi: 10.1016/j.cose.2015.09.009.

[40] P. A. H. Williams and A. J. Woodward, “Cybersecurity vulnerabilities in medical devices: A complex environment and multifaceted problem,” *Medical Devices: Evidence and Research*, vol. 8. Dove Medical Press Ltd, pp. 305–316, Jul. 20, 2015. doi: 10.2147/MDER.S50048.

[41] K. V. Sheelavathy and V. Udaya Rani, “Detection IoT attacks using Lasso regression algorithm with ensemble classifier,” *International Journal of Pervasive Computing and Communications*, 2022, doi: 10.1108/IJPCC-09-2022-0316.

[42] B. R. Mudhivarthi, P. Thakur, and G. Singh, “Aspects of Cyber Security in Autonomous and Connected Vehicles,” *Applied Sciences*, vol. 13, no. 5, p. 3014, Feb. 2023, doi: 10.3390/app13053014.

[43] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, Oct. 2015, doi: 10.1109/COMST.2015.2444095.

[44] M. Aqeel, F. Ali, M. W. Iqbal, T. A. Rana, M. Arif, and Md. R. Auwul, “A Review of Security and Privacy Concerns in the Internet of Things (IoT),” *J Sens*, vol. 2022, pp. 1–20, Sep. 2022, doi: 10.1155/2022/5724168.

[45] A. Attkan and V. Ranga, “Cyber-physical security for IoT networks: a comprehensive review on traditional, blockchain and artificial intelligence based key-security,” *Complex and Intelligent Systems*, vol. 8, no. 4, pp. 3559–3591, Aug. 2022, doi: 10.1007/s40747-022-00667-z.

[46] N. Mhaskar, M. Alabbad, and R. Khedri, “A Formal Approach to Network Segmentation,” *Comput Secur*, vol. 103, Apr. 2021, doi: 10.1016/j.cose.2020.102162.

[47] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, “A survey of intrusion detection in Internet of Things,” *Journal of Network and Computer Applications*, vol. 84. Academic Press, pp. 25–37, Apr. 15, 2017. doi: 10.1016/j.jnca.2017.02.009.

[48] S. Godala and R. P. V. Vaddella, “A study on intrusion detection system in wireless sensor networks,” *International Journal of Communication Networks and Information Security*, vol. 12, no. 1, pp. 127–141, 2020, doi: 10.17762/ijcnis.v12i1.4429.

[49] H. Mliki, A. Kaceam, and L. Chaari, “A Comprehensive Survey on Intrusion Detection based Machine Learning for IoT Networks,” *ICST Transactions on Security and Safety*, vol. 8, no. 29, p. 171246, Nov. 2021, doi: 10.4108/eai.6-10-2021.171246.

[50] A. Imanbayev *et al.*, “Research of Machine Learning Algorithms for the Development of Intrusion Detection Systems in 5G Mobile Networks and Beyond,” *Sensors*, vol. 22, no. 24, Dec. 2022, doi: 10.3390/s22249957.

[51] “mqtt-v3.1.1-errata01-os-complete Specification URIs,” 2015. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.doc

[52] V. Seoane, C. Garcia-Rubio, F. Almenares, and C. Campo, “Performance evaluation of CoAP and MQTT with security support for IoT environments,” *Computer Networks*, vol. 197, Oct. 2021, doi: 10.1016/j.comnet.2021.108338.

[53] “mqtt-v3.1.1-os Specification URIs,” 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc

[54] E. Atilgan, I. Ozcelik, and E. N. Yolacan, “MQTT Security at a Glance,” in *14th International Conference on Information Security and Cryptology, ISCTURKEY 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 138–142. doi: 10.1109/ISCTURKEY53027.2021.9654337.

[55] M. M. Raikar and S. M. Meena, “SSH brute force attack mitigation in Internet of Things (IoT) network : An edge device security measure,” in *ICSCCC 2021 - International Conference on Secure Cyber Computing and Communications*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 72–77. doi: 10.1109/ICSCCC51823.2021.9478131.

[56] S. Abaimov and M. Martellini, “Advanced Sciences and Technologies for Security Applications Machine Learning for Cyber Agents Attack and Defence.” [Online]. Available: https://link.springer.com/bookseries/5540

[57] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan, “A Comprehensive Survey on Machine Learning-Based Intrusion Detection Systems for Secure Communication in Internet of Things,” *Comput Intell Neurosci*, vol. 2023, pp. 1–24, Jan. 2023, doi: 10.1155/2023/8981988.

[58] M. W. B. Azlinah, M. Bee, and W. Yap, “Supervised and Unsupervised Learning for Data Science Unsupervised and Semi-Supervised Learning Series Editor: M. Emre Celebi.” [Online]. Available: http://www.springer.com/series/15892

[59] D. Mehta, “State-of-the-Art Reinforcement Learning Algorithms.” [Online]. Available: www.ijert.org

[60] E. Horvitz and D. Mulligan, “Data, privacy, and the greater good,” *Science (1979)*, vol. 349, no. 6245, pp. 253–255, Jul. 2015, doi: 10.1126/science.aac4520.

[61] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[62] R. P. Ram Kumar, S. Polepaka, S. F. Lazarus, and D. V. Krishna, “An insight on machine learning algorithms and its applications,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11 Special issue 2, pp. 432–436, Sep. 2019, doi: 10.35940/ijitee.K1069.09811S219.

[63] “Machine Learning Algorithms-A Review,” 2019, doi: 10.21275/ART20203995.

[64] H. A. Park, “An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain,” *J Korean Acad Nurs*, vol. 43, no. 2, pp. 154–164, 2013, doi: 10.4040/jkan.2013.43.2.154.

[65] H. H. Patel and P. Prajapati, “Study and Analysis of Decision Tree Based Classification Algorithms,” *International Journal of Computer Sciences and Engineering*, vol. 6, no. 10, pp. 74–78, Oct. 2018, doi: 10.26438/ijcse/v6i10.7478.

[66] K. Taunk, S. De, S. Verma, and A. Swetapadma, “A brief review of nearest neighbor algorithm for learning and classification,” in *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 1255–1260. doi: 10.1109/ICCS45141.2019.9065747.

[67] C. Tu, H. Liu, and B. Xu, “AdaBoost typical Algorithm and its application research,” in *MATEC Web of Conferences*, EDP Sciences, Dec. 2017. doi: 10.1051/matecconf/201713900222.

[68] C. Bentéjac and A. Csörg″ O B Gonzalo Martínez-Muñoz, “A Comparative Analysis of XGBoost.” [Online]. Available: https://www.researchgate.net/publication/337048557

[69] Y. Zhang, J. Liu, and W. Shen, “A Review of Ensemble Learning Algorithms Used in Remote Sensing Applications,” *Applied Sciences (Switzerland)*, vol. 12, no. 17. MDPI, Sep. 01, 2022. doi: 10.3390/app12178654.

[70] “Ensemble Methods Foundations and Algorithms.”

[71] D. R. C. and J. R. S. Adele Cutler, *Ensemble Machine Learning*. Springer US, 2012. doi: 10.1007/978-1-4419-9326-7.

[72] S. A. N. Alexandropoulos, C. K. Aridas, S. B. Kotsiantis, and M. N. Vrahatis, “Stacking Strong Ensembles of Classifiers,” in *IFIP Advances in Information and Communication Technology*, Springer New York LLC, 2019, pp. 545–556. doi: 10.1007/978-3-030-19823-7_46.

[73] R. Sibindi, R. W. Mwangi, and A. G. Waititu, “A boosting ensemble learning based hybrid light gradient boosting machine and extreme gradient boosting model for predicting house prices,” *Engineering Reports*, Apr. 2022, doi: 10.1002/eng2.12599.

[74] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.

[75] D. Miorandi, S. Sicari, F. de Pellegrini, and I. Chlamtac, “Internet of things: Vision, applications and research challenges,” *Ad Hoc Networks*, vol. 10, no. 7. Elsevier B.V., pp. 1497–1516, 2012. doi: 10.1016/j.adhoc.2012.02.016.

[76] M. Roopak, G. Y. Tian, and J. Chambers, “An Intrusion Detection System Against DDoS Attacks in IoT Networks,” in *2020 10th Annual Computing and Communication Workshop and Conference, CCWC 2020*, Institute of Electrical and Electronics Engineers Inc., Jan. 2020, pp. 562–567. doi: 10.1109/CCWC47524.2020.9031206.

[77] D. Tsany Rahmantyo, B. Erfianto, and G. Bayu Satrya, “Deep Residual CNN for Preventing Botnet Attacks on the Internet of Things,” in *Proceedings - 2021 4th International Conference on Computer and Informatics Engineering: IT-Based Digital Industrial Innovation for the Welfare of Society, IC2IE 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 462–466. doi: 10.1109/IC2IE53219.2021.9649314.

[78] E. Ciklabakkal, A. Donmez, M. Erdemir, E. Suren, M. K. Yilmaz, and P. Angin, “ARTEMIS: An intrusion detection system for mqtt attacks in internet of things,” in *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, IEEE Computer Society, Oct. 2019, pp. 369–371. doi: 10.1109/SRDS47363.2019.00053.

[79] A. Alhowaide, I. Alsmadi, and J. Tang, “Ensemble Detection Model for IoT IDS,” *Internet of Things (Netherlands)*, vol. 16, Dec. 2021, doi: 10.1016/j.iot.2021.100435.

[80] S. Bharati and P. Podder, “Machine and Deep Learning for IoT Security and Privacy: Applications, Challenges, and Future Directions,” *Security and Communication Networks*, vol. 2022. Hindawi Limited, 2022. doi: 10.1155/2022/8951961.

[81] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, “Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset),” in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 73–84. doi: 10.1007/978-3-030-64758-2_6.

[82] M. Ahmad, Q. Riaz, M. Zeeshan, H. Tahir, S. A. Haider, and M. S. Khan, “Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set,” *EURASIP J Wirel Commun Netw*, vol. 2021, no. 1, Dec. 2021, doi: 10.1186/s13638-021-01893-8.

[83] F. Mosaiyebzadeh, L. G. Araujo Rodriguez, D. Macedo Batista, and R. Hirata, “A Network Intrusion Detection System using Deep Learning against MQTT Attacks in IoT,” in *Proceedings - 2021 IEEE Latin-American Conference on Communications, LATINCOM 2021*,

Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/LATINCOM53176.2021.9647850.

[84] L. Dhanabal and S. P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, 2015, doi: 10.17148/IJARCCE.2015.4696.

[85] A. B. M. Sultan, S. Mehmood, and H. Zahid, "Man in the Middle Attack Detection for MQTT based IoT devices using different Machine Learning Algorithms," in *2nd IEEE International Conference on Artificial Intelligence, ICAI 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 118–121. doi: 10.1109/ICAI55435.2022.9773590.

[86] A. Kumari and A. K. Mehta, "A Hybrid Intrusion Detection System Based on Decision Tree and Support Vector Machine," in *2020 IEEE 5th International Conference on Computing Communication and Automation, ICCCA 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 396–400. doi: 10.1109/ICCCA49541.2020.9250753.

[87] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study," *Computer Networks*, vol. 188, Apr. 2021, doi: 10.1016/j.comnet.2021.107840.

[88] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a Lightweight Intrusion Detection System for the Internet of Things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019, doi: 10.1109/ACCESS.2019.2907965.

[89] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine learning-based IoT-botnet attack detection with sequential architecture," *Sensors (Switzerland)*, vol. 20, no. 16, pp. 1–15, Aug. 2020, doi: 10.3390/s20164372.

[90] IEEE Communications Society. Indonesia Chapter., Universitas Telkom., and Institute of Electrical and Electronics Engineers, *Proceedings, the 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology : July 7-8, 2020, Bali, Indonesia.*

[91] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, and S. Sivamohan, "Anomaly-Based Intrusion Detection System Using Support Vector Machine," in *Advances in Intelligent Systems and Computing*, Springer, 2020, pp. 723–731. doi: 10.1007/978-981-15-0199-9_62.

[92] T. V, Vellore Institute of Technology, Institute of Electrical and Electronics Engineers. Madras Section, IEEE Communications Society., and Institute of Electrical and Electronics Engineers, *Conference proceedings, International Conference on Vision Towards Emerging Trends In Communication and Networking (ViTECoN 2019) : 30-31, March 2019, Vellore, Tamilnadu, India*.

[93] A. V. Kachavimath, S. V. Nazare, and S. S. Akki, "Distributed Denial of Service Attack Detection using Naïve Bayes and K-Nearest Neighbor for Network Forensics," in *2nd International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2020 - Conference Proceedings*, Institute of Electrical and Electronics Engineers Inc., Mar. 2020, pp. 711–717. doi: 10.1109/ICIMIA48430.2020.9074929.

[94] A. Verma and V. Ranga, "Machine Learning Based Intrusion Detection Systems for IoT Applications," *Wirel Pers Commun*, vol. 111, no. 4, pp. 2287–2310, Apr. 2020, doi: 10.1007/s11277-019-06986-8.

[95] H. Siddharthan, T. Deepa, and P. Chandhar, “SENMQTT-SET: An Intelligent Intrusion Detection in IoT-MQTT Networks Using Ensemble Multi Cascade Features,” *IEEE Access*, vol. 10, pp. 33095–33110, 2022, doi: 10.1109/ACCESS.2022.3161566.

[96] M. Latah and L. Toker, “An Efficient Flow-based Multi-level Hybrid Intrusion Detection System for Software-Defined Networks.”

[97] A. A. Abdulrahman and M. K. Ibrahem, “EVALUATION OF DDOS ATTACKS DETECTION IN A CICIDS2017 DATASET BASED ON CLASSIFICATION ALGORITHMS,” 2018. [Online]. Available: https://ijict.edu.iq

[98] Y. Wang, Y. Shen, and G. Zhang, “Research on Intrusion Detection Model using ensemble learning methods,” in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, IEEE Computer Society, Jul. 2016, pp. 422–425. doi: 10.1109/ICSESS.2016.7883100.

[99] O. Ruhrpott Meetup, “Analysing Networks with NMAP,” 2019.

[100] M. M. Mariočagalj, “The Secure Shell (SSH) Protocol.” [Online]. Available: www.openssh.com

[101] M. M. Raikar and S. M. Meena, “SSH brute force attack mitigation in Internet of Things (IoT) network : An edge device security measure,” in *ICSCCC 2021 - International Conference on Secure Cyber Computing and Communications*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 72–77. doi: 10.1109/ICSCCC51823.2021.9478131.

[102] M. M. Raikar and S. M. Meena, “SSH brute force attack mitigation in Internet of Things (IoT) network : An edge device security measure,” in *ICSCCC 2021 - International Conference on Secure Cyber Computing and Communications*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 72–77. doi: 10.1109/ICSCCC51823.2021.9478131.

[103] Y. Ahmad, “Preventing Vulnerabilities and Mitigating Attacks on the MQTT Protocol.”

[104] Y. Liang and N. Vankayalapati, “Machine Learning and Deep Learning Methods for Better Anomaly Detection in IoT-23 Dataset Cybersecurity.”

[105] “Global Information Assurance Certification Paper A Beginners Guide to tcpdump,” 2004. [Online]. Available: http://www.giac.org/registration/gsec

[106] D. Abeles and M. Zioni, “MQTT-PWN Documentation Release 1.0,” 2020.

[107] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, “Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset),” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.15340

[108] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, “Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset),” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.15340

[109] C. V. Gonzalez Zelaya, “Towards explaining the effects of data preprocessing on machine learning,” in *Proceedings - International Conference on Data Engineering*, IEEE Computer Society, Apr. 2019, pp. 2086–2090. doi: 10.1109/ICDE.2019.00245.

[110] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” 2002.

[111] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[112] Z. Karimi, “Confusion Matrix Some of the authors of this publication are also working on these related projects: Data Cleaning Process View project.” [Online]. Available: https://www.researchgate.net/publication/355096788

[113] S. Yang and G. Berdine, “The receiver operating characteristic (ROC) curve,” *The Southwest Respiratory and Critical Care Chronicles*, vol. 5, no. 19, p. 34, May 2017, doi: 10.12746/swrccc.v5i19.391.

[114] S. Wu, P. A. Flach, S. Wu SHAOMINWU, and P. Flach PETERFLACH, “A scored AUC Metric for Classifier Evaluation and Selection Managing risk and uncertainty in warranty servicing policy View project Reliability of Building Services Systems View project A scored AUC Metric for Classifier Evaluation and Selection,” 2005. [Online]. Available: https://www.researchgate.net/publication/245053518