Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science in Telecommunication Engineering

Master Thesis

Machine leaning-based DoS attacks detection for MQTT sensor networks

Candidate:
Ali Ghannadrad, 918371

Supervisor:
Prof. Alessandro Enrico Cesare Redondi

Co-Supervisors:
Dr. Edoardo Longo, Dr. Eugenio Moro

Academic year 2020-2021

Sommario

Negli ultimi anni, la rapida diffusione dell'Internet of Things (IoT) ha contribuito a generare un'enorme quantità di dati. I dispositivi IoT sono utilizzati nelle cosiddette case intelligenti, i quali permettono di condividere i dati sull'utilizzo dei consumatori e automatizzare diverse attività a seconda delle necessità dell'utente. Di contro, qualsiasi nuova tecnologia (superficie d'attacco) può attirare l'attenzione di attori malevoli (hackers) che cercano di sfruttarla e abusarne attraverso l'uso di tecniche come il Denial of Service (DoS). Lo scopo principale di questo lavoro di tesi è rilevare e classificare gli attacchi DoS per le reti di sensori MQTT. In questo progetto, simuliamo uno scenario di casa intelligente per raccogliere dati affidabili. Raccogliendo e segregando i dati legittimi da quelli dannosi, proponiamo un set di dati realistico basato su MQTT che contiene flusso di traffico legittimo e dannoso. Presentiamo soluzioni di machine learning online e offline per rilevare e classificare gli attacchi DoS. I classificatori online proposti sono in grado di rilevare attacchi in diverse durate del flusso per osservare il compromesso tra l'accuratezza dei classificatori e la durata del flusso. Mostriamo che siamo in grado di rilevare con successo il traffico dannoso e legittimo come classificazione binaria; inoltre, nel caso di classificazione multivalore, riusciamo a classificare le categorie dell'attacco DoS. Infine, presentiamo il miglior classificatore per ciascuna durata del flusso, oltre a discutere il compromesso tra le durate del flusso e l'accuratezza dei classificatori.

Abstract

In recent years, the rapid deployment of the Internet of Things (IoT) applications has generated a massive amount of data. IoT devices are utilized in smart homes, which are frequently working together, sharing consumer usage data and automating activities depending on the homeowners' preferences. Predictably, any new technology that is extensively used by people can attract the attention of cyber-attackers who seek to exploit and abuse it through the use of sophisticated techniques such as Denial of Service (DoS). The main aim of this thesis work is to detect and classify DoS attacks for MQTT sensor networks. In this project, we simulate a smart home scenario to gather reliable data. By collecting the legitimate and malicious data, we propose a realistic MQTT-based dataset that contains legitimate and malicious flow-level traffic. We present online and offline machine learning solutions to detect and classify DoS attacks. Our proposed online classifiers are able to detect attacks in different flow durations to notice the trade-off between classifiers' accuracy and flow durations. As part of the evaluation of this work, we are able to successfully detect malicious and legitimate traffic as a binary classification; furthermore, our classifiers are capable to detect and classify categories of the DoS attack in a multi-value classification. Finally, we present the best classifier for each flow duration, and the trade-off between flow durations and classifiers' accuracy are discussed.

Contents

List of Figures

List of Tables

List of Abbreviations

- **IoT:** Internet of Things
- **MQTT:** Message Queuing Telemetry Transport
- **DoS:** Denial of Service
- **TCP:** Transmission Control Protocol
- **IP:** Internet Protocol
- **OASIS:** Organization for the Advancement of Structured Information Standards
- **QoS:** Quality of Service
- **PCAP:** Packet CAPture
- **CSV:** Comma-Separated Values
- **ANOVA:** Analysis of Variance
- **ML:** Machine Learning
- **RF:** Random Forest
- **KNN:** K-nearest neighbour
- **SVM:** Support Vector Machine
- **TP:** True Positive
- **TN:** True Negative
- **FP:** False Positive
- **FN:** False Negative
- **CT-Markov chain:** Continuous-time Markov chain
- **LED:** light-emitting diode
- **VM:** Virtual machine

Acknowledgment

I would like acknowledge and give my warmest thanks to my supervisor Prof. Alessandro Enrico Cesare Redondi who made this work possible. It is a great honor to work under his supervision.
I would like to express my deepest thanks and gratitude to Dr.Edoardo Longo and Dr.Eugenio Moro for their encouragement, creative and comprehensive advices until this work came to existence.
I would like to extend my appreciation to all my professors at Politecnico di Milano for everything they taught me, these years have been highly formative and have made me grow professionally and personally.
A very special thanks to my family, who have been supporting me from the distance, and my beloved Paria for her patience and always being my side.
Last but not the least, I would like to thank my friends and colleagues, especially, Ebrahim Azizi, Alireza Raeisi Ardali and Antonio Frighetto.

Ali

Chapter 1

Introduction

1.1 Overview

Nowadays, our daily life and communication systems are joined with the Internet of Things (IoT) which is a consolidated technology. In IoT applications, objects elaborate, process, and communicate with other IoT devices or more complicated systems. Essentially, User's demands determine differences in the Internet of Things applications. For instance, IoT applications that monitor exercises through IoT devices, checking the perimeter of residential and industrial environments, or sensors to calculate the temperature and humidity of an area. The fundamental aim of IoT in our lives is to enhance the quality of human life and offer new opportunities for a lot of applications [1]. As the use of these devices is increasing, there are several concerns over their security systems. The reason IoT devices are seen as potential security threats is that they are connected to the internet and can be remotely controlled, they consume sensitive data, and they are provided with lower security measurements so that they remain economical and user-friendly.
According to predictions, by 2025 the total number of IoT devices will be approximately 75 billion, as shown in Figure 1.1 [2].

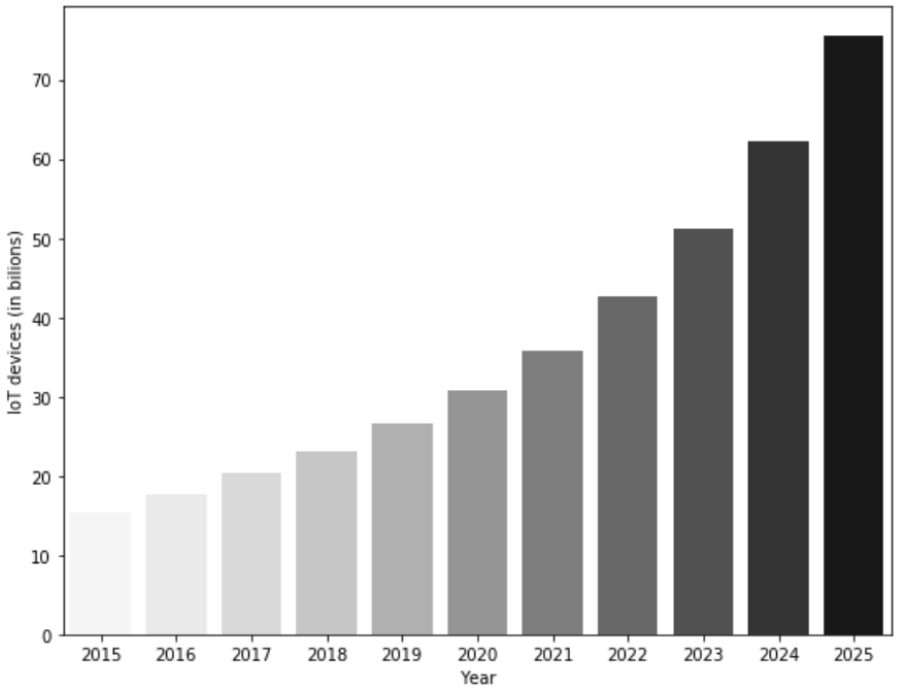

Figure 1.1: IoT devices installed base worldwide from 2015 to 2025

IoT devices need to operate continuously and reliably for a long time; thus, they need to be power-efficient. MQTT is arguably the most popular publish/subscribe protocol specifically designed for IoT devices. An MQTT broker is a server that receives all the clients' messages and then routes them to the proper destination clients. In some cases, the MQTT broker is not properly secure against Denial of Service (DoS) attacks to remain simple and power-efficient; therefore, it compromises the valid IoT sensors. In the next section, the main goal of this thesis work is explained.

1.2 Thesis Objective

The aim of this thesis is to detect and classify malicious attacks with machine learning models. There is an enormous 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 [3], however, fewer studies have been conducted regarding machine-learning-based DoS attack detection for MQTT sensor networks. To collect the data on which the machine learning analysis will be based, we have simulated a realistic smart home scenario. In this project, several MQTT-based IoT sensors with dif-

ferent traffic behaviors were developed to generate legitimate traffic, and we simulated three different DoS-based attackers to publish malicious data. Our simulated sensors communicate through the MQTT broker. In our scenario, attackers tried to saturate MQTT broker resources to exhaust the server. After developing the realistic sensors, we launched the simulated IoT devices and attackers to generate the data. In meantime, we sniffed the network to gather flow-level data.
TCP flows contain TCP packets with the same source and destination IP and port addresses which are captured during a limited time interval.
After capturing the flow-based data, we generated our Full-Featured realistic dataset. The Full-Featured dataset involves legitimates and anomalies flow-level traffic. We analysed statistically the Flow-level features to extract 10 best features. Then, we proposed the 10-Best Features dataset. After generating the datasets, we used online and offline machine learning techniques to detect and classify flows belonging to malicious attacks. In our scenario, online classifiers can detect a DoS attack in different flow durations, however, the offline classifiers are able to detect an attack after learning the entire training dataset. Our proposed classifiers are:

- **Binary Classifiers**, we trained these classifiers, to detect DoS attacks traffic. In the case of online machine learning, these classifiers are able to detect the DoS attacks in different flow durations.
- **Multi-value Classifiers**, the goal of these classifiers is to classify different DoS attack categories and legitimate flows. Our proposed online multi-value classifiers can detect categories of DoS attack in different flow durations.

1.3 Thesis Structure

In this section, the content of each chapter is briefly discussed.

- **Chapter 2**: In second chapter, we review the investigation of the other researchers, which are related to this topic, and what takes apart our scenario from other research.
- **Chapter 3**: IoT sensors and attackers implementation are discussed in chapter 3. The technologies, tools that we used to simulate the sensors and attackers are fully described in this section.
- **Chapter 4**: In chapter 4, we explain the experiment details and data acquisition. The procedure of creating the dataset and feature selection are described in this chapter.

- **Chapter 5**: In chapter 5, We discuss machine learning models and performance evaluation metrics, plus we propose our online and offline machine learning solutions. The deployment of our solution on 5G network is explained in this chapter.
- **Chapter 6**: In chapter 6, the online and offline machine learning results are demonstrated and classifiers are compared.
- **Chapter 7**: In the last chapter, the conclusion and the procedure of this thesis project explain, and at the end, the possible future works are discussed.

Chapter 2

Related works

2.1 State of the Art

This chapter presents the investigations that have been done for this thesis work. Internet of Things often is used for the correct functioning of critical infrastructures, and swiftly detecting ongoing attacks can give the opportunity to put countermeasures in place. Considering the importance of this topic, several kinds of research have been done in this area. In this chapter, a few relevant works on the problem statement under consideration are presented.

In [4], ten IoT sensors of various types, including temperature, motion detector sensor, humidity sensor, door locker, and etc. communicate across a network to simulate various scenarios such as home automation, critical infrastructure monitoring, or various industries. To test this strategy, legitimate traffics was mixed in with other malicious traffics targeting the MQTT network. The authors proposed a new MQTT dataset that covers legitimate and malicious traffics. The dataset is known as MQTTset, and it is publicly available in [4]. The packet-level features were extracted from raw PCAP and balanced and imbalanced datasets were created. They used and compared different machine learning methods such as neural network, random forests, naïve bayes, decision trees, gradient boost, and multilayer perceptron to validate the dataset. The obtained results indicate that, all the algorithms have obtained an accuracy level above 98%, while the F1 score is found to always be above 97%. Although it is quite difficult to classify attacks accurately, the accuracy and F1 score values are very high. This error is happened due to the imbalanced dataset. The number of legitimate traffic instances are much more than the sum of the instances of all malicious traffics, hence, they tried

to balance the dataset. After balancing, obtained results are clearly different in terms of accuracy and F1 score. The algorithms have an accuracy and F1 score between 87% and 91%, except the naïve bayes algorithm, where the results are around 64% in accuracy and 68% in F1 score. As a result, The obtained results illustrate that the unbalanced dataset has high accuracy and F1 score due to a large number of samples for legitimate traffic, which has an impact on the final results. The balanced dataset, on the other hand, reported low metrics but a correct data distribution in the confusion matrices.

Authers in [5], presented a DoS attack detection framework for MQTT attacks in IoT environments. Normal and attack traffic are captured by the attack detection testbed. Furthermore, they extracted MQTT control packet field size/length features and Count-based features. Then, two datasets which are known as Four-class and Seven-class were generated.The four-class dataset contained Normal, MQTT-DOS, MQTT-FUZZ and TCP-DOS attacks, however, the seven-class dataset involved four sub-classes of MQTT-DoS attacks presented in this work which were:MQTT-DOS-BF1, MQTT-DOS-BF2, MQTT-DOS-BF3 and MQTT-DOS-IAUTHS.The effectiveness of the proposed feature set was validated using three fundamentally different machine learning algorithms namely, AODE based on Naive Bayes, C4.5 based on Decision Tress and MLP based on ANN. To measure the detection accuracy of normal and attack classes, the performance of the classifiers were tested with count-based flow features and field length features. The achieved results shows, the AODE classifier obtained the highest classification accuracy in detecting the attack traffic for both four-class and seven-class datasets. The accuracy of the AODE classifier in Four-class dataset is more than 99% and in case of Seven-class is around 88%. The MQTT DoS attack modelling results indicate that the adversaries can cause large scale impact with just basic access to the MQTT broker by launching the invalid subscription flooding attack. However, the invalid authentication attacks were found to cause little impact with a single attack source machine, as these attacks depended on a large volume of attack packets. In addition, using a malformed CONNECT request, a high memory utilization on broker machines was witnessed, which could be exploited during a memory-exhaustion attacks. The DoS detection model demonstrated that the proposed MQTT features yielded high detection capabilities, especially when the control packet field size-length based features were selected. Hence, the flow-level features can be effectively used in detecting DoS attacks in IoT networks.

In [6], they demonstrated that packet-level machine learning DoS de-

tection can accurately predict legitimate and DoS attack traffic from IoT devices. Due to the low computing power of IoT devices, they used a limited features which is important for real-time classification. They evaluated five different machine learning classifiers which are Random Forest, Descision Tree, K-nearest neighbor, Support vector machine with linear kernel and Neural Network on a dataset that covers legitimate and DoS attack traffic collected from an experimental consumer IoT devices. All five algorithms had a test set accuracy higher than 0.99. These introductory results trigger additional research into machine learning anomaly detection to protect networks from vulnerable IoT devices.

Authors in [7] introduced Bot-IoT, a novel dataset that includes both typical IoT-related and other network traffic, as well as several types of botnet-related attack traffic. This dataset was created on a realistic testbed and labeled, with the label features indicating an attack flow, an attack category, and a subcategory for multiclass classification goals. Additional features were generated to improve the predictive abilities of the trained classifiers on this model. A subset of the original dataset, consisting of the 10-best features, was created by statistical analysis.They chose Support Vector Machine (SVM), a Recurring Neural Network (RNN) and a Long Short-Term Memory RNN (LSTM-RNN) as classfiers to detect the attacks. Then, four metrics were used to compare the effectiveness of the dataset, specifically Accuracy, Precision, Recall, Fall-out. The achieved result indicates that the highest accuracy from the SVM model that was trained on the full-featured dataset which is more than 99%, while the lowest accuracy from the SVM model of the 10-best feature dataset version which was around 88%

We presented a new balanced flow-based dataset which is labeled for three different categories of DoS attacks and legitimate flows. Unlike [6] and [4] which they extracted packet-level features, we provide a more flexible flow-level features. All the investigations [4], [5], [6] and [7] evaluated offline machine learning models on their dataset, however, in this thesis work, not only we proposed an offline machine learning solution but also a new online machine learning solution which can validate flow-level features in different flow durations were presented. Therefore, it allows us to tune the number of packets that need to be captured and it lets us trade-off the flow length with accuracy.
In the next section, we compared the similar datasets to our proposed dataset.

2.2 Comparison different datasets

In this section, we compare 6 different IoT datasets contains attacks and malicious traffic found in the literature with our proposed dataset.
In these years, different datasets have been used to train different machine learning models. Particularly, deep learning methods are used to detect cyber-attacks by training with the KDDCUP99 dataset, and for developing an intrusion detection system, random forest, decision tree, and gradient boost algorithms are trained with the KDDCUP99. Although KDDCUP99 is extensively used in cyber-security, it is not a suitable fit for IoT scenarios because it was not designed for this purpose and includes attacks on conventional ICT networks, which are difficult to adapt to IoT contexts [4].
In UNSW-NB15 dataset, HTTP, DNS traffics which are generated by IoT sensors are publicly available, however, MQTT traffics is not available [8], [4].
IoT-23 is used to compare the effectiveness of classification algorithms on detecting anomalies and legitimate traffic in the IoT context. This dataset involves DNS traffic focused on Mirai, Torii, IoT Trojan, Kenjiro, Okiru, Haji me, and other botnets. Focusing only on DNS traffic is the lack of this dataset [9].
N-BaIoT is a dataset focused on Wi-Fi communication that is used to detect and prevent botnet attacks in the IoT environment [10]. Despite the fact that the adopted datasets are particularly fascinating and diverse in this case, the authors did not make them publicly available. As a result, the possibilities of using them in study is quite limited [4]. Another interesting MQTT dataset for the detection approach based on machine learning is called TON-IoT [11]. TON-IoT is publicly available, however, does not cover all MQTT packets send and received during a connection, in other words, The authentication process for MQTT and TCP, which involves the sensor and broker, is missing from the dataset. As a result, the dataset is called incomplete because the authentication and disconnection phases are essential parts of IoT device communications [4].
Another exciting dataset that contains legitimate and malicious in the IoT context is MQTTset [4]. MQTTset is publicly available and the traffic is MQTT-based, however, as we discussed before, the dataset features are packet-level, not flow-level [4].
A review of the available datasets and their missing features, compared to our proposed dataset is shown in Table 2.1.
Unlike the reported dataset, in this thesis project, we proposed a new realistic MQTT-based dataset that involves flow-level features which is publically available in both PCAP and CSV formats.

Dataset	Citation	Flow-level Features	Realistic Scenario	Focused on IoT	Focused on MQTT	Compelete dataset
KDDCUP99	[12]	T	T	F	F	T
UNSW-NB15	[8]	T	T	F	F	T
IoT-23	[9]	T	T	T	F	T
N-BaIoT	[10]	T	T	T	F	T
TON-ToT	[11]	T	T	T	T	F
MQTTset	[4]	F	T	T	T	T
Proposed Dataset	-	T	T	T	T	T

Table 2.1: Comparison different available datasets

Chapter 3

IoT System implementation

In this chapter, the architecture of the simulated IoT sensors and attackers are explained. The fundamental step for training machine learning models is data. If we collect reliable data, the machine learning model prediction would be more accurate. Thus, it is extremely critical to generate realistic data. In our scenario, several IoT sensors with different traffic behaviors were developed to generate legitimate traffic, plus different types of DoS attackers were simulated to publish malicious traffic. In this chapter, by describing MQTT protocol and Mosquitto broker in section 3.1 and 3.2 respectively, we explain how IoT sensors and attackers communicate with each other. In section 3.3, development, traffic behavior, and legitimate IoT sensors specifications are fully explained. Then, in section 3.4, the simulated DoS-based attackers are completely described.

3.1 MQTT Protocol

MQTT is a lightweight messaging protocol that operates with a broker. MQTT is based on the publish-subscribe architecture and runs on top of Transmission Control Protocol / Internet Protocol(TCP/IP) for reliable message delivery. The MQTT protocol was created by Andy Stanford-Clark and Arlen Nipper in 1999. It is currently in the OASIS (Organization for the Advancement of Structured Information Standards) standard [13]. Due to the simple model and low bandwidth utilization, MQTT was initially created for remote site communication and is now widely utilized in IoT applications. In MQTT, messages are exchanged through the brokers using MQTT control packets [13]. The most significant control packet types are CONNECT, CONNACK, PUBLISH, PUBACK, SUBSCRIBE, SUBACK and DISCONNECT. The clients report their status to the broker using keep-alive messages

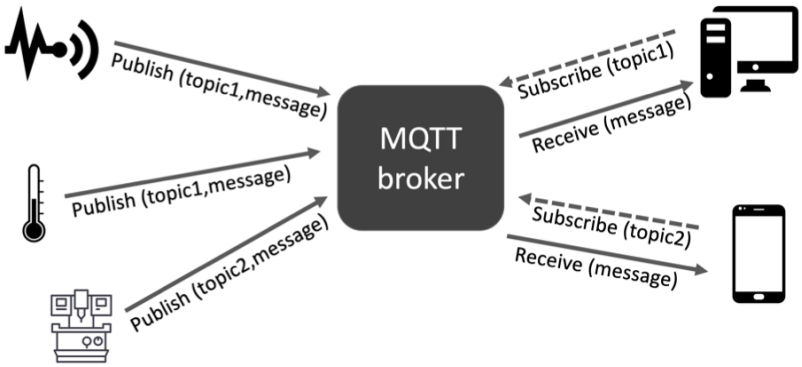

Figure 3.1: Publish-subscribe architecture

Source: [1]

by exchanging PINGREQ (request) and PINGRESP (response) control packets [5]. The MQTT protocol also supports three levels of Quality of Service (QoS) that could be used by publishers based on message delivery demands. QoS levels are shown in table 3.1 [13].
As demonstrated in Figure 3.1 the publishers send the generated data to the MQTT broker. The published data is specified by a topic that is subscribed by subscribers to receive notifications from the broker whenever new data is sent to the topic; hence, subscribers receive the data through the broker.

```
MQTT topic example:
apartment/room/smartbulb/state
```

QoS Types	Message Guarantee	behaviour
QoS 0	At most once	Messages sent to all subscribers once, no retries and no acknowledgment from receivers
QoS 1	At least once	Messages sent to all subscribers at least once and is acknowledged by receivers
QoS 2	Excatly once	Messages sent to all subscribers exactly once, no duplicates, extra acknowledgement messages to avoid duplicate messages and guaranteed message delivery

Table 3.1: MQTT QoS Types

3.2 Mosquitto broker

In this project, Mosquitto version 2.0.6 is used as the message broker. Eclipse Mosquitto is an open-source message broker that implements the MQTT protocol. Mosquitto is lightweight and is suitable for use on all devices from low-power single-board computers to full servers.
The Mosquitto project also includes a C library for creating MQTT clients, as well as the widely used command-line MQTT clients mosquitto pub and mosquitto sub [14].

3.3 IoT sensors

In this section, the general structure of the developed IoT sensors is presented. In this project, we simulated a Smart Home scenario. Smart homes are defined as homes that can meet the demands of residents with the help of the devices used, making their life easier and providing a safer, more comfortable, and cost-effective living environment [15]. IoT devices in smart homes must have several features such as power-efficient and low-cost. We were taken into account these features when IoT devices were selected for this scenario. In our scenario, as Figure 3.2 shows, a smart plug, smart thermostat, smart bulb, smart TV, smart vacuum cleaner, smart door lock, smart motion detector, and smart fire detector which are the most important appliances in the homes were developed.

The simulated IoT sensors publish the data to the MQTT topic through the Mosquitto broker and at the same time, attackers try to saturate the Mosquitto broker resources. In section 3.3.1, the development of the sensors is explained, then we study traffic behaviors and more details about the simulated IoT sensors in sections 3.3.2 and 3.3.3 respectively.

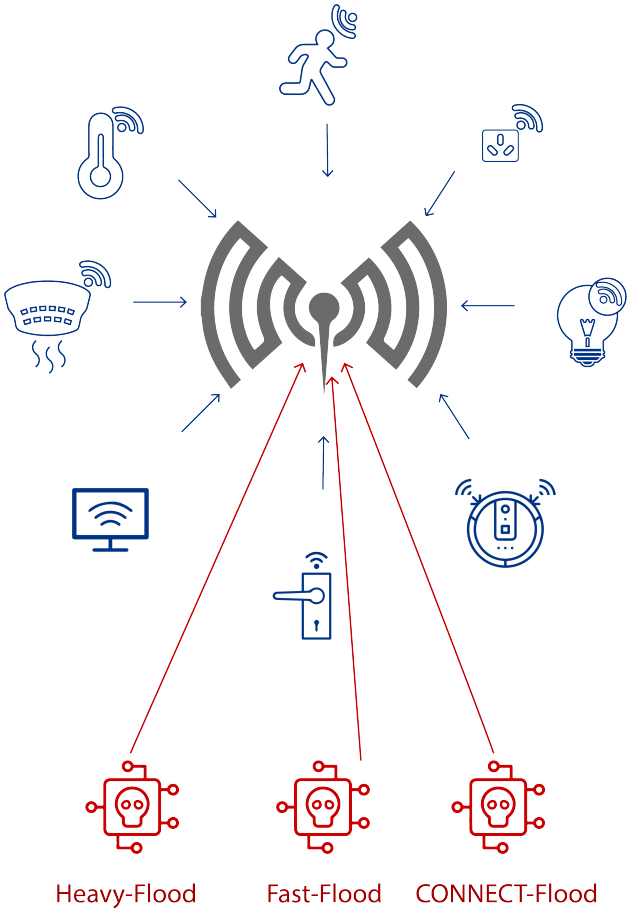

Figure 3.2: IoT sensors architecture

3.3.1 Development

In this study, IoT sensors and attackers were developed by Python 3.9 programming language. We used Paho-MQTT library version 1.5.1 which is available for the Python. This library includes a client class that allows devices to connect to an MQTT broker for publishing messages or subscribing to a particular MQTT topic. The Paho-MQTT client class has several methods. The main methods are: connect(), disconnect(), subscribe(), unsubscribe() and publish(). The first step of publishing or subscribing is a

connection to the broker which connect() method handles. Once you have a connection to the broker, you can start to publish messages with publish() method. subscribe() and unsubscribe() methods are used for subscribing and unsubscribing an MQTT topic.

3.3.2 Traffic behavior

Depending on the sensor's behavior, we considered two main traffic types, 'burst' and 'constant'.
For example, a temperature sensor publishes data of the measured temperature at predefined time intervals, e.g., every second so the traffic behavior is constant however, a motion sensor publishes the data to the broker only when a movement is detected; hence, the behavior is burst. Two-states sensors, such as Smart bulb which is ON or OFF, were modeled as continuous-time Markov chain.
A continuous-time Markov chain is a continuous stochastic process that changes state according to an exponential random variable for each state and then moves to another state.

3.3.3 IoT sensors specifications

In this section, the details of each simulated IoT sensor in terms of Traffic behavior, QoS, Payload size and publish rate, are clarified.
In our smart home scenario, due to the importance of the IoT devices' messages, we considered QoS as level 1 for all the sensors, which means messages are published to all subscribers at least once and are acknowledged by clients. The table 3.2 indicates the comparison between simulated IoT sensors.

3.3.3.1 Smart Plug

Smart plug is a smart sensor that it goes between the power socket and the appliance you want to plug in. These devices are considered 'smart' because they can control the appliances from an app on the phones from everywhere. The simplest smart plug feature is to turn the appliance ON or OFF, however, our simulated Smart plug is publishing the information about how much energy the appliance connected to it is consuming. The publishing rate of simulated Smart plug is 8000 publishes per hour. The application protocol is MQTT and sensor publishes the energy consumption of the connected appliances to the "apartment/room/plug/state" MQTT Topic. QoS is set to 1 due to the importance of the clients' acknowledgement. Due to the publishing the consumed energy regularly, the traffic behavior of

the Smart plug is considered as constant. The average payload size of this sensor is 123 Bytes.

3.3.3.2 Smart Thermostat

Another simulated sensor that publishes constantly is Smart Thermostat. The smart thermostat measures the temperature of the place which is located and publishes it to the "apartment/room/temp/state" topic continuously. This developed smart sensor publishes the temperature of the place every minute approximately. Smart Thermostat is MQTT-based and QoS is set to 1, so the messages is acknowledged by the user. The average payload size is considered 118 bytes which is the temperature of the environment.

3.3.3.3 Smart Bulb

A smart bulb is an LED light bulb that can be configured, scheduled, and controlled remotely. In our scenario, The traffic behavior of this MQTT-based smart bulb is considered as Burst due to the publishing periodically. The smart bulb publishes the state of itself which could be ON or OFF to "apartment/room/bulb/state" topic. The average payload size is 104 bytes. This sensor modeled as two states (ON-OFF) Continuous-time Markov chain. The exponential parameter for transitioning from state OFF to ON varies depending on the time of the day. This sensor is more active between 21:00 to 24:00 so we considered this time as peak-time. The exponential variable in peak-time is set to 1/500 for changing from state OFF to ON and for going from state ON to OFF is set to 1/480. In the other time of the day, the exponential variable for changing the state from OFF to ON is set to 1/120 and for transitioning from state ON to OFF is 1/100. The average publish rate for this sensor is 300 publishes per hour. By considering the first level of QoS, messages are acknowledged by clients.

3.3.3.4 Smart TV

A smart TV is a conventional television that is connected to the Internet. Smart TV allows users to stream music, videos, browsing the internet, etc. In our scenario, we simulated an MQTT-formed smart TV that updates the subscribers by sending in average 15 messages per hour to "apartment/-room/tv/state" MQTT topic. The average payload size is 112 bytes which is the state of the TV. Like Smart Bulb, Smart TV is modelled as two states CT-Markov chain. This sensor is more active from 20:00 to 24:00 that is a peak-time of this sensor. In peak-time, the exponential variable for changing from OFF to ON is 1/20 and from ON to OFF is 1/18, however, in other

times of the day, the exponential variable for changing from OFF to ON is 1/10 and from ON to OFF is considered as 1/8.

3.3.3.5 Smart Vacuum Cleaner

The smart vacuum cleaner is a smart IoT device that goes around the home for cleaning. Our developed Smart vacuum cleaner is MQTT-based and publishes in average 14 messages per hour to "apartment/room/vacuum/state" MQTT topic; hence, the traffic behavior is burst. The QoS is 1 and the average payload size which can be ON or OFF, is 113 bytes. Due to the burst traffic behavior, this sensor is modeled as CT-Markov chain. The peak-time of this sensor is set from 9:00 to 12:00. In peak-time, the exponential random parameter for swapping from state OFF to ON is 1/20 and from state ON to OFF is 1/15, however, in other times of the day, the exponential variable for changing from OFF to ON is 1/10 and from ON to OFF is 1/8.

3.3.3.6 Smart Lock Door

It happens for everybody to forget that they whether have locked the door or not. The smart Lock Door is a smart sensor that fits on the door. This sensor publishes the state of the door whether locked or not periodically to "apartment/room/lock/state" MQTT topic. The traffic behavior is burst; SO, we modeled it as two states (Locked - Not Locked) CT-Markov chain. The publishing rate in average is 9 publishes per hour. The peak-time of this sensors is set from 7:00 to 9:00. In peak-time, the exponential random parameter for switching from state OFF to ON is 1/20 and from state ON to OFF is considered as 1/15, on the other hand, in other times of the day, the exponential variable for changing from OFF to ON is 1/10 and from ON to OFF is 1/8. The QoS is 1, so messages sent to all subscribers at least once and are acknowledged by receivers. The average payload size considered 113 bytes.

3.3.3.7 Smart Motion Detector

Smart Motion Detection applies an advanced algorithm to distinguish between human and vehicular shapes in a scene and only publishes alarms when a person or vehicle is identified. Due to the burst traffic behavior, The publishes rate of the simulated motion detector is just 1 publish per hour which is "Motion detected" to "apartment/room/motion-detector/state" MQTT topic. The application protocol is MQTT and the payload size is 122 Bytes.

3.3.3.8 Smart Fire Detector

The smart fire detector publishes "Fire detected" to the "apartment/-room/fire/state" MQTT topic when it detects the smoke. The users who subscribe to that topic notice the alarm. The traffic behavior is burst and it publishes 1 message per hour and the average payload size is 116 bytes.

Sensor	**Traffic behavior**	**Average Rate**	**Average Payload size**	**Protocol**	**QoS**
Smart Plug	Constant	8000 publishes/hour	123 Bytes	MQTT	1
Smart Thermostat	Constant	400 publishes/hour	118 Bytes	MQTT	1
Smart Bulb	Burst	250 publishes/hour	103 Bytes	MQTT	1
Smart TV	Burst	14 publishes/hour	112 Bytes	MQTT	1
Smart Vaccum Cleaner	Burst	13 publishes/hour	113 Bytes	MQTT	1
Smart Lock Door	Burst	13 publishes/hour	113 Bytes	MQTT	1
Smart Motion Detector	Burst	1 publish/hour	122 Bytes	MQTT	1
Smart Fire Detector	Burst	1 publish/hour	116 Bytes	MQTT	1

Table 3.2: IoT Sensors Comparison

3.4 Attackers

A broker server's main function is to route messages between publishers and subscribers. In MQTT scenarios which messages are exchanged via the message broker, DoS attacks can interrupt the broker functions. Due to the finite resource of the brokers, saturating the broker's resource can lead to the server crash or message dropped. The current maximum payload size for MQTT is 256 MB; thus broker will require more resources if they receive messages with high payloads. An attacker could take advantage of this and saturate client and broker resources, causing service to be denied [13].
In addition to payload size attacks, the attackers can also use the QoS levels provided by MQTT protocol to cause DoS. In comparison to QoS levels 1 and 0, which are described in Table 3.1, messages published with QoS level 2 demand higher broker resources. The broker also holds messages which sent with QoS level 1 and 2 until messages are delivered to the subscribed users. With QoS level 2, attackers can send a huge quantity of messages and seize the broker's resources [13]. The attackers details are demonstrated in Table 3.3. In sections 3.4.1, 3.4.2 and 3.4.3, Connect-Flood, Heavy-Flood and Fast-Flood attackers are explained respectively.

Sensor	Protocol	QoS	Average Payload size	Average Rate	Count
Connect-Flood attack	MQTT	-	-	5000 publishes per hour	1024
Heavy-Flood attack	MQTT	2	250 MB	900000 publish per hour	1
Fast-Flood attack	MQTT	2	4 B	900000 publish per hour	1

Table 3.3: Attackers Specifications

3.4.1 Connect-Flood Attack

The Attacker starts to flood the broker with CONNECT packets.
The CONNECT packet is used by the MQTT user to launch an MQTT session with the broker and includes Broker IP address, the network port of the broker host which connects to, and Keepalive. If there are no additional messages being exchanged, then the client will send a PINGREQ and expect to receive a PINGRESP from the broker for a keepalive period. This message exchange confirms that the connection is open and working excellently.

Listing 3.1: Connect() Function

```
client.connect("172.20.10.2",port=1883,keepalive=1000)
```

When the broker server gets a CONNECT packet, it examines the user identity and other optional parameters to determine whether or not to allow the client to connect.
Attackers publish multiple CONNECT packets with different client identifiers which can saturate server resources [13]. We developed 1024 Connects flood attackers that publish CONNECT packets to exhaust the mosquitto broker. The rate of the publishes is 5000 per hour which is very high concerning the normal legitimate device. By setting Keepalive period to 1000 seconds, the attacker can at least occupy each connection for 1000 seconds concerning the normal Keepalive which is 60 seconds. By increasing the keepalive value, the exchange of PINGREQ-PINGRESP control messages increases; thus, the broker remains occupied.

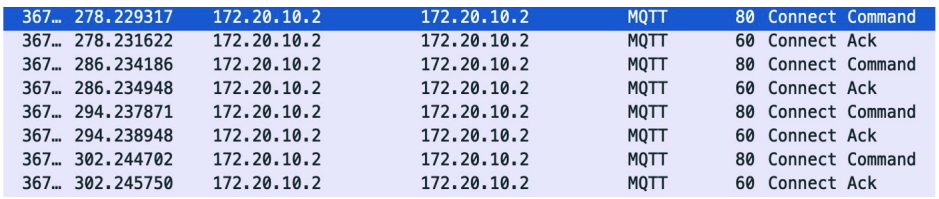

Figure 3.3: Connect-flood attack

3.4.2 Fast-Flood Attack

The Fast-Flood attacker publishes multiple packets with a high rate to seize the broker's resources. Publish rate is defined as Heavy-Flood attacker which is 900000 per hour, however, the payload size is not as large as Heavy-Flood payload. The Fast-Flood attacker's QoS is defined to 2, therefore, all the messages are guaranteed to be received by the broker and eventually subscribed clients, as a result, higher broker resources are required.

3.4.3 Heavy-Flood Attack

The malicious user sending enormous payload messages to saturate broker and subscriber resources.
The payload up to 256 MB could be attached in the MQTT packets; therefore, attackers can publish multiple messages with a heavy payload size to use the resources of both broker and subscribers who are listening to the topic. Attackers can apply higher levels of QoS with heavy payload messages to consume broker resources. The broker is needed to save the information until they are given to all the subscribed clients due to the highest level of QoS.
We defined the Heavy-Flood attacker to publish multiple large payload messages as fast as possible. The payload size is 250 MB and the publish rate is set to the 900000 per hour. The highest QoS guaranteed that all the messages will be delivered to all subscribed. As a result, with a high rate, massive payload and highest level of QoS, this attacker can seize the resources of the broker.

Chapter 4

Simulation and Dataset generation

As we dicussed in chapter 3, It is crucial to create a reliable and realistic dataset to train the machine learning classifiers excellently. In this chapter, The simulation of the IoT network and the procedure of generating datasets are fully presented. In section 4.1, we explain the tools that we used in this experiment. Then, in section 4.2 the procedure of the simulation and the data acquisition is clearly described. After that, the Full-Featured dataset and cleaning the data are presented in section 4.3. In section 4.4, we explain the feature selection method and 10-Best-Features dataset is proposed.

4.1 Tools

In this project, Wireshark was used for sniffing the network and collecting the data. To extract TCP flows' features, we used the Argus tool. In this section, these tools are defined.

4.1.1 Wireshark

Wireshark is the most popular and commonly used packet analyzer, and sniffer in the world [16]. It captures network traffic on the local network and saves that for further investigations. Wireshark captures packet-level data of Ethernet, Bluetooth, Wireless (IEEE.802.11), and some other technologies [16]. A packet is a single message from any network protocol (i.e., TCP, DNS, etc.) [17].

4.1.2 Argus

Argus is a real-time flow monitor that is developed to make comprehensive data network traffic analysis [18]. Argus is the first network flow system, created by Carter Bullard at Georgia Tech in the early 1980s and adopted by Carnegie Mellon's Software Engineering Institute for cyber security in the late 1980s. The network flow technology has become a crucial aspect of modern cyber security [19].
A TCP flow is the end to end connection, and shows how the data flow through the network. TCP flows cover all the packets with the same source and destination IP and port addresses which are captured during a limited time interval. Argus analyzes packet data and creates network flow information. If you have packets and want to know what is happening under the hood, argus() is a wonderful approach to look at features of the data that packet analyzers cannot easily provide. The main benefits of using the Argus tool are to understand that how many hosts are communicating, who is communicating with whom, how frequently is one address sending the traffic [18].

4.2 Simulation and Data acquisition

After the developing phase of the IoT sensors and attackers, we launched them to start simulating the Smart home scenario to gather legitimate and malicious data. We used different devices for each actor of the simulation. Mosquitto broker version 2.0.6 was installed on the MacBook Pro 2017, 2.3 GHz Dual-Core Intel Core i5, 8 GB of RAM which running macOS Catalina. All the IoT sensors were simulated on a Virtual Machine with 3 GB of RAM running Ubunto 19.10 which was installed on the MacBook Pro 2017 (same machine as the broker). In terms of the attackers, Fast-Flood and Connect-Flood were simulated on Apple iPad 2018 Quad-core 2.34 GHz CPU, 2GB of RAM. According to the nature of the Heavy-Flood attacker, more RAM was required to publish massive payload continuously; thus, we implemented Heavy-Flood on Asus VivoBook S15 2019, Core i7-8565U CPU, 16 GB of RAM, which running windows 10. Wireshark application was installed on the MacBook Pro 2017 to sniffed the network. For connecting these devices, we considered a Local Area Network, All these devices were connected wirelessly to the same network. The IP addresses are shown in Table 4.1. After implementing the simulated sensors and broker, we started the experiment by launching the Mosquitto broker and Wireshark to do the eavesdropping. After 1 minute, the IoT sensors started to publish legitimate traffic to the

MQTT topics through the Mosquitto broker. After 10 minutes of launching the IoT sensors, Heavy-Flood and Fast-Flood were flooding the messages through the Mosquitto broker simultaneously for approximately 1 hour and 30 minutes. Then, we disconnected the Heavy-Flood and Fast-Flood; furthermore, Connect-Flood was launched for almost 1 hour to flood CONNECT packets to the broker. As we explained in chapter 3, 1024 Connect-Flood attackers were simulated; hence, due to the high number of Connect-Flood attackers in comparison to Heavy-Flood and Fast-Flood attackers, we launched it much less time as opposed to other types of attackers. For collecting more data, we activated Heavy-Flood and Fast-Flood again for almost 1 hour. After 3 hours, the broker, and all the sensors were deactivated. Table 4.1 shows our testbed details.

Device	**Type**	**IP address**	**Time of Experiment**
MacBook Pro	Mosquitto broker	172.20.10.2	3 hours
Apple iPad	Fast-Flood attack	172.20.10.3	2 hours and 30 minutes
Apple iPad	Connect-Flood attack	172.20.10.3	1 hour
Asus VivoBook	Heavy-Flood attack	172.20.10.4	2 hours and 30 minutes
Ubunto virtual machine	Legitimate sensors	172.20.10.5	3 hours

Table 4.1: Testbed details

After terminating the simulation, the PCAP file was exported from Wireshark. The PCAP file contained all the network's packets which were legitimate traffic and malicious traffic. Since our study was flow-level, not packet-level, the Argus tool was used to generate the relevant network flows. The PCAP file was imported to the Argus tool to convert to Flows. Since TCP flows are captured during a limited time interval, we defined 5 different flow durations which are 5, 10, 25, 30, and 60 seconds. After collecting the network's flow for each duration, we extracted flow-level features from the Argus file and converted them to the CSV file for further investigations.
Figure 4.1 presents the procedure of extracting TCP flows. The final features provided by Argus during the network flow extraction process are listed in Table 4.2 and Table 4.3.
In the next chapter, our proposed datasets are discussed.

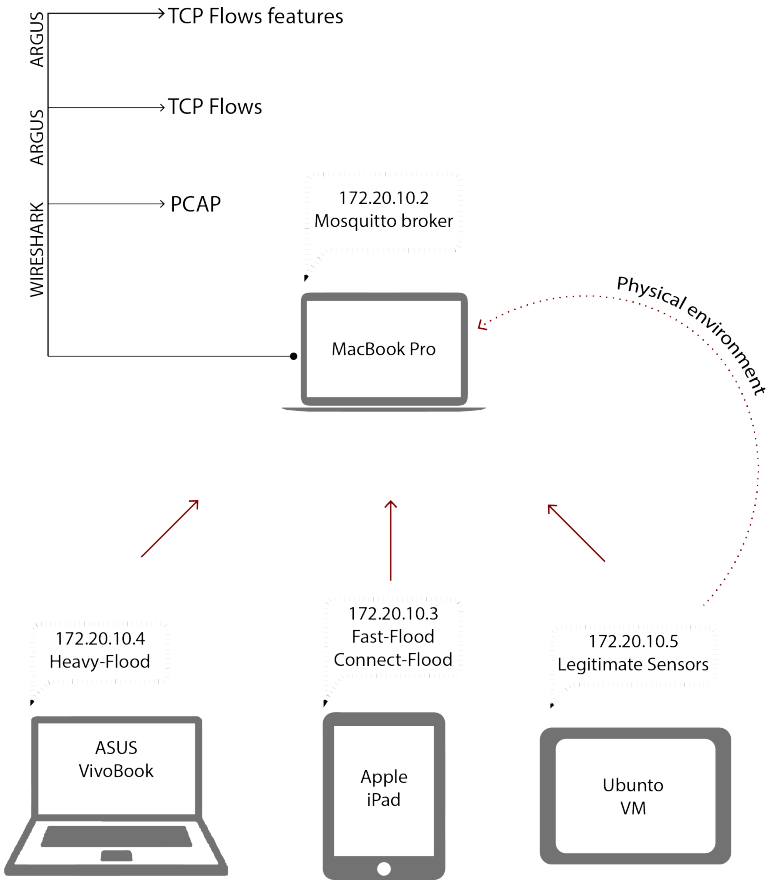

Figure 4.1: Feature extraction process

4.3 Full-Featured Dataset

After collecting all the Flow-level features, the Full-Featured dataset was generated. The full-Featured dataset has 44 flow aggregated features. The size of the CSV format is 10.6 MB and there are 34036 instances in the dataset. Due to the importance of having a reliable and realistic dataset, we cleaned and labeled the Full-Featured dataset. In the following sections, we discuss how we obtained cleaned Full-Featured dataset. In Table 4.2 and Table 4.3, all the features are demonstrated.

Number	Feature	Description
1	StartTime	Record start time
2	LastTime	Record last time
3	Flgs	flow state flags seen in transaction
4	Dur	record total duration
5	RunTime	total active flow run time
6	IdleTime	time since the last packet activity.
7	Mean	average duration of aggregated records
8	StdDev	standard deviation of aggregated duration times
9	Sum	total accumulated durations of aggregated records
10	Min	minimum duration of aggregated records
11	Max	maximum duration of aggregated records
12	SrcAddr	source IP addr
13	DstAddr	destination IP addr
14	Proto	transaction protocol
15	Sport	source port number
16	Dport	destination port number
17	TotPkts	total transaction packet count
18	SrcPkts	src -> dst packet count
19	DstPkts	dst -> src packet count
20	TotBytes	total transaction bytes
21	TotAppBytes	total application bytes
22	SrcBytes	src -> dst transaction bytes
23	DstBytes	dst -> src transaction bytes
24	Loss	pkts retransmitted or dropped

Table 4.2: Generated flow features

Number	Feature	Description
25	SrcLoss	source pkts retransmitted or dropped
26	DstLoss	destination pkts retransmitted or dropped
27	pLoss	percent pkts retransmitted or dropped
28	Retrans	pkts retransmitted
29	SrcRetra	source pkts retransmitted
30	DstRetra	destination pkts retransmitted
31	pRetran	percent pkts retransmitted
32	Rate	pkts per second
33	SrcRate	source pkts per second
34	DstRate	destination pkts per second
35	Dir	direction of transaction
36	SintPkt	source interpacket arrival time (mSec)
37	SintDist	source interpacket arrival time distribution
38	State	transaction state
39	SrcWin	source TCP window advertisement
40	DstWin	destination TCP window advertisement
41	TcpRtt	TCP connection setup round-trip time
42	SynAck	the time between the SYN and the SYN-ACK packets
43	AckDat	the time between the SYN-ACK and the ACK packets
44	Load	bits per second

Table 4.3: Generated flow features

4.3.1 Labeling

For further analyzing the dataset, we had to label each flow.
as we discussed before, our dataset contains legitimate and malicious flows; hence, firstly we labelled them based on the attack or non-attack for binary

classification, secondly, each flow was labelled based on the types of attack, which are Heavy-Flood, Connect-Flood, Fast-Flood and legitimate flow for multi-value classification.
For binary classification, we performed labelling based on the IP addresses of the publishers. As we discussed before, the IP addresses of the legitimate sensors have been specified, therefore, all the sensors with source IP 172.20.10.5 were considered as a non-attack flow and labelled as 2. All the other flows were considered as an attack flow and we labelled them as 0. In the case of multi-value classification, we labelled the flows based on the IP addresses, however, because Connect-Flood and Fast-Flood were publishing from the same device, we had to consider the start time of the sensors for labelling them correctly.
Labelling of binary and multi-value classifications are demonstrated in table 4.4 and table 4.5.

Type	**Class**
Malicious traffic	0
Legitimate traffic	1

Table 4.4: Binary classification

Type	**Class**
Heavy-Flood traffic	0
Fast-Flood traffic	1
Connect-Flood traffic	2
Legitimate traffic	3

Table 4.5: Multi-value classification

4.3.2 Balancing

A balanced dataset includes an equal or approximately equal number of samples from the legitimate and malicious class. For achieving a more reliable, accurate and balanced detection rate machine learning model, we must

have a balanced dataset for our classification model. Due to the different number of sensors and different rate of publishing, our dataset was not balanced. As figure 4.2 illustrated, In the case of multi-value classification, the number of Connect-Flood attack was more than the others and Fast-Flood was the least one. 240 flows of each label were chosen randomly to have the balanced dataset.

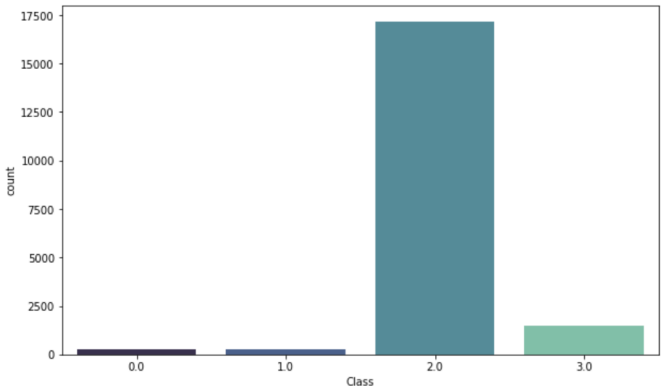

Figure 4.2: Unbalnced dataset - Multi-value classifcation

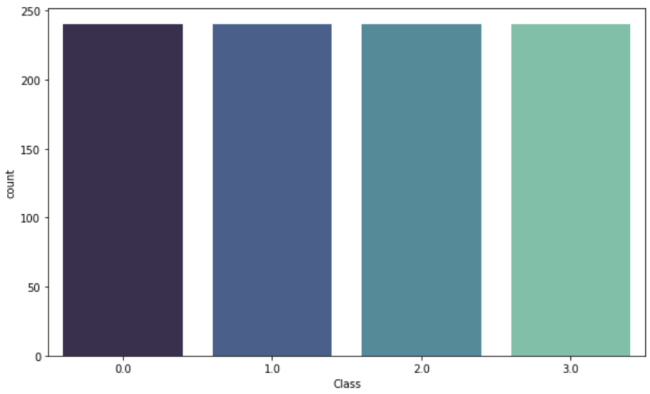

Figure 4.3: Balanced dataset - Multi-value classifcation

The Full-Featured dataset was unbalanced in Binary classification, therefore, we chose 500 samples of each label randomly to create the balanced dataset.

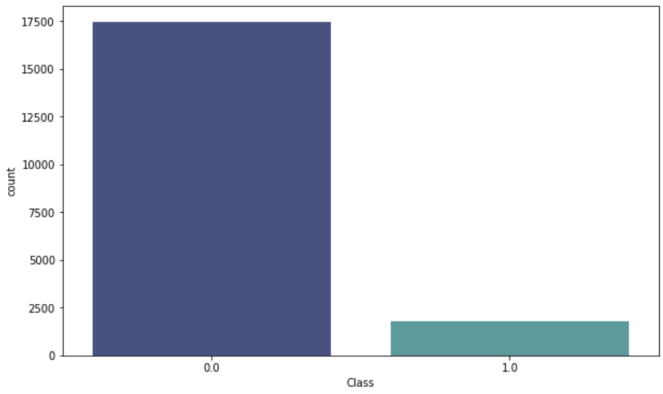

Figure 4.4: Unbalnced dataset - Binary classifcation

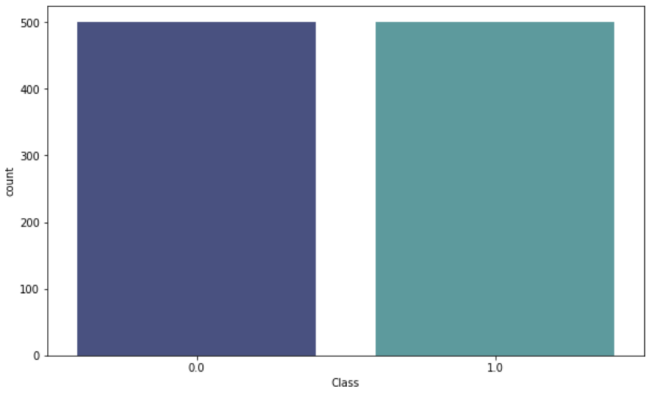

Figure 4.5: Balanced dataset - Binary classifcation

4.3.3 Label Encoding

Label encoding is the process of converting labels into the numeric form so that they can be read by machines. Machine learning algorithms can make better decisions about how those labels should be used. It is an essential preprocessing step for the structured dataset in supervised learning.
The representation of categorical data can be more expressive with a one-hot encoding. Many machine learning algorithms are unable to operate directly with categorical data. The categories must be numerically transformed [20]. In our Full-Featured dataset, "State" and "Flags" were presented categorically, hence, we applied one-hot encoding to extract numerically features from them.

4.3.4 Redundant features elimination

In the labeling procedure, we used "StartTime", "LastTime", "SrcAddr", "DstAddr" features to label the data; thus, after labeling all of them were removed from dataset. "sport" and "dport" which was fixed for all the flows were eliminated. Due to the high number of missed values, we removed "SIntPkt", "SIntDist", "SrcWin" and "DstWin".

The details of the Full-Featured datasets are demonstrated in table 4.6.

Dataset	Features	Instances	CSV size
Full-Featured - Before cleaning	44	34036	10.6 MB
Full-Featured Multi-value	37	960	266 KB
Full-Featured Binary	37	1000	201 KB

Table 4.6: Full-Featured datasets

4.4 Feature Selection

Feature selection is a fundamental concept in machine learning that has a significant impact on the model's performance. Feature selection can reduce the training time and improve the accuracy of the classifiers [21]. The data attributes that use to train the machine learning models will have a significant impact on the achieved results. In our scenario, we performed feature selection to reduce the number of features because reduction of the features lead to the lower machine learning training time and efforts. A further reason for doing feature selection is to improve the model generalization capability. In this work, we used Filter-based selection technique. After applying the filter-based technique, redundant columns are filtered out from the model. we have to choose a statistical measure that suits our data, then the module calculates a score for each feature column. The columns are passed ranked by their feature scores. In our scenario, because most of the data were numerical and the target was categorical, Analysis of Variance (ANOVA) was chosen as a filter-based selection method.
As Figure 4.6 shows, ANOVA is a parametric statistical hypothesis test for determining whether the means from two or more samples of data (often three or more) come from the same distribution or not [20]. For applying

the ANOVA method on our dataset we used f-classif() function which is provided by the scikit-learn library. This function can be used for feature selection strategy, like selecting the top k most relevant features via the SelectKBest class. For instance, we can define the SelectKBest class to use the f-classif() function and select all features, then transform the train and test sets. After that, We printed the scores for each feature to find out the 10 best features [22].

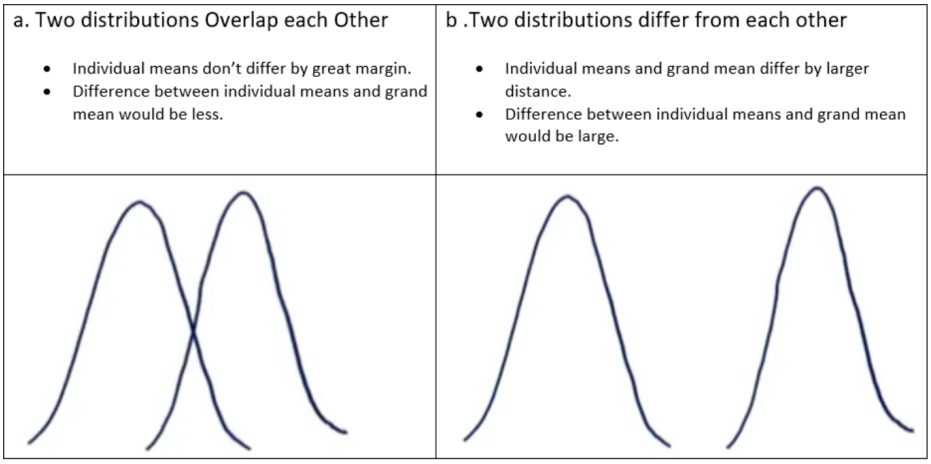

Figure 4.6: Behavior destributions

Source: [23]

4.4.1 Top-10 Best Features Dataset

After applying feature selection, we created datasets for multi-value and binary cases with 10 best features which are obtained statistically. 10 best features and ANOVA scores of them, for binary and multi-value classification problems are demonstrated in Table 4.7 and Table 4.8.

Feature	Description	ANOVA Score
State-CON	Reported transaction is active	1091.797346
State-FIN	Reported transaction is closed	773.240791
DstRate	Destination pkts per second	268.743461
SrcRate	Source pkts per second	252.760566
Rate	Pkts per second	172.975403
TcpRtt	TCP connection setup round-trip time	133.061976
AckDat	The time between the SYN-ACK and the ACK packets	130.633730
SynAck	The time between the SYN and the SYN-ACK packets	120.424206
Load	Bits per second	32.709531
Dur	Record total duration	31.573044

Table 4.7: Top-10 Best Features dataset - Binary classification

Feature	Description	ANOVA Score
Flgs-e *	Both Src and Dst loss/retransmission	1032.465791
Flgs-e	Ethernet encapsulated flow	448.155430
DstLoss	Destination pkts retransmitted or dropped	400.635048
State-FIN	Reported transaction is closed	252.980545
Flgs-e s	Src loss/retransmissions	248.745935
pLoss	percent pkts retransmitted or dropped	207.513853
Load	bits per second	159.877985
State-CON	Reported transaction is active	159.767769
DstPkts	src -> dst packet count	158.117338
DstBytes	dst -> src transaction bytes	146.474182

Table 4.8: Top-10 Best Features dataset - Multi-value classification

Chapter 5

Machine learning models

In this chapter, we describe our proposed machine learning models which are capable to detect and classify malicious attacks in case of multi-value and binary classification problems.
In section 5.1 we explain the machine learning algorithms adopted in this work. After that, In section 5.2, the performance evaluation techniques are described. In section 5.3, we define our proposed offline and online machine learning models. At the end of this chapter, in section 5.4, the deployment is discussed.

5.1 Machine learning algorithms

Machine learning is a branch of artificial intelligence (AI). The main focus of machine learning is on the use of data and algorithms to learn automatically without human assistance. There are two main kinds of machine learning, supervised and unsupervised learning. Supervised learning relies on valuable information in labeled data. Classification is the most popular task in supervised learning, however, labeling data is costly and takes considerable time. Consequently, the lack of sufficient labeled data forms the main disadvantage of supervised learning. In opposite to supervised learning, unsupervised learning selects feature information from unlabeled data, making it much easier to obtain training data. However, the detection performance of unsupervised learning methods is lower than supervised learning methods [3].
In our scenario, as we explained in chapter 4, we labeled our data based on binary and multi-value, plus our main goal is to classify and detect attacks; so, we have to apply classification which is a supervised learning concept. The main classification algorithms are Random Forest, Support Vector Ma-

chine, and K-nearest neighbor. In the following sections, these algorithms are briefly described.

5.1.1 Random Forest

As the name implies, Random Forest is a technique that uses Ensemble Learning and is based on the bagging algorithm. It generates as many trees as possible on a subset of the data and then merges the results of all the trees and the class with the most votes becomes the prediction of our model. As a result, the overfitting problem in decision trees is reduced; thus, the accuracy will be improved. In addition, Random Forest works very well with both categorical and numerical features. Overfitting is a statistical modeling error that arises when a function is too tightly fitted to a small number of data points. Figure 5.1 shows how Random Forest algorithm combine trees and predict the result.

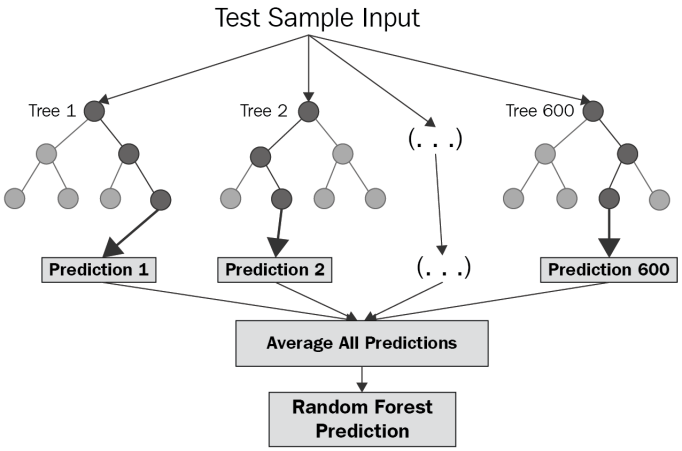

Figure 5.1: Random Forest algorithm

Source: [24]

5.1.2 K-nearest neighbour

The k-nearest neighbors (KNN) algorithm is one of the simplest, supervised machine learning algorithm that can solve both classification and regression problems. The KNN algorithm works as a majority voting mechanism. It collects data from a training dataset and uses this data later to make predictions for new records. For every new record, the k-closest records of the training data set are defined. Based on the value of the target attribute of the closest records, a prediction is performed for the new record.

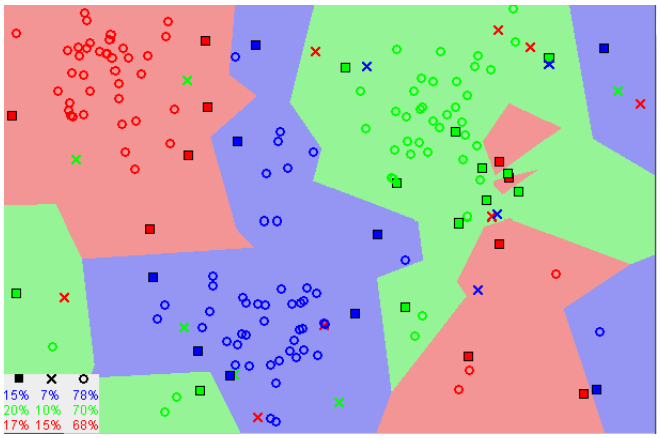

Figure 5.2: K-nearest neighbour algorithm

Source: [25]

As Figure 5.2 illustrates, similar data points are frequently gathered together. The KNN algorithm relies on this assumption being correct in order to function properly. For capturing the idea of similarity, KNN calculates distances between data points. For measuring the distances, KNN uses different distance metrics, however, the most common distance metric is the Euclidean distance function which is the default in the SKlearn KNN classifier library in Python. Euclidean distance formula is shown in equation 5.1.

$$d(x, y) = \sqrt{\sum_{n=1}^{n} (x_i - y_i)^2} \tag{5.1}$$

5.1.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. As Figure 5.3 shows, the support vector machine algorithm's goal is to find a hyperplane in an N-dimensional space (N is the number of features) that distinguishes between data points.

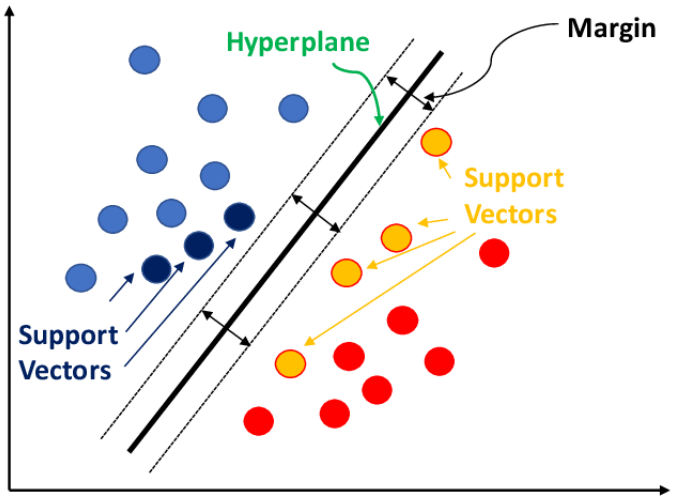

Figure 5.3: Support Vector Machine algorithm

Source: [26]

If you consider Figure 5.3, Hyperplace is a line that linearly classifies a set of data. The margin is the distance between the hyperplane and the nearest data point. To increase the probability that new data being classified correctly, we have to choose a hyperplane with the greatest possible margin between the hyperplane and any point in the training set.

5.2 Performance evaluation metrics

Evaluating the prediction performance of a supervised classification method concerning unseen data is critical in machine learning. A machine learning model can be trained perfectly, however without evaluation, it is not trustworthy. In our scenario, since our goal is to propose reliable and generalize classifiers, we have to be cautious about final performance indexes such as F1-score, plus we have to consider the generalization of the classifiers to avoid overfitting or underfitting. In this section, we explain the main performance evaluation metrics that we used to examine the performance of the classifiers.

5.2.1 Confusion matrix

One of the best ways to visualize the performance of machine learning is the confusion matrix. Each row of the matrix describes the instances in an actual class and each column describes the instances in a predicted class. Regarding binary classifier, as shown in figure 5.4, the Confusion Matrix is

defined by 4 entries. The True Positives TP and True Negatives TN describe the correct classified instances, in our scenario, they indicate respectively how many attacks have been correctly detected and how many are recognized truly as legitimate flow. On the other hand, the False Positives FP and False Negatives FN report the cases of false classification, FP contains cases in which a legitimate flow has been detected as an attack flow, also called false alarms, while FN indicates the number of opposite cases, i.e. those in which an attack flow has been identified as a legitimate flow.

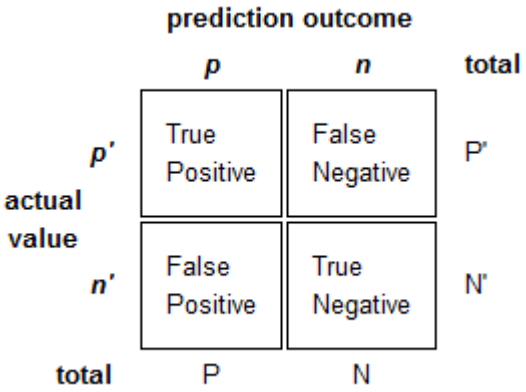

Figure 5.4: Confusion Matrix

5.2.2 Performance Measures

The F1 score also called the F score or F measure is a measure of classification accuracy. The F1 score is defined as the harmonic mean of the classification precision and recall.
Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives and the definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. For calculating F1, we must measured Precision and Recall from the confusion matrix as:

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

Then, from Precision and Recall, we can compute F1-Score:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

On the other hand, the simplest intuitive performance metric is accuracy, which is just the ratio of properly predicted observations to all observations, however, Accuracy is not the best choice in the real world due to the imbalanced datasets; hence, we decided to use F1-score as our main metric.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.5}$$

5.2.3 Learning curves

Before describing the learning curve, we have to explain the concept of overfitting and underfitting in machine learning. Overfitting is a statistical modeling error that arises when a machine learning model is too tightly fitted to the training data; therefore, the machine learning performance on training data is very well, however, the performance is poor in case of the unseen data. Specifically, overfitting happens if the model shows low bias but high variance. On the other hand, underfitting is a concept that indicates a model that cannot model the training data and cannot generalize to new data. intuitively, underfitting occurs if the model or algorithm shows low variance but high bias. Underfitting is often a result of an excessively simple model. One of the best tool in machine learning to diagnose overfitiing and underfitting is learning curve. Learning curves allow us to diagnose bias and variance in supervised learning models. They show the relationship between training set size and the chosen evaluation metric (e.g. F1-score, accuracy, etc.) on the training and validation sets. Generally, by increasing the number of training samples, the performance of the model improves. Learning curves can be used to determine how well a model can generalize to new data and to determine if the model has learned everything it can from the dataset. As Figure 5.5 shows, in a Good-fitted model, by increasing the training samples the cross-validation score improving and at the end, the training score and cross-validation score converged together.

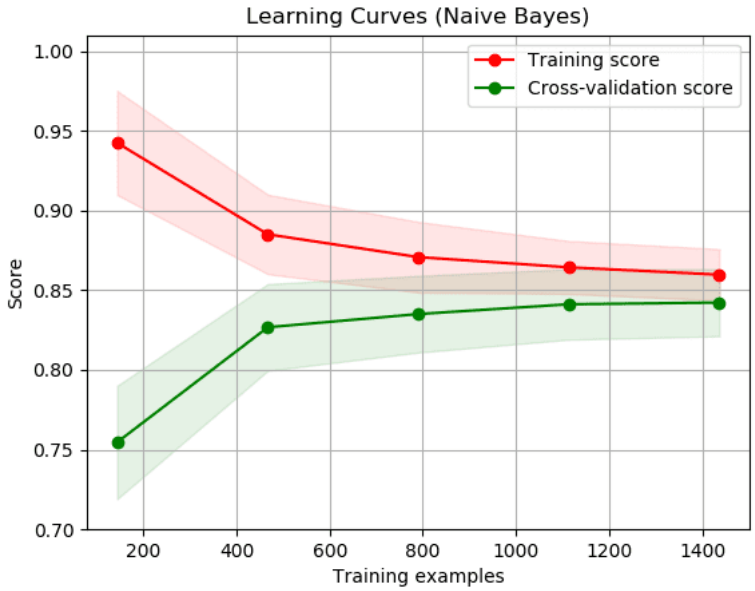

Figure 5.5: Good-fitted model Learning curve

Source: [27]

5.3 Machine learning models

In this section, we describe our proposed online and offline machine learning models. Online machine learning is a technique of machine learning in which data becomes available in consecutive series. Online machine learning is used to find the best classifier for future data at each step, on the other hand, offline machine learning techniques recommend the best classifier by learning on the entire training dataset.In section 5.3.1, we explain how we trained the offline classifiers, and in section 5.3.2 the online machine learning models are described.

5.3.1 Offline machine learning models

In offline machine learning, we considered the entire datasets for validating the classifiers. The maximum flow duration in entire datasets is 60 seconds. We trained Random Forest, KNN and SVM models on Full-Features and 10-Best Features datasets. Scikit-learn library were used for training and validating the classifiers. The classifiers were trained on a train set which was 70% of the dataset then they evaluated on the remaining part which was

30% of the dataset. Our proposed offline machine learning models are capable to detect and classify malicious and legitimate traffic in case of binary, and classify attack categories in a multi-value problem.

5.3.2 Online machine learning models

In cyber-attack detection, it is crucial to detect malicious attacks at different moments of time; ergo, online machine learning detectors are able to solve this problem. In our scenario, as the anomaly TCP packets were publishing to the broker consecutively, we proposed online machine learning models which are capable to detect and classify the anomalies in different TCP flow durations, from flows truncated up to 5 seconds to 60 seconds flow duration. As we defined before, TCP flows contain TCP packets with the same source and destination IP and port addresses which are captured during a limited time interval. For instance, imagine a sensitive application that desires to detect the ongoing attacks promptly or an application that requires to classify attacks at least in 60 seconds. Our proposed cyber-attack detector can be located on the edge of network to detect the malicous attack in different flow durations. The main advantage of this classifier is the ability to detect attack in just 5 seconds. For implementing the online machine learning classifiers, we considered three main ranges for TCP flow durations as a threshold which are low-range, mid-range, and high-range. low-range contains 5 and 10 seconds, mid-range is 25 and 30 seconds, and high-range is 60 seconds. We trained three main classifiers, Random Forest, Support Vector Machine, and K-nearest neighbor on Full-Featured and Top 10-Best-Features datasets to predict different classes in case of binary and multi-value classifications in each flow duration threshold. In this project, 60 seconds flow duration was considered for training the classifiers because by nature, DoS attacks have relatively longer durations; thus, in 60 seconds the flow features would reach statistical stability. To notice the trade-off between accuracy and flow durations, we decided to evaluate our models on 5, 10, 25, 30, and 60 seconds flow durations. For examining the accuracy of our model in 60 seconds, the dataset was split into train set and test set. The classifiers were trained on a train set which was 70% of the dataset and then they evaluated on the remaining part which was 30% of the dataset.
We proposed a reliable classifier for each flow duration threshold which can be used depends on the application. For implementing the ML models, we used Scikit-learn machine learning library which is availabe for Python programming language, and to interpret the plots, Matplotlib and Seaborn were used. In the next chapter the results of the online machine learning models are discussed.

5.4 Deployment

In this section, the deployment of our proposed classifiers on the 5G network is described. 5G is a brand-new cellular network that seeks to achieve massive gains in quality of service (QoS), such as higher throughput and reduced latency [28]. Multi Access Edge computing (MEC) is a way of gathering, storing, processing, and analyzing data close to the client, rather than in a centralized data-processing center [29]. As Figure 5.6 shows, Edge computing analyzes data from IoT devices at the edge of the local network before sending them to the main server; thus, it reduces latency and improves reliability. In our scenario, there are many advantages to install our proposed classifiers on the edge of the network. By implementing our proposed classifiers on the Edge of the local network, due to proximity to the end-users, classifiers can detect and classify malicious attacks with minimal latency in comparison to the main data center, plus there is no need to filter out other networks traffic. In addition, our classifiers were trained on the selected features; therefore, they do not required too much computation power. For instance, in a sensitive-delay application that is connected to the 5G network, in case of a DoS attack, our proposed online classifier which is located on the Edge of the network can detect an attack in real-time with lower latency in comparison to attack-detection installed on the cloud.

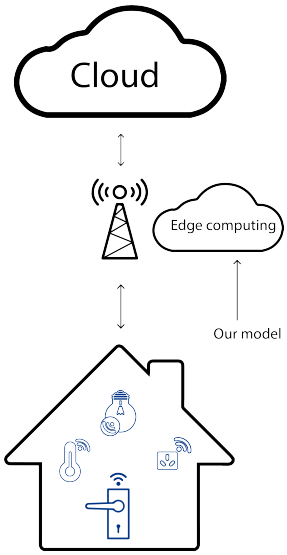

Figure 5.6: Deployment

Chapter 6

Results

In this chapter, we explain the achieved results of the binary and multi-value classification by applying machine learning algorithms on the Full-Featured and 10-Best Features datasets. In section 6.1, we discuss the offline machine learning models' results, then, in section 6.2, the results of online machine learning models are fully described.

6.1 Offline machine learning results

As we discussed in Chapter 5, RF, KNN, and SVM classifiers were trained on 70% of the Full-Featured and 10-Best Features datasets, then they evaluated on 30% of the Full-Featured and 10-Best Features datasets. In this case, due to the offline classification, the flow duration is considered as maximum which is 60 seconds. The multi-value classification and binary classification results are described in sections 6.1.1 and 6.1.2 respectively. In Table 6.1, offline machine learning results are shown.

6.1.1 Multi-value classification

In the case of multi-value classification, the results of the offline classifiers which are evaluated on the Full-Featured dataset indicate that Random Forest and KNN have the highest score with a 0.95 F1-score, and SVM is able to detect and classify attack categories with a 0.86 F1-score; thus, in this case, RF and KNN are the best classifiers, however, the results of the offline classifiers which are evaluated on the 10-Best Features dataset pointed, KNN is the best classifier with a 0.95 F1-score. RF and SVM with a 0.91 and 0.86 F1-score can detect malicious attacks. The obtained results indicate that the KNN classifier is the best model obtained from 10-Best Features and in the

case of the Full-Featured dataset, KNN and RF are the best models.

6.1.2 Binary classification

In the case of binary classification, the results of RF and KNN from the Full-Featured dataset are 0.996 and 0.98 F1-score, however, SVM F1-score is 0.95. In the case of 10-Best Features dataset, the best classifier is RF with 0.997, KNN, and SVM with a 0.99 and 0.84 F1-score can detect malicious attacks.

Dataset	Model	Mult-value F1-Score	Binary F1-score
Full-Featured	RF	0.95	0.996
	SVM	0.86	0.95
	KNN	0.95	0.98
10-Best Features	RF	0.91	0.997
	SVM	0.86	0.84
	KNN	0.95	0.99

Table 6.1: Offline machine learning results

6.2 Online machine learning results

In online machine learning, due to the longer duration of DoS attacks, 60 seconds flow durations was considered for training the classifiers; thus, in 60 seconds the flow features would reach the state of stability. To notice the trade-off between accuracy and flow durations, we decided to evaluate our models on 5, 10, 25, 30, and 60 seconds flow duration. In sections 6.2.1 and 6.2.2, the results of the binary and multi-value classification are explained.

6.2.1 Multi-value classification

In multi-value classification, we would like to predict Connect-Flood, Heavy-Flood, Fast-Flood and legitimate flows which are completely described

in chapter 4.
In the following sections, we show the machine learning obtained results from Full-Feature dataset and 10-Best Features dataset.

6.2.1.1 Full-Featured dataset results

As it is shown in Table 6.2, our proposed classifiers are able to detect and classify the malicious attacks in just 5 seconds flow length. Random Forest can detect an attack with a 0.76 F1-score which is the highest between other classifiers in 5 seconds flow duration. KNN with a 0.6 F1-score is more accurate in comparison to SVM with a 0.55 F1-score. If you consider 10 seconds flow duration, there is an improvement in the results of the classifiers. RF can classify with a 0.86 F1-score, KNN is able to detect attacks with a 0.68 F1-score, however, the worst one in detection is SVM with a 0.63 F1-score. So, in 10 Seconds flow duration, Random Forest was the best classifier. In 25 seconds flow duration, there is a huge improvement in KNN, The F1-score of KNN increased to 0.88. SVM F1-score is improved slightly to 0.64. Like 5 and 10 seconds, in 25 seconds, the best predictor was RF with a 0.91 F1-score. As Table 6.2 shows, in 30 seconds flow length, KNN with a 0.92 F1-score is the best classifier. The F1-score of the RF has not changed with respect to the 25 seconds, however, there is a big enhance in SVM F1-score which is 0.77 in 30 seconds. In 60 seconds flow length, although all the classifiers are capable to detect attacks very well, RF and KNN with a 0.95 F1-score predict better respect to the SVM F1-score which is 0.86. As obtained results indicate, in this case, The best classifier in Multi-value classification is Random Forest. As we expected, there is a trade-off between flow durations and F1-score, in other words, by increasing the flow duration, the F1-score improved.

Flow duration	Model	F1-Score
5 Seconds	RF	0.76
	SVM	0.55
	KNN	0.60
10 Seconds	RF	0.86
	SVM	0.63
	KNN	0.68
25 Seconds	RF	0.91
	SVM	0.64
	KNN	0.88
30 Seconds	RF	0.91
	SVM	0.77
	KNN	0.92
60 Seconds	RF	0.95
	SVM	0.86
	KNN	0.95

Table 6.2: Multi-value online classification results - Full-Featured

6.2.1.2 10-Best Features dataset results

The results of multi-value classification which are obtained by applying ML algorithms on 10-Best features are demonstrated in Table 6.3. The results in 5 seconds flow length show that the Random Forest model with a 0.67 F1-score is the best classifier. KNN and SVM with a 0.6 and 0.55 F1-score respectively are able to detect malicious attack categories in 5 seconds flow duration. In 10 seconds flow duration, RF with a 0.8 F1-score has the

highest score. In 10 seconds flow duration, the F1-score of KNN and SVM are close together which are 0.68 and 0.63 respectively. In 25 seconds, KNN and RF F1-score are the same which is 0.88, however, SVM F1-score increased slightly to 0.64. If you consider 30 seconds flow duration, KNN is the best classifier with a 0.92 F1-score, however, Random Forest can detect attack categories with a 0.89 F1-score, and F1-score for SVM is 0.77 in 30 seconds flow length. Finally, KNN is able to detect and classify with a 0.95 F1-score which is the highest score among the other classifiers in 60 seconds flow duration. RF and SVM F1-score in 60 seconds are reported as 0.91 and 0.86 respectively.
As the achieved results pointed, Random Forest is the best classifier to detect and classify attacks in 5 and 10 seconds flow duration, however, for detecting attacks in more than 25 seconds flow duration, KNN would be the best option.

Flow duration	Model	F1-Score
5 Seconds	RF	0.67
	SVM	0.55
	KNN	0.60
10 Seconds	RF	0.80
	SVM	0.63
	KNN	0.68
25 Seconds	RF	0.88
	SVM	0.64
	KNN	0.88
30 Seconds	RF	0.89
	SVM	0.77
	KNN	0.92
60 Seconds	RF	0.91
	SVM	0.86
	KNN	0.95

Table 6.3: Multi-value online classification results - 10 Best-Features

Generally, the classifiers were trained by the Full-Featured dataset have higher accuracy in comparison to the classifiers that were trained by 10-Best features. For finding the reason for the lower results after feature selection, we plot the learning curves of the Random Forest model in 60 seconds flow duration. We considered 60 seconds flow duration due to the stability of the flow's features. The F1-score of the RF in the Full-Featured dataset is 0.95 however the obtained result from 10-Best features is 0.91 F1-score. As Figure

6.1 shows, Although the final result of the RF in the case of Full-Featured is higher, there is an evidence of overfitting in the case of Full-Featured, however, the learning curve of 10-Best Features (Figure 6.2) shows a good fitting; therefore, we proposed the RF classifier which trained on 10 Best Features as the best classifier for 60 seconds flow duration.

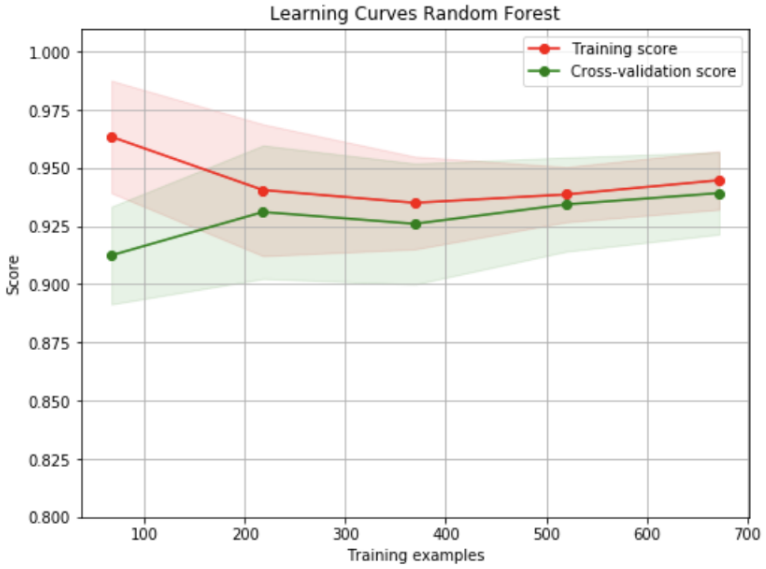

Figure 6.1: RF learning curve - Multi-value - Full-Featured

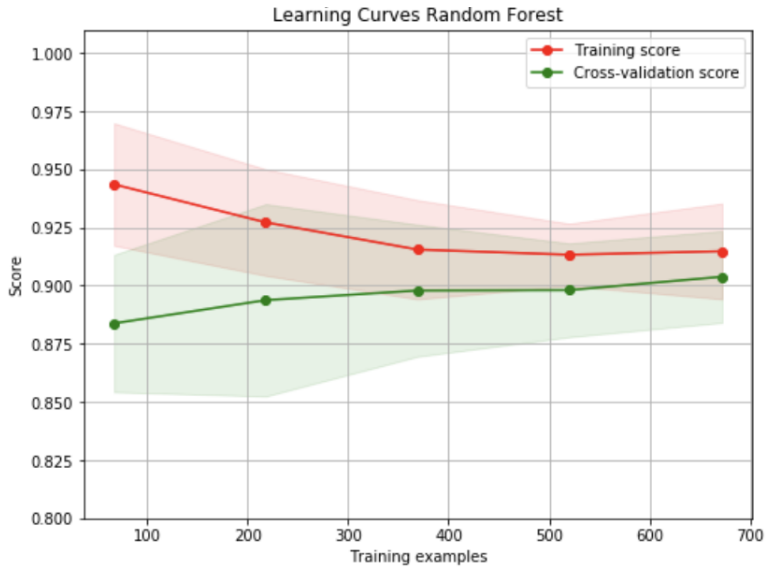

Figure 6.2: RF learning curve - Multi-value - 10 Best-Features

After plotting the learning curves, we demonstrate the confusion matrix for multi-value classification to see more details of the predictions. As we explained in chapter 4, label 0 is Heavy-Flood, 1 is labeled as Fast-Flood, Connect-Flood is considered as label 2 and legitimate traffic is 3. In Figure 6.3, the confusion matrix of the Random Forest classifier at 60 seconds flow duration obtained from the Full-Featured dataset is demonstrated. Legitimate flows were correctly detected, there is just one error in Connect-Flood which is labeled as Heavy-Flood incorrectly. 58 out of 65 Fast-Flood were truly classified and there are 6 errors in Heavy-Flood which were detected as Connect-Flood incorrectly.
The confusion matrix of the Random Forest classifier at 60 seconds flow duration which obtained from the 10-best features dataset is shown in Figure 6.4. As the Figure shows, 17 out of 74 Connect-Flood were labeled as legitimate flows incorrectly. 2 instances of Heavy-Flood were detected as Connect-Flood falsely. From 81 Fast-Flood instances, 4 instances were labeled incorrectly as Heavy-Flood and Connect-Flood. In the case of the legitimate flow, RF detected 75 instances correctly and just 3 instances were classified as Connect-Flood wrongly.

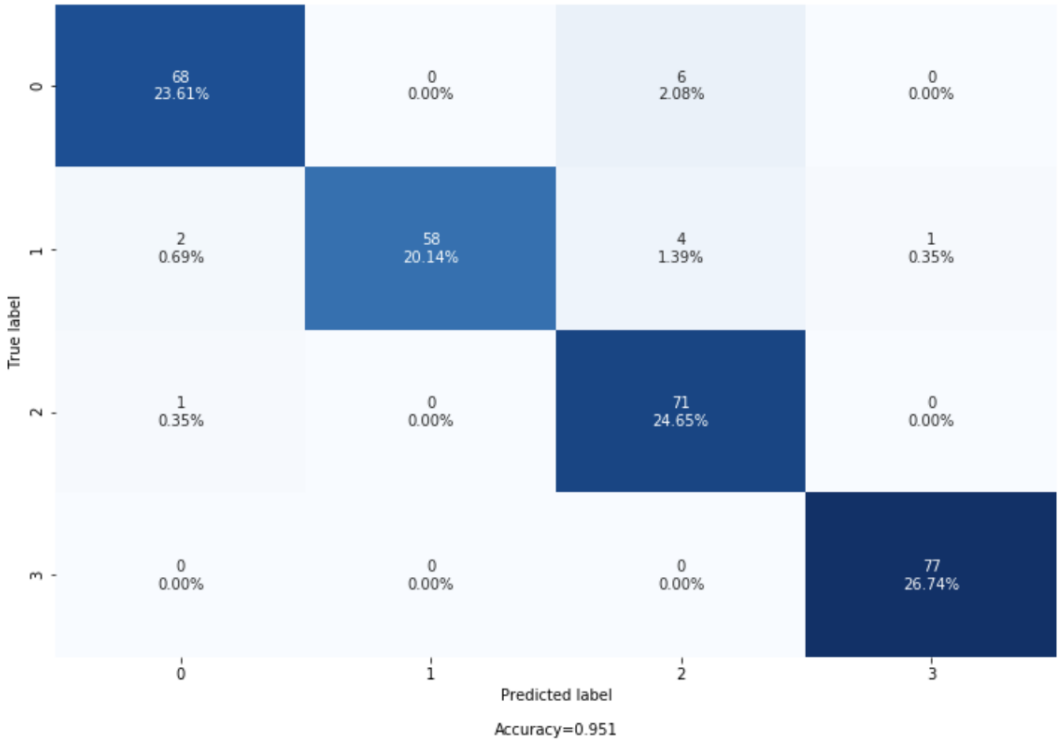

Figure 6.3: RF Confusion matrix - Multi-value - Full-Featured

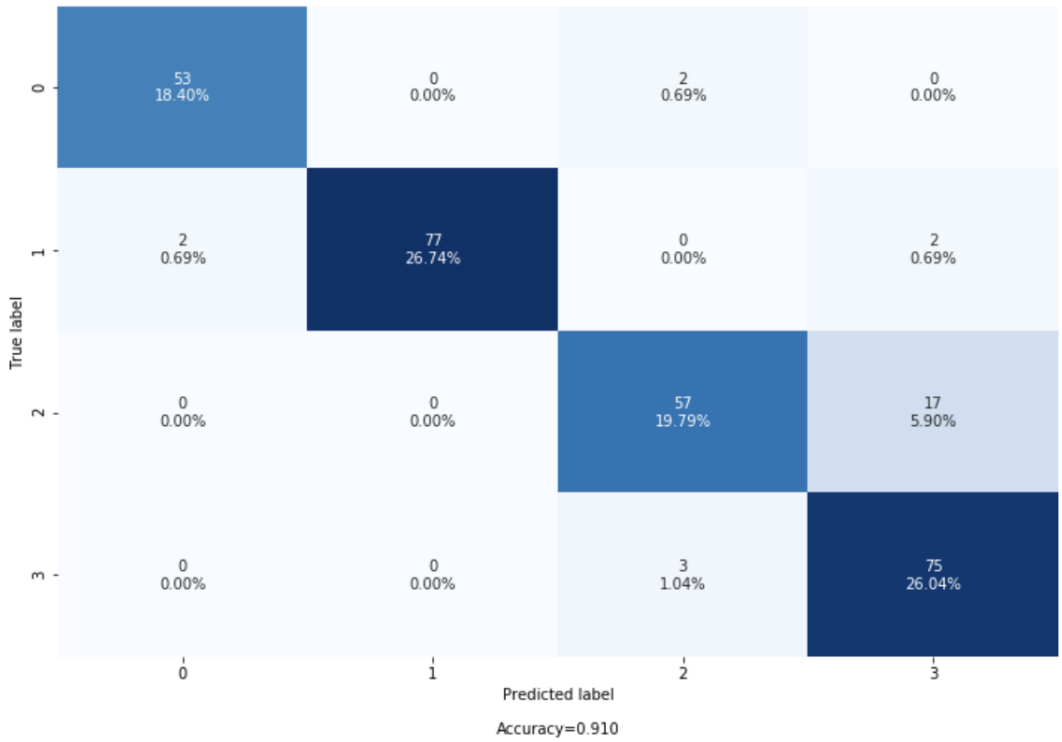

Figure 6.4: RF Confusion matrix - Multi-value - 10 Best-Features

As a final result of the Online machine learning in multi-value classification, by increasing the duration of the flows, the accuracy of the classifiers are improved. In Figure 6.5, The standard deviation and average score between RF, SVM, and KNN models for each flow duration is demonstrated. As it illustrates, our proposed models can classify the attack categories and legitimate flow in 5 seconds with an average 0.55 F1-score, however, in 60 seconds flow duration, the average F1-score is around 0.9 F1-score. Mid-range durations F1-score are around 0.8. The results show that there is a trade-off between flow durations and classifier accuracy.

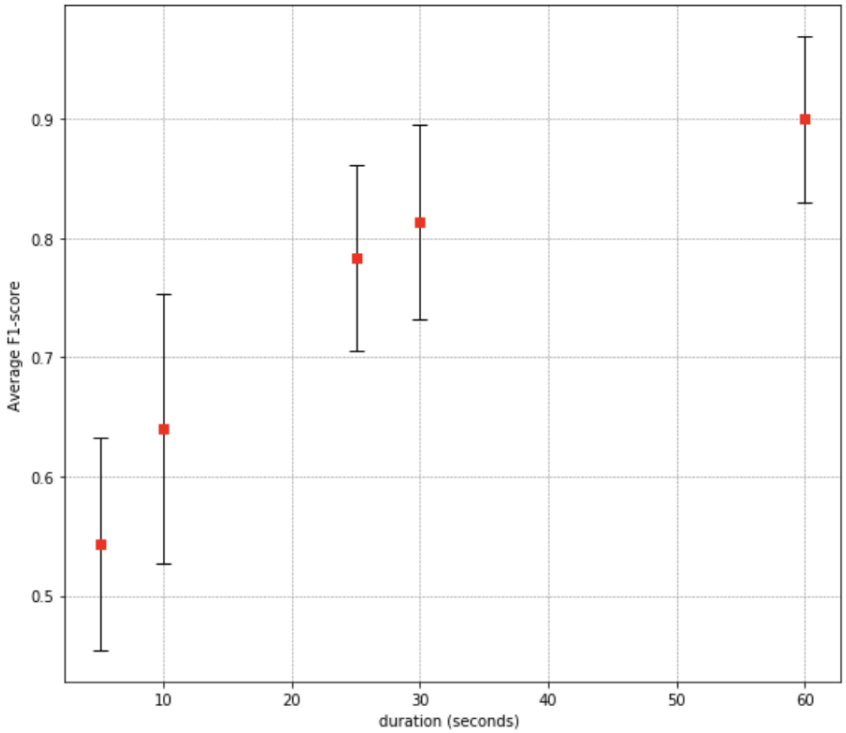

Figure 6.5: Average scores - online multi-value classification

6.2.2 Binary classification

In binary classification, we try to classify and detect legitimate and malicious attacks.
In the following sections, we describe the results of applying ML algorithms on Full-Featured dataset and 10-Best-Features dataset.

6.2.2.1 Full-Featured dataset results

As Table 6.4 illustrates, our classifiers can detect malicious attacks in the case of binary classification in 5 seconds flow duration. In 5 seconds flow duration, the Random Forest model is able to classify attacks with a 0.97 F1-score. F1-score for SVM and KNN are 0.85 and 0.66 respectively, so, the best classifier in 5 seconds flow duration is Random Forest. In 10 seconds flow duration, RF F1-score is 0.99 which is higher than SVM with a 0.92 F1-score and KNN with a 0.91 F1-score. If we consider 25 seconds flow duration, RF remains with a 0.99 F1-score however there is an improvement in the F1-score of SVM and KNN. SVM score is 0.96 and KNN score is 0.93 in 25 seconds flow. RF and KNN are capable to classify attacks in 30 seconds flow

duration with 0.99 and 0.97 F1-score respectively, however, SVM F1-score in 30 seconds is 0.93. In 60 seconds flow duration, all the classifiers can detect attacks with higher accuracy. RF F1-score is 0.996 and KNN F1-score jump to 0.98, SVM can detect malicious flows with a 0.95 F1-score.
Like multi-value classification, there is a trade-off between classifier accuracy and flow duration here. In other words, as Table 6.4 shows, by increasing the flow duration, the classifier accuracy enhances.

6.2.2.2 10-Best features dataset results

In Table 6.5, the ML results achieved from the 10 best features are shown. As table 6.5 shows, Random Forest is the best classifiers in 5, 10, 25, 30, and 60 seconds flow duration. The F1-score of RF in 5 seconds flow duration is 0.98 and in 10 seconds duration, it increased to 0.99. In the case of 25, 30, and 60 seconds, the F1-score is 0.997 which is the highest score. On the other hand, in this case, the SVM is the worst classifier. The F1-score of SVM in 5 seconds flow duration is 0.74, however, in 10 seconds it reached 0.76. Then by increasing the flow duration, the F1-score of SVM is improving, so, in 25, 30, and 60 seconds, the result is 0.78, 0.79, and 0.84 respectively. If you consider KNN, in 5 and 10 seconds, the F1-score is 0.92 and in 25 and 30 seconds, the F1-score increased to 0.95. The highest F1-score of KNN is in flows with 60 seconds duration.
As the obtained results show, in terms of F1-score, in all the flow durations, RF is the best classifier. In flows in which the duration is truncated up to 5 seconds, RF is able to detect attacks with a 0.98 score, however, in longer flows, the accuracy of the classifier is better.

Flow duration	Model	F1-Score
5 Seconds	RF	0.97
	SVM	0.85
	KNN	0.66
10 Seconds	RF	0.99
	SVM	0.92
	KNN	0.91
25 Seconds	RF	0.99
	SVM	0.96
	KNN	0.93
30 Seconds	RF	0.99
	SVM	0.93
	KNN	0.97
60 Seconds	RF	0.996
	SVM	0.95
	KNN	0.98

Table 6.4: Binary online classification results - Full-Featured

Flow duration	Model	F1-Score
5 Seconds	RF	0.98
	SVM	0.74
	KNN	0.92
10 Seconds	RF	0.99
	SVM	0.76
	KNN	0.92
25 Seconds	RF	0.997
	SVM	0.78
	KNN	0.95
30 Seconds	RF	0.997
	SVM	0.79
	KNN	0.95
60 Seconds	RF	0.997
	SVM	0.84
	KNN	0.99

Table 6.5: Binary online classification results - 10 Best-Features

If we consider 60 seconds flow duration, The obtained result of RF and KNN from 10 best features are better than the Full-Featured dataset, however, in SVM, the F1-score is decreased from 0.95 to 0.84; hence, we plotted the learning curves of SVM to see what happened. Figure 6.6 shows the learning curve before feature selection and Figure 6.7 illustrates the Learning curve after feature selection. As the figures show, Although the achieved result from the Full-Featured dataset is higher, there is an overfitting in the

learning curve, however, the obtained result of the 10 best features is good-fitted and there is no evidence of overfitting or underfitting. As a result, in case of 60 seconds flow duration, SVM is the best classifier in binary classification.

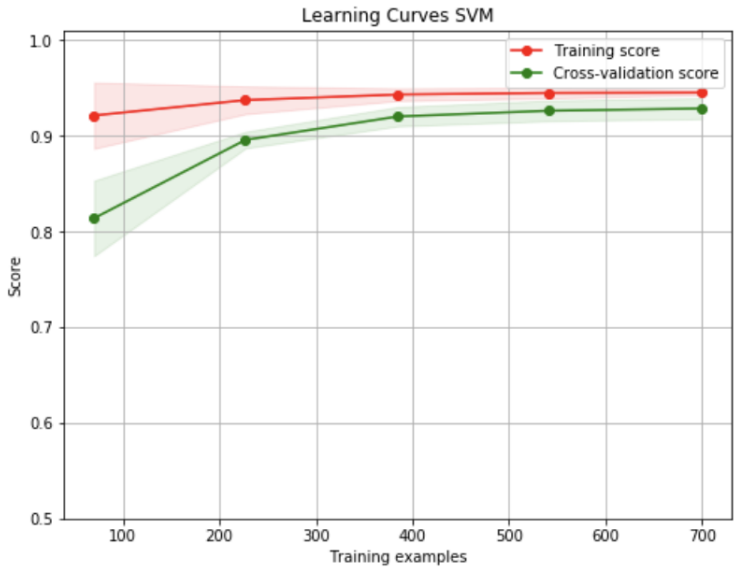

Figure 6.6: SVM learning curve -Binary - Full-Featured

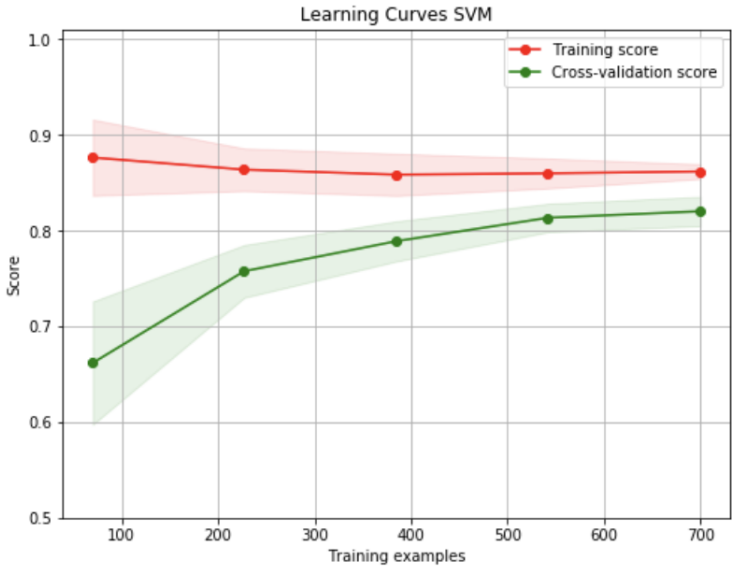

Figure 6.7: SVM learning curve - Binary - 10 Best-Features

In Figure 6.8 and Figure 6.9, we show the confusion matrix for binary classification. As discussed in chapter 4, In binary classification, we consider label 0 as a malicious attack and label 1 as legitimate traffic. for evaluating the performance 60 seconds flow duration is considered. In Figure 6.8, which shows the confusion matrix of the SVM model obtained from the Full-Featured dataset, 5 attack flows are incorrectly reported as legitimate flows and 11 legitimate flows are falsely announced as a malicious attack, however, 146 attack flows and 138 legitimate flows are correctly classified.
In Figure 6.9, the results of the SVM model which achieved from 10 Best-Features are evaluated. As Figure 6.9 shows, 125 attack flows is correctly classified, however 34 attack flows detected as legitimate wrongly. In this case 122 legitimate flows correctly classified, but just 19 legitimate flows detected as attacks.

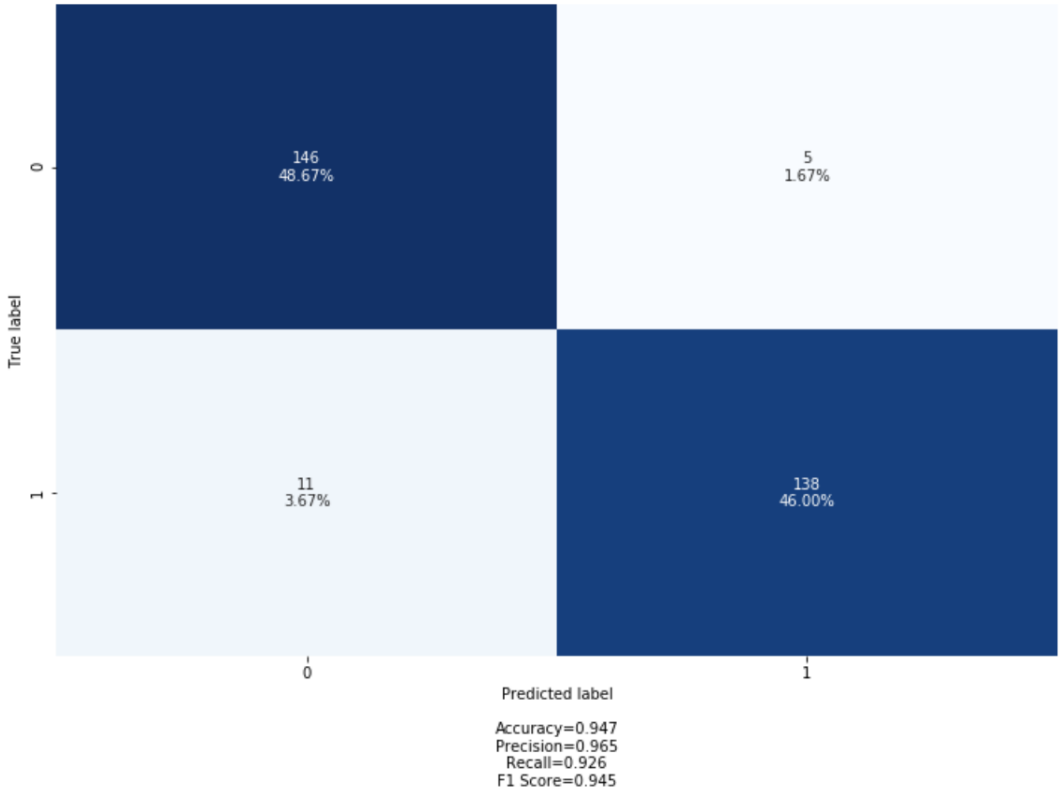

Figure 6.8: SVM Confusion matrix - Binary - Full-Featured

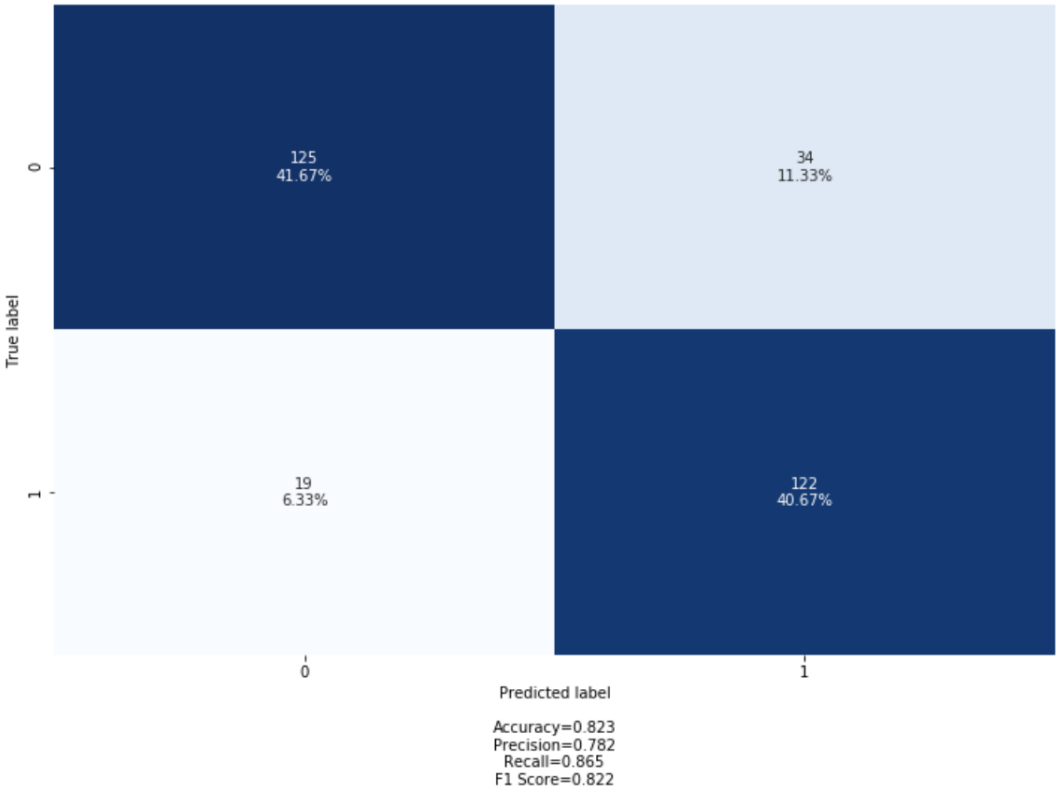

Figure 6.9: SVM Confusion matrix - Binary - 10 Best-Features

The standard deviation and average scores of Machine learning models for each flow duration in case of binary classification is shown in Figure 6.10. As the results are shown, there is a trade-off between flow durations and accuracy of the predictions. Our classifiers can detect the malicious attack in 5 seconds with a higher than 0.85 F1-score, however, if we go around 60 seconds, we can reach a 0.99 F1-score.

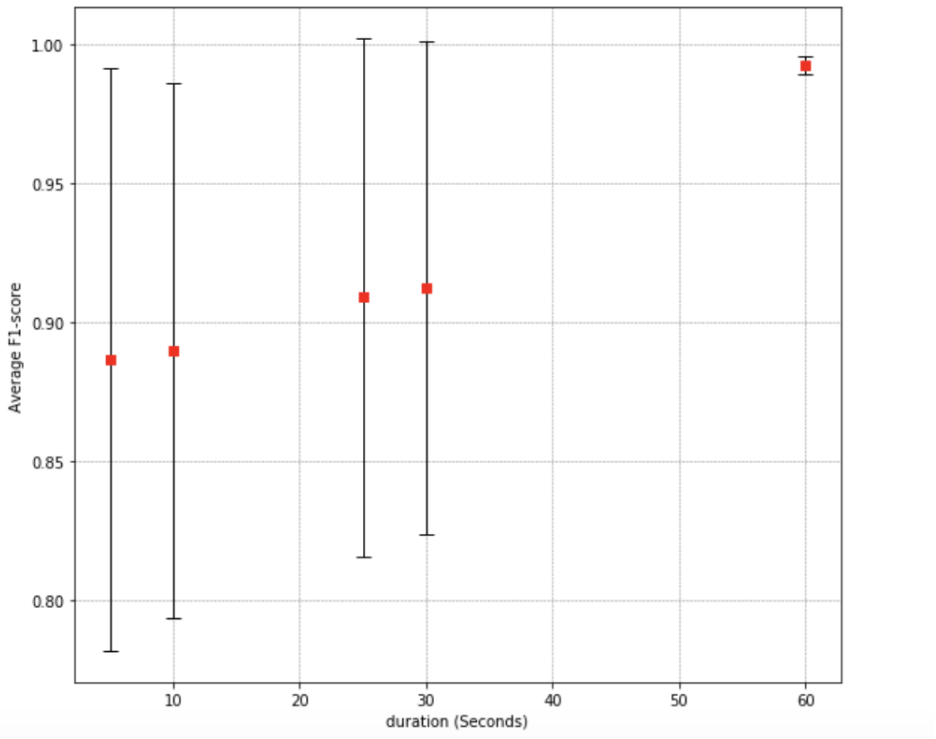

Figure 6.10: Average scores - online binary classification

Chapter 7

Conclusion and Future works

7.1 Conclusion

The main goal of this thesis was detect and classify DoS attacks for MQTT sensor networks. Therefore, we simulated a smart home scenario contains eight Internet of Things sensors to generate legitimate traffic. To publish malicious traffic, three different types of DoS attackers which are Heavy-Flood, Connect-Flood, and Fast-Flood were developed. After that, we started the simulation and sniffed the network in meantime. After capturing the network packets with Wireshark, the TCP flows were extracted by the Argus tool from the PCAP file. The first version of the dataset was made by extracting 44 flow-level features. Then, we labeled the dataset based on binary and multi-value classification. In binary classification we classify malicious and legitimate flows, however in multi-value classification, our classifiers can predict each type of DoS attack and legitimate flow. After cleaning and labeling the first version of dataset, the Full-Featured dataset was generated with 34 flow-level features. Then, we performed the ANOVA features selection method to have more accurate and comprehensive machine learning models. By applying ANOVA, we obtained 10 best features to make the 10 Best-Features dataset. Afterward, Random Forest, K-nearest neighbor, and Support Vector Machine algorithms were trained with the Full-Featured and 10-Best Features dataset to achieve the offline and online machine learning models. After evaluating the classifiers on the entire dataset as an offline machine learning, we tested the online classifiers on different flow durations, to notice the trade-off between flow durations and classifiers' accuracy. Then, we evaluated classifiers performance by using Confusion Matrix and Learning curves.
The results obtained indicate that our online classifiers can detect and classify

DoS attacks in binary and multi-value cases within different flow durations, however, by increasing the flow durations, we achieve more accurate results. After evaluating the performances of all algorithms we have proposed:

- **Binary Classifier**, we have chosen SVM as the best classifier which can detect and classify in 5 seconds flow duration with a 0.74 F1-score and in 60 seconds flow duration with a 0.84 F1-score.
- **Multi-value Classifier**, In multi-value classification, depends on the application, if higher accuracy is required, we suggest the RF classifier which is able to classify attacks with a 0.91 F1-score at 60 seconds flow duration. On the other hand, in the case of delay-sensitive applications, our proposed RF classifier can detect an attack in 5 seconds flow duration with a 0.67 F1-score.

7.2 Future works

In this section, we explain the possible extension of this thesis work.

- In this project, we considered smart home scenario for detecting the DoS attacks, however, automated transportation, smart energy management systems, smart cities can be taken into account.
- Another possible extension of this work could be the diversity of attacks. In this project, we detect and classify 3 types of DoS attacks, however, Phishing or Man in the middle attacks could be considered as future works.
- In this thesis work, we trained RF, KNN, and SVM as classification algorithms; however, applying Deep learning algorithms on our proposed datasets can be taken into account as an extension of this work.
- In online classification, we detect the DoS attack in different flow durations; however, another online classifier could be added to classify attacks after receiving different number of packets in each flow.
- Other future works could be the reaction of the broker after detecting the malicious attack. For instance, when the attacker is detected, the broker can deactivate it.

Bibliography

[1] I. Vaccari, M. Aiello, and E. Cambiaso, "Slowtt: A slow denial of service against iot networks," *Information*, 2020.

[2] "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 @ONLINE." `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`.

[3] L. Hongyu and L. Bo, "Machine learning and deep learning methods for intrusion detection systems: A survey," 2019.

[4] I. Vaccari, C. Giovanni, A. Maurizio, M. Maurizio, and C. Enrico, "Mqttset, a new dataset for machine learning techniques on mqtt," pp. 5–13, 2020.

[5] N. F. Syed, Z. Baig, I. Ahmed, and C. Valli, "Denial of service attack detection through machine learning for the iot," *Journal of Information and Telecommunication*, pp. 482–503, 2020.

[6] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," pp. 29–35, 2018.

[7] N. Koroniotis, N. S. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics:bot-iot dataset," *Future Generation Computer Systems*, p. 779–796, 2019.

[8] N. Moustafa and S. Jill, "a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," *Military Communications and Information Systems Conference (MilCIS)*, 2015.

[9] K. Bobrovnikova, S. Lysenko, and P. Gaj, "Technique for iot cyberattacks detection based on dns traffic analysis," *CERU*, 2020.

[10] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, pp. 12–22, 2018.

[11] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton-iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, pp. 165130–165150, 2020.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kddcup99 data set," pp. 1–6, 2009.

[13] N. F. Syed, Z. Baig, I. Ahmed, and C. Valli, "Modelling and evaluation of malicious attacks against the iot mqtt protocol," *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber*, pp. 748–755, 2017.

[14] "Mosquitto website @ONLINE." `https://mosquitto.org/`.

[15] D. Murat, O. Yunus, and B. Cevat, "Creating scenarios and using for smart systems generated with wireless sensor networks," *World Conference on Technology, Innovation and Entrepreneurship*, 2015.

[16] "wireshark website @ONLINE." `https://www.wireshark.org/`.

[17] "How to use wireshark @ONLINE." `https://www.varonis.com/blog/how-to-use-wireshark/`.

[18] "qosient website @ONLINE." `https://www.qosient.com/argus/gettingstarted.shtml`.

[19] "argus website @ONLINE." `https://openargus.org/`.

[20] "One hot encode sequence data in python @ONLINE." `https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/`.

[21] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[22] "Anova @ONLINE." `https://machinelearningmastery.com/feature-selection-with-numerical-input-data/`.

[23] "Feature selection anova @ONLINE." https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476.

[24] "Rf structure image - medium website@ONLINE." https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f.

[25] "Knn image - towords data science @ONLINE." https://shorturl.at/kyBVW.

[26] O. Manjrekar and M. Duduković, "Identification of flow regime in a bubble column reactor with a combination of optical probe data and machine learning technique," *Chemical Engineering Science*, 2019.

[27] "Plotting learning curves @ONLINE." https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html.

[28] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, vol. 7, pp. 127276–127289, 2019.

[29] H. Lin, X. Xu, and J. Zhao, "Dynamic service migration in ultra-dense multi-access edge computing network for high-mobility scenarios," *Wireless Com Network 2020*, 2020.