UCLouvain

École polytechnique de Louvain

Hey IoT devices! Is anyone there?

Presence detection through network monitoring of IoT devices

Author: **Constantin van Ypersele de Strihou**
Supervisor: **Ramin Sadre**
Readers: **Ramin Sadre, Siegfried Nijssen, Gorby Kabasele Ndonda**
Academic year 2022–2023
Master [120] in Computer Science and Engineering

Acknowledgement

I would like to express my sincere gratitude to everyone who has contributed to the completion of this Master thesis. Especially for my roommates and my mum, who agreed to turn their house into a smart home. Without their participation this work wouldn't have been possible.

A special thanks to my supervisor, Professor Ramin Sadre, for his help and advice throughout this year and to the readers of this master thesis.

Finally, I would like to thank the École polytechnique de Louvain for these five years of hard work. Although it hasn't been easy every day, I have learned a lot and when I look back on my academic career, I am proud of it.

Abstract

The rapid proliferation of Internet of Things (IoT) devices has revolutionized the way we interact with technology in our daily lives. More and more of these devices are infiltrating our society and particularly our homes. Innocent at first sight, they facilitate our activities. While the number of connected devices inside our house explodes, their impact on our privacy should be everyone's concern. These devices often suffer from weak security, but are they safe even with better encryption, firewall, passwords, etc? More and more studies show that we can infer information only by passively "sniffing" their wireless traffic.

In this thesis, we will focus on smart homes. We will try, based on real smart environments, to detect the absence or the presence of people inside the home, only by passively capturing unencrypted packet headers from the smart devices inside the house. We will build an approach to achieve this. Afterward, we will evaluate and analyze it.

Contents

Chapter 1

Introduction

Did you know that in 2020, the connected devices around the world outnumbered the unconnected devices? Did you know that by 2030 we expect 25.4 billion of connected devices ([1])? It's almost twice as many as today ! All these objects make up what we call the *Internet of Things* (IoT). These devices are connected to the Internet and share information around the world. They can be accessed from anywhere, at any time. The IoT market is estimated to be worth around $800 billion and is present in every sector. Nowadays, new terms such as *smart cities* or *smart homes* are appearing, referring to their connectedness. Some call it the Fourth Industrial Revolution (4IR or Industry 4.0). While most of these objects may not concern the average person, some of them are present in our daily lives. That's the case with smart homes. With the growth of IoT, smart homes are becoming more and more popular and we expect just under 500 million of smart homes worldwide by 2025 ([2]). Smart homes are homes with connected devices such as cameras, sensors or heating that can be controlled remotely via a smartphone or computer. The aim is to make our lives easier and safer... but is this really the case? What about the security of these objects? Can someone gain access to cameras inside a house? By placing a smart camera in my living room, am I improving the security of my home, or is it a a Trojan horse? These objects interact with their environment and form a bridge between the "real" world and the "online" world. In addition to the risk of leaking your private information, a malicious attacker could try to use them to learn some information about your habits. Do you think that someone would be interested in knowing when you are at home? Do you think that someone who knows that you are never at home on a Friday between 4 pm and 6 pm would use this information? As you can see, these seemingly harmless objects can be turned into real spies. So it's important to make people aware of the potential thread they represent.

Today's security standards include authentication, encryption, secure communications and more. For example the content of cameras is encrypted to prevent an attacker from seeing what the camera is seeing. Unfortunately, several studies have shown that encryption is not enough to guarantee home security ([3], [5], [6], [9], [10], [4]). Passive network observers can infer information simply by intercepting and monitoring ("sniffing") wireless packets on the home network. For example, the rate of traffic can reveal potentially sensitive user interactions ([8]). It's also possible to classify IoT devices and identify their activities ([6], [5]).

In this master thesis, we will focus ourselves on smart homes. More specifically, we will analyse how an attacker can infer presence information in a smart home by passively sniffing packet headers from smart devices. Two real smart environments will be used and,

contrary to what has been done in some related works ([3], [5]), only a few devices will be present inside. This scenario is typically representative of people with a small budget who want to modernise their home in a simple way. The aim of this work is to demonstrate the feasibility of presence detection based on the network monitoring of a few devices. The number of people in the house and their activities don't matter. The objective is to provide a simple but effective model and to analyse the lessons learned from it.
In this report, you will find in the following order: a discussion of the prerequisites and related works (Chapter 2), a description of the experimental scenarios (Chapter 3) and the approach used to solve the problem (Chapter 4)[1]. Chapter 5 and 6 provide an analysis of the results and the limitations of the approach. The work ends with a discussion on future improvements and a final conclusion.

[1]The GitHub repository of this work is available at the following link: `https://github.com/Constantinvy/Master-thesis/tree/master`

Chapter 2

Background and related works

2.1 Prerequisites

Before jumping into this work, it's a good idea to remind some basic knowledge. In the following subsections you will find a brief reminder about computer network and about the functioning of IoT devices. The explanations are a simplified version/scientific vulgarisation in order to understand the general idea without going too much into details.

2.1.1 Computer networks

The XXI century is marked by the worldwide deployment of the Internet. Internet has revolutionised all the domains, including our communications. Nowadays, you can make a live video call with anyone anywhere in the world. Although for some people Internet is an obscure witchcraft, it's in reality a world-wide network of computers, servers, smartphone, ... These devices form what we call a *computer network*. All the devices in this network communicate with each others using defined *protocols* and each protocol is associated to a *layer*. For this master thesis you will need to understand the 4 first layers of the OSI model [18], where each layer is based on the previous one (begin with number 1):

4. **Transport**: the fourth layer uses the previous layers to provide several services such as reliable communications, connection or connection-less communication, flow control, multiplexing, ... [21]The protocol data unit is called a *segment* and has again a header and a payload. The most popular protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Both are based on the IP protocol of the network layer. TCP is a reliable protocol, meaning that it ensures that all the data are correctly transmitted between the hosts. To assure this, it uses what we call *acknowledgement* segments (also abbreviated as *ack*). Acks are used to notify the sender that the data were correctly received. They don't contain data.

3. **Network**: the third layer is one of the most important layer. It ensures the communication between different nodes (computers, servers, ...) across the whole network (Internet). The protocol data unit is called the **packet**. Again, it is made of a *header* containing the meta-data useful to transfer the packet and a *payload*, containing the exchanged data. The most popular protocol of this layer is the Internet Protocol (IPv4 and IPv6 [20]). In this protocol, each host has an *ip address*. Inside a network, each device has it own unique address and it uses it to communicate with

others hosts. We find thus in the meta-data a source ip address and a destination ip address. In this master thesis, we will mainly work with this layer and we will thus handle packets.

2. **Datalink**: this second layer receives the data from the physical layer and merge it into *frame* (= protocol data unit). Inside the frame we distinguish the *header* (useful to detect transmission errors) and the *payload* (contains the data). The principal goal of this layer is to detect transmission errors and sometimes to guarantee reliable communications. A typical protocol is the Logical Link Control (LLC) [19].

1. **Physical**: it's the first layer. This layer deals with the hardware of networks and each device is physically connected to each others using electric wires, optical fibres, wireless signals, ... The protocol data unit is called the *bit*: 1 or 0. This layer is *unreliable*, meaning that during an exchange, a bit can be created or removed and its value can be modified. A typical protocol for this layer is Bluetooth.

Some of you may struggle with the concept of metadata. For a better understanding, think of the post office: the envelope contains your message (= data) and the meta-data (= header) are the street, the number, the city, ... While the data can be encrypted to ensure confidentiality, headers are never encrypted and thus everyone can read them. Using again the example of the post office, when you write a message to someone, you place your sheet of paper inside an envelop, such that no one can read your text (= confidentiality). Then you write the destination address on it such that the postman (and everyone else) can read the address. This example is similar to the encryption of your data, which ensures its confidentiality, and to the packet headers, which ensures the exchange of your data.

2.1.2 Internet of things

The Internet of Things (IoT) refers to a network of physical devices, vehicles, appliances, and other objects embedded with sensors, software, and connectivity that enables them to collect and exchange data over the Internet. These devices are sometimes referred to as *smart* devices or *connected* devices. The concept of *smart homes* refers to habitations that contain smart devices. We can distinguish different types of devices for smart homes: large and expensive devices like a connected boiler or a connected heater, and then smaller and cheaper ones like connected cameras, connected lights, ... In this paper, we will focus on the second category, which poses a greater threat to privacy. More specifically, we will use the following 6 devices:

1. **Tapo C200 smart camera**: this indoor security smart camera costs 25€ on Amazon [14]. It has a night vision, motion detection, speaker, audio sensors and is remotely controllable. It uses the wireless protocol IEEE 802.11.

2. **Woox smart plug (2)**: smart plugs are plugs that we put into standard outlets and that we can remotely turn on/off. We can then plug any device into it and transform it into a smart device. That's useful for a large set of device: from electric heaters to lamps. One smart plug costs 10€ on Amazon [15].

3. **Yueyang movement sensors (2)**: indoor movement sensors are very useful to interact with their environment. For example to turn on connected lamps when someone pass-by. You can find them for 26€ on Amazon [16].

4. **Govee smart thermometer**: you can see in live on a mobile application the temperature and the percentage of humidity. It's also possible to receive notifications when the temperature or humidity reach some threshold values. It costs 40€ on Amazon [17].

This gives us a global price of 137€. All the devices are connected to WiFi and can be control through a smartphone application. The reality of the scenario being a key point of this work, having a motion sensor in every room, with 5 cameras wouldn't have been adapted. These devices have been selected because they are cheap, present at first sight a privacy concern, and especially because they are representative of the objects that someone could buy to make their home connected with a small budget.

To describe in few sentences how these devices work, the wired devices (the camera and smart plugs) are permanently connected to the internet. They exchange data continually. The thermometer sends exactly every 10 minutes its data to an AWS server, regardless of anything (temperature, humidity, environment, ...). Finally, motion sensors connect themselves and exchange data on the internet only when movement is detected. In some rare cases, sensors may make false detections. Otherwise, they aren't connected to the Internet and disappear from the network.

2.2 Related works

This section describes the works on the same subject. There are a lot of papers on IoT devices security, IoT classification, etc. In this section we will focus ourselves on a subset of them, based on their similarity with this thesis. If two papers should be considered as a guideline for this work it should be the one presented in the RAID conference in 2021 [3]. It proposes "SniffMislead", a privacy stand-alone solution against wireless packet sniffers. The second one [5] was presented in the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks in July 2020. Both of them used a network monitoring approach in a complete smart environment, with multiple IoT devices.

2.2.1 IoT devices classification

Some would say that the first step for an attacker would be to understand what are the type of smart devices inside the target home: cameras, sensors, door lock, ... Indeed, each device interacts differently with its environment. As previously explained, each device has also its own functioning. As you can guess, the behaviour of each device follows patterns and is easily identified. Previous works have shown that we can identify them with a accuracy near 100% based on network monitoring ([5], [6], [7],[11], [13]). Different techniques were used but the general approach can be resumed as follows:

1. Collect network traffic from different devices. It can be done using pcap files for example.

2. Characterise your data using metrics such as traffic rate, payload size, port number, ... The goal is to find patterns inside your data that are representative of the devices behaviour.

3. The techniques used for the last step differ from papers but the global idea is to develop a model that classify the traffic between the devices (recurrent models are random forest, decision tree, kNN). It uses the patterns/traffic characteristic of the previous step to determine the corresponding device.

Despite its relative simplicity, this approach produced good results. To save time, this paper simplified the problem by assuming that the attacker knew the ip address of each device. These papers allow us to relativise this simplification. In fact, they have demonstrated that it's possible, and relatively easy, to identify each type of device. This can be considered as a solved problem.

2.2.2 Activities detection

Another interesting part is activities detection. By activities we mean user interactions with IoT devices. It can be inside or outside the house. For example an activity could be walking in front of a movement sensor or turning on/off a smart plug. Each type of device has its own type of activities (hence the importance of identifying the type of device). Recent works have demonstrated that activities detection and classification are possible using passive packets sniffing in highly connected environment ([3], [5]). Researchers managed to identify with a precision $> 90\%$ tasks such as leaving/returning home, bathing or falling asleep.

As example, the approach presented in [5] follows these steps:

1. Device identification: they used a technique similar as the one presented in the previous section. They used mean packet length, mean inter-arrival time, and standard deviation in packet lengths to characterise the network traffic. Then, using a kNN, they were able to classify the devices with an overall accuracy of 93 %.

2. Device state detection: they observed during activities that the network traffic changes of behaviour. They transformed packets sequences into a supervised learning dataset using a time interval. Then, they extracted features for each interval and assigned each interval to an activity (or to "no activity"). This formed a features dataset, on which any supervised learning algorithm can be applied. Finally, they managed to detect activities with a maximal F1 score of 94 %.

3. Device state classification: the goal is to determine in which state is each device. Again, features were extracted from the network traffic and machine learning models were used to build a solution that reach a mean f1 score of 94 % (this score depends on the devices and the state).

4. User activity inference: this last step must infer user activities inside the smart environment. To achieve this, different device features must be extracted from network traffic data such as timing information, sensor information, device states, location, etc.

   They also clarify which type of activities they will infer by differentiating time-independent activities (like turning on/off a smart plug) and time-dependent activities (like a user walking from a point A to a point B and passing in front of sensors).

After that, they regroup activities in 6 categories and use a Hidden Markov Model to classify them.

They were able to reach a F1 score of 100% for the time-independent activities and a F1 score of at least 91% for time-dependent activities. They justify this difference partially by the fact that a time-depend activity is made of multi-stage. In order to be count as correctly guessed, all the stage must be correctly classified, which may lower the end-to-end successful inference rate.

These results may be alarming but we must put them into perspective: the smart home environment used in this paper included Samsung SmartThings hub, Samsung multipurpose sensor, Samsung motion sensor, Netgear Arlo security camera, Philips Hue smart light, Ecobee Smart Thermostat, and August Smart Lock. For their training set they collected network traffic data from 10 different users for different user activities and for their test set they collected user activity data for a week from 15 different people (total 30 datasets). They had thus a various and complete dataset with a huge amount of different devices.

It is also the case for the other paper [3] were their set up uses more than 25 smart devices (dispatched in 3 rooms), including presence and movement sensors, bed button, door contact, etc.

Others works ([8],[10], [12]) identify activities by finding patterns in the traffic network. To achieve this, they plot the traffic rate of each device in different scenarios. The results were quite obvious. For example they plot the traffic rate of a camera. When the camera detects motion, it triggers clearly observable spikes in the network traffic. This predictable variability in network send/receive rates would allow an attacker to observe the presence and frequency of motion inside a smart home.

2.3 Distinctive characteristics

As explained in the previous sections, there are already papers on IoT classification, activity detection, etc. This work is based on the same basis as the other works (network monitoring of IoT devices), but it differs in several points and it's important to list them:

1. **Protocols**: there are many different protocols for IoT devices. The most popular are WiFi, Bluetooth, Zigbee, MQTT, Z-Wave, ... Each of these protocols has its own characteristics and its own advantages and disadvantages. In this thesis we will only use devices connected to the WiFi. The main reason for this choice is that the previously cited works have shown identical results using the different protocols. The second reason for this choice is simplicity. Simplicity of installation (you just need to connect the device to you private WiFi) and simplicity of traffic capture (using tools like Wireshark or *tcpdump*).

2. **Set up**: the cited works use a lot of devices. In particular [3] which is the most similar work. They have several smart devices in each room of the flat. You'll find more information about the experimental set up in the next section but keep in mind that the previous results were obtained in a hifhly connected environment while this one doesn't even have one device per room.

3. **Reality**: an important point of this thesis is to propose a very realistic approach. This criteria has determined the choice of devices but also the choice of environments. Two radically different environments have been chosen to test the approach (using the same devices). In comparison, the most similar work [3] proposes a single environment with a single person living in it.

Chapter 3

Experimental scenario

In this chapter provides all the information about the experimental scenario. This includes a description of the smart environments and a characterisation of the threat model.

3.1 Smart environments

In this thesis, two smart environments were used. In both cases, the people were asked to live their lives as usual, with the time of absence (= when no one is at home) reported for each day.

1. The first environment was implemented in a student collocation in Louvain-La-Neuve (figure 3.1). This collocation is on a single floor and 4 students live there. Each of them has their own lifestyle, which includes parties, working time in the living room, having dinner with guests, and other activities that young people may have. The only habit was that the students were there from Sunday night to Friday afternoon. No one was there on Saturday or Sunday morning/afternoon.

2. The second environment was implemented in an apartment (figure 3.2). In this apartment lives a single person. Some weekends her children come back. They can come for a few hours or for the whole weekend. Sometimes the person leaves the flat for 2-3 days but this happens rarely. Most of the time the person is alone. The only significant habit was a daily activity between 17:00 and 19:00.

As you can see , the same devices were used for both environments, leaving several rooms without any devices. The 2 figures are just there to represent the smart homes but don't have the same scale. The second one is much larger than the first one, so don't compare their size.

For a better understanding of the range of the camera and the motion sensors, the green square is the area that the camera sees, and the red rectangles are the areas where a motion sensor can detect a motion.

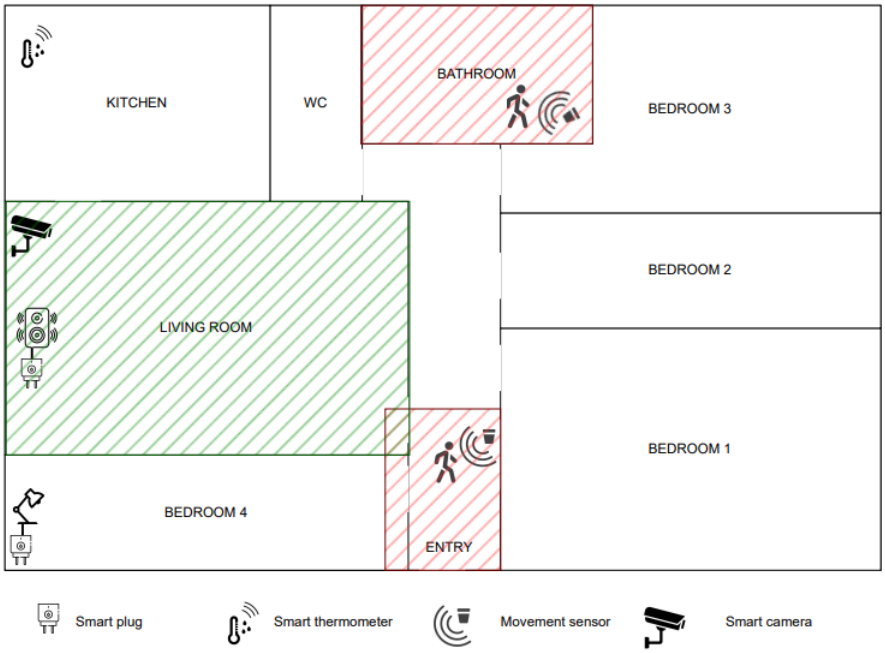

Figure 3.1: Experimental environment 1

Figure 3.2: Experimental environment 2

3.1.1 Private network

To facilitate the network monitoring, a private WiFi was established using a Raspberry Pi 4B. It was connected to the WiFi box via Ethernet and appears as an access point to the smart devices. The raspberry pi was turned into an access point using this tutorial: [22]. It was used only by the devices, and static *ip* addresses were assigned to each of them in order to identify them and prevent bugs.

Unfortunately, the WiFi emitted by the Raspberry wasn't very strong. This limited the devices to be in a range of $\sim$ 7m. This was not a problem for the first environment because it was small enough, but for the second one it forces all the devices to be close to each other, leaving a huge unconnected space.

3.2 Attacker model

For this work, several assumptions were made about the attacker capabilities:

1. The attacker is inside the network of the smart home. He can see and capture all the traffic but **only** from the smart devices on the network. It's a misuse of language to say that "the attacker is inside the network". In reality we mean that he controls a device (a compromised IoT device for example) that is inside the network. The attacker himself is not physically inside/around the house. Thus he can't see when someone enters or leaves the house, he just has access to a device inside the house that is connected to the local network.

2. The attacker will only capture and use the unencrypted packet headers. Apart from their size, no information is extracted from the encrypted payload. The capture is done using *tcpdump*. The attacker is passive, i.e he only sniffs packets without interfering with the network traffic.

3. The attacker knows which ip address corresponds to which device. This assumption is made to save time and, as explained in the previous part, related work have shown that classification of IoT devices can be achieved with almost a perfect accuracy.

4. The attacker has no additional information such as the position of the devices inside the house, the house itself, the number of people living in it, their lifestyle, ...

The attacker's goal is to determine the absence or the presence of someone in the smart home (regardless of the number of people and regardless of their activities). This threat model is relevant for two main reasons:

- **Weak security**: the attacker only needs a *spy* inside the network with whom he can communicate. He can install the sniffer only once and access it remotely or it could be a compromised device inside the smart home that becomes a sniffer. Unfortunately, IoT devices are not known for their outstanding security. So it's possible for a determined hacker to gain access to a device in your home.

- **Network protocol intrinsic properties**: in fact, this approach only uses the metadata of the different network protocols. These metadata are not encrypted and remain accessible in clear.

Chapter 4

Approach and implementation

In this chapter you will find a description of the approach and how it was implemented. Of course, the final solution was not found at the first attempt but is the result of an evolution of different approaches. Only the final solution will be explained in detail but keep in mind that the choices made were obtained by experiments. This explanation will follow a chronological order. Note that the same approach was used for both environments and that all scripts were written in Python 3.8.10.

4.1 The big picture

On the following figure 4.1 you can find a global representation of the solution. The solution has been split into 6 steps, where the output of each step is used as input for the following ones. Each section of this chapter describes one of this step. Note that for clarity reason, the splitting of the dataset into the training and testing set is not showed on the schema. Of course, this split is made such that information of the test set are never used during the different stages. The test set is made of 7 consecutive days (to form a complete week).

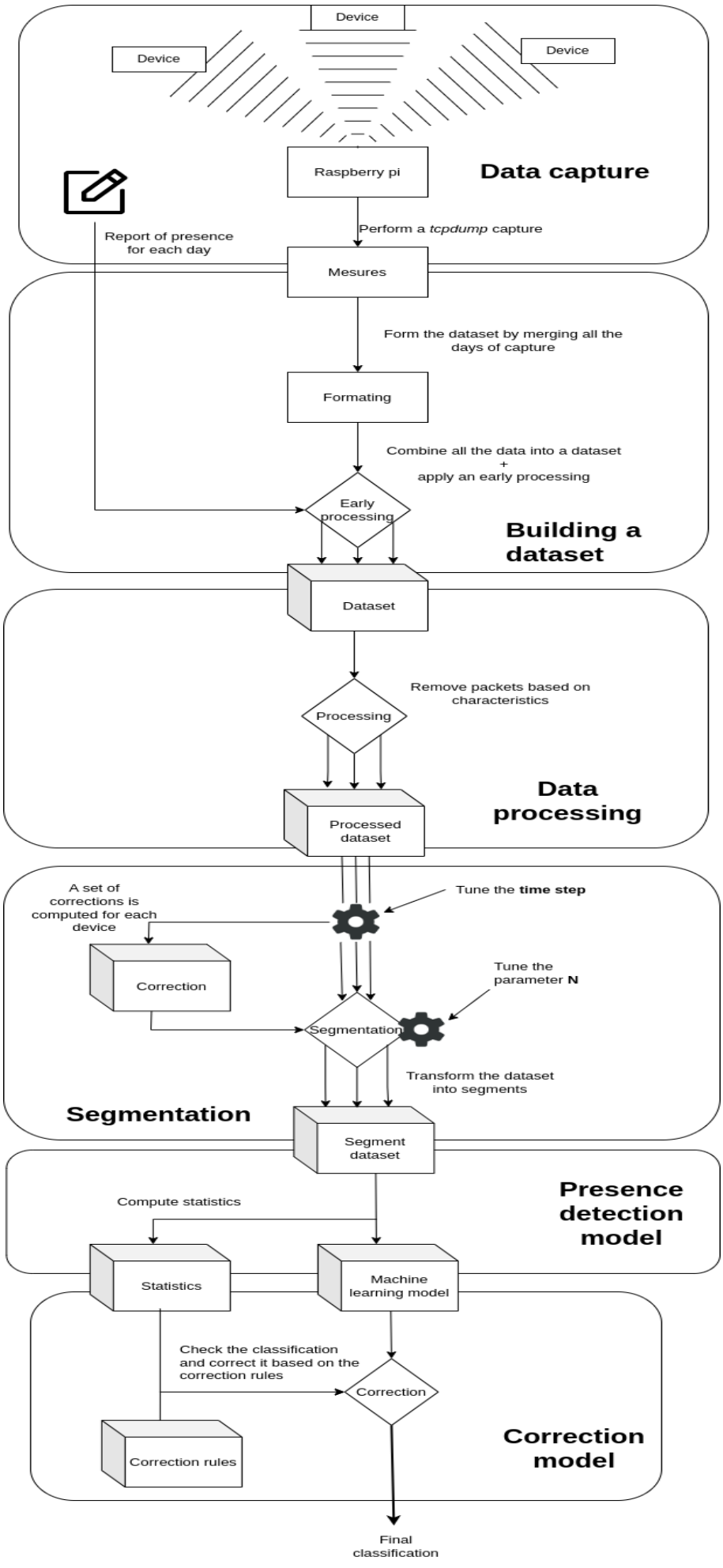

Figure 4.1: The Big Picture

4.2 Step 1: capturing data

The first step is to build a dataset. Based on the attacker model, the hacker has access to the whole traffic of the IoT devices inside the smart environment. He also knows the ip address of each device and they are static. These two hypotheses allow him to identify and capture the exchanges of the smart devices.

As you can see on the figure 4.2 below, to facilitate the capture, it was made directly on the Raspberry Pi (all the traffic of the devices goes through it). You can also observe that the presence inside the smart home was reported every day. This was the hardest part of this first step. Indeed, we had to manually report every times of absence. The times of absence have a precision up to the minute. It means that if a person arrives at 12:24:47 in an empty home, the *absence* time ends at 12:23:59 and the *presence* time begins at 12:24:00. This task was a real challenge because the different inhabitants had to coordinate to note their arriving/leaving time. These daily reports contain also small description of the activities of the day.

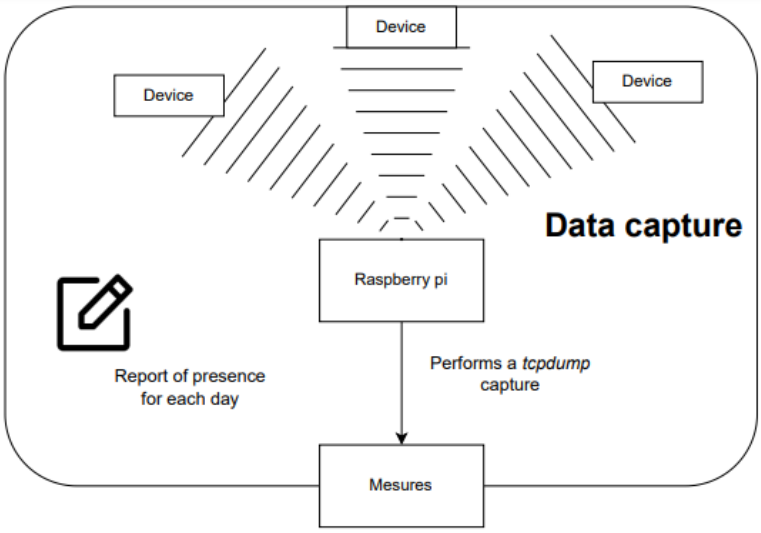

Figure 4.2: Data capture model

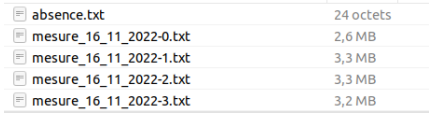

Figure 4.3: Files produced for a day of capture

4.2.1 Tools

To capture the traffic, several tools were available such as *tcpdump*, wireshark and others network monitoring tools. With the concern to propose a simple and easily reproducible approach, *tcpdump* has been chosen. It is a command line tool [23] that allows to capture network traffic. The exact command used on the raspberry pi was

sudo -i tcpdump -s 96 -n -i wlan0 host

where *host* is the list of the ip addresses of the 6 devices. This command was used in a python script that captures 24 hours a day, 7 days a week the traffic. We can retrieve information such as the time of the capture, the source and destination ip addresses, flags, sequence number, the size of the payload, ...

The measures were made using a script. To avoid problems on the Raspberry, such as too large files, the script runs the above command every 6 hours and captures all traffic. The daily measurements were stored in 4 *txt* files in the associated directory. Figure 4.3 shows the files generated for the capture of the 16^{th} November. A clever observer will already have an idea during which part of the day there was the less activity... (hint: look at the size of the files).

4.2.2 Network stability

Even at this stage, external factors influence the measurements. The stability of the network also plays a role. Network stability means

- **The internet access**: between the Raspberry and the Internet Service Provider (ISP). For inexplicable reasons, after setting up the private WiFi inside the environments, problems occurred. VOO was the ISP in the first environment and Proximus in the second one. In both environments we had blackouts of $\sim$ 10 minutes during the which we lost our internet connection. These events were unpredictable but luckily not too frequent. During a blackout it was as if the internet box had been switched off, so neither the smart devices connected to the Raspberry nor the other users on the main WiFi had access to Internet.
- **The state of the devices**: sometimes one device had problems. That can be a motion sensors with empty batteries or a device that had connection problems. For example, in the second environment, one of the smart plug and the camera struggled sometimes to access the private WiFi. These internet problems were due to the concrete walls that attenuated the signals. Luckily, theses problems were rare.

A few days were unusable due to the above points, but for each "human" error, measures were taken to prevent it from happening again. For the other problems, such as Internet blackouts, all we could do was report them. Unfortunately, it's possible that the blackout happened unnoticed and was therefore not reported.

4.3 Step 2: building a first dataset

The next step was to build the dataset. From now on, all the steps are performed on a computer and the Raspberry isn't involved anymore.

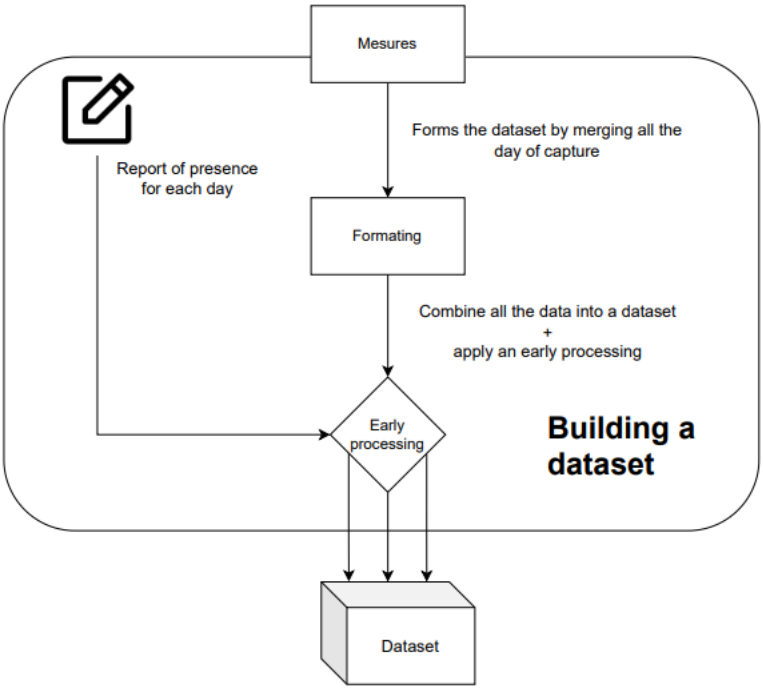

Figure 4.4: Building a data set

So after retrieving the captured data from the Raspberry, the generated files for each day of capture (figure 4.3) are merged into a single file. This is the *formating* step. After that, a first processing is applied:

- The Address Resolution Protocol (ARP) entries are removed. ARP is used as a communication protocol between the datalink layer (using MAC addresses) and the network layer (using the IP protocol) within the local network. These entries have the following structure:

  00:00:19.482238 ARP, Request who-has 192.168.4.5 tell 192.168.4.1, length 28

  00:00:19.485112 ARP, Reply 192.168.4.5 is-at 1c:61:b4:e9:49:b6, length 28

- The packets with as destination ip address 255.255.255.255 are dropped (**but not for the motion sensors**)! In the Internet Protocol standards, this address is used as a broadcast address, meaning that the packets are sent to all of the hosts on the local network. These entries are, for example:

  13:34:23.264709 IP 192.168.4.19.49154 > 255.255.255.255.6666: UDP, length 174

  13:34:23.919791 IP 192.168.4.13.63333 > 255.255.255.255.6667: UDP, length 172

The purpose of this early processing is to remove *noise* (= irrelevant data) from the data. These entries are considered as noise because they are not correlated to any external behaviour, except for the motion sensors ! In fact, they connect themselves to the Internet when they detect a motion. Thus when they broadcast packets to the ip address

255.255.255.255, it means that a movement has recently been detected. For the others devices, permanently connected to internet, these entries don't provide any information. They just show how computer protocols work.

After this processing, the presence information is added to form the output dataset. It's a csv file where each entry has 10 features:

1. **Date**: represents the full date of the capture.
2. **Week_day**: represents the day of the week of the capture (from 0 to 6).
3. **Protocol**: the protocol of the capture (IP or UDP).
4. **Source_ip**: the source ip address.
5. **Source_port**: the source port.
6. **Dst_ip**: the destination ip address.
7. **Dst_port**: the destination port.
8. **Length**: the size of the payload.
9. **Info**: contains the additional information of the *tcpdump* capture.
10. **Presence**: the binary target feature representing the presence (1) or the absence (0) of at least one person at the time of capture.

4.4 Step 3: data processing

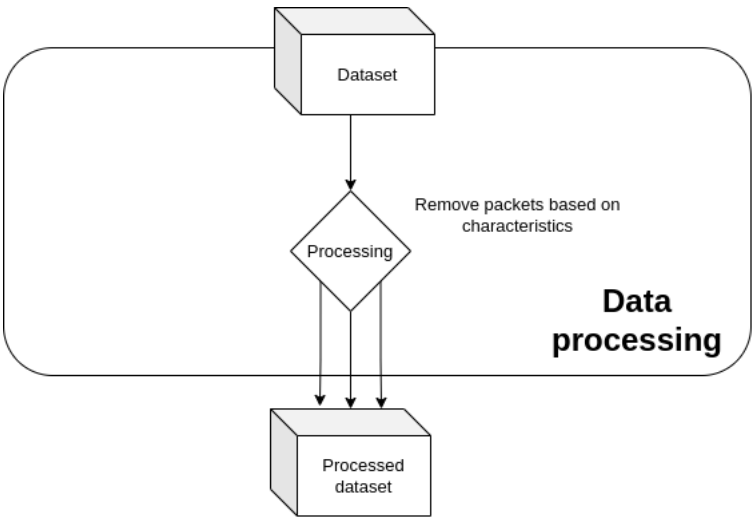

Figure 4.5: Processing of the data

In the processing step, we will clean the data so that it is meaningful and with as little noise as possible. Data are meaningful when they reflect an interaction with the environment. For example, packets sent by a motion sensor are always meaningful because we know that this type of device only connects itself to the Internet when it detects motion. So

far, we have already discarded the ARP and broadcast packets. We distinguish 3 types of packets that might be interesting to analyse:

- **Packets not sent by the smart device**: indeed, if we want to reduce the noise, it can be interesting to keep only the packets emitted by the devices. This will avoid keeping unwanted packets sent from the cloud to the devices. But we have to take into account the type of the device! The smart plugs receive meaningful information from the cloud: when a remote user turns it on/off.
- **Acknowledgement packets**: acknowledgement packets only confirm the successful exchange of data but do no carry it. So they don't contain any useful information.
- **Packets with a length of 0**: theses packets are either acknowledgements or Internet Group Management Protocol (IGMP) messages or Domain Name System (DNS) query/response:

... igmp v2 report 224.0.0.1 ...

... 39444+ A? m2.tuyaeu.com. (31) ..

39444 3/0/0 A 3.66.126.37,[/domain] ..

Filtering rules have been written to purify the data from these type of packets. After applying these rules, we have a *processed dataset* that will be passed to the next crucial step.

4.5 Step 4: the segmentation

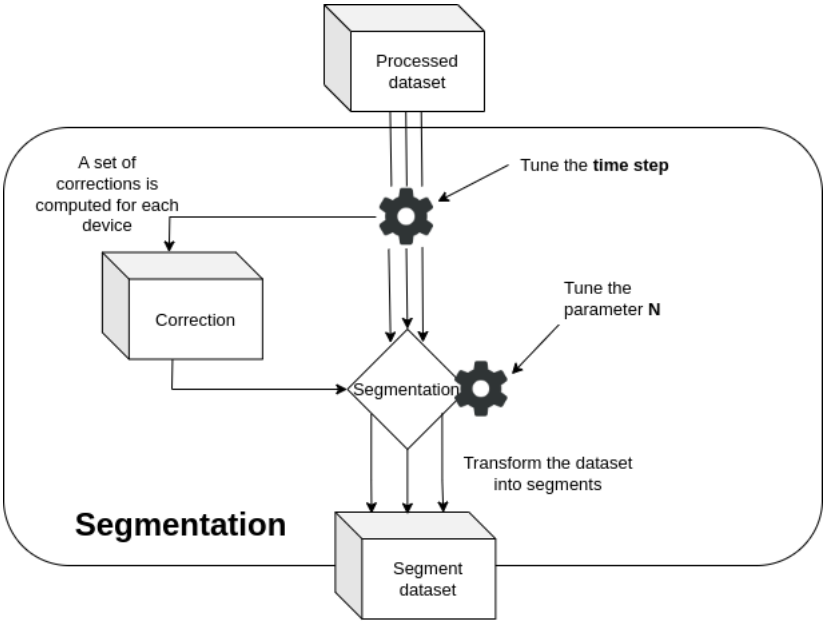

Figure 4.6: Grouping of the data into *segments*

The segmentation step is the last step that modifies the data. The goal is to regroup all the packets within the same time interval together. This group of packets is called

a *segment*. To do this, we define a **time step** parameter (in minutes). We start by computing a set of correction for each device, for the given time step. Then we regroup all the data within the same time interval into a segment. For each segment, we will basically report 2 information per device: the number of packets exchanged and their average size (in byte). These information will be corrected with the correction. Finally, the parameter $\boldsymbol{N}$ introduces redundancy in the dataset to make the temporal dependency between segments more explicit. You will find in each of the subsection a more detailed explanation. The *overview* subsection summarises this crucial step.

4.5.1 Correction

The objective of the correction set is to compute measurements of the data during periods of absence. In fact, we don't yet have any information on the number of packets exchanged by the devices. These measures will help us to better understand their behaviour. We will use a subset of absence times in the training set. This subset consists of full days (24 hours) when nobody is in the smart home. For example, with a time step of 10 minutes, a subset of absence made of 4 days and a processed dataset obtained with the 3 processing rules (see section 4.4), it produces:

$$192.168.4.5 : [100.0, 38.521, 37.5, 1842.191, 1750.625]$$

$$192.168.4.13 : [4.167, 8.167, 8.25, 841.667, 856.0]$$

Where

- The key is the ip address of the device. Here '192.168.4.5' is the camera and '192.168.4.13' is the motion sensor in the entry.
- The first value is the percentage of interval in which we retrieve packets of the device. Here we see that the camera is active during every time step of 10 minutes, even when no motion was detected. Surprisingly, the motion sensor sends data during 4.167 % of the total number of intervals. This is most likely due to wrong detection. Indeed, it happens (rarely) that I received notification of movement while no one was in the smart home.
- The second and third values are the mean and median number of packets per time step during activity. The camera exchanges an average of 38.521 packets per 10 minutes and the sensor 8.167 packets. To avoid confusion, this last value is computed during the intervals in which the motion sensors falsely detect motion.
- Finally, the last 2 values are the mean and median size of the packets sent during activity. For the camera we have a median size of 1750.625 bytes and for the sensor 856 bytes.

The idea of this correction set is to calculate the characteristics of the exchanged data during the period of inactivity. This gives us an idea of the intrinsic behaviour of the objects during the period of inactivity. These data can be seen as noise and thus this process allows us to characterise this noise. Later on, we will use these information to remove the noise from the data: we will remove the average number of packets and the average size from each segment. The correction step can be seen as a sort of normalisation.

However, an exception is made for the motion sensors. In fact, they are the only devices that only send data when there is activity. When they're not active, they may make a few false detections, but for the most part they don't send any packets. It would be incorrect to remove from all segments the mean number of packets produced by a rare, "fake" activity.

An important point to note here is that, thanks to the percentage value, we can build an automated approach where the correction is applied only to devices that send packets during absence times with a percentage greater than a threshold. **This would allow us to build an approach without knowing the type of each device !**

Let's illustrate this with the following example:

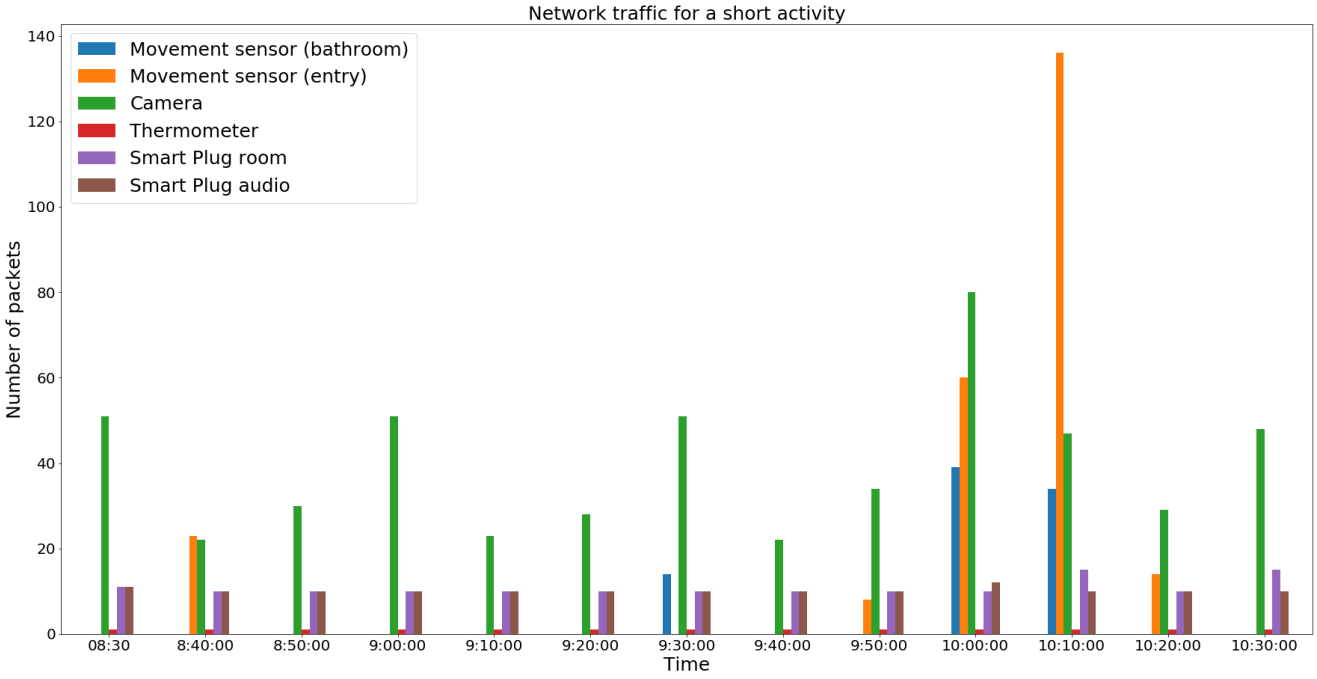

Figure 4.7: Number of packets using a time step of 10 minutes, without correction

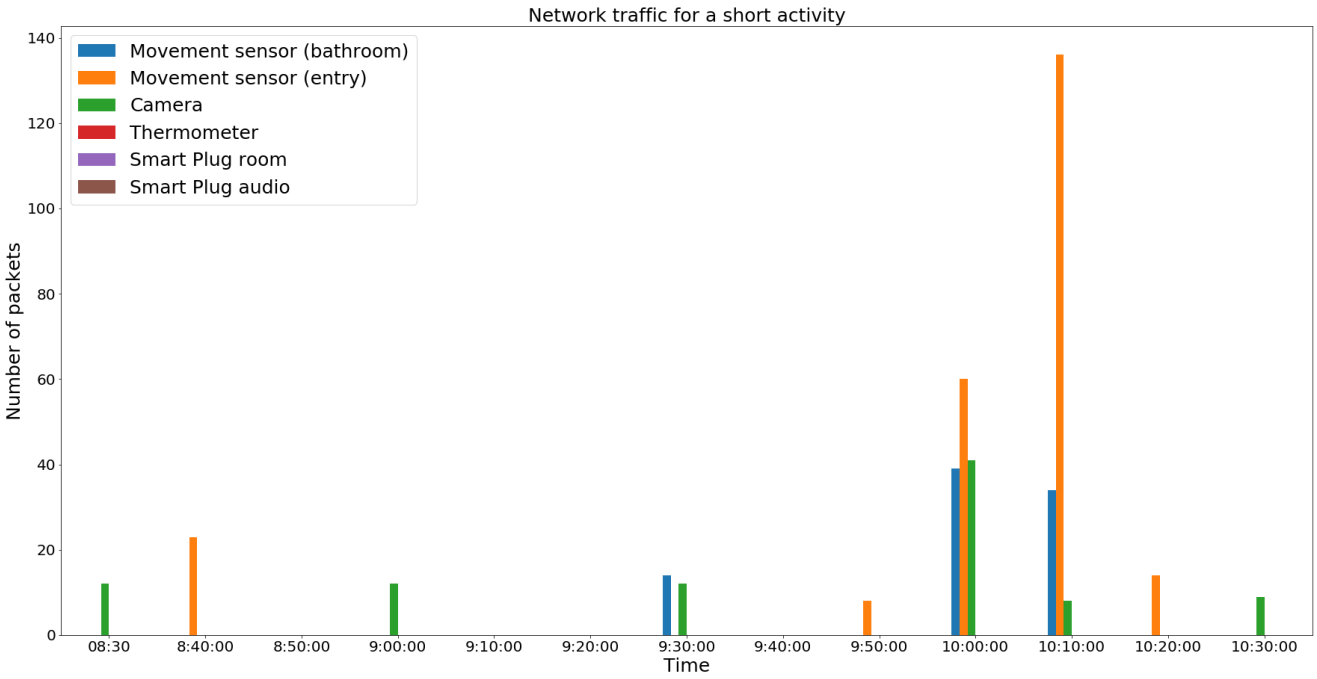

Figure 4.8: Number of packets using a time step of 10 minutes, with correction

To contextualise the figures above, they were captured in the first smart environment (figure 3.1) on a Thursday[1]. Two students were asleep. The first one wakes up around 08:40 and then leaves the house. The second student, who was sleeping in the bedroom 4 (in front of the entrance sensor), woke up a little bit before 10:00, took a shower in the bathroom and then had breakfast in the living room. He left the house at 10:20 am.

The figures show the number of packets sent by the devices for each time step. In the first figure 4.7, we can see patterns such as the (almost) constant number of packets for the smart plugs or for the thermometer. We also observe peaks for the sensors and the camera at 08:40 and between 10:00 and 10:20. If we just look at this figure, it's hard to deduce the scenario above. We could imagine that someone was working in the living room and then someone woke up or arrived. This would explain the peaks of data for the camera and for the sensor in the entry. Without correction, it's difficult to distinguish when the devices interact with their environment (when they detect a movement, when the user interacts with the device, ...). The data contains noise, this noise is all the intrinsic data sent by the devices, even in absence of activity.

In the second figure 4.8, the plot represents the number of packets per time step for each device, after being corrected. The average number of packets per time step has been removed for each device expect for the motion sensors! The smart plugs and the thermometer are no longer present. Activity is clearly observable at 08:40 and between 10:00 and 10:20. We can easily see that it involves the two motion sensors and the camera. The data are cleared from the noise of the devices and are more meaningful. The small peaks

[1]I recommend the reader to keep in mind the position of each devices inside the first environment

at 08:30, 09:00, 09:30 and 10:30 don't have real explanation. The student was still asleep and/or no one was there. For the bathroom sensor, it is likely a wrong motion detection. For the camera we see that every 30 minutes we have a peak (even if we look later during the day) so it's likely a normal behaviour (maybe the camera was communicating some internal states / periodical report with the cloud).

A small parenthesis to the attention of the reader. According to the attacker model, we don't know the position of the device inside the smart home. I let you as exercise to imagine the consequences of an attacker that would know their positions ...

4.5.2 Segmentation

The second goal of this step is the *segmentation*. We will transform our processed dataset and we will group the information of all the packets within the same time interval (defined by the *time step*) into a *segment*. A *segment* can be seen as a summary of the data exchanged during that time step. Among other information, it counts the number of packets exchanged for each device, as well as their average size (in bytes). These segments will form the final dataset: the *segment dataset*. It will then be used to train a machine learning model, so it's important to build a reliable dataset that provides enough information.

The problem of labelling

An important point is the classification of the segment. In fact, each entry in the processed dataset has the binary target feature $'Presence'$ where a value of 1 indicates that at least one person was in the smart home at that time, and a value of 0 otherwise [2]. When regrouping the packets per time step, it may happen that we have a transition of the $'Presence$ value: the last person inside the house lives. This means that inside a single segment, we have packets belonging to both binary classes. The $'Presence'$ feature that will be assigned to the segment is determined by the following rule:

$$\text{Presence } s_x = \begin{cases} 1, & (\sum_{j=\text{start time}}^{\text{start time}+\text{time step}-1} p_{t_j} = 1) \geq \frac{\text{time step}}{2} \\ 0, & \text{otherwise} \end{cases}$$

Where s_x is the x-th segment and p_{t_j} is the presence value at minute t_j. The -1 in the sum comes from the fact that each segment lasts *time step* minutes. Since we begin at *start time*, the interval ends at *start + time step -1*.
In other words, the segment x has a $'Presence'$ value of 1 if at least half the time there was someone in the smart home and a value of 0 otherwise. This implies that some minutes will be assigned to the wrong class. The worst scenario being the case where someone was present during exactly $\frac{\text{time step}}{2}$ minutes: the segment is "half mislabelled".

This segmentation has two consequences. The first is the introduction of an error e corresponding to the percentage of incorrectly labelled minutes. This error can be computed using the formula:

[2] As reminder, the absence/presence times have a precision up to the minutes. It means that if a person arrives at 08:05:52 in an empty home, the *absence* time ends at 08:04:59 and the presence time begins at 08:05:00.

$$e = \frac{\sum \text{minutes wrongly labelled}}{\sum \text{minutes}}$$

And the second one is the fact that every absence in a segment, with a duration smaller than $\frac{time\ step}{2}$, will be ignored.

The parameter N

As you can see on the schema 4.6, the segmentation introduces a parameter N. This parameter has a positive integer value and influences the building of the *segment dataset*. It represents the number of previous segments that we add as features for the current segment. So if $N = 3$, it means that each entry in the *segment dataset* will contain information about the current segment and also the information contained in the 3 previous segments.

The segments in the dataset are sorted by date so if segment x starts the 15^{th} December at 10:10 with a time step of 5 minutes, it ends at 10:14:59. If $N = 2$, the information from the segments starting at 10:00 and 10:05 will be added. The idea behind the parameter N is to provide extra information about the past of segment x. It can be seen as the size of the history of the current segment [3].

4.5.3 Overview

To summarise this 4^{th} step, we start by characterising the noise of each device, for a given time step. The noise characteristics for each device are contained in the correction set. Then we regroup all the data within the same time step into a segment. For each segment we will basically report 2 information per device: the number of packets exchanged and their average size (in byte). Then we will use the correction set to remove the noise inside these information. Finally, we introduce the parameter N that can be seen as the history size: we will join to the current segment, the information of the N previous segments. The final dataset has thus the following features:

- **Date, Hour, Minutes, Day_part, Week_day**: these are temporal information that uniquely identify each segment. The *Date* feature is not used for the machine learning model but only for graphical representation. Note the new *Day_part* feature that has been added. It splits a day into 4 parts: the night (from 00:00 to 07:59), the morning (from 08:00 to 13:59), the afternoon (from 14:00 to 18:59) and the evening (from 19:00 to 23:59).

- **Presence**: the binary target feature. The segment is assigned to one of the two classes: absence (0) or presence (1). As explained previously, this introduces a small error as the whole set of entries inside the segments are assigned to a single class. In reality, it's possible that both classes were present inside the segment.

- **Number of packets per device ***: the segment has a feature per device that counts the number of packets exchanged during the time interval. Except for motion

[3] For readers with some knowledge in Natural Language Processing (NLP), this approach can be compared to the N-gram approach where the N determines the size of the context that the model considers

sensors, we subtract from this count the correction, with a minimum value of 0. The correction consists of the average number of packets computed during periods of absence.

- **Mean size of the packet sent per device ***: the second information computed for each device is the mean size of the packet sent. Again, we subtract for all devices, except for motion sensors, the correction (or 0 if it produces a negative value). Here, the correction is the mean size of the packet computed during absence periods.
- **Depending on** N: we will copy/paste the information of the N previous segments as features for the current ones. The information that we took back have an *. If $N = 0$, then the dataset consists only of the above features.

4.6 Step 5: presence detection model

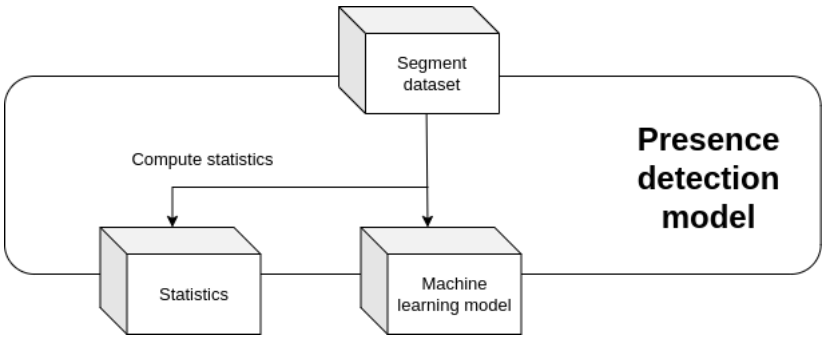

Figure 4.9: The use of machine learning to classify presence

After four previous steps spent on collecting, processing and rearranging the data, we are finally ready to achieve our goal: detecting the absence or the presence of people inside the smart environment. To reach our objective, we will split it in two parts: the first part (this step) will build a classifying model but with the accent set on the presence detection. We will also compute presence statistics for each days of the week. In the second part (next step), we will use these statistics to correct the prediction made by our model. You may be wondering why we're doing this, and that's what the next section will explain.

4.6.1 The challenge

Before we get to the model, it's important to understand the real problem and what we're working with. In this thesis, the goal is to determine whether someone is in the smart home or not. To do this, we will analyse the traffic of the IoT devices in the environment. In case of presence, we expect to observe some irregularities in the traffic, some changes (as in figure 4.8). These irregularities reflect the interactions between people and devices: someone passes in front of a motion sensor, someone turns on their smart plug, the camera detects a movement, ... But that's just an expectation! Users are not *forced* to interact with the devices.

The real challenge here will be to distinguish the absence of interaction (=inactivity) from the absence of people! Unlike the previous works 2.2, the environments have many

rooms without any devices, leaving a huge "blind" space. At night, for example, the absence of any irregularity could be due to people sleeping or the house being empty... Who knows? That's why we're going to focus first on activity detection. Then, in a second step, we will calculate statistics to distinguish inactivity from absence.

4.6.2 Machine learning model

With the challenge in mind, different models were tested with more or less success: decision trees and their variants, logistic regression classifier, k-Nearest Neighbors (kNN), Gaussian Naive Bayes classifier and even more advanced models like Multilayer Perceptron classifier (MLP). All these models are already implemented in the python library *sklearn* so it's pretty easy to test them. The segments dataset being the result of successive processing, we can directly use it to train a model. You will retrieve in the result section more details about their performances.

4.6.3 Statistics

As previously explained, we will compute a set of statistics from the segment dataset. This set consists into a report of the daily presence for each day of the week. For each day, we compute for all time interval the total number of segments labelled as positive (= someone was there) and labelled as negative. This can be represented by a dictionary d where an entry has the format:

$$\{\text{week\_day: } \{ \text{ time interval: } [n_s, p_s] \ \}\}$$

where n_s and p_s are respectively the number of segments, for the given week day and for the given time interval, labelled as negative and as positive. For example, let's say that we use a time step of 10 minutes and only 2 Monday's on 7 someone was there between 00:30:00 and 00:39:59 , the entry will look like this:

$$\{0.0 : \{3.0 : [2, 5], ...\}\}$$

4.7 Step 6: the correction model

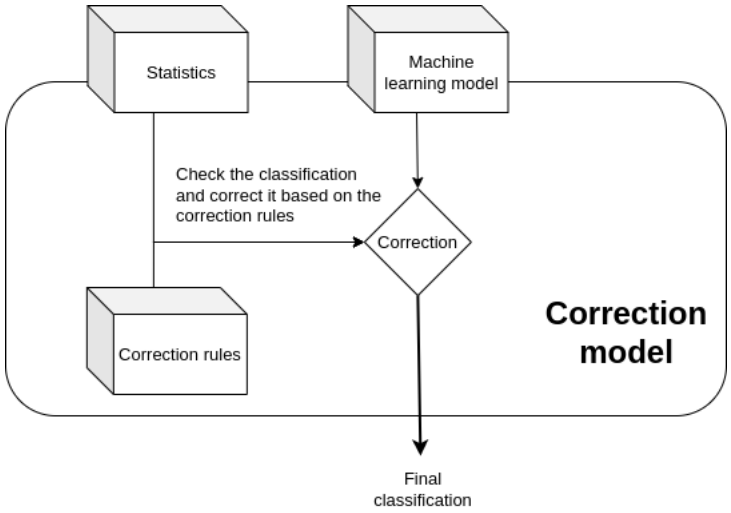

Figure 4.10: A correction is applied on the classification of the machine learning model

In the final step of the approach, we will use the trained machine learning model to classify the segments. We will then use the statistics, together with a set of pre-defined rules, to correct -or not- the classification.

If you think again about the classification challenge (see section 4.6.1), it's likely that the model will struggle to distinguish inactivity from absence for some periods of the day. One way to deal with this unavoidable fact is to use probabilities. So, if the model has trouble to clearly identify the presence/absence, we can use them to assign the most likely class for the given day of the week and the given time interval. These probabilities reflect habits of the residents. As you can imagine, this correction is very useful for correcting classification errors when there is no activity, for example during the night.

4.7.1 The corrections rules

The correction rules will determine in which cases we correct the classification... but we must respect the attacker model described in section 3.2. The correction rules must be general and we can't make specific rules for each environment. For example, in the first environment live students. So evening outings are more likely to happen than in the second environment. But we can't design rules that take this into account.

With this in mind, a correction model has been built. Its general idea is to prioritise the positive class (which is generally well recognised) and then to rely on both the pre-calculated probabilities and on the initial classification to find the most appropriate class. You can find the complete code of the correction model in the annex (section .1).

Chapter 5

Experimental results

This chapter will discuss and analyse the results of the approach on each dataset. Each dataset will be analysed in its specific section and then we will draw general comments. But before jumping to the results, it may interesting to think about the problem and its parameters.

5.1 Processing

As explained in the approach, the original data are processed several times. The first processing happens during the second step (section 4.3). It removes all ARP packets and the broadcast packets sent by all devices except the motion sensors. This early processing removes, on average for both datasets, 67.35 % of the total entries. This huge value can be explained by the fact that the devices broadcast a lot of packets to inform the network of their presence (other devices can rely on them to operate). Especially the motion sensors, which appear and disappear on the network depending on the interactions.

In the third step (section 4.4), we have defined 3 filtering rules to remove noise from the data. Applying them will drop 63.3 % of the remaining entries of the first dataset and 61 % for the second one. We observe a similar percentage for both datasets. This result is logical because they used the same devices. The table below shows the distribution of the data between the devices for each environment.

The first two rows represent the percentage of packet captured during a presence and the percentage of minutes with a presence. The sensor entry is the motion sensor in the entry (see the environments section 3.1 3.2) while the second sensor being the other one. The first smart plug is the one inside the bedroom 4 in the first environment and in the entrance in the second environment.

At this stage, we can already make some interesting observations. We can see that in both cases the camera exchanges the most data, which is not surprising as it handles images and audio. The percentage of the smart plugs may be surprising but in fact these devices exchange 3 packets per minute with an Amazon Web Service (AWS) server, probably to synchronise the state of the device (there is an on/off switch on the plugs).

Another observation is the very small percentage of the thermometer. This is due to the fact that it sends data every 10 minutes. Its behaviour is not affected by any inter-

Information	Environment 1	Environment 2
Presence (quantity)	66 %	80.36 %
Presence (time)	58.4 %	75.98 %
Sensor entry	16.41 %	12.85 %
Sensor 2	2.43 %	8.92 %
Camera	39.61 %	41.95 %
Smart plug 1	20.01 %	18.38 %
Smart plug 2	20.52 %	16.96 %
Thermometer	1.03 %	0.94 %
Monday	4	8
Tuesday	4	8
Wednesday	4	7
Thursday	5	7
Friday	5	6
Saturday	5	8
Sunday	5	8

Table 5.1: Datasets characteristics

actions. Unfortunately, the device is useless for this work, as it's impossible to deduce anything from it (but we'll take it into account anyway).

Finally, we can see that the second sensor has a higher percentage of the total data in the second environment. This is because it was in the bathroom in the first smart home. The presence there was less than in the living room of the second smart home.

5.2 Classification results

In this section you will find the results for both datasets. As you will see, we will limit the number of figures in this section to limit the size of the report. The annex at the end of the report contains more detailed results and extra figures, you are encouraged to look at them for more information.

A word about machine learning models

The goal of this work is not to find the models providing the best possible performances ever, but it still aims to have decent results. The problem with our task is that evaluation scores (recall, precision, ...) are not always representative of the real performance of the model. A graphical analysis was always needed to put them into perspective. Fortunately, the classification of a entire test week is almost instantaneous, allowing to use a greedy search approach to find the optimal parameters. In fact, some models were very constant (like the logistic regression) and their meta-parameters don't have a big impact on their performance. For others models, such as decision tree, the meta-parameters had a bigger impact. For these, their meta-parameters were found by testing several combinations and validating their results with the graphical representation. The meta-parameters chosen for each model can be found in the appendix.

Finally, it quickly becomes apparent that for some machine learning models, the time features of the dataset (Hour, Minutes, Day_part and Week_day) tend to cause some overfitting. Throughout the rapport, the term *temporal features* refers to these features. In fact, the models were unable to detect unusual absences and gave too much importance to time. On the contrary, they perform significantly better without these features.

A time step of reference

The goal is to determine for each segment whether or not at least one person was present in the smart home. The time of a segment is determined by the time step. This parameter is the most important one in the sense that everything depends on it. It also determines the sensibility of the model: a time step of 1 minute means that we expect from the model to detect changes immediately. Even if someone leaves the house during 1 minute to throw away garbage, the model should detect it. On the contrary, if we use a time step of 30 minutes, it means that every absence shorter than 15 minutes will not be considered[1].

Another important point in relation to the time step value is the number of incorrectly labelled minutes for all segments. With big values, small absences will be ignored, thus increasing the error. With a value of 1 minute, each segment regroups all the packets within the same minute and so there are no errors at all !

For the two reasons above, a reference value of 10 minutes has been chosen for the time step parameter. Models have been selected based on their performances using this time step but the approach can very well be used with another reference value.

5.2.1 First environment

For this dataset, using a time step of 10 minutes, 278 minutes (0.603%) were incorrectly labelled. Several machine learning models perform quite well, but we will only analyse the 3 best ones: the logistic regression, the MultiLayer Perceptron (MLP) and the decision tree. In the table below, you will find the best mean precision and mean f1 score using a 3-fold cross-validation (the test set is always a week of 7 consecutive days) [2]:

Model	*N*	Accuracy	Recall	Precision	F1 score
Logistic regression	0	80.44%	72.77%	**94.38%**	82.16%
	3	86.25%	85.92%	91.43%	**88.58%**
Decision tree	0	89.84%	90.5%	**92.87%**	91.62%
	2	91.69%	94.43%	92.33%	**93.37%**
MultiLayer Perceptron	1	90.57%	92.34%	**92.47%**	92.39%
	4	90.82%	93.88%	91.49%	**92.66%**

Table 5.2: Comparison of machine learning models using a time step of 10 minutes

At first sight, all the classifiers perform quite well. The logistic regression doesn't use the temporal features. So far, these results are just numbers. We need to put them into

[1] Remember how a segment is labelled (see section 4.5.2)

[2] For more detailed results see *Result annex*

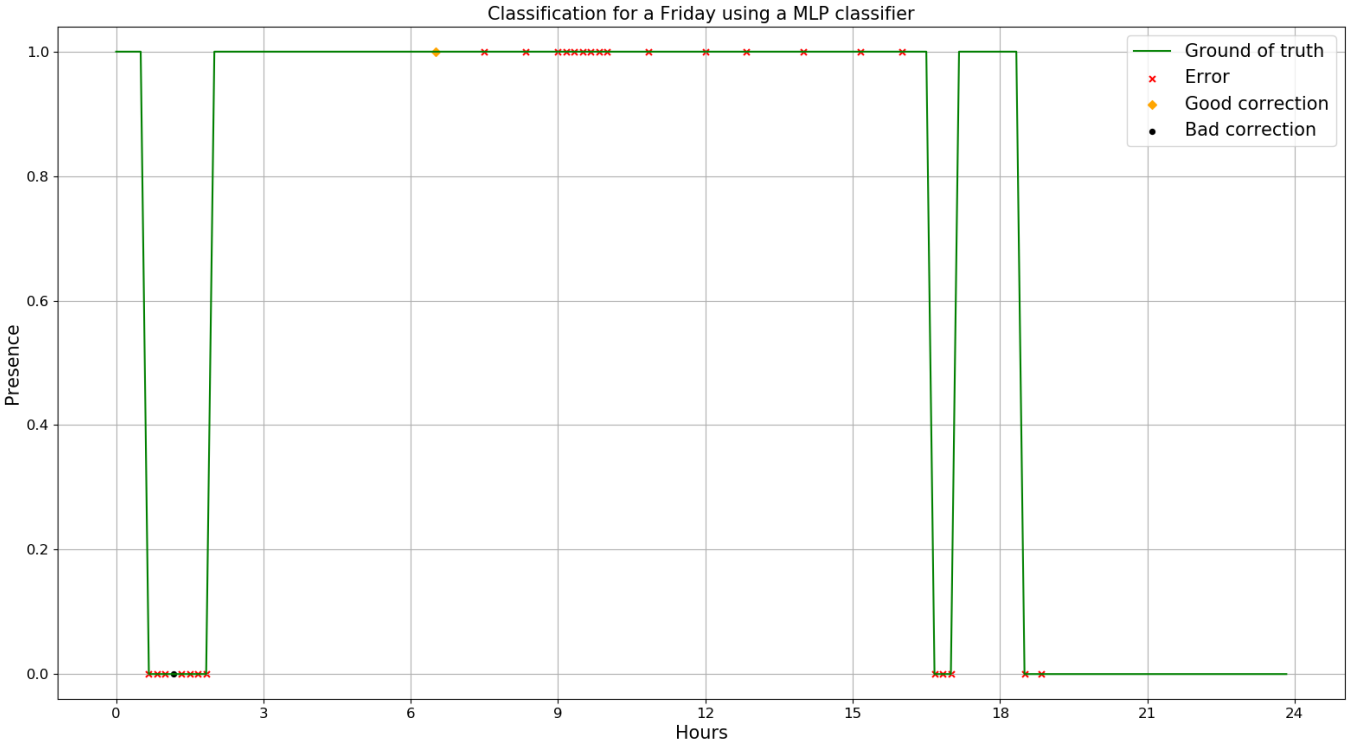

Figure 5.1: MLP classifier with a time step of 10 minutes and $N = 4$

perspective and show their performance graphically. The analysis will be done for the MLP classifier, but it's the same principle for the other models.

Figure 5.1 represents a Friday. When the green line has a value of 1, it means that at least one person was in the smart home. When it has a value of 0, it means that no one was present. Each red cross is a classification error. An orange diamond is an initially incorrect classification that is correctly corrected by the correction model (e.g at 04:10). The black dots are the initially well classified segments that were incorrectly corrected by the correction model. For this graph the time step equals 10 minutes, meaning that every symbol corresponds to the classification of a segment of 10 minutes.

As we can see, a first absence between 00:30 and 01:50 was not detected. This is explained by the fact that this absence was an unusual outing. This explains why the correction model didn't correct it and also why it incorrectly corrected a segment at 01:00.

Another interesting observation, is the group of errors between 09:00 and 11:00. According to the presence report, only 2 students were asleep this morning. One student, which we will call student A, woke up around 07:40 and left at 08:20. The other student slept until 11:00. Student A came back from its lesson around 10:05 and then worked in his bedroom without leaving it. After his return, the model successfully identifies his presence inside the house. This observation is very interesting because it illustrates well the challenge of presence detection: when there is movement, activity, ... the classification is "easy", but when people are at home but don't interact with the devices, the detection becomes much more difficult.

The errors in the afternoon are probably due to the fact that there was only one student in the smart home at that time. He was working in his bedroom without moving a lot in the house, so the model misclassified some segments. The errors between 16:40 and 17:00 are related to the parameters N and will be discussed in the section 5.3.2.

To finish with a positive point, the model successfully classified the period between 17:00 and 18:10, despite the fact that only one person was in the house, working in his room (without moving in the house). If you are interested by seeing how the two others classifier performs for this day, you can find their schema in the result annex (figures 1, 2).

5.2.2 Second environment

The same approach was used for the second environment. Again, a time step of 10 minutes was chosen and 489 minutes (0.654%) were mislabelled. The same machine learning models were tested as for the first dataset. The logistic regression classifier, which performed well in the first environment, don't perform very well on this dataset. The table below shows the 2 best models with their mean best precision and F1 score using a 6-fold cross-validation:

Model	N	**Accuracy**	**Recall**	**Precision**	**F1 score**
MultiLayer Perceptron	5	81.15%	94.28%	**82.9%**	87.63%
	6	81.25%	94.67%	82.78%	**87.74%**
Decision Tree	4	83.91%	93.7%	85.71%	89.06%

Table 5.3: Comparison of machine learning models using a time step of 10 minutes for the second dataset

None of the models use the 'Week_day' feature. In fact, unlike in the first environment where this feature was useful to identify habits (such as Saturdays without anyone), in the second environment it had a negative impact. Indeed, the person living in the second smart home had a more "unpredictable" lifestyle. The use of this feature tended to rely too much on the day of the week, when in fact it provided no reliable information.

When looking at the results, they are slightly less good than those of the first environment. Again, we need to put them into perspective:

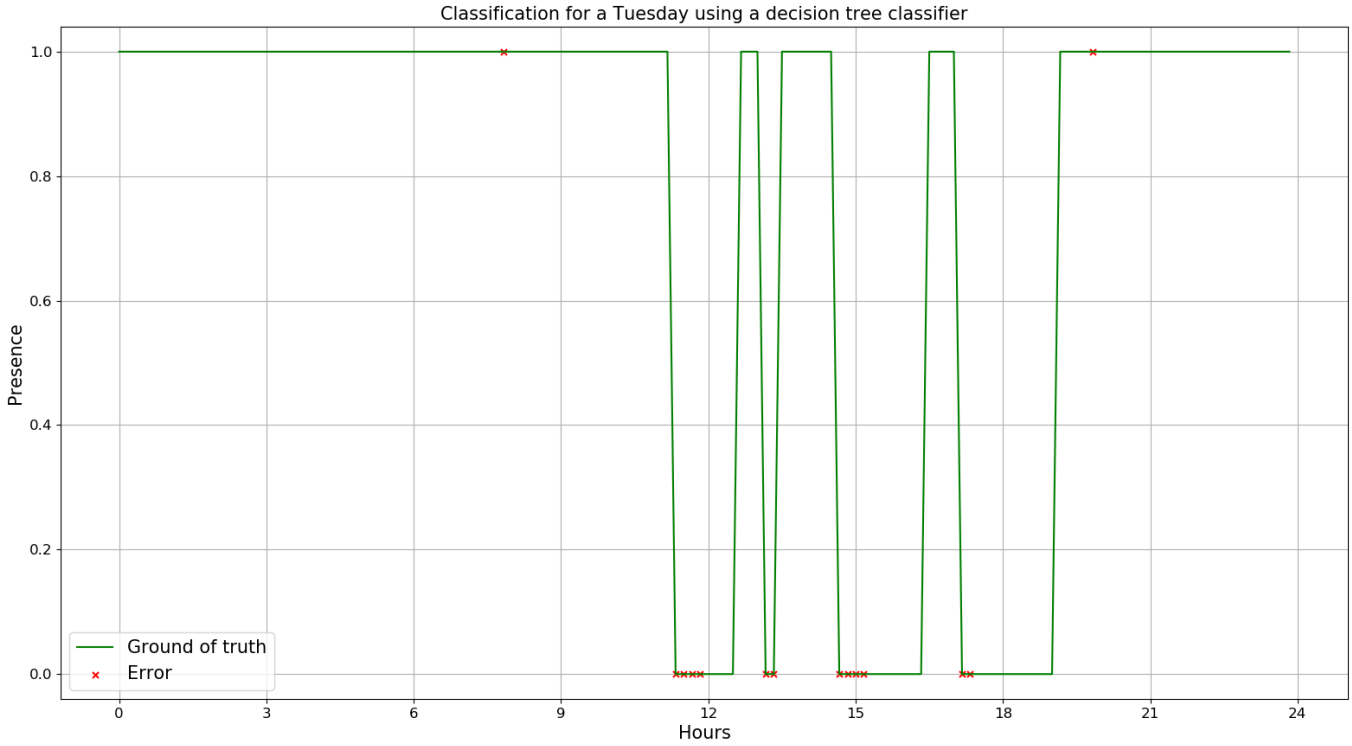

Figure 5.2: Decision tree classifier with a time step of 10 minutes and $N = 4$

The figure 5.2 above is the classification obtained for a Tuesday. You can find the classification for the same day using an MLP classifier in the results annex (figure 3).

At first sight, the classification is very good. Except for 2 segments, the classification for the positive class is perfect. We notice that the first segments of each absence are incorrectly classified. This is related to the value of N and will be explained in the next section.

We can also observe the absence of the correction model. In fact, with machine learning models using temporal features, the impact of the correction model is significantly smaller.

5.3 General observations

The goal of this section is to highlight key aspects of the approach. These are the important lessons to be learned from this approach and, more generally, from the problem itself. Both datasets are used to illustrate them.

5.3.1 The correction model and the temporal features

So far, the gain of the correction model has not been numerically evaluated. The table below presents the performance gain of the correction model, using the best performing models and the reference time step.

Model	N	Gain in precision	gain in F1 score
Logistic regression	0	3.7%	24.94%
	3	3.71%	17.28%
Decision tree	0	0.05%	0.33%
	2	-1.74%	0.5%
MultiLayer Perceptron	1	0.26%	1.77%
	4	0.49%	1.71%

Table 5.4: Performance gain for the correction model on the first dataset

Model	N	Gain in precision	gain in F1 score
MultiLayer Perceptron	5	-0.42%	1.17%
	6	-0.41%	1.23%
Decision tree	4	-0.05%	0.17%

Table 5.5: Performance gain for the correction model on the second dataset

The first observation is the gain almost negligible and sometimes even negative. In fact, the only significant gain is for the logistic regression. After some analysis, it quickly becomes clear that the gain in performance is strongly related to the non-use of temporal features. Indeed, the logistic regression is the only model that doesn't use any temporal information. Let's look at its performance using a time step of 10 minutes with $N = 3$:

Temporal features	Accuracy	Recall	Precision	F1 score
Yes	89.9%	91.89%	91.85%	91.87%
No	86.25%	85.92%	91.43%	88.58%

Table 5.6: Performance comparison for the logistic regression with and without temporal features

As you can see, the model using temporal features performs better... but is this really the case in practice? Well, if you look at the figures below, you will see on figure 5.3 that has difficulties in detecting absences, but almost all the segments belonging to the positive class are correctly classified. The correction model is useless.

On the contrary, the second figure 5.4 shows a much better model. It classifies the absences with more success, but we can see that the initial classification for the positive class was not so good before the correction model was applied. In fact, the initial model (before the correction) seems to be less efficient to make the distinction between absence and inactivity. This result is normal because the models don't have any temporal information. Therefore, it relies entirely on the traffic information, without being able to learn habits (this explains why many segments were corrected during the night). That's why the correction model that encapsulates the habits of the inhabitants is crucial for models that do not use temporal features.

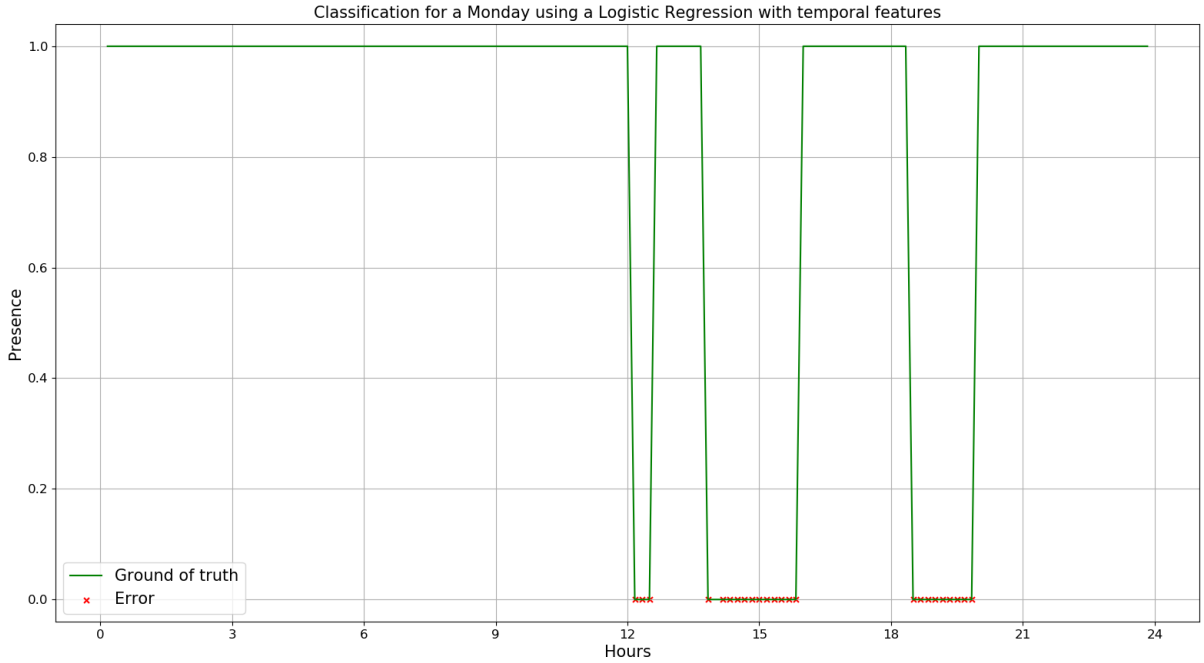

Figure 5.3: Logistic regression classifier using temporal features (time step of 10 minutes and $N = 3$

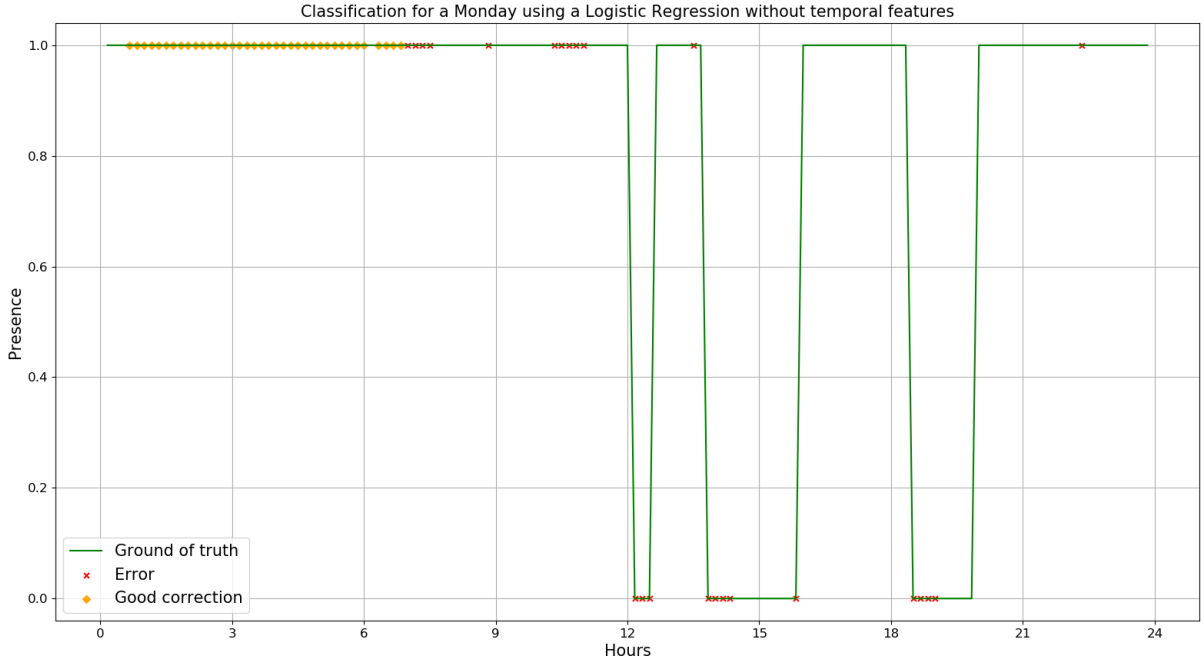

Figure 5.4: Logistic regression classifier without using temporal features (time step of 10 minutes and $N = 3$

After further analysis, it was concluded that models that don't use temporal features perform better than models using temporal features for absence classification. Especially when there are unusual absences. This is explained by the fact that these models rely only on the traffic data. They need a correction model to make the distinction between inactivity and absence.
On the other hand, models using temporal features are much more successful in detecting

presence, but have more difficulty in detecting absences. The correction model has a smaller gain, and may even in some cases wrongly correct the initial classification. These models perform well at classifying common scenarios (such as a Saturday of absence in the first environment), but are less efficient at detecting unusual scenarios.

5.3.2 Analysis of the parameter N

The parameter N and its utility have been explained in the segmentation section (4.5.2) but now we need to analyse its effect. Of course, this parameter alone doesn't explain all the results but it's important to have an idea, an intuition, of what this parameter N represents.

A good start may be to look at the variation of a classifier's f1 score as a function of N. If we take as an example the decision tree model, we will observe different results for the two datasets. The figures 5.5 5.6 below were obtained by computing the f1 score on a k-fold cross-validation on each dataset, with the reference time step of 10 minutes. In the first one, the f1 score has an increasing median value until $N = 2$. In the second, the median f1 score is almost constant but its variability decreases with increasing N.

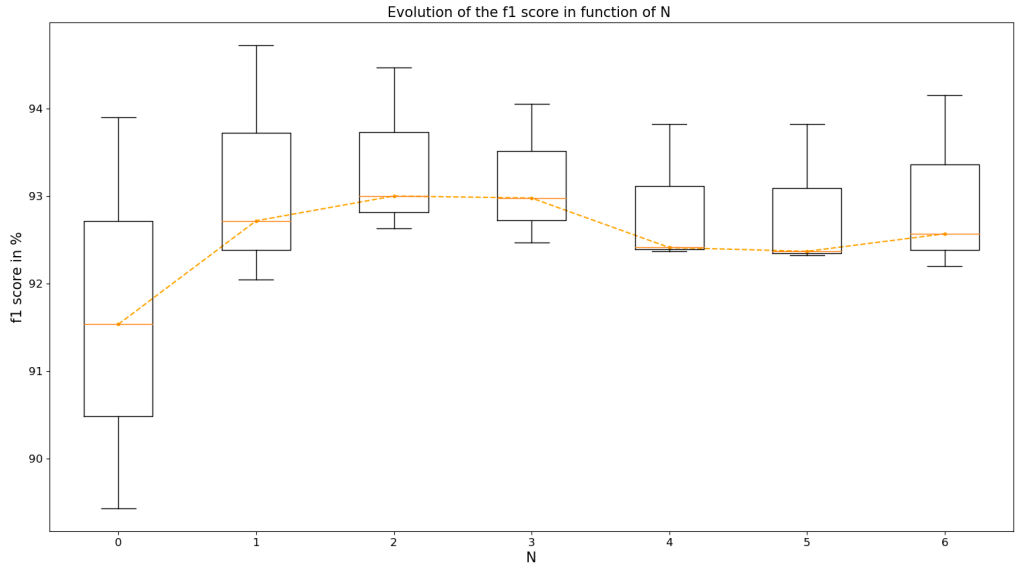

Figure 5.5: Analysis of the parameter N using the decision tree of the first dataset

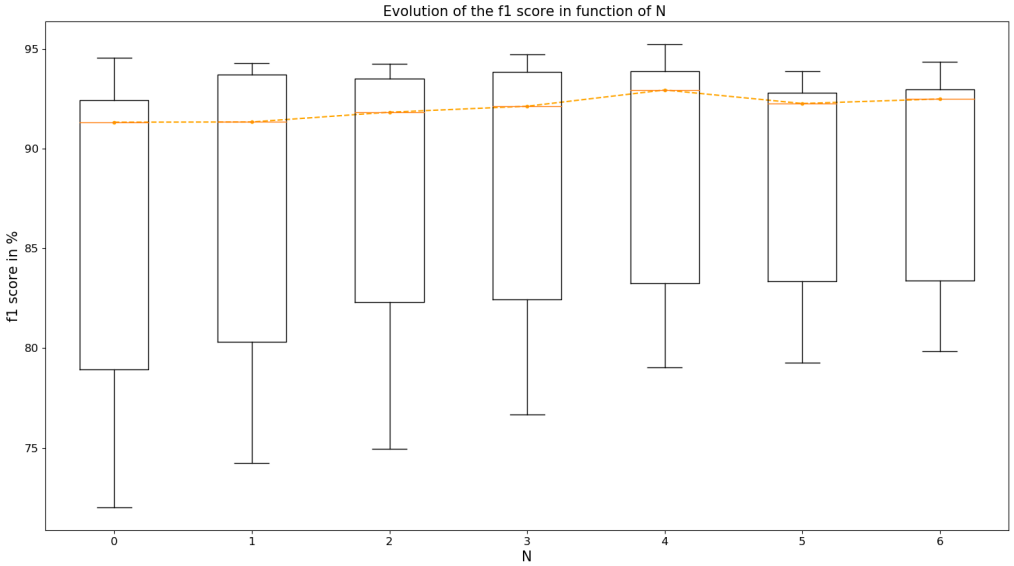

Figure 5.6: Analysis of the parameter N using the decision tree of the second dataset

Again, these results need to be put into perspective and their graphs analysed. The conclusion of this analysis is that bigger values for N are not specially increasing the performances. As we can see, a value between 2, 3 or 4 provides generally good results. For bigger values, the models tends to have more false positive (absences are more likely to be incorrectly classified).

Another observation made during the analysis, is that lower values of N tend to better classify segments during the absence time. This is especially the case for the first dataset. The parameter N can be interpreted as the **sensitivity** of the model. With lower value, the model is more dependent, sensitive to the current state of the traffic. This explains why, with a lower N, absence times are better classified. The model doesn't have access to the information of segments passed a long time ago. So it relies more on the current state of the traffic.
On the contrary, when the value of N increases, the models tend to not detect small absences. More precisely, there are more cases like the one shown below.

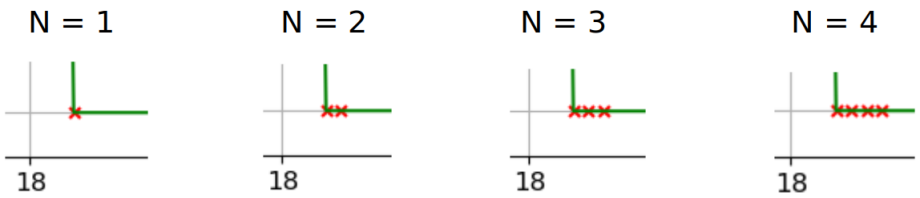

Figure 5.7: Analysis of the parameter N on absence detection

This figure was obtained using the logistic regression classifier on the first dataset. As we can observe, with increasing N, we have the same increase of errors at the begin of an absence. Let's distinguish 2 cases:

- **Error for the first segment of an absence**: this is illustrated by the figure N=1. This type of error is very common and is probably not due to the value of

N. In fact, it's rare for an absence to start exactly at the beginning of a new time interval. For example, if we use a time step of 10 minutes, each interval begins at a multiple of 10 minutes. In practice, it's very likely that most of the absences will not start exactly at a multiple of 10. As a result, the segments at the beginning of an absence will contain traffic generated during the time of presence. For example, when someone leaves the smart home, they will pass in front of the entry sensors and the first segment of absence will contain packets from the sensor. As you can imagine, there are more packets generated during presence than during absence. So, in some scenarios, most of the data of the first segment were generated during the few minutes of presence and they can have a bigger impact on the classification. This explains why this error appears recurrently. We found the same error for the last segment of an absence for the same reasons. It can also happen, but that's more rarely, that the last/first segment of a presence (before/after an absence) is classified as an absence. In that case, the explanation is the reverse of what has just been explained.

- **Other cases**: they are illustrated by the 3 other values of N. We notice that the number of segments misclassified at the beginning of an absence is the same as the value of N. The reason for this is probably that the model relies too much on the previous segments. In order to detect an absence, it needs to have all the segments (the current one + the N previous ones) captured during an absence. The consequence of this is that smaller absences (with a time $< N * \textit{time step}$) are much less likely to be detected, increasing the number of false positives. This is the reason why the precision tends to decrease as N increases, especially for the first environment (you can observe this in the result annex section .2)

Another general observation with the use of bigger N, is the decrease of false negatives. Indeed, bigger values of N allow the models to use more past information to distinguish inactivity from absence. This explains why for most models, the recall tends to increase with bigger N (again, you can observe this in the annex result section .2).

5.3.3 Impact of the time step

So far we have used the reference time step of 10 minutes. Although it's fixed for all other analyses, it's important to analyse its effect. As already explained, with a time step value of X minutes, all absences with a time $t \leq \frac{X}{2}$ are not taken into account. Its value influences the performance of the machine learning models. So it also determines the selection. But the most important thing to remember is the feasibility of the approach. In fact, the approach works for time steps of different values. The choice of this value depends more on the purpose of the attacker than on the performance optimisation: does he want a model that detects even very small absences? Or is he more interested in a model that only detects large absences? Depending on his choice, the attacker will then build a model that maximises the performance for the given time step.

That being said, it's interesting to observe the performances of the models, built for a time step of 10 minutes, on dataset using a different time step. But before jumping to the analysis, we need to compute the amount of mislabelled minutes for each time step:

Dataset	Time step	Percentage of minutes mislabelled
Dataset 1	1	0.0%
	5	0.28%
	10	0.603%
	15	0.959 %
Dataset 2	1	0.0%
	5	0.348%
	10	0.654%
	15	1.048%

Table 5.7: Percentage of minutes assigned to the wrong class per time step for each dataset

As expected, the percentage increases with the time step. The percentage remains small but with bigger values it could be problematic.

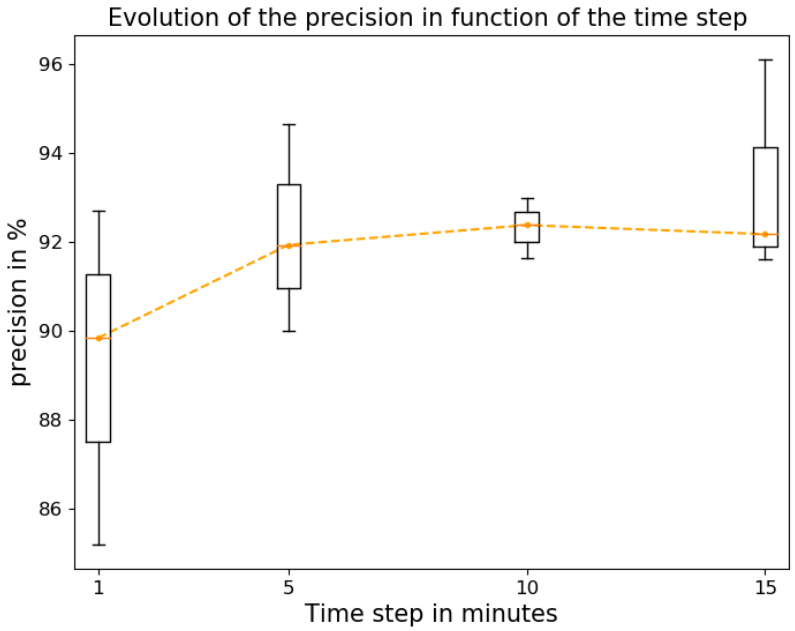

Figure 5.8: Analysis of the precision of a decision tree with N=2 on the first dataset

This figure shows the evolution of the precision as a function of the time step. Obviously, the variance and median precision are the best for a time step of 10 minutes. We can see that the other time steps have a bigger variance and that lower time steps have a less good precision.
When looking at the graphical representation, it appears that the model using a time step of 1 minute almost always predicts the positive class (despite a good precision on paper, the model is useless). A part from that, the results for other time steps are similar to those obtained for a time step of 10 minutes.

Chapter 6

Discussion and limitations

In this chapter we will put into perspective the presented approach. We will discuss about its utility, how it can be applied in practice and its limitations. In a first step, we will discuss about the role of the environment. Then, in a second step, we will see how an attacker could use this approach in a real case scenario.

6.1 Impact of the environment

In this section we will analyse the role of the environment. The environment regroups everything in the smart home (devices + people). Each of these parameters is correlated with each other. Together they define the characteristics of the environment and determine whether the approach will be successful or not.

6.1.1 IoT devices

The connected devices inside the smart home are, of course, the starting point of this thesis. Without them nothing is possible, but are all devices equal? Does the number of devices have an impact on the performance? Based on the approach and the environments used, we can distinguish some parameters that need to be taken into account:

1. **The type of device**: each device is a source of information. Now the question is "what do we learn from this information?". For some devices the answer is quite simple: "nothing". That's the case of the thermometer. On the other hand, some devices are a gold mine: motion sensors and cameras. The conclusion is easy to understand: the more the device's behaviour is dictated/influenced by its environment, the better it is. That's the reason why the machine learning models rely more on motion sensors than on smart plugs. The information provided by the sensors are more representative of the environment.

2. **The number of device**: it determines the amount of data available to the attacker. Of course, the more devices we have, the more data it will generate. The point here is more about the minimal number of devices needed for the approach to work. There isn't a numeric answer to this question, since it will also depend on the type of device and the size of the environment. However, this thesis has shown that we don't need one device per room and that we can detect whether someone is in the house or not with just a few devices.

3. **Location in the smart home**: this is especially true for devices such as sensors or cameras. The location will be crucial. In this thesis, the devices have been placed in places with a high passage rate such as the entrance or the living room. This will generate more data and activities will be easier to detect. In reality, everyone has their own preferences and will place the devices where it suits them best. For some people it will be a big no to having a camera in the living room, and for the others it won't be a big deal. In any cases, places where people usually pass through when they are in their home (entrance, living room, bathroom,...) will provide more relevant information.

6.1.2 The inhabitants

The inhabitants of the smart home also have a role to play in the performances. Some lifestyles are more predictable than others. The number of occupants is also crucial: more people means more activity, fewer people means the opposite. To illustrate this, two contrasting environments were deliberately chosen for this work. The four students in the first environment have a somewhat unusual lifestyle with late nights and late mornings. They often have visits and activities such as dinners, working sessions, etc. On the contrary, the single person in the second environment had a more classical lifestyle with early wakings and early bedtimes. The whole classification task relies on her activities. This is harder because the whole network traffic relies on her actions. For example, when she takes a nap, the environment appears to be empty. With more persons in the house, it's more likely that at least one person will generate network traffic by moving or by doing some activity.

After analysing the results of the different models, it seems that the habits are very often well classified. For example, Saturdays in the first environment are completely correctly classified. On the contrary, complete unexpected absences are not well classified (figure 4 in the result appendix illustrates this). In fact, for a full day of unexpected absence, the models confuse the absence with inactivity, especially in the periods of typically expected presences (night and evening).

6.2 Practical use

The last point to be discussed, and perhaps the most interesting, is the practical implementation of this work. So far we have evaluated the results of trained models on the data of the environments they have to classify, but is it possible to build a general machine learning a model and then use it on the data of a new environment? To answer this question, we must return to the various crucial points.

Assuming that an attacker manages to compromise one of the IoT device, accessible from the Internet, and assuming that he has access to the traffic network, he will then have to follow these steps:

1. **Characterise the local network**: his first step will be to analyse the network of the target home. In our approach we assume that the ip address of each device is known, but in practice we only use this information for the second step of the

approach (see section 4.3). In this step, we apply an early preprocessing, specific for each type of device. After analysing the results with and without this preprocessing, no significant changes appear in the results. **This means that it would be possible to use this approach, without knowing the type of device associated to each ip address !** The attacker will also need to distinguish IoT devices from regular devices (such as smartphone and laptops) but this is a piece of cake: you just need to look at the quantity of exchanged data. Laptops and smartphone exchange a LOT more of data.

2. **Apply the model**: after analysing the local network and identifying which devices are smart devices, the attacker should capture network data and process them like in the approach. Depending on the attacker's goal and on the resources of the hijacked device, he can capture data for X days and then perform the classification, or he can perform live classification.

The challenge now is to have a trained machine learning model that he can use in the second step to classify his data. Of course, we can't anticipate the exact number of devices inside the house so he will have to adapt this parameter. To give an idea of the feasibility of this step, let's train the model of each environment to classify the other one. Here, both environments use the same devices so it's not representative of the reality but it still gives an idea.

Model	N	Accuracy	Recall	Precision	F1 score
Decision tree	2	70.63%	75.25%	84.41%	79.57%
Logistic regression classifier	3	71.61%	74.93%	85.92%	80.05%
MLP classifier	4	68.28%	78.87%	79.29%	79.08%

Table 6.1: Results of testing the first environment models on the 2^{nd} dataset

Model	N	Accuracy	Recall	Precision	F1 score
Decision tree	4	75.04%	92.87%	72.32%	81.31%
MLP classifier	6	72.73%	92.64%	70.22 %	79.89 %

Table 6.2: Results of testing the 2^{nd} environment models on the first dataset

These results were obtained by using an entire dataset as the training set and then the entire other one as the test set. For each model trained on dataset x, they used the value of N that gave the best f1 score on x. They all used the reference time step of 10 minutes. As you can see, all the models have a F1 score near 80%.

Again, we need to put these results into perspective and to look at the graphs. You can find these figures in the appendix (figures 8, 9, 5, 6, 7). After analysis, the models perform surprisingly well ! A first general observation is that the correction model is more likely to misclassify. This was expected because it is based on probabilities and habits. Since the habits are different for the two datasets, it was expected to have errors.

For the models of the second dataset, it appears that they perform well on the first dataset. The classification is good, especially the presence detection for days during the

week. For the weekend days, they perform less well, especially during the night and in the evening, where we have a high error rate. But again, this was expected, as no one was there on Saturdays and the students only came back on Sunday evening. On the contrary in the second dataset (= the training set), there was more activity during the weekend. In addition to that, the models don't use the *week_day* feature, so they try to classify the week-end like other days of the week.
We also observe that small absences are less likely to be detected, due to the higher value of N (as explained in the previous section 5.3.2).

Furthermore, the models from the first dataset and tested on the second one also perform well. Logistic regression is excellent at detecting absence and activity. As expected, it struggles to distinguish inactivity from absence, so the correction model is very useful to correct the classification during the night and in the evening. Again, we find a number of errors at the beginning of absences equal to the value of N. The MLP, a model giving more weight to habits, doesn't perform very well. It struggles to detect absences and gives particularly bad results for complete unexpected days of absence! Finally, the decision tree is also very efficient, especially for the week days. During the weekend it has less good results. Again, this was to be expected as it was trained on the first dataset, where almost no activity took place during the weekend. On the contrary, in the second dataset, more activities, visits, etc happened during the weekends.

Finally, a more general observation is that some models struggle to classify unseen absences in the training set. Presence is easier to detect, but unexpected absences (especially at the night or in the evening) can be difficult, but not impossible, to classify.
The correction model is also more likely to misclassify. This is more the case for some models than others, but compared to the original approach, the total number of errors is higher. This was expected and shows the importance of having general correction rules. The calculated statistics representing the habits are, of course, related to the environment, so it's difficult to estimate them in advance. This last part could be solved by having different sets of pre-calculated statistics, associated with different type of environment. The attacker could then decide which set to use based on some knowledge of his target, but this has not been tested and is outside the scope of this work.

Chapter 7

Conclusion and future works

7.1 Conclusion

The increasing number of IoT devices in our homes raises privacy concerns. Recent studies have shown that such devices can be identified, and this only by passively sniffing wireless traffic. This is a major concern because this type of attack relies on the basic principle of network protocols. It has also been shown that it is possible to identify activity using this technique. This master thesis went even more further by demonstrating the feasibility of presence detection, even in environments with only few devices. The approach is based on 6 simple steps and relies on machine learning to perform the classification task. Parameters such as the time step and N allow to tune the classification, according to the attacker's goal. Several classical machine learning models such as decision trees, logistic regression and neural networks shown good performance. This also demonstrates how "simple" and reproducible this solution is.
In order to better evaluate the solution, two radically different smart environments have been used. Despite that having the best performances was not the initial goal, the approach was able to reach f1 scores superior to 90% ! This result means that smart devices, although seemingly harmless, are a real threat of our privacy. The question now is: "Can an attacker use this approach to train a model using his own data, and then use it against new smart homes?". If we look at the results of the models trained on the data captured in one environment, and then tested on the second one, the answer seems to be "yes". Further research is needed to be done to confirm this hypothesis, but so far, this work has demonstrated the feasibility of presence detection in one smart home, using data captured in another smart home using the same devices.

7.2 Future works

This work has revealed a real security concern. But there are still some points that need to be explored:

- **More diverse environments**: by environment we mean the devices and the house itself. In fact, only a few smart devices have been used for this work. It would be interesting to analyse the impact of a range of devices, from different type but also from different brands (maybe the same devices but from different brands don't work in the same way). This could include more expansive devices, but also the use of connection types other than WiFi.

The architecture of the house itself also plays a role. In this work we have only used flats. Flats are generally smaller than houses, which limits the range of activities. With houses new challenges will arise. For example, if the IoT devices are connected via WiFi, they are likely to be close to the WiFi box, leaving a large uncover space. A lot of activity can happen in these areas, reducing thus the interaction with the devices.

- **Easier attacker model**: for this work, strong constraints were assumed, as described in section 3.2. The attacker only has access to the traffic of the IoT devices, he doesn't have any prior information about his target, etc. It would be interesting to analyse the impact of these assumptions and the performance gain that we could achieve by relaxing the problem. For example, if the attacker has access to the traffic network of other devices (laptops, phones, ...) it would help to distinguish inactivity from absence. Or if the attacker has some prior knowledge about the lifestyle of the inhabitants, etc.

- **Automated solution**: with the idea of this work in mind, it would be interesting to build a complete automated solution that now tries to reach the best possible performance. To achieve this, the work of other papers should be taken into account, for example to identify the devices ([6], [7], [11], [13]) or activities ([3], [5]). The different papers could be merged into one big work, so that everything is automated and ready to use.

- **Defence**: an important point not covered in this paper is how to defend smart homes against these attacks. Some works propose an "active defence", with traffic shaping and packets injection ([3], [5], [10], [12]). Indeed, the approach relying only on the number of packets and on their size, if you want to counter it, you need to make the resources of the attack unusable (or at least less meaningful). Another defence would be to prevent IoT devices from being hijacked (or at least make them more secure). In this way, an attacker wouldn't be able to access the home's traffic.

- **Societal, economic and legal consequences**: the various paper cited in this work, and this work itself, demonstrate the existence of a privacy leak. This raises legal questions such as "Who is held responsible for a privacy breach?". Should the companies selling IoT devices be held responsible? Or is it the user's responsibility to protect their own privacy? In both cases, the answer has social and economic consequences. If the companies are held responsible, they will probably provide a better security (new protocol, encryption, device behaviour,...) but this will probably increase the cost of the devices. If the user is held responsible, then he must be informed of the problem, but who is going to warn him? Maybe not the IoT companies... As you can see, this is a complex problem, involving several fields.

Bibliography

[1] J. Howarth, "80+ Amazing IoT Statistics (2023-2030)", https://explodingtopics.com/blog/iot-stats, accessed: 2023-01-14.

[2] Dr. J. Lasquety-Reyes, "Number of Smart Homes forecast in the World until 2025", https://www.statista.com/forecasts/887613/number-of-smart-homes-in-the-smart-home-market-in-the-world, accessed: 2023-01-14.

[3] X. Liu, Q. Zeng, X. Du, S. Likitha Valluru, C. Fu, X. Fu, B. Luo, SniffMislead: Non-Intrusive Privacy Protection against Wireless Packet Sniffers in Smart Homes" in RAID '21: 24th International Symposium on Research in Attacks, Intrusions and Defenses, 2021, pp. 33-47.

[4] Vivian, Martin. Reverse-engineering the physical configuration of Smart Homes. Ecole polytechnique de Louvain, Université catholique de Louvain, 2021. Prom.: Sadre, Ramin. http://hdl.handle.net/2078.1/thesis:30697

[5] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2020, pp. 207–218.

[6] M. R. Shahid, G. Blanc, Z. Zhang and H. Debar, "IoT Devices Recognition Through Network Traffic Analysis," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5187-5192

[7] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray and I. Ray, "IoTSense: Behavioral Fingerprinting of IoT Devices", 2018

[8] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic", 2017

[9] B. Copos, K. Levitt, M. Bishop and J. Rowe, "Is Anybody Home? Inferring Activity From Smart Home Network Traffic," 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 2016, pp. 245-251

[10] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, N. Feamster, "Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic", 2017

[11] A. Sivanathan et al., "Characterizing and classifying IoT traffic in smart cities and campuses," 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 2017, pp. 559-564

[12] N. Apthorpe, D. Yuxing Huang, D. Reisman, A. Narayanan, N. Feamster, "Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping", Proceedings on Privacy Enhancing Technologies, 2019-03, pp. 128-148

[13] A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," in IEEE Transactions on Mobile Computing, vol. 18, no. 8, pp. 1745-1759, 1 Aug. 2019,

[14] Indoor smart camera, https://www.amazon.fr/gp/product/B07XLML2YS/ref=ppx_od_dt_b_asin_image_s00?ie=UTF8&th=1, accessed: 25/10/2022

[15] Indoor smart plug, https://www.amazon.fr/gp/product/B07N7B7NXF/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&th=1, accessed: 25/10/2022

[16] Indoor movement sensors, https://www.amazon.fr/gp/product/B07STZPCJ9/ref=ppx_od_dt_b_asin_title_s01?ie=UTF8&psc=1, accessed: 25/10/2022

[17] Indoor connected thermometre, https://www.amazon.fr/gp/product/B08Y8PQ7XZ/ref=ppx_od_dt_b_asin_title_s01?ie=UTF8&psc=1, accessed: 25/10/2022

[18] OSI model, https://en.wikipedia.org/wiki/OSI_model, accessed: 2023-02-08

[19] Logical link control, https://en.wikipedia.org/wiki/Logical_link_control, accessed: 2023-02-06

[20] Internet Protocol, https://en.wikipedia.org/wiki/Internet_Protocol, accessed: 2023-02-08

[21] Transport layer, https://en.wikipedia.org/wiki/Transport_layer, accessed: 2023-02-06

[22] Setting up a Routed Wireless Access Point, https://www.raspberrypi.com/documentation/computers/configuration.html#setting-up-a-routed-wireless-access-point, accessed: 2022-11-6

[23] tcpdump, linux manual page, https://linux.die.net/man/8/tcpdump, accessed: 2022-10

Annex

.1 Correction model

```
def default_rule(pres_proba, abs_proba):

    if abs_proba <= pres_proba:
            return 1
    return 0

def correction_rules(pres_proba, abs_proba, time_pres_proba, time_abs_proba, hour):

    # CASE 1 : activity is 'clearly' detected and
    # it's not triggered by a false classification
    if (pres_proba >= 0.6 and time_pres_proba > 0.02):
        return 1

    # CASE 2 : no activity is 'clearly' detected
    # and we are during the night
    if hour < 7 or hour >= 22:

        if time_pres_proba >= 0.75:
            return 1
        if time_abs_proba > 0.75:
            return 0

    # CASE 3 : very frequent absence/presence during the day
    if time_pres_proba >= 0.85 and pres_proba >= 0.5:
        return 1

    if time_abs_proba >= 0.85 and abs_proba >= 0.5:
        return 0

    return default_rule(pres_proba, abs_proba)
```

.2 Result annex

.2.1 First environment

This subsection contains all the results related to the first environment. The models used in the first environment:

- A logistic regression classifier from the python library *sklearn.linear_model*, without temporal features:

$$LogisticRegression(max\_iter = 170, C = 3)$$

- A decision tree classifier from the python library *sklearn.tree*, using all the features of the dataset:

$$DecisionTreeClassifier(max\_depth=6,\ min\_samples\_split=2,\ min\_samples\_leaf=5,\ random\_state=0)$$

- A MultiLayer Perceptron classifier from the python library *sklearn.neural_network*, using all the features of the dataset:

$$MLPClassifier(solver="adam",\ activation="logistic",\ hidden\_layer\_sizes=(100,\ 100),\ max\_iter=100,\ random\_state=1)$$

Model	time step	N	Accuracy	Recall	Precision	F1 score
Logistic regression	10	0	80.44%	72.77%	**94.38%**	82.16%
		1	84.51%	80.36%	93.76%	86.54%
		2	85.66%	84.15%	92.05%	87.92%
		3	**86.25%**	85.92%	91.43%	**88.58%**
		4	86.05%	86.99%	90.2%	88.56%
		5	85.91%	87.26%	89.77%	88.48%
		6	85.97%	**87.312%**	89.81%	88.52%
Decision tree	10	0	89.84%	90.5%	**92.87%**	91.62%
		1	91.49%	93.9%	92.47%	93.16%
		2	**91.69%**	**94.43%**	92.33%	**93.37%**
		3	91.45%	93.95%	92.39%	93.17%
		4	91.12%	93.35%	92.4%	92.87%
		5	91.08%	93.34%	92.4%	92.84%
		6	91.25%	93.6%	92.36%	92.97%
MultiLayer Perceptron	10	0	90%	91.67%	92.17%	91.92%
		1	90.57%	92.34%	**92.47%**	92.39%
		2	**90.86%**	93.03%	92.34%	92.65%
		3	90.16%	92.45%	91.77%	92.10%
		4	90.82%	93.88%	91.49%	**92.66%**
		5	89.62%	93.15%	90.38%	91.74%
		6	90.52%	**94.87%**	90.31%	92.52%

Table 1: Comparison of machine learning models for the first environment

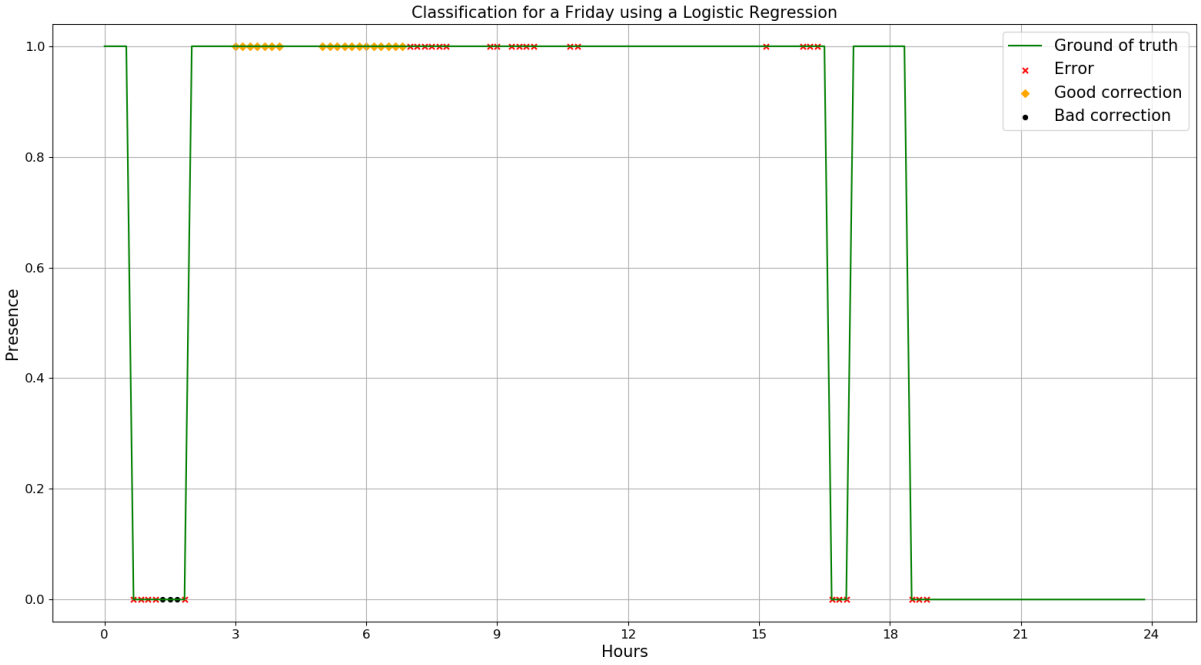

Figure 1: Logistic regression classifier with a time step of 10 minutes and $N = 3$

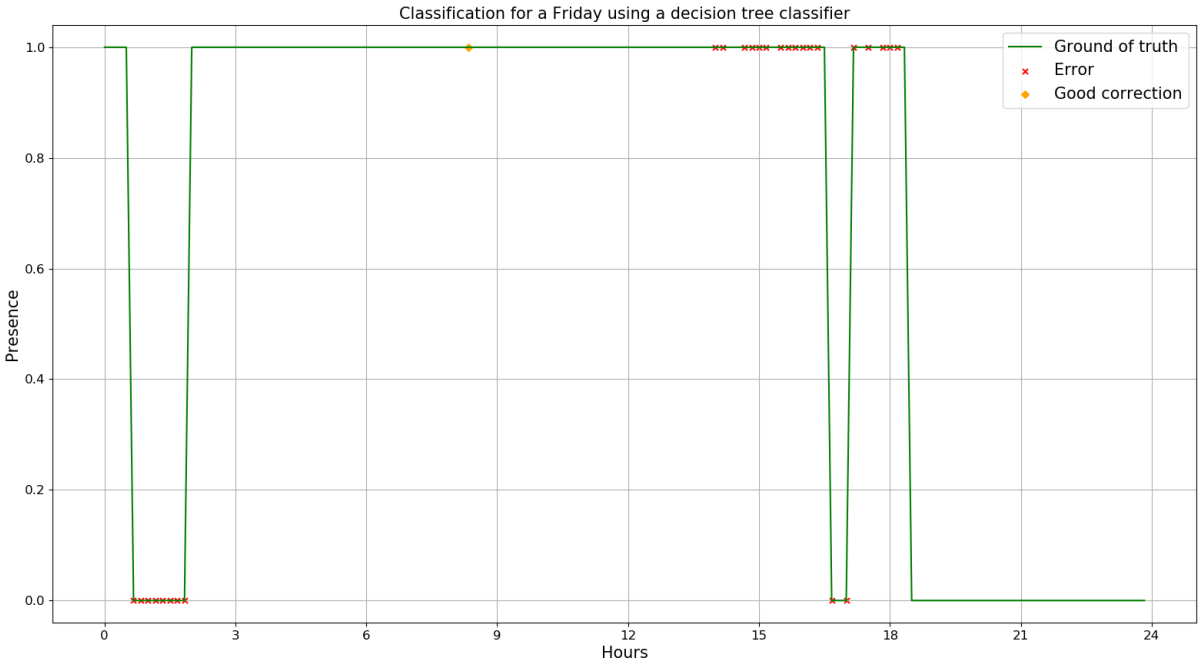

Figure 2: Decision tree classifier with a time step of 10 minutes and $N = 2$

.2.2 Second environment

This subsection contains all the results related to the second environment. The models used in the second environment are:

- A MultiLayer Perceptron classifier from the python library *sklearn.neural_network*, without using the 'Week_day' attributes:

  MLPClassifier(solver="sgd", activation="logistic", hidden_layer_sizes=(100, 100), max_iter=500, random_state=1)

- A decision tree classifier from the python library *sklearn.tree*, without using the 'Week_day' attributes:

  DecisionTreeClassifier(max_depth=13, min_samples_split=47, min_samples_leaf=15, random_state=0)

Model	**time step**	N	**Accuracy**	**Recall**	**Precision**	**F1 score**
MultiLayer Perceptron	10	0	73.95%	**97.94%**	75.46%	84.38%
		1	76.17%	93.56%	79.95%	85.02%
		2	78.95%	93.22%	81.98%	86.34%
		3	80.54%	93.75%	82.8%	87.22%
		4	80.6%	93.73%	82.82%	87.27%
		5	81.15%	94.28%	**82.9%**	87.63%
		6	**81.25%**	94.67%	82.78%	**87.74%**
Decision Tree	10	0	79.63%	90.15%	83.63%	86.14%
		1	80.97%	91.08%	84.47%	87.05%
		2	81.83%	91.86%	84.95%	87.67%
		3	82.52%	93.05%	84.89%	88.21%
		4	**83.91%**	**93.7%**	**85.71%**	**89.06%**
		5	83.04%	92.51%	85.7%	88.5%
		6	83.38%	93.21%	85.57%	88.77%

Table 2: Comparison of machine learning models for the second environment

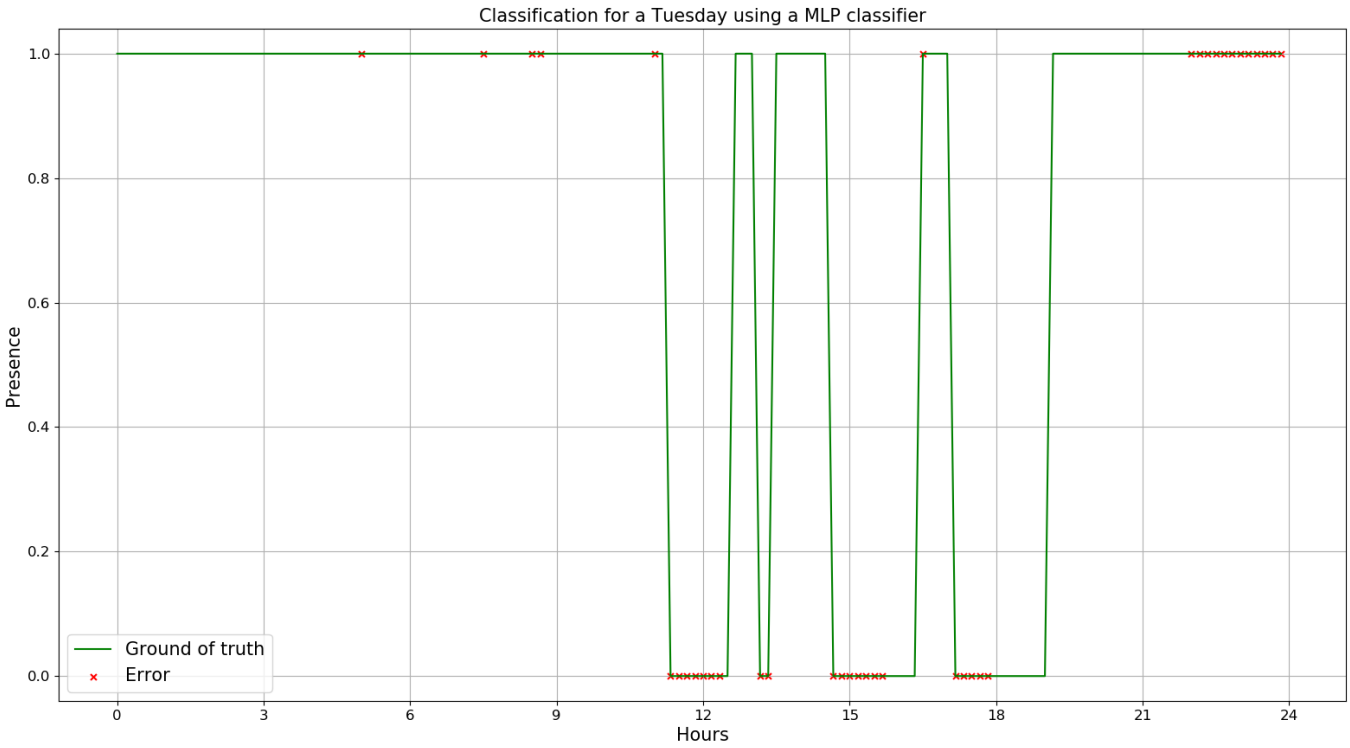

Figure 3: MLP classifier with a time step of 10 minutes and $N = 6$

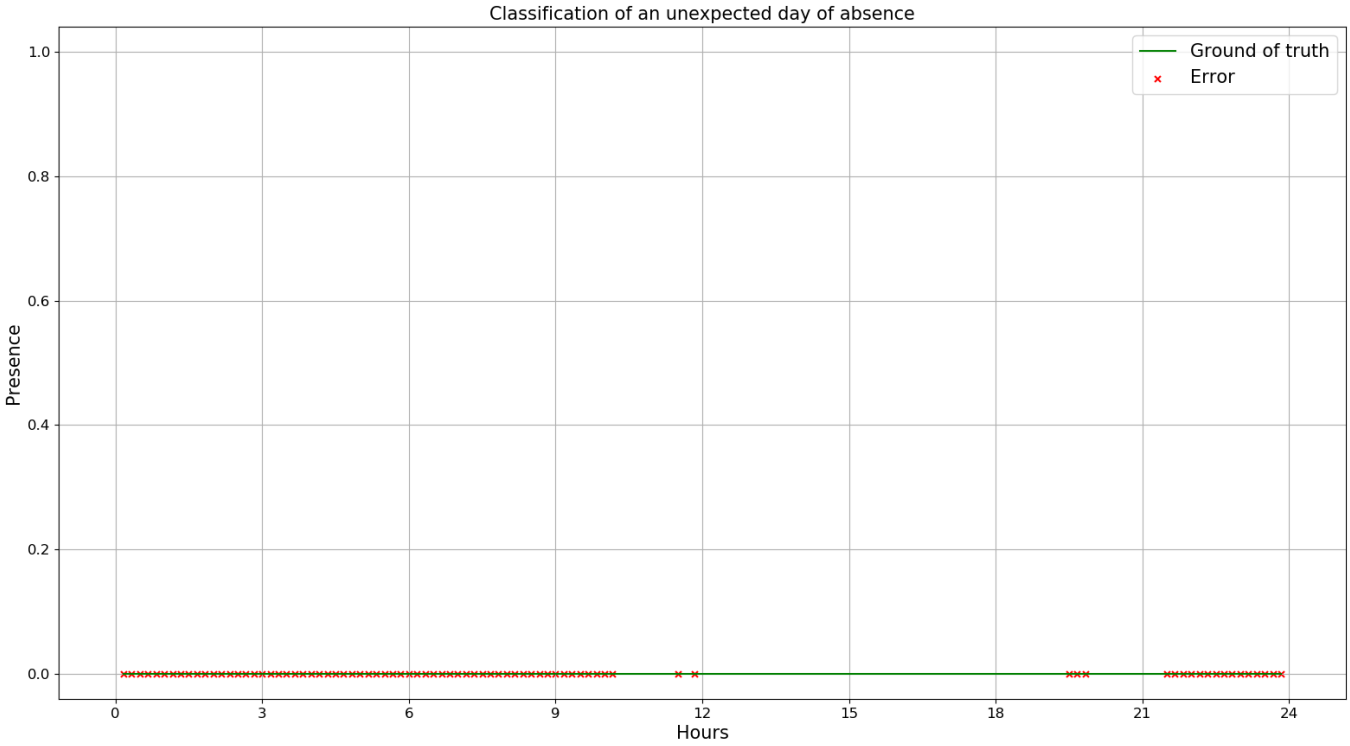

Figure 4: Classification of an unexpected day of absence using a decision tree with a time step of 10 minutes and $N = 4$

.2.3 Cross-Dataset Performance Evaluation

In this subsection you will find the models of each dataset evaluated on the others dataset. Figures 5, 6 and 7 train the models of the first dataset and evaluate them on the second dataset. Figures 8, 9 train the models on the second dataset and use the first dataset as test set.

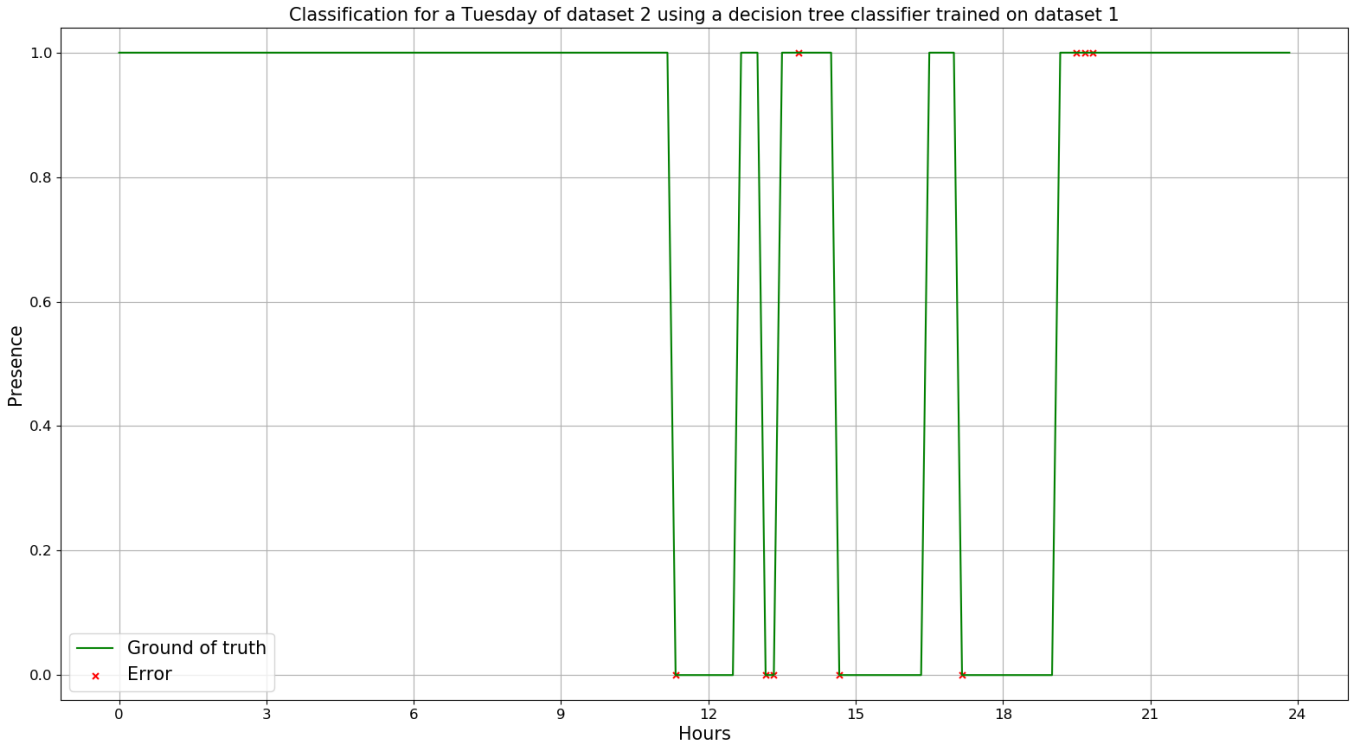

Figure 5: Decision tree of the first dataset with a time step of 10 minutes and $N = 2$

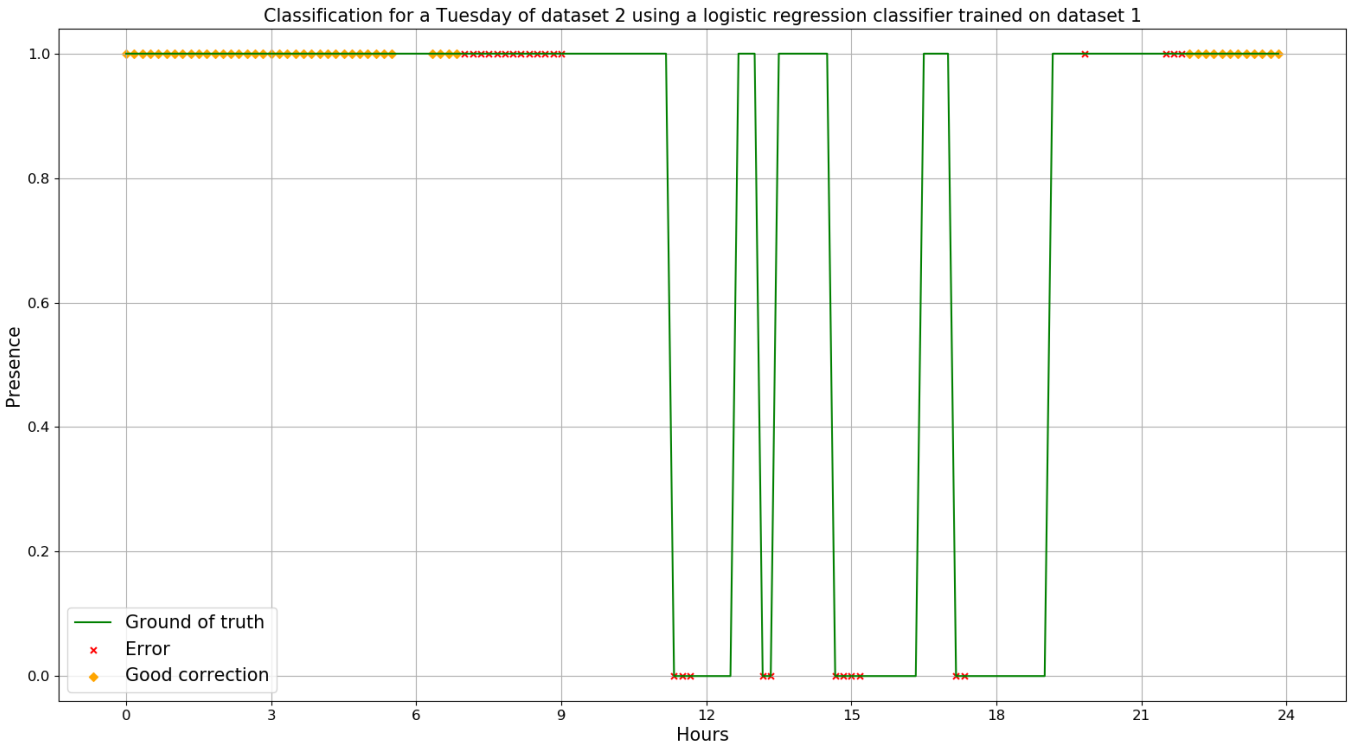

Figure 6: Logistic regression classifier of the first dataset with a time step of 10 minutes and $N = 3$

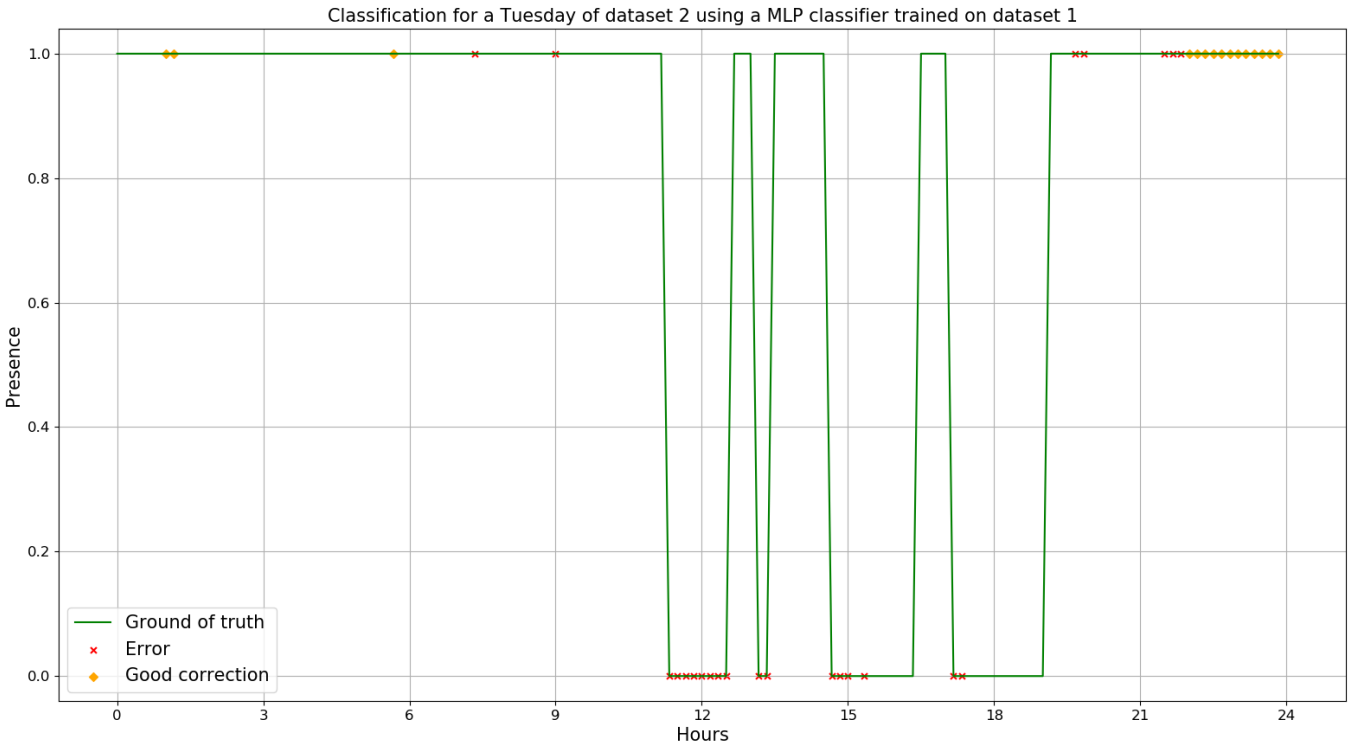

Figure 7: MLP classifier of the first dataset with a time step of 10 minutes and $N = 4$

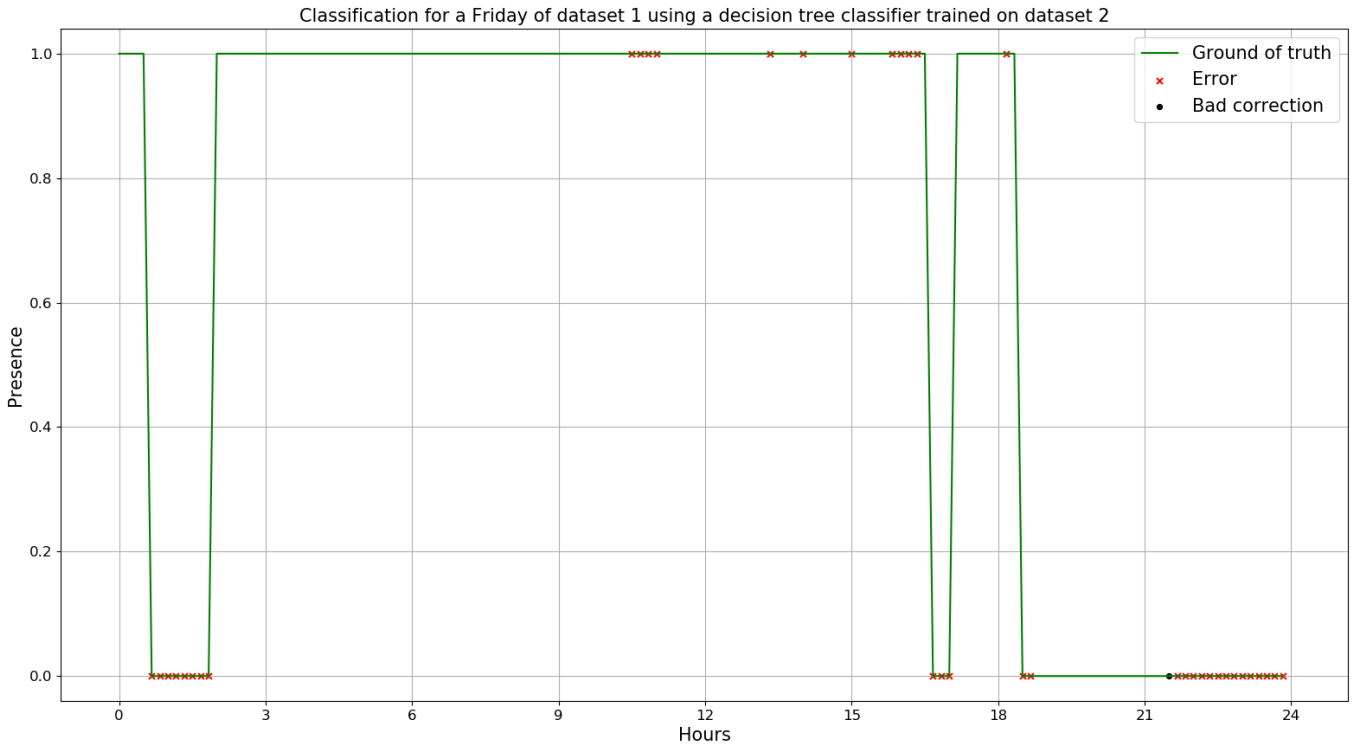

Figure 8: Decision tree of the 2^{nd} dataset with a time step of 10 minutes and N=4

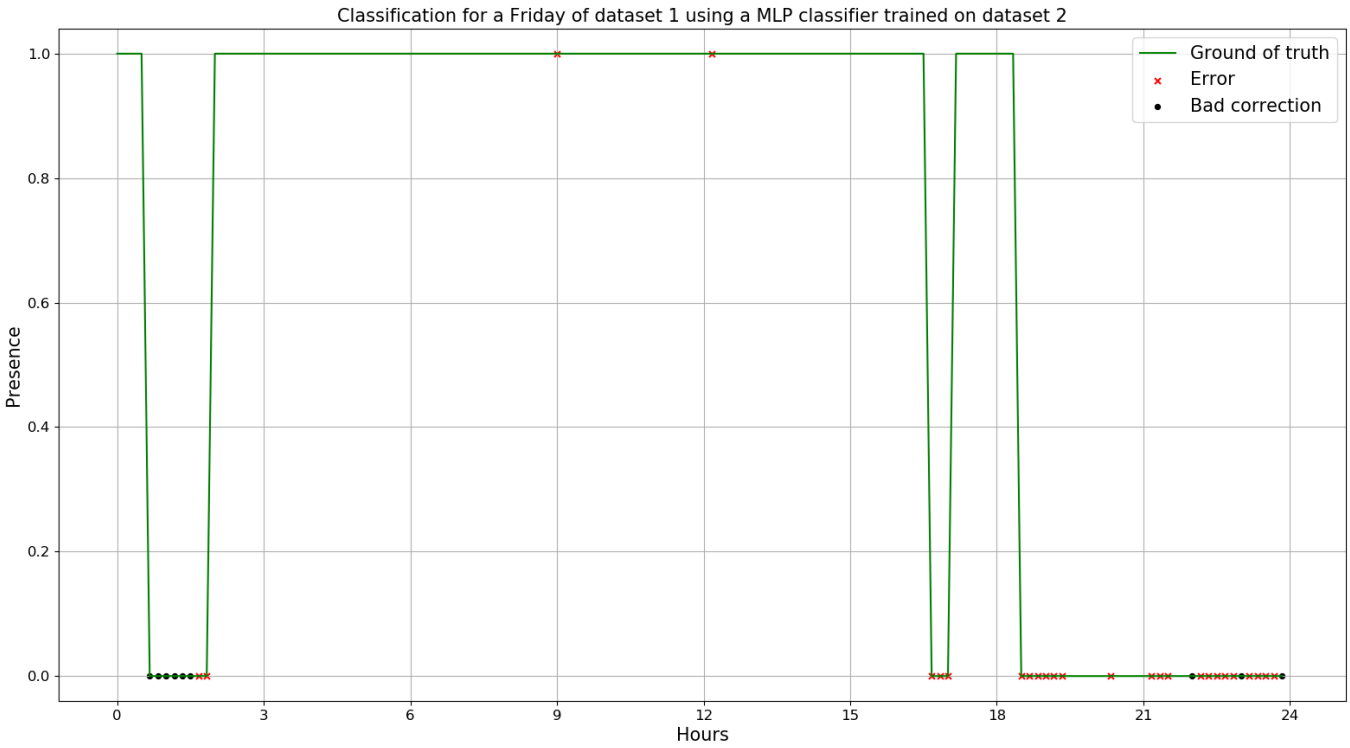

Figure 9: MLP classifier of the 2^{nd} dataset with a time step of 10 minutes and $N = 6$

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**