

École polytechnique de Louvain

An exploration of possible applications of online data recovery, in the context of energy communities

Author: **Arthur FÉRIER**

Supervisor: **Axel LEGAY**

Readers: **Mathias DE SCHIETERE DE LOPHEM, Yves DEVILLE, Lionel DRICOT**

Academic year 2023–2024

Master [120] in Computer Science and Engineering

Abstract

In a world undergoing a drastic energy transition, many ingenious ideas are being developed to tackle this crisis. The energy community concept is one of them. By sharing the production of green and local energy among a group of homes or businesses located close to each other, the various players benefit both economically and socially in a sustainable way. The aim of this project is to explore the possible advantages of real-time energy monitoring. One such advantage is providing members of these communities with the best indication of when it would be worthwhile to use green energy production. The solution proposed is based on an existing forecast algorithm developed by the UCLouvain. To achieve this, the means of transmitting data are being explored using smart meters. Two algorithms are presented. The first aims to detect habit changes and indicate whether the forecast algorithm needs to be retrained. The second focuses on making very short-term electricity consumption recommendations to exploit the community's production as much as possible.

Acknowledgements

First and foremost, I would like to sincerely thank Mathias de Schietere de Lophem. I benefited from his advice on the direction to take, his in-depth review, and his experience on the subject throughout the whole year. I really enjoyed the countless hours of meetings every two weeks in room B.074, with its very non-functional projector and empty board markers. I would also like to thank my supervisor, Prof. Axel Legay who proposed this topic with Mathias.

I also want to thank WeSmart, who put their trust in me by lending their equipment and data for this project.

Lastly, I'd like to express my gratitude to those close to me for their moral support during the writing process, and particularly to Briec, with whom I spent many hours writing our respective reports.

Contents

1	Introduction	1
1.1	WeSmart	1
1.2	Context	2
1.3	Energy communities and their role in society	4
1.3.1	European legal framework	4
1.3.2	Belgian legal framework	5
1.3.3	Types of Energy communities in Belgium and Europe	10
2	Setting up the online data transmission	14
2.1	Introduction	14
2.2	Smart grids and smart meters	15
2.3	Transmitters	15
2.3.1	Theoretical review	16
2.3.2	Practical review of the available transmitters	17
2.4	Communication protocols	20
2.4.1	Review of the communication protocols	21
2.4.2	Choice of the protocol	26
2.4.3	MQTT broker and server set up	27
2.5	Conclusion	29
3	Online data filtering	30
3.1	Hole extrapolation	30
3.1.1	Extrapolation algorithms	31
3.1.2	Practical testing	33
4	Change analysis algorithm	36
4.1	Introduction	36
4.2	Change of habits and daily consumption data	37
4.3	Change Point Detection Algorithm	40
4.3.1	ChangeFinder	41
4.3.2	Bayesian online change point detection algorithm	42

4.3.3	Cumulative Sum algorithm	43
4.3.4	Choice of the algorithm	44
4.4	Changepoint identification	55
4.4.1	Implementation choices of the Change Point Identification	56
4.4.2	Final results	57
5	Energy community excess production indicator	60
5.1	Introduction	60
5.2	Algorithm	61
5.2.1	Final result analysis and conclusion	64
6	Discussion and conclusion	66
A	Annexes	71
A.1	dsmrTelegram.ino base and modified version	72
A.2	Bash script of the data retrieving	80
A.3	Python script of the data retrieving	81
A.4	Pseudo python code of the CP identification	85
A.5	Graphical analyse of the CP identification	89
A.5.1	Stockel dataset 0	89
A.5.2	Stockel dataset 1	90
A.5.3	Stockel dataset 2	91
A.5.4	Stockel dataset 3	92
A.5.5	Stockel dataset 4	93
A.5.6	Stockel dataset 5	94
A.5.7	Stockel dataset 6	95
A.5.8	Greenbizz dataset 0	96
A.5.9	Greenbizz dataset 1	97
A.5.10	Greenbizz dataset 2	98
A.5.11	Greenbizz dataset 3	99
A.5.12	Greenbizz dataset 4	100
A.5.13	Greenbizz dataset 5	101
A.6	pseudo code of the excess production indicator	102

List of abbreviations

All words are sorted alphabetically by their abbreviation in this list.

ACK Acknowledgement

BOCPD Bayesian Online Change Point Detection

CEC Citizen Energy Community

CF Change Finder

CoAP Constrained Application Protocol

ComP Communication Protocol

CP Change Point

CPD Change Point Detection

CPI Change Point Identification

CSC Collective Self-Consumption

CUSUM Cumulative Sum algorithm

CWaPE Walloon Commission for Energy

DTLS Datagram Transport Layer Security

EC Energy Community

EU European Union

IEMD Common Rules for the Internal Electricity Market Directive

LEC Local Energy Community

MQTT Message Queuing Telemetry Transport

P2P Peer To Peer

QoS Quality of Services

REC Renewable energy community

RED II The Renewable Energy Directive

REST Representational State Transfer

SG Smart Grid

SM Smart Meter

TCP Transmission Control Protocol

TLS/SSL Transport Layer Security and Secure sockets layer

UDP User Datagram Protocol

VREG Flemish Regulator for the Electricity and Gas Market

Chapter 1

Introduction

The goal of this work is to explore the possibilities of online data monitoring for energy communities (ECs) in a real environment. To gather the data and test the solutions, we worked in association with WeSmart, a company specializing in the monitoring of energy communities. They aim to add an excess production indicator that can inform members of the best moments to consume the community's energy from an online perspective and in the very short term.

1.1 WeSmart

As part of this end-of-studies project, we've had the chance to work in collaboration with WeSmart¹, a Brussels-based scale-up that aims to support the development of ECs at different stages. They help create the EC from the initial idea to the follow-up, once the community is well established.

For that, they offer a digital platform that connects all the participants, visualizes how the community is evolving, provides keys to the distribution of local energy among the participants, manages the invoicing of every actor in the EC, and more.

The collaboration between WeSmart and UCLouvain enables research to move forward in the field, with practical feedback on what is being studied. WeSmart, for its part, benefits from results directly linked to its area of expertise. The two entities began the project after obtaining funding from Innoviris², a public administration that funds and supports research and innovation in the Brussels-Capital Region.

¹<https://www.wesmart.com/>

²<https://innoviris.brussels/>

1.2 Context

An energy forecast algorithm has already been created by UCLouvain. It predicts the consumption and production of the members of the EC over a relatively long period, i.e., one month, to identify where there may be excess energy. It relies on the EC's historical data over previous years to learn the habits of the consumers and predict their behavior for the coming month. The more recent the data, the more weight it will have. This historical information is coupled with factors that may influence electricity consumption, such as holidays, weather, and temperature. By learning the consumption patterns of the members of the EC under these parameters, the algorithm can make a forecast based on the weather and holidays for the month ahead, taking all these factors into account. The historical and forecast weather data are provided by the Forecast.Solar RESTful API³, which gives specific information for solar panel production.

Now, the interest of WeSmart in this project is to add recommendations based on live data to their digital platform. These recommendations are meant to be more precise and on shorter terms than the general forecast algorithm. Its objective is to tell the participants of the EC when the energy consumption will be on the electricity of the community for the next three hours at most. This should encourage community participants to shift their consumption loads to times when there is more production. There is a clear financial interest in this application, but it can also help optimize the internal consumption of the community, increase the proportion of green energy consumed compared to the electricity of the grid, and even possibly reduce energy consumption at peak times, when energy providers struggle to satisfy the demand. For example, the platform should inform the participants that the electricity source for the next 3 hours comes from the solar panels, so that they can launch high-energy processes when they see the indication. Note that these short-term recommendations are still based on the forecast algorithm. The idea is that if the forecast is close enough to the live data, we can conclude that it is safe to trust it with high confidence to make an accurate prediction for the next few hours.

A second use of the online data retrieval that is developed in this project is the following. Every month, with a certain delay, the energy providers can send the data of their clients with the quantity of electricity consumed every 15 minutes. This ensures that roughly every month, the forecast algorithm can be retrained with the latest consumption patterns to have the highest chance of making a good prediction. The problem is that we cannot train the model every day or every

³<https://forecast.solar/>

few weeks, and sometimes there is a real need for re-training because of a clear change in consumption patterns. For example, a habit change can occur in the first week of the month, and the consumption history can be received from the energy providers two weeks after the end of the month. This potentially introduces more than a month of poor predictions due to one change in habit.

Another aim of this dissertation is therefore to compare the possibilities of recovering online data and developing a trend change detection algorithm that can give a confidence interval for recommendations. With online data monitoring, we are able to compare in (near) real time if the predictions of consumption are trustworthy and re-train the prediction algorithm during the month if they're not.

To achieve this goal, WeSmart allowed us to work with the data of two real and already effective ECs that they manage. One is located in the Greenbizz incubator, so every participants are offices. The other is a more standard EC located in Stockel and is a flat. The two ECs are thus all located in the Brussels region. We have at our disposal the history of energy data.

We could also compare two different emitters imposed by WeSmart. We present a theoretical choice to set up the transmission of power consumption and a review of the communication protocol in chapter 2.

Then we set up a server to receive and store the data and make extrapolations if there are holes in the consumption in chapter 3.

Then come the final results of the goals stated here above. For the confidence estimation, we tested several algorithms of Change Point Detection (CPD) as a basis for our solution to the problem. On top of that, we added another logical layer to verify the CPD algorithm and detect if the anomaly detected really is a change in habits of consumption, an extreme point, or a problem with the predictions. This is developed in Chapter 4.

Lastly, the excess production indicator algorithm is developed in chapter 5.

Before diving into the main subject, we first need to introduce and develop the different types of energy communities across Belgium and Europe. Their different frameworks, goals, and solutions to the energy crisis are also developed in the next sections.

1.3 Energy communities and their role in society

According to the European definition and explained by the European Federation of Citizen Energy Cooperatives, Rescoop⁴, Energy Communities are Local Energy Systems that were developed to support the *"fast growth of energy production from renewable energy sources in constrained networks"*. By decentralizing the current energy network into a flexible one that better supports the growing number of active consumers, [1].

The concept is still new and has attracted a great deal of interest from many organizations, companies, and researchers in recent years [2]. Belgium and the European government also took an interest in the domain and have stated laws to regulate and support the emergence of these new ways of consuming.

1.3.1 European legal framework

With the two directives on the promotion of the use of energy from renewable sources of the 11 December 2018 [3] and 5 June 2019 [4], the European parliament has provided significant push for the emerging of ECs.

The Renewable Energy Directive (RED II) (EU) 2018/2001

The first report has a more global approach to the ECs than the second one. The main topic of discussion is renewable energy, and it states that ECs have a significant role to play in the energy transition. In that case, countries should provide information, technical support, and financial support about the creation of such communities. Administrative requirements should also be reduced. The European Council also recognizes and points out the fact that ECs help for *"greater participation by citizens in energy transition"*. Indeed, it is the citizens who make the choice of starting an EC. And the more effort they put into it, the more results they will see over time.

Common Rules for the Internal Electricity Market Directive (IEMD) (EU) 2019/944

This directive continues the work of the first one by describing more laws in favor of ECs, Smart Grids (SG) and smart metering systems. The European Union (EU) recognizes that *"ECs initiatives demonstrate their potential to facilitate the uptake of new technologies and consumption patterns,*

⁴<https://www.rescoop.eu/>

including smart distribution grids" and "ECs can also advance energy efficiency at the household level and help fight energy poverty through reduced consumption and lower supply tariffs" (discussion 43 of the 2019 directive). This gives hope for more laws that will help the creation and productivity of ECs in the next few years.

Then, the directive also states that all categories of entities could be members of an EC. It is thus not reserved to the citizens only to take part in communities but also for organizations like companies, non-profit associations, and so on.

The importance is set on their place in the energy market. It is written that they should play at the same level as normal energy providers, with the same laws and rights.

Finally, countries should promote the use of smart grids to decentralize energy production and increase efficiency. Same for the promotion of smart metering, which engages better consumption from the user by receiving near-real-time feedback on their energy consumption. Article 54 of the directive states that countries should install smart meters for anyone who wants them under *"fair and reasonable conditions"*⁵.

An important thing to note in this directive is the differentiation of ECs into two sub-categories. We therefore have the Renewable Energy Community (REC) introduced by the RED II directive and the Citizen Energy Community (CEC) introduced by the IEMD directive, as described in Section 1.3.3.

1.3.2 Belgian legal framework

In Belgium, the European directives are applied differently depending on the three regions. Each of them defines more refined sub-categories for ECs to match the context of their situation.

The majority of the Belgian legislation information was taken from the Rescoop energy cooperative [5]. It is a Belgium-based European non-profit federation of citizen energy cooperatives. It is deployed in 8 different countries as *"board of directors"* and votes in a democratic way for its general objectives. The main purposes are to *"represent the voice of citizens and energy cooperatives to European policy makers"*, provide support for starting and established energy cooperatives, and *"facilitate international exchanges and cooperation between energy cooperatives"*. There is also a score given to each region in the European Union (EU) that has implemented the ECs in their laws.

⁵These terms are highly interpretable and intentionally left that way to allow the countries of the European Union to implement the article as they understand it.

The Rescoop score is divided into two parts. The first one is for the definitions of the ECs and the covering of the two European decrees. The second is more practical and evaluates the frameworks and support that are set up to encourage the creation of ECs.

Each region has a different independent energy regulator. Their goal is to provide support for consumers in the energy market. They all encourage the deployment of energy communities and try to follow the two European directives. The regulator of Brussels is Brugel⁶, Flanders has the Flemish Regulator for the Electricity and Gas Market (VREG)⁷ and for Wallonia, it is the Walloon Commission for Energy (CWaPE)⁸ in charge.

Brussels

For Brussels, the Rescoop score is 4/5 for both the general definitions and the enabling frameworks. The Energy Cooperative approves the designated authority to oversee Brugel. It has the power to maintain the guidelines of the ECs and can withdraw a license if the community fails to meet its commitments. The EU framework on energy sharing and energy communities was transposed with the ordinance of March 17, 2022, published on April 20.

Brussels has the most definitions out of the three regions. They are well written and specify clearly the goals of the ECs in terms of autonomy, openness, voluntary participation, and transparency. Indeed, the criteria for participation are objective, transparent, and non-discriminatory, thus ensuring the open and voluntary side of the ECs. Autonomy and effective control are covered with specifications for every type of community. All legal entities are allowed, but subject to conditions to ensure that the main goal of the EC's is preserved. Speaking of purposes, all communities must describe their interpretation of the objectives in their statutes, which are to pursue environmental, social, or economic objectives over the pursuit of financial gain.

The criteria of the EU definition are well reflected in all types of EC. We notice that Brussels goes beyond and makes a distinction between energy sharing and energy communities. Sharing cannot aim at making a financial profit. Energy sharing includes peer-to-peer trading and collective self-consumption if the active consumers are located in the same building. Brussels also defines a new type of EC, which is the Local Energy Community (LEC), which is a subset of the

⁶<https://www.brugel.brussels/>

⁷<https://www.vreg.be>

⁸<https://www.cwape.be/>

REC and meant to pursue local renewable generation of energy. All these definitions (4 in total) have good coherence and show that Brussels has gone beyond the EU requirements. Note that the different types of ECs are described in Section 1.3.3.

For the enabling framework and support schemes, the legislation does not cover most elements that must be addressed. Only the basis is covered, although a more detailed version is planned to come. The great news is that a non-governmental organization, "Energie Commune"⁹, has been mandated to provide information and advice for future changes. "Brussels Environnement"¹⁰, the energy public administration, has been given the task of performing an assessment of potential and barriers to the development of EC, but there is no follow-up at present time. Concerning the fair, proportionate, and transparent registration and licensing procedures, Brugel has made it very clear and simple to register a new EC. Registration is mandatory, but the documents to be provided are well-made, simple, yet sufficient to ensure that ECs are based on organizational principles and are respected. This should help build public confidence in communities and encourage them to join and create new ones.

One of the points the EU mentions for the development of the EC's is the removal of unjustified regulatory and administrative barriers. At the moment, no supplier license is required to share electricity between members, but it must balance responsibility or delegate responsibility to a third party. There is also the non-discriminatory treatment of market participants that is mentioned. Indeed, ECs are by definition market participants in the sense that they can sell their excess electricity on the grid. For that, they need some special treatment to ensure that their insertion in the market is secure, but no details are written about it. Finally, accessibility for low-income or vulnerable households and ensuring citizen participation are totally forgotten in the decree.

Wallonia

The Rescoop Walloon score is 4/5 for the general definitions and 3/5 for the enabling frameworks and support scheme. The CWaPE is well noted in terms of influence by Rescoop. The documents to fill out for the creation of a new EC are complete, and it is specified that the CWaPE may use the data to monitor the development of ECs and also to fulfill any legal or regulatory mission assigned to it. The transposition of the second European decree was done on May 5, 2022 [6].

⁹<https://energiecommune.be/>

¹⁰<https://environnement.brussels/>

The EU definitions are well reflected in the Walloon interpretation, which is open and voluntary membership, autonomy, eligibility, and effective control for both the REC and CEC with only minor changes. We do note that some points are emphasized.

For RECs, energy companies and large businesses are excluded from participation. For the CECs, the restrictions are less severe, and these actors can participate, but not in effective control. The latter point is also clearly defined. Only the participants located near the production facilities that it owns or over which it has a right to use can have control over the REC. The Walloon government also defines their interpretation of proximity, which is primarily defined in terms of eligibility to perform energy sharing, and it is the government that regulates the requirement of proximity.

The autonomy of the ECs from its members is developed beyond the EU definition in the Walloon draft. An EC is autonomous if *"a member or shareholder does not hold, alone or jointly with one or more related persons, fifty percent or more of the energy community's voting rights"*. It also specifies some special cases where the EC is not deemed autonomous, for example, when an entity or a natural person has too much control over the EC and can dismiss the majority of the members of the management. At last, it also mentions independence from the other market players.

The purpose of the ECs, which is the environmental, social, and/or economic objective, is clearly defined, like in EU decrees. We do note that the acceptance of the legal entities and citizen participation are not mentioned and thus not ensured.

This has led Wallonia to define two definitions in the Walloon decree : the REC and the CEC. The interpretation of the communities mostly refers to the European one, thus maintaining the coherence between the definitions.

Concerning the enabling frameworks and support schemes, the Walloon government has communicated objectives for EC's, but concrete measures still need to be developed.

For the process of reducing the admin, the ECs don't need any electricity supply license but still need an authorization to start sharing electricity and are responsible for invoicing the shared electricity. The government has room to simplify all of these procedures but has not yet done so.

On the registration and licensing procedures front, the CWaPE has established a well-designed registration process with a form to submit on their website. All ECs whose purpose is one or more activities on the electricity market must be

notified to the regulator before the start of the activities. The EC also has to make a request to the network operator via a standard form made by the CWaPE in consultation with the energy providers. This ensures a transparent and fair registration of any new project of EC.

We note that the government is empowered to set up a financial support mechanism for the EC. However, the decree only references energy sharing, and no real action has been set up. There is also no mention of non-discriminatory treatment as a market participant or accessibility to low-income and vulnerable households.

Flanders

Flanders has really good definitions and translations of the European decree with a 4/5 Rescoop score, but it lacks practical support to encourage the creation of projects and has only a mark of 2/5. The transposition of the EU decree was accepted in April 2021 and put into practice on July 9. Overall, the Flemish decree is highly open to interpretation, and the VREG is keeping a close eye on developments.

Regarding the definitions, it has been chosen to combine the definitions of active consumer and self-consumer as one. The same was applied to CEC and REC. The goal is to ensure coherency around the concept. However, they do insist on the goals of the ECs, which are to strengthen the involvement of citizens in the energy transition to have better social acceptance and involvement in the energy market and combat energy poverty. Creating ECs is not for economic purposes only. This definition still remains vague and is subject to interpretation.

In the end, Flanders has only two definitions in their decree, one for the consumers and the other for the ECs. These definitions are open to interpretation, but they remain very consistent with each other. It has great potential, if the VREG does a good job with strong monitoring to ensure coherence with the original concept defined in the EU decree. It is written that each EC has to notify the VREG of their existence, goals, and how they interpret the concept of technical and geographical proximity. These interpretations are published on the regulator's website to ensure the concept is not abused and enhance trust. There is still a strong risk of corporate capture and abuse. The Rescoop opinion on that is "*A more rigorous screening process around compliance with the criteria to be registered as an energy community is necessary in order to ensure citizens can maintain trust in the energy community concept*".

The participation of legal entities in ECs is not clearly defined. *"Most likely, only cooperatives and not-for-profit organizations comply because of the criteria that apply to RECs and CECs"*, says Rescoop. The open participation of the citizen is also not ensured, but it was stated that the objectives of the ECs are to *"strengthen the involvement of citizens, local authorities, non-commercial institutions, and companies in the social acceptance of the energy transition"*.

Concerning the enabling frameworks and support schemes, the Flemish government still needs to develop a number of measures to make the launch of new ECs more attractive. A good beginning is that the VREG and the Flemish minister of energy have a duty to carry out an assessment of the potential and barriers to the development of ECs.

The important supports that were not spoken of are the removal of unjustified regulatory and administrative barriers, non-discriminatory treatment as market participants, and accessibility to low-income and vulnerable households. On the other hand, the support schemes adapted for the RECs are really good. The federal government has foreseen the possibility for citizens to participate in the offshore wind tender project. Local municipalities are encouraging the creation of new ECs with direct participation in the projects.

1.3.3 Types of Energy communities in Belgium and Europe

The Citizen Energy Community

The CEC is a legal entity that enables individuals, local authorities, and small companies to get involved in the electricity sector and benefit from it. This is a large definition since it doesn't necessarily involve renewable resources or geographical proximity. It can combine different energy manipulations, such as generation, supply, storage, distribution, and consumption. Participation must be open and voluntary, while control can only be exerted by a local authority or a small company. So big companies and energy sector enterprises cannot be part of the controlling members but can still be members of the community.

The Renewable Energy Community

From the explanations in the 2023 paper about citizen and renewable ECs [7], the REC is a more restrictive type of EC in certain aspects. The main motivation behind the creation of this concept is the opportunity for citizens and groups to participate directly in the development of renewable energy sources and benefit from them. This is not exactly the same goal as the CEC mentioned earlier. In addition, RECs are not limited to electricity only. They can, for example, deal with heating from geothermal energy.

Regarding the participants, large companies and enterprises that are related to the energy sector are forbidden to take part in the project. In addition, the controlling authority must be close to the geographical location. Lastly, the community must be autonomous for its members. All of these rules for the members emphasize the philosophy of a local and decentralized initiative to take part in the energy revolution in a democratic way. Like the CECs, RECs have the possibility to produce, consume, store, and sell the renewable energy they produce. With all these rights, countries are also requested to enable frameworks that promote and facilitate the development of RECs. The goal is to remove all unjustified regulatory and administrative barriers. The differences between the two types of EC are presented in figure 1.1.

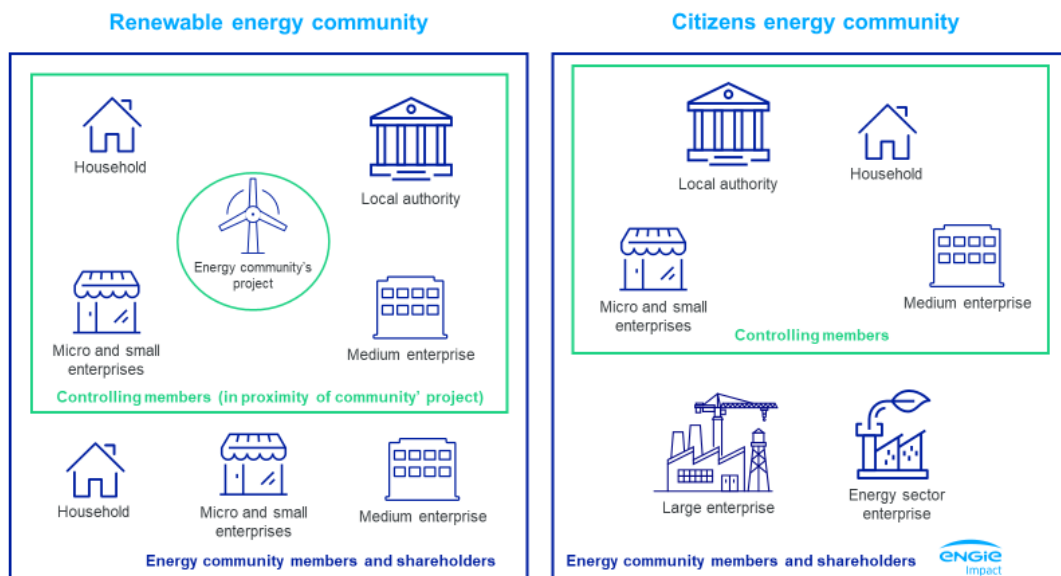


Figure 1.1: Membership and effective control criteria of RECs and CECs [8]

The Peer to Peer Trading

From [9], peer-to-peer trading (P2P trading) is defined as a horizontal transaction of electricity between two consumers or prosumers. The term *"horizontal"* emphasizes the fact that the participants in a P2P network are on the same level for the exchange of electricity. This differs from the traditional *"vertical"* transactions between one electricity producer company that sells its energy to many consumers (see figure 1.2 for a visual representation).

The RED II directive also defines P2P trading in its own way as *"the sale of renewable energy between market participants by means of a contract with pre-determined conditions governing the automated execution and settlement of the transaction, either directly between market participants or indirectly through a certified third-party market participant, such as an aggregator"*.

This transaction type has emerged due to the feed-in tariff reduction in recent years. It has become more advantageous to store the energy produced by solar panels for later use rather than resell it on the grid. This poses a problem for the European Union, which sees consumers as *"the center of the future energy systems"*. Thus, P2P trading and ECs emerge as *"a promising solution to empower the role of the end-users in energy systems"* [10]. Countries should allow P2P trading to sell their energy at a lower price than the feed-in tariff to make it advantageous for the consumer, but at a higher price than the retailer tariff for the prosumer. This way, P2P trading achieves a win-win situation by making flexible energy trades between peers with better economic efficiency.

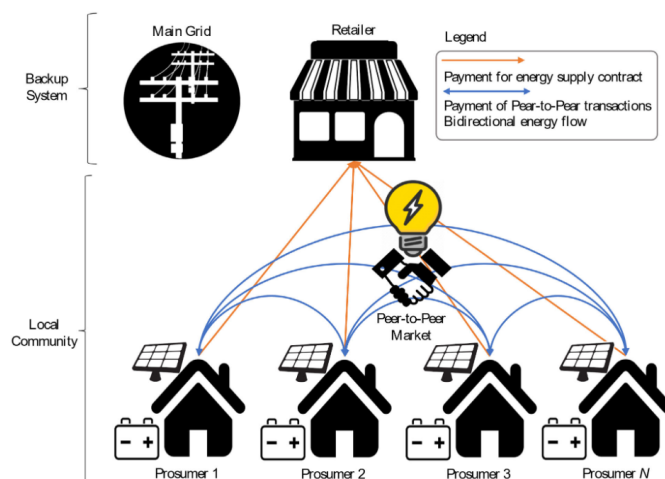


Figure 1.2: Example of a P2P community of prosumers [10]

The Collective Self Consumption

Article 21 of the RED II introduces two new definitions: renewable self-consumers and jointly acting renewable self-consumers, better known as collective self-consumption (CSC).

A renewable self-consumer is *"a final consumer who generates renewable energy, including for their own consumption, stores, and sells their excess production of renewable electricity, including through renewable power purchase agreements, electricity suppliers, and peer-to-peer trading arrangements"*. One point to mention is that the sale of excess production may not be the primary commercial or professional activity and shall not be subject to discriminatory or disproportionate procedures and charges.

Jointly acting renewable self-consumers are a group of at least two renewable self-consumers who are located in the same building or multi-apartment block.

The Local Energy Community

The last category of EC is introduced by the Brussels region decree and is therefore only possible there. As explained in the web pages of Energy Communities Hub [11] and Rescoop [5], the Local Energy Community (LEC) benefits from the agency of Brussels. An example is when many roofs are shared and a third-party investor in photovoltaic panels can participate in the EC. This offers wider possibilities than the other types of ECs, but needs stronger monitoring from Brugel to prevent deviations from the original concept. We do note that the government can stipulate further conditions and details on a case-by-case basis during project creation.

Because of the potential participation of third-party investors, the rules of energy production, consumption, storage, and sharing are limited to the community only. The production of energy is also limited to renewable solutions. This helps recall the primary objective of the ECs, which is the sensitization of the citizen to the energy transition, even when they are less involved in energy production because of the third-party investor.

Chapter 2

Setting up the online data transmission

2.1 Introduction

As discussed in the Section 1.2 about the problematic of WeSmart, there is a need for real-time data gathering. In this chapter, we talk about the possibilities of how ECs can send their consumption data to a remote server and analyze it. We focus here on the emitting part. We explore the concepts of smart grids, smart meters, emitters, and communication protocols and how the development around them creates possibilities to send and monitor data.

Note that the explanations about the communication protocols and the chapter on extrapolation are not relevant if the analysis of the data is done locally in the EC. Indeed, in our case, everything is centralized on the Wesmart server so that the company can use the data to show the results on their digital platform, specific to each EC that they oversee. If an EC wants to monitor its own data, they can do it easily with the already existing applications linked to the Internet of Things and don't need to send it through the internet.

2.2 Smart grids and smart meters

Smart grids (SG) are one of the priorities of the Trans-European Networks for Energy for the integration of renewable technology. They are also a very good solution for consumers to better regulate their energy consumption through smart meters (SM). They are advanced utility meters that digitally monitor and record energy consumption in real-time, which is at the center of this study. According to the official EU website [12], SGs are energy networks that can automatically monitor energy flows and adjust to changes in energy supply and demand accordingly. They are also designed to better manage the variations created from renewable sources and decentralized energy creation.

SMs are massively replacing the old analog meters in our homes. According to the commission opinion of April 30, 2020 [13], 100% of customers in Flanders shall have a smart meter by the end of 2034. 80% of smart meters will be installed in Wallonia by the end of 2029, with a priority for specific cases. For the Brussels region, almost all residents should have a smart meter installed by the end of 2030, according to Sibelga, the distribution network operator for electricity and natural gas in the region¹.

To begin the data monitoring, the owners of SMs have to connect a transmitter to the P1 port of the SM to see their consumption in real time. This helps them see which activity consumes the most and reduce their use. This is an application of the Internet-of-Things (IoT) technology. In our case, we send the data to a server and make our observations remotely.

2.3 Transmitters

Several different transmitters are available on the market. They all do mostly the same job, but some come with different visualization applications to monitor the data. The following sections describe their specific features to help us choose the best transmitter for our project. First off, we make a theoretical review of the available transmitters in Belgium. Then, we make a practical comparison between the two transmitters that WeSmart has chosen. We also finish the section with a review of the different communication protocols to send the information remotely.

¹<https://www.sibelga.be/en/connections-meters/smart-meters/whos-eligible-for-a-smart-meter>

2.3.1 Theoretical review

There are various transmitters that were found on the net and that seem suitable for our use. They are presented in the following table, which lists different points to look up when choosing a transmitter.

Table 2.1: Technical informations of the transmitters, May 2024

Model	Price	Connection	Display app
P1-dongle	< 20€	WI-FI	Yes
HomeWizard P1-meter	30€	WI-FI	Yes
EasyMETER P1	59€	WI-FI	Yes
Smart Meter WI-FI Gateway	35€	WI-FI	Yes
SweenCom USB	+150€	WI-FI/4G/Eth	SweenApp

Model	Open-Source	Raw data recovery
P1-dongle	Yes	MQTT with TLS
HomeWizard P1-meter	No	Open API
EasyMETER P1	No	Unknown
Smart Meter WI-FI Gateway	No	MQTT with TLS
SweenCom USB	No	MQTT with TLS

The SweenCom USB stands out from the rest because of the connection type, the display application, and also because of the price. It is estimated to be 150€ if it works on the P1 port only, but it may still need to be adapted. The fact that it can work with different types of internet connections is a really good advantage in cases of connection issues, for example, when the smart meter is in a cave or when no wireless router is available. However, the display application is fixed to their product, which is not a good deal for the problem we want to solve.

The base price for the product is 396€ with the possibility of discounts for large orders. This is explained by the different usage scenarios. It can retrieve data directly from the photovoltaic panels but also *"control a stationary battery or a car charging point using a collective logic (smart charging controlled by the energy available in the community)"*, according to Sween. These can be really interesting options, but we are not interested in them in our case. The SweenCom transmitter is further explained in practice in Section 2.3.2.

For the rest of the transmitters, the P1 has the lower price. The openness of its software and its construction is a real advantage for a company to make its own version that meets its specific needs. However, they have to respect the conditions of the open source philosophy and make their code open to everyone (see Section 2.3.2).

One thing to mention is that all the reviewed emitters (except the Sween one) work only with WI-FI behind a private network, like normal IoT devices. For this type, the connection is simple, and we just have to connect to them via Home assistant² or all the other smart applications available online. It is thus not planned to send packets to remote servers. However, some transmitters have other operating options.

The HomeWizard P1-meter has an open API that allows to recover the data in JSON format by sending HTTP requests. If the WI-FI router is configured, it should be possible to send HTTP requests from the server, but the data is not encrypted, so everyone on the net can access it.

The EasyMETER P1 home page just talks about compatibility with Home Assistant; no information about other types of connections is provided.

The Smart Meter WI-FI Gateway from Smart Gateways is compatible with many IoT applications (Domoticz, Homey, DSMR-Reader, etc.) but is also capable of using the MQTT protocol with a TLS option if on the web. More information about the communication protocols is available in Section 2.4.

With all this information in mind, we can conclude that the HomeWizard P1-meter and the P1-dongle are the best choices among those available for their price and ease of recovering raw data. For the rest of the transmitters, they sometimes have their own particularities, but the general principle remains the same.

2.3.2 Practical review of the available transmitters

We were able to test two emitters from the list presented. The P1 dongle and the SweenCom USB. They were provided thanks to our collaboration with WeSmart, which had contacts with Ideta, and a collaboration with Sween to get free devices for testing purposes in exchange for platform developments. It is a great thing that we could analyze these two emitters, but this is also a constraint from WeSmart because these are the only ones that we could review without even selecting them ourselves.

²<https://www.home-assistant.io/>

P1 Dongle

Before setting the dongle up, we made some minor changes to the code so that all the information we wanted was sent in one MQTT packet to the topic of our choice (more information is in Section 2.4.3). The modifications were pretty easy to do because the project is entirely done with the Arduino language³. The dongle is a modified ESP-32⁴. The changes made weren't compulsory, but they helped to retrieve the data more easily from the backend in our case. The modified Arduino code is in the annex A.1.

After uploading the code to the chip, the last thing to do was set up the dongle so that it could start transmitting. There is a well-written tutorial available in the Github repository⁵ of the project, and it explains in detail every step of the installation.

To test the dongle, we had the opportunity to work with Ideta⁶, an agency focused on inter municipal development in Wallonia. WeSmart is working with them on a different project from ours, but also involving real-time data retrieval. WeSmart could get some assembled P1-dongles to test for free. While testing the emitter, we had the help of one of the company's employees, which saved a lot of testing time. We note that the sending of the data is not 100% stable and there are occasional losses that need to be fixed.

As the writer of this TFE and on a personal basis, I think that the open source aspect of this transmitter is a great added value to the project. If its use becomes widespread in ECs and among individuals who want to monitor their energy consumption, there could be an open-source community developing around the improvement of its features and its durability. There are tons of projects that were created this way, and Home Assistant is one example that was already cited in this writing that is a successful open source project with its community improving it continuously.

³<https://www.arduino.cc/en/software>

⁴<http://esp32.net/>

⁵<https://github.com/plan-d-io/P1-dongle/wiki>

⁶<https://ideta.be/>

SweenCom USB

The installation of the SweenCom module is really straightforward if the usage of the emitter is limited to the P1 port. There isn't any help needed from the company to place it. However, all the data recorded is automatically transferred to their servers, and they then send it to us afterwards. This is explained by the fact that their module was not designed to be used by a third party. It would need some refactoring of the internal code of the emitter to redirect the data flow, but the UCLouvain and Wesmart don't have the rights to modify it.

After connecting the device to the SM, we did not receive the data directly because Sween had to create the redirection of the flow. At first, they had a small server bug, which prevented them from sending us the data for several days. Once the problem was fixed, we received the data for several days. After a moment, we had an issue that appeared after a long connection time. We asked Sween to solve the problem, and the communication resumed not long after the reporting.

There were many periods of varying length where there was no consumption at all. If there is no change, the SweenCom USB sends the same MQTT packet in a loop, with the same timestamp as the first packet sent with the last update of value. This ensures that the packet has been received and allows us to know what the last value sent was.

The P1-dongle works differently in this case. We have to set the time between each transmission, and for each delivery, the dongle takes the timestamp at which it sends the packet, even if there are no changes.

The fact that we have to ask Sween to solve problems for our usage has advantages and constraints. If there is a bug, the user of the SweenCom USB (WeSmart) doesn't have to invest resources to solve the problem. On the other hand, there is a dependence on external resources, and there's nothing else to do while the bug is being dealt with other than wait. We also note that the various stages the data goes through are all an addition to the risk of bugs that would prevent the data from reaching its destination.

Choice based on the practical review

The most promising transmitter between the two is probably the P1-Dongle, although they all do the same job. The reasons are the openness of its software and build, which allow production on a greater scale, and also code modification to meet our needs. We're also not dependent on an external company like Sween,

who are the only ones who're able to solve a potential problem if there is one. The P1-dongle is not perfect either but has more room for development in the direction we want to invest in.

2.4 Communication protocols

Communication protocols (ComP) for IoT are essential to enable devices to exchange data efficiently and securely. These protocols define the rules and standards for how devices communicate with each other and other systems over networks. Each protocol has its pros and cons. Heavy ComPs tend to be more robust and reliable in terms of packet loss, security, and error handling. They also often offer flexible ways of tuning to fit the requirements. Lightweight ComPs, on the other hand, are generally more efficient regarding message size, network bandwidth, and power consumption. They are also less complex and are easier to implement and manage.

In our case, the ideal ComP should have these characteristics:

- **Lightweight:** The sending of packets should not take too much network bandwidth to avoid disrupting the WI-FI (or 4G) network. As said previously, being lightweight also means easier management. This would allow for better management of the desired behavior and fewer potential bugs. In any case, the quantity of information to send per packet is very low, so it would be counterproductive to have a bigger packet header than the carried information.
- **Secure:** The data is sent over the web to reach the server and don't stay behind WI-FI routers. The type of data sent is user-sensitive, so it must be encrypted to preserve privacy.
- **No loss of information:** The communication must ensure that every piece of information has arrived at its destination. In our case, we have a very low packet-sending frequency compared to the usual IoT usage. We tend to send one packet every minute, fifteen minutes, or even every day (more about that in chapter 4). If the server misses a packet, it has to guess what happened or wait until new information arrives. This would cause a chain decrease in robustness for the layers of data analysis and can be easily prevented with packet acknowledgement (ACK). This is the opposite of streams, where we want high data debit and don't care about some loss of information during the transport (within reasonable limits).

2.4.1 Review of the communication protocols

We present here some popular protocols in the IoT industry to get a better vision of the possibilities and choose the best option for the project. The list is not exhaustive since there are too many protocols to review and would need an entire study to compare them all. Instead, four of the most promising and used ones were selected for investigation based on a review of 2022 [14] which are :

- Message Queuing Telemetry Transport
- Constrained Application Protocol
- WebSockets
- Extensible Messaging and Presence Protocol

Message Queuing Telemetry Transport

According to a 2017 survey about the protocol [15], Message Queuing Telemetry Transport (MQTT) is a standardized publish/subscribe push protocol. In this configuration, there are two main components:

- **Broker:** It is the central part of protocol. It controls the distribution of information. It receives all messages from the clients, filters them, decides who is interested in them, and then sends the messages to all subscribed clients.
- **Clients:** They can be publishers and/or subscribers. They always establish the network connection with the broker.

Figure 2.1 illustrates the broker and client communication. Note that the Device/Sensor is also a client of the broker and publish/subscribe to topics at the same level as the others.

On setup, each client subscribes to a specific topic of the broker to receive any message that is sent from other clients to the same topic. This is the method used for group chats. When one client sends a message to a topic, the broker redistributes the message to all the subscribers.

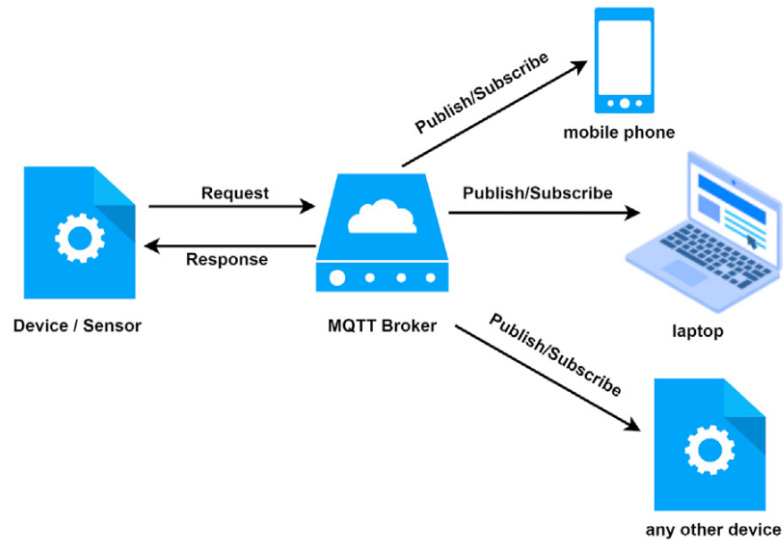


Figure 2.1: Structural behavior of MQTT protocol [14]

This protocol defines quality of services (QoS) levels. This is the amount of assurance we want for an information to be distributed correctly. There are three levels of QoS :

1. **QoS0, At most once:** The message is sent at most once and there is no guarantee of message delivery. This is the most lightweight option.
2. **QoS1, At least once:** As said in the name, the communication is set for the message to be received at least once by the subscriber. It means that duplicates can occur.
3. **QoS2, Exactly once:** This type uses the most bandwidth out of the three levels because the sender has to ensure that every messages is received exactly once. It can also add delay in case of low connection.

Reliable sessions are also defined in MQTT. If a client has connection problems and gets logged out for a moment, the broker can send the missed messages again when the connection resumes. This means that the broker saves all data entries.

Finally, MQTT supports Transport Layer Security and Secure Sockets Layer (TLS/SSL). When activated, the data will be encrypted so that only subscribers with the encryption key can read the messages.

Constrained Application Protocol

From research on the domain over different papers [14], [16], [17], the Constrained Application Protocol (CoAP) is a client/server protocol. For this configuration, the client always initiates the communication. In our case, the client is the emitter because it will send data to the server and has to initiate the communication to begin the transmission.

The framework is based on the Representational State Transfer (REST) model, like HTTP, and the transport protocol is the User Datagram Protocol (UDP). The REST model is based on a request/response model, where each piece of data sent by one side should be asked for by the other side. UDP is a transport layer protocol that is at a lower level than the protocols presented. It is often compared to the Transmission Control Protocol (TCP). The characteristics of UDP in contrast to TCP are the following:

- **Connection Less:** Each packet is sent independently, and there is no guarantee of delivery or receiving order.
- **Packet Structure:** Packets have smaller headers than TCP. It helps reduce the use of bandwidth and improves the latency.
- **Overhead:** UDP uses less overhead, which is the additional data resources beyond the essential information. This is linked to the fact that it is connection less and also minimizes latency.

As for MQTT, CoAP also includes different levels of QoS, but only levels 0 (at most once) and 1 (at least once). This is sufficient for our application because the server gets the timestamp of each packet, so it can easily remove all duplicates. More information is in Section 2.4.3.

In terms of security, CoAP uses Datagram Transport Layer Security (DTLS). This is a hot topic in the field because there are areas where DTLS is lacking and can present a threat to security [18].

WebSockets

WebSockets are an improvement of the traditional HTTP protocol. It uses the client/server protocol but without the request/response mechanism. For the setup, the server or the client has to initiate the connection with the other. Every side can send data once the connection is initiated, as we can see in figure 2.2. The connection will remain until the client or the server-side party closes it actively or if there is a bad network connection [19].

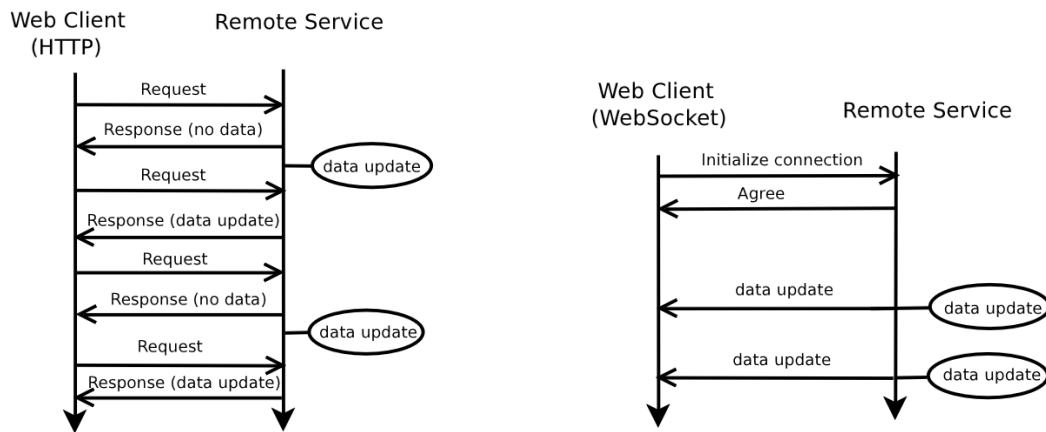


Figure 2.2: Comparison of communication between HTTP (left) and WebSockets (right) protocols [20]

The communication is done on top of the TCP transport layer, so there is a guarantee of delivery, but with a loss of speed and increased network bandwidth. TCP and connection maintenance are good news for cases where there is a bad network connection. If a packet is lost during a connection error, the client or server will notify us and can handle it in various ways depending on how we configured it. We can program the protocol to reconnect automatically with exponential back-off strategies. The principle is to increase delays between re-connection attempts and resume sending where there was a problem.

A good point about web sockets is the versatility of their behavior. We can specify the reaction of client/server in any given case, but this can increase the risk of potential bugs and require more time to set everything up.

Concerning the encryption and security of the data, webSockets uses TLS as the security layer. This is the same method as MQTT and also used in the secure version of HTTP protocol which is HTTPS.

Extensible Messaging and Presence Protocol

The Extensible Messaging and Presence Protocol (XMPP) is an IP-based protocol that supports the publish/subscribe mechanism. The connection is thus persistent, and either side can send data asynchronously with no packet loss. The data transfer is encrypted with TLS, and the protocol has set up authorization, access control, and privacy measures to keep the data private on the net [17]. The structure of the information is encoded in XML to keep the messages structured, with extensibility possibilities like the JSON format.

Originally, the framework was designed for a type of web where the equipment is rich in resources. This led to a protocol that is heavy on resources compared to others, like MQTT.

As said in the article of Wang & Al. in 2017 [21], *"The XMPP protocol needs to be optimized to meet the requirements of IoT, e.g., downsized protocol, publish/subscribe service, and sleeping mechanism"*.

XMPP systems have 3 different actors for interoperability, an illustrative example can be seen in figure 2.3:

- **Server:** It has 3 responsibilities:
 1. Save the messages of the clients
 2. Manage connections between entities
 3. Route the messages between clients

These possibilities of action reminds us that XMPP is an internet protocol at first, before drifting to the IoT world.

- **Gateway:** Used to communicate with the different versions of the XMPP protocols. The Gateway is placed between two servers that wants to pass messages of their respective clients
- **Client:** (IoT) devices that communicates between each others through the servers and Gateways.

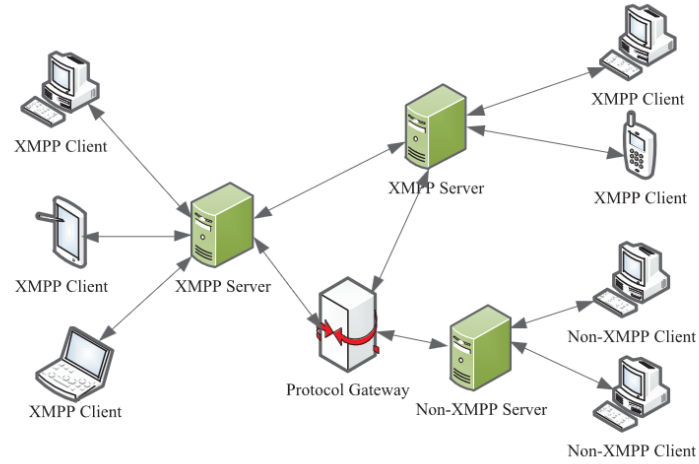


Figure 2.3: XMPP communication system architecture [21]

Note that in our case, we don't have the need for gateways since we program the two sides of the application, which are the server and the client.

2.4.2 Choice of the protocol

We present here a table summing up the principal characteristics of the ComPs explained previously:

Table 2.2: Communication protocols characteristics

Protocol	Behavior	Security	Transport layer protocol
MQTT	publish/subscribe	TLS	TCP
CoAP	request/response	DTLS	UDP
WebSockets	request/response	TLS	TCP
XMPP	publish/subscribe	TLS*	TCP

We are more interested in a publish/subscribe model because the data transfer is one-sided, like in our case. It is easy to receive the packets when they are ready without doing anything on the server side. However, the request/response model can also be considered if the server receiving the data sends requests for updates, which is not too difficult to implement.

On the security side, the DTLS protocol might present some security breaches. For the rest, TLS is a good option, and the XMPP protocol further enhances security with access control, privacy measures, and other advantages.

For the transport layer protocol, TCP is a better option for packet loss and bad connectivity. Like for the request/response model, we can also get the same results with UDP with a level of QoS of 1. At this level, we would need to filter the duplicate packets on the server side, but it is not difficult to implement. The disadvantages of TCP compared to UDP are not relevant in our case because the transmission speed of the packets is completely negligible for the highest sending frequency of 15 minutes per packet. The same applies for network saturation because of the limited amount and size of the data transmitted.

This leaves us with two options, the MQTT and XMPP protocols. As noted in the review of XMPP, the framework was originally designed for resourceful web applications. We also note that the emitters don't need powerful processors to transfer the electricity usage and remain economical, like the ESP-32 on the P1-dongle. This is why it is preferable to use a lightweight protocol. XMPP also needs to be further optimized for IoT requirements, unlike MQTT, which is more recent but more suitable for IoT thanks to its extremely lightweight format.

With all the requirements for our needs and the characteristics of the protocols, we can conclude that MQTT is the most suitable protocol for our needs, regardless of the fact that it is used by the two reviewed transmitters. However, this also confirms our theoretical choice, since we have reached the same conclusions as Sween and the open source community.

If MQTT was not available for use, we would have chosen XMPP in the hope that the versatility of this protocol can help us configure a version that can transform the emitter's data into XMPP before sending it over the Internet.

2.4.3 MQTT broker and server set up

As to be expected, the communication protocol of choice for this project is MQTT, which is also the protocol of the reviewed emitters. The broker is set up with Mosquitto⁷ on an Ubuntu server of Digital Ocean⁸. Mosquitto is an open-source command-line tool that implements the MQTT protocol easily. A basic broker is automatically set up right after the installation of Mosquitto and can be adjusted to manage safety with TLS and other parameters.

⁷<https://mosquitto.org/>

⁸<https://www.digitalocean.com/>

The changes made to the P1-dongle code are simple when we understand the structure of the code. Changes were made to the **dsmrTelegram.ino** file to modify the information sent and set the topic to which we send the data. As mentioned in the review of the P1 dongle, the base and modified versions of the code are in the annex [A.1](#). The information transmitted in the JSON format is:

- **totConDay** : The index of electricity consumed at day
- **totConNight** : The index of electricity consumed at night
- **totInDay** : The index of electricity injected at day
- **totInNight** : The index of electricity injected at night
- **timestamp** : The Unix timestamp at which the indexes are raised

For the SweenCom USB, as said before, the data is sent to a distant broker in France. The format of the information is a bit different than the P1-dongle, but the finality is the same. The values, also in JSON, are:

- **timestamp**: Same as the P1-dongle
- **status**: The status of the SweenCom USB
- **cumulative_injected_power**: The index of power injected into the grid
- **cumulative_withdrawn_power_index0**: The index of power consumed from the grid at day
- **cumulative_withdrawn_power_index1**: The index of power consumed from the grid at night
- **productions**: The index of the raw production of the electrical production, not used in our case

The code to retrieve the data is originally done in bash⁹ and the two scripts are essentially the same. It can be found in Annex [A.2](#). There is also a Python script version to directly begin the data analysis in Annex [A.3](#). It is sensibly bigger than the bash one because it's already filtering the data to only keep a 15-minute spacing (the data has an arrival frequency of 1 minute).

The incoming information is saved in CSV files named by the day of the data retrieval and stored in a temporary file for the bash script while it is in creation. Once the file is done being written, it is stored in two different folders, one to work with the data and one for the backup.

⁹<https://www.gnu.org/software/bash/>

2.5 Conclusion

After every step described in this chapter, we finally have data saved on the server in good format and with whom we can work. We've seen that remote data retrieval has been simplified by smart grids and smart meters through the emitters that are designed for smart homes. Although we note that emitter technology can still be improved, the development is on a good path, and emitters will likely become more and more common with time.

There are good options available for communication protocols. MQTT is certainly the best option considering its publish/subscribe behavior, ease of use, and lightweight format. It is also the choice of the developers of the P1-dongle and Sween.

Concerning raw data recuperation and storage, the most widely used tool is the open-source Mosquitto solution. Two different software language approaches were set up to retrieve the data, namely Python and Bash. We note that the Bash one does almost the same job with fewer lines and allows for project structuring and separation of tasks.

Chapter 3

Online data filtering

The problem with online data recovery via the internet is that there may be outages due to various reasons. In the case of the indicator, it would prevent us from making recommendations during this time. For the change of habits, if the interruption happens during the time slot of the data collection, we can't just assume that there was no consumption during the outage.

For this reason, we can try extrapolating the data to achieve better approximations than simply taking the last value seen or assuming that the consumption has remained unchanged. In the next section, we present the possible extrapolation solutions and compare their performance to the constant extrapolation to see if the results are better or not.

3.1 Hole extrapolation

Hole extrapolation is an estimation technique for points beyond the known data set. It is more difficult to have results as good as the interpolation, and the two should not be confused. In the case of time series, interpolation learns from the data before and after the hole, where extrapolation only has the data before.

We might ask ourselves about the use of a prediction algorithm to estimate the data, but it is not the optimal solution because it requires much more computation to produce results, and holes can appear at any time. In our case, we just fit the mathematical function to the latest points of the data and don't take into account many other parameters that influence consumption.

3.1.1 Extrapolation algorithms

We present here linear algorithms that don't require intensive computation, because that would be excessive for data that doesn't follow any simple equation and has little time dependency. As we can see in figure 3.1, electricity consumption in [kW_15minutes] can vary greatly between 15-minute intervals¹. The goal is thus to see if the extrapolation can achieve better performances than keeping the same consumption as the last value seen.

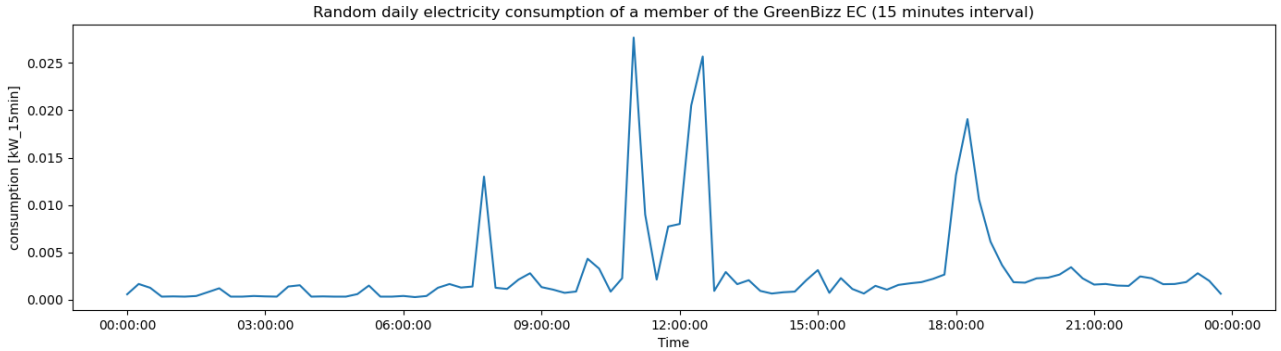


Figure 3.1: Example of a daily electric consumption of a member of the Greenbizz EC

The presented algorithms are the linear and polynomial extrapolations, but the second is just a generalization of the first. We limit ourselves to these algorithms because more advanced ones require better mathematical knowledge about the data, which we don't have. We compare these solutions to the constant extrapolation. These algorithms try to estimate the data as linear functions, even though the data is non-linear, as said before. Other algorithms assume certain properties of the dataset, like the exponential and logarithmic extrapolations, which work best if the data follows these kinds of trends.

¹The unity is [kW 15-minutes], because we get indexes as consumption values. Indexes are power consumption values in [kW] that have been constantly increasing since the beginning of the measurements. From these indexes, we take the difference of the values of the indexes every 15 minutes to get the energy consumed at every 15-minute interval.

Linear extrapolation

The linear extrapolation is the most basic extrapolation. This method can also be used for interpolation and performs best if the data follow a linear trend of the type $y = ax + b$ which is the equation of a straight line.

The formula when fitting the data to two points $p1 = (x_1, y_1), p2 = (x_2, y_2)$ is simple:

$$y(x) = x_1 + \frac{x - x_1}{(x_2 - x_1) * (y_2 - y_1)} \quad (3.1)$$

If we want to fit the line to more than two points, the linear extrapolation becomes a regression technique. This is due to the fact that we can't fit the data perfectly into more than two points if they are not aligned. The equation becomes:

$$y(x) = \hat{\beta}_0 + \hat{\beta}_1 x \quad (3.2)$$

$$\hat{\beta}_1 = \frac{\sum_{i=0}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^n (x_i - \bar{x})^2} \quad (3.3)$$

$$\hat{\beta}_0 = \hat{y} - \hat{\beta}_1 \bar{x} \quad (3.4)$$

We note that the linear regression is indeed a general case of the linear extrapolation, as the equation 3.2 will have the same result as the 3.1 if $i = 1$ (two points).

Polynomial extrapolation

As mentioned previously, the polynomial extrapolation is just a generalization of the previous algorithm. We try to find the best representation of the cloud of data points (which can be a subset of the data set) with polynomials of variable degree d set by us. The formula is:

$$y(x) = \sum_{d=0}^n \hat{\beta}_d x^d \quad (3.5)$$

where $\hat{\beta}_n$ is the coefficient to be found by regression, and we can see that when $d = 1$, we get the equation 3.2.

We need to be careful, though, that the higher d gets, the higher the chances of over fitting the data. We also know that a higher degree of polynomial has a higher chance of getting disproportionate values when we move away from the training data.

Constant extrapolation

This is the simplest technique that can't really count as a real extrapolation. When we have a loss of data, we just assume that the consumption is at the same power as the last reported value. This method also serves as a base to compare the other extrapolations and see if it has meaning to try extrapolating the data.

3.1.2 Practical testing

Holes creation

As said in the introduction of WeSmart in Section 1.2, we can work with the data of two different types of ECs: a group of offices and a flat. For Greenbizz, which is the group of offices, we have 2 years and 6 months of consumption for 6 members. For the EC located in Stockel, 1 year and 5 months of consumption by their 8 members are available.

To test the methods, we have masked values at random places in the datasets, but at the same place for every member of the ECs. The reason is that if we have a break in the data flow, we have no more information for everyone in the EC at the same time. They are spaced far enough apart to ensure that extrapolations are not based on artificial data, which are the filled holes. The holes have variable lengths to simulate different durations of breaks. With 15 minutes of data and holes with a maximum length of 6, we simulate breaks that last at most 1 hour and 30 minutes. These are the number of holes in the two datasets:

Table 3.1: Listing of the duration of data outage

Hole duration :	15'	30'	45'	1h	1h15	1h30
Greenbizz	160	153	147	142	141	132
Stockel	85	78	80	78	63	82

results analysis

The figures 3.2, and 3.3 show the results of the extrapolation of the two EC's. When there is an outage, we look at the previous consumption information of each member of the EC and make extrapolations member by member.

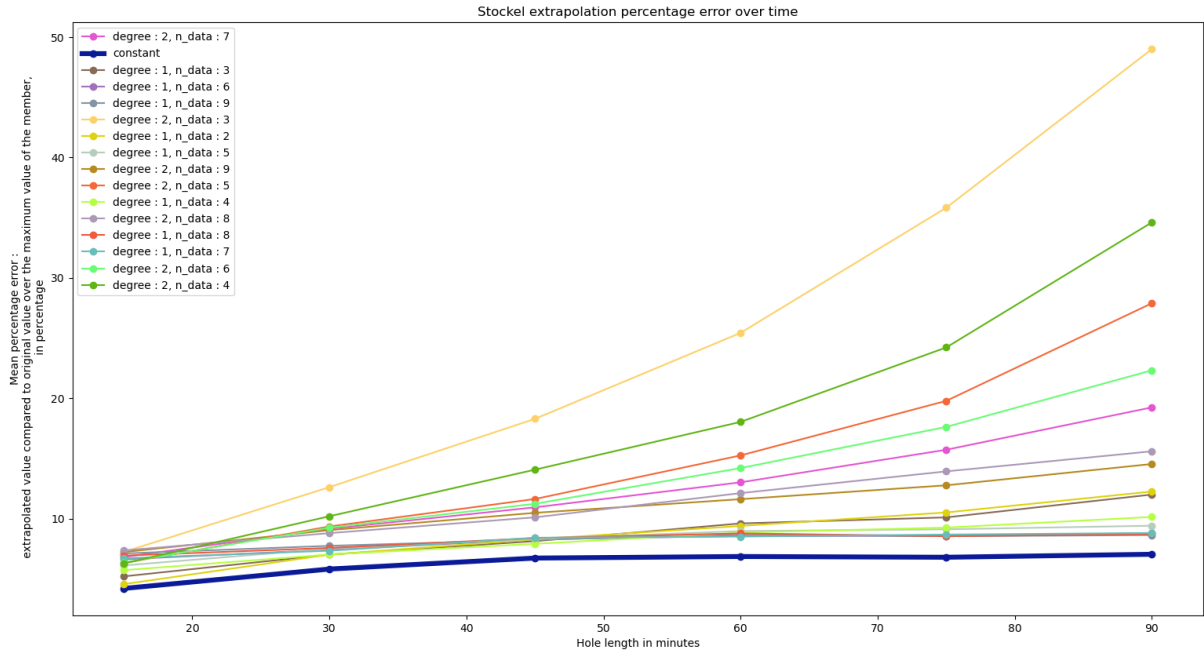


Figure 3.2: Extrapolation results for the Stockel EC

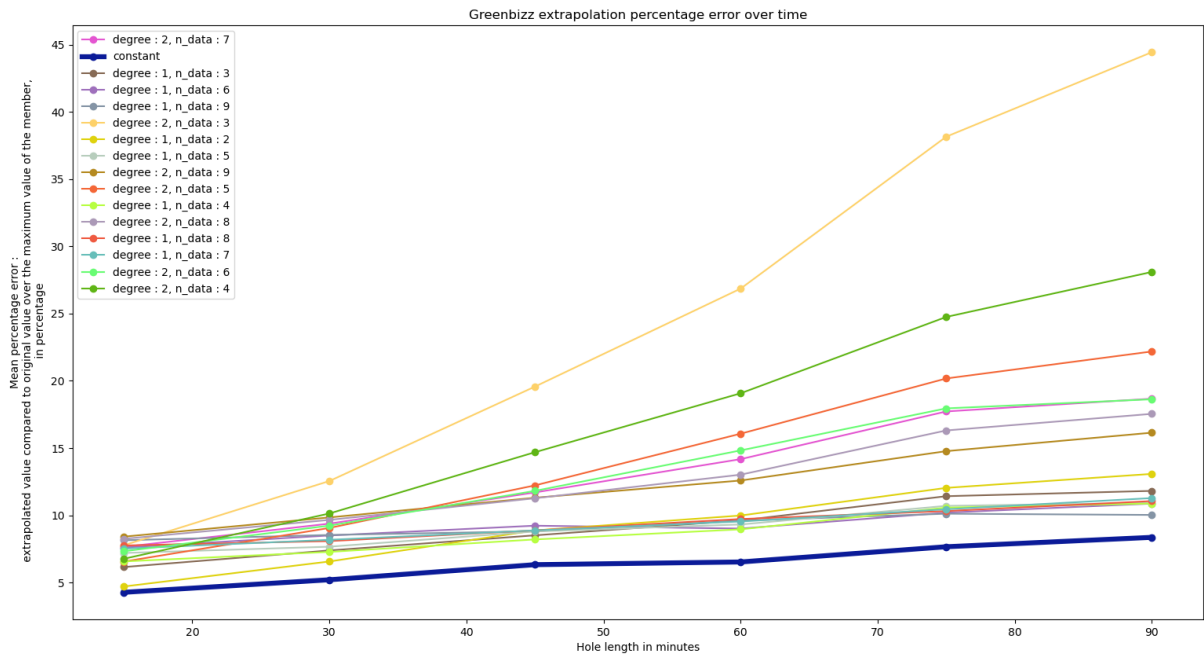


Figure 3.3: Extrapolation results for the Greenbizz EC

We show the results by performance over the size of the break in data transmission, and the error is calculated by taking the mean of the percentage of error with the error formula:

$$percentage_error = \frac{abs(diff) * 100}{maximum_value(column)} \quad (3.6)$$

$$diff = true_value - extrapolated_value \quad (3.7)$$

The *maximum_value* of the column from the equation 3.6 gives an order of how well the extrapolation predicted, compared to the kind of data in the time series. For example, an extrapolation of $1kWh$ and a real consumption of $2kWh$ is a better prediction in the case where values turn around $0 - 15kWh$ than $0 - 3kWh$.

The thing that stands out is the performance of the polynomial extrapolation compared to the constant one. The constant extrapolation is better for every hole size. This could perhaps be explained by the fact that the data has very little dependency over time, and there is nearly an equal chance that the next amount of energy consumed will be larger or smaller.

We conclude this chapter by stating that extrapolation algorithms are not efficient for estimating the energy consumption during breaks in transmission of short duration. On the other hand, we notice that the constant extrapolation doesn't exceed an average error of 15% in the Greenbizz and Stockel ECs. This tells us that we can simply assume that the energy consumption stays mostly the same during holes in data and that there is not much chance that the consumption will go wild when no data is available, which is good news.

Longer breaks have not been tested in this project, but it would be surprising to have better performances in the long term since the accuracy of the extrapolation decreases over time. In this case, the better option would be to trust the forecast algorithm since it is his job to estimate the quantity of energy used ahead of time.

Chapter 4

Change analysis algorithm

4.1 Introduction

The goal of this chapter is to detect changes in habits in energy consumption in an online manner and identify the type of change.

As a reminder, we train the prediction algorithm every month, at best, with consumption records from energy suppliers. We can possibly train it every day, but this is not efficient if there is no need to do so. The data we record may also differ from the data sheet sent every month by the energy providers.

The utility of the presented algorithm is thus to know if there is a need to re-train the prediction model before the end of the month if the prediction differs too much from the real consumption.

The detection is designed to run every day, and we see in Section 4.2 why it is no longer relevant to use 15-minute intervals anymore. It must also remain low in computational power. This is the reason we first use a Change Point Detection (CPD) algorithm to detect anomalies. The presentation and choice of the CPD algorithms are described in Section 4.3. If the CPD algorithm has detected a change point (CP), we use a second algorithm that concludes the type of change by analyzing the daily consumption around the CP. It is described in Section 4.4.

4.2 Change of habits and daily consumption data

For the algorithm to give the best performance, we need to feed it interpretable data to detect meaningful anomalies among consumers. To know if there is a need for the forecast algorithm to be retrained, we want to detect changes that last for more than several days and that have a significant shift in the mean. The input also has to be smooth enough to help the CPD algorithm find interesting changes and not noise (like spikes in consumption) in a short amount of time.

If we look at the two graphs of the figure 4.1 which is the data of one month for a member of the Greenbizz EC, the information grouped by day seems to be more usable than the one with 15-minute intervals. It is also more relevant to use daily data to obtain the desired results. Detecting that a particular consumption has been recorded at a different time of day is not our objective. The daily data also has another advantage, which is precision. The forecast algorithm performs better on this scale, according to the developer of the algorithm.

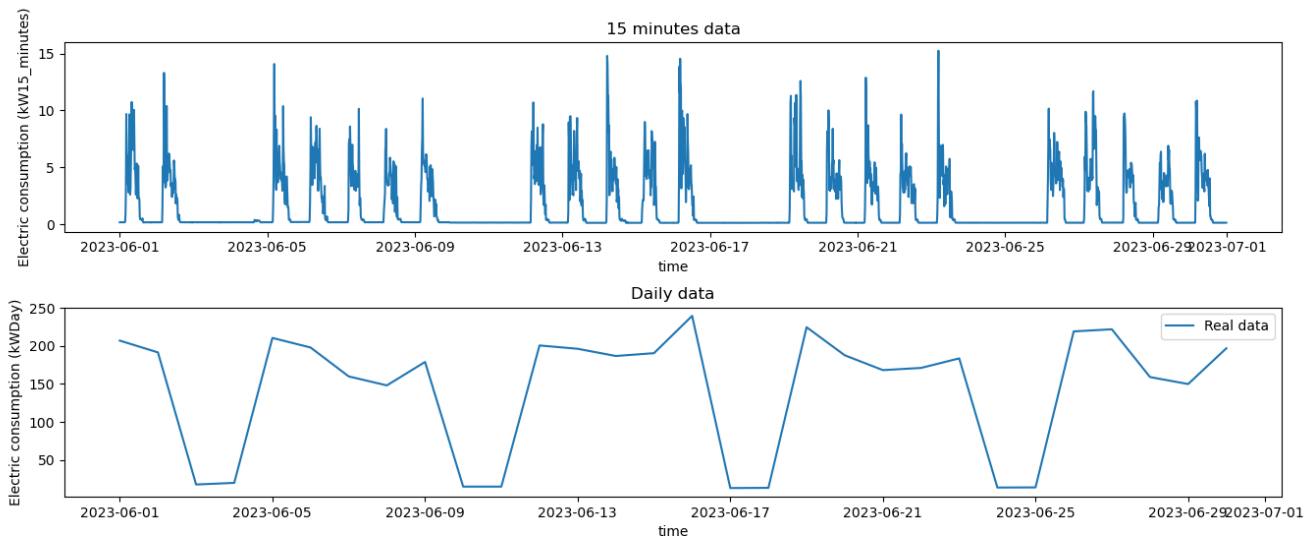


Figure 4.1: Consumption of a random month of a member of the Greenbizz EC

From now on, the rest of the graphs of the chapter shows the energy data in [kW_Day] because the consumption is summed on a daily basis. This looks like a good starting point to try detecting the changes in habits, but we can still do better. As we can see, the consumption is one of an office, which is open only during work hours. If we train the algorithm on the daily data, it

can possibly learn that this consumption pattern is normal, but what happens if there's a bank holiday on a Monday after the weekend or a one-week office closure?

Also, what if the forecast algorithm had predicted that the office would be closed this week? Should we raise the alert in this case? The answer is no, because the algorithm does not need to be retrained, as it has done its job properly. In this case, the problem becomes discovering changes in habits that the forecast could not predict. The method to consider in this case is a simple one. We just take the difference between the forecast and the real consumption as input to the CPD algorithm. The difference between the real consumption in figure 4.1 and the forecast is shown in figure 4.2.

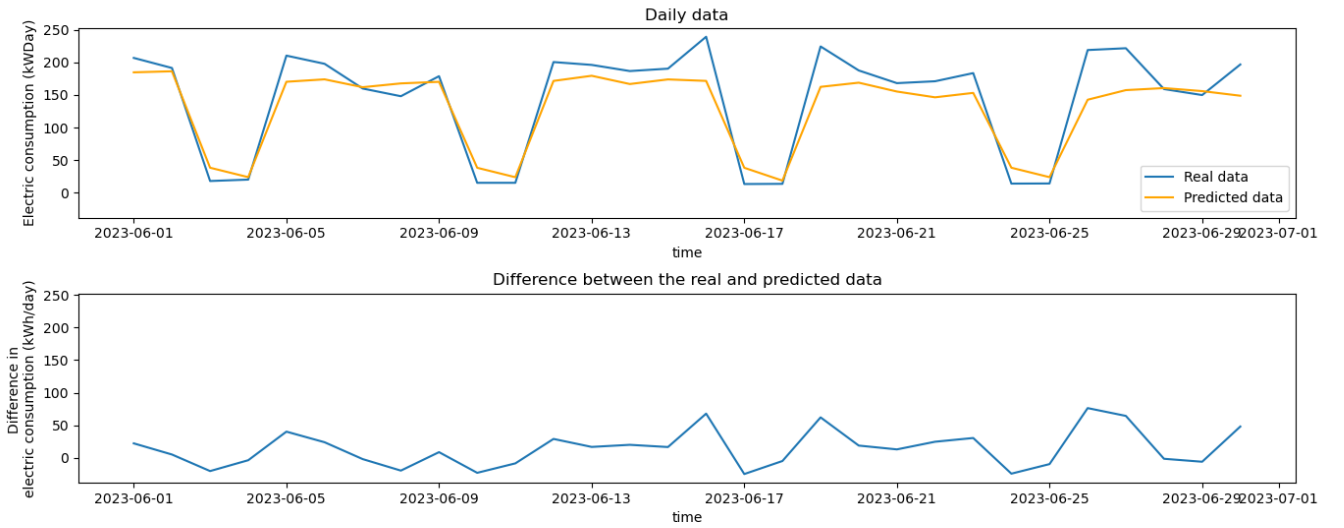


Figure 4.2: Difference of the real and predicted consumption of a member of the Greenbizz EC in [kW_Day]

To visualize where there is a change of habit that the forecast did not see coming, the figure 4.3 shows data from mid-June to the end of July 2023. We notice that the office has a real change of habit after 3 weeks, due to the summer holidays, and the consumption stops almost completely. The forecast did not anticipate that and shows a completely false prediction. With this extra knowledge in mind, we can indeed observe on the second plot that there is a change in the data behavior, and this is exactly what we want to detect in this chapter.

The last advantage of using the forecast in the input of the CPD algorithm is the nullification of seasonality. The prediction is able to deal with the difference in

consumption between the winter and summer, so we don't have to take that into account when searching for the best parameters of the CPD algorithm.

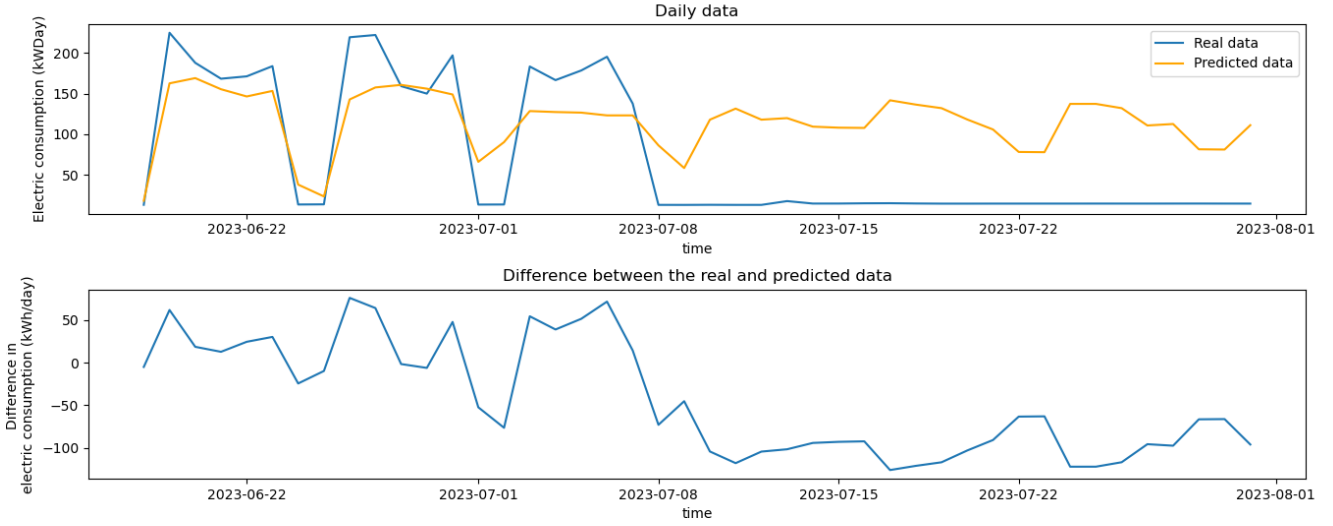


Figure 4.3: Difference of the real and predicted consumption of a member of the Greenbizz EC in [kW_Day]

The only remaining concern about the input data is the case where a change of habit occurs at the beginning of the month. Indeed, we synchronize the CPD algorithm with the forecast, and when the CPD algorithm is still learning what kind of data is incoming, there is no difference in the behavior since the change happened on the first day. We assume in this project that no changes of habit occur in the very first days of the month.

To conclude this discussion about the input of the CPD algorithm, we've found a good way of feeding the information in the form that we want to the algorithm, and it should maximize the chance of detection of true habit changes. Data preparation is not to be underestimated, as it is almost the most time-consuming process in data science.

4.3 Change Point Detection Algorithm

As mentioned in the previous section, the goal here is to find the best algorithm that detects habit changes or prolonged abnormalities in the dataset in near-real time¹. The data is now grouped by day, but we still want it to run every day, and the dataset is thus not finished. To sum up, we want the algorithm to have these characteristics:

- **Online:** The data is updated continuously; the algorithm must work without knowing what comes next but also not have to redo all the calculations once new data arrives.
- **Resource-efficient:** This comes in pair with the online characteristic. The algorithm has to run every day, and we do not want it to use too many resources so that the program can run for several ECs.
- **Adapted for time series:** It must be able to work on a continuous stream of numerical values.
- **Insensitive to extreme cases:** We want to detect habit changes, not outliers. This is the most crucial point of the parameter testing phase of the selected algorithms. In practice, online algorithms cannot completely ignore extreme cases as they are part of the data, but some perform better than others.
- **Unsupervised:** We don't know the shape and exact properties of the time series, as they can vary from case to case, and we want to discover patterns in unlabeled data.

As for the review of the communication protocols in Section 2.4, many options have been set aside because they missed one or more of the characteristics presented above. The choice was based on the survey paper about time series CP detection in 2016 by Samaneh Aminikhanghahi & Diane Joyce Cook [22] and particularly on the section about unsupervised methods resumed in figure 4.4. The most restrictive characteristics are the online requirement and the discovery of CPs in time series.

With all this information in mind, three algorithms were selected. The algorithms are the Bayesian Online Change Point Detection, the Cumulative Sum, and the Change Finder. They are presented in the next sections, and the choice of the

¹The algorithm does not have to detect the change of habit exactly when it begins to appear, because at this moment, there is no difference with an outlier

best option is done in Section 4.3.4.

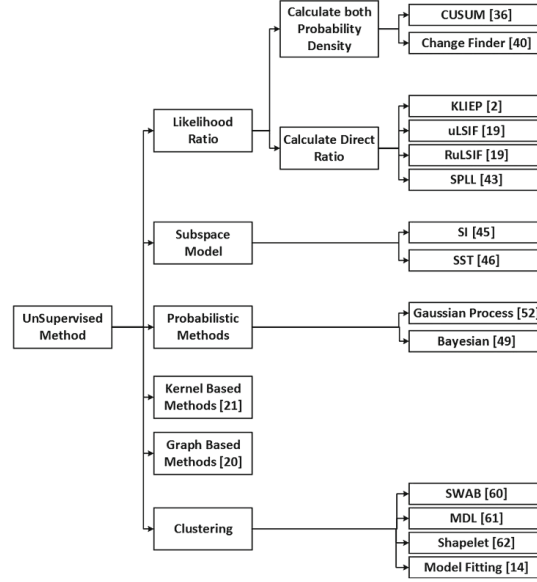


Figure 4.4: Unsupervised CPD algorithms classification [22]

4.3.1 ChangeFinder

The ChangeFinder (CF) algorithm [23] is based on sequentially discounting auto-regression model learning (SDAR), which is an analysis technique to find changes in patterns where the most recent points have more impact than the oldest ones. The problem with the SDAR model is that it is too heavy for online analysis, and CF has been developed for this reason. A remarkable feature of the algorithm is that it uses two-stage learning when new data arrives. The first one is the outlier detection, and the CP detection is using the model learned at the first iteration.

The drawback of this algorithm is that it has a pretty long learning phase, which can be a problem for our application. Indeed, the time series consists of monthly consumptions with one information per day, which gives a maximum length of 31. The learning phase can take up to two weeks with basic tuning parameters, and it could miss some changes during this time or cause the monitoring window to be too small.

The distinctive features of the model can be resumed as :

- **Lightweight:** It makes only one pass through the data and require not much computational power to run.
- **Two stage learning:** When a new data arrives, it is processed twice to first check if it stands out as an outlier, and then be classified as a CP or not.
- **Versatile:** Through good model parameters choosing, it can be a good option for many types of CP detection and time series.
- **Long learning phase:** Depending on the robustness we want to give it and the parameters choosing, the learning phase where the decision value is constantly above the threshold can take much time.

4.3.2 Bayesian online change point detection algorithm

The Bayesian Online Change Point Detection algorithm (BOCPD) is the most mathematically complex algorithm of the ones presented and was introduced in the 2007 paper of Adams & MacKay [24]. It uses statistical inferences to discover the most probable changes in the distribution of time series distributions. As the name suggests, the BOCPD uses the theorem of Bayes to find CPs. As a reminder, Bayes states that, for two events A and B, the posterior distribution $P(A|B)$ (the probability of A knowing that B) has the property:

$$P(A|B) = P(A, B)/P(B) \quad (4.1)$$

where $P(A, B)$ is the probability that A and B occurs at the same time.

In our application, the two events are the run length and the sequence of observations up until the present time. Thus, for every new piece of data, it computes the probability of the presence of a CP at each point in the dataset. The advantage of the online version of the algorithm is the recursive formula to avoid recalculating every distribution at each step. We could dive deeper into the theory, but the goal of this end-of-studies project is not to explain in detail every step of the algorithm, which is very well explained in the article cited above.

To sum up, the particularities of the algorithm, in addition to the ones cited in Section 4.3 are the following :

- **Full reassessment at every time step:** By using recursion, it can discover the location of CPs efficiently. The drawback is that it has a growing run time with the increasing data set. In our case, the lengths of the time series are really short, as explained for the CF algorithm, so it is not a problem.
- **Probabilistic method:** It is the only proposed method that uses probabilities of distribution to find CPs.
- **Adaptability with no parameters:** The algorithm has no input parameters and can adapt to every kind of time series.
- **Computational intensity:** It is the most computationally intensive algorithm of the ones reviewed notably because of the reassessment of every point. It can be mitigated by discarding the points that are too far from the last seen value. As said before, this is not a concern with our type of data.

These features make the BOCPD truly different from other online CPD algorithms by being as unbiased as possible with no parameters and being probability-based. The reassessment at every step is also unique, but the goal is still to find the CPs as soon as possible. We cannot afford to find a CP more than two weeks after it appeared.

4.3.3 Cumulative Sum algorithm

The last algorithm used in this work that meets the requirements of the project is the Cumulative Sum (CUSUM) algorithm. It was first introduced by Page in 1954 [25], and many offline algorithms use his work as a basis to analyze the data.

CUSUM stands out for the simplicity of its implementation. To resume it briefly, it works by summing the new arriving data minus the mean of the already observed data sequentially. If the cumulative sum exceeds a fixed threshold, a CP has been found.

Note that this definition could have some flaws in its behavior. When a CP occurs, it may not stand out enough so that the cumulative sum exceeds the threshold directly. It's reassuring in a way because the model could take some more data to confirm that the CP occurred, but to find the real turning point, we need to use another technique to localize it.

The model can also be sensitive to extreme cases. As opposed to the previous example, a point that differs too much from the range of observed data can raise the cumulative sum too quickly and trigger the threshold. We will dive more in detail into these cases in Section 4.4 about CP identification.

To resume, the algorithm stands off by the following properties :

- **Simplicity:** The model is simple and very lightweight.
- **Learning phase of variable length:** Like any non-parametric CPD algorithm, the model has to learn the behavior of the data before concluding. We can set a minimum time series size before it can detect CPs so that it can learn the initial behavior of the data.
- **Sensitive to outliers:** A single extreme case can trigger the threshold, but this can be mitigated by setting the limit parameter to a higher value.
- **Detect CPs but not their location:** The CPs can be found afterwards but it needs a different algorithm.

4.3.4 Choice of the algorithm

Now that the CPD algorithms have been presented, we have to choose the best option that will detect, with maximum accuracy, the possible change of habits that the forecast did not predict. There are two main possibilities for evaluating the different CPD algorithms:

- **Synthetic datasets:** The idea is to get data that simulates classic consumption without any CPs in it, and then add perturbations at known locations. The advantage is that we know perfectly where the CPs are, but the synthetic dataset might not reflect the real world well enough.
- **Annotated datasets:** We use the real data here to manually find CPs in it, if any. The difficulty lies in the objectiveness of the human annotation. This is what has been done in the 2022 research about the "*Evaluation of Change Point Detection Algorithms*" [26] with multiple annotators that could interpret the data. Obviously, the more annotators there are, the more confident we are in the exact location of the CPs.

Both options have pros and cons, and the two were investigated. The first method has been tried with synthetic consumption datasets from Synergrid². However, the idea was quickly discontinued due to the difficulty of recreating a valid testing dataset that recreates the same conditions as the real data.

For the annotation of the datasets, we didn't have the time to have them annotated by people outside the project to maximize their objectiveness. In this case, a new question also arises about the objective of the CPD algorithm: what type of habit change do we want to discover?

It is indeed difficult to annotate graphs that may have the same behavior as presented in figure 4.5, which is the consumption of a member of the Greenbizz EC. Is there any change of habit in this graph? Maybe yes in the two orange zones where the forecast overestimates and then underestimates the real consumption.

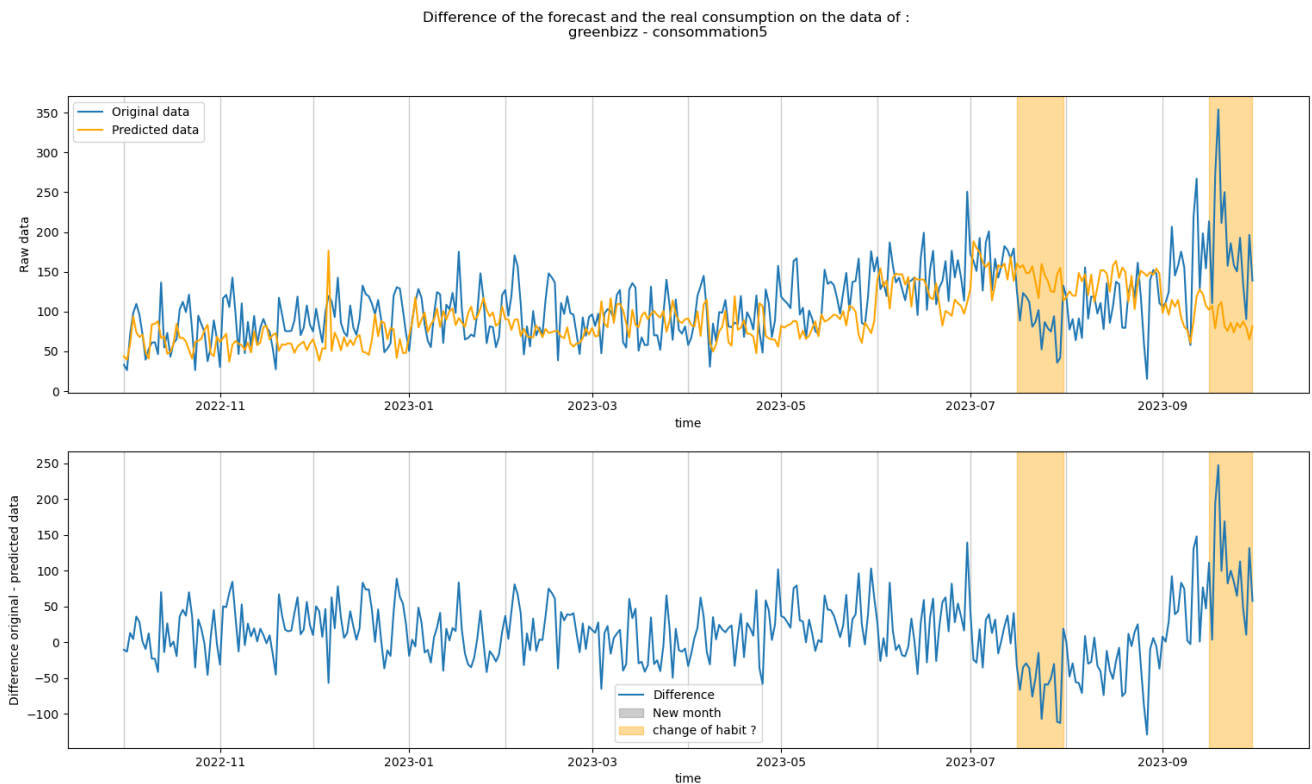


Figure 4.5: Example of time series to analyse in [kW_Day]

²<https://www.synergrid.be/fr/centre-de-documentation/statistiques-et-donnees/profils-slp-spp-rlp>

There are also cases in the data at our disposal where there are more obvious cases of habit changes with which we can work. We could find cases on each dataset of the Greenbizz and Stockel ECs, which are shown in figures 4.6 and 4.7. Indeed, we can easily observe in the data that there is one big habit change that the forecast did not predict in each time series. For the next section, we keep these consumptions to detail our algorithm selection, although the real choice is based on each dataset available.

If needed, the graphs of each dataset are available in the annex A.5. To save on report length, these are the same graphs as the final analysis, but do not take into account the third data plot and the annotations of the graphs for the moment.

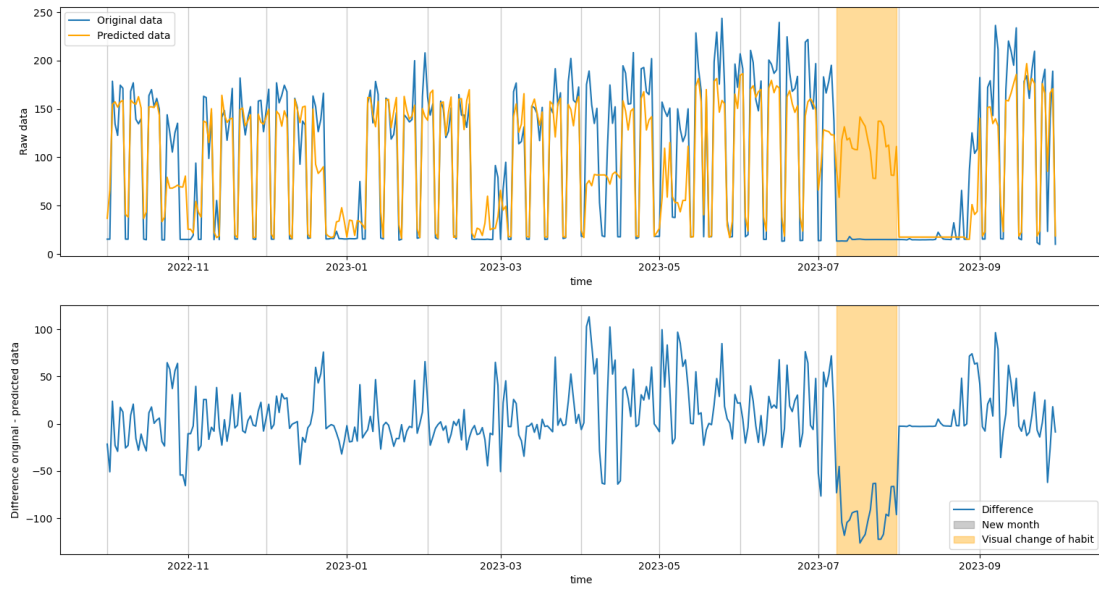


Figure 4.6: Greenbizz time series to base the algorithm selection on [kW_Day]



Figure 4.7: Stockel time series to base the algorithm selection on [kW_Day]

Qualitative performance assessment

Now that the method used to select the best algorithm is clarified, we can begin the testing. To avoid introducing errors in the implementation, the 3 algorithms tested were imported from publicly available Python libraries³. The libraries are bocd⁴ for the BOCPD, changefinder⁵ for the CF algorithm and ocpdet⁶ for the CUSUM implementation.

Note that the BOCPD algorithm still had to be modified to add an update method for every new incoming data. However, this was not a big deal, and the results remained the same before and after the change.

As a reminder, only the CUSUM and CF algorithms have model parameters that need to be tuned, as explained in Section 4.3. The results of the CP detection with the best parameters found of the two CPD algorithms are shown in figures. 4.8 and 4.9 for the CF algorithm. For the CUSUM model, it is shown in figures 4.10 and 4.11.

³Python being the used programming language in this project

⁴<https://github.com/gwgundersen/bocd>

⁵<https://pypi.org/project/changefinder/>

⁶<https://pypi.org/project/ocpdet/>

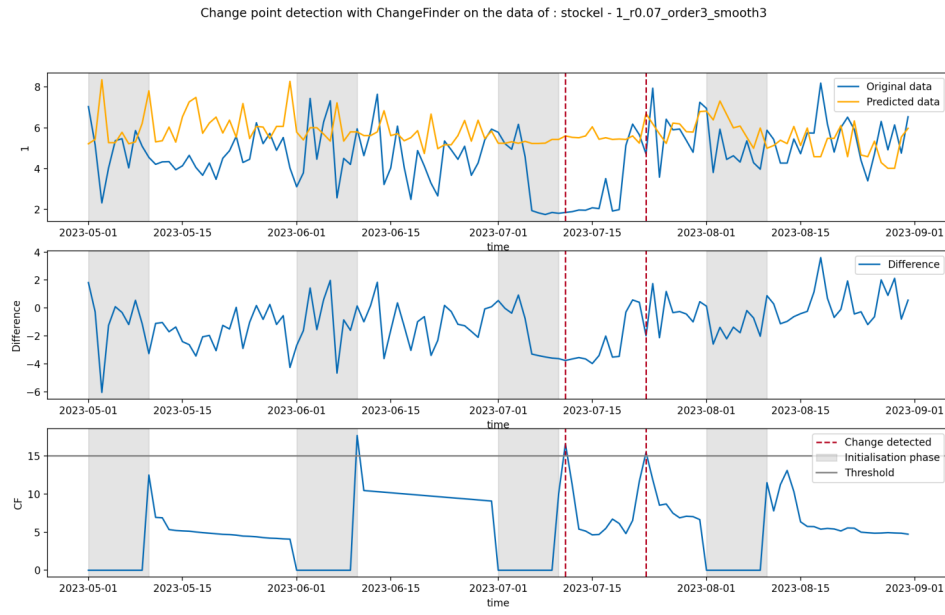


Figure 4.8: performance of the CF algorithm on the selected Stockel dataset [kW_Day]

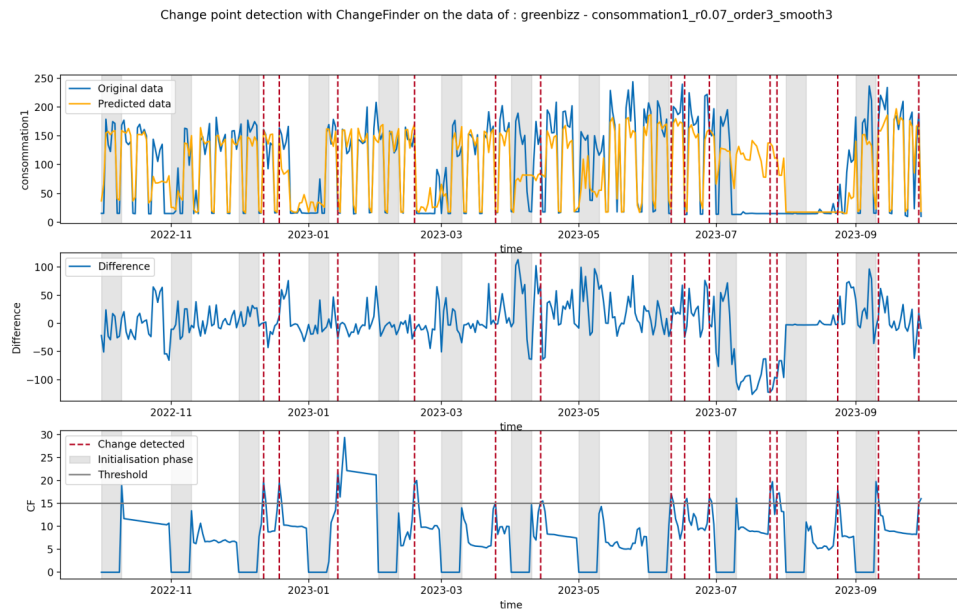


Figure 4.9: performance of the CF algorithm on the selected Greenbizz dataset [kW_Day]

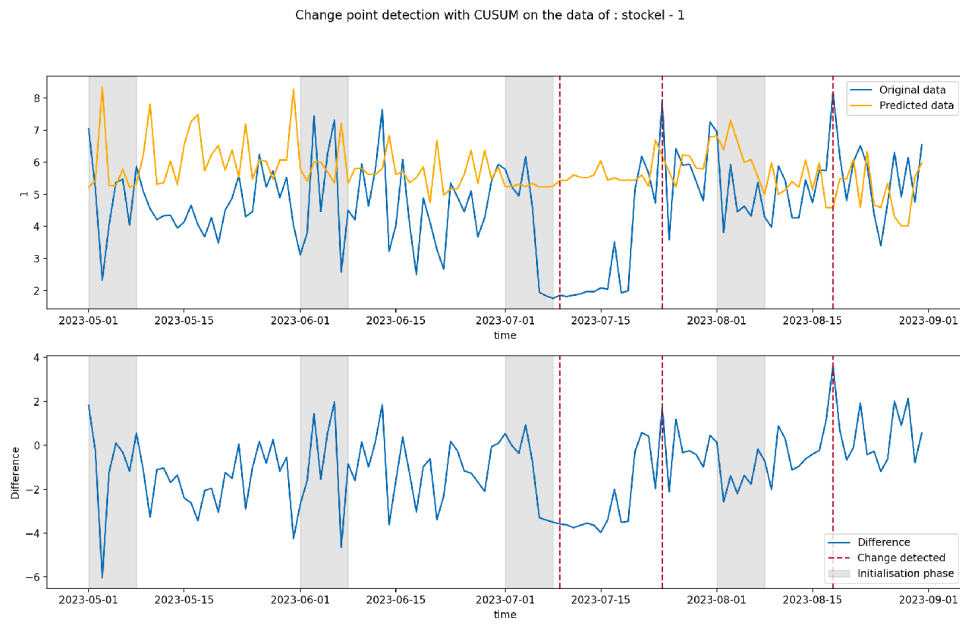


Figure 4.10: performance of the CUSUM algorithm on the selected Stockel dataset [kW_Day]

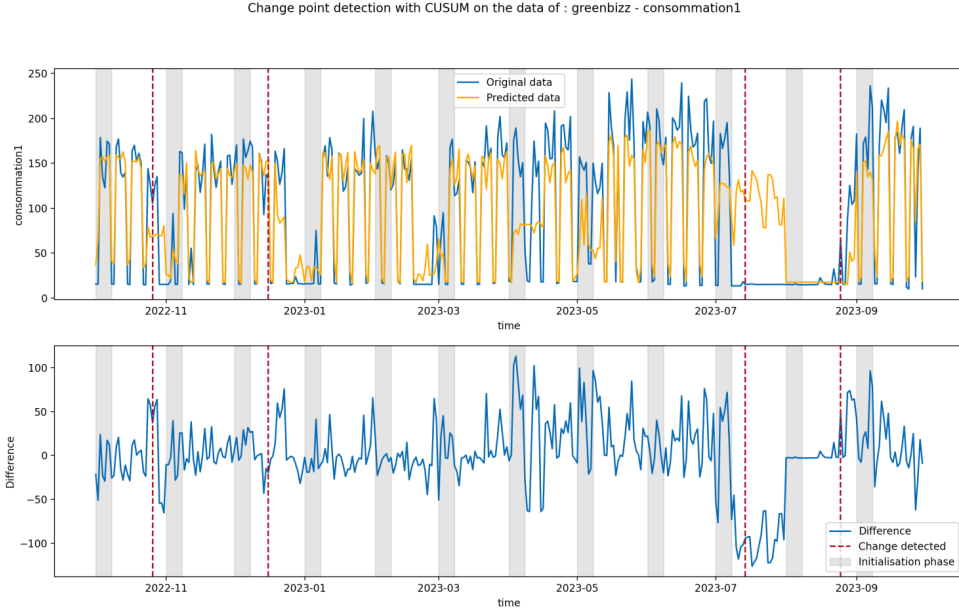


Figure 4.11: performance of the CUSUM algorithm on the selected Stockel dataset [kW_Day]

There is different information available in the graphs of the two algorithms. The most striking one is the initialization phase. It has been set to 7 days as a lower bound to allow one week of learning for the algorithms to gather information on the data. We can observe that the initialization phase of CF with the best parameters takes 9 days and also has a decreasing change score after a large spike in most cases. It can be visualized on the third plot of the CF graphs. We note that the only information that the CUSUM library gives us is the presence of CP in the given data or not. The vertical red dotted lines are the times at which the algorithms find CPs.

The two algorithms have nearly the same behavior for the Stockel dataset, with one CP triggered by an outlier for CUSUM. The notable differences occur for the Greenbizz dataset. The CF algorithm detects many outliers as CPs and finds the habit change really late in the month. This indicates that the CUSUM algorithm might be a better choice out of the two.

Now let's analyze the output of the BOCPD algorithm. The results are shown in figures 4.13. and 4.14. The third plot is a visualization of the probabilities of the run lengths. We note that there aren't any vertical dotted lines that indicate where the changes are spotted because it introduces biases in logical analysis. We therefore need to interpret, in our own way, the analysis of the BOCPD algorithm.

As explained in the paper that introduced the BOCPD algorithm [24], the darker the point, the higher the chance that the run length has lasted since the days equal to the height of the point in the graph. As a reminder, the run length is the time during which the time series might have the same statistical distribution. As we can see in figure 4.12 and to explain it visually, the probabilities of run length are shaped like triangles. If, at a given time, we are at the coordinates of the darker point, which is usually at the top of the triangle, it means that the series follows the same statistical distribution since the extreme left point of the triangle.

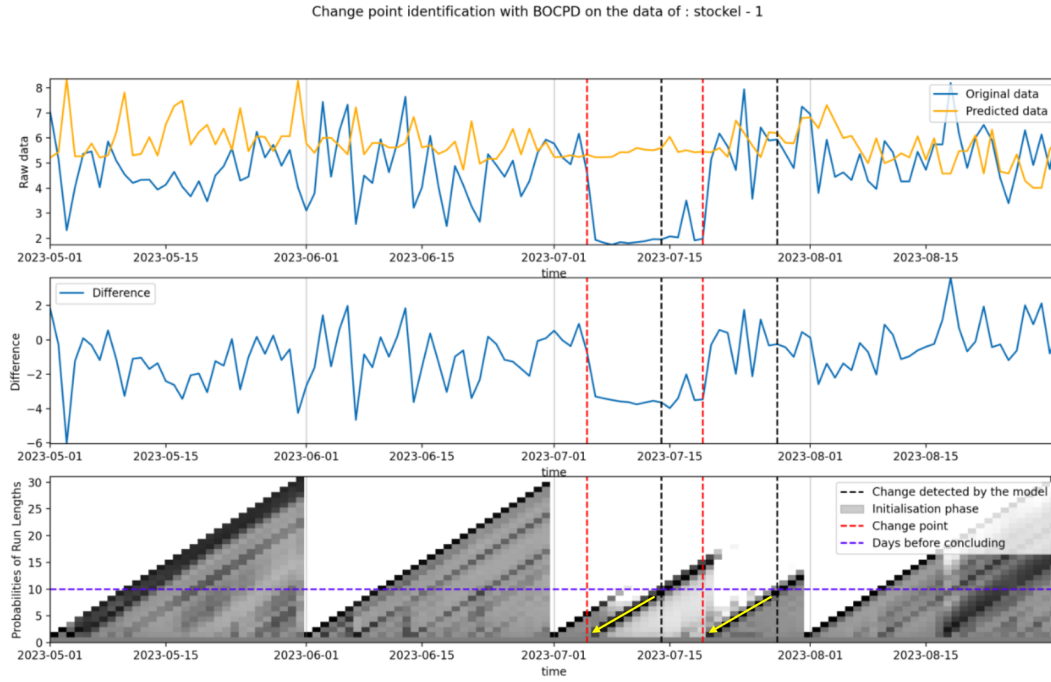


Figure 4.12: Visual example on the CP detection and position finding [kW_Day]

In other words, the algorithm not only gives us a way of detecting CPs but also the time at which they happened (visually represented by the yellow arrows). This is due to the reassessment of every point of the time series.

Regarding the visual analysis of the CPs in the two datasets, we notice that the Stockel CP is clearly discovered by the algorithm, but it also finds another one in the last month. The Greenbizz dataset shows a different behavior with very small run lengths until the CP appears, and the BOCPD clearly finds it by increasing the run length when it happens. It means that it also found the CP in the Greenbizz dataset.

We conclude that the BOCPD algorithm does a really great job of CP detection without being sensitive to extreme cases and with no parameter tuning at all. This removes the bias that we can introduce with this process, except for the interpretation of the run length. The algorithm also has little to no initialization phase, which is really restrictive for the CUSUM and CF options. For all these reasons, this is the algorithm that we will use in the next section to find and identify all the CPs of the time series.

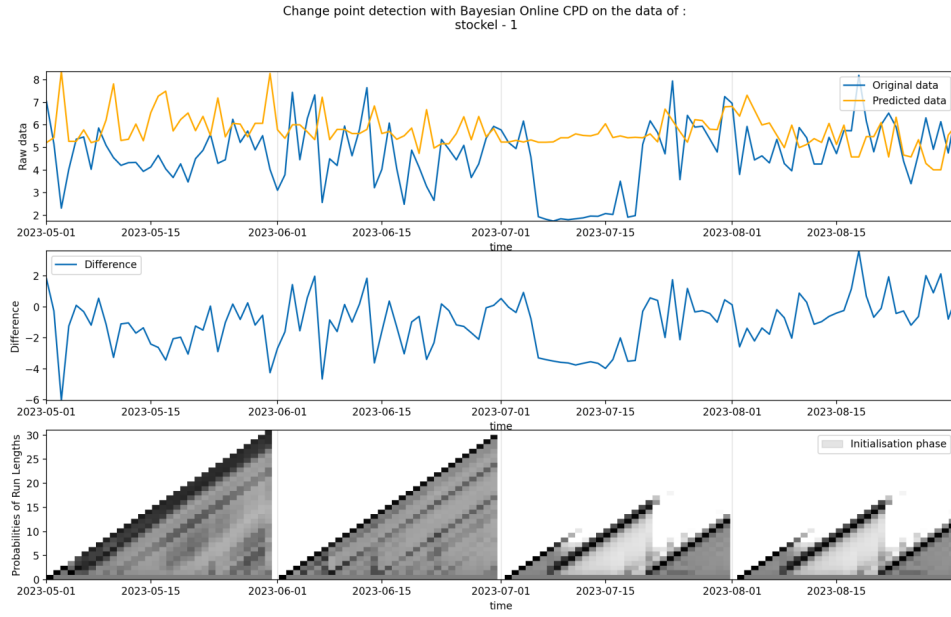


Figure 4.13: performance of the BOCPD algorithm on the selected Stockel dataset [kW_Day]

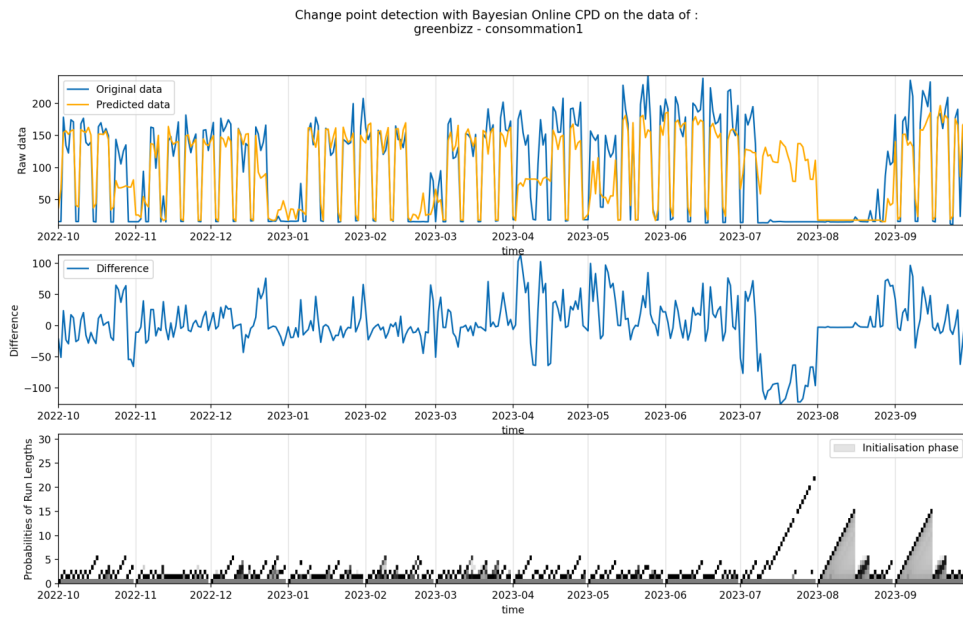


Figure 4.14: performance of the BOCPD algorithm on the selected Greenbizz dataset [kW_Day]

4.4 Changepoint identification

Now that we have chosen the best algorithm to detect changes in habits that the forecast did not predict, we can use it as a base to monitor how the forecast algorithm performs daily and if there is a need to retrain it during the month. The final expected results of the online algorithm are:

- **Inform over the current state of the forecast algorithm:** We want a daily report of the state we could be in, i.e.:
 - **A possible CP:** A CP is a lasting change; thus, by definition, we have to wait a number of days defined by ourselves before we can reach a conclusion. We've set the number of days before concluding to 10 because a change in habits should last at least more than a week. If the run length drops before 10 days, we conclude that it was not a big enough change and move on to the next run. Note that an increasing number of days before concluding gives more confidence in the result of the CP but also has a 10-day delay before concluding the origin of the CP.
 - **A normal state:** This is the case when no run length, except the one at the beginning of the month, has exceeded the threshold. The first run length doesn't count as a CP since we suppose that no change of habit can appear right after the training of the forecast.
- **Find the time when the CP appeared:** With the behavior of the BOCPD algorithm, this is easy because the algorithm counts the number of days since the change of distribution happened.
- **Conclude about the type of CP:** When a CP has been detected by the model, it can be:
 - **A change in habit:** When the mean of the real consumption, before and after the CP, differs more than the forecast mean.
 - **An error of prediction:** The opposite case of the habit change.
 - **Others:** This is the case where the distribution changes, but the difference between the mean of the forecast and real consumption differs too little to conclude about the change type.

The case of a spike in consumption is ruled out by the behavior of the BOCPD algorithm.

- **Find if the CP ends before the end of the month:** If the habits return to normal, the model should inform that everything is clear. This is easy because the model will generate a new CP if the statistical distribution changes again.
- **Send the accuracy of the forecast:** We send the accuracy (same as the one defined in the Section 3.1.2) of the forecast since the beginning of the month every day to inform over the current performances of the forecast.

The implementation, results, and analysis of the final results of the online Change Point Identification (CPI) with these requirements are discussed in the next sections.

4.4.1 Implementation choices of the Change Point Identification

The algorithm is a bit too complex to be reviewed in the main text of the report. Instead, the implementation choices and conditions are discussed here. The pseudo-python code is in the annex A.4 if needed.

The first noticeable aspect is the CP detection of the BOCPD algorithm. As mentioned shortly in the previous sections, the outputs of the BOCPD algorithm are interpretable, in contrast to the CUSUM and CF algorithms. With the BOCPD method, we can define the time to wait to confirm if a true CP has occurred and thus also point exactly where it appeared. If we want to conclude that a CP has appeared after 10 days, we have to check on each column of probabilities to see if the 10th value is the highest; if yes, we have found a CP that occurred 10 days ago. This example can be visualized in figure 4.12 that was shown earlier. The changes are detected 10 days after the real change point, as visualized by the black vertical lines. The horizontal blue line represents the 10-day threshold. The intersection happens at the darker point of the "stair," like the output of the BOCPD, and we go "downstairs" to find at what time the CP began, represented by the yellow arrows.

The second remarkable behavior is the CP categorization. The BOCPD algorithm is meant to detect every change in the statistical behavior of the data. It also includes small, lasting changes that are not worth considering as habit changes. To address this problem, we perform an analysis of the mean of the real and forecast data around the predicted CP. Once we have these averages, we perform simple conditional statements to consider the CP as one of these four categories:

- **Not a real CP:** If the difference of the two averages is smaller than a factor 0.3, or if their difference is smaller than 0.5 kilo Watt day.
- **End of a CP:** Ran after the first case, if the difference of average is smaller after the CP.
- **Error of prediction:** If the average of the forecast differs more than the average of the real data, before and after the CP.
- **Habit change:** If the CP passes no condition stated above.

These conditions were found by doing a visual analysis of the output of all the CP detections in the datasets. It is thus a biased analysis of the problem, but the end result would still be validated in the same way. Also, we want to detect habit changes that show a change in the amount of energy consumed. This can be discovered pretty easily by the human eye without introducing too much bias to the conclusion.

4.4.2 Final results

The CPI process ran on the two bases of consumption introduced at the beginning of the chapter, as shown in figures 4.15 and 4.16. As we directly see, the results are exactly the same as what we described for the figures 4.6 and 4.7, with a label that indicates that these are changes in habits that the forecast did not predict.

As a reminder on how to read the graphs. There is a light vertical plain line indicating the beginning of each month to notify us that the forecast algorithm has been retrained and we reset the time series that we make the online CPD algorithm analyze. All the results and conclusions are simulated like a real daily online algorithm, with one output a day and no coming back, except to notify the real location of the change point.

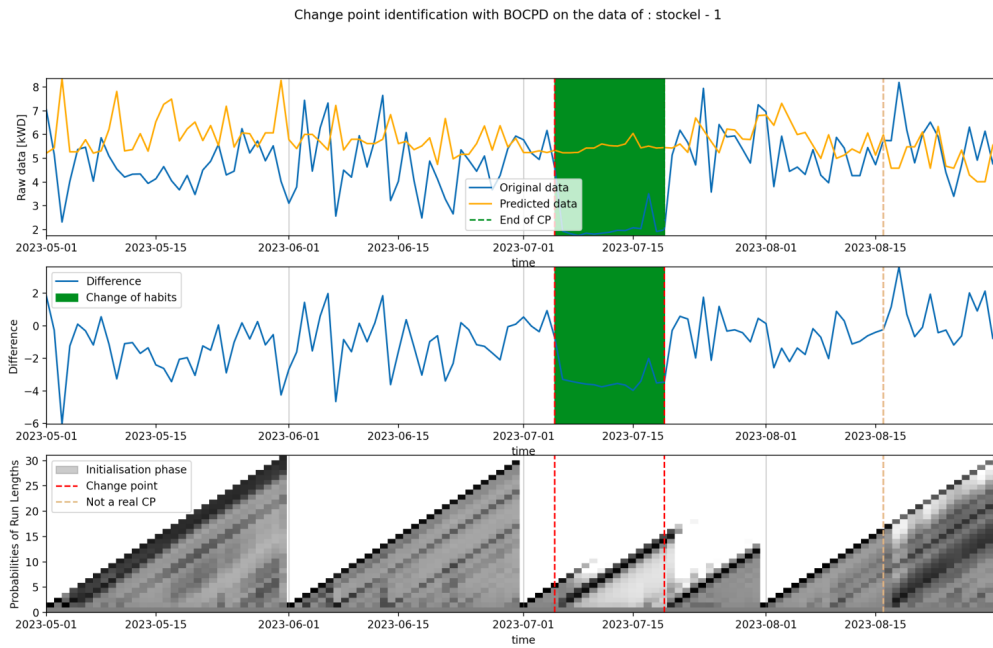


Figure 4.15: Performance of the CPI on the Stockel dataset

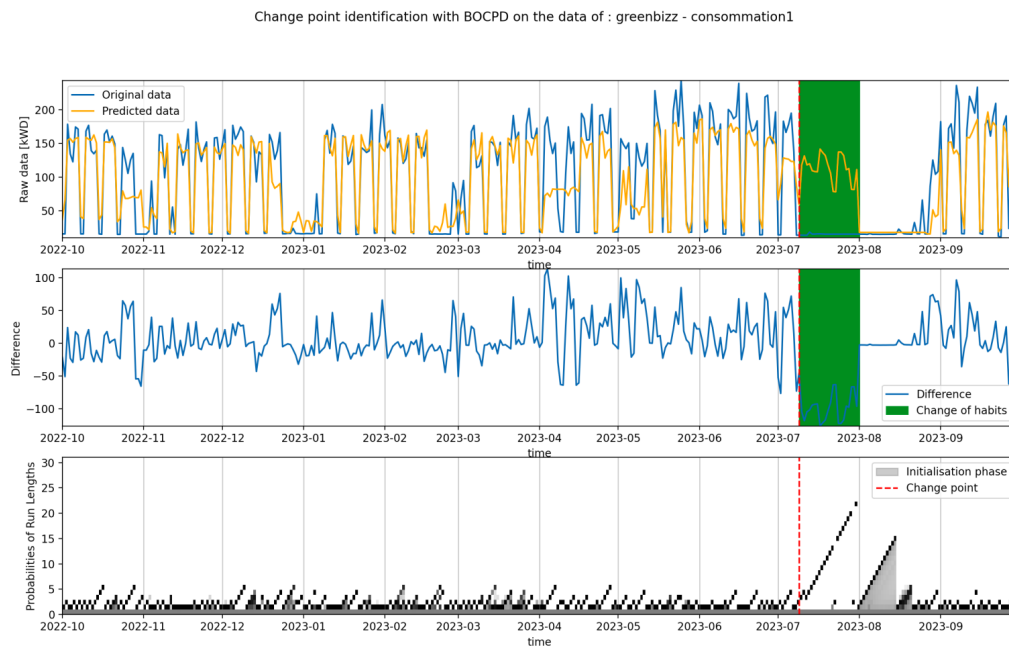


Figure 4.16: Performance of the CPI on the Greenbizz dataset

This is proof that the algorithm succeeded in the desired task. However, we could argue that these are base cases with clear evidence that a habit change has taken place. The algorithm also discovers more complex habit changes, and the results of the 13 different patterns that we could experiment with are shown and fully described in the annex [A.5](#).

Chapter 5

Energy community excess production indicator

5.1 Introduction

This chapter shows one of the possibilities of online data monitoring, which is the measurement of the excess production of ECs. By having the real-time production and sum of all consumption by the members, we know how much electricity is sold to the grid and not consumed by the community. This can be seen as a loss of money, as the resale of electricity in Belgium was on average less than 0.05€/kWh¹ in April 2024, and the consumption cost was on average 0.3€/kWh, also in April².

Another advantage is the reduction of energy consumption during peak hours and the complement of the forecast algorithm. In the case of a perfect prediction, we wouldn't need real-time monitoring. Because of the current political and energy crisis with the war in Ukraine, the European Parliament published an "*Emergency intervention to address high energy prices*" on October 6, 2022 [27]. It introduces measures to reduce energy consumption, in particular during peak hours, and thus mitigate rising energy prices.

The excess production indicator can also be useful for solving this problem. By showing if the electricity production can handle the EC demand, the members can distribute their consumption during these peak hours to improve the community's behavior and maintain self-sufficiency.

¹<https://www.test-achats.be/maison-energie/energie-renouvelable/news/cout-energie-solaire-injectee>

²<https://www.creg.be/fr/professionnels/fonctionnement-et-monitoring-du-marche/evolution-prix-de-lenergie-belgique-et-pays>

5.2 Algorithm

The algorithm is pretty easy to implement and shows good performance. The pseudo code is available in Annex A.6 to have a visual support, but the next paragraphs should tell clearly enough how it works.

Every 15 minutes, the new live data on the latest injection and consumption of the members of the EC arrives. At that moment, the algorithm is launched to see if there is an excess of injection compared to consumption. As a reminder, the injection of a participant is the excess of production that he didn't use for his personal consumption. The consumption of this particular participant should thus not be taken into account in the algorithm when there is an injection from his side.

From now on, we assume that consumption is the sum of the energy use of all the members of the EC, because we are no longer interested in individual behaviors. The same applies for the injection if there is more than one prosumer in the EC. There are two main scenarios:

- **Consumption > injection:** This case is simple, the model indicates that there is no excess now.
- **Consumption < injection:** If it happens, the excess indicator indicates that it is safe to consume now. Then it enters a second phase to see if we can safely predict the duration of this situation.

When trying to predict the duration of the excess injection, the algorithm first compares the forecast to the live data. If the forecast is close enough to the real data, we conclude that it is safe to trust the prediction algorithm at that moment. Then, we analyze for how long the consumption of the forecast is lower than the injection and stop the iteration at the right time.

We limit the forecast to three hours at most because we want to stay within a short-term recommendation with high accuracy. We assume that 3 hours is a fair amount of time for the members of the EC to launch high-energy-demanding processes. Note that even if the algorithm indicates excess for the next few hours, it will still be re-launched as soon as new data arrives, so it can adjust its recommendations.

Four different methods were used to indicate if we can trust the forecast algorithm or not, depending on the difference with the live data and whether the consumption is lower than the injection:

If it happens, the excess indicator indicates that it is safe to consume now. Then it enters a second phase to see if we can safely predict the duration of this situation.

- **Base version:** We consider that it is always safe to consume for the next three hours. This helps us visualise if the forecast algorithm is useful.
- **Always trust the forecast:** In this case, we don't check if the forecast is close to the live data. It allows us to see whether there is any real interest to run the condition.
- **Trust the forecast by a fixed factor:** We check if the forecast of the sum of injection and sum consumption are close enough to a fixed factor separately. If any of the two condition fails, the excess indicator just stays at the base indication that is *"safe to consume now"*.

The figure 5.1 shows the precision results of the two ECs. There are four months of data for the Stockel dataset and six for the Greenbizz one. The precision is simply calculated as the number of correctly predicted *"safe to consume"* divided by the total number of predictions for a given time in the future.

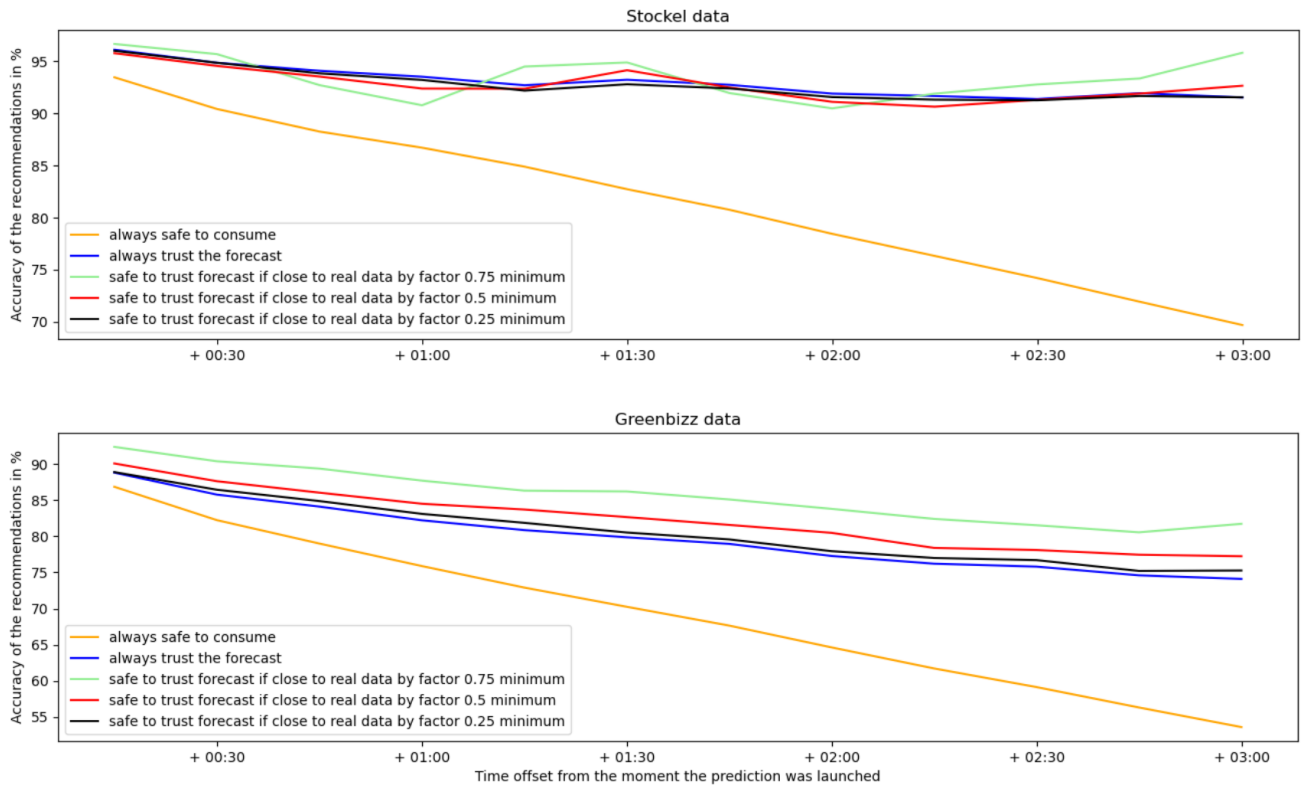


Figure 5.1: Precision of the forecast to predict if there is excess for up to 3 hours, when there is excess at time +00:00

These two graphs show pretty good results in terms of global accuracy, and they highlight two remarkable features.

The "*always safe to consume*" data begins, as expected with a high accuracy, because the consumption is lower than the injection at present time. We note that it decreases linearly in time in the two cases and shows clearly the advantage of using the forecast algorithm for an increasing duration of prediction.

The second, and most interesting, conclusion is that there is no real utility in putting a safety level on the forecast algorithm to trust its prediction. As we can observe in the table 5.1, there is a difference of 6% with the highest factor of trusting at 3 hours on the Greenbizz; the "*always trusting*" condition has made more than 3 times as many predictions as the "*0.75 trusting factor*".

Table 5.1: Additional information at 3 hours prediction for the Greenbizz test

Version	Total number of prediction	Accuracy
0.75 trusting factor	1697	74%
Always trusting	5993	81.7%

We thus come to the conclusion that the best choice for the algorithm is the "*Always trusting*" model for the two EC's behavior that we have at our disposal. The loss of precision due to the blind trust in the forecast is considered insignificant compared to the increase in prediction numbers.

5.2.1 Final result analysis and conclusion

We can visualize the result of the algorithm on a typical day of the Greenbizz energy data in figure 5.2. For visualization purposes, each time we predict an excess with the forecast algorithm, we stop the excess indicator algorithm until the end of the forecast of the excess injection. This helps visualize the time for which the algorithm predicts that it is safe to consume. The real behavior of the algorithm is to re-assess the data every 15 minutes.

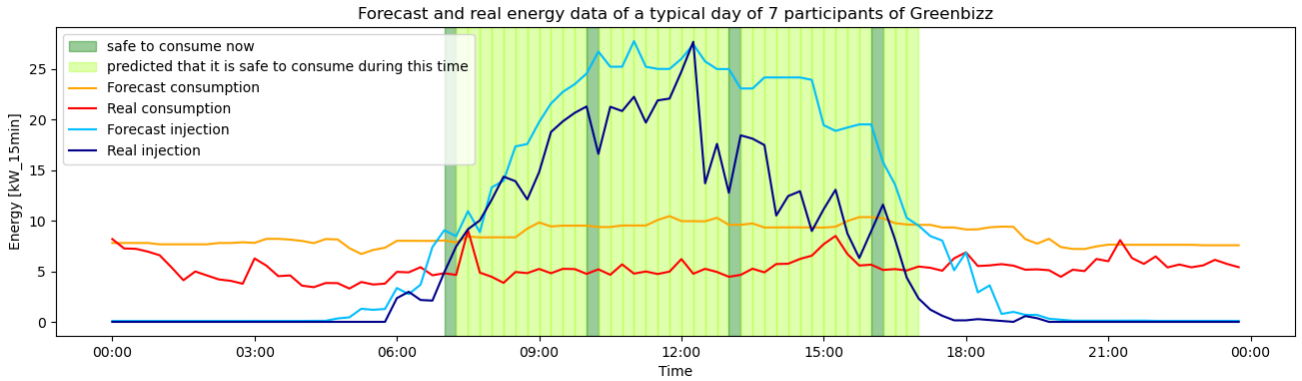


Figure 5.2: Excess injection indicator on the data of the 17th June 2023, Greenbizz

The first thing to note on this graph is the three maximum times of excess injection predicted by the algorithm. In this case, it was right to make this prediction because the injection was higher than the consumption from 7 a.m. to 5 p.m. The model makes one mistake in its prediction at the last predicted excess because the forecast predicted the end of the excess injection roughly 15 minutes late. The indicator algorithm said that it was safe to consume for the next hour, but in reality, it was only for three quarters of an hour.

We notice that the forecast algorithm also does a good job in this example. It is thanks to its good reliability that the excess indicator can have the high accuracy shown in the previous graphs. The predictions overestimate the injection and consumption quantities, but the general trend is well described. Remember that the excess indicator is only interested in the timings where the injection or consumption becomes higher than the other.

As a conclusion of this chapter, we can say that the online monitoring clearly adds real short-term confidence to the forecast algorithm with the verification of the excess injection. This application requires not much calculation or development compared to the change point identification introduced in chapter 4, but it is worth its investment. It gives some leads to improving the energy efficiency of an EC and also reducing the peak consumption, which is a real problem for energy providers.

Chapter 6

Discussion and conclusion

This work covers several different areas, both at the algorithmic level with the various uses of online data monitoring and at the hardware level with communication protocols and transmitters. All these topics have been the subject of rather exploratory research, based on well-known concepts, but not used in the field of energy communities or mentioned in the literature. One example is the change point detection for time series, for which there are in fact very few algorithms suitable for the desired result, considering the given input.

Main findings

Nevertheless, the results of the work show great potential and meet WeSmart's basic request, which is to set up an excess production indicator. We have also been able to explore the detection of changes in habit to know if the forecast algorithm has to be retrained before the end of the month. This also gives confidence in the prediction if no alert is issued.

By proceeding in order of appearance in the chapters, which is also the logical order of the project, we can draw conclusions about each part studied. Starting with the implementation of data recovery, we note that the most suitable communication protocol is most probably MQTT. This was also the choice made by the developers of the two transmitters that were tested in practice. As far as the transmitters are concerned, we didn't have the opportunity to test those of our choice, and the practical test suite contained too few elements for us to draw any real conclusions. The theoretical review allows us to list the possibilities of the market, but without real testing, it is also hard to tell which emitter is the best option.

We did not spend much time on the extrapolation of data in the event of a flow interruption. On the basis of data from the 2 communities available to us, the results show that linear extrapolation is no better than assuming that the members of the EC are consuming the same way as the last data received. This is a rather surprising result that may need more investigation, particularly for gaps of longer duration than those studied and for other types of EC.

In contrast to the extrapolation, a great deal of work has been spent on the change point algorithm, which has led to change point identification. The idea of taking the difference between the prediction and actual consumption as input, coupled with the Bayesian Online Change Point Detection algorithm, works surprisingly well. This is surely due to the fact that the algorithm re-evaluates each past entry at every new data point with decent computational resources. With the precision of the Bayesian algorithm, change point identification is pretty easy, knowing that the time of the potential habit changes has already been discovered by the CPD algorithm.

Finally, the excess production indicator achieves very good results when coupled with the prediction algorithm on the basis of very simple conditions. When predicting an excess of injection for up to 3 hours in the future, it has more than 80% accuracy on the Greenbizz dataset and 90% on the Stockel one. A remarkable discovery is the blind confidence that can be placed in the forecast algorithm to predict how long the injection will remain higher than the consumption, in the case of a higher injection at present.

Limitation of the project and possible improvements

This work is a bit limited in the amount of data that it has been given. Indeed, the two datasets of the energy communities gives different types of consumption that is offices and households, but there are other existing types to explore.

Although the solution is functional, the test suite of the emitters is also very limited and needs further investigation, as stated earlier.

An additional condition may be added for the CP detection. Indeed, it was stated in the hypothesis of the algorithm that no habit change could occur at the first day of the data monitoring. This can happen in the real life and needs to be taken into account.

Finally, there are surely other ideas to explore with the online data monitoring that were not mentioned in this work and that we could not think of.

Bibliography

- [1] J. Roberts, D. Frieden, and S. d’Herbement, “Energy community definitions,” *compile project: Integrating community power in energy islands*, 2019.
- [2] G. P. C. R. M. S. Francesca Ceglia, Elisa Marrasso, “The state of the art of smart energy communities: A systematic review of strengths and limits,” *MDPI*, 2022. [Online]. Available: <https://www.mdpi.com/1996-1073/15/9/3462>
- [3] “promotion of the use of energy from renewable sources,” 2018. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2018.328.01.0082.01.ENG
- [4] “common rules for the internal market for electricity and amending directive,” 2019. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0944>
- [5] Rescoop.eu. [Online]. Available: <https://www.rescoop.eu/>
- [6] “Décret modifiant diverses dispositions en matière d’énergie dans le cadre de la transposition partielle des directives 2019/944/ue du 5 juin 2019 concernant des règles communes pour le marché intérieur de l’électricité et 2018/2001/ue du 11 décembre 2018 relative à la promotion de l’utilisation de l’énergie produite à partir de sources renouvelables et en vue d’adapter les principes relatifs à la méthodologie tarifaire,” 2022. [Online]. Available: <https://wallex.wallonie.be/eli/loi-decret/2022/05/05/2022033591/2022/10/15>
- [7] N. Rossetto, “Beyond individual active customers: citizens and renewable energy communities in the eu,” *IEEE*, 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10158968>
- [8] F. T. Clément ALATON, “Energy communities in the clean energy package: Best practices and recommendations for implementation,” *Publications Office of the European Union*, June 2020. [Online]. Available: <https://op.europa.eu/en/publication-detail/-/publication/4b7d5144-91c9-11eb-b85c-01aa75ed71a1/language-en>

- [9] R. Faia, J. Soares, T. Pinto, F. Lezama, Z. Vale, and J. M. Corchado, “Optimal model for local energy community scheduling considering peer to peer electricity transactions,” *IEEE Access*, vol. 9, pp. 12 420–12 430, 2021.
- [10] N. K. Lucila de Almeida, Viola Cappelli and H. van Soest, “Peer-to-peer trading and energy community in the electricity market - analysing the literature on law and regulation and looking ahead to future challenges,” *SSRN*, April 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3821689
- [11] energycommunitieshub.com. [Online]. Available: <https://energycommunitieshub.com/country/belgium/>
- [12] Smart and meters. [Online]. Available: https://energy.ec.europa.eu/topics/markets-and-consumers/smart-grids-and-meters_en
- [13] “Commission opinion on the implementation plan of belgium,” 2020. [Online]. Available: https://energy.ec.europa.eu/system/files/2020-05/adopted_opinion_be_en_0_1.pdf
- [14] C. Bayılmış, M. A. Ebleme, Ünal Çavuşoğlu, K. Küçük, and A. Sevin, “A survey on communication protocols and performance evaluations for internet of things,” *Digital Communications and Networks*, vol. 8, no. 6, pp. 1094–1104, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352864822000347>
- [15] D. Soni and A. Makwana, “A survey on mqtt: A protocol of internet of things(iot),” 04 2017. [Online]. Available: https://www.researchgate.net/publication/316018571_A_SURVEY_ON_MQTT_A_PROTOCOL_OF_INTERNET_OF_THINGS_IOT
- [16] R. A. Rahman and B. Shah, “Security analysis of iot protocols: A focus in coap,” in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, 2016, pp. 1–7.
- [17] B. H. Çorak, F. Y. Okay, M. Güzel, Murt, and S. Ozdemir, “Comparative analysis of iot communication protocols,” in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1–6.
- [18] R. A. Rahman and B. Shah, “Security analysis of iot protocols: A focus in coap,” in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, 2016, pp. 1–7.

- [19] S. Wei, K. Yu, W. Dai, B. Liang, and X. Zhang, “A study of internet of things real-time data updating based on websocket,” 12 2015, p. 97940T.
- [20] J.-P. Erkkilä, “Websocket security analysis,” 2012. [Online]. Available: <https://juerkkil.iki.fi/files/websocket2012.pdf>
- [21] H. Wang, D. Xiong, P. Wang, and Y. Liu, “A lightweight xmpp publish/subscribe scheme for resource-constrained iot devices,” *IEEE Access*, vol. 5, pp. 16 393–16 405, 2017.
- [22] S. Aminikhanghahi and D. J. Cook, “A survey of methods for time series change point detection,” *Knowledge and Information Systems*, vol. 51, pp. 339 – 367, 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:15595198>
- [23] J. Takeuchi and K. Yamanishi, “A unifying framework for detecting outliers and change points from time series,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 4, pp. 482–492, 2006. [Online]. Available: <https://ieeexplore.ieee.org/document/1599387>
- [24] D. J. M. Ryan Prescott Adams. (2007) Bayesian online changepoint detection. [Online]. Available: <https://arxiv.org/pdf/0710.3742>
- [25] E. S. Page, “Continuous inspection schemes,” *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954. [Online]. Available: <http://www.jstor.org/stable/2333009>
- [26] G. J. J. van den Burg and C. K. I. Williams, “An evaluation of change point detection algorithms,” 2022.
- [27] “Emergency intervention to address high energy prices,” 2022. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32022R1854>

Appendix A

Annexes

A.1 dsmrTelegram.ino base and modified version

Listing A.1: Arduino code example

```
1 void splitTelegram(String rawTelegram){
2     sinceMeterCheck = 0;
3     jsonData = "[";
4     mTimeFound = false;
5     meterError = true;
6     int delimStart = 0;
7     int delimEnd = 0;
8     int eof = rawTelegram.lastIndexOf('\n');
9     //Serial.println(rawTelegram);
10    while(delimEnd < eof){
11        delimEnd = rawTelegram.indexOf('\n', delimStart);
12        String s = rawTelegram.substring(delimStart, delimEnd);
13        delimStart = delimEnd+1;
14        byte starts = s.indexOf('(');
15        String key = s.substring(0, starts);
16        if(key == "0-0:1.0.0"){
17            byte ends = s.indexOf(')');
18            String sub = s.substring(starts+1, ends);
19            int yeari = 2000 + sub.substring(0, 2).toInt();
20            int monthi = sub.substring(2, 4).toInt();
21            int dayi = sub.substring(4, 6).toInt();
22            int houri = sub.substring(6, 8).toInt();
23            int minutei = sub.substring(8, 10).toInt();
24            int secondi = sub.substring(10, 12).toInt();
25            String DST = sub.substring(12);
26            dm_time.tm_sec = secondi;
27            dm_time.tm_hour = houri;
28            dm_time.tm_min = minutei;
29            dm_time.tm_mday = dayi;
30            dm_time.tm_mon = monthi - 1;
31            dm_time.tm_year = yeari - 1900;
32            dm_timestamp = mktime(&dm_time);
33            if(DST == "S"){
34                dm_timestamp = dm_timestamp - (2*3600);
35            }
36            else{
37                dm_timestamp = dm_timestamp - (3600);
38            }
39            mTimeFound = true;
40            if(!wifiSTA) unitState = 2;
```

```

41         else unitState = 4;
42         meterError = false;
43     }
44     else{
45         //Serial.println(key);
46         for(int i = 0; i < sizeof(dsmrKeys)
47             / sizeof(dsmrKeys[0]); i++){
48             if(key == dsmrKeys[i][0]){
49                 if(dsmrKeys[i][1] == "0"){
50                     byte ends = s.indexOf(' ');
51                     String sub = s.substring(starts+1, ends);
52                 }
53                 else if(dsmrKeys[i][1] == "1"){
54                     byte ends = s.indexOf(' ');
55                     String sub = s.substring(starts+1, ends);
56                     int value = sub.toInt();
57                     processMeterValue(i, value, 0, false,
58                         "", dm_timestamp);
59                 }
60                 else if(dsmrKeys[i][1] == "2"){
61                     byte ends = s.indexOf(' ');
62                     String sub = s.substring(starts+1, ends);
63                     float value = sub.toFloat();
64                     processMeterValue(i, 0, value, true,
65                         "", dm_timestamp);
66                 }
67                 else if(dsmrKeys[i][1] == "3"){
68                     byte ends = s.indexOf(' ');
69                     String sub = s.substring(starts+1, ends);
70                     byte units = sub.indexOf('*');
71                     String valuetext = sub.substring(0, units);
72                     int value = valuetext.toInt();
73                     String unit = sub.substring(units+1);
74                     processMeterValue(i, value, 0, false, unit,
75                         dm_timestamp);
76                 }
77                 else if(dsmrKeys[i][1] == "4"){
78                     byte ends = s.indexOf(' ');
79                     String sub = s.substring(starts+1, ends);
80                     byte units = sub.indexOf('*');
81                     String valuetext = sub.substring(0, units);
82                     float value = valuetext.toFloat();
83                     String unit = sub.substring(units+1);

```

```

84         processMeterValue(i, 0, value, true, unit,
85                             dm_timestamp);
86     }
87     else if(dsmrKeys[i][1] == "5"){
88         byte ends = s.indexOf(' ');
89         String sub = s.substring(starts+1, ends);
90         int yeari = 2000 + sub.substring(0, 2).toInt();
91         int monthi = sub.substring(2, 4).toInt();
92         int dayi = sub.substring(4, 6).toInt();
93         int houri = sub.substring(6, 8).toInt();
94         int minutei = sub.substring(8, 10).toInt();
95         int secondi = sub.substring(10, 12).toInt();
96         String DST = sub.substring(12);
97         mb1_time.tm_sec = secondi;
98         mb1_time.tm_hour = houri;
99         mb1_time.tm_min = minutei;
100        mb1_time.tm_mday = dayi;
101        mb1_time.tm_mon = monthi;
102        mb1_time.tm_year = yeari - 1900;
103        mb1_timestamp = mktime(&mb1_time);
104        if(DST == "S"){
105            mb1_timestamp = mb1_timestamp - (2*3600);
106        }
107        else{
108            mb1_timestamp = mb1_timestamp - (3600);
109        }
110        sub = s.substring(ends+2);
111        ends = sub.indexOf(' ');
112        sub = sub.substring(0, ends);
113        byte units = sub.indexOf('*');
114        String valuetext = sub.substring(0, units);
115        float value = valuetext.toFloat();
116        String unit = sub.substring(units+1);
117        if(unit == "m3") unit = "m3";
118        processMeterValue(i, 0, value, true, unit,
119                            mb1_timestamp);
120    }
121 }
122 }
123 }
124 }
125 if(!meterError) sumMeterTotals();
126 }

```

```

127
128 void processMeterValue(int dsmrKey, int imeasurement, float fmeasurement,
129 boolean floatValue, String unit, unsigned long meterTime){
130     //fmeasurement = round2(fmeasurement);
131     if(dsmrKeys[dsmrKey][0] == "1-0:1.8.1") totConDay =
132         fmeasurement;
133     else if(dsmrKeys[dsmrKey][0] == "1-0:1.8.2") totConNight =
134         fmeasurement;
135     else if(dsmrKeys[dsmrKey][0] == "1-0:2.8.1") totInDay =
136         fmeasurement;
137     else if(dsmrKeys[dsmrKey][0] == "1-0:2.8.2") totInNight =
138         fmeasurement;
139     else if(dsmrKeys[dsmrKey][0] == "1-0:1.7.0") totPowCon =
140         fmeasurement;
141     else if(dsmrKeys[dsmrKey][0] == "1-0:2.7.0") totPowIn =
142         fmeasurement;
143     else if(dsmrKeys[dsmrKey][0] == "0-1:24.2.3") totGasCon =
144         fmeasurement;
145     else if(dsmrKeys[dsmrKey][0] == "1-0:32.7.0") volt1 =
146         fmeasurement;
147     else if(dsmrKeys[dsmrKey][0] == "1-0:52.7.0") volt2 =
148         fmeasurement;
149     else if(dsmrKeys[dsmrKey][0] == "1-0:72.7.0") volt3 =
150         fmeasurement;
151     else if(dsmrKeys[dsmrKey][0] == "1-0:1.4.0") avgDem =
152         fmeasurement;
153     else if(dsmrKeys[dsmrKey][0] == "1-0:1.6.0") maxDemM =
154         fmeasurement;
155     String jsonOutput;
156     DynamicJsonDocument doc(1024);
157     doc["sensorId"] = "utility_meter.";
158     + dsmrKeys[dsmrKey][3].substring(dsmrKeys[dsmrKey][3]
159         .lastIndexOf('/')+1);
160     if(floatValue) doc["value"] = round2(fmeasurement);
161     else doc["value"] = imeasurement;
162     if(unit != "") doc["unit"] = unit;
163     doc["timestamp"] = meterTime;
164     if(meterConfig[dsmrKey] == "1"){
165         serializeJson(doc, jsonOutput);
166         jsonData += jsonOutput;
167         jsonData += ",";
168         jsonOutput = "";
169     }

```

```

170     doc["entity"] = "utility_meter";
171     String friendly_name = String(dsmrKeys[dsmrKey][2]);
172     friendly_name.toLowerCase();
173     friendly_name = "Utility meter " + friendly_name;
174     doc["friendly_name"] = friendly_name;
175     if(dsmrKeys[dsmrKey][4] != "") doc["metric"] = dsmrKeys[dsmrKey][4];
176     doc["metricKind"] = dsmrKeys[dsmrKey][5];
177     serializeJson(doc, jsonOutput);
178     if(meterConfig[dsmrKey] == "1"){
179         if(mqtt_en) {
180             if(sinceLastUpload >= (upload_throttle * 1000)){
181                 pubMqtt(dsmrKeys[dsmrKey][3], jsonOutput, false);
182             }
183         }
184     }
185 }
186
187 void sumMeterTotals(){
188     totCon = (totConDay + totConNight)*1.0;
189     totConToday = totCon - totConYesterday;
190     totIn = (totInDay + totInNight)*1.0;
191     netPowCon = (totPowCon - totPowIn)*1.0;
192     gasConToday = totGasCon - gasConYesterday;
193     if(dm_time.tm_mday != prevDay){
194         totConYesterday = totCon;
195         gasConYesterday = totGasCon;
196         prevDay = dm_time.tm_mday ;
197     }
198     if(mqtt_en){
199         for(int i = 0; i < 3; i++){
200             String totalsTopic = "";
201             String jsonOutput;
202             DynamicJsonDocument doc(1024);
203             if(i == 0){
204                 totalsTopic = "total_energy_consumed";
205                 doc["sensorId"] = "utility_meter." + totalsTopic;
206                 doc["value"] = round2(totCon);
207                 doc["unit"] = "kWh";
208                 doc["timestamp"] = dm_timestamp;
209                 serializeJson(doc, jsonOutput);
210                 jsonData += jsonOutput;
211                 jsonData += ",";
212                 jsonOutput = "";

```

```

213         doc["entity"] = "utility_meter";
214         doc["friendly_name"] =
215             "Utility meter total energy consumed";
216         doc["metric"] = "GridElectricityImport";
217         doc["metricKind"] = "cumulative";
218     }
219     else if(i == 1){
220         totalsTopic = "total_energy_injected";
221         doc["sensorId"] = "utility_meter." + totalsTopic;
222         doc["value"] = round2(totIn);
223         doc["unit"] = "kWh";
224         doc["timestamp"] = dm_timestamp;
225         serializeJson(doc, jsonOutput);
226         jsonData += jsonOutput;
227         jsonData += ",";
228         jsonOutput = "";
229         doc["entity"] = "utility_meter";
230         doc["friendly_name"] =
231             "Utility meter total energy injected";
232         doc["metric"] = "GridElectricityExport";
233         doc["metricKind"] = "cumulative";
234     }
235     else{
236         totalsTopic = "total_active_power";
237         doc["sensorId"] = "utility_meter." + totalsTopic;
238         doc["value"] = round2(netPowCon);
239         doc["unit"] = "kW";
240         doc["timestamp"] = dm_timestamp;
241         serializeJson(doc, jsonOutput);
242         jsonData += jsonOutput;
243         jsonData += "];";
244         jsonOutput = "";
245         doc["entity"] = "utility_meter";
246         doc["friendly_name"] =
247             "Utility meter total active power";
248         doc["metric"] = "GridElectricityPower";
249         doc["metricKind"] = "gauge";
250     }
251     totalsTopic = "data/devices/utility_meter/" + totalsTopic;
252     serializeJson(doc, jsonOutput);
253     if(sinceLastUpload >= (upload_throttle*1000)){
254         pubMqtt(totalsTopic, jsonOutput, false);
255     }

```

```

256     }
257     if (sinceLastUpload >= (upload_throttle*1000)){
258         sinceLastUpload = 0;
259     }
260 }
261 }

```

Listing A.2: modified part of the code

```

166
167 void sumMeterTotals(){
168     totCon = (totConDay + totConNight)*1.0;
169     totConToday = totCon - totConYesterday;
170     totIn = (totInDay + totInNight)*1.0;
171     netPowCon = (totPowCon - totPowIn)*1.0;
172     gasConToday = totGasCon - gasConYesterday;
173     if (dm_time.tm_mday != prevDay){
174         totConYesterday = totCon;
175         gasConYesterday = totGasCon;
176         prevDay = dm_time.tm_mday ;
177     }
178     if (mqtt_en){
179         String totalsTopic = "measures_modified";
180         // MODIFIED : the new topic to which we send the data
181         totalsTopic = "data/devices/utility_meter/" + totalsTopic;
182         String jsonOutput;
183         DynamicJsonDocument doc(1024);
184
185         // ----- values to send -----
186         doc["totConDay"] = round3(totConDay);
187         doc["totConNight"] = round3(totConNight);
188         doc["totInDay"] = round3(totInDay);
189         doc["totInNight"] = round3(totInNight);
190         doc["timestamp"] = dm_timestamp;
191         serializeJson(doc, jsonOutput);
192         jsonData += jsonOutput;
193         jsonData += " ]";
194         jsonOutput = "";
195         // ----- values to send -----
196
197         serializeJson(doc, jsonOutput);
198         if (sinceLastUpload >= (upload_throttle*1000)){
199             pubMqtt(totalsTopic, jsonOutput, false);
200         }

```

```
201         if (sinceLastUpload >= (upload_throttle*1000)){
202             sinceLastUpload = 0;
203         }
204     }
205 }
```

A.2 Bash script of the data retrieving

```
1
2 mosquitto_sub -h localhost -t data/devices/utility_meter/
   measures_modified |
3 while IFS= read -r line
4 do
5     timestamp=$(echo $line | jq -r '.timestamp')
6     totConDay=$(echo $line | jq -r '.totConDay')
7     totConNight=$(echo $line | jq -r '.totConNight')
8     totInDay=$(echo $line | jq -r '.totInDay')
9     totInNight=$(echo $line | jq -r '.totInNight')
10
11     date_readable=$(TZ='Europe/Brussels' date -d "@$timestamp" +"%Y
   -%m-%d")
12     date_precise=$(TZ='Europe/Brussels' date -d "@$timestamp" +"%Y-%
   m-%d %H:%M:%S")
13
14     filename="$date_readable.csv"
15
16     if [ ! -f "tmp_files/$filename" ]; then # if there is no file
   named filename in tmp_files
17         if [ -n "$old_filename" ]; then # if variable "
   old_filename" is already assigned
18             cp "tmp_files/$old_filename" "backup_data_p1"
19             cp "tmp_files/$old_filename" "data_p1"
20             rm "tmp_files/$old_filename"
21         fi
22         old_filename="$date_readable.csv"
23         touch "tmp_files/$filename"
24     fi
25
26     echo "$date_precise,$totConDay,$totConNight,$totInDay,
   $totInNight" >> "tmp_files/$filename"
27 done
```

A.3 Python script of the data retrieving

```
1
2 import subprocess
3 import time
4 import os
5 import json
6 import csv
7 from datetime import datetime, timezone, timedelta
8
9 # script adapted to keep only the data every 15 minutes with an
10 # emitter sending data every minute
11
12 # global variables
13 last_15_minutes = None
14 delta_injection = None
15 delta_consumption = None
16 last_values = []
17 values_subtract_fifteen = []
18
19 def run_script():
20     current_datetime = datetime.now()
21     global last_15_minutes
22     global last_values
23     last_15_minutes = datetime.now()
24     - timedelta(minutes=current_datetime.minute % 15,
25                 seconds=current_datetime.second,
26                 microseconds=current_datetime.microsecond)
27     last_15_minutes = last_15_minutes.strftime("%d/%m/%Y %H:%M:%S")
28
29     bash_command =
30     ("mosquitto_sub -L mqtt://iot.sween.fr:993/wesmart/app/v1.0/# "
31      "--insecure -u wesmart -i wesmart "
32      "--cafile /home/marty/unif/2023-2024/memoire/"
33      "algorithm_project/sween/sween-fr-chain.crt "
34      "-P 6L7BpkFc52Vs4Th9 -q 1")
35     # Create a subprocess.Popen object
36     process = subprocess.Popen(
37         bash_command,
38         shell=True,
39         stdout=subprocess.PIPE,
40         text=True,
41     )
```

```

42
43 while True:
44     # Read the next line of output (non-blocking)
45     output = process.stdout.readline().strip()
46
47     if output != "":
48         process_output(output)
49     else:
50         time.sleep(0.5)
51
52     if len(last_values) > 1:
53         processed_value = last_values[-1]
54         if processed_value is not None:
55             # add to the document
56             for value in processed_value:
57                 add_value_doc(value)
58
59 """
60 takes an output of the command and process it as a
61 list of values and add it to the last_processed values
62 """
63 def process_output(measure):
64     global last_values
65     data = json.loads(measure)
66
67     # Access and store values in variables
68     consumption_timestamp = data['consumption']['timestamp']
69     cumulative_injected_power =
70         data['consumption']['cumulative_injected_power']
71     cumulative_withdrawn_power_index0 =
72         data['consumption']['cumulative_withdrawn_power_index0']
73     cumulative_withdrawn_power_index1 =
74         data['consumption']['cumulative_withdrawn_power_index1']
75
76     utc_datetime = datetime.fromtimestamp(consumption_timestamp)
77         .replace(tzinfo=datetime.timezone.utc)
78     brussels_timezone_offset = timedelta(hours=1)
79     brussels_datetime = utc_datetime
80         .astimezone(datetime.timezone(brussels_timezone_offset))
81     formatted_brussels_datetime = brussels_datetime
82         .strftime("%d/%m/%Y %H:%M:%S")
83
84     informations = (formatted_brussels_datetime,

```

```

85         cumulative_withdrawn_power_index0 +
86         cumulative_withdrawn_power_index1 ,
87         cumulative_injected_power)
88
89     # if we are in a case of repetition of the same information
90     if abs(utc_datetime - datetime.now(timezone.utc))
91         > timedelta(seconds=30):
92         formatted_brussels_datetime = datetime.now()
93             .strftime("%d/%m/%Y %H:%M:%S")
94         informations = (formatted_brussels_datetime ,
95                         cumulative_withdrawn_power_index0
96                         + cumulative_withdrawn_power_index1 ,
97                         cumulative_injected_power)
98
99     if len(last_values) < 5: # keeps 15" of the lasts values
100         last_values.append(informations)
101     else:
102         last_values.append(informations)
103         last_values = last_values[1:]
104
105
106 def add_value_doc(data):
107     global delta_injection
108     global delta_consumption
109     global last_15_minutes
110     global values_subtract_fifteen # injection consommation
111     # data of type : string date, withdrawn power,
112     # injected power, extrapolation or not
113     if not os.path.exists("sween_data.csv"):
114         with open("sween_data.csv", mode='w', newline='') as file:
115             writer = csv.writer(file)
116             writer.writerow(["Column1", "INJECTION-TOTALE",
117                             "CONSOMMATION-TOTALE", "interpolation"])
118             writer.writerow([data[0], "None", "None", "None"])
119             values_subtract_fifteen = [data[1], data[2]]
120     else:
121         if not data:
122             with open("sween_data.csv", mode='a', newline='') as file:
123                 writer = csv.writer(file)
124                 writer.writerow([last_15_minutes,
125                                 delta_injection,
126                                 delta_consumption,
127                                 1])

```

```
128     else :
129         delta_injection = data[2] - values_subtract_fifteen[1]
130         delta_consumption = data[1] - values_subtract_fifteen[0]
131         with open("sween_data.csv", mode='a', newline='') as file :
132             writer = csv.writer(file)
133             writer.writerow([data[0],
134                             delta_injection,
135                             delta_consumption,
136                             data[3]])
137         values_subtract_fifteen = [data[1], data[2]]
138
139 run_script()
```

A.4 Pseudo python code of the CP identification

```
1
2
3 """
4 Final algorithm to simulate the income of data online ,
5 do the change point detection and analyse the type of the change point
6 """
7
8 from bayesian_CP_detection_basic import GaussianUnknownMean, OnlineBocd
9 import numpy as np
10 import pandas as pd
11
12 # ----- PARAMETERS -----
13 # number of days to wait and analyse if the change is still there
14 # to conclude what is the CP type
15 days_to_wait_before_conclusion = 10
16 # number of days to take before the change point
17 # to measure the precedent mean
18 days_mean_before_change_point = 7
19
20 # -----
21
22 def check_changepoint(data):
23     return model.update(data)
24
25
26 def analyse_point(just_received_data, count_day_of_month,
27                  days_before_concluding=10):
28     column = check_changepoint(just_received_data
29                               - predicted_data[count_day])
30     index_max = np.argmax(column)
31
32     # check if we are in the first run length
33     if index_max > count_day_of_month - 2:
34         return "No CP detected", column
35
36     if days_before_concluding + 4 > index_max
37       >= days_before_concluding - 1:
38         return "CP detected", column
39     else:
40         return "No CP detected", column
41
```

```

42
43 count_day = -1
44 CP = False # if we are in a change point
45 to measure the accuracy of the prediction
46 biggest_value_consumption_seen = -1 #
47 already_found_CPs = []
48 predicted_data = pd.read_csv("predicted_data.csv")
49
50 difference_before_first_difference = None
51
52 gaussian = GaussianUnknownMean(0, 2, 1) # random values
53 model = OnlineBodc(gaussian, 1/100, 31) # random values
54
55 # We simulate with the for loop that we just received a new data
56 for just_received_data, day in zip(original_data[data_column],
57                                   original_data['time']):
58
59     count_day += 1
60
61     data_memory.append(just_received_data)
62     data_difference = just_received_data
63                     - predicted_data[data_column][count_day])
64
65     report, column = analyse_point(just_received_data, count_day,
66                                   days_to_wait_before_conclusion)
67     supposed_real_cp_best = count_day - np.argmax(column)
68
69     if supposed_real_cp_best == already_found_CPs[-1]:
70         pass
71     elif report == "CP detected" and not CP:
72         CP = True
73         highest_argmax = np.argmax(column)
74         # find what mean has the most changed between
75         # the real consumption and the forecast since the CP
76         # check if the days_mean_before_change_point does not
77         # point to earlier of the beginning of the month
78         index_data_before_CP = count_day - days_mean_before_change_point
79                             - supposed_real_cp_best
80
81         if count_day_of_month - days_mean_before_change_point
82           - supposed_real_cp_best < 0:
83             index_data_before_CP = count_day - count_day_of_month
84             index_CP = supposed_real_cp_best

```

```

85     index_after_CP = count_day
86
87     index_real_before_CP = days_mean_before_change_point
88     if index_real_before_CP > count_day_of_month:
89         index_real_before_CP = -count_day_of_month
90
91     mean_pred_before = np.mean(predicted_data[data_column]
92                                [index_data_before_CP:index_CP])
93     mean_pred_after = np.mean(predicted_data[data_column]
94                                [index_CP+1:index_after_CP])
95     mean_real_before = np.mean(data_memory
96                                [-(index_real_before_CP+np.argmax(column)):
97                                 -np.argmax(column)])
98     mean_real_after = np.mean(data_memory[-np.argmax(column)+1:])
99
100    diff_mean_pred = abs(mean_pred_after - mean_pred_before)
101    diff_mean_real = abs(mean_real_after - mean_real_before)
102    type_change = ""
103
104    if 0.7 < diff_mean_real/diff_mean_pred < 1.3
105        or (diff_mean_real - diff_mean_pred < 0.5):
106        type_change = "not a real changepoint"
107    elif abs(mean_pred_before - mean_real_before)
108        > abs(mean_pred_after - mean_real_after):
109        type_change = "return to normal"
110    elif diff_mean_pred > diff_mean_real:
111        type_change = "error prediction"
112    else:
113        type_change = "change of habits"
114    report = "CP detected at date " + str(supposed_real_cp_best)
115           + "and of type : " + type_change
116    already_found_CPs.append(supposed_real_cp_best)
117
118    elif report == "CP detected" and CP:
119        report = "End of CP"
120        CP = False
121        already_found_CPs.append(supposed_real_cp_best)
122
123
124    # accuracy of the prediction
125    if just_received_data > biggest_value_consumption_seen:
126        biggest_value_consumption_seen = just_received_data
127    if biggest_value_consumption_seen == 0 and data_difference != 0:

```

```
128         prediction_accuracy = 0
129     else:
130         prediction_accuracy = abs(data_difference) * 100
131         / biggest_value_consumption_seen
132
133     print((day, report, prediction_accuracy))
```

A.5 Graphical analyse of the CP identification

A.5.1 Stockel dataset 0

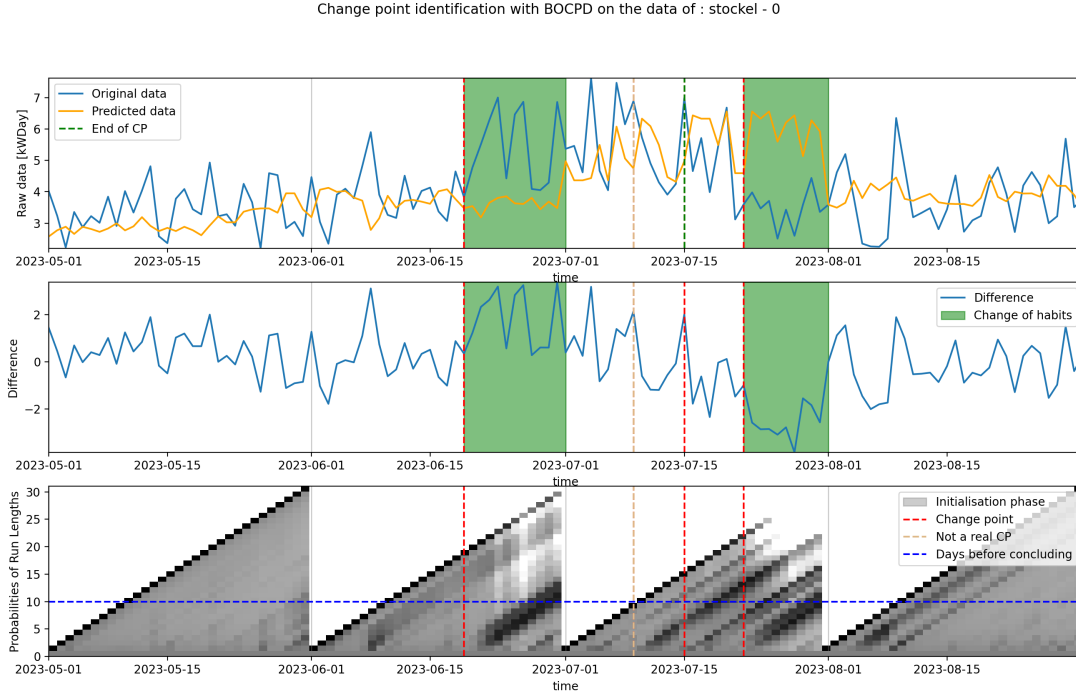


Figure A.1: Change point identification Stockel dataset 0

The graphs of this figure shows a consumption pattern that is not regular and makes it difficult for the forecast algorithm to predict, but also for the CPI algorithm to analyse. The June month and the July months indeed shows many different probabilistic runs as shown in the two visual "stairs" in the third plot.

The BOCPD algorithm still finds some CPs that lasts over 10 days. This consumption pattern clearly shows the need to do an identification of the CPs, as there is a CP that is classified as insignificant through the burlywood vertical dotted line. This day was defined as a CP by the BOCPD, but the CPI algorithm found that the change of means of the prediction and real consumption, before and after the CP are too little to be classified as a change in habit of error of prediction.

there are also two CPs that were classified as habit changes, and we can observe that the consumption grows larger than the prediction in the first green zone, and smaller in the second zone. We can also note that change of habit period is stopped by the ed of the month and

resetting of the forecast algorithm, as well as de BOCPD.

A.5.2 Stockel dataset 1

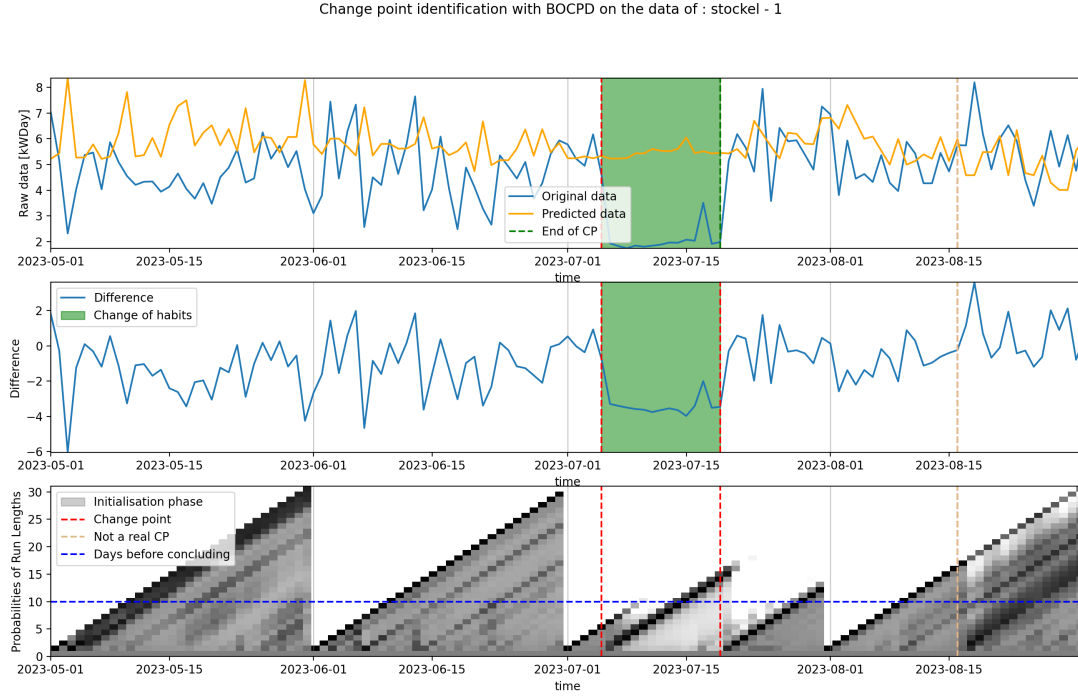


Figure A.2: Change point identification Stockel dataset 1

The second graph is the one taken as example to explain the methodology of the CPI algorithm of the chapter 4. The clear habit change is correctly classified, and there is also a CP interpreted as a insignificant change, which seems logic.

A.5.3 Stockel dataset 2

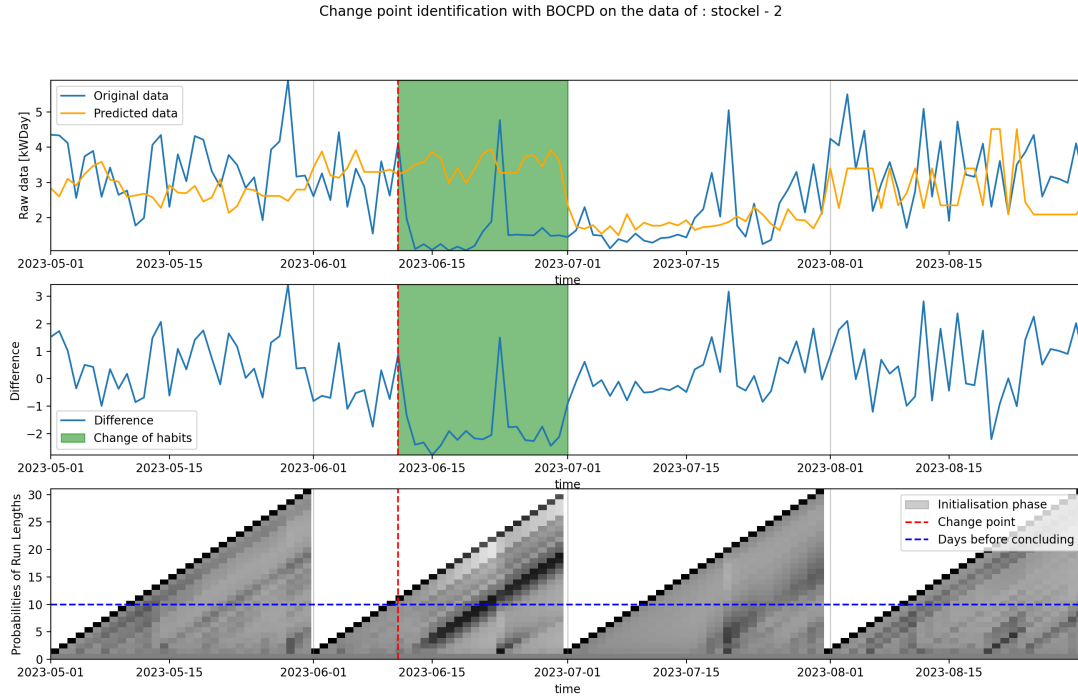


Figure A.3: Change point identification Stockel dataset 2

A remarkable detail about this graph is the presence of a spike in consumption during the habit change period. We notify a clear black point in the third plot, just under the spike, that could have resulted in a new statistical run if the change lasted longer.

A.5.4 Stockel dataset 3

Change point identification with BOCPD on the data of : stockel - 4

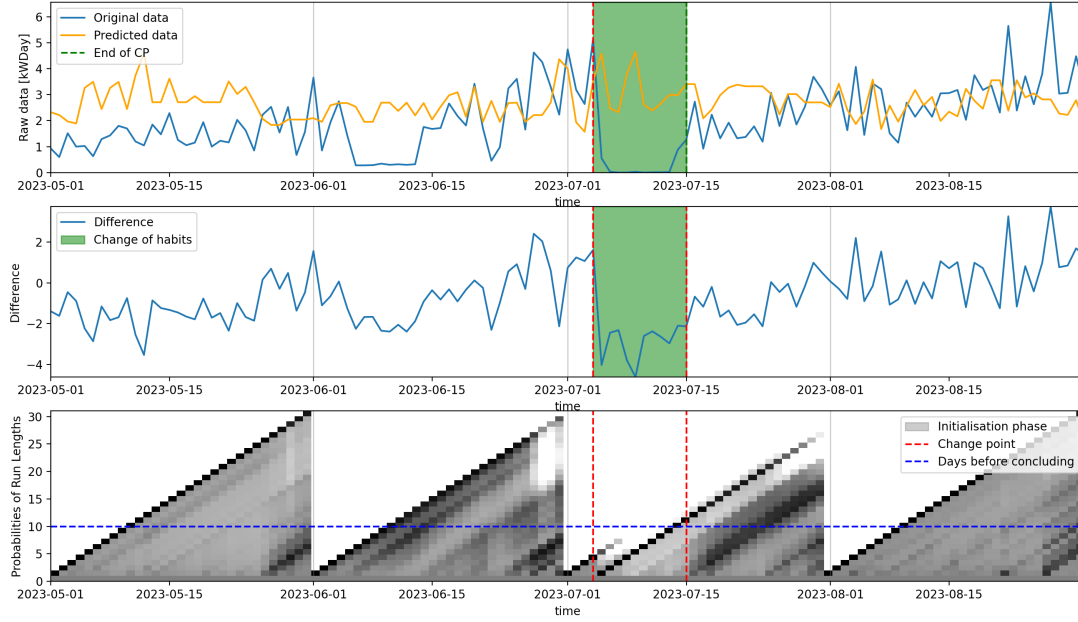


Figure A.4: Change point identification Stockel dataset 3

In this graph, we can notice that the forecast predicts too much consumption right at the beginning of the first month. In this case, the BOCPD detects no CP because no change in the statistical run has occurred. This is a limitation of the performances of the algorithm.

A.5.5 Stockel dataset 4

Change point identification with BOCPD on the data of : stockel - 5

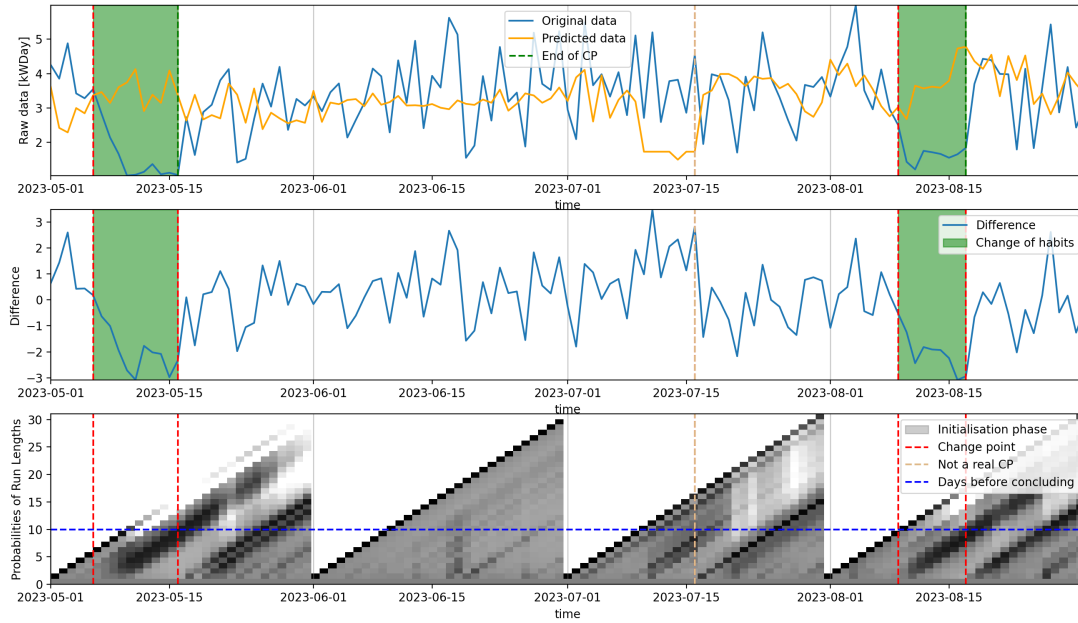


Figure A.5: Change point identification Stockel dataset 4

Here, a CP is detected and identified as a false alarm right away. This is the same case as for the figure [A.1](#).

A.5.6 Stockel dataset 5

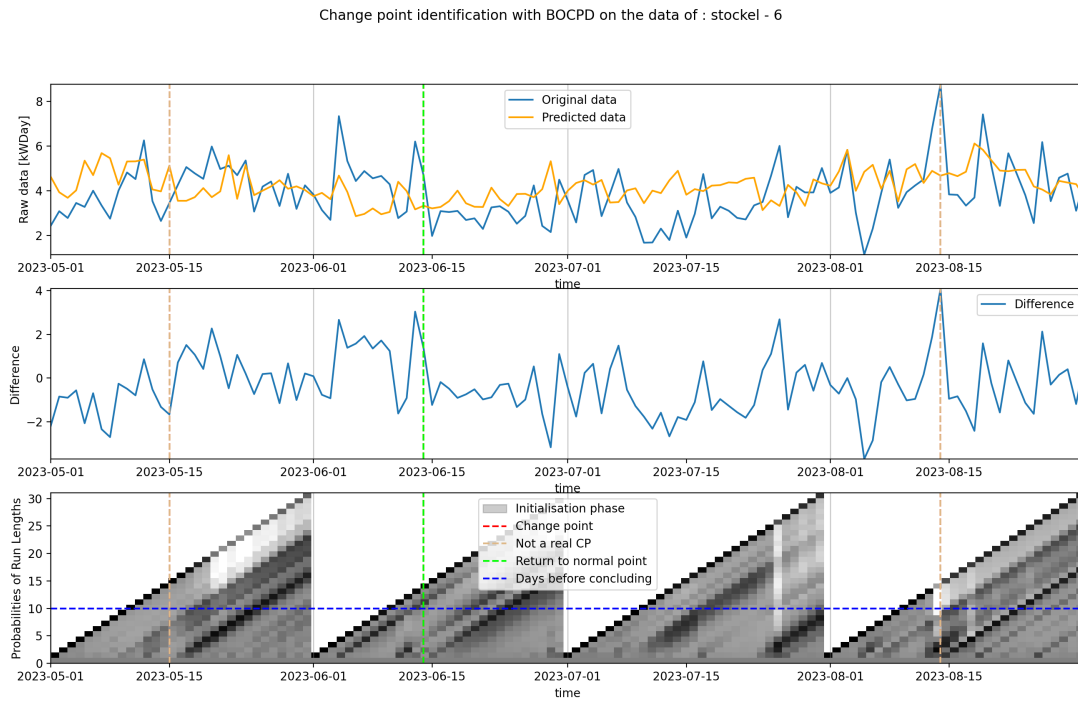


Figure A.6: Change point identification Stockel dataset 5

in this graph, the false alarms clearly show that the identification algorithm is doing its job properly. There is no need for the forecast algorithm to be retrained in these cases.

A.5.7 Stockel dataset 6

Change point identification with BOCPD on the data of : stockel - 7

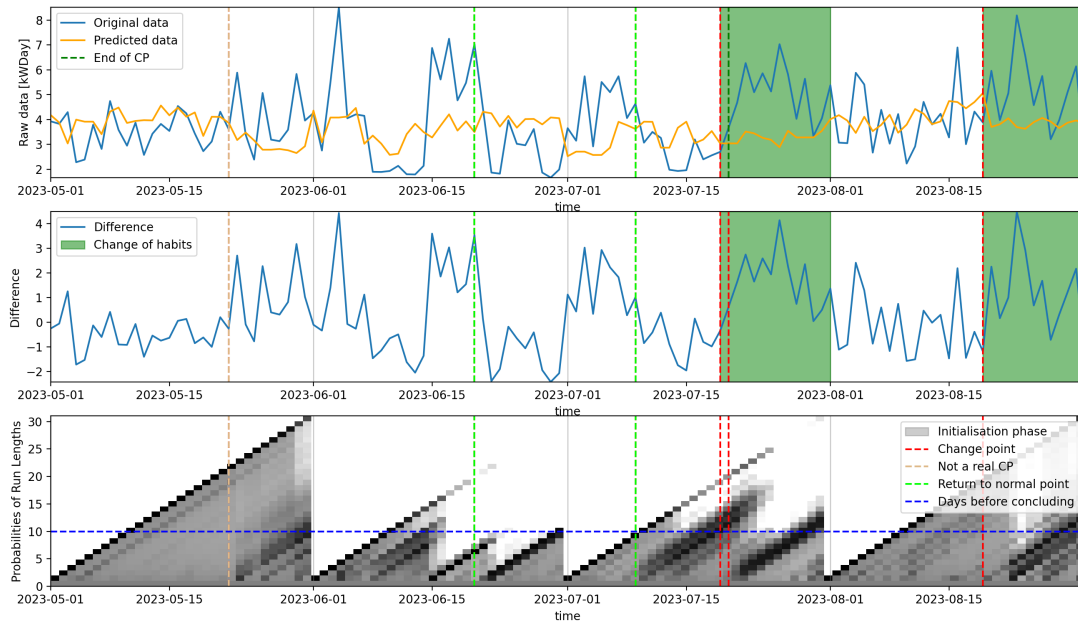


Figure A.7: Change point identification Stockel dataset 6

A.5.8 Greenbizz dataset 0

Change point identification with BOCPD on the data of : greenbizz - consommation1

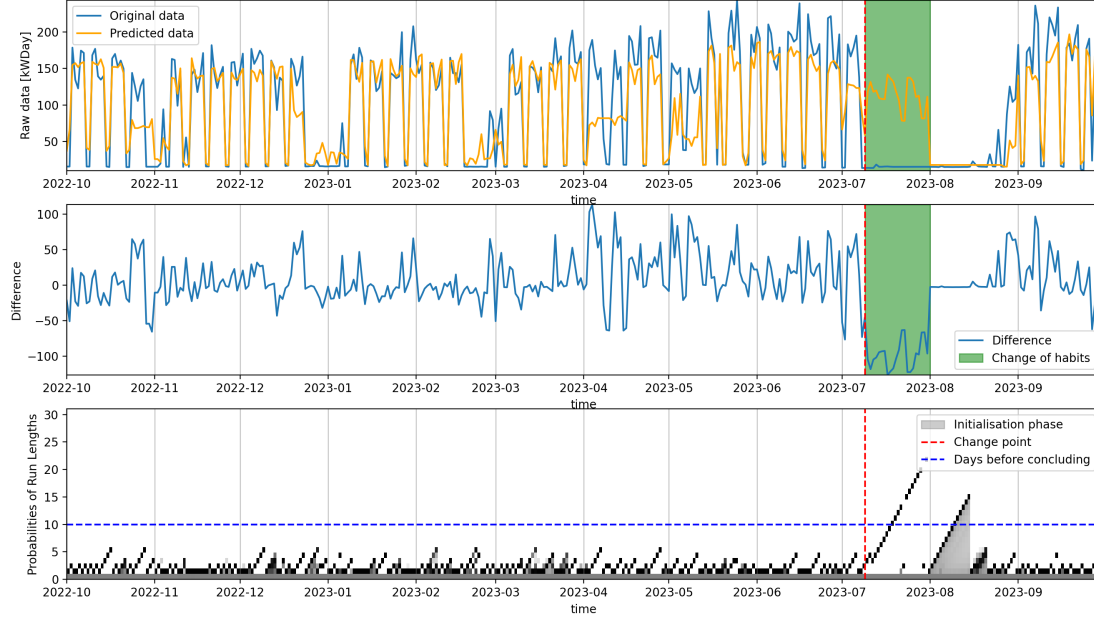


Figure A.8: Change point identification Greenbizz dataset 0

This is the case reviewed in the chapter 4. In addition to the analysis made in the chapter, we note the visualisation of the BOCPD algorithm identifies a lot of statistical runs. This is a different behavior than the Stockel datasets, but it also does the job, as it clearly identifies the habit change introduced in the chapter. It is interesting to note that the algorithm has difficulties to identify different run lengths in the graph, but as soon as the C appears, the statistical run length then becomes the biggest one in the whole dataset.

A.5.9 Greenbizz dataset 1

Change point identification with BOCPD on the data of : greenbizz - consommation2

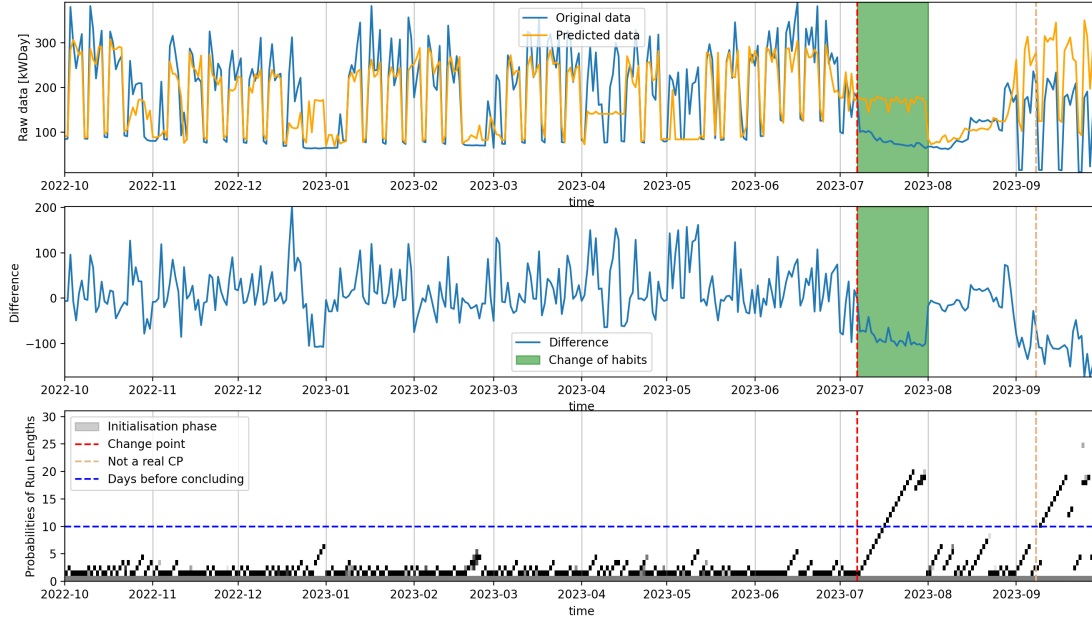


Figure A.9: Change point identification Greenbizz dataset 1

Here, the interesting part happens in the last month, where a CP is categorized as not important because the mean of the forecast and the real consumption does not differ enough, before and after. However, there is a clear difference between the forecast values and the real consumption, but it happened right at the beginning of the month so the model did not understand this case well enough. This case shows a limitation of the CPI model developed in this project.

A.5.10 Greenbizz dataset 2

Change point identification with BOCPD on the data of : greenbizz - consommation3

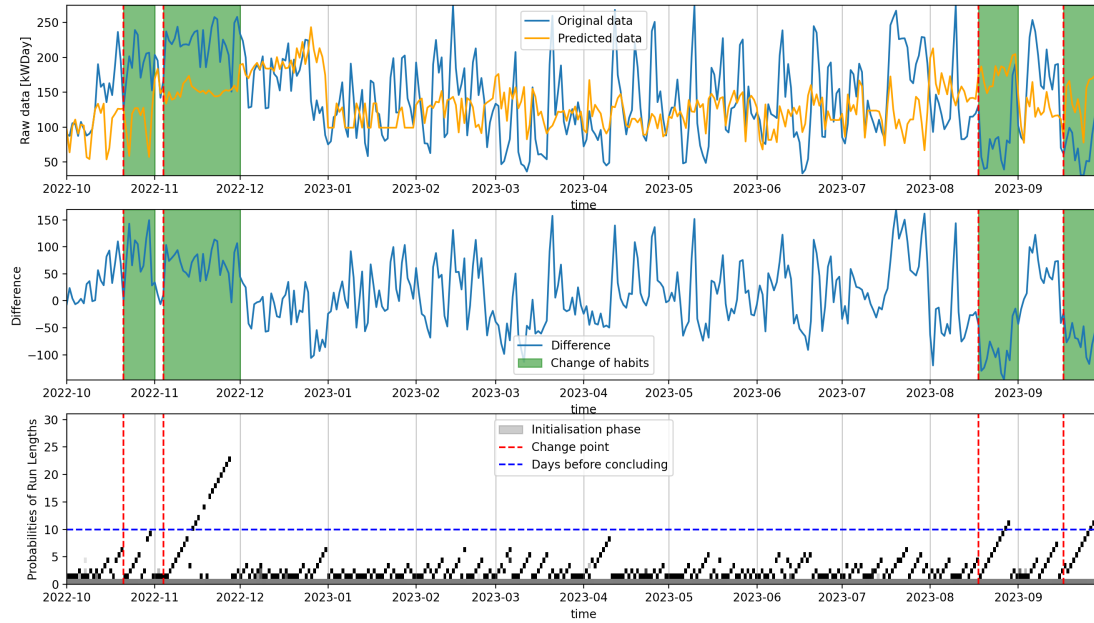


Figure A.10: Change point identification Greenbizz dataset 2

Despite having a very noisy graph, the BOCPD algorithm does a fine job of detecting long periods where the consumption or forecast are not in phase.

A.5.11 Greenbizz dataset 3

Change point identification with BOCPD on the data of : greenbizz - consommation4

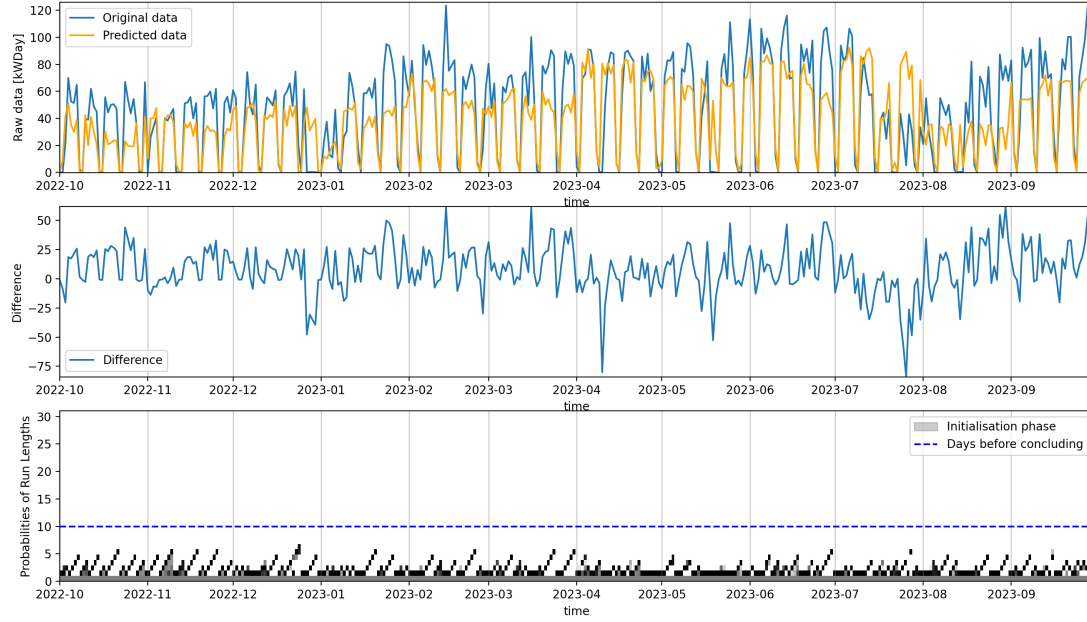


Figure A.11: Change point identification Greenbizz dataset 3

This dataset shows a typical office consumption with very precise consumption hours. This is a case where the forecast algorithm can have a very good precision, and we notice that there is no CP detected by the BOCPD, which seems correct.

A.5.12 Greenbizz dataset 4

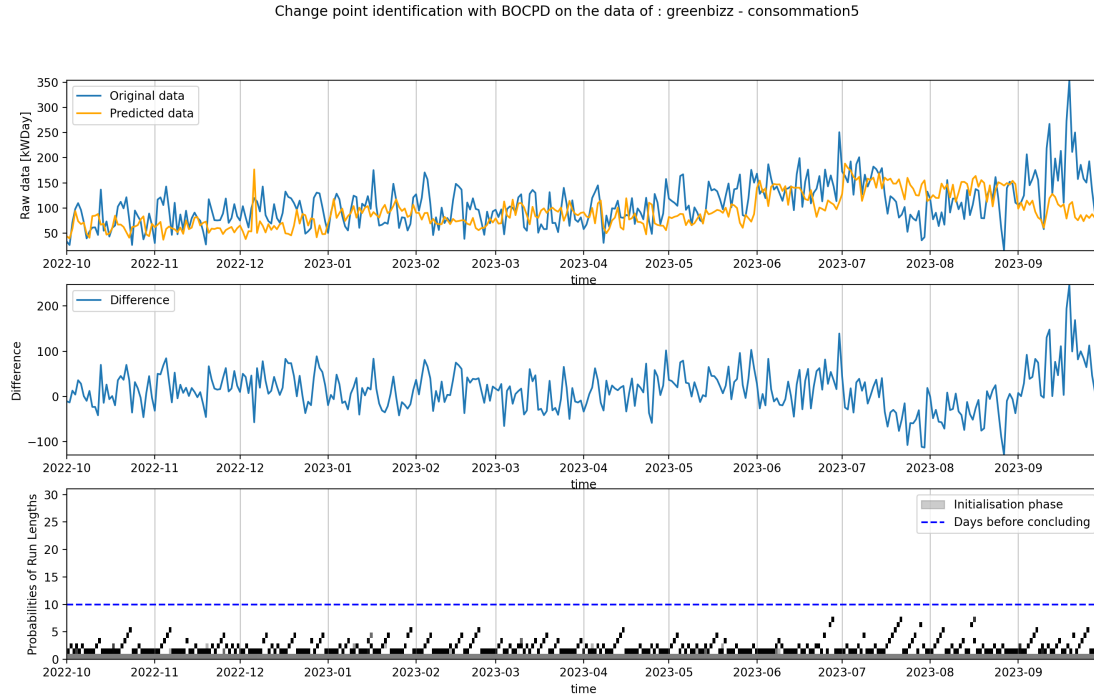


Figure A.12: Change point identification Greenbizz dataset 4

Again, the consumption data to analyse looks very noisy and makes it hard for the forecast to do a precise job. There is no CP detected because the statistical run lengths are too small. We notice that the forecast doesn't predict enough consumption, but the statistical run lengths changes too often to detect a CP. In this case, this could be a limitation of the BOCPD algorithm, rather than the CPI algorithm as in the dataset 1 of Greenbizz [A.5.9](#). On the other hand, maybe there wasn't a real change of habit that occurred at the last month, but the primary focus of the algorithm is to detect change of habits, so it's not wrong either. In this case, maybe the objectives of the algorithm should be reviewed to get a better result.

A.5.13 Greenbizz dataset 5

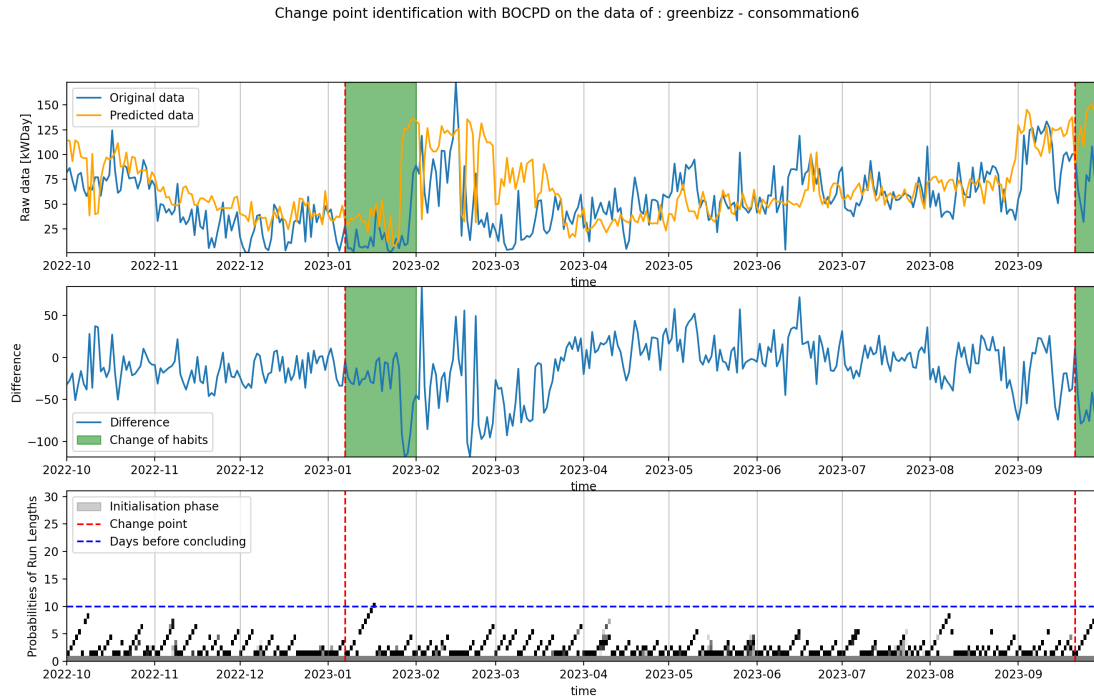


Figure A.13: Change point identification Greenbizz dataset 5

This last graph also shows a very noisy data. The BOCPD finds two CPs where the forecast predicts too little consumption, so they are classified as habit changes. There could also be other periods classified as CP, but the data is already too noisy at the beginning of the month, so the BOCPD interprets it as normal, and don't raise any CP.

A.6 pseudo code of the excess production indicator

Algorithm 1 ExcessProductionIndicator

```
1: Input: The injection [kWh/15min] at the current time  $T$  :  $injT$ 
2: Input: The sum of all consumption [KWh/15min] at the current time  $T$  :  $consT$ 
3: Input: The list of forecast injection [kWh/15min] for the rest of the month :  $predInj$ 
4: Input: The list of forecast of all consumption [KWh/15min] for the rest of the month :  $predCons$ 
5: Input: The minimal fraction of production to set the indicator to green  $f$ 
6: Output: Indication of consumption

7: function EXCESSPRODUCTIONINDICATOR( $injT$ ,  $consT$ ,  $predInj$ ,  $predCons$ )
8:   if  $consT < injT$  then
9:     bool  $consumptionCond \leftarrow$ 
10:  $consT * factorTrusting < predCons < consT * (2 - factorTrusting)$ 
11:     bool  $injectionCond \leftarrow$ 
12:  $injT * factorTrusting < predInj < injT * (2 - factorTrusting)$ 
13:     if  $consumptionCond$  and  $injectionCond$  then
14:       int  $i \leftarrow 1$ 
15:       while  $predInj[i] > predCon[i]$  and  $i < 12$  do
16:          $i \leftarrow i + 1$ 
17:       end while
18:       return "Safe to consume for the next  $i * 15$  minutes"
19:     end if
20:   else
21:     return "The consumption is higher than the production"
22:   end if
23: end function
```

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl