PRELUDE

Prescient building Operation utilizing Real Time data for Energy Dynamic Optimization

WP5 – Scale up and integration

v1.0

D5.3 – Web-app tool for proactive building integration in DH networks

Issue date:	31/03/2023
Author(s):	Hicham Johra, Daniel Leiria, Markus Schaffer, Martin Veit, Anna Marszal-Pomianowska, Michal Zbigniew Pomianowski
Editor:	Hicham Johra (Aalborg University)
Lead Beneficiary:	AAU – Aalborg University
Dissemination level:	Public
Type:	Other
Reviewers:	Pablo de la Fuente Casal (1A INGENIEROS S.L.P), Christian Heschl (FORSCHUNG BURGENLAND GMBH)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement N° 958345

PRELUDE KEY FACTS

Project Title	Prescient building Operation utilizing Real Time data for Energy Dynamic Optimization
Starting date	01/12/2020
Duration in months	42
Call (part) identifier	H2020-NMBP-ST-IND-2020-singlestage
Topic	LC-EEB-07-2020 Smart Operation of Proactive Residential Buildings (IA)
Fixed EC Keywords	-
Free Keywords	Free running, model based predicted control, dynamic building simulation, demand side flexibility, proactive buildings, predictive maintenance, occupancy models, smartness assessment
Consortium	21 organisations

PRELUDE CONSORTIUM PARTNERS

	Participant organisation name	Country
1	AALBORG UNIVERSITET	DK
2	TAMPEREEN KORKEAKOULUSAATIO SR	FI
3	ASOCIACIÓN DE INVESTIGACIÓN METALÚRGICA DEL NOROESTE	ES
4	POLITECNICO DI TORINO	IT
5	FORSCHUNG BURGENLAND GMBH	AT
6	UNISMART - FONDAZIONE UNIVERSITÀ DEGLI STUDI DI PADOVA	IT
7	BRUNEL UNIVERSITY LONDON	UK
8	EMTECH DIASTIMIKI MONOPROSOPI IDIOTIKI ETAIREIA	EL
9	CORE INNOVATION AND TECHNOLOGY OE	EL
10	ESTIA SA	CH
11	EUROCORE CONSULTING	BE
12	IREN SMART SOLUTIONS SPA	IT
13	LIBRA AI TECHNOLOGIES PRIVATE IDIOTIKI KEFALAIOUCHIKI ETAIREIA	EL
14	STAM SRL	IT
15	LA SIA SRL	IT
16	TREE TECHNOLOGY SA	ES
17	1A INGENIEROS S.L.P	ES
18	DIMOS ATHINAION EPICHEIRISI MICHANOGRAFISIS	EL
19	BLOK ARCHITEKCI SPOLKA Z OGRANICZONA ODPOWIEDZIALNOSCIA	PL
20	CAISSE DE PREVOYANCE DE L'ETAT DE GENEVE	CH
21	INNOVACION Y CONSULTING TECNOLOGICOSL	ES
22	CORE INNOVATION CENTER NPO	GR

EXECUTIVE SUMMARY

In recent years, the amount of data generated in buildings and their energy systems has increased dramatically. However, merely having access to large amounts of data is not a guarantee of improved performance or correct decision-making towards energy efficiency. Therefore, to derive valuable insights from data, it is essential to develop tools that can effectively analyze data and visualize key metrics to provide a higher level of understanding of the building and systems' performance.

This document presents the development process (methodologies and algorithms), features, and architecture of the PRELUDE District Heating Meter Data Analysis web-app. This web-app has been created to help the building owners and district heating utility companies analyze the data collected by heat meters and thus assess the performance of buildings connected to the district heating network and identify faulty systems or building operations.

Several features of the PRELUDE District Heating Meter Data Analysis web-app have been implemented. They are based on data analysis methods and algorithms elaborated and validated within the PRELUDE project in the scope of WP 5 (Task 5.3) and published in peer-reviewed scientific articles (Schaffer, Tvedebrink and Marszal-Pomianowska, 2022; Leiria et al., 2022; Leiria et al., 2023). These methods are related to the following topics:

- Algorithm for disaggregation of heating energy usage profile for space heating and domestic hot water production from the total heating energy use.
- Overview of the different key performance indicators of multiple buildings connected to the district heating grid on an interactive map.
- Overview of the different key performance indicators of an individual building regarding its heating usage.
- Fault detection in the heating installations of buildings connected to the district heating network.
- Support building and heating systems energy performance analysis and decision-making.

The further development of the tool, tuning, and implementation of additional analysis methods is foreseen in the scope of WP 7, where District Heating data and specific cases will be used as demonstration cases. The tool is applied to datasets of the Aalborg district heating network to inform the utility company about possible problematic situations and faults detected within their client portfolio. The continuous improvement of the web-app tool aims to automatically diagnose faulty buildings through machine learning techniques and interoperability with the FusiX platform. The tool can be used alone or in conjunction with other FusiX applications. An API will be implemented to enable the direct use of the analysis algorithms implemented in the web-app.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
DH	District Heating
DHW	Domestic Hot Water
DST	Daylight-saving time
SH	Space Heating
SHM	Smart Heat Meter(s)
SQL	Structured Query Language
SPMS	Smooth - Pointwise Move - Scale
SVR	Support vector regression

1. INTRODUCTION

Over the past few years, there has been a significant increase in the amount of data being generated by buildings and their energy systems. However, simply having access to large amounts of data does not guarantee improved performance or effective decision-making regarding energy efficiency. To extract valuable insights from this data, it is necessary to develop tools that can efficiently analyze and visualize key metrics, providing a better understanding of building and system performance.

One potential source of valuable data is the recently installed smart heat meters in district heating networks. These meters generate a substantial amount of data that can provide insight into the buildings' dynamic performance, including identifying system errors and faulty operation, as well as customer features such as occupancy profiles and heating practices. By exploiting the hourly data on heat usage and combining it with other readily available information, such as weather data, public building registry or energy performance reports, there is an opportunity to create new business models that offer cost-effective, customized and user-centric solutions to optimize energy usage, enable proactive building management, and generates renovation roadmaps for building clusters.

Currently, smart meter data is primarily used for billing purposes, and the potential value of this data is being overlooked. Utilities and smart heat meter providers lack the necessary tools and methods to analyze the data effectively and leverage it to create value for their customers. Therefore, there is a clear need to develop multi-objective methods that use smart meter data to enable all buildings, regardless of their technology level, to become an active part of the energy community and help with reaching the sustainability goals of the latter.

These methods should not only aim to identify critical customers but also to identify the reasons behind these problematic energy demand profiles or DH sub-stations' behaviour such as errors in the heating or domestic hot water production systems, unusual end-user practices (e.g., using a single radiator to heat the whole house). This information can help utilities establish communication with customers, provide advice and feedback on energy management, and propose remedies that can be easily implemented with little effort or expense.

Finally, these methods can be implemented into a web-based tool that supports customer activation by sending alerts or informative messages indicating potential failures and encouraging users to self-manage the energy performance of their building. This would turn traditional utility company and customer business models into collaborative communities where everyone contributes to a reliable, cost-effective and sustainable district heating network that can operate at lower supply temperatures, minimize heat losses and integrate renewable energy sources.

The original title of this deliverable was "Tools for DH peak power and consumption management". This title has been changed to "Web-app tool for proactive building integration in DH networks" to better represent the intent and capabilities of the tool developed within WP5 and described in the present document.

2. CORE OF THE DELIVERABLE

2.1 CONCEPT AND OBJECTIVES

The Task 5.3 of the PRELUDE project focuses on the use and analysis of data readily available from smart heating meters (SHM) that are now systematically installed in the buildings connected to district heating (DH) networks. These SHMs produce a large amount of data which, if employed correctly, can be very useful to assess the performance of buildings connected to the heating grid. To address the challenge of processing that data to get a better insight into the building stock, a web-based application tool is developed to enable the simple use of innovative data analysis methods and algorithms. This application aims to provide building owners and DH utility companies with the capabilities they need to extract information from their data without requiring specialized knowledge in data analysis. The District Heating Meter Data Analysis Web-App is part of a collection of building data analysis tools gathered on the FusiX platform, which, in turn, allows interoperability between the different tools developed by PRELUDE the project (see Figure 1).

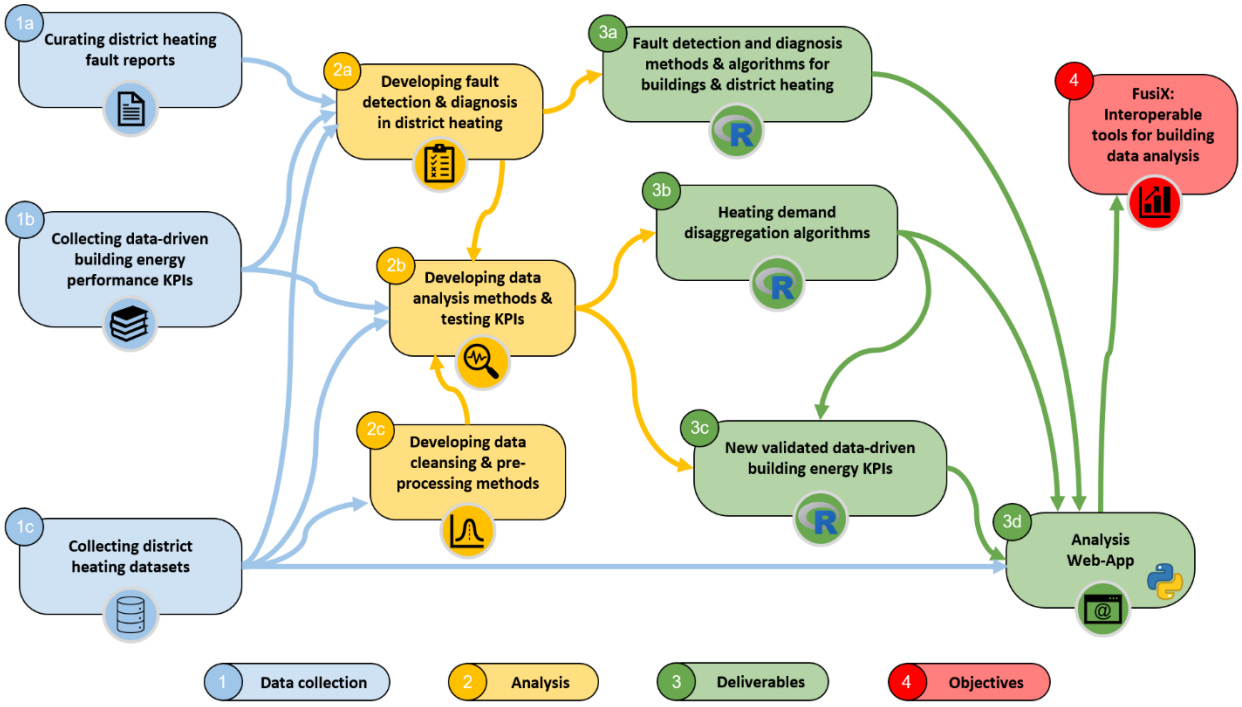


Figure 1: Workflow of WP5.3 – Proactive building integration in district heating networks.

In particular, this web-app can perform the following data treatment and analysis:

- Support building and sub-station operation energy performance analysis.
- Fault detection in buildings connected to the district heating.
- Disaggregation of heating energy usage profile for space heating and domestic hot water production from the total heating energy use and, as a result, provide more appropriate information for renovation/intervention roadmap.
- Overview of the different key performance indicators of multiple buildings on an interactive map.
- Overview of the different key performance indicators of a single building.

2.2 DATA FLOW AND DATA MANAGEMENT

The web-app tool has been designed to provide users with various features, including data visualization and predictive calculations, to improve decision support. However, these features rely heavily on the quality and completeness of the data inputs. Therefore, it is important to carefully consider the data requirements before using the tool to avoid potential errors or inaccuracies in the results.

This report section will provide a comprehensive overview of the data requirements for the web-app tool, including the type of data required, data sources, and data quality standards. By understanding these requirements, users can ensure that they can fully leverage the tool's capabilities and make informed decisions based on the insights provided.

The dataset from the SHM and the weather station must respect the following requirements:

- **Time resolution:** For all the different algorithms (data pre-treatment, heating demand disaggregation, and data visualization) in the tool, the necessary sampling time resolution is one hour. However, in the data visualization section (after the calculations), the user can select other resolutions, e.g., daily, weekly, or monthly. Furthermore, the measurements should also be equidistant from each other. The pre-treatment algorithm has an integrated process to shape the measurements in this manner if the data entries do not follow this condition.
- **Data resolution:** The smart energy meter data should be collected at a high level of resolution. However, in most cases, the measurement data is restricted to a recording resolution of 1 kWh with rounded-down measurements, as described in the data pre-treatment section. Therefore, the Smooth - Pointwise Move - Scale (SPMS) algorithm was integrated into the tool to increase the dataset's resolution before applying the heating demand disaggregation.
- **Missing values quantity:** A maximum of 2% of data is allowed to be missing.
- **Data features:** The measurements from the SHM (i.e., the energy, the water volume, the volume times the supply temperature, and the volume times the return temperature) must be cumulative to be able to use the missing data imputation technique in the pre-treatment stage. Regarding the required data inputs for the heating disaggregation and visualization, one can read Table 1.

Table 1: List of required data inputs.

Input category	Inputs	Input units	Heating demand disaggregation	DH customers mapping	Individual customer analysis
Building features	Heated floor area	m^2		X	X
	Building address	-		X	
	Latitude and longitude	º		X	
Weather data	Outdoor temperature	ºC	X		X
	Global solar radiation	W/m^2	X		
Smart heat meters	Energy usage	kWh	X	X	X
	Water volume	m^3		X	X
	Volume x Supply temperature*	m^3ºC		X	X
	Volume x Return temperature*	m^3ºC		X	X

*The temperature difference is calculated by the subtraction between supply and return of these parameters divided by the water volume mentioned in the table.

There are two different procedures to import the data into the web-app (see Figure 2):

1. The user manually uploads the different input raw data files (heat meter recordings, local weather data, building characteristics, building geolocation) from a local computer onto the web-app.
2. The user imports pre-treated and pre-analysed data of a large building portfolio from a secure SQL (Structured Query Language) database onto the web-app. This is a secure and GDPR-compliant way of managing sensitive heat meter data with building location, building information and household information. The data is pre-treated and pre-analysed on a secure local computer of the utility company before being loaded onto a secure server.

Various programs can be used to import pre-analyzed data from the SQL database, one of which is Python. To establish a connection with the database, a username and password provided by the server administrator, along with a specific ID for the database, are required.

The database comprises different tables, some of which have password protection while others do not. A password is necessary for tables with sensitive data, allowing authorized users with access and the correct password to query the data according to their requirements efficiently. In cases where no password is required for a table in the database, only a username and password to the server are necessary.

A code example and a simple description of the database can be found in Appendix A.

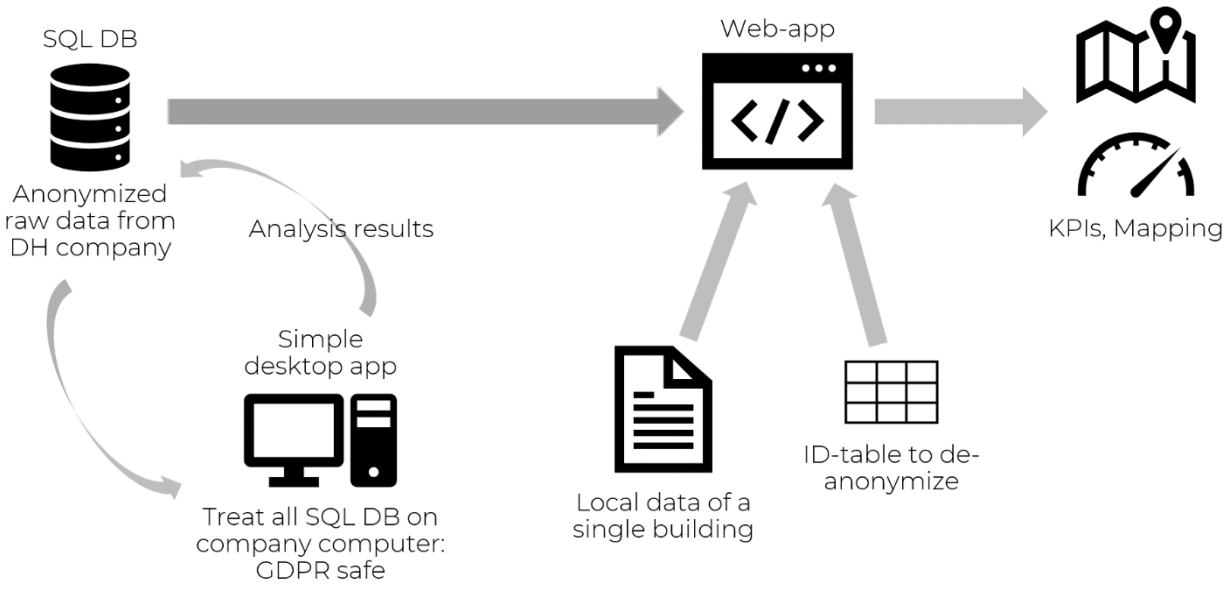


Figure 2: Data management and data flow for the PRELUDE District Heating Meter Data Analysis web-app.

2.3 PRE-ANALYSED DATA FLOW FROM AND TO THE SECURED SQL DATABASE

Smart energy meters must follow strict guidelines regarding their technical reliability and measurement accuracy (EN 1434). Nevertheless, due to errors in the data collection infrastructure or data warehouses of utility companies, it cannot be assumed that the data is error-free or complete. Furthermore, smart energy meter data is commonly not equidistant, hindering or complicating data use. Therefore, data pre-treatment is necessary.

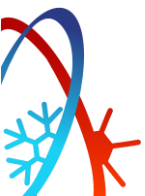
The data treatment used in this application is based on the framework developed by Schaffer, Tvedebrink and Marszal-Pomianowska (2022). First, the cumulative trend of the data is verified, and data points that do not obey this trend are removed. As the second optional step, all data points between 2:00 and 3:00 on the day when the daylight-saving time (DST) ends are removed. This optional processing step is necessary if the data does not include information about the time zone. Missing time zone information leads to the problem that datapoint between 2:00 and 3:00 can be both summer- and wintertime, and their order is consequently no longer uniquely identifiable. As the third step, linear interpolation is used to interpolate all existing cumulative measurements to the next full hour.

Due to transmission problems and from the first two data processing steps, missing values can exist in the smart energy meter data. To impute these missing values, first, the hourly quantities are calculated from the cumulative measurements, and then a weighted smoothing average with linear weighting and a window size of 48 values is used for imputation. Additionally, the results are scaled so that they "fit" into the amount of missing quantity, which is known due to the cumulative nature of the data (as the missing data is only missing at the utility company, the cumulative data remains correct even if data is missing). From this data treatment, equidistant data points without erroneous values are obtained.

From recent research such as Broholt et al., 2022; Kristensen et al., 2018; Leiria et al., 2022, one frequently reported problem is the low resolution of the smart energy meters and transmitted data. Commonly the transmitted energy data has only a measurement resolution of 1 kWh, whereby all values are rounded down, i.e., all values between 1 and $1.\overline{9}$ are rounded down to 1.0. This reduction in resolution is made to save bandwidth when transmitting data and to fit into utility companies' existing billing structures. While this reduced resolution does not hinder the current primary usage of energy smart meter data, which is billing, it can create a significant problem if data is analyzed on an hourly level, as it can introduce an on/off-like pattern if the energy use is relatively low. To mitigate this problem, Schaffer et al. (2023) proposed the method SPMS, which combines smoothing with a ruleset. SPMS uses first a moving average, with a centred window of size 5 with a linear weighting for smoothing. After the smoothing, it is ensured that the maximum pointwise deviation to the transmitted data is not greater than ±0.4 kWh and that over one day, the obtained data does accumulate to the same amount as the original smart energy meter data.

To upload the processed data onto the PostgreSQL server, a secure local computer uses a Python script specifically made for the specific database. While uploading data, the virtual tunnel must be open, and a specific identifier to the database on the server is needed. A connection to the database therefore requires a specific username and password, an identifier for the database, and a virtual tunnel. Access to a specific database is granted only to users with access to the singular database. Users with access to the same server but a different database cannot query data.

The Python script to upload data opens a file, reads each line, and appends it to a list. The data is then committed to the specific database. This workflow is performed in parallel, meaning that the script can commit data from the secure local computer to the server from multiple files at the same time. Therefore, the data is split into multiple files which are parsed and committed to the server simultaneously.



2.4 FEATURES AND ANALYSIS METHODS IMPLEMENTED IN THE WEB-APP

In this section, a comprehensive overview of the various algorithms and methodologies that are implemented in the web-app tool can be found.

Heating demand disaggregation

Utility companies can easily access data collected from smart energy meters, providing valuable insight into heat load patterns in buildings. However, if only one meter is installed per household, as is the case in Denmark, the data collection process has a major drawback: it collects total heat usage without distinguishing between the demand for space heating (SH) and domestic hot water (DHW) production, which are essential for gaining a deeper insight into the building and its occupants. This is because SH demand depends on various factors, such as external weather conditions, construction features, occupants' comfort, and installed systems, while DHW is linked to people's routines and installed hot water production systems. Therefore, understanding these different heating demands is critical for making informed decisions on energy-saving strategies.

The heat demand disaggregation methodology is based on the initial assumption that the SH system constantly operates throughout the heating season, whereas the DHW usage occurs irregularly throughout the day. Therefore, out of approximately 24 heating measurements recorded in a day, a smaller number of them reflect the combined SH and DHW usage ($E_{Total} = E_{SH} + E_{DHW}$), while the remaining data points represent only the SH usage ($E_{Total} = E_{SH}$). To work with this premise, the methodology involves two stages. The first stage is to distinguish the data points that involve DHW production from those that do not (*Energy separation*). The second stage is to estimate the proportion of SH usage ($E_{SH,estim}$) in the data points identified with DHW usage (*Energy estimation*). In Figure 3, it is seen a scheme of the methodology process.

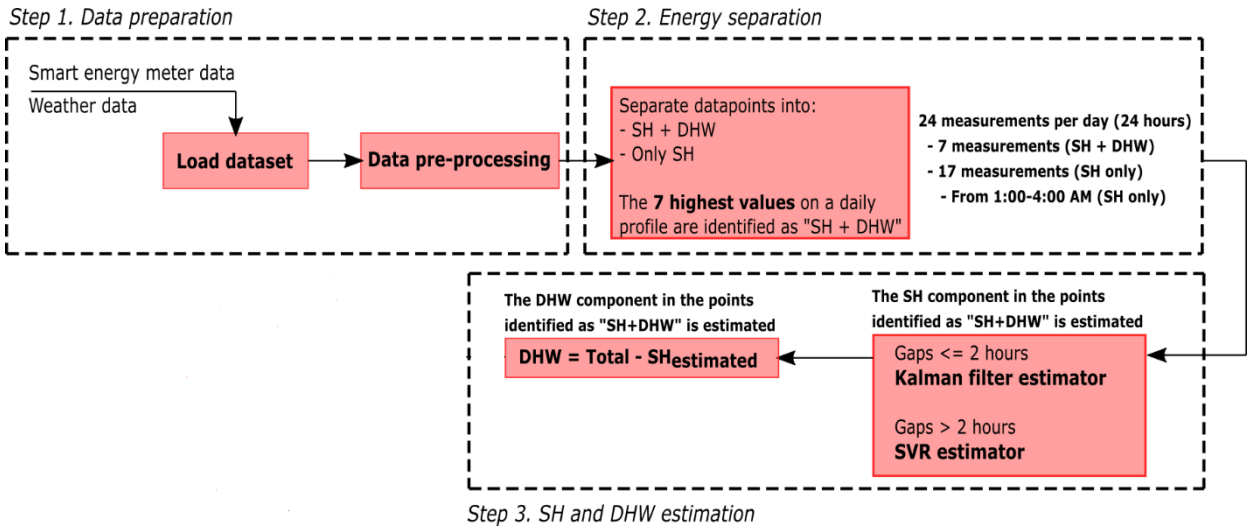


Figure 3: Representation of the heating disaggregation methodology.

To use this methodology, there are a few requirements that the dataset from the SHM must follow:

- Hourly resolution of energy measurements
- Measurements with decimal values (from *Data pre-treatment* step)
- Weather data (outdoor temperature and global solar radiation with hourly resolution)

Energy separation

The energy separation step of the method follows the same principle as Bacher et al. (2016), in which the slight variations in outdoor temperature and the building's inertia contribute to steady fluctuations in SH throughout the day. Therefore, all peaks captured by the meters can be attributed to DHW usage. As a result, the method identifies and records all daily highest points and categorizes them as SH and DHW usage ($E_{Total} = E_{SH} + E_{DHW}$). The method designates the top seven recorded values as DHW usage for each day, while the rest are considered SH usage solely. Additionally, there is a designated sleeping period from 1:00 - 4:00 every day, during which no DHW is required, and any high values recorded are due to the operation of the SH system. The separation method is depicted in Figure 4, where all points linked to DHW production are virtually removed from the dataset, and only SH measurements remain. These remaining SH data points are then utilized to estimate the SH from the removed recordings using the estimation algorithm (*Energy estimation*).

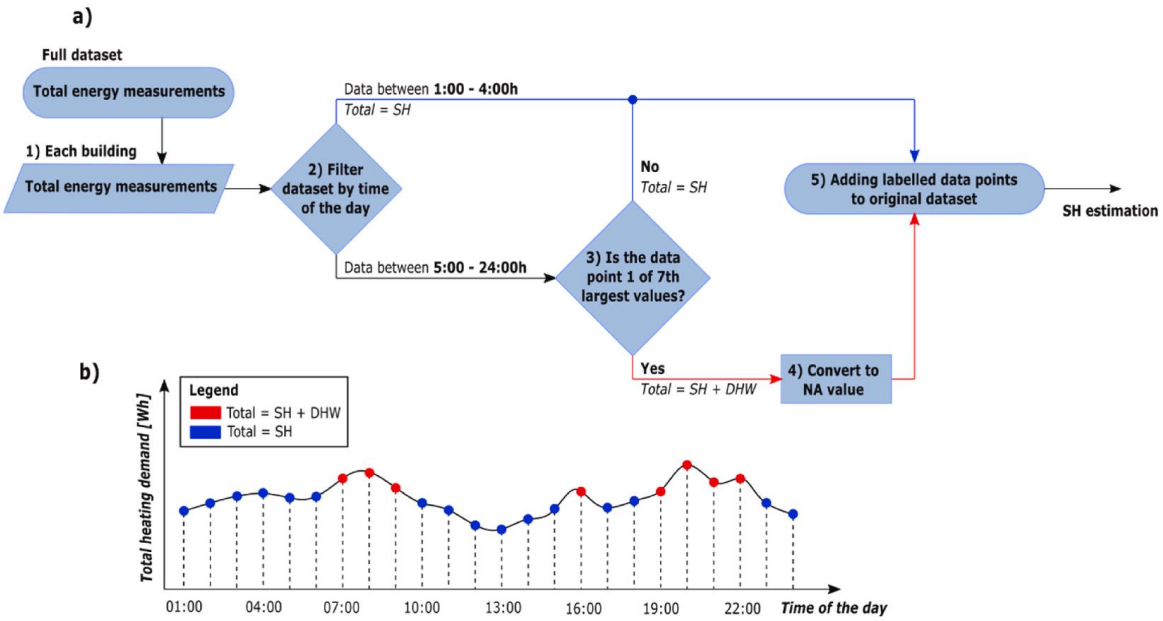


Figure 4: a) Data flow diagram; b) Method representation (Leiria et al., 2023).

Energy estimation

The energy estimation step involves calculating the SH usage ($E_{SH,estim}$) for the virtually removed data points to afterwards calculate the DHW usage ($E_{DHW,estim}$). Equation 1 is employed.

$$E_{DHW,estim} = E_{Total} - E_{SH,estim} \quad (1)$$

As per the same principle of energy separation, the SH demand fluctuates smoothly due to minor outdoor temperature oscillations. Therefore, the removed data points' SH proportion is predicted based on the neighboring SH measurements present in the dataset. To achieve this, a smoothed Kalman filter algorithm is used, basing it on a linear Gaussian state-space model for univariate time series from the "StructTS" function in the R-package *imputeTS* (Moritz and Bartz-Beielstein, 2017). According to Leiria et al. (2023), the Kalman smoothing technique effectively predicts the SH demand in missing values. However, relying on adjacent points for estimation can lead to inaccuracies when several points are removed sequentially, i.e., a large gap. To overcome this issue, the algorithm is optimized to use the smoothed Kalman filter only when the consecutive hours removed are equal to or below two hours (gap ≤ 2 hours). For larger data gaps, a support vector regression (SVR) is used instead. The SVR is a machine learning regressor that is trained with the known SH points remaining in the dataset and considers various inputs instead of only the neighbouring points to calculate the SH usage.

These inputs include the outdoor temperature, and global solar radiation measured two and one hours prior to the missing point, along with the smart meter measurements before and after the missing point. The SVR model employs a radial kernel function with parameters C (cost) and γ (gamma) set to 7 and 0.01, respectively. The SVR algorithm is obtained from the R-package *e1071* (Meyer, 2023). The estimation algorithms are presented in Table 2.

Table 2: Estimation methods' description.

Method	Parameters	Condition
Kalman filter	Model: StructTS Smoothed: True	Gap ≤ 2 hours
SVR	Kernel: Radial C = 7 ; γ = 0.01	Gap > 2 hours

DH customers' installations mapping

The web-based interface also provides an overview of the DH characteristics of a given cluster of buildings or cities. The interface allows the user to navigate the map and display overlaying coloured points representing various building characteristics, including the DH supply temperature, the temperature difference between supply and return, the yearly volume of water passing through the substation of the building, the yearly heating demand per floor heated area, clustering categories (according to what was defined by the user initially). The user can filter the visible building data points on the map by selecting the filtering range corresponding to the aforementioned parameters. The interface also enables the user to display a summary of the building characteristics (if added by the user to the initial dataset), yearly measurement values, together with the address. An example can be seen in Figure 5, retrieved from one of our published materials (Leiria et al., 2021).

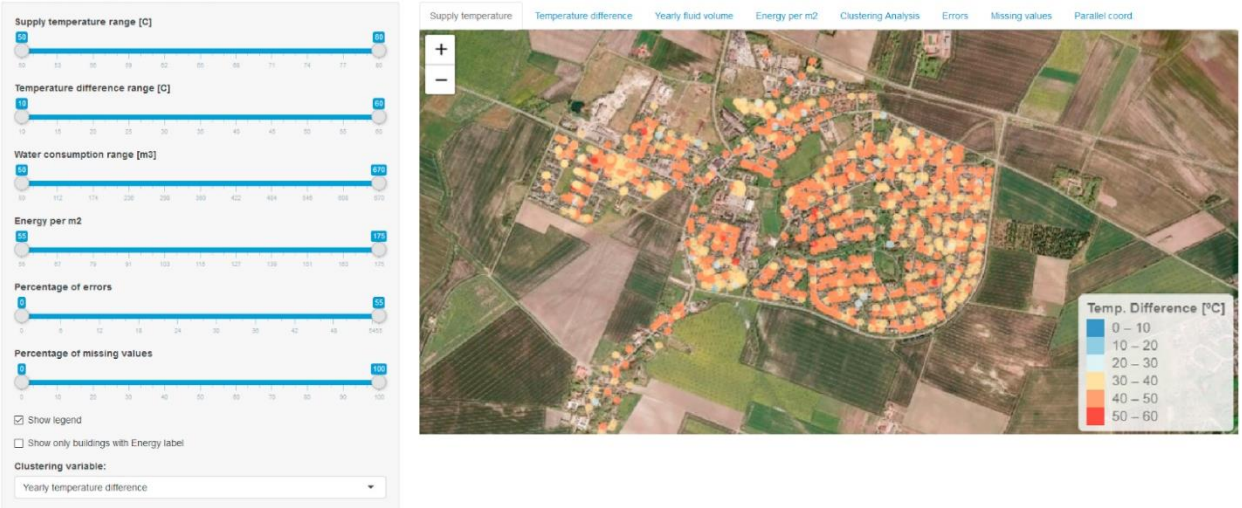


Figure 5: Example of SHM data mapped in a web-app tool in Shiny (Leiria et al., 2021).

Individual DH customer analysis

District heating (DH) systems are an essential part of modern urban infrastructure, providing an efficient and sustainable way of heating buildings. However, faults in DH substations can cause significant energy waste, resulting in increased operating costs and reduced system reliability. To tackle this problem, fault detection and diagnosis (FDD) has become a critical research area in the field of DH systems. In this context, we incorporate in the web-app tool certain indicators and algorithms to perform fault detection (FD) in DH substations.

Visualization of the measurements

The assessment is done individually per customer that has had a meter installed in the primary circuit of their heating installation. As shown in Figure 6, one should first select the ID attributed to the SHM (customer) and the data resolution of the measurements. The original resolution of the measurements is usually hourly and must be uploaded in the tool as such. However, the user can select other resolutions for the assessment, such as daily, weekly, and monthly.

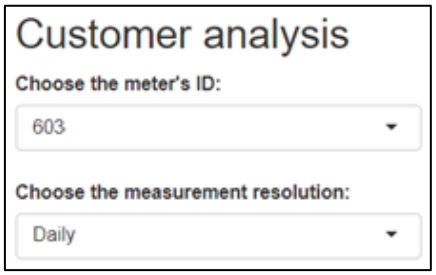


Figure 6: Drop-down list to select the customer's ID (anonymized) and the measurement resolution of the SHM.

After selecting the meter's ID and the data resolution, time-series interactive plots are generated for the measured energy, water volume, and temperature difference (ΔT). The time-series plots are an essential tool for visualizing and analyzing data from the SHM to gain insights into the performance of the DH system.

The energy time-series plot (Figure 7) shows the amount of energy used by the building over time, measured in kilowatt-hours (kWh). This plot can be used to identify trends and patterns in energy usage for SH and DHW demand, as well as to detect abnormal energy usage that may indicate a fault or inefficiency in the heating installation. Each plot also has integrated the time-series of the outdoor temperature.

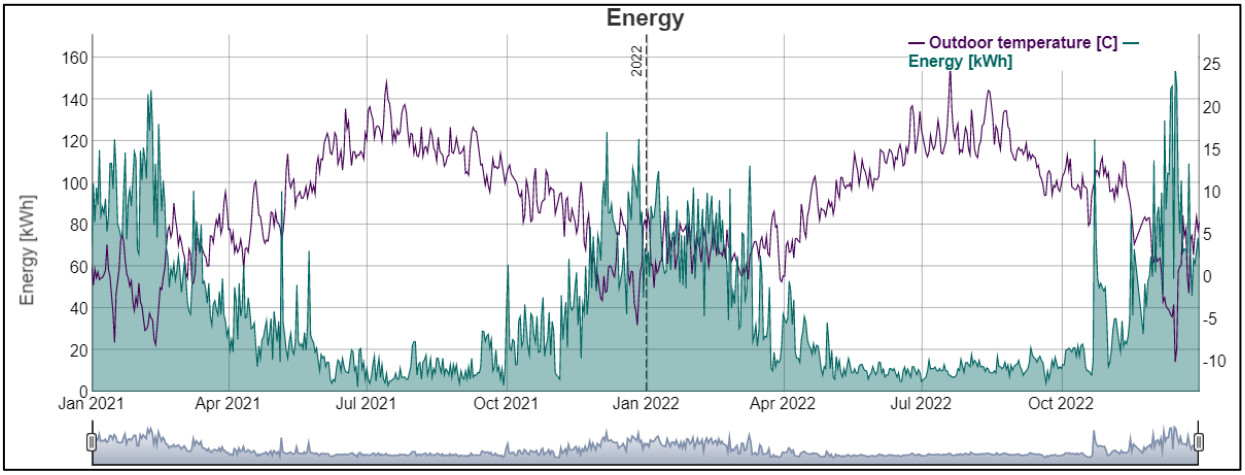


Figure 7: Time-series of the energy demand.

The water volume time-series plot (Figure 8) shows the amount of water flowing through the DH system over time, measured in cubic meters (m^3). This plot can be used to monitor the overall water usage of the system and detect changes in water flow that may indicate a fault or leak in the system.

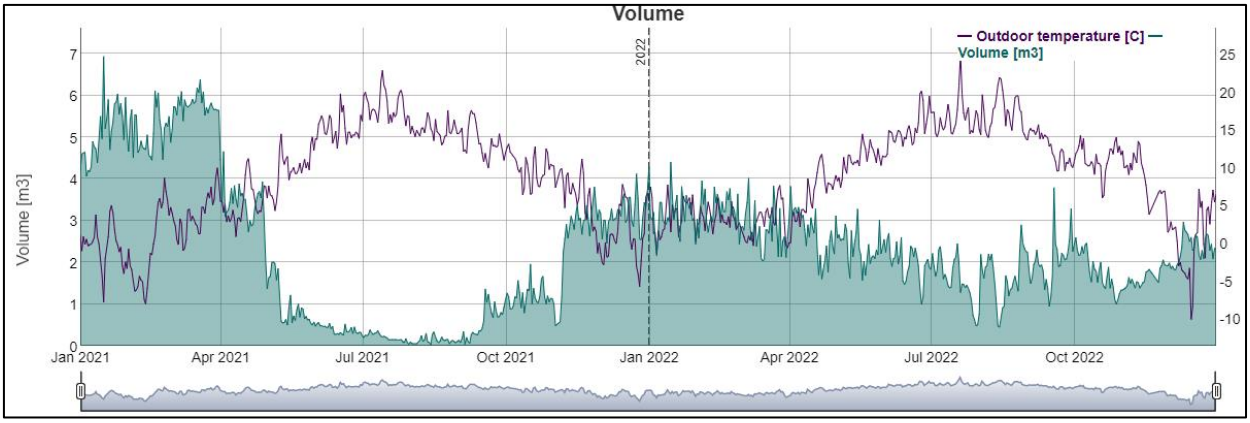


Figure 8: Time-series of the water volume.

The temperature difference (ΔT) time-series plot (Figure 9) shows the difference in temperature between the supply and return pipes of the DH system over time, measured in degrees Celsius (ºC). This plot can be used to monitor the efficiency of the DH system and detect changes in temperature that may indicate a fault or inefficiency in the system.

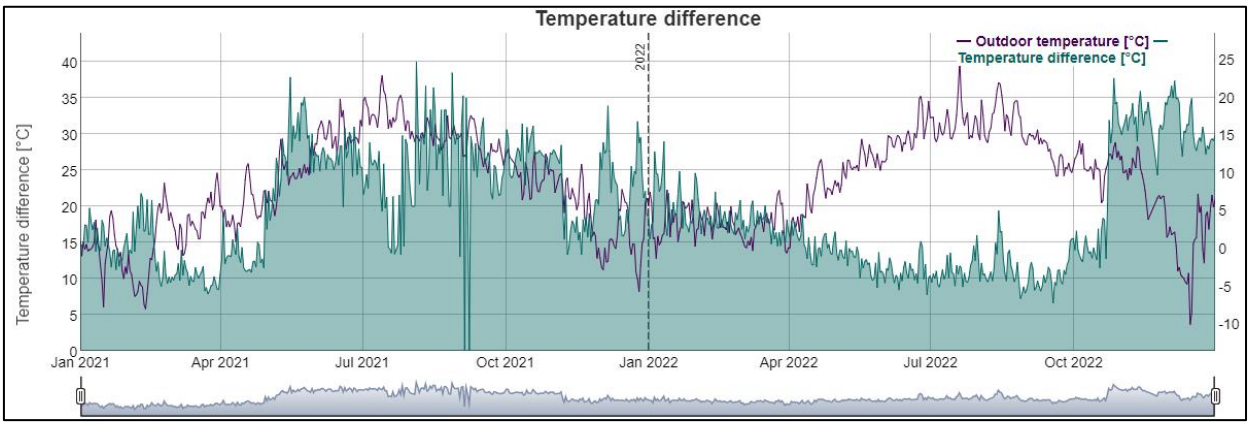


Figure 9: Time-series of the temperature difference.

By analyzing these time-series plots, one can gain valuable insights into the performance of the DH system, identify heating and no-heating seasons, possible faults or inefficiencies, and make appropriate decisions regarding their systems. The plots can also be used to track the effectiveness of corrective actions and monitor the system's ongoing performance, for example, before and after an intervention.

Visualization of control and fault detection indicators

In this section of the report, we present several plots generated by a web-tool app designed to detect faults in the heating installations of buildings. The plots included in this section provide insights into various aspects of the heating systems and are based on the literature review on the subject.

The indicators implemented in the tool are:

ΔT distribution

The ΔT parameter measures the temperature difference between the supply and return pipes in a heating system. The indicator displayed in the tool is the distribution of different ΔT-values during the heating and no-heating seasons throughout a user-defined time resolution (e.g., hours, days, weeks, etc.). As shown in Gadd and Werner (2014)., low ΔT-values are symptoms of faulty systems, and these values placement during the heating or the non-heating seasons represent different types of faults.

The heating season usually has ΔT higher than the non-heating season, as the SH and DHW systems operate simultaneously. However, if the largest concentration of low ΔT-values is in the heating season, then the fault might be located in the SH system. In contrast, during the non-heating season, DHW usage is the primary reason for heating demand. Therefore, if small ΔT measurements are recorded dominantly during the warmer months, the fault is due to the DHW production system. An example of this type of plot is seen in Figure 10.

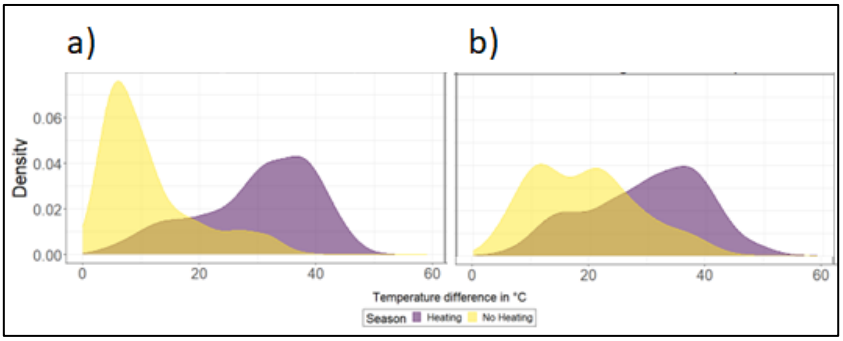


Figure 10: ΔT distribution for a) buildings with faulty DHW systems, b) buildings with faulty SH systems.

Energy signature

The energy signature plot is a scatterplot of a building's heating demand per m^2 of heated floor area (in kWh/m^2) versus the external temperature (in °C) and is used to detect faults in the heating installation by examining the relationship between these two variables. Ideally, the plot should show a clear negative correlation between heating demand and outdoor temperature for the colder temperatures and a zero correlation during the warmer months. However, if there is a faulty sensor or control valve, the graph may show erratic points or drops in heating demand that are not correlated with outdoor temperature. Moreover, if there is a blockage or leak in the heating system, the plot may show a reduced slope, indicating that the heating system is less responsive to changes in outside temperature. An example of this type of plot is seen in Figure 11.

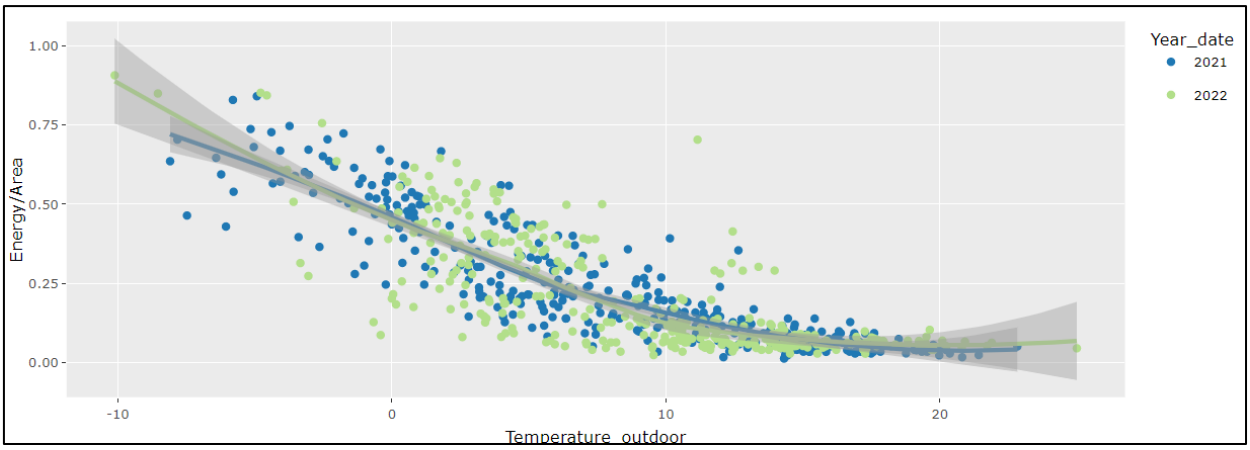


Figure 11: Example of the energy signature plot.

Supply temperature control

Similarly to the energy signature, the supply temperature (in °C) is plotted as a function of the ambient air temperature (in °C). The maximum at the left endpoint of the curve is constrained by the system's physical capabilities, while the minimum temperature to the right is set to provide a sufficiently high temperature

for domestic hot water usage without the risk of bacterial growth, such as Legionella. An example of this type of plot is seen in Figure 12.

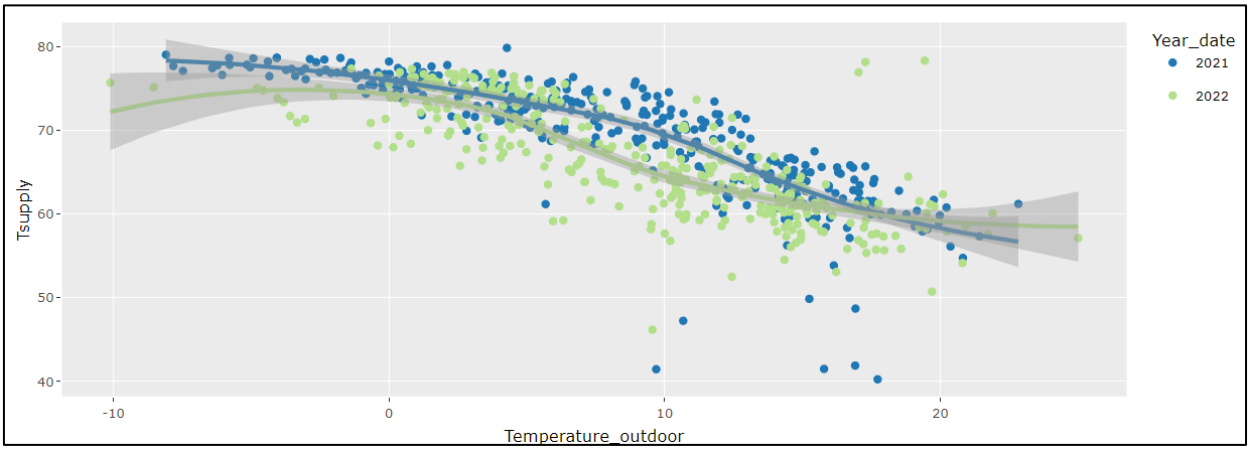


Figure 12: Example of the supply temperature control plot.

Hourly energy usage heatmap

The heatmap indicator is a powerful tool for analyzing energy demand in buildings and detecting heating control strategies. It is an effective method for identifying patterns in time series data that might be challenging to detect using conventional time series. The heatmap, also known as carpet plot, visually represents the data, where color-coded cells represent the energy demand [kWh] at different times. An example of this type of plot is seen in Figure 13. In this example, a carpet plot is created for a specific building for all 12 months (x-axis) and 30 days of each month (x–axis), and 24 hours per day (y-axis).

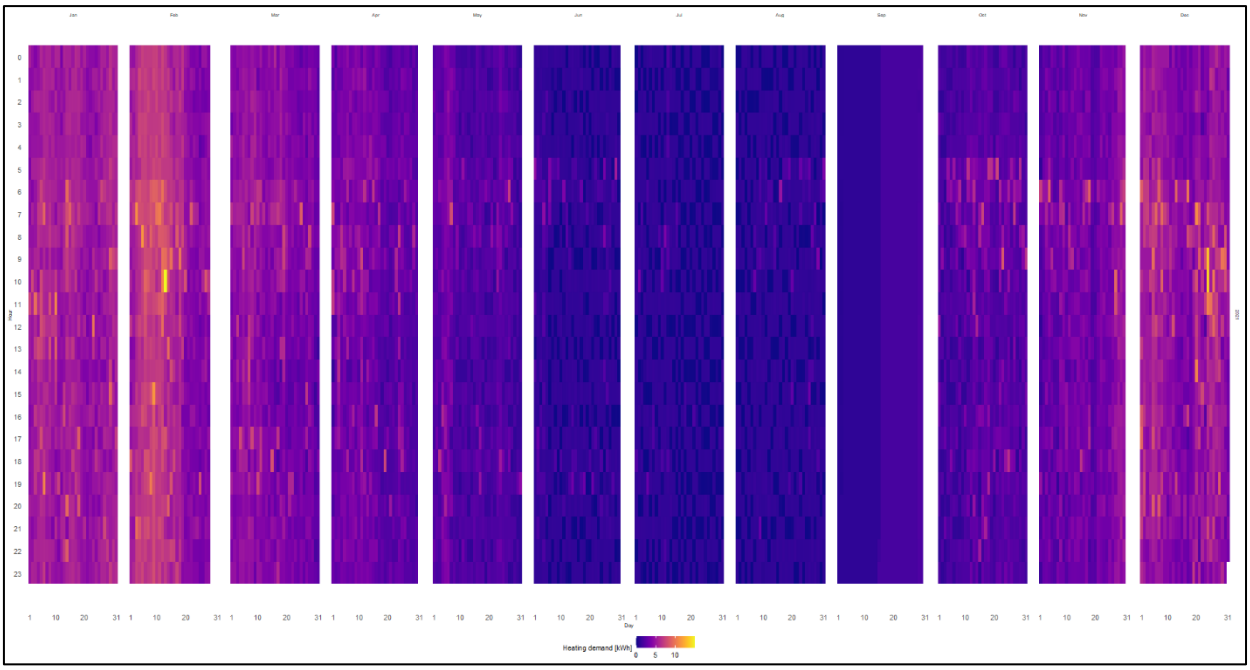


Figure 13: Example of an hourly energy usage heatmap.

Utilizing the heatmap makes it possible to identify energy consumption patterns and areas where energy demand can be reduced. An example of the usefulness of this method is the overview of the daily peaks per month, the identification of night setback control, as well as other types of controls.

Overflow

The overflow (V_{over}) indicator, mentioned in Gadd and Werner (2014), is calculated by subtracting the ideal volume (V_{ideal}) from the measured volume by the SHM ($V_{measured}$). The ideal volume is calculated based on an ideal temperature difference (ΔT_{ideal}) of 45 ºC while considering the same energy demand as the one measured by the SHM ($E_{measured}$), and it is represented in m^3. The overflow formula is seen below.

$$V_{over} = V_{measured} - V_{ideal} = V_{measured} - \frac{E_{measured}}{\rho c_p \Delta T_{ideal}} \ [m^3] \quad (2)$$

In other words, this parameter represents how close or far the water volume consumption is from its ideal consumption. The overflow, as an indicator, does not require large data samples to be calculated, making it a good performance indicator for a quick and short-term analysis. The literature regarding the overflow also shows that the indicator is higher during the heating season for faults involving SH systems. While similarly, it is observed that when the fault is in the DHW system, the overflow is higher during the no-heating season. In Figure 14, is represented an example of the overflow calculated through a daily time-series plot.

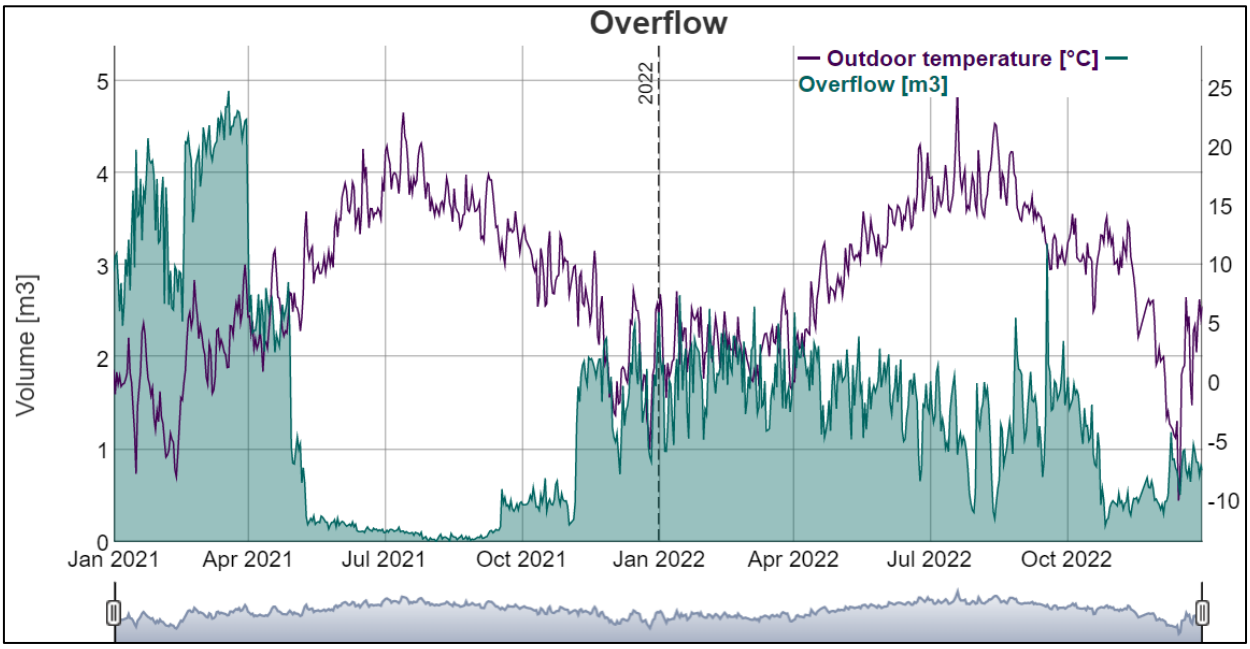


Figure 14: Example of the overflow.

Volume-temperature ratio

The volume-temperature ratio is a novel indicator developed at Aalborg University to identify faulty heating installations. It is calculated by dividing the water volume per heated floor area by the temperature difference (ΔT) as a function of the measured outdoor temperature. This indicator is calculated through equation 3:

$$VT_{ratio} = f(T_{out}) = \frac{V_{measured}}{A_{heated} \cdot \Delta T_{measured}} \ [m^3/m^2 °C] \quad (3)$$

A well-performing heating installation will display a linear change in the ratio throughout the variation of outdoor temperature conditions. Any data points that do not follow this linear profile are marked as faulty. This indicator is useful for identifying and troubleshooting problems in installations, allowing for more efficient maintenance and repairs. In Figure 15, is presented an example of the volume-temperature ratio generated through a scatterplot.

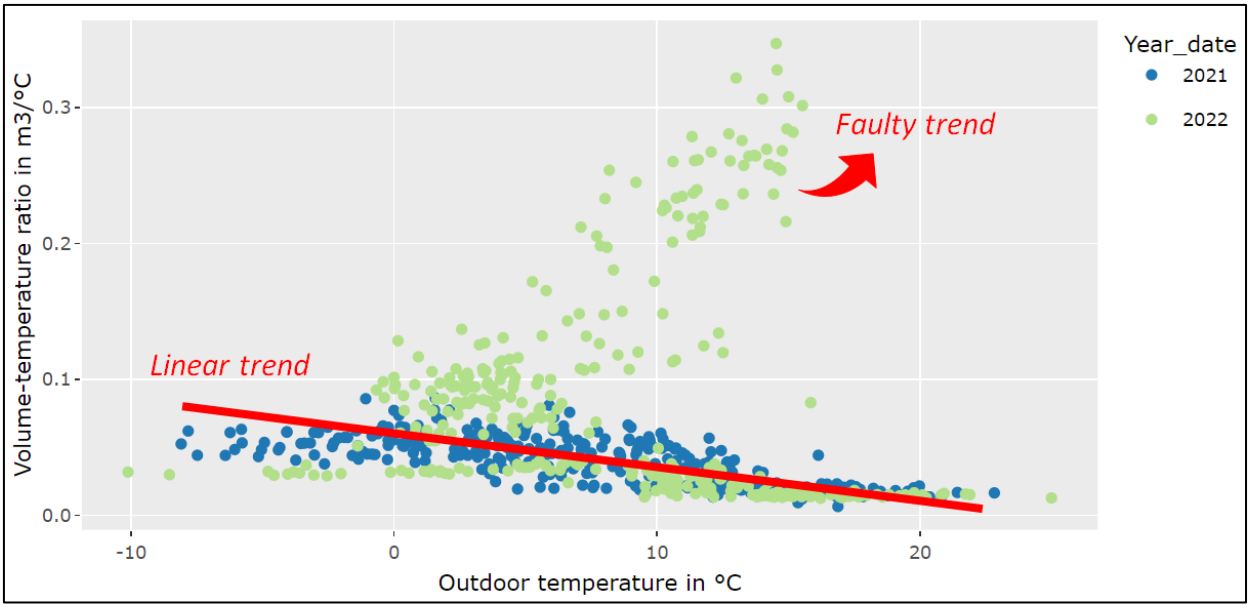


Figure 15*: Example of the volume-temperature ratio.*

2.5 IMPLEMENTATION OF THE WEB-APP

This web-app is developed with the Python language as a "Dash-app". "Dash" is a web application framework (Python library) designed to create interactive data visualizations, dashboards, and graphical user interfaces. It is built on top of the Flask web framework and Plotly.js, a JavaScript graphing library. Dash is a popular framework for developing data visualization web dashboards and web-tools among data scientists.

The web-app described in this document follows the general coding architecture recommended by the Dash framework. It mainly consists of generating a multi-page web-app with a navigation sidebar to change pages and rendering HTML and CCS content on the different pages of the app. The content of the pages includes texts, buttons, drop-down menus, tables, input components, pop-up windows, and interactive figures. The update of the page content as a function of the user interaction is managed by a number of callback functions. The necessary data is temporarily stored locally in the cache memory of the user's web browser via *dcc.Store* elements.

The general program/code structure of the Dash web-app is presented in *Figure 16* below.

The web-app is deployed on an Apache server of Aalborg University (Denmark).

Code structure / architecture:

```
dash_app.py
├── Documentation
├── Begin Process
│   ├── Import libraries
│   ├── Common Classes
│   ├── Common Functions
│   └── Pre-Defined Internal App Parameters
├── Initialization
│   ├── Security
│   ├── Import Additional App Parameters
│   ├── Import Internal App Data
│   └── Generate Main Front End App Structure
│       ├── Create Dash App
│       ├── Create Side Bar (nav)
│       ├── Create Main Page
│       └── Launch Dash App
├── Dash App Callbacks
│   ├── Callbacks Side Bar (nav)
│   │   ├── Side Bar (nav) - Main
│   │   ├── Render Content Home Page - SubFunction
│   │   ├── Render Content Import Data - SubFunction
│   │   ├── Render Content Compute Analysis - SubFunction
│   │   ├── Render Content Visualize Results - SubFunction
│   │   ├── Render Content Data Mapping - SubFunction
│   │   └── Render Content Download Results - SubFunction
│   ├── Callbacks Home Page
│   │   └── Home Page - Main
│   ├── Callbacks Import Data
│   │   ├── Import Data - Main
│   │   └── SubFunction...
│   ├── Callbacks Compute Analysis
│   │   ├── Compute Analysis - Main
│   │   └── SubFunction...
│   ├── Callbacks Visualize Results
│   │   ├── Visualize Results - Main
│   │   └── SubFunction...
│   ├── Callbacks Data Mapping
│   │   ├── Data Mapping - Main
│   │   └── SubFunction...
│   └── Callbacks Download Results
│       ├── Download Results - Main
│       └── SubFunction...
└── Run Main: Local Server Mode
```

Project structure:

```
aau_prelude_app
├── .idea
├── assets
├── config_files
├── lib
├── test data
├── venv
└── dash_app.py
```

Web-app front-end structure:

```
app_root_url
├── /   (Home page)
├── /page_import_data
├── /page_compute_analysis
├── /page_visualize_results
├── /page_data_mapping
└── /page_download_results
```

Figure 16: Program/code structure of the District Heating Meter Data Analysis Web-App.

2.6 OVERVIEW AND USER GUIDE OF THE WEB-APP INTERFACE

The District Heating Meter Data Analysis Web-App is intended to provide a simple yet efficient tool with a clear and straightforward workflow (see Figure 17).

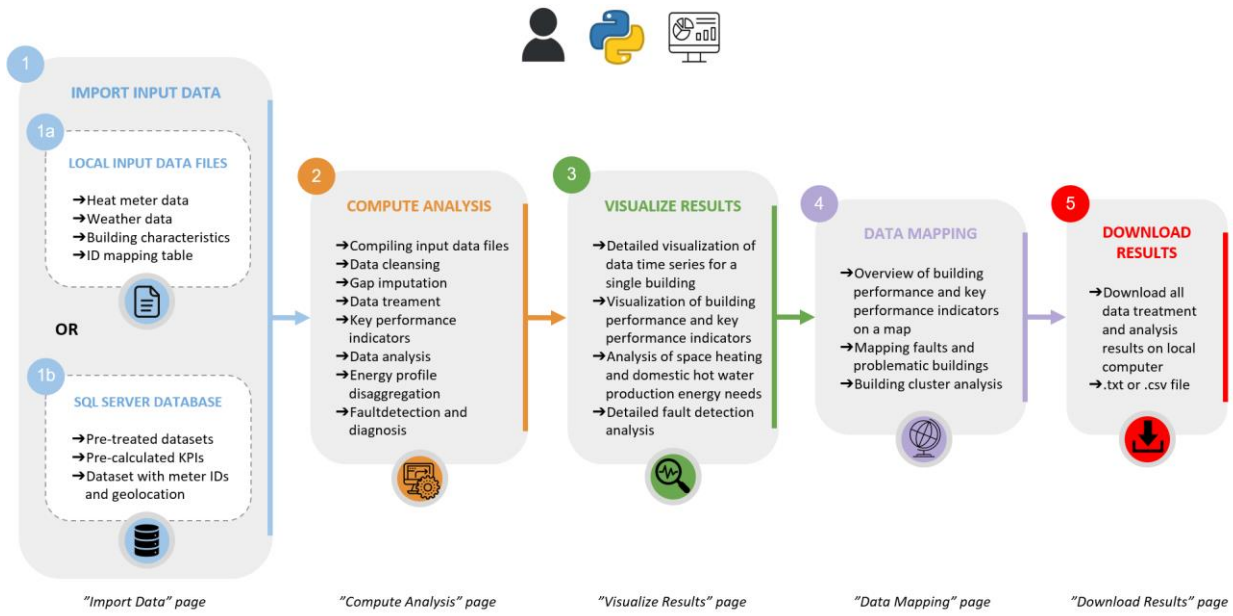


Figure 17: Workflow of the District Heating Meter Data Analysis Web-App.

The web-app is hosted on an Aalborg University server. It is accessible with a standard web browser. An "offline" version of this web-app is also available as a stand-alone computer desktop application without any data exchange with the Aalborg University server. In that case, it can be used on a secured computer to handle sensitive data.

The web-app is a multi-page application, with each page dedicated to a specific part of the workflow (see Figure 17). To perform data analysis, the user must follow the steps corresponding to each page: from the first page on the top of the sidebar to the last page of the sidebar. The first page (see Figure 18) gives an overview of the web-tool intent and capabilities. It also provides direct links to guidelines, all the scientific publications describing the different algorithms implemented in the tool, the FusiX platform and the webpage of the PRELUDE project.

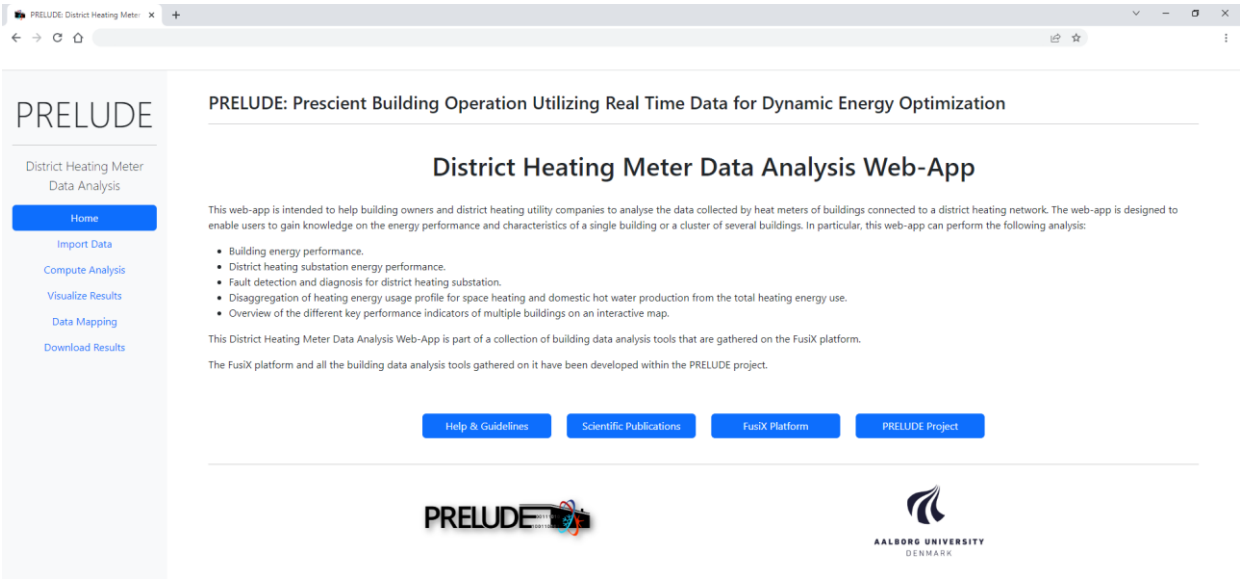
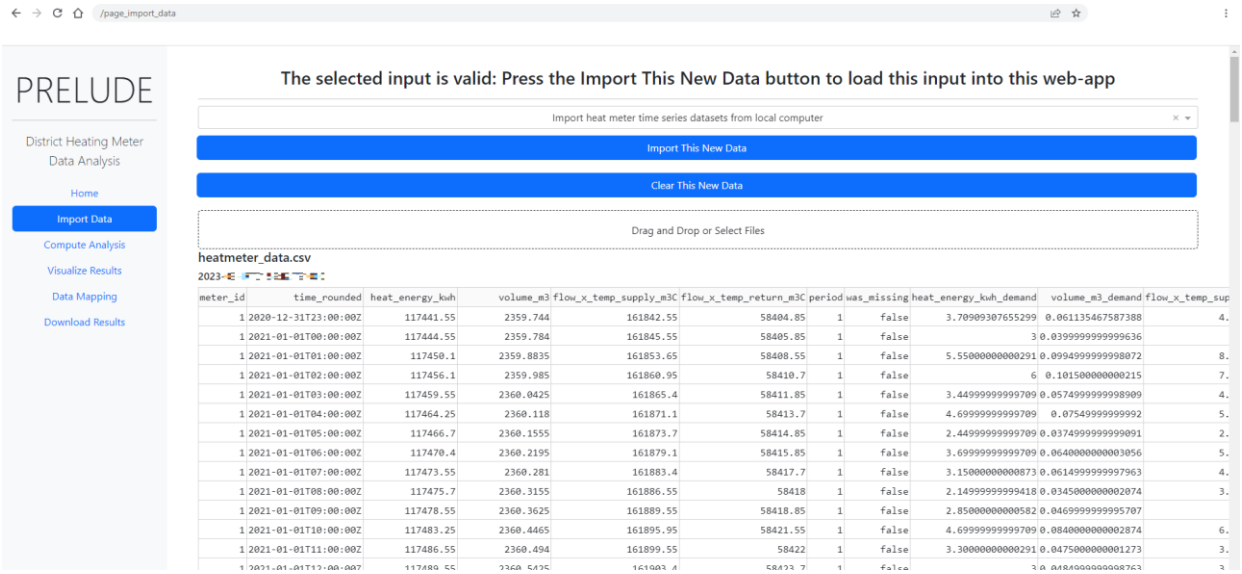


*Figure **18**: Home page of the District Heating Meter Data Analysis Web-App.*

The user starts the procedure with the "Import Data" page (see Figure 19). On this page, the user can select different types of data import methods: import input data files (heat meter data file, weather file, building characteristics file) located on a local computer (.txt or .csv files located on the local computer that the user employs to navigate on the web-app) or import all pre-computed data files and results from a secured-access SQL database on a server.

For the import of input files from a local computer, the user can select several files at a time. One should then press "Import The New Data" to validate the input and load it on the tool for analysis. The web-tool displays the raw data contained in the current input files. The current input data can be deleted by pressing the "Clear The New Data" button.



meter_id	time_rounded	heat_energy_kwh	volume_m3	flow_x_temp_supply_m3C	flow_x_temp_return_m3C	period	was_missing	heat_energy_kwh_demand	volume_m3_demand	flow_x_temp_sup
1	2020-12-31T23:00:00Z	117441.55	2359.744	161842.55	58404.85	1	false	3.70909307655299	0.061135467587388	4.
1	2021-01-01T00:00:00Z	117444.55	2359.784	161845.55	58405.85	1	false	3	0.0399999999999636	
1	2021-01-01T01:00:00Z	117450.1	2359.8835	161853.65	58408.55	1	false	5.55000000000291	0.0994999999998072	8.
1	2021-01-01T02:00:00Z	117456.1	2359.985	161860.95	58410.7	1	false	6	0.101500000000215	7.
1	2021-01-01T03:00:00Z	117459.55	2360.0425	161865.4	58411.85	1	false	3.44999999999709	0.0574999999998909	4.
1	2021-01-01T04:00:00Z	117464.25	2360.118	161871.1	58413.7	1	false	4.69999999999709	0.07549999999992	5.
1	2021-01-01T05:00:00Z	117466.7	2360.1555	161873.7	58414.85	1	false	2.44999999999709	0.0374999999999091	2.
1	2021-01-01T06:00:00Z	117470.4	2360.2195	161879.1	58415.85	1	false	3.69999999999709	0.0640000000003056	5.
1	2021-01-01T07:00:00Z	117473.55	2360.281	161883.4	58417.7	1	false	3.15000000000873	0.0614999999997963	4.
1	2021-01-01T08:00:00Z	117475.7	2360.3155	161886.55	58418	1	false	2.14999999999418	0.0345000000002074	3.
1	2021-01-01T09:00:00Z	117478.55	2360.3625	161889.55	58418.85	1	false	2.85000000000582	0.0469999999995707	
1	2021-01-01T10:00:00Z	117483.25	2360.4465	161895.95	58421.55	1	false	4.69999999999709	0.0840000000002874	6.
1	2021-01-01T11:00:00Z	117486.55	2360.494	161899.55	58422	1	false	3.30000000000291	0.0475000000001273	3.

Figure 19: Import Data page of the District Heating Meter Data Analysis Web-App.

Once all the input data files have been imported into the tool, the user moves to the next page "Compute Analysis". On that page, the user should press the "Compute" button to perform all data treatment and analysis (presented in previous sections) from the imported input data (see Figure 20).

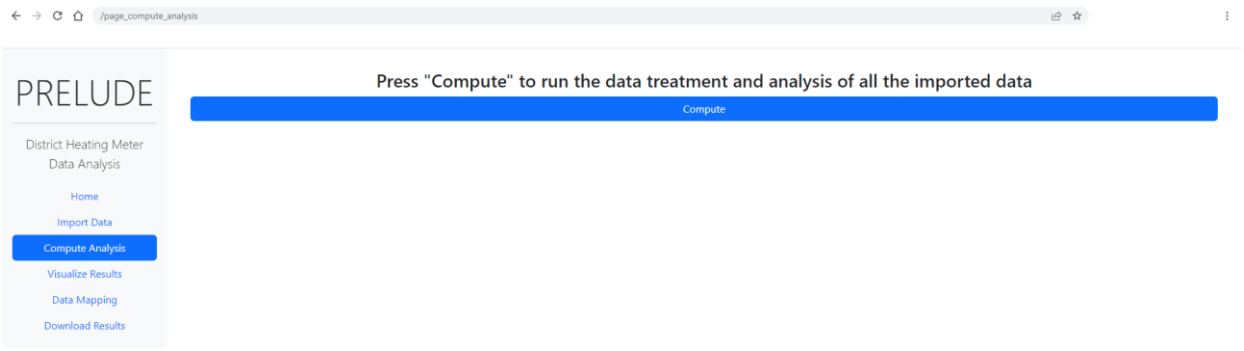


Figure 20: Compute Analysis page to compute all data treatment, analysis and key performance indicators from the imported input data.

Once the data treatment and analysis are completed, the user can move to the next page to visualize the results of the computation. The "Results Visualization" page provides a detailed analysis of individual buildings (one meter ID at a time) with different data time series of raw data, key performance indicators and other variables of interest generated by the data treatment and analysis computation (see example in Figure 21).

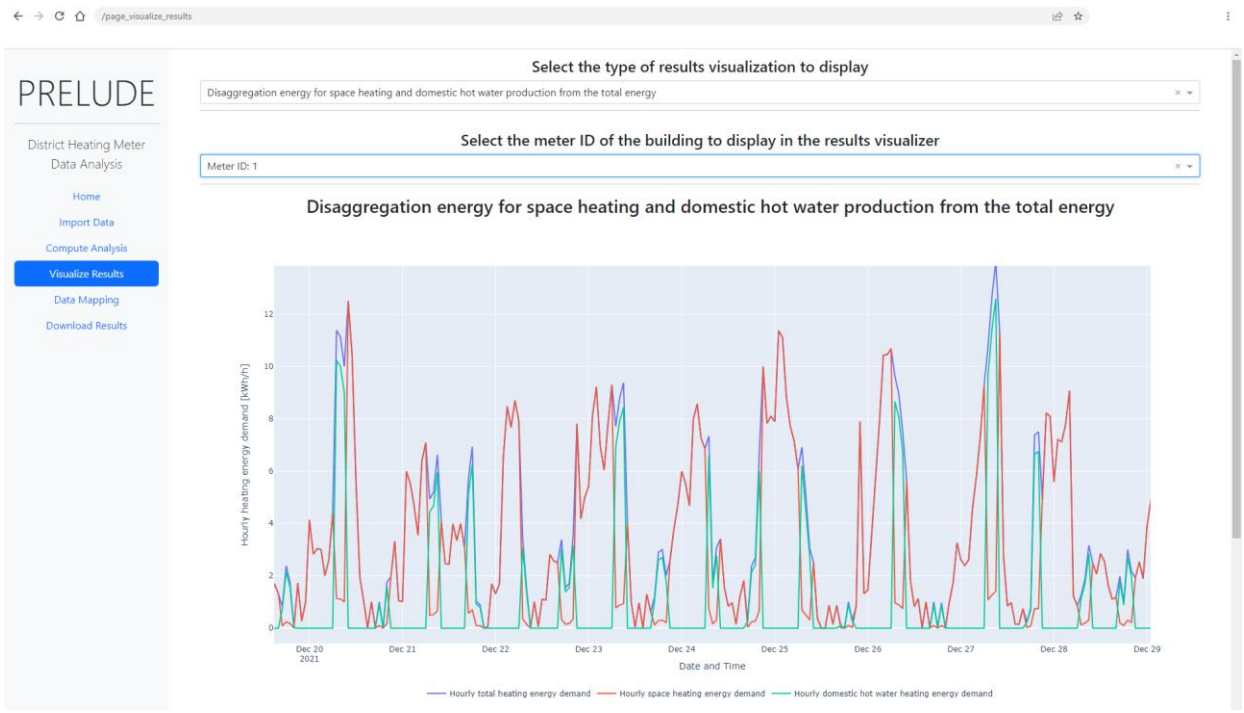


Figure 21: Example of Results Visualization: Disaggregation energy for space heating and domestic hot water production from the total energy for building with meter ID: 1.

In addition to detailed analysis of the data of a single building (meter ID) at a time, the user can get an overview of certain key performance indicators in the "Data Mapping" page (see Figure 22). One can thus clearly identify which buildings perform well or not in a cluster of buildings. Different filtering sliders allow the user to restrict the buildings visible on the map.

Figure 22: Example of Data Mapping: Overview temperature difference between supply and return fluid to the buildings of a town.

Finally, the user can download all computed data treatment and analysis results on a local computer in a .csv file by using the "Download Results" page.

3. CONCLUSIONS

The PRELUDE District Heating Meter Data Analysis web-app has been developed to assist building owners and district heating utility companies in analyzing the data collected by smart heat meters, evaluating the performance of buildings connected to the district heating network and detecting faulty systems.

Some of the web-app's features are based on validated methods developed within the PRELUDE project and published in peer-reviewed scientific articles. These methods include a data pre-treatment and data gap imputation method, a disaggregation algorithm for heating energy demand, an overview of key performance indicators for single buildings and building clusters, and fault detection in heating installations.

It is believed that this app will significantly improve the data analysis of DH utility companies and improve the overall reliability and energy efficiency of their energy system, together with a better insight in their customers, higher customer satisfaction, and new opportunities to provide energy advice services and fault detection and diagnosis.

4. FURTHER WORK

The implementation of new functionalities and features for this web-app continues to improve the analysis capabilities of this tool and improve interoperability with the FusiX platform. These new upgrades will be carried out within the PRELUDE task T5.4.

This web-app can be used as a stand-alone tool and download analysis results in files that have a format that is compatible with the input files of other applications of the FusiX platform. In addition, an API (application programming interface) will be implemented in this web-tool to enable direct use of the implemented data analysis algorithms by the other applications of the FusiX platform.

Based on the current work on the web-app tool, several potential future work tasks can be pursued to improve further the tool's capabilities and utility in WP 7. These tasks include the integration of an additional fault diagnostic algorithm. The current web-app tool relies on a combination of visual analysis and performance indicators to detect faulty buildings, which is the traditional approach in fault detection. However, a range of additional diagnostic algorithms could be integrated into the tool to provide more comprehensive results. The difference between the current state of the tool and future development is that the analysis is highly dependent on the user and that by doing so, it is only possible to detect faulty systems, with no means of knowing which component is causing the fault exactly. In contrast, a diagnosis algorithm might automatize this type of analysis in detecting faulty installations and predicting the reasons behind the underperformance of the heating systems.

This diagnosis algorithm will be based on machine learning classification techniques and on data gathered by technicians when they visit the installations due to corrective maintenance initiatives. Therefore, the models will be trained with installations when underperforming and well-performing. By doing so, the algorithm can, to some extent, detect faults automatically, i.e., with no user analysis in the first stage of the assessment, and predict the sources of the fault and propose a specific correction action plan.

Overall, the future work tasks outlined above represent a range of opportunities for improving the web-app tool for diagnosing faulty buildings before and after technician intervention. It is envisioned that the tool can be used in two different applications. The first is the current one, in allowing the operator to have a user-friendly overview of the SHM data. The second is the integration of an automatic algorithm to detect and diagnose existing faults in the heating installations without the involvement of the user in the assessment.

Therefore, by continuing to refine and optimize the tool, it has the potential to become a valuable resource for building managers and technicians, helping to identify faults and improve the overall performance of buildings.

5. BIBLIOGRAPHY

5.1 Online sources

Meyer, D. (2023) *Support Vector Machines: The Interface to libsvm in package e1071, R-project.org*. Available at: https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf (Accessed: March 16, 2023).

5.2 Publications

Bacher, P., de Saint-Aubain, P. A., Christiansen, L. E. and Madsen, H. (2016) "Non-parametric method for separating domestic hot water heating spikes and space heating," *Energy and Buildings*, 130, pp. 107–112.

Gadd, H. and Werner, S. (2014) "Achieving low return temperatures from district heating substations," *Applied Energy*, 136, pp. 59–67.

Broholt, T. H., Christensen, L. R. L. and Petersen, S. (2022) "Effect of measurement resolution on data-based models of thermodynamic behaviour of buildings," in *CLIMA 2022 Conference*. doi: 10.34641/CLIMA.2022.196.

Kristensen, M. H., Hedegaard, R. E. and Petersen, S. (2018) "Hierarchical calibration of archetypes for urban building energy modeling," *Energy and Buildings*, 175, pp. 219–234.

Leiria, D., Johra, H., Marszal-Pomianowska, A., Pomianowski, M. Z. and Heiselberg, P. K. (2021) "Using data from smart energy meters to gain knowledge about households connected to the district heating network: A Danish case," *Smart Energy*, 3(100035), p. 100035.

Leiria, D., Johra, H., Belias, E., Quaggiotto, D., Zarrella, A., Marszal-Pomianowska, A. and Pomianowski, M. Z. (2022) "Validation of a new method to estimate energy use for space heating and hot water production from low-resolution heat meter data," *E3S Web of Conferences*, 362, p. 10001.

Leiria, D., Johra, H., Marszal-Pomianowska, A. and Pomianowski, M. Z. (2023) "A methodology to estimate space heating and domestic hot water energy demand profile in residential buildings from low-resolution heat meter data," *Energy*, 263(125705), p. 125705.

Moritz, S. and Bartz-Beielstein, T. (2017) "ImputeTS: Time series missing value imputation in R," *The R Journal*, 9(1), p. 207.

Schaffer, M., Tvedebrink, T. and Marszal-Pomianowska, A. (2022) "Three years of hourly data from 3021 smart heat meters installed in Danish residential buildings," *Scientific Data*, 9(1), p. 420.

Schaffer, M., Leiria, D., Vera-Valdés, J. E. and Marszal-Pomianowska, A. (2023) "Increasing the Accuracy of low-resolution commercial Smart Heat Meter Data and analysing its Error," in *2023 European Conference on Computing in Construction*.

APPENDIX A – Description and Python code for the database

This appendix gives a short description of the data collected from the Aalborg district heating network and shows the Python code used to upload all the data to the database on a secure server.

The datasets have 15 parameters (features) in total and can be seen in Table 3. The parameters are committed to a single table in a database.

Table 3: Collected datasets' features/parameters.

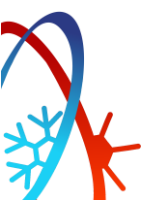
Parameter name	Description	Data type	Attributes
row_id	Row number	Integer	Primary Key
customer_id	Customer ID	Integer	–
meter_id	Meter ID	Integer	Index
read_time	Time when data is read	Datetime	–
heat_energy_kWh	Measurement of heat energy	Float	–
volume_m3	Measurement of volume	Float	–
time_counter_h	Hours elapsed	Float	–
flow_x_temp_supply_m3C	Calculated metric	Integer	–
flow_x_temp_return_m3C	Calculated metric	Integer	–
supply_temp_C	Supply temperature	Float	–
return_temp_C	Return temperature	Float	–
supply_flow_m3	Supply flow	Float	–
heat_effect_kw	Heat effect	Float	–
meter_type	Meter type	Text	–
rounded_read_time	Time when data is read (rounded)	Datetime	–

A.1 PYTHON CODE STRUCTURE

The Python code is divided into different functions and classes, each with a different purpose.

The principle of the code is that a function reads a file and parses the data, such that data is saved in a list. This data is committed to the database, which is performed in parallel. By doing this, data can be read and committed to the database significantly faster than sequentially.

Finally, the script uses an argument parser, such that it can be performed from the command terminal, and no usernames and passwords are, therefore, stored in the script. This makes the script safer, as people would need access to more than the script to gain access to the database.



A.2 PYTHON CODE

```
import sqlalchemy as db
from sqlalchemy.orm import declarative_base, Session
from multiprocessing import Process
import time
import glob
import csv
import concurrent.futures
import argparse
import sqlalchemy

Base = declarative_base

class MetersExt(Base):
    __tablename__ = "metering_data_ext"
    row_id = db.Column(db.Integer, primary_key=True)
    customer_id = db.Column(db.Integer)
    meter_id = db.Column(db.Integer, index=True)
    read_time = db.Column(db.DateTime)
    heat_energy_kwh = db.Column(db.Float)
    volume = db.Column(db.Float)
    time_counter_h = db.Column(db.Float)
    flow_x_temp_supply_m3C = db.Column(db.Integer)
    flow_x_temp_return_m3C = db.Column(db.Integer)
    supply_temp_C = db.Column(db.Float)
    return_temp_C = db.Column(db.Float)
    supply_flow_m3 = db.Column(db.Float)
    heat_effect_kw = db.Column(db.Float)
    meter_type = db.Column(db.Text)
    rounded_read_time = db.Column(db.DateTime)


def meter_data_ext(engine, data):
    with Session(engine) as session:
        counter = 0
        for item in range(len(data)):
            counter += 1
            query_data = MetersExt()
            query_data.customer_id = data[item]['customer_id']
            query_data.meter_id = data[item]['meter_id']
            query_data.read_time = data[item]['reading_time']
            query_data.heat_energy_kwh = data[item]['heat_energy_kwh']
            query_data.volume = data[item]['volume_m3']
            query_data.time_counter_h = data[item]['time_counter_h']
            query_data.flow_x_temp_supply_m3C =
data[item]['flow_x_temp_supply_m3C']
            query_data.flow_x_temp_return_m3C =
data[item]['flow_x_temp_return_m3C']
            query_data.supply_temp_C = data[item]['supply_temp_C']
            query_data.return_temp_C = data[item]['return_temp_C']
            query_data.supply_flow_m3 = data[item]['supply_flow_m3']
            query_data.heat_effect_kw = data[item]['heat_effect_kw']
            query_data.meter_type = data[item]['meter_type']
            query_data.rounded_read_time = data[item]['time_rounded']
            session.add(query_data)
            if counter % 1000 == 0:
                session.commit()
```

```
        # Send the remaining data
        session.commit()


def process_single_file(process_nr, file_list, connection_string):
    print(f'Starting #{process_nr}')
    parallel_process(file_list, connection_string)
    print(f'Stopping #{process_nr}')


def parallel_process(file_name, connection_string):
    engine = sqlalchemy.create_engine(connection_string)
    connection = engine.connect()
    with open(file_name, newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile, delimiter='\t')
        data = []
        for row in reader:
            data.append(row)

    for item in range(len(data)):
        if data[item]['time_counter_h'] == '':
            data[item]['time_counter_h'] = str(float(data[item-
1]['time_counter_h'])+1)
        if data[item]['heat_effect_kw'] == '':
            data[item]['heat_effect_kw'] = float('nan')

        data[item]['customer_id'] = int(data[item]['customer_id'])
        data[item]['meter_id'] = int(data[item]['meter_id'])
        data[item]['heat_energy_kwh'] = float(data[item]['heat_energy_kwh'])
        data[item]['volume_m3'] = float(data[item]['volume_m3'])
        data[item]['time_counter_h'] = float(data[item]['time_counter_h'])
        data[item]['flow_x_temp_supply_m3C'] =
int(data[item]['flow_x_temp_supply_m3C'])
        data[item]['flow_x_temp_return_m3C'] =
int(data[item]['flow_x_temp_return_m3C'])
        data[item]['supply_temp_C'] = float(data[item]['supply_temp_C'])
        data[item]['return_temp_C'] = float(data[item]['return_temp_C'])
        data[item]['supply_flow_m3'] = float(data[item]['supply_flow_m3'])
        data[item]['heat_effect_kw'] = float(data[item]['heat_effect_kw'])

    # # Write meter data to server
    meter_data_ext(engine, data)


def file_name_list():
    return glob.glob('data_AF_*.txt')


def setup_db(connection_string):
    engine = sqlalchemy.create_engine(connection_string)
    Base.metadata.drop_all(engine)
    Base.metadata.create_all(engine)


def setup_args():
    parser = argparse.ArgumentParser(description=
'''
_____
_\n
```

```
                This program will read a file, obtain connection to
server and insert that data on the server to the database \n
_____________________________________________________________________
__
''',

formatter_class=argparse.RawTextHelpFormatter)


    parser.add_argument('user_name',
                        type=str,
                        help='The username for uploading to the database.')
    parser.add_argument('password',
                        type=str,
                        help='The password for uploading to the database.')
    parser.add_argument('ip',
                        type=str,
                        help='The ip to the server.')
    parser.add_argument('db_name',
                        type=str,
                        help='The database name on the server.')
    parser.add_argument('--drop',
                        '-d',
                        type=bool,
                        help='',
                        default=False)

    args = vars(parser.parse_args())
    return args


def main():
    args = setup_args()
    connection_string =
f'postgresql://{args["user_name"]}:{args["password"]}@{args["ip"]}/{args["db_
name"]}'
    setup_db(connection_string)

    file_lists = file_name_list()
    print(f'Handling {len(file_lists)} files')

    start_time = time.time()
    all_processes = list()
    for i, fl in enumerate(file_lists):
        p = Process(target=process_single_file, args=(i, fl,
connection_string))
        p.start()
        all_processes.append(p)
    for p in all_processes:
        p.join()
    execution_time = (time.time() - start_time)
    print(f'Time: {execution_time}')

if __name__ == '__main__':
    main()
```