This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957047.

H2020-LC-SC3-EE-2020-1/LC-SC3-B4E-6-2020

Big data for buildings

Building Information aGGregation, harmonization and analytics platform

Project Nº 957047

D5.2 – The BIGG Artificial intelligence toolbox for building data

Responsible:	Helexia (Antony Uhlmann)
Document Reference:	D5.2
Dissemination Level:	Public
Version:	2.0
Date:	30/05/2023

Contributors Table

DOCUMENT SECTION	AUTHOR(S)	CONTRIBUTOR(S)	REVIEWER(S)
Initialization of the document	Romain Hollanders (Energis), Antony Uhlmann (Helexia)		Nicolas PASTORELLY (CSTB), Guillaume PICINBONO (CSTB), Chris DEVELDER (UGent), Stéphane LEROY (Helexia)
Introduction	Romain Hollanders (Energis)	Antony Uhlmann (Helexia)	
Data Collection and Storage	Romain Hollanders (Energis), Riccardo Devivo (Energis)	Antony Uhlmann (Helexia)	
Detailed list of function blocks	Pierre Lehanneur (Helexia), Antony Uhlmann (Helexia), Romain Hollanders (Energis)	Riccardo Devivo (Energis), Gerard Mor (CIMNE), Sampath Mukherjee (Inetum), Niels Tiben (UGent)	
Existing tools and standards used	Riccardo Devivo (Energis), Niels Tiben (U-Gent), Sampath Mukherjee (Inetum)	Romain Hollanders (Energis), Antony Uhlmann (Helexia)	
Application A1	Gerard Mor (CIMNE)	Jordi Carbonell (CIMNE)	
Application A2	Gerard Mor (CIMNE)	Jordi Carbonell (CIMNE)	
Application A3	Romain Hollanders (Energis), Riccardo Devivo (Energis)	Antony Uhlmann (Helexia)	
Application A4	Riccardo Devivo (Energis), Romain Hollanders (Energis)	Antony Uhlmann (Helexia)	
Application A5	Sampath Mukherjee (Inetum)	David Bouret (Inetum)	
Application A6	Niels Tiben (UGent)	Chris Develder (UGent)	
Integration within the Bigg Data Model	Antony Uhlmann (Helexia)	Romain Hollanders (Energis)	
Conclusion	Antony Uhlmann (Helexia)	Romain Hollanders (Energis)	

Table of contents

I. INTRODUCTION ... 7

I.1. Scope and audience ... 7

I.2. Principles and objectives ... 7

I.3. Structure of the document ... 8

I.4. Evolution of the AITB from the Preliminary version ... 8

II. GENERIC CORE FUNCTIONALITIES ... 9

II.1. Data collection and storage ... 11

II.1.1. Data collection and data format ... 11

II.1.2. Data storage ... 11

II.2. Detailed list of function blocks ... 12

II.2.1. Data preparation ... 13

II.2.2. Data transformation ... 15

II.2.3. Performance metrics ... 20

II.2.4. Compliance with main library APIs ... 21

II.2.5. Reinforcement learning techniques ... 22

II.3. Existing tools and standards used ... 25

II.3.1. Python libraries ... 25

II.3.2. R libraries ... 26

II.3.3. GitHub ... 27

II.3.4. Lifecycle code management - ML Flow ... 27

II.3.5. Code conventions ... 28

II.3.6. Test and verification process ... 29

III. BUSINESS DRIVEN PIPELINES ... 30

III.1. Application A1: Energy benchmarking of buildings ... 30

III.1.1. Business application ... 30

III.1.2. Technical solution as a Pipeline ... 30

III.2. Application A2: EEM Results Benchmarking ... 33

III.2.1. Business application ... 33

III.2.2. Technical solution as a Pipeline ... 33

III.3. Application A3: Baseline identification for Energy Performance Contracts ... 35

III.3.1. Business application 35

III.3.2. Technical solution as a Pipeline 35

III.4. Application A4: Occupancy pattern detection for Comfort and energy optimisation 37

III.4.1. Business application 37

III.4.2. Technical solution as a Pipeline 38

III.5. Application A5: Energy consumption forecasting 39

III.5.1. Business application 39

III.5.2. Technical solution as a Pipeline 40

III.6. Application A6: Gas demand response 43

III.6.1. Business application 43

III.6.2. Technical solution as a Pipeline 43

IV. INTEGRATION WITHIN THE BIGG DATA MODEL 46

IV.1. Definition of the BIGG Data Model 46

IV.1.1. BIGG Ontology 46

IV.1.2. BIGG Data Model 46

IV.1.3. Necessity of Harmonized Data 47

IV.2. Integration of Pipelines within the BIGG Data Model 47

IV.2.1. Data Harmonization layer 47

IV.2.2. Example of data harmonization with Application A1 48

CONCLUSION 50

REFERENCES 51

Table of Figures

Figure 1 - AITB development methodology 9
Figure 2 - Cross Industry Standard Process for Data Mining 10
Figure 3 – The BIGG toolbox within publicly available the GitHub. 10
Figure 4 – Example of pipeline 21
Figure 5 - The elements and relations in an MDP, where *k* is the current time-step number. 22
Figure 6 – Example of a function docstring 29
Figure 7 - Application A1 algorithm flowchart 32
Figure 8 - Application A2 algorithm flowchart 34
Figure 9 - Application A3 algorithm flowchart 37
Figure 10 - Application A4 algorithm flowchart 39
Figure 11 - Application A5 algorithm flowchart 40
Figure 12 - Example of the raw input data 41
Figure 13 - Shape of the (desired) final output (predicted power consumption) 42
Figure 14 - Data architecture diagram of the ultimate solution with the corresponding tools/resources ... 42
Figure 15 - Application A6 algorithm flowchart 45
Figure 16 - BIGG data model development and use 46
Figure 17 - Data Harmonization layer within the AITB 48

Table of Acronyms and Definitions

Acronym	Definition
AI	Artificial Intelligence
AITB	BIGG Artificial Intelligence Toolbox for Buildings
API	Application Programming Interface
BC	Business Case
BDHF	BIGG Harmonized Format
BMS	Buildings Management System
CVRMSE	Coefficient of Variation of the root mean square error, CV(RMSE). This basically assess how close you are to the individual data points (such as monthly utility bills).
DR	Demand Response
DSF	Demand Side Flexibility
EEM	Energy Efficiency Measure
EnPC	Energy Performance Contract
ESCO	Energy Service Company
Function	Function in this document refers to a needed operation, transformation, analytic task or modification to be performed on a given set of data.

Function Block	Function block is used in the document to describe a single set of code developed to perform a specific Function identified by the BIGG WP5 team as a singular element. A Function Block is defined by its inputs, its Function and the output it provides.
HVAC	Heating Ventilation and Air Conditioning
INSPIRE	The INSPIRE Directive, establishing an infrastructure for spatial information in Europe to support Community environmental policies, and policies or activities which may have an impact on the environment entered into force in May 2007. INSPIRE is based on the infrastructures for spatial information established and operated by the Member States of the European Union. The Directive addresses 34 spatial data themes needed for environmental applications. See https://inspire.ec.europa.eu/
NMBE	Normalized Mean Bias Error. This assesses whether you globally over or under-predict the consumption.
Pipeline	A pipeline is generally defined as a linear sequence of processes chained together to perform an instruction. In this document Pipeline refers to one or several Function Blocks assembled and packaged together to perform a larger task. It can be seen as a Function Block composed of other Function Blocks.
R^2	R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable.
RAF	Reference Architecture Framework
RL	Reinforcement Learning (RL) is a type of machine learning algorithm that is used to train agents Ato make decisions in an environment based on feedback in the form of rewards or punishments.
RES	Renewable Energy Source
UC	Use Case. In this document, the various use cases mentioned are taken from the D6.1 (Josep M. Granollers (ICAEN, 2021) and D6.2 (Oriol Escursell Jové (ICAEN, 2022) and detailed according to a chosen formalism.
UML	Unified modelling language (UML) diagrams describe the structure of a system, the objects within the system, and how they all behave.

I. INTRODUCTION

I.1. Scope and audience

The scope of this technical document is to provide an overview of the BIGG Artificial Intelligence Toolbox for Buildings (AITB) developed for building energy savings using Python and R programming languages. This toolbox makes use of standard libraries like scikit-learn[1] and caret,[2] which are adapted to the context of building data and used as a base to develop specific missing functionalities, e.g., extract relevant engineered features from the data.

The AI toolbox developed in this project is aimed at data scientists, developers, and technical business analysts who have an interest in using AI for building energy savings. The toolbox offers a range of tools for data pre-processing, feature engineering, modelling, and evaluation, which are specifically designed for building applications. With this toolbox, users can quickly and easily develop AI models to analyse building energy usage and optimize energy consumption.

The document is intended to serve as a reference guide for users who want to learn about the toolbox and its functionalities. It also provides instructions on how to use the toolbox and implement it in specific building applications.

I.2. Principles and objectives

The principles behind our AI Toolbox for Buildings are rooted in the need to develop practical solutions for building energy savings that are tailored to the unique characteristics of building data. Our approach is grounded in a problem-based methodology, where the applications are the starting point for the development of the toolbox. By working closely with actors from the energy sector, such as ESCOs and tech companies, we were able to identify specific problems and develop AI tools that were tailored to meet their needs.

The objectives of our AI toolbox are to provide users with a suite of powerful and flexible tools for building energy savings. Our toolbox is designed to make it easy for users to pre-process building data, study relevant features, develop, and evaluate models, and deliver AI solutions to specific energy-saving problems. The toolbox provides a variety of algorithms and techniques that are optimized for building data, including tools for dealing with time-series data, normalization techniques, and KPIs such as degree days.

Our toolbox is designed to be scalable, enabling users to work with large and complex datasets and easily adapt the AI pipelines to new applications. It has implemented reading and writing data functions for the direct usage of datasets harmonized to the BIGG Ontology[3] (CSTB, 2022), which provides the power of linked data to building information.

We have developed our toolbox as an open-source project, allowing for collaborative development, community contributions, and the ability to share best practices and knowledge. We have also prioritized

[1] https://scikit-learn.org/

[2] https://topepo.github.io/caret/

[3] https://github.com/biggproject/Ontology/tree/version1.0

ease of use, using GitHub as interface where users can find the code of every developed module, one example pipeline for each application for which a solution has been provided in the project as well as clear documentation to ensure they can quickly and easily begin using the toolbox.

Overall, the principles and objectives of our AI toolbox reflect our commitment to developing practical solutions for building energy savings that are grounded in real-world problems and are tailored to the unique characteristics of building data. We believe that our toolbox will enable users to make significant contributions to reducing energy consumption and creating more sustainable buildings.

I.3. Structure of the document

This document is structured into three main sections that cover the Generic Core Functionalities, Business-Driven Pipelines, and Integration with the BIGG Data Model of our AI toolbox for building energy savings.

The first section, "Generic Core Functionalities", provides an overview of the basic functionalities of our toolbox. This includes a detailed description of the data collection process, the function blocks that make up our toolbox, and the existing tools and standards that we have used. This section is intended to provide a comprehensive understanding of the underlying architecture of our toolbox, as well as the key features and functionalities that it offers.

The second section, "Business Driven Pipelines", provides detailed information on the specific AI pipelines that we have developed for each of the applications of our project. This section includes a detailed description of the business problem, the data pre-processing steps, feature engineering, model development, and evaluation metrics for each pipeline. This section is intended to provide an in-depth understanding of how our toolbox can be used to solve specific problems related to building energy savings.

The third section, "Integration with the BIGG Data Model," describes how our toolbox integrates with the other work packages of the BIGG project, and particularly how it uses the BIGG data model to harmonize building data under a single universal format. This section is intended to provide an understanding of how our toolbox fits into the broader context of the BIGG project, and how it can be used in conjunction with other tools and systems to create more sustainable buildings.

I.4. Evolution of the AITB from the Preliminary version

The preliminary version of the AITB was intended to present the library of Function Blocks identified and developed separately to be used as part of the technical solution for the identified Business Cases, as described in Deliverable D6.1 (Josep M. Granollers (ICAEN, 2021) and evaluated in Deliverable D6.2 (Oriol Escursell Jové (ICAEN, 2022). Deliverable D5.1 (Helexia, 2022) described the way function were defined and how it should be adapted to the Business Cases identified.

From the preliminary AI toolbox created, as seen in Figure 1 below, Function Blocks were assembled and packaged together into Pipelines (Task 5.4). The final version of the AITB is intended to enable the use of Pipelines without additional development. To get a viable final product, it is necessary to give the final user the possibility to use both granular Function Blocks individually, possibly combining them with other existing open-source libraries, and pre-established pipelines developed for the specific use cases of the BIGG project.

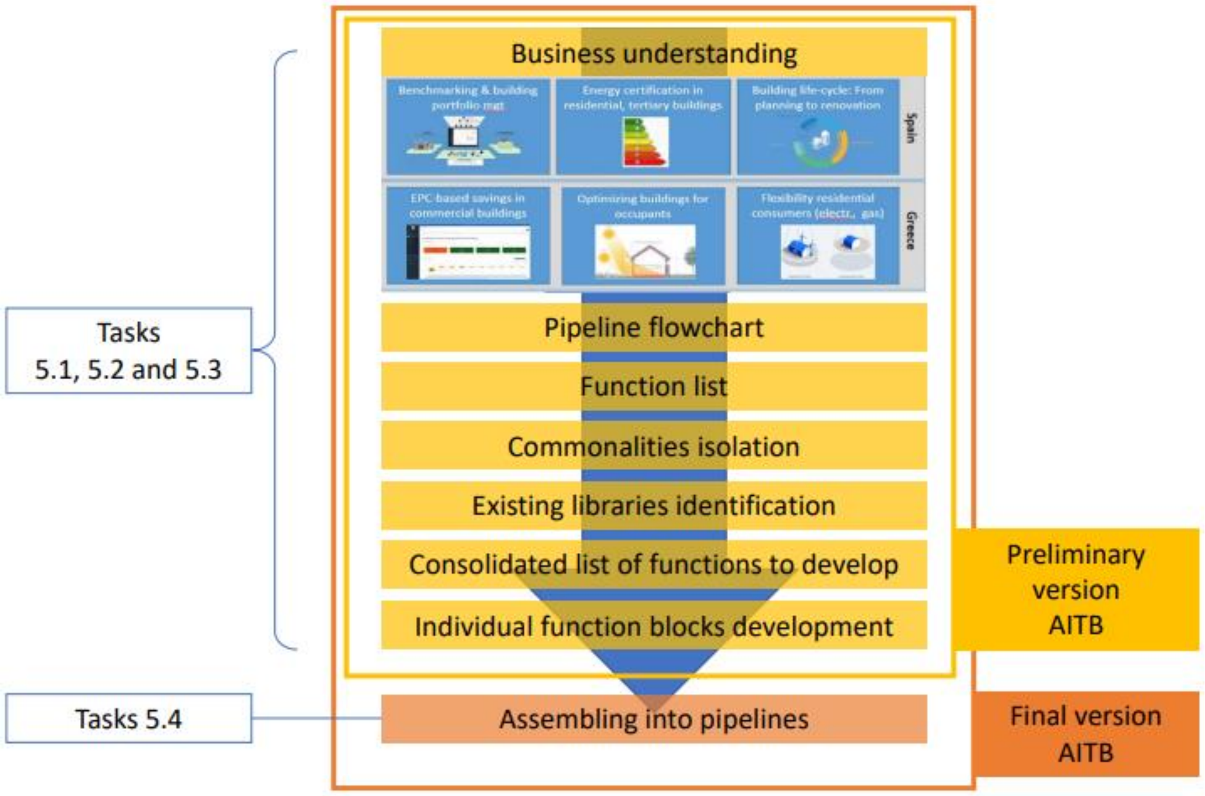

Business understanding
Benchmarking & building portfolio mgt
Energy certification in residential, tertiary buildings
Building life-cycle: From planning to renovation
Spain
EPC based savings in commercial buildings
Optimizing buildings for occupants
Flexibility residential consumers (electr., gas)
Greece
Pipeline flowchart
Function list
Commonalities isolation
Existing libraries identification
Consolidated list of functions to develop
Individual function blocks development
Assembling into pipelines
Tasks 5.1, 5.2 and 5.3
Tasks 5.4
Preliminary version AITB
Final version AITB

Figure 1 - AITB development methodology

II. Generic core functionalities

In this section, we provide an overview of the generic core functionalities of our AI toolbox for building energy savings. This includes the key building blocks that make up our toolbox, the data collection process, and the existing tools and standards that we have used.

Our toolbox is designed to provide a comprehensive set of tools for analysing building data and developing AI models to optimize building energy savings. To achieve this, we have leveraged the power of the Python and R programming languages and have adapted existing libraries such as scikit-learn and Caret to the specific context of building energy savings.

One of the key challenges that we faced in developing our toolbox was the unique nature of building data. Building data is not like any other data and requires specific pre-processing steps and normalization techniques in order to be useful for AI applications. To address this, we have developed a set of generic core functionalities that can be used to pre-process, organize, and normalize building data, regardless of the specific application, as described in the Figure 2.

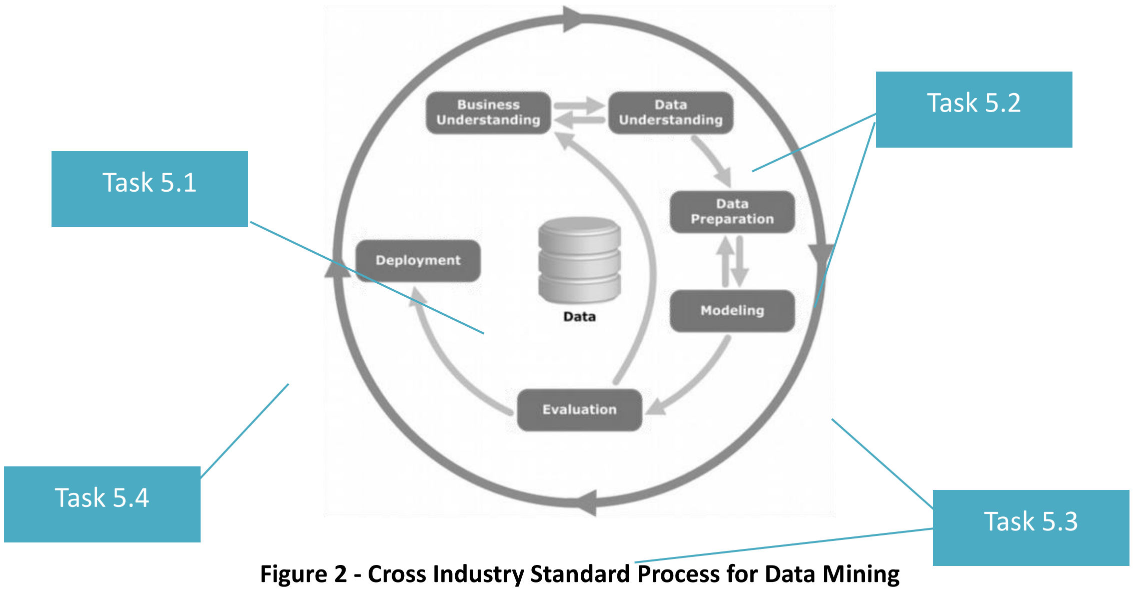
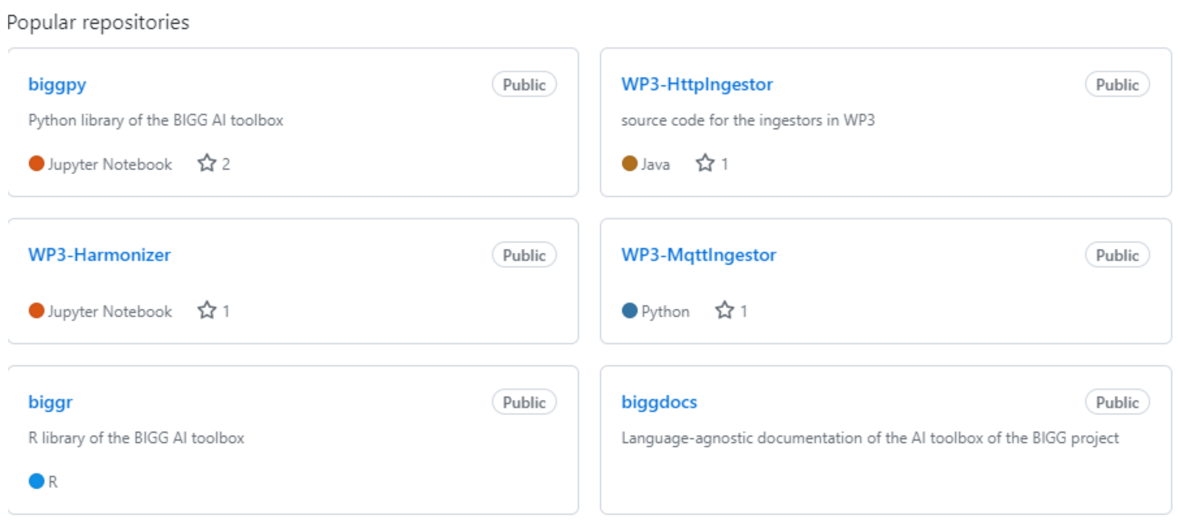


Figure 2 - Cross Industry Standard Process for Data Mining

In this section, we describe the key building blocks that make up our toolbox in detail, including data collection, pre-processing, feature engineering, model development, and evaluation. We also describe the existing tools and standards that we have leveraged, such as the Python and R libraries as well as code and testing conventions.

Overall, our toolbox provides a powerful set of tools for data scientists, developers, and technical business analysts who are interested in building energy savings. By providing a comprehensive overview of our generic core functionalities, we hope to provide a clear understanding of the underlying architecture and key features of our toolbox. Figure 3 shows the architecture of the Toolbox within the dedicated GitHub.


Popular repositories

biggpy Public
Python library of the BIGG AI toolbox
● Jupyter Notebook ☆ 2

WP3-HttpIngestor Public
source code for the ingestors in WP3
● Java ☆ 1

WP3-Harmonizer Public
● Jupyter Notebook ☆ 1

WP3-MqttIngestor Public
● Python ☆ 1

biggr Public
R library of the BIGG AI toolbox
● R

biggdocs Public
Language-agnostic documentation of the AI toolbox of the BIGG project

Figure 3 – The BIGG toolbox within publicly available the GitHub.

II.1. Data collection and storage

The success of any AI toolbox depends on the quality and accessibility of the data provided to it. In this section, we describe the data formats that our toolbox can handle, including data that has been harmonized under the BIGG data model. We also discuss the importance of data storage and the need for users to have their own resources to host their data and code.[4] By understanding the requirements for data collection, format, and storage, users can effectively leverage the power of our toolbox to generate meaningful insights and solutions to their building energy efficiency problems.

II.1.1. Data collection and data format

The AI toolbox is designed to be flexible with the data format it receives, while also being able to work with harmonized data under the BIGG data model. The toolbox does not perform data collection itself, but it can handle multiple data formats, including instantaneous measurements, pulses, and index values for consumption data, which are common in building energy data.

While the harmonized BIGG data model is the preferred data format for the toolbox, each function block can be used independently from the BIGG data model. This means that the toolbox can accommodate a wide range of data formats and structures, allowing it to work with different types of data sources.

It is worth noting that each pipeline in the toolbox is designed to work with harmonized input and provide output in a harmonized format. However, the building blocks of the pipelines themselves do not require harmonized input and output. Overall, the toolbox's flexibility with data formats makes it a powerful tool for working with building data, regardless of the specific format it comes in.

II.1.2. Data storage

As an AI toolbox, our software is designed to provide powerful functionality for analysing and modelling data, but it does not include any data storage capabilities. Users are expected to provide their own storage solutions for their data and code, whether it be on-premises or in the cloud.

This approach provides users with the flexibility to choose the storage solutions that best fit their specific needs and requirements. By decoupling the AI toolbox from storage, users can use existing infrastructure. Additionally, users have control over their data and can ensure it is stored and managed in compliance with relevant regulations and standards.

While our AI toolbox does not include data storage capabilities, it is designed to integrate seamlessly with a wide range of storage solutions. This includes on-premises storage, cloud storage, and distributed storage systems. By providing flexible input and output options, our toolbox enables users to easily connect with their existing data storage solutions and seamlessly integrate their data and code with our AI functionality.

[4] Data storage capabilities were defined in the context of the task T5.1 – Provision of data storage infrastructure as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. (INETUM REALDOLMEN BELGIUM)

II.2. Detailed list of function blocks

In this section, we provide a comprehensive list of the function blocks that make up our AI toolbox. These blocks were developed to address the specific needs of the various pipelines and applications within the project, and they can be used independently or in combination to create custom solutions. We include details on which applications and pipelines each function block is used in, as well as any relevant information on its functionality.

The development of these function blocks was done collaboratively on GitHub, and all the related documentation can be accessed directly on the BiggDocs repository: github.com/biggproject/biggdocs.

The purpose of this repository is to provide a generic language-agnostic documentation, even though all the functions are well documented using docstrings and adopting well known coding style standards.

Function blocks are defined in a generic way, allowing multiple building related usages. They are specified by their (i) inputs, (ii) function, and (iii) outputs. They are organized in different modules and module blocks. A module is a group of function blocks which all refer to a given aspect of data management or analytics and module blocks are dividing modules in 4 main categories:

1. Data preparation modules,

   Function blocks linked to the early phase of data management such as data quality checking, outlier detection, time stamp management, etc.

2. Data transformation modules,

   Function blocks involving data classification and secondary dataset management such as calendar elements, weather data elements, etc.

3. Modelling modules,

   Function blocks involving data model generation, assessment and testing.

4. Reinforcement learning modules.

   Function blocks related to the creation and the training of a reinforcement learning agent. Reinforcement learning is a specific type of machine learning.

In BiggDocs the language-agnostic documentation is presented to provide readers with an idea of the functionalities provided by the BIGG AI Toolbox. Nevertheless, in case of usage of biggr[5] or biggpy,[6] or testing any of the pipelines presented in this deliverable, we strongly recommend to follow the specific documentation contained in each library and each pipeline.

[5] biggproject/biggr: R library of the BIGG AI toolbox (github.com)

[6] biggproject/biggpy: Python library of the BIGG AI toolbox (github.com)

II.2.1. Data preparation[7]

This section of encompasses various subsections that outline the essential functions and processes involved in preparing data for analysis and modelling within the AITB framework. This section focuses on ensuring the data is properly formatted, cleaned, transformed, and harmonized to enable effective AI-driven analysis and decision-making in the building sector. The following subsections provide a detailed overview of the key functions, their respective descriptions, and references to the specific applications (A1 to A5) within the AITB pipelines or use cases where they are utilized. Details of the applications can be found in section III.Business driven pipelines.

II.2.1.a. Time Stamps Alignment

II.2.1.a.1. detect_time_step

The function infers, i.e., automatically deduces from the input data, the minimum time step (frequency) that can be used for the input time series, represented with a string alias formatted according to the ISO 8601.

Used in: A1, A2, A3, A4

Available in: biggr, biggpy

II.2.1.a.2. align_time_grid

The function aligns the frequency of the input time series with the output frequency given as an argument using the specified aggregation function.

Used in: A1, A2, A3, A4

Available in: biggr, biggpy

II.2.1.a.3. clean_ts_integrate

The function converts a cumulative (counter) or onChange (delta) measurement to instantaneous.

Used in: A1, A2

Available in: biggr, biggpy

II.2.1.b. Outlier detection

II.2.1.b.1. detect_ts_min_max_outliers

This function detects elements of a time series outside the allowed range in which you know the data should be. In the case of energy consumption, it should be a positive value and not exceeding the total capacity permitted. Sometimes, this value is not easy to define. Additionally, with the minSeries and maxSeries arguments, these ranges can be set differently along the period.

[7] The data transformation features were developed in the context of the task T5.2 – Data analytics tools design and identification of commonalities as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002.

Used in: A1, A2

Available in: biggr, biggpy

II.2.1.b.2. detect _ts_zscore_outliers

This function detects elements of the time series out of a Z-score threshold, applied on the whole time series or a rolling window of predefined width.

Used in: A1, A2, A3, A4

Available in: biggr, biggpy

II.2.1.b.3. detect_ts_calendar_model_outliers

This function detects elements of the time series out of a confidence threshold based on linear model of the calendar variables (month, weekday, hour). It estimates the outliers of a time series based on a quantile regression model that uses calendar features as input variables. These calendar features, that normally correspond to common seasonality, are transformed using Fourier components. However, there are two exceptions of model features that are not transformed using this technique: the intercept, which is a fixed term during all the period, and HOL (holidays) feature, which is a 0-1 depending on if the day is holiday or not.

Regarding mandatory features, the intercept is the only one that will be considered even if it is not specified in the calendarFeatures argument. Another interesting point of this argument is that it allows the interaction between terms. Thus, if we set a HOL intercept term, a different intercept will be estimated for holidays and non-holidays.

Used in: A1, A2

Available in: biggr

II.2.1.b.4. detect_static_min_max_outliers

This function detects which numerical elements are outside the min-max range. It should be used to filter outliers of static data (e.g., building areas, year of construction)

Used in: A1, A2

Available in: biggr

II.2.1.b.5. detect_static_reg_exp

This function detects which string element satisfy the regular expression. To test regular expressions configured in the regExpValues argument, youcan use the web application https://regexr.com/.

Used in: A1, A2

Available in: biggr

II.2.1.b.6. detect_disruptive_period

This function detects a disruptive period in a consumption time series. It evaluates different date ranges to find the most suitable one containing a disruptive period that should not be considered in the training phases of statistical models. An example of a disruptive period could be the Covid lockdown when building consumption radically changes due to the different occupancy and activity patterns. The model used to detect the disruptive period considers the consumption relationship between weekdays and temperature.

Used in: A1, A2

Available in: biggr

II.2.1.b.7. detect_holidays_in_tertiary_buildings

The function detects holiday periods in buildings with highly seasonal patterns. An analysis of turn points in density function is performed to identify "overrepresented" low consumption dates. Additional post-processing is done to take care of additional constraints related to holidays period.

Used in: A1, A2

Available in: biggr

II.2.1.c. Missing Data Management

II.2.1.c.1. fill_ts_na

The function imputes values to Not Available (NA) elements of a time series, based on the outlier estimation from functions implemented in the Outlier Detection module block of this library. It requires the previous usage of the Outlier Detection functions. An interpretation of the maxGap[8] and a resample of the fillMask[9] time step is done, considering the actual time step of the data time series. Actual methods to fill the NA elements are quite simple, but in future more complex implementation of this imputation could be integrated.

Available in: biggr, biggpy

II.2.2. Data transformation[10]

II.2.2.a. Profiling

II.2.2.a.1. clustering_dlc

The function clusters similar daily load curves based on the load curves themselves, calendar variables and outdoor temperature. Spectral clustering is used to infer the unknown daily load curve patterns. The minimum frequency allowed of the argument's consumption and temperature to cluster daily load curves is hourly.

Used in: A1, A2

[8] maxGap: Maximum allowable gap or missing values in a dataset.

[9] fillMask: Technique for filling missing values in a dataset.

[10] The data transformation features were developed in the context of the task T5.2 – Data analytics tools design and identification of commonalities as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002.

Available in: biggr

II.2.2.a.2. classification_dlc

The function classifies daily load curves based on the outputs of a clustering or a labelled dataset and a new set of data. The minimum frequency allowed of the argument's consumption and temperature to cluster daily load curves is hourly.

Used in: A1, A2

Available in: biggr

II.2.2.a.3. weekly_profile_detection

The function returns the weekly profile of the input time series.

Available in: biggpy

II.2.2.a.4. yearly_profile_detection

The function returns the yearly profile of the input time series.

Available in: biggpy

II.2.2.a.5. add_weekly_profile

The function derives the weekly profile of an input time series at hourly frequency or higher, repeating it for the entire time range and adds it to the input set of features. The repeated weekly profile of the energy consumptions can indeed be used to enhance the performance of some linear models.

Used in: A3 (transformer version)

Available in: biggpy

II.2.2.a.6. generate_extended_weekly_profile

The function derives the weekly profile of an input time series at hourly frequency, repeating it for the specified time range, which can be several years. It can be used to generate the weekly profile feature for several years in the prediction phase.

Used in: A3

Available in: biggpy

II.2.2.b. Holidays

II.2.2.b.1. add_holiday_component

The function adds to the input set of features the public holiday feature based on the specified country, province and state in the time range of the input time series.

Used in: A3, A4 (transformer version)

Available in: biggpy

II.2.2.c. Calendar

II.2.2.c.1. add_calendar_components

The function decomposes the time into many features (e.g., date, day of the year, day of the week, day of the weekend, working day, non-working day, season, month, hour, minute). The transformation must be done considering the local time zone. Typically, the features generated by this function are used as model inputs for modelling the user behaviour seasonality of energy consumption.

Used in: A1, A2, A3, A4, A5, A6

Available in: biggr, biggpy

II.2.2.c.2. trigonometric_encode_calendar_components

This function returns a transformer that encodes all the calendar components added to the input data into sin and cosine trigonometric cyclic components. This type of encoding greatly boosts the predictive capabilities of some models.

Used in: A3, A4

Available in: biggpy

II.2.2.d. Weather

II.2.2.d.1. degree_days

The function calculates the degree-days with the desired output frequency and considers cooling or heating mode.

Used in: A1, A2, A3, A4

Available in: biggr, biggpy

II.2.2.d.2. degree_raw

The function calculates the difference between outdoor temperature and a base temperature without considering the frequency of the original data.

Used in: A1, A2

Available in: biggr

II.2.2.d.3. get_change_point_temperature

The function finds the optimal change point temperature based on the correlation between energy consumption timeseries and outdoor temperature data.

Available in: biggr

II.2.2.e. Autoregressive processes

II.2.2.e.1. lag_components

The function shifts in time a set of features for model training and prediction. It is an essential step for the multi-step prediction of Autoregressive models, where the estimated output is directly used in the subsequent predictions.

Used in: A1, A2

Available in: biggr

II.2.2.e.2. lpf_ts

This function computes the first-order low pass filter for smoothing a time series. This function can be used in different cases: (1) Consumption time series, it helps remove artificial fluctuation; (2) Outdoor temperature time series, it helps to linearize the relation between consumption and outdoor temperature, as it simplifies the modelling of the thermal inertia of the building; (3) Wind speed time series, it helps to smooth the wind speed data; (4) Solar radiation time series, it helps to linearize the relation between consumption and solar radiation, as it simplifies the modelling of the solar gains of the building.

Used in: A1, A2

Available in: biggr

II.2.2.e.3. get_lpf_smoothing_time_scale

The function calculates the smoothing time scale parameter of the first-order low pass filter over an input variable, considering a specific time constant in hours.

Used in: A1, A2

Available in: biggr

II.2.2.f. Fourier Series

II.2.2.f.1. fs_components

The function obtains the components of the Fourier Series in sine-cosine form. It helps linearize the relationship of a seasonal input time series (e.g., solar azimuth, solar elevation, calendar features) to some output (e.g., energy consumption, indoor temperatures). Essentially, it decomposes a cyclic time series into a set of sine-cosine components used as inputs for modelling some output, where each of the components linearly depends on the output.

Used in: A1, A2

Available in: biggr

II.2.2.g. Modelling[11]

Model candidates

The modelling infrastructure used in biggr is the caret R-package, which is an already developed and widely extended library to model data in R using a large list of model candidates that are already pre-defined. Caret also contains functions for all the module blocks presented above: model validation, assessment, hyperparameters optimization, etc. However, in biggr, custom model candidates were developed to fulfil the modelling requirements of BC1 pipelines.

II.2.2.g.1. RLS

This function is a custom model wrapper for caret R-package to train and predict linear models fitted using the Recursive Least Square method. The model coefficients of this kind of model are time-varying; thus, the relation between inputs and output changes over time to better fit the data. Complete data transformation (including related biggr functions) and modelling pipelines can be executed using this wrapper.

Used in: A1, A2

Available in: biggr

II.2.2.g.2. GLM

This function is a custom model wrapper for caret R-package to train and predict Generalized Linear Models. Complete data transformation (including related biggr functions) and modelling pipelines can be executed using this wrapper.

Used in: A1, A2

Available in: biggr

II.2.2.h. Cross Validation

II.2.2.h.1. BlockingTimeSeriesSplit

This class is a splitter performing a special type of time series partitioning to be used in the cross-validation framework. Differently from TimeSeriesSplit, this method will generate disjoint partitions of the dataset in each iteration.

Used in: A3

Available in: biggpy

II.2.2.i. Model Assessment

II.2.2.i.1. evaluate_model_cv_with_tuning

This function performs a nested cross-validation (double cross-validation), which includes an internal hyper-parameter tuning, to reduce the bias when combining the two tasks of model selection and generalization

[11] The Modelling features were developed in the context of the task T5.3 – AI/ML techniques as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002.

error estimation. However, the purpose of this function is not to select the best model instance of a model family but instead to provide a less biased estimate of a tuned model's performance on the dataset.

Used in: A3, A4

Available in: biggpy

II.2.2.j. Model Identification

II.2.2.j.1. identify_best_model

This function implements a complete generalized pipeline for supervised learning to find the best model among different model families, each one associated with a specific parameter grid, given an input time series and a scoring function.

Used in: A3, A4, A5

Available in: biggpy

II.2.2.k. Model Persistence and Prediction

II.2.2.k.1. serialize_model

This function serializes and saves a model instance, with a given file format, to a specific path on the file system.

Used in: A3, A4

Available in: biggpy

II.2.2.k.2. deserialize_and_predict

This function deserializes a model, inferring the file format from the file name, applies the model on the X_data and returns the predicted values in the form of a time series.

Used in: A1, A2, A3, A4

Available in: biggpy

II.2.3. Performance metrics

This module contains additional performance metrics that are not implemented in the standard libraries but are needed to assess the performance of ML models in the BIGG use cases.

II.2.3.a.1. mean_bias_error

The Mean Bias Error (MBE) captures the bias of the model when predicting. According to the ASHRAE guidelines, a positive value implies a positive bias and an underestimation of the output variable, while a negative bias implies an overestimation. For example, if we are predicting the electricity consumption, a positive value of the MBE would mean that the model under-predicts the measured consumptions. However, the MBE suffers from cancellation errors, i.e., the sum of positive and negative values could reduce the value of MBE.

Used in: A3, A4

Available in: biggpy

II.2.3.a.2. cv_rmse

The Coefficient of Variation of the Root Mean Squared Error (CVRMSE) is a normalized version of the RMSE. It measures the variability of the errors between measured and simulated values. It does not suffer from cancellation errors because of the square operator but, on the other side, it punishes larger errors.

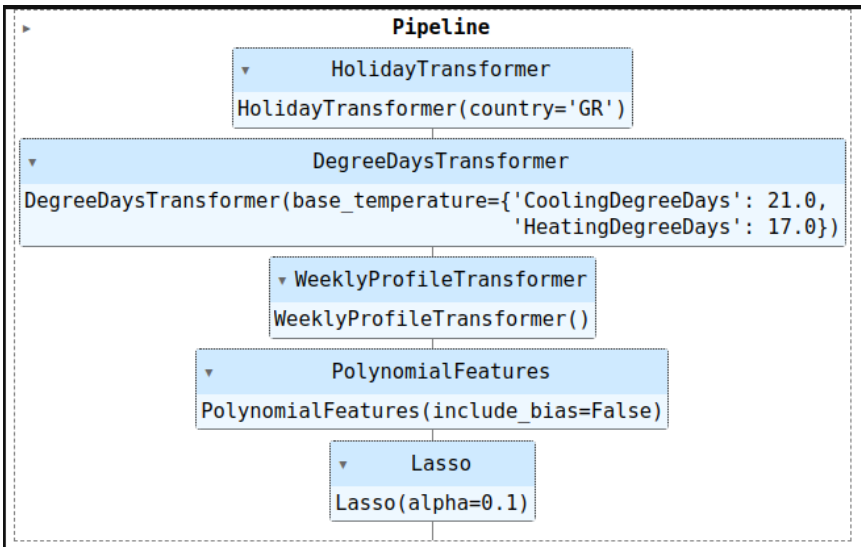
Used in: A3, A4

Available in: biggpy

II.2.4. Compliance with main library APIs

There may be differences in the implementation and the signature of the functions according to the specific programming language. In this regard, it took some effort to implement our components according to the API of the libraries that constitute the foundations of this toolbox, i.e., scikit-learn for Python and Caret for R. We firmly believe that this alignment to their programming interface represents an added value for our library, since it would extend the pre-existing functionalities with other core components specific of the energy domain.

For example, for most of the Python functions of the module Data Transformation we provide both a Python function and a scikit-learn transformer to extract engineered features from the data. Transformers can be used inside sklearn Pipelines, structured as seen in Figure 4 – Example of pipelineFigure 4, to define a sequence of pre-processing steps that must be applied to the data before fitting a model.



```
Pipeline
HolidayTransformer
HolidayTransformer(country='GR')
DegreeDaysTransformer
DegreeDaysTransformer(base_temperature={'CoolingDegreeDays': 21.0,
                                        'HeatingDegreeDays': 17.0})
WeeklyProfileTransformer
WeeklyProfileTransformer()
PolynomialFeatures
PolynomialFeatures(include_bias=False)
Lasso
Lasso(alpha=0.1)
```

Figure 4 – Example of pipeline

This has the advantage of generalizing and simplifying the prediction phase. One of the reasons is that an entire pipeline, which includes a chain of pre-processing steps and a final model, can be serialized and stored like a black-box function on the system. This black box can then be applied effortlessly on the input data whenever a prediction is needed. The other reason is that other than the hyperparameters of the ML model, also the parameters of the pre-processing steps can be fine-tuned using an optimization framework based on cross-validation, such as GridSearchCV. For example, we can ask the optimization algorithm to choose the best between different base temperatures for a degree day feature or whether to include or not the holiday feature based on its relevance.

II.2.5. Reinforcement learning techniques

II.2.5.a. Reinforcement Learning (RL) in the AI Toolbox for Buildings

Reinforcement Learning (RL) is a type of machine learning algorithm that is used to train agents to make decisions in an environment based on feedback in the form of rewards or punishments. This technique is applied in Application A6 (gas demand-response) to estimate future residential gas consumption. These estimation s are used to select the optimal set of gas boilers when offering demand-response services to the gas operator.

II.2.5.b. Example of RL in Gas Demand-Response application

Reinforcement Learning problems are typically formulated as a Markov Decision Process (MDP). An MDP is a mathematical framework for modelling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. It consists of a set of states, actions, rewards. A state represents the current situation of the system being modelled and captures all the relevant information needed to make decisions. An action refers to the decision made by an agent in a particular state. A reward is a numerical value that quantifies the desirability or quality of a particular state-action pair. It represents the immediate benefit or cost of taking a specific action in a specific state. The relationship between the state, action and reward is essential in an MDP. At each time time-step, the agent observers the current state, selects an action based on a policy and receives a reward based on the state-action pair, as shown in Figure 5 below:

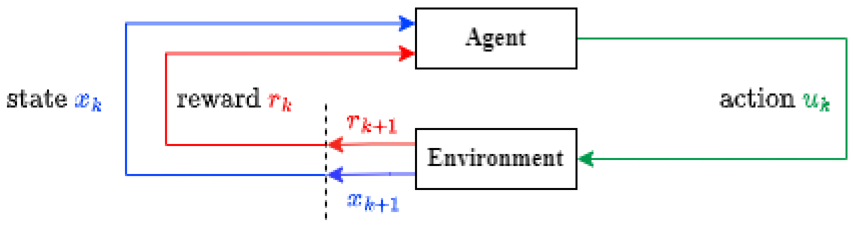


Figure 5 - The elements and relations in an MDP, where *k* is the current time-step number.

In the case of estimating residential gas usage, the states represent different environmental conditions such as boiler (internal) temperatures, ambient temperatures, and time of day. The actions represent a heating action. The rewards represent the value associated with each state and action, which is energy consumption.

II.2.5.c. Markov Decision Process

RL is often built upon the MDP framework. This means that when the RL problem and objective is explained, it should be started with a definition of the MDP. This section dives into each of the three MDP components (state, action & reward), by first explaining more in depth about the definition and usage of the component, which is followed on how this component is used in practice for the specific use case.

II.2.5.c.1. States

A state represents the current situation or configuration of the environment at a specific point in time. It captures all the relevant information necessary to make decisions. In an MDP, states are often described as Markov states, meaning that the future state depends only on the current state and action taken, not on the history of previous states and actions.

A variant on the state is the partially observable state. A partially observable state refers to a situation where the agent does not have complete access to the underlying state of the environment. Instead, the agent only has partial or incomplete information about the true state and must rely on observations or sensory inputs to make decisions.
In other words, the agent's knowledge about the environment is limited to a set of observations that it receives based on its perception or sensing capabilities. These observations are often influenced by noise, uncertainty, or other factors that can further obscure the true underlying state.

In the current application, states are defined based on the assumption that a single household can be modelled as partially observable. To provide the agent with sufficient, causal information, we use the following features:

- time (t)
- Outside Air Temperature (t_out)
- Room Temperature Setpoint (t_r_set)
- Past Room Temperatures ([t_r,t-k, ..., t_r,t-1])
- Past Boiler Modulations ([b_m,t-k, ..., b_r,t-1])
- Current Room Temperature (t_r)

The value of k (referred to as depth) is dependent on the data frequency and is set such that the state comprises information from the past 48 time-steps (equals past 4 hours). In other words, when the measurement frequency is 5 minutes, then k should be set to 48 to obtain a 4-hour depth. Parameter time is integer between 0 and 1,440, multiples of 5.

II.2.5.c.2. Actions

An action is a decision or choice made by an agent in a particular state. It represents the behavior or strategy that the agent selects to transition from one state to another. Actions can be deterministic or stochastic, depending on the nature of the environment. A deterministic action leads to a specific outcome, while a stochastic action has some degree of randomness or uncertainty associated with its consequences.

In the current application, the actions are assumed to be boiler setpoint values that are transformed into binary ON/ OFF values. The threshold for the boiler setpoint value is set to 20°C, implying any value above this leads to action ON (1) and a setpoint value below this is considered as action OFF (0). The setpoint value of 20°C is chosen because all boiler in the pilot responded correctly to this value. The threshold value is a configurable parameter and be adjusted to suit other boiler models or can be extended to a continuous action space to control the heating of a boiler more precisely.

II.2.5.c.3. Rewards

A reward is a numerical signal that represents the instantaneous desirability or quality of an agent's interaction with the environment. It provides feedback to the agent after taking an action in a specific state. The reward signal guides the agent's learning process by assigning higher values to desirable outcomes and lower values to undesirable ones. The goal of the agent is typically to maximize the cumulative rewards it receives over time.

In the current application, the instantaneous rewards are modelled as the boiler modulation at the current instant (b_m,t). Any Q-function trained using this reward leads to a value corresponding to the cumulative gas consumption of that house.

For receiving measurements, we use a frequency of one measurement per minute d. A measurement includes two types of parameters: (1) Boiler side parameters include any values obtained from the boiler.

These are values as water temperatures, boiler setpoints, modulations, etc. (2) Boiler aggregator values include values related to rooms, such as the room temperature, room setpoint, etc. Note that neither of the sensors measure the “true” gas consumption, however the boiler modulation is considered as a good proxy for it. The boiler modulation describes the gain of the boiler, which can be translated back to a real power value (kW) if the characteristics of the boiler are known.

II.2.5.d. Data

An offline model-free RL algorithm needs historical time-series data about states, actions and rewards that occurred on the (real-world) environment. After an agent is trained, real-time states and actions can be fed to the agent to obtain a reward estimation, which is called inference.

Time series data refers to a sequence of data points collected or recorded in chronological order at regular intervals. It involves observations or measurements taken over a specific period, such as seconds, minutes, hours, days, months, or years. Time series data can be used for forecasting and predicting future states or rewards in RL. By analyzing historical patterns and trends, RL agents can make predictions about the future state of the environment, enabling proactive decision-making and planning.

More specifically for A6, the time-series data contains measurements of two types of parameters: (1) Boiler side parameters include any values obtained from the boiler. These are values as water temperatures, boiler setpoints, modulations, etc. (2) Boiler aggregator values include values related to rooms, such as the room temperature, room setpoint, etc. Note that neither of the sensors measure the “true” gas consumption, however the boiler modulation is considered as a good proxy for it. The boiler modulation describes the gain of the boiler, which can be translated back to a real power value (kW) if the characteristics of the boiler are known.

II.2.5.e. Algorithm

To train the agents, we follow a backward FQI approach (Develder, Claessens, & Gokhale, 2022). The backward FQI (Fitted Q-Iteration) approach is a way to train a reinforcement learning agent by iteratively estimating the optimal action-value function for a Markov Decision Process (MDP). The action-value function tells the agent how good it is to take a certain action in a certain state. The optimal action-value function gives the maximum possible value the agent can achieve by taking any action in any state. We modify this approach by evaluating an existing policy, instead of taking an optimal action and defining a new policy. Using this approach, we can approximate the future accumulative reward given a certain state and action.

The backward FQI approach works by first collecting a dataset of state-action pairs and their associated rewards. Then, it iteratively fits a function to this data that estimates the optimal action-value function.

Starting with an initial estimate of the optimal action-value function, the backward FQI approach uses a supervised learning algorithm to fit a new estimate to the data. This new estimate is then used to update the agent's policy, which determines which action to take in each state. The agent then collects a new dataset of state-action pairs using the updated policy, and the process repeats. This process is called an iteration. The backward FQI approach is "backward" because it works by estimating the optimal action-value function starting from the final state of the MDP and working backwards towards the initial state. The "Fitted" part of the name refers to the fact that the function is fit to the collected data, rather than being learned through direct interaction with the environment.

The number of iterations is chosen based on the data frequency to obtain a Q-function corresponding to the cumulative gas consumption for a day (24 hours). E.g., for a data frequency of 5 min, leading to 288 total iterations (and function approximators). To provide additional stability, we follow an ensemble-based approach called meanQ, which uses a set of *N* function approximators per iteration (*N* is configurable) and computes the mean value of their prediction for calculating the Q-value. This function has been implemented in the AgentAgent[12] as meanQ.

Additionally, because we use neural networks as functional approximators, the obtained data must be scaled before the regression step. To enable proper scaling for different environments, we use the transformation_classtransformation_class. [13] This takes as input the house_id and produces a transformation object that can be used by the agent for all scaling and re-scaling operations.

II.2.5.f. Output

This repository is used for training the RL agents. Once training is done, the models are saved for later usage. In A6, this corresponds to usage inside a DR-coordinator function, where the RL-agent is employed to find the most suitable household for a DR-response.

Used in: A6

II.3. Existing tools and standards used

This section explores the tools and standards utilized in the development and management of the project. It discusses the Python and R libraries employed, the use of GitHub for code versioning, the incorporation of ML Flow for lifecycle code management, adherence to code conventions, and the implemented test and verification process.

II.3.1. Python libraries

Along with the Function Blocks to be developed, the AI Toolbox relies on well-known open-source Python libraries, used as a basis to implement other functions to support the BIGG use cases. A non-exhaustive list of the dependencies is:

- **Pandas**: provides the basic data structures and algorithm to represent, explore and manipulate datasets (pandas.pydata.org).
- **Numpy**: provides data structures and algorithms related to the numerical computation, like arrays, matrices, linear algebra functions, etc. (numpy.org)
- **Sklearn**: used especially for the machine learning modelling part. It provides the implementation of several machine learning algorithms and other tools to support and evaluate the modelling process, like optimization and cross-validation frameworks, performance metrics, etc... . (scikit-learn.org/stable/)

[12] https://raw.githubusercontent.com/biggproject/biggpy/main/gas_demand_response/rl_training/rltraining/rl_agents/offline_fqi.py

[13] https://raw.githubusercontent.com/biggproject/biggpy/main/gas_demand_response/rl_training/rltraining/rl_agents/utils/data_transformations.py

- **Statsmodels:** complements the Sklearn library, offering other machine learning algorithms more suited for time-series data and functions for conducting statistical tests and data exploration. (statsmodels.org/stable)

II.3.2. R libraries

In the case of R, multiple external libraries (a.k.a. packages) where used along biggr implementation, such as:

- **Lubridate:** provides tools that make it easier to parse and manipulate dates. (lubridate.tidyverse.org)
- **Readr:** provides a fast and friendly way to read rectangular data (like 'csv', 'tsv', and 'fwf'). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. (readr.tidyverse.org)
- **Tidyr:** is a set of tools to help to create tidy data, where each column is a variable, each row is an observation, and each cell contains a single value. This package contains tools for changing the shape (pivoting) and hierarchy (nesting and 'unnesting') of a dataset, turning deeply nested lists into rectangular data frames ('rectangling'), and extracting values out of string columns. It also includes tools for working with missing values (both implicit and explicit). (tidyr.tidyverse.org)
- **Zoo:** is an S3 class with methods for totally ordered indexed observations. It is mainly aimed at irregular time series of numeric vectors/matrices and factors. The package key design goals are independence of a particular index/date/time class and consistency with ts and base R by providing methods to extend standard generics. (cran.r-project.org/web/packages/zoo/index.html)
- **Roll:** provides a fast and efficient computation of rolling and expanding statistics for time-series data. (cran.r-project.org/web/packages/roll/index.html)
- **Padr:** transforms datetime data into a format ready for analysis. It offers two core functionalities; aggregating data to a higher-level interval (thicken) and imputing records where observations were absent (pad). (cran.r-project.org/web/packages/padr/vignettes/padr.html)
- **Quantreg:** provides a framework to estimate and infer models of conditional quantiles. Specifically, Linear and nonlinear parametric and non-parametric (total variation penalised) models for conditional quantiles of a univariate response and several methods for handling censored survival data (cran.r-project.org/web/packages/quantreg/index.html)
- **Testthat:** is a testing framework for R that is easy to learn and use, and integrates with your existing 'workflow' (testthat.r-lib.org).
- **Kernlab:** provides kernel-based machine learning methods for classification, regression, clustering, novelty detection, quantile regression and dimensionality reduction. Among other methods, includes Support Vector Machines, Spectral Clustering, Kernel PCA, Gaussian Processes and a QP solver. (cran.r-project.org/web/packages/kernlab/index.html)
- **fastDummies:** create dummy columns with categorical variables (character or factor types). Much faster than model.matrix() integrated in base R package.

  (cran.r-project.org/web/packages/fastDummies/index.html)

- **Caret**: provides miscellaneous functions for training and plotting classification and regression models. Caret (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for: data splitting, pre-processing, feature selection, model tuning using resampling, variable importance estimation, as well as other functionality. (cran.r-project.org/web/packages/caret/index.html)
- **penalised**: is used for fitting possibly high dimensional penalised regression models. The penalty structure can be any combination of an L1 penalty (lasso and fused lasso), an L2 penalty (ridge) and a positivity constraint on the regression coefficients. (cran.r-project.org/web/packages/penalized/index.html)
- **onlineforecast**: is a framework for fitting time-adaptive forecasting models. Provides a way to use forecasts as input to models, e.g., weather forecasts for energy-related forecasting. The models can be fitted recursively and easily set up for updating parameters when new data arrives. (cran.rstudio.com/web/packages/onlineforecast/index.html)

II.3.3. GitHub

GitHub provides cloud hosting for software development and version control through Git. In essence, it provides distributed version control and source code management (SCM) functionality, plus other tailored features. Every project has access control and collaboration features including bug tracking, feature requests, task management, continuous integration, and wikis.

All code and documentation repositories of the BIGG project are hosted in GitHub. A BIGG user account was created containing the R and Python AI-toolbox libraries, so-called biggr and biggpy respectively, and the language-agnostic documentation of the AI toolbox, so-called biggdocs. All repositories are public, so freely available to everybody.

Besides, the GitHub task management functionality was used in the biggdocs repository for project management of the AI-toolbox implementation. A similar management framework will be used for the implementation of the business cases pipelines.

- BIGG GitHub account: https://github.com/biggproject
- Documentation of the AI Toolbox: https://github.com/biggproject/biggdocs
- Python implementation of the AI Toolbox: https://github.com/biggproject/biggpy
- R implementation of the AI Toolbox: https://github.com/biggproject/biggr

II.3.4. Lifecycle code management - ML Flow

It was decided to use MLflow as an open-source framework to manage the entire lifecycle of the AI Toolbox applications. This ML platform is based on four components:

- MLflow Tracking: allows any piece of data science code to be recorded as a run and organized as an experiment. The tracking component keeps trace of two main elements: artifacts, such as figures, serialized models, configuration files, model summaries and machine learning entities, such as model hyper-parameters, performance metrics, and other metadata related to machine learning. Artifacts can be logged in a local or remote artifact store, for example in an Amazon S3 bucket, while MLflow entities are usually recorded in local or remote databases, such as PostgreSQL. MLflow also allows us to store interactive html figures, that comes in handy when monitoring the pipeline. This way, one can double-check if an intermediate step of the workflow has produced the expected outcome, for example inspect the outlier plot, an autocorrelation plot or the predictions of a model.

- MLflow Projects: allows any data science project to be packaged together with its dependencies, entry points and environment. This way, the code can run on several platforms in a reusable and reproducible way. For example, the environment on which to run the code can be a docker container with the latest version of the AI toolbox already pulled from GitHub and installed.
- MLflow Models: allows to package and deploy machine learning models using standard formats. For example, a model can be deployed as a self-contained Docker image with a REST API endpoint and serve prediction requests. However, models can also be deployed on cloud platforms like Microsoft AzureML or Amazon Sagemaker.
- MLflow Model Registry: is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. Other than storing the model, this component provides also model versioning, model lineage and stage transitions.

All these components together facilitate and speed up the monitoring, deploying and model versioning phases, without reinventing custom solutions.

II.3.5. Code conventions

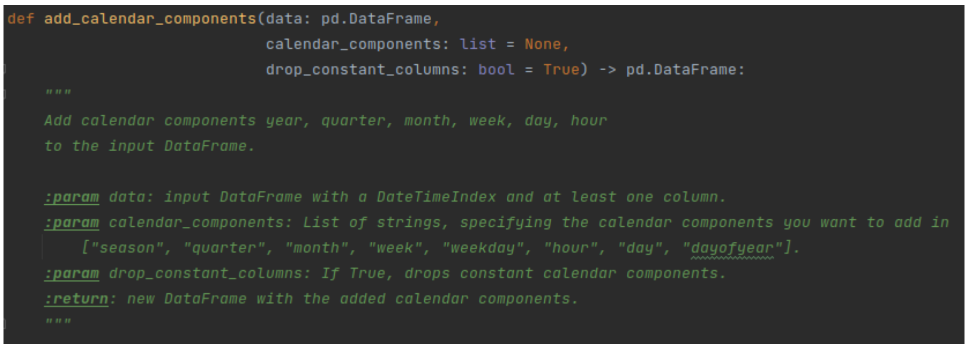
The AI Toolbox follows the common coding style conventions and best practices.

For Python, it is compliant with the following guidelines:

- PEP8 (https://www.Python.org/dev/peps/pep-0008/),
- PEP257 (https://www.Python.org/dev/peps/pep-0257/).

PEP8 defines general coding conventions for Python to improve the code readability, make it consistent across different libraries and projects and ease collaboration between developers. PEP257 addresses one aspect of coding conventions: docstrings. A docstring is a string literal that is used to document a segment of code and occurs as the first statement in a module, function, class, or method definition.

For example, as seen in Figure 6, the docstring of a function usually begins with a brief description of the functions, the parameters and the return values.



```
def add_calendar_components(data: pd.DataFrame,
                            calendar_components: list = None,
                            drop_constant_columns: bool = True) -> pd.DataFrame:
    """
    Add calendar components year, quarter, month, week, day, hour
    to the input DataFrame.

    :param data: input DataFrame with a DateTimeIndex and at least one column.
    :param calendar_components: List of strings, specifying the calendar components you want to add in
        ["season", "quarter", "month", "week", "weekday", "hour", "day", "dayofyear"].
    :param drop_constant_columns: If True, drops constant calendar components.
    :return: new DataFrame with the added calendar components.
    """
```

Figure 6 – Example of a function docstring

II.3.6. Test and verification process

The testing framework used to test the Python code is "unittest", which is based on the main concepts of test case, test suite and test fixtures: https://docs.Python.org/3/library/unittest.html. A test suite, i.e., an aggregation of tests executed together, was written for each module of the AI Toolbox. Each function of the toolbox can be tested versus different sets of inputs, whenever possible, to check if its response matches the expected one. This is the general concept of a test case. A test fixture is some code that prepares the test environment for an entire test suite such as importing data, connecting and creating databases or directories and implementing clean-up actions at the end. Generally, the purpose of writing tests is to make sure that some code works as expected in different environments and that new code added to the library does not break the other functionalities.

The other tool used for the test and verification process is called "tox": https://tox.wiki/en/latest/. It is a command line tool to check that some package or library can be installed and tested successfully in multiple environments, such as using different Python interpreters. Tox will create one virtual environment for each Python interpreter specified in a configuration file, install the package or library in that environment and run all the tests. This is particularly useful to ensure that the AI Toolbox works correctly with different versions of Python.

To provide further support to the documentation process, we created and included for most of the functions in the AI Toolbox a Jupyter notebook which serves as a guideline on how to use them: https://github.com/biggproject/biggpy/tree/main/ai_toolbox/notebooks.

Some of the datasets imported in the Jupyter notebooks and in the tests are "toy datasets" already integrated in other libraries as submodules, such as "sklearn.datasets". The dataset used for testing models developed for Application A6 is a residential heating dataset, including past temperature and gas consumptions. The data is handled, and models are trained using submodules and functionalities provided by "pytorch_lightning".

In the case of R, the "testthat" library was used to run tests every time the library is compiled, mainly checking that each function is providing an expected response to a given set of inputs. The implementation is quite similar to the 'unittest' used in the Python library and has the same objectives. Regarding the user tutorials implemented in R, multiple datasets containing electricity consumption and weather data of six different buildings are included inside the biggr package.

III. Business driven pipelines

The AI toolbox described in the previous chapter provides a set of core functionalities that can be used to develop specific AI solutions for various applications. In this chapter, we present the 6 applications for which the project has developed AI solutions. For each application, we first describe the business need and then the technical solution offered using the AI toolbox[14]. The technical solution is described through the developed pipeline that uses the core modules of the toolbox. To facilitate the understanding of these pipelines, we provide links to the relevant GitHub repository where a version of the pipeline is available as a notebook (e.g. a Jupyter notebook). This chapter thus serves as a practical demonstration of the versatility and effectiveness of the AI toolbox in addressing specific business needs in the energy sector.

III.1. Application A1: Energy benchmarking of buildings

III.1.1. Business application

Application A1[15] is about Benchmarking and monitoring energy consumption in buildings. Its main objectives are:

1. First, to assess the energy consumption of a single building using its historical consumption and weather data as input (also known as longitudinal benchmarking). It essentially consists of disaggregating the whole consumption into three components: baseload, heating and cooling. With these components and static information of the building, several Key Performance Indicators (KPIs) are estimated to assess the usage, cost, savings and emissions due to energy consumption over time.
2. Second, to compare the individual KPIs generated during the longitudinal benchmarking process versus similar buildings in terms of characteristics and weather conditions. In literature, this is referenced to as cross-sectional benchmarking.

In Application A1, both objectives are completely covered from a data perspective. Thus, Application A1 needs harmonized input data to start the analysis and terminates when the harmonized data outputs are written in files.

III.1.2. Technical solution as a Pipeline

The input needed for the computation of this Application are the datasets that have been previously harmonized to the BIGG data model, initially coming from different sources. This process aims to link the raw data sources to a common model that provides clear relationships between building characteristics,

[14] The pipelines required to bring solutions to the different applications were developed in the context of the task T5.4 – AI/ML based service modules as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. The applications are attached to specific Business Cases (BCs) and Use Cases (UCs) described in the project proposal. The relation with BCs and UCs will be indicated for each application described in the present chapter.

[15] Application A1: energy consumption benchmarking relates to the Business case #1 - Benchmarking and energy efficiency tracking in public buildings, use case #1 - Benchmarking and monitoring of energy consumption (BC1/UC1) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation were the responsibility of CIMNE.

locations, metering devices, and other energy-related information about buildings. In this application, the needed input datasets are:

- Smart-metered electricity time series (at least hourly frequency) gathered through Datadis, which is the Spanish DSOs data platform.
- Monthly gas or electricity consumption gathered through Gemweb, a company that stores the energy consumption of the Catalan government buildings during the last 4 years for energy auditing purposes.
- Building characteristics obtained from the Spanish Cadaster INSPIRE-harmonised datasets and the GPG repository, which is the Catalan government buildings catalog.
- Local weather data gathered through the Darksky online services. In the case of solar radiation, the information is gathered using the Copernicus Atmospheric Monitoring Service (CAMS).

The outputs generated for each building in analysis are weather and building-size normalized Key Performance Indicators (KPIs) related with energy consumption at different time spans. Among others, estimations of energy, cost and emissions are compared to previous periods.

In the GitHub repository (https://github.com/biggproject/A1-Benchmarking) all source code, documentation, and test data is available. Figure 7 depicts the pipeline flowchart of Application A1 and the context with the deployment environments where it's running.

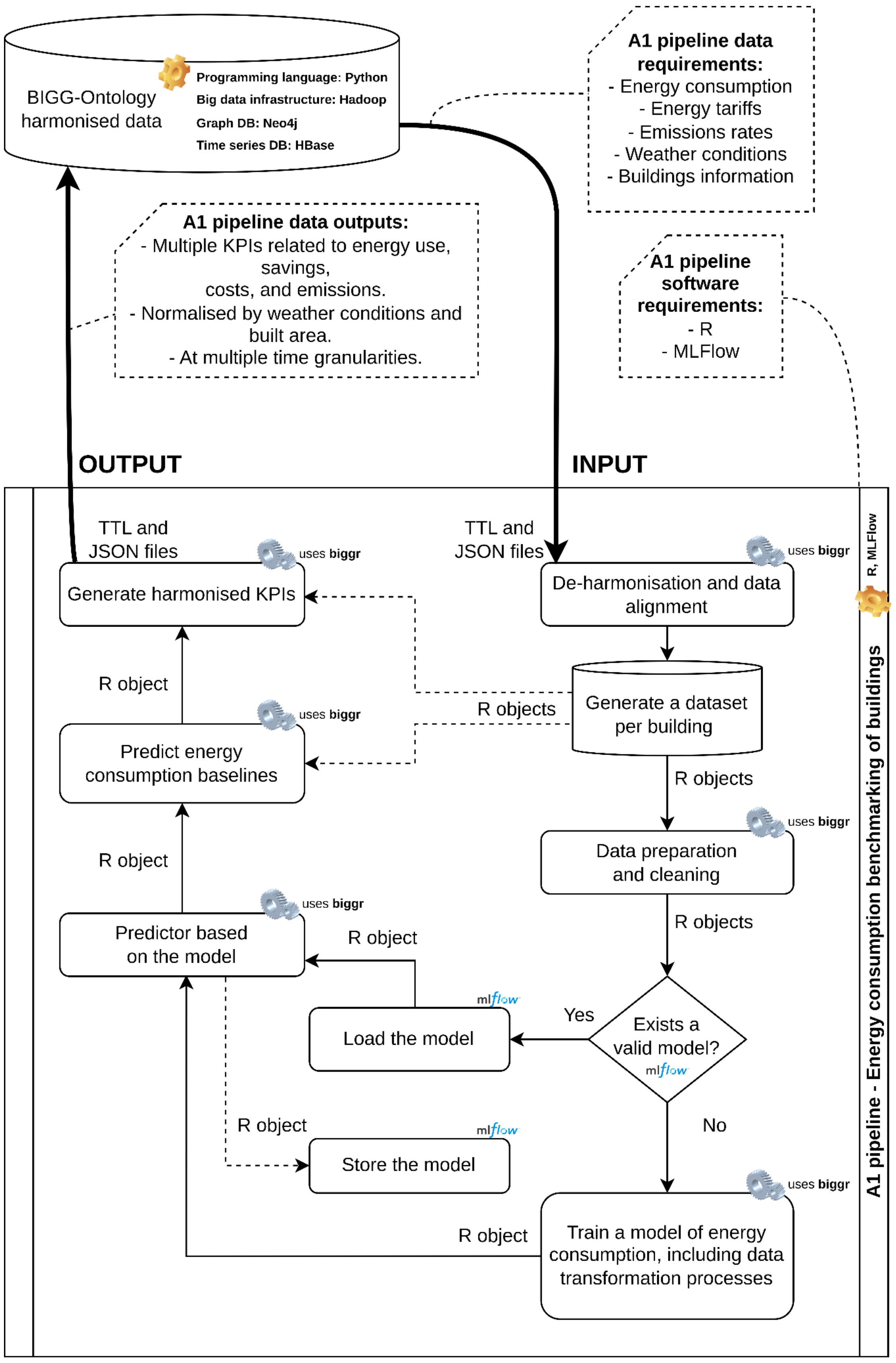

BIGG-Ontology harmonised data
Programming language: Python
Big data infrastructure: Hadoop
Graph DB: Neo4j
Time series DB: HBase
A1 pipeline data requirements:
- Energy consumption
- Energy tariffs
- Emissions rates
- Weather conditions
- Buildings information
A1 pipeline data outputs:
- Multiple KPIs related to energy use, savings, costs, and emissions.
- Normalised by weather conditions and built area.
- At multiple time granularities.
A1 pipeline software requirements:
- R
- MLFlow
OUTPUT
INPUT
TTL and JSON files
uses biggr
Generate harmonised KPIs
TTL and JSON files
De-harmonisation and data alignment
R object
R objects
Generate a dataset per building
Predict energy consumption baselines
R objects
Data preparation and cleaning
R object
Predictor based on the model
R object
R objects
mlflow
Load the model
Yes
Exists a valid model?
No
R object
Store the model
R object
Train a model of energy consumption, including data transformation processes
R, MLFlow
A1 pipeline - Energy consumption benchmarking of buildings

Figure 7 - Application A1 algorithm flowchart

III.2. Application A2: EEM Results Benchmarking

III.2.1. Business application

The Application A2[16] is about Energy Efficiency Measures assessment. Its main objectives are:

1. To assess the savings in terms of energy, cost and emissions generated by the implementation of a certain energy efficiency measure in a building. This assessment is based on the data-driven modelling of the energy consumption of a building before the implementation of an EEM and estimates its counterfactual consumption after the implementation. In similar terms to the energy benchmarking use case, this objective corresponds to the longitudinal assessment of EEM.
2. To do a cross-sectional assessment between all the estimated savings produced by some EEMs, with the objective of estimating savings that a certain EEM could produce in certain building.

For the sake of simplicity, Application A2 only covers the first objective, from a data perspective. Thus, it begins with an initial set of harmonized data and terminates when the harmonized data outputs are written in files.

III.2.2. Technical solution as a Pipeline

The inputs needed for the computation of these Applications are the datasets that have been previously harmonised to the BIGG data model, initially coming from different sources. This process aims to link the raw data sources to a common model that provides clear relationships between building characteristics, locations, metering devices, and other energy-related information about buildings. In this business case, the needed input datasets are:

- Smart-metered electricity time series (at least, hourly frequency) gathered through Datadis, which is the Spanish DSOs data platform.
- Monthly gas or electricity consumption gathered through Gemweb, a company that stores the energy consumption of the Catalan government buildings during the last 4 years for energy auditing purposes.
- Building characteristics obtained from the Spanish Cadaster INSPIRE-harmonised datasets and the GPG repository, which is the Catalan government buildings catalog.
- Energy Efficiency Measures (EEMs) implemented in each building with information about the application date(s), the typology of the measure, if it was applied jointly to other measures... This data originally comes from ICAEN databases.
- Local weather data gathered through the Darksky online services. In the case of solar radiation, the information is gathered using the Copernicus Atmospheric Monitoring Service (CAMS).

[16] Application A2: EEM results benchmarking relates to the Business case #1 - Benchmarking and energy efficiency tracking in public buildings, use case #2 - Energy Efficiency Measures assessment (BC1/UC2) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation were the responsibility of CIMNE.

In GitHub repository (https://github.com/biggproject/A2-EEM-assessment) all the source code, documentation, and testing data are available. Figure 8 depicts the pipeline flowchart of Application A2 and the context in terms of the deployment environments where the respective components are running.

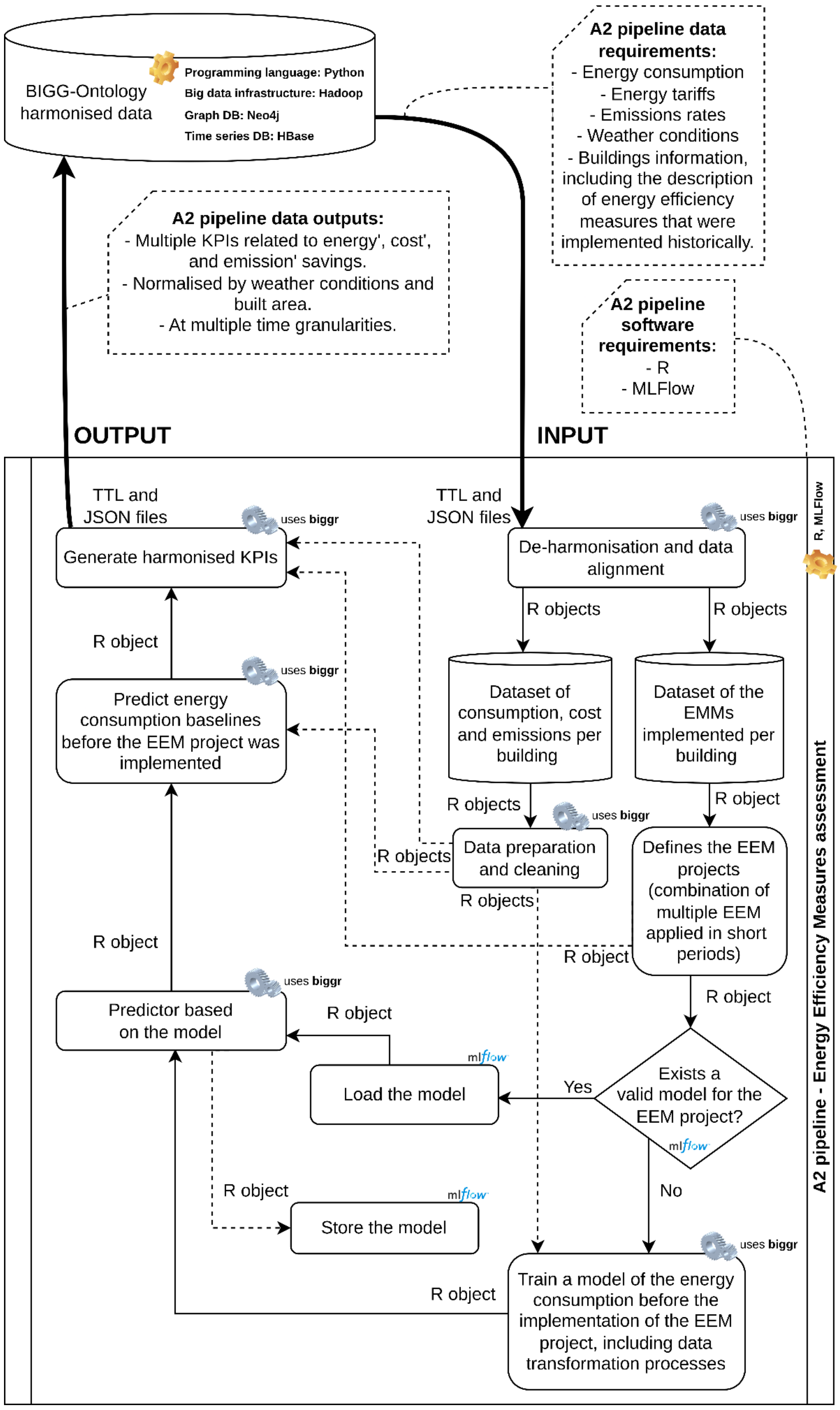

BIGG-Ontology harmonised data
Programming language: Python
Big data infrastructure: Hadoop
Graph DB: Neo4j
Time series DB: HBase
A2 pipeline data requirements:
- Energy consumption
- Energy tariffs
- Emissions rates
- Weather conditions
- Buildings information, including the description of energy efficiency measures that were implemented historically.
A2 pipeline data outputs:
- Multiple KPIs related to energy', cost', and emission' savings.
- Normalised by weather conditions and built area.
- At multiple time granularities.
A2 pipeline software requirements:
- R
- MLFlow
OUTPUT
INPUT
TTL and JSON files
TTL and JSON files
uses biggr
Generate harmonised KPIs
De-harmonisation and data alignment
R, MLFlow
R objects
R objects
R object
Predict energy consumption baselines before the EEM project was implemented
Dataset of consumption, cost and emissions per building
Dataset of the EMMs implemented per building
R objects
R object
Data preparation and cleaning
R objects
Defines the EEM projects (combination of multiple EEM applied in short periods)
R objects
R object
R object
Predictor based on the model
R object
R object
Load the model
mlflow
Yes
Exists a valid model for the EEM project?
No
R object
Store the model
R object
Train a model of the energy consumption before the implementation of the EEM project, including data transformation processes
A2 pipeline - Energy Efficiency Measures assessment

Figure 8 - Application A2 algorithm flowchart

III.3. Application A3: Baseline identification for Energy Performance Contracts

III.3.1. Business application

The Energy Performance Contract (EnPC) management application aims to streamline the Measurement and Verification (M&V) process, which is a crucial step for Energy Service COmpanies (ESCOs) in managing EnPCs. The M&V process involves identifying a baseline consumption model against which the actual consumption post-retrofit/renovation can be compared to estimate savings accurately. However, this process is typically managed manually with Excel sheets, which is time-consuming and prone to errors.

To standardize the M&V process, the BIGG consortium developed a solution using the AI toolbox to identify baselines accurately and flexibly.[17] The toolbox's core modules were used to build a pipeline that can adapt to any EnPC contract's requirement and identify a consumption baseline regression model from historical data, using weather, occupation and calendar data as input. The developed pipeline allows for a high degree of flexibility in terms of the types of models used to identify the baseline, enabling users to adapt the resulting models to their needs while ensuring interpretability by non-expert users.

The identified models can then be used as a building block of global solutions such as those developed in the context of the BIGG projects where the M&V process is managed from start to end, including the follow up of the savings, the financial benefits, both for the ESCO and the building owner, the reporting, etc. Such a complete solution will be described in D6.3.

III.3.2. Technical solution as a Pipeline

We describe the pipeline for identifying regression models through its input, process steps and output.

The pipeline input is composed of:

- The historical consumption data (typically hourly data in harmonized format from the BIGG data model),
- The weather data (temperature, irradiance, typically in harmonized format from the BIGG data model),
- The site meta-information (mainly its country, typically harmonized format from the BIGG data model),
- The training period (i.e., a pre-retrofit/renovation period spanning at least 1 year of time, and which must be representative of the normal conditions of the building),
- Model constraints (e.g., polynomial regression only).

The pipeline includes the following steps:

[17] Application A3: Baseline identification for Energy Performance Contracts relates to the Business case #4 - Energy Performance Contract-based savings in commercial buildings, use case #9 - Actual savings tracking realised by the Energy Efficiency Measures (EEMs) (BC4/UC9) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation were the responsibility of Helexia and Energis.

- Pre-processing of the data,
- Outlier detection,
- Feature engineering (e.g., weekly patterns, calendar features such as day of the week, hour of the day, etc.),
- Identification of the best regression model using the nested cross-validation for timeseries approach, evaluating the models' performances according to the IPMVP protocol, with optimization based on criteria such as CVRMSE (Coefficient of variation of the root mean square error), NMBE (Normalized mean bias error), and R^2 (the coefficient of determination) although other custom criteria can also be specified.

The output of the model includes:

- The best identified regression model,
- Confidence statistics (e.g., the RMSE which can be interpreted as a standard deviation),
- The objective function used for evaluation (combined criteria),
- The list of removed outliers.

The flowchart corresponding to this pipeline is illustrated in Figure 9.

Reference to GitHub: https://github.com/biggproject/A3-EPC-baseline-identification

Finally, this pipeline is quite generic to identify regression models of building consumption and can therefore be adapted to other applications where such a model would be required. Typical use cases include smart alerting or performance monitoring.

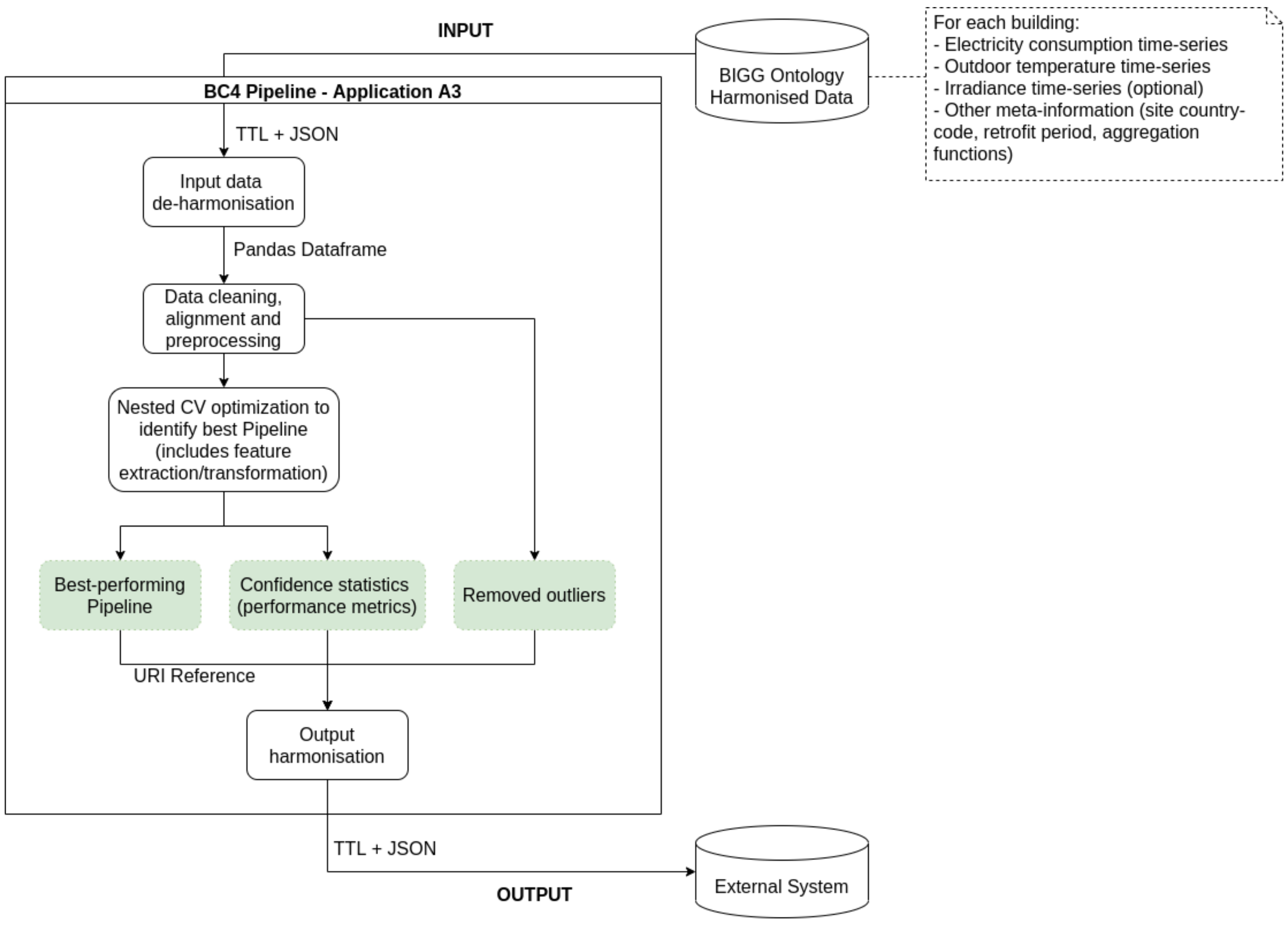

INPUT
BIGG Ontology Harmonised Data
For each building:
- Electricity consumption time-series
- Outdoor temperature time-series
- Irradiance time-series (optional)
- Other meta-information (site country-code, retrofit period, aggregation functions)
BC4 Pipeline - Application A3
TTL + JSON
Input data de-harmonisation
Pandas Dataframe
Data cleaning, alignment and preprocessing
Nested CV optimization to identify best Pipeline (includes feature extraction/transformation)
Best-performing Pipeline
Confidence statistics (performance metrics)
Removed outliers
URI Reference
Output harmonisation
TTL + JSON
OUTPUT
External System

Figure 9 - Application A3 algorithm flowchart

III.4. Application A4: Occupancy pattern detection for Comfort and energy optimisation

III.4.1. Business application

One of the challenges faced by energy experts is to optimize the energy usage of buildings while maintaining a comfortable environment for the occupants. Indeed, traditional Building Management Systems (BMS) are only focusing on the comfort conditions which is not compatible with the ever-growing need of building owners to reduce their energy bills. Moreover, most of the BMS algorithms do not take into account the current conditions of the building spaces, e.g., occupied/unoccupied, forecasted weather conditions and exceptional scenarios, e.g., holidays, exceptionally unoccupied. To achieve this conjunction of goals, controllers for HVAC systems must be adapted to exploit newly available information such as weather data, forecasts, and other factors like on-site production, and thereby enabling reduction of the energy consumption without compromising on comfort. Most importantly, an essential component is to be able to accurately detect occupancy patterns in the building because it determines when maintaining comfort conditions is critical and conversely when more flexibility exists for energy savings. Identifying such patterns can be a complex task when done manually.

The AITB can be used to automatically detect occupancy patterns in a building,[18] thus reducing the workload for energy experts and improving the accuracy of the data collected. The pipeline described below relies on movement sensor data as well as official holidays and calendar features. With this capability, the toolbox can be a valuable asset for energy experts looking to optimize energy usage and improve occupant comfort in buildings.

III.4.2. Technical solution as a Pipeline

We describe the pipeline for identifying occupancy patterns through its input, process steps and output.

The pipeline input is composed of:

- The activity counter time series, alias movement sensor data (typically hourly data in harmonized format from the BIGG data model),
- Holidays data (optional, using official holidays based on the country, province and state of the site by default),
- The site meta-information (mainly its country, typically harmonized format from the BIGG data model),
- The training period (i.e., a representative period spanning at least 1 year of time and ideally 3 years).

The pipeline includes the following steps:

- Pre-processing of the data (clean up, alignment of the values in time, etc.),
- Outlier detection,
- Feature engineering (mainly calendar features such as day of the week, hour of the day, etc., converted into cyclic components and public holidays),
- Identification of the best classification model using a nested cross-validation procedure.

The output of the model includes:

- The best identified predictive occupancy model, stored in serialized format,
- Confidence statistics (e.g., the RMSE which can be interpreted as a standard deviation),
- The objective function used for evaluation (combined criteria),
- The list of removed outliers.
- The training and validation sets used in cross-validation (essential to ensure the reproducibility of the results).

[18] Application A4: Occupancy pattern detection for Comfort and energy optimisation relates to the Business case #5 - Buildings for occupants: Comfort case, use case #12 - Optimisation using occupancy forecasts (BC5/UC12) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation was the responsibility of Helexia and Energis.

The flowchart corresponding to this pipeline is illustrated in Figure 10**Erreur ! Source du renvoi introuvable.**.

Reference to GitHub: https://github.com/biggproject/A4-Occupancy-pattern-detection

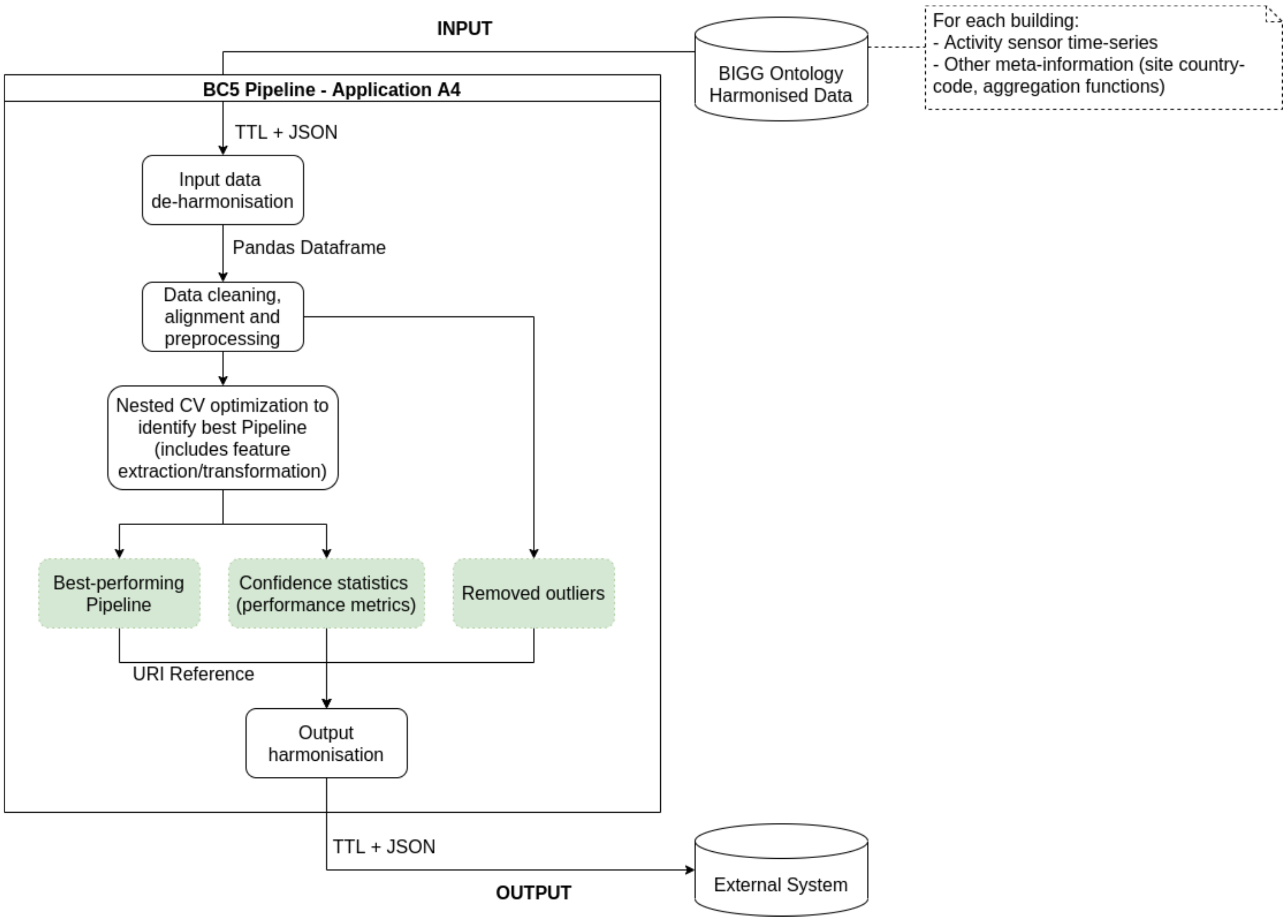

INPUT
BIGG Ontology Harmonised Data
For each building:
- Activity sensor time-series
- Other meta-information (site country-code, aggregation functions)
BC5 Pipeline - Application A4
TTL + JSON
Input data de-harmonisation
Pandas Dataframe
Data cleaning, alignment and preprocessing
Nested CV optimization to identify best Pipeline (includes feature extraction/transformation)
Best-performing Pipeline
Confidence statistics (performance metrics)
Removed outliers
URI Reference
Output harmonisation
TTL + JSON
OUTPUT
External System

Figure 10 - Application A4 algorithm flowchart

III.5. Application A5: Energy consumption forecasting

III.5.1. Business application

Forecasting energy consumption is crucial for effective energy management and planning. Accurate predictions can assist both energy providers and consumers in making informed decisions. To achieve this, an intelligent algorithm for energy consumption prediction can significantly improve the accuracy of forecasts and help develop more sustainable and cost-effective energy systems. Moreover, it can aid in reducing greenhouse gas emissions and mitigating climate change.

The main objective of Application A5[19] is to develop an AI algorithm that can accurately predict the future demand for energy. The algorithm can help energy providers to plan their energy production and distribution, optimize energy usage, and avoid blackouts. For energy consumers, accurate predictions can help manage their energy usage, reduce costs, and take advantage of off-peak hours. The algorithm can also identify trends and patterns in energy consumption, which can be used to develop efficient energy systems and policies, and ultimately create a more sustainable and reliable energy future for all.

Overall, this process will benefit both customers and energy providers by reducing energy costs, balancing energy demand, and reducing the strain on the energy grid during peak periods. This contributes to a more sustainable and efficient energy system. The flowchart corresponding to this pipeline is illustrated in Figure 11.

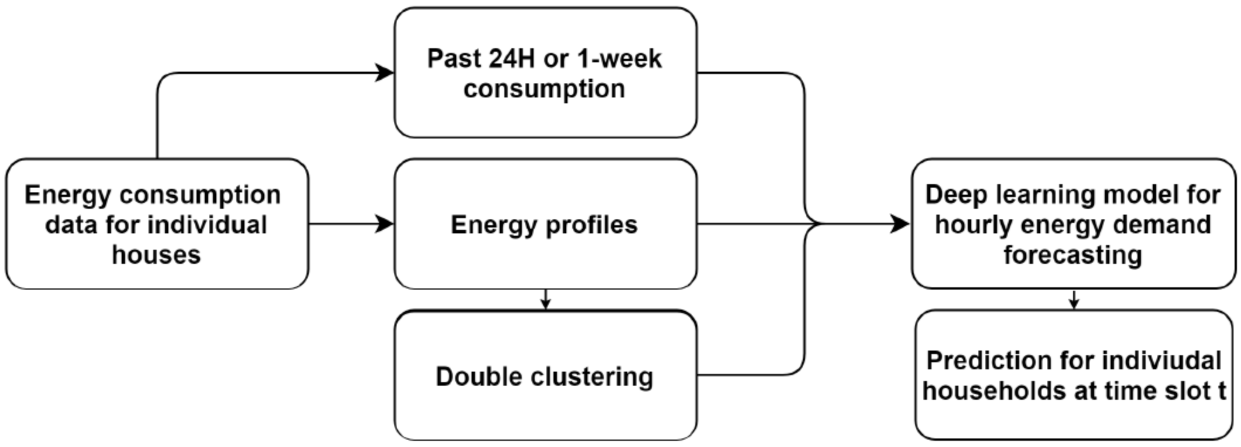

Past 24H or 1-week consumption
Energy consumption data for individual houses
Energy profiles
Double clustering
Deep learning model for hourly energy demand forecasting
Prediction for indiviudal households at time slot t

Figure 11 - Application A5 algorithm flowchart

III.5.2. Technical solution as a Pipeline

Historical data from smart meters is cleaned and processed to train machine learning models, which are evaluated to choose the most accurate one for forecasting residential power consumption. The model incorporates calendar components and non-time series calculated quantities that contribute to power consumption to provide a precise and reliable forecast. The model forecasts consumption data per device for the next 24-48 hours. Then we calculate overall costs by taking into account energy usage and rates. A dynamic threshold is set to limit energy usage without exceeding a customer's budget, and recommendations are provided to optimize energy usage and reduce costs.

[19] Application A5: Energy consumption forecasting relates to the Business case #6 - Flexibility potential of residential consumers on electricity and natural gas, use case #14 – Electricity Demand Response (BC6/UC14) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation were the responsibility of Inetum.

As seen in Figure 12, the pipeline input is composed of:

- A dataframe with three columns: Timestamp, Device_id, and Power (watt). Each row of the table represents a specific measurement of power consumption from a particular device at a specific time.
- The first column, Timestamp, records the date and time of each measurement in UTC time zone.
- The second column, Device_id, identifies the device from which the measurement was taken, ranging from the first device (Device_id=1) to the last device (Device_id=…). Note that the device id is an 8-10 alphanumeric character, for simplicity we use numeric values.
- The third column, Power (watt), records the amount of power consumed by the device at the corresponding timestamp, measured in watts. The data includes hourly measurements for each device over that time period.

Timestamp	Device_id	Power (watt)
2021-07-01 00:00:00+00:00	1 (first data of the first device)	1112.439992
…	…	…
2021-07-20 23:00:00+00:00	1 (last data of the first device)	14.147903
2021-07-01 00:00:00+00:00	2 (first data of the second device)	1054.166761
	..	…
2021-07-20 23:00:00+00:00	2 (last data of the second device)	93.285707
…	…	…
…	… (all the intermediary devices)	…
…	…	…
2021-07-01 00:00:00+00:00	32 (first data of the last device)	1146.482836
	…	…

Figure 12 - Example of the raw input data

The pipeline includes the following steps:

- Pre-processing of the data (clean up, alignment of the values in time, etc.),
- Outlier detection,
- Feature engineering (mainly calendar features such as day of the week, hour of the day, etc),
- Generate energy profiles to learn consumption patterns and characteristics of individual consumers. A "double" clustering procedure is conducted to group households with similar energy profiles, resulting in an encoding for each energy profile based on its distance from each cluster's centroid. Both the energy profiles and the clustering distances are used as additional input features for the neural network that predicts the hourly energy demand for individual households.
- A data augmentation technique is employed in which the predicted hour and the preceding 23 hours of data are utilized to generate the next hour's prediction. This iterative process is repeated until a 24 or 48-hour forecast is produced.

As seen in Figure 13, the pipeline input is composed of:

- The predicted power consumption in Watt (W) for each device at different timestamps on July 21st. The Device_id corresponds to each device, and the timestamp indicates the time of the day for which the prediction is made. For example, the first device has a predicted power consumption of 1,215.43 W at 00:00 , followed by hourly prediction and ends with 82.87 Wat 23:00. Similarly, the second device. The same goes for all intermediary devices, with each device having a predicted power consumption value for each hour of the day.
- Overall, the output can be used to optimize energy operations and manage energy usage effectively for each device. Now depending on usage, a recommendation algorithm will suggest if the device can be operated on that day or not.

Timestamp	Device_id	Predicted power (watt)
2021-07-21 00:00:00+00:00	1 (first device at 00:00)	1215.43
…	…	…
2021-07-21 23:00:00+00:00	1 (first device at 23:00)	82.87
2021-07-21 00:00:00+00:00	2 (second device at 00:00)	919.26
	..	…
2021-07-21 23:00:00+00:00	2 (second device at 23:00)	121.32
…	… (all the intermediary devices)	…
2021-07-21 00:00:00+00:00	32 (last device at 00:00)	1366. 37
	…	…
2021-07-21 23:00:00+00:00	32 (last device at 23:00)	224. 55

Figure 13 - Shape of the (desired) final output (predicted power consumption)

The flowchart corresponding to this pipeline is illustrated in Figure 14.

Reference to GitHub: https://github.com/biggproject/biggpy/tree/usecase14

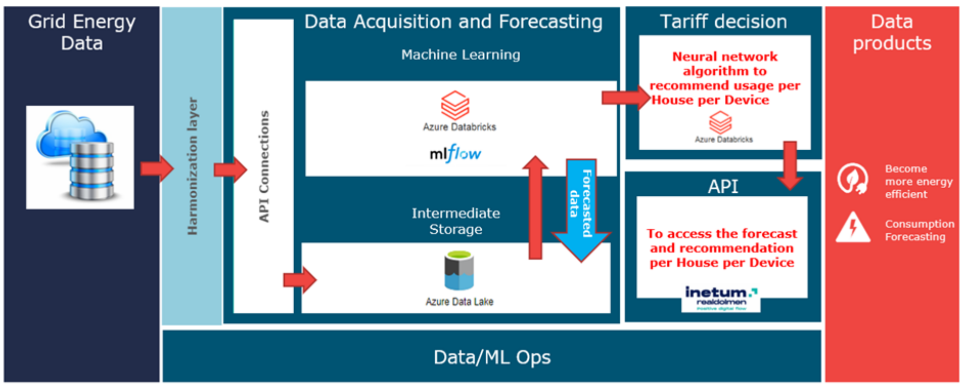


Figure 14 - Data architecture diagram of the ultimate solution with the corresponding tools/resources

III.6. Application A6: Gas demand response

III.6.1. Business application

The goal of Application A6[20] is to develop a demand response (DR) scheme exploiting gas flexibility in space heating for residential complex. Gas providers can avoid additional costs or CO2 emissions by letting customers play an active role in DR, thereby restoring the balance between supply and demand. The flexible assets used in this project for providing DR are gas-based domestic hot water boilers used for residential hot water and heating. The objective of the pilot is to meet a targeted gas consumption level. This is done to optimize energy efficiency and minimize gas usage, resulting in cost savings and reduced environmental impact. The pilot achieves this objective by jointly controlling the gas consumption of multiple boilers, ensuring they operate collectively to meet the target consumption. When controlling, the comfort of the user must be taken into account: the house may not cool down or warm up too far from the thermostat temperature. The cost deviations of additional heating or cooling (compared to normal consumption) also play a role. For example, with a high target consumption (where several boilers will be activated), ideally boilers are activated first whose heat demand was already expected for a later moment.

We utilize a reinforcement learning (RL) approach to develop a demand response controller policy. The RL agent learns from historical and/or simulated data to determine optimal actions based on the input raw data. The learned policy is then applied in real-world scenarios to implement the demand response strategy. The specific decisions made by the policy depend on the input data and the objectives of the demand response, such as adjusting the operation of boilers to meet target gas consumption or optimizing energy usage in response to changing conditions. The pipeline in Figure 15 shows the flow of raw data, that is used to learn an RL agent that implements the DR.

III.6.2. Technical solution as a Pipeline

Training the RL policy necessitates the definition of an agent and an environment. The environment is defined as real-world environment, a household, that translates an action (heating / no heating) into new state at every time-step. The agent is the entity that takes the actions, which consist of disabling or enabling the room heating. Enabling results in an increasing room temperature and an immediate consumption of gas usage; disabling sets the gas consumption to zero, and slowly declines the room temperature. The reward is defined as the immediate gas consumption given an action is taken. As gas consumption in kWh is not given, we use a modulation level (power percentage of total gas boiler power) as proxy. We refer to the RL-Training documentation on GitHub (link) for more details about the state, action and reward definition.

In this deliverable, we focus on modelling flexibility, which is expressed as preponing or postponing heating actions against a cost. Here, the cost is the accumulative gas consumption over the period (e.g., 24 hours) after the taken action. An offline model-free RL-approach is used to estimate this cost. Historical heating

[20] Application A6: Gas demand response relates to the Business case #6 - Flexibility potential of residential consumers on electricity and natural gas, use case #15 – Natural Gas Demand Response (BC6/UC15) as described in the BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002. Its development and implementation were the responsibility of IMEC (UGent).

actions and temperature and gas usage parameters are used to train the RL-agent. As each household has different heating demands and parameters, there will be an RL-agent for each household.

RL-agent inputs: historical residential data collected per household including:

- Timestamp (DD/MM/YY hh:mm:ss),
- Gas consumption/modulation,
- Room temperature/Boiler temperature,
- Boiler set points,
- Outside temperature, and,
- Room temperature set points.
- Proposed action (heating/no heating)

Outputs:

- Estimated gas consumption of upcoming configurable horizon-period (set to 24 hours)

The pipeline comprises periodic processes, such as retrieving the latest measurement data and training an RL-agent. Other processes are executed when a DR-event is triggered. Both types of processes are listed and explained next.

<u>Periodically (e.g., every hour or every day)</u>:

- Retrieving (real-time) measurement data required for training a RL-agent, or for monitoring during DR events. Typically, during business-as-usual (when there is no ongoing DR-event), the data retrieval interval can be lower than during a DR-event, where a higher measurement frequency is desired for fine observation of power values.

- Training a reinforcement learning agent (policy evaluation). An agent is trained separately for each household. This agent can provide a prediction of the accumulative energy consumption for the next horizon (e.g., 24 hours) at any time of the day, given the activity of the boiler and a set of properties, including internal boiler temperatures, the thermostat temperature and indoor/outside temperature.

<u>During DR events:</u>

- Using the reinforcement learning agent (inference) to predict the expected gas consumption. In addition, the target consumption is calculated that is aimed for during the DR event.

- Participating boilers are ranked on the highest expected gas savings, which is calculated by the outputs of the RL agents. To be more concise, from each agent the expected gas consumption difference is calculated between the default -business as usual- action and the proposed action. This value is also called the advantage value. The proposed action depends on the DR-objective: upwards DRs have proposed action ‘heating’ and downwards DRs have action ‘no heating’. In other words, for downward DR events, the expected savings equals taking an alternative (proposed) action instead of the action that would otherwise would have happened. The expected gas savings are calculated while comfort constraints are considered. The comfort limits in the pilot are set at a maximum (negative and positive) deviation of 1 degree Celsius from the thermostat temperature.

  A separate entity, called the ‘action-coordinator’ gathers all agents and calculates the expected savings for each household, after which it ranks each household based on the advantage value from low to high. A low (negative) advantage value means a high saving. The households that have the highest expected savings are selected first when a DR event is triggered, after which additional (less economic viable households) are activated one-by-one if the current set of households does not suffice in the DR-response.

- The continuous sending of boiler commands, so that the actual consumption approaches the target consumption as closely as possible. A proportional integral (PI) controller is used to control a discrete value that determines the number of participating boilers and the magnitude of the actions.
- Monitoring the observed boiler values on an online platform. This platform also shows an indication of how much flexibility is being 'used' at any moment of time during a DR event, by measuring how many households are activated.

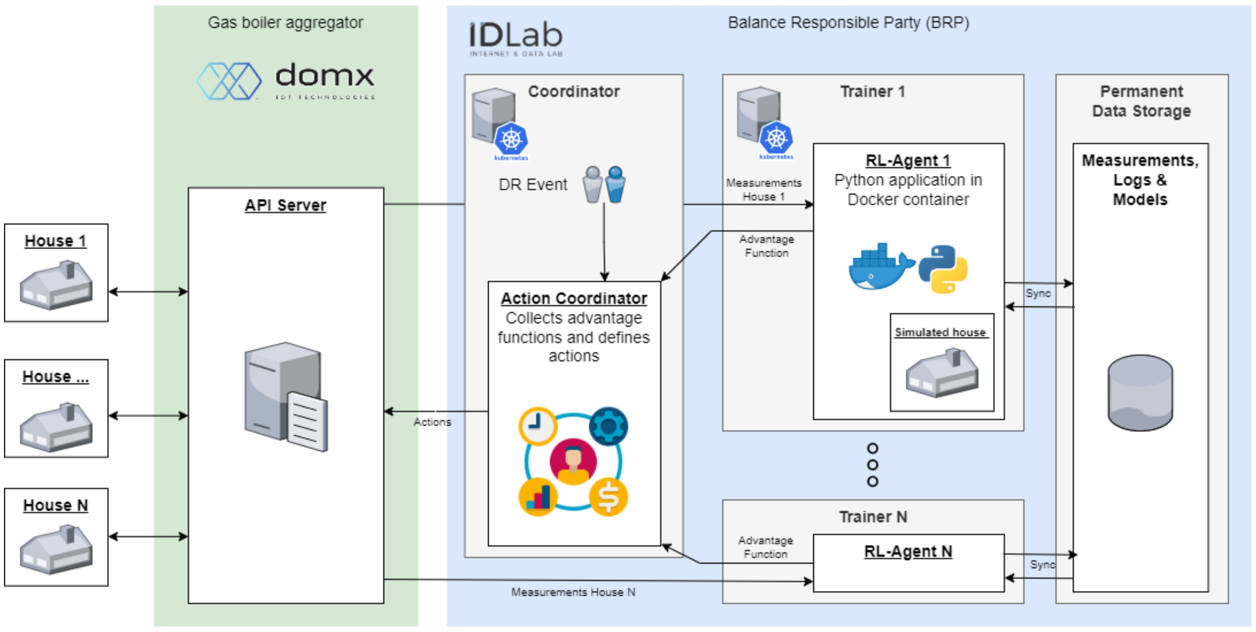


Figure 15 - Application A6 algorithm flowchart

Reference to GitHub: https://github.com/biggproject/biggpy/tree/main/gas_demand_response

IV. Integration within the BIGG Data Model

IV.1. Definition of the BIGG Data Model

IV.1.1. BIGG Ontology

As outlined in D4.1 (Stoyan Danov (CIMNE), 2022), ontologies form a crucial component of the semantic web approach defined by the World Wide Web Consortium (W3C, 2015). Ontologies are a formal and explicit specification of knowledge in a particular domain expressed in both human-readable and machine-readable formats, facilitating their sharing across different environments. They define classes, properties, relations between classes, and taxonomies, and can be combined to cover complex domains. Many ontologies are published as Open Data resources, based on international standards, and aligned with each other. The BIGG ontology is used to fully conceptualize the knowledge required to support BIGG Use Cases in the building domain and align this knowledge with concepts from existing ontologies and data standards, enabling independent sharing irrespective of specific implementations.

IV.1.2. BIGG Data Model

The BIGG Data Model, as described in Figure 16, is a comprehensive and detailed representation of the data structure, containing UML diagram, class descriptions, attribute information, and relationships that can be utilized for mapping to external database schemas. The main objective of this model is to enable the sharing of data across various systems and platforms, and to provide a standardized framework for data integration from different sources. The model follows the principles of data standardization and interoperability, and provides a common language and structure for data. It includes critical components such as building attributes, energy consumption, weather data, and occupancy information, and establishes logical connections between them, thus facilitating the management, analysis, and interpretation of data related to building energy consumption through harmonized data.

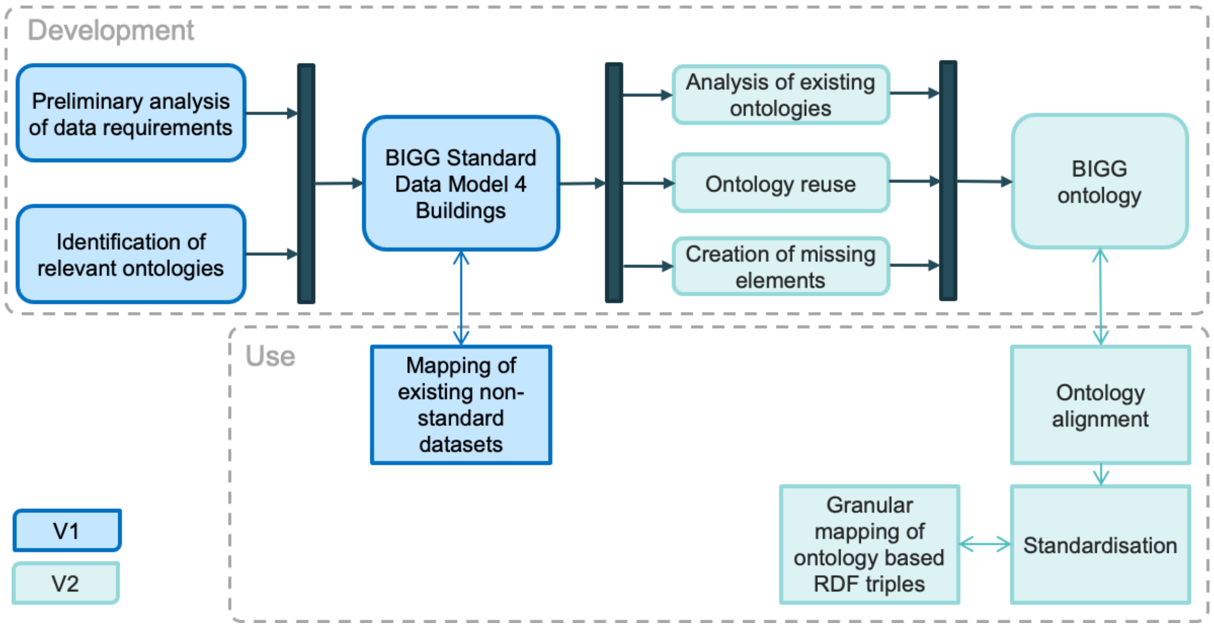

Development
Preliminary analysis of data requirements
Identification of relevant ontologies
BIGG Standard Data Model 4 Buildings
Analysis of existing ontologies
Ontology reuse
Creation of missing elements
BIGG ontology
Use
Mapping of existing non-standard datasets
Ontology alignment
Granular mapping of ontology based RDF triples
Standardisation
V1
V2

Figure 16 - **BIGG data model development and use**

For additional information, please refer to the existing Description of the preliminary Harmonized layer as described into Deliverable D4.1.

IV.1.3. Necessity of Harmonized Data

The adoption of digital technologies and the integration of sensors and controllers in buildings offer great potential for increasing efficiency in both new and existing buildings. However, the lack of standardization and harmonization of data definitions across applications and databases makes it difficult to exchange, compare, and combine data, hindering the potential benefits.

The BIGG project aims to overcome this obstacle by developing an open-source Big Data Reference Architecture and AI Analytics Toolbox (AITB) with demonstrated capabilities to support a variety of use cases and applications covering the whole building life cycle.

The project's success depends on harmonizing data across different systems and platforms through semantic technologies, such as ontologies, that encode the meaning of data and their interrelations in a machine-processable form. Reuse and alignment of existing ontologies are essential good practices for ontology development to establish relations between different ontologies while preserving the originals.

In the context of the AITB, pipelines intend to process data and extract insights or predictions related to building energy consumption. The integration of pipelines with harmonized data is a key element of this process, as it ensures that the data being processed is standardized and can be easily used across different applications.

IV.2. Integration of Pipelines within the BIGG Data Model

IV.2.1. Data Harmonization layer

To ensure that pipelines use harmonized data as inputs, the input data must be formatted according to the BIGG data model. Each function block within the pipeline is designed to work with the harmonized data format, which ensures that the data is consistent and compatible throughout the entire pipeline.

By standardizing the data format using the BIGG Standard Data Model for Buildings, the harmonization layer sketched in Figure 17 ensures that data from different sources can be brought together and made comparable. This makes it possible to integrate the developed tools with legacy systems and technologies, as well as with new developments, without having to worry about compatibility issues.

Overall, the harmonization layer components support the integration of AI Toolbox for Buildings (AITB) Business Cases by enabling the alignment, harmonization, and standardization of data from different sources, making it easier to extract insights and predictions related to building energy consumption, and ensuring that the data is consistent and compatible throughout the entire pipeline.

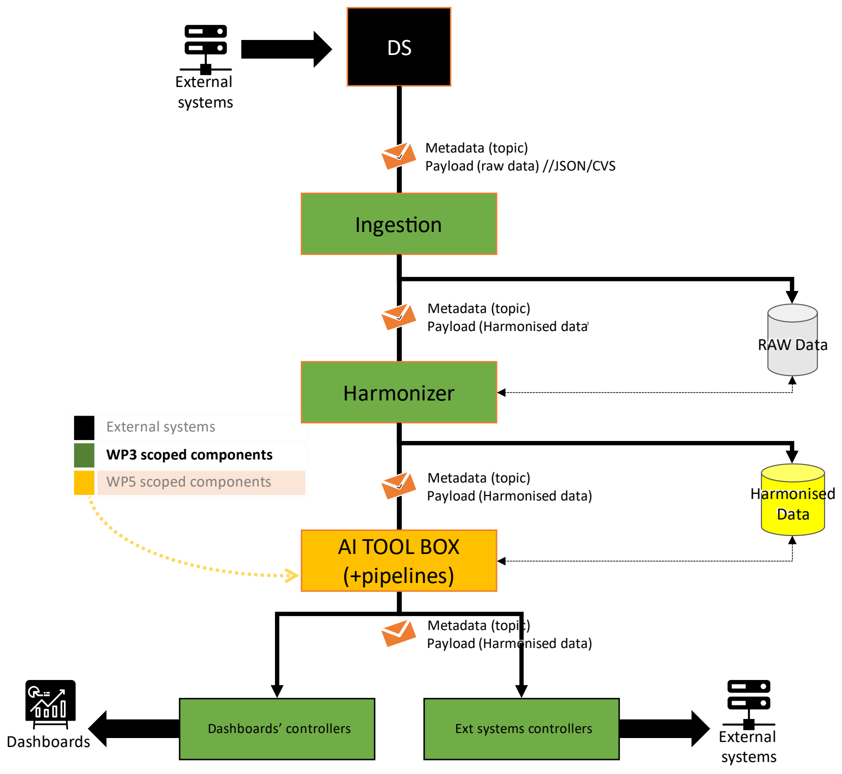


Figure 17 - Data Harmonization layer within the AITB

IV.2.2. Example of data harmonization with Application A1

IV.2.2.a. Use of BIGG Standard Data Model 4 Buildings

The data harmonization in BIGG aims to achieve semantic interoperability, allowing for both standardized and non-standardized data to be used with the BIGG Reference Architecture Framework (RAF) and AITB in various applications. The development of the BIGG Standard Data Model 4 Buildings is the first step in achieving the harmonization layer, followed by the development of a Mapping Tool and a Harmonization Component. The development methodology aims to align the BIGG data model with existing ontologies through ontological engineering while ensuring optimal data storage, processing, and retrieval for operational efficiency in the implementation of the BIGG RAF technologies for big data analytics in buildings.

IV.2.2.a. Preliminary task - Analysis of the Use Case

The analysis of use cases in the harmonization layer involves an ontology engineering approach, which starts with an analysis of the data that needs to be encoded with meaning. This analysis includes identifying the characteristics of the data, such as their value range and the domain concepts they describe.

A data schema is then defined, which includes the structural organization and logical relations among the entities involved.

Next, a model is created that encapsulates the needed concepts and their relations. Existing ontologies that match the chosen concepts are then surveyed and a selection process is conducted to integrate the appropriate external ontology terms.

The application ontology is then completed by adding any missing classes and properties according to existing naming conventions. This process ensures that the harmonization layer can handle both standardized and non-standardized data and enables semantic interoperability within the BIGG Reference Architecture Framework.

IV.2.2.b. Mapping of Application A1 datasets

IV.2.2.b.1. Data sources

During the initial phase, the data model served as a reference for mapping available data sources to enable the development and testing of the first version of the pilot solutions and the AI Analytics Toolbox. The mapping process had two objectives, namely, to associate the attributes of the data source with those of the standard data model and to validate the BIGG data model and identify any missing attributes.

The mapping was performed using an Excel template that contained the description of the BIGG data model's classes, attributes, and taxonomies, along with a set of rules for mapping the external data sources. The rules were designed to capture the structural relationships between the external data sources and the BIGG data model and allow for interpretation.

Additionally, information about the format of the source data, the way they are available, and the version and date of the mapping were collected. The mapping distinguished between static data and time-series data, which were mapped on separate sheets of the template to capture the differences in relationships between the classes in BIGG and the external datasets.

IV.2.2.b.2. Time series

Harmonizing time series data means mapping the attributes of the input data to the corresponding attributes of the BIGG data model, which is represented in columns on a sheet. The colour code indicates whether a particular attribute is mandatory, desired, or optional.

To map time series data successfully, each row of the time series input was mapped implicitly or explicitly using the colour keys provided. Enumeration type attributes that match those in the BIGG data model were mapped explicitly using the attribute source name. In contrast, implicit mapping was used when input attributes include information about other attributes implicitly. For example, units are often implicit within time series data because the property measured has a commonly known/used unit of measurement, or when data providers know it without putting it explicitly. The harmonized data is then stored in the harmonized data layer for use by AITB.

IV.2.2.b.3. Outputs

The output data from AITB was structured and formatted in a specific way to ensure it is harmonized with the BIGG data model. The data was organized according to the defined entities in the BIGG data model, which includes Building, Consumption, and Energy Performance Indicators.

IV.2.2.b.1. Harmonization module

In order to enhance data interoperability and streamline the mapping process of Application datasets, a dedicated Harmonization module is currently under development as part of the WP3/4 activities. This module aims to provide a comprehensive solution for harmonizing diverse datasets used within the use case. By integrating this module into the Use Case pipelines, the project will benefit from a standardized approach to data harmonization, ensuring consistency and compatibility across different datasets. This will facilitate seamless data integration and enable efficient analysis and processing of the data, ultimately contributing to the successful implementation of Applications and the achievement of overall project objectives.

CONCLUSION

The development of the AI Toolbox for Buildings (AITB) represents a major step forward in achieving energy efficiency and sustainability goals in the building sector. The preliminary version of the AITB was previously described in Deliverable D5.1 (Helexia, 2022), while this updated version builds on that work by focusing on updating and ordering the function blocks per business cases and its integration within the BIGG Data Model. This has resulted in a more efficient and streamlined tool that can be used to predict energy usage, identify areas for energy savings, and optimize energy performance in buildings quickly and accurately.

The practical applications of the AITB in real-world scenarios are described in this document, demonstrating its usefulness and applicability in a variety of contexts. The AITB has been designed to be scalable, enabling users to work with large and complex datasets and easily adapt the AI pipelines to new applications. This allows building managers, Energy Service Companies (ESCOs), energy consultants, data scientists, and researchers to benefit from the toolbox's ability to accurately predict energy usage and identify areas for energy savings.

In addition to its powerful predictive capabilities, the AITB now includes harmonization capabilities that allow users to work with datasets harmonized to the BIGG Ontology. This makes it easier to share data among different stakeholders and facilitates data analysis, enabling the development of new AI algorithms and the delivery of AI solutions to specific energy-saving problems. The BIGG Ontology has been designed to be coherent with other existing data models, which enhances data interoperability and sets the foundation for a high level of data sharing and collaboration. The maintenance of the referenced GitHub repositories ensures continued support and availability of resources for the foreseeable future.

In conclusion, the development of the AITB, together with its harmonization capabilities and practical applications on Business Cases, is a significant contribution to achieving energy efficiency and sustainability goals in the building sector. Its powerful and flexible tools provide a scalable solution that can benefit a wide range of users, making it a key part of the BIGG project's efforts to create more sustainable buildings. The AITB represents a major step forward in the use of AI to optimize energy consumption in buildings, and its potential impact on the building sector is significant.

REFERENCES

CSTB. (2022). *BIGG Ontology.* Retrieved from https://github.com/biggproject/Ontology/tree/version1.0

Develder, C., Claessens, B., & Gokhale, G. (2022). *PhysQ: A Physics Informed Reinforcement Learning Framework for Building Control.* Retrieved from Cornell University: https://arxiv.org/abs/2211.11830

Helexia. (2022). *D5.1 - Initial Description of the BIGG Artificial intelligence toolbox.*

INETUM REALDOLMEN BELGIUM. (n.d.). *BIGG project proposal, Proposal-SEP-210648186, Project-P164496-002.*

Josep M. Granollers (ICAEN. (2021). *D6.1 – Detailed description of pilot's technical assets: ICT tools and accessibility to data set.*

Oriol Escursell Jové (ICAEN. (2022). *D6.2 - First evaluation of the BIGG pilots results on use cases.*

Stoyan Danov (CIMNE). (2022). *D4.1 Description of the preliminary harmonization layer.*