# Intelligent Assistants for Flexibility Management
# (Grant Agreement No 957670)

# D3.2 Revised hybrid-modelling module

# Date: 2022-05-31

# Version 1.0

**Published by the iFLEX Consortium**

**Dissemination Level: PU - Public**

# Document control page

| | |
|---|---|
| **Document file:** | D3_2__Revised_hybrid_modelling_module_V1.0.docx |
| **Document version:** | 1.0 |
| **Document owner:** | **VTT** |
| | |
| **Work package:** | WP3 Artificial Intelligence for forecasting and automated flexibility management |
| **Deliverable type:** | DEM – Demonstrator, pilot, prototype |
| | |
| **Document status:** | ☒ Approved by the document owner for internal review |
| | ☒ Approved for submission to the EC |

# Document history:

| Version | Author(s) | Date | Summary of changes made |
|---|---|---|---|
| 0.1 | Jussi Kiljander, Janne Takalo-Mattila (VTT) | 2022-05-20 | Initial ToC |
| 0.4 | Jussi Kiljander, Janne Takalo-Mattila, Dušan Gabrijelčič | 2022-06-13 | Updated section 4, updated automated modelling pipeline, initial modelling method for apartment building has been updated (approach 1) and new approach presented (approach 2) |
| 0.9 | Janne Takalo-Mattila, | 2022-06-23 | Finalized for the internal review |
| 1.0 | Jussi Kiljander | 2022-08-02 | Improved based on internal review. Ready for EC submission. |

# Internal review history:

| Reviewed by | Date | Summary of comments |
|---|---|---|
| Christos Krasopoulos | 2022-6-22 | Accepted with minor text suggestions and comments. |
| Poria Divshali | 2022-6-22 | Accepted with minor changes. |

# Index:

# 1   Executive summary

This deliverable documents the digital twin repository module of the iFLEX Framework for phase one and work done related to project phase 2. This deliverable updates the status of the work from *D3.1 Initial Hybrid modelling module*. The digital twin repository consists of models to forecast and simulate consumer's baseline load, flexibility and response to flexibility signals. The models in the digital twin repository can be used to evaluate the impact of individual consumers, as well as, to optimize flexibility management on a consumer level. The modelling methodologies studied in the work combine advanced machine learning methods with physics-based modelling and expert knowledge in an innovative way. Moreover, we study novel applications of federated learning in the context of modelling individual prosumers.

The digital twin repository documented in this deliverable consists of digital twins for an apartment building and a household. For the apartment building, different approaches to build digital twins have been tested. The apartment building's digital twins are based on a hybrid-modelling approach that combines machine learning with physics and expert knowledge. The first digital twin consists of three types of models: two machine learning models for electricity and district heating baseline load forecasting, and a simple physics-based grey-box model for estimating the flexibility of a building's heating system. Different Artificial Neural Network architectures and models were implemented, trained and evaluated in order to find an appropriate model for baseline load forecasting (both electricity and district heating). The heating flexibility model consists of an indoor temperature model and energy consumption models for space heating, provided by a heat pump and district heating. The second type of digital twin for the apartment building utilizes machine learning both for electricity and district heating baseline forecasting. In this approach, machine learning is also used to estimate the reaction for the demand response control signals and a physics-based model is used for the indoor temperature model to calculate the flexibility capacity.

The household digital twin is based on four types of models: a household thermal model, an energy consumption model, a flexibility model and an occupant activity model. The initial baseline of the models is explained and presented. The data used for the modelling is introduced. The models consist partly of neural networks and partly of gradient boosting based solutions. Potential use and further improvements in next phases of the project are also provided. In addition, federated learning has been introduced and a simple example of its usage is presented. Further application of federated approach to the addressed household models will be evaluated already in the first piloting phase.

This deliverable reports intermediate results on the model evaluation. This deliverable also documents the initial results of the work that was required for the pre-pilot and is required for phase 2 pilots. Updates and improvements to the approaches, methods and implementations will be documented in *D3.3: Final Hybrid-modelling module*.

## 2    Introduction

### 2.1    Purpose, context and scope

This document is the second of the three revisions (D3.1, D3.2 and D3.3). The first revision D3.1 documented the digital twins used in the pre-pilot phase. This deliverable documents the intermediate results of task 3.1 - Digital twin of the consumer and task 3.2 - Hybrid-models for energy system modelling. The role of these tasks is to create a digital twin of a consumer and his energy systems (i.e., building, HVAC, renewables, appliances, etc.) in order to forecast baseline loads, flexibility and response of the consumer. For brevity reasons, the terms consumer and prosumer are used for the whole metering point, including the persons and infrastructure. The digital twins documented in this deliverable form a digital twin repository that can be used to instantiate models for different situations. In particular, the digital twins are used for the following main purposes:

- To forecast and evaluate the flexibility impact of the consumer in order to provide individual incentives and rewards.

- To perform model-based planning and control that adapts to the consumer behaviour and optimizes flexibility with respect to external prices and incentives.

Two types of consumers are considered so far: households and building communities. A household type consumer is a family comprising of one or more people. A building community consists of several families that live in an apartment building. The space heating, domestic hot water (DHW), and common electric consumption (e.g. ventilation, elevators, lighting, Sauna) of the apartment building are paid collectively by the building community (the costs are typically included in the rent or in the maintenance fees).

### 2.2    Content and structure

The deliverable is structured as follows:

- Section 3 provides an overview, mapping the contents of the deliverable to the use cases and the iFLEX architecture.

- Section 4 introduces the main methods and approaches applied in the work.

- Section 5 describes the digital twin repository implementation, as well as details on the approaches applied in different models of the digital twins.

- Section 6 concludes the deliverable.

### 2.3    Main changes compared to the initial version

- New requirements have been added to 3.1 based on the goals of the second phase of the piloting

- Section 4 has been updated with more details on the hybrid modelling methodologies studied in the project.

- Section 5.2 describes a new automated modelling pipeline used by the digital twin repository which enables creating digital twins as automated as possible

- The initial hybrid modelling approach has been updated. The idea is still the same but the model implementation has been redone based on the results and feedback collected in phase 1. Furthermore the documentation (in section 5.3.1.1) has undergone a major revision.

- Section 5.3.1.2 describes a new approach to model the district heating and electricity power during and after demand response events using machine learning

# 3 Overview

## 3.1 Relation to use cases

The project has defined a number of use cases in the D2.1 - Use cases and requirements. Some of the use cases are directly related to the work described in this deliverable and in task T3.1. The following requirements were prepared for the first phase of the deliverable, while selection has been influenced by the goals of the first phase piloting:

- IF-62: Household thermal model, related to the use case PUC-5
- IF-63: Household electricity model, related to use cases HLUC-1, PUC-4, PUC-6, PUC-8, PUC-10
- IF-64: Household flexibility model, related to use cases PUC-8, PUC-4, PUC-5, PUC-6, PUC-10
- IF-65: Household occupant flexibility model, related to use cases PUC-4, PUC-5, PUC-6, PUC-8, PUC-10
- IF-66: Apartment building district heating model, related to use cases HLUC-3, PUC-8, PUC-10
- IF-67: Apartment building electricity model, related to use cases HLUC- 3, PUC-8, PUC-10
- IF-68: Apartment building flexibility model, related to use cases HLUC-3, PUC-6, PUC-8, PUC-9 and PUC-10.

The following requirements were added for the second phase of the deliverable, where selection is again based on the goals of second phase piloting:

- IF-105: Data collection from demand response tests, related to the use case PUC-8

- IF-106: Machine learning based apartment building district heating and electricity flexibility models: Related to the use case PUC-8

The requirements are managed in the project Jira service, as is presented in Figure 1. Within Jira, the requirements are described, prioritised and developed. The current status of the requirements is given in Appendix 9 of this deliverable.
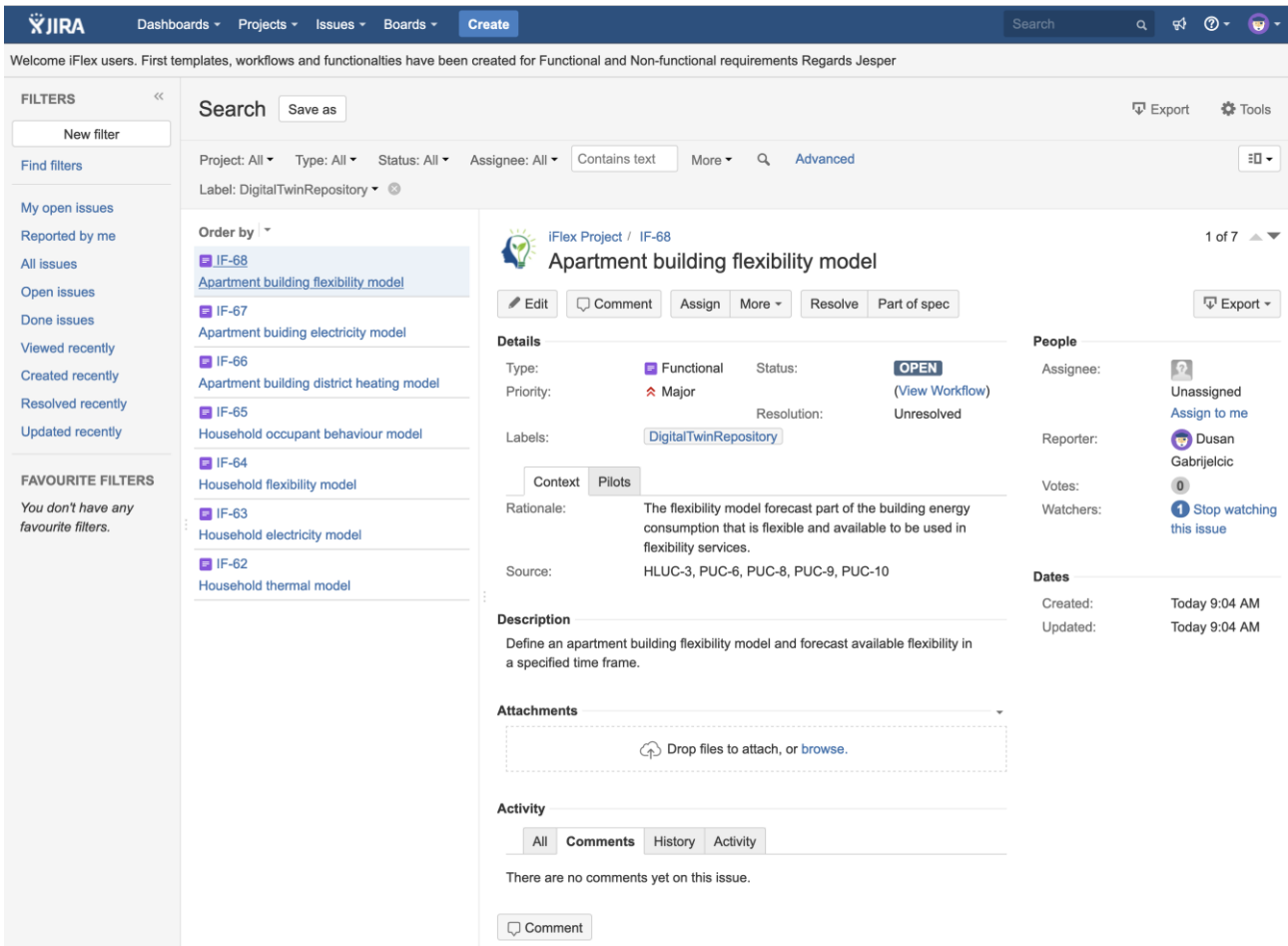
Figure 1: Digital twin requirements captured in the project Jira

## 3.2 Relation to the functional architecture of the iFLEX Framework

Figure 2 highlights the Digital twin repository module, whose initial implementation is documented in this deliverable. The Digital twin repository module is capable of forecasting and simulating the baseline loads, flexibility and response of the consumer/prosumer. This functionality is utilized by the Automated flexibility management module (revised version to be documented in *D3.8 - Revised Automated flexibility management module*) for model-based planning and control.
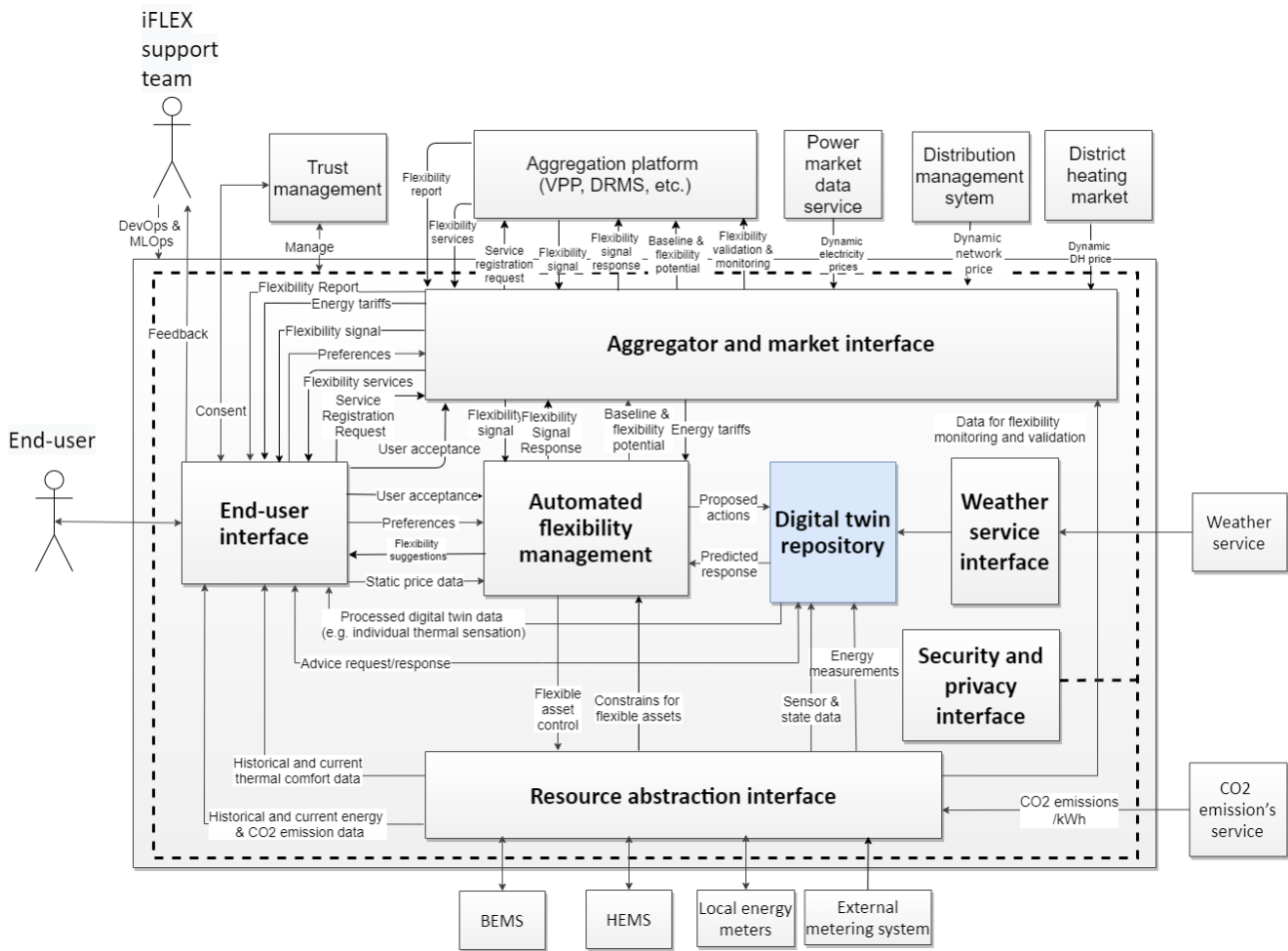
Figure 2: Functional view of the iFLEX Framework with the Digital twin repository module highlighted.

## 3.3    Focus of the project phases

The implementation of the Digital twin repository module is targeted for phase 1 and phase 2 pilot deployments, specified in *D7.2 – Revised Pilot specifications*. The goal in phase two is to utilize the initial methods and models from phase 1 (pre-pilot), fine-tune and finalize model, improve the modelling with new hybrid approaches and build new functionalities on top of those.

There are two types of digital twins that were identified already in the first phase of the project and are now being extended: building community and household. For the first phase, the following models were developed for the digital twin of the building community: electricity baseline model, district heating baseline model, and flexibility model. Relatively standard Artificial Neural Network models were developed to forecast the baselines. For the flexibility modelling, two different approaches are tested in  phase 2. In the first approach, a simple physics-based grey-box model was developed to model flexibility provided by the building's space heating (i.e., indoor temperature, and energy used for space heating). In the second approach, a physics-based model was used for indoor temperature modelling to estimate flexibility capacity and machine learning was used to model energy consumption in different scenarios.

# 4   Methodology and approach

The models developed in this work package form a digital twin of the consumer that can be used to forecast and optimize consumer flexibility management. The digital twin of the consumer consists of models to forecast consumer energy loads, flexibility and the response of flexible assets with respect to various control inputs. Both the behaviour of people and the dynamics of buildings need to be modelled by the digital twin repository. Depending on the measurement infrastructure, these aspects are also typically highly interlinked and therefore modelled in the same digital twin in these situations. On the building side, the focus is both on detached houses and apartment buildings.

In order to realize the digital twin repository of consumers, data-efficient, robust and adaptive methods are studied and developed to model consumers and flexible assets availability at the consumer's premises. The main innovative concepts to be studied in the project include: hybrid-modelling approach that combines artificial neural networks with physics-based models, utilization of transfer learning to improve the data-efficiency, and utilization of federated learning to provide a secure and privacy aware solution for the models' implementation. This deliverable presents the intermediate results of the work, as well as directions and ideas for future development. Some of the approaches and methods, including transfer learning, will be introduced in the final version of this deliverable.

## 4.1   Machine learning

Machine learning (ML) approaches grew out of and alongside the development of artificial intelligence in 1960s. Technological advances in computer hardware and metering allowed the development and application of ML techniques to various fields of scientific study. As the name implies, ML algorithms allow the computer to solve a problem without explicit and exact programming, but rather though learning from the sample data. There are two types of learning: supervised and unsupervised. The former employs learning on sample data with known outcomes, while the latter works with sample data as is. These two approaches of learning from data broadly classify various ML techniques. A mix of these two main types gives other learning approaches, mainly reinforced learning, semi-supervised learning, and dimension reduction, among others.

The main requirement for ML methods, regardless of the learning approach, is having a large enough sample data set to learn from. Sample data is also called training data. In the case of supervised learning, the result of each record of the training data is known or expected, which helps to shape the algorithm's optimization function. Supervised learning approaches work well with classification and regression problems. On the other hand, unsupervised learning finds the structure of the unlabelled training data and uses it for clustering and classification. Based on the problem domain and the nature of the training data, various ML models were developed, such as Artificial Neural Networks (ANNs), decision trees, support vector machines, Bayesian networks, and genetic algorithms. Prevalent among those are the ANN, whose type varies (e.g. Feed Forward, Radial Basis Network, Recurrent Networks, Autoencoder) and applications (e.g. pattern recognition in visual, auditory, and text data, biological classification, genomics pattern finding or time-series data forecasting). Artificial neural networks imitate the architecture of the human brain and the functions of the biological neurons. When the topology of the network includes several specifically interconnected layers, we can apply the so-called deep learning approach, which enables the modelling of highly complex, non-linear, heterogeneous data, as well as data with a temporal component (Nielsen, 2020).

A special type of ANN is called Long-Short Term Memory (LSTM) network. It is a deep-learning recurrent neural network that can utilize long term time dependencies. As in any ANN, there are three distinctive layers: input, hidden, and output layer. Input and output layers serve as a representation of input and output values. One or more hidden layers serve to optimize the classification or regression function in the parameter space. Each neuron in one layer, except for the input layer, is fully connected with all others from the previous layer. Each connection has a weight, which is constantly updated during the optimization process. The optimization algorithm during the training typically searches for minimum prediction or classification errors according to the objective (loss) function. In other words, during the training of the network, candidates for weight values are determined with a stochastic gradient descent algorithm and then updated using a backpropagation of error algorithm. Error is evaluated by the objective function.

Among reasons for using LSTM networks for energy load prediction are the inert long term (seasonality) and short-term (daily patterns) of load data in the forecast model. This characteristic also enables flexibility in the forecast horizon from day-ahead to week or month-ahead predictions. The largest amount of the work was done for short-term and really short-term predictions with application of various statistical techniques (multiple regression method, exponential smoothing, stochastic time series), artificial intelligent techniques, knowledge

based expert systems, and hybrid techniques (Srivastava, Pandey, & Singh, 2016). Comparatively to other univariate load prediction methods: SARIMA (Chakhchoukh, Panciatici, & Mili, 2011), NARX, SVR (Ko & Lee, 2013), and NNETAR (Hyndman & Athanasopoulos, 2018), the LSTM network had a better forecast accuracy on the data of building consumption at the 15-minute interval (Zheng, Xu, Zhang, & Li, 2017).

Robust approximations of load consumption patterns using a combination of deep-ANN topologies with the probabilistic baseline load estimation (Ovdes, Souvent Ovdes, & Ovdes, 2020), will help iFLEX Assistant to evaluate the flexibility potential for individual users, in the context of their digital twin, and will allow suggesting of the effective spread of user's energy demand based on their own limitation and preferences, price signals and flexibility demands from the network.

## 4.2   Combining machine learning with physics-based modelling

Artificial neural networks are universal function approximators, meaning that they can in theory model any function. Deep ANNs (i.e., deep learning) have also been very successful in many practical applications ranging from perception (Krizhevsky, Sutskever, & Hinton, 2017) and natural language processing (Graves, Mohamed, & Hinton, 2013) to protein folding (OpenAI, 2020). Despite their success, deep learning has also limitations. For instance, they typically require a lot of data to provide good results. Additionally, even with large amounts of data, deep learning models have challenges to generalize beyond training data distribution (Bengio et al., 2019). A key idea in iFLEX is to utilize AI technologies to form accurate digital twins of consumers (persons and infrastructure) which can be used to predict the response of the system with respect to a different type of inputs (HVAC setpoints, flexibility signals, etc.). In this situation, it is not possible to cover all possible situations in the training data, which is a challenge for ANNs and other types of pure data-based approaches.

An alternative approach to model energy systems is to utilize physics. Physics-based models are useful in situations where there is a limited amount of data on a given system, while the fundamental physics behind the system are known. For instance, it would not be practical to collect a vast amount of data on how a building responds to different temperature setpoints, which would be required to train a pure ANN model to forecast the flexibility provided by the building's thermal mass. To this end, the approach utilized in iFLEX is based on combining physics-based modelling with ANNs. This type of combination of symbolic logic with ANN models is referred to as Neuro-Symbolic AI (or neurosymbolic AI) in the literature (d'Avila Garcez & Lamb, 2020) and it is seen as the third wave of AI.

There are many ways to combine physics-based modelling with ANNs. For example, Koponen *et al.* (Koponen, Niska, & Mutanen, 2019) studied various combination approaches, including residential hybrid, constraining models, physically based input forecasts, and ensemble forecasting. Another interesting way to combine ANNs with physics-based modelling is to utilize physics-based simulation models to generate training data for an ANN model (De Wilde et al., 2013)(Chou & Bui, 2014)(Turhan, Kazanasmaz, Uygun, Ekmen, & Akkurt, 2014). This type of approach with an innovative twist is also investigated in the project. The initial results of the work are published in the paper (Kannari et al., 2021). The key idea in the novel approach is that we use simulated data from a large pool of buildings to train a single ANN model (in contrast to training building specific ML models as in previous research). Our hypothesis is that this forces the ML models learn to approximate the physics of encoded into the simulation model to generalize to new buildings. Another advantage of this approach is that it makes it possible to forecast buildings energy demand just based on building's characteristics and thus makes it possible to use ML models in situations with no energy consumption data of a specific building. Furthermore, the main benefit of the proposed approach is that once the ANN model has been trained, we can do not need the original simulator. This is important since it makes it possible to truly combine the benefits of ML with physics-based modelling. The approach seems promising but requires still more work to be deployed into the pilot buildings at operational environment (TRL 7). We will continue the work on this and investigate the possibility to include these approaches for phase 3 deployment.

The approach implemented for phase 2 pilots is based on Neuro-Symbolic AI architecture that combines ANN models and physics-based models to create a digital twin of the consumer (including the building infrastructure). The models implemented in phase 2 are presented in more detail in section 5.

## 4.3   Federated learning

Classical machine learning is done by data scientists by collecting raw data from data owners and training their model on this data. Since they have access to data it is easy for data scientists to violate data owner's privacy.

Federated learning is a technique in machine learning which allows data owners not to reveal any data.

The idea is very simple: let data owners train our model on their data and then combine those models into a better one.
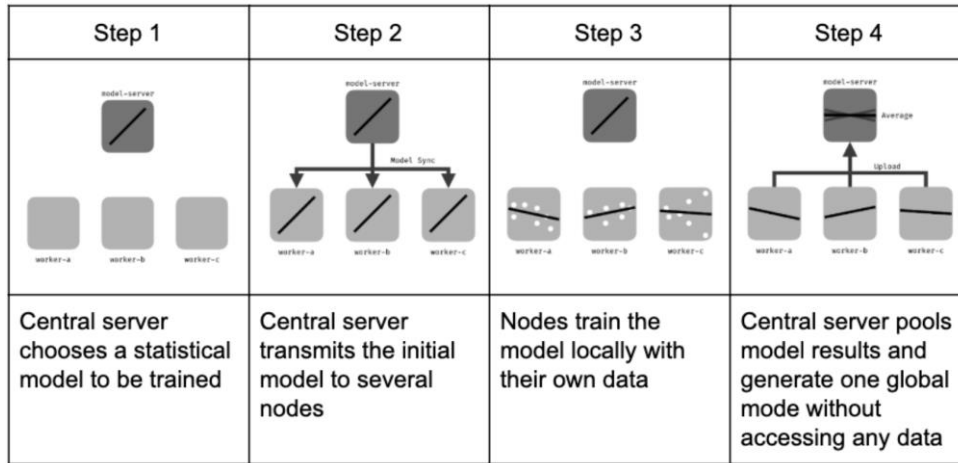


Figure 3: Federated learning basic concepts

### 4.3.1 Privacy concerns

A major concern and an obvious question to ask seems to be: is it possible to reverse engineer individual models to get access to some private data after it was trained and sent to us by data owners? Generally speaking, it's possible. To avoid this, an important idea from differential privacy can be used.

First, we need to understand how to combine different models into a new one in federated learning. It's simple - we just average over the corresponding weights. Averaging has an important and useful property illustrated in the following scenario.

Imagine having a web service and wanting to know the average age of users. One way to get the answer is to simply ask for their age and calculate the average value. But what if age was a sensitive piece of information? To avoid knowing exact ages, users could be asked to send in their age plus some (uniformly chosen) random number between -100 and 100 instead. Each such result would tell us little about the exact age of each user, but when summing them up, random numbers would cancel each other up and we'd be able to calculate the exact average value (provided there are enough users for statistics to work).

The same idea can be used in federated learning - some random noise is added to each weight by each individual model before sending it to the data scientist. When combining models, random numbers in weights cancel each other out and we are still able to get the same combined model as if there was no noise on each individual weight.

To sum up: by adding noise we make it practically impossible to reverse engineer private data from model, meanwhile preserving the ability to successfully combine models.

The federated learning with possible applications in some of the project-based solutions is further showcased and evaluated in Section 5.3.2.5.

# 5 Implementation of the digital twin repository

## 5.1 Overview

The *Digital twin repository* module documented in this deliverable consists of various models that can be used to predict the response of the consumption site in different situations. The models are implemented with Python programming language. Tensorflow 2.0 (Abadi et al., 2016) with Keras API (Chollet, 2018) is used for implementing the ANN models. The physics-based models are implemented mainly with NumPy (Harris et al., 2020). The Scikit-learn (Pedregosa et al., 2011) predictive data analytics tool is used for identifying some of the physical model parameters, as well as, for data clustering. Pandas (McKinney, 2010) is used for timeseries analysis and processing.

## 5.2 Automated machine learning pipeline

An automated machine learning (AutoML) (Feurer et al., 2015) pipeline is a toolset that can be used to automatically generate machine learning-based time-series forecasting models. High level of automation is required because the benefits obtained from single households or consumers are typically limited and therefore replication methods with minimal human effort are the key to create sustainable business in flexibility markets.

Developed pipeline can generate multivariate time series forecasting models that can use one or more time-dependent variables x to forecast future values. Toolset consists of reading and processing historical data, formatting it into training and validation sets, training the machine learning models and validating their performance, as seen in the center of Figure 4**Error! Reference source not found.**. Presented automated machine learning toolchain contains various ready-made feature engineering pipelines and new pipelines can be easily added. Data is read either directly from files, where the data is stored in multiple .csv files that are read and parsed together, or fetched from databases such as oBIX-store using Python-client presented in D4.2. Other components in the pipeline have also been developed using Python programming language. Various modelling technologies such as Scikit-Learn, ANN and Tensorflow are already supported in the pipeline and new methods can be easily added. Models created with automated machine learning pipeline can be used as a part of digital twins such as household digital twin and building community twin presented in more detail in the next sections.
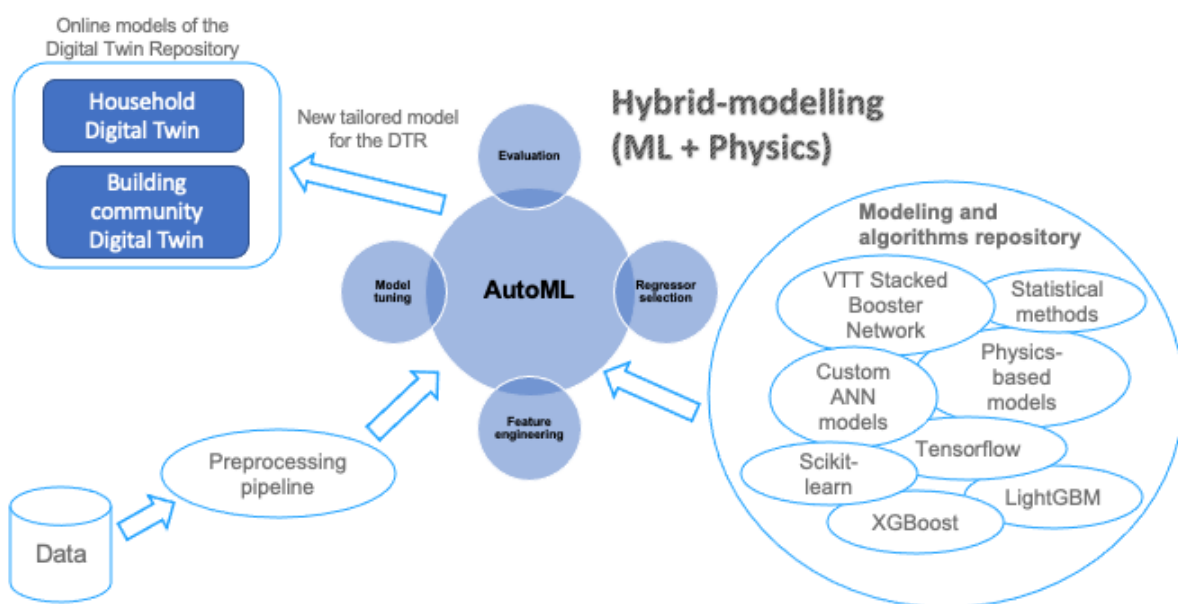


Figure 4. Automated machine learning pipeline.

### 5.2.1 Components of the automatic machine learning pipeline

Python-script named *train_eval_tool* is the main script for creating time-series. Train_eval_tool has three roles. First, train_eval_tool can train the models and save them into a folder defined by the project. Second, it can evaluate the trained models. Third, it can deploy the best model (in terms of model performance) into a separate folder named *best_models*. Models located in the *best_models* are the instances of *Forecaster class*, which is described in detail in the next sections.

Train_eval_tool is configured using *models_conf* configuration file. Configuration options include:

1) **Targets**: Target of the forecast. Typically, in this case it is either district heating power or electricity power, but the tool is not limited to these. Currently, only one target can be trained and evaluated at time.

2) **Forecast options**: Lead time describes how many timesteps (data frequency) are before the first forecast period. Forecast length defines the length of the forecast in data frequency timesteps. Update rate defines how often the new forecast is made. Currently models created with the pipeline update their forecast in each timestep. Lead time and forecast length can be freely selected. Forecast options are visualized in Figure 5.

3) **Features**: Features can be freely customized. However, automated machine learning pipeline already supports various ready-made features that can be utilized. *Lagged target* (i.e. target value in prior time steps) can be used as a feature. Pipeline makes it possible to use 0-N number of last target values. *Weekday* is a predefined feature that presents the day of the week in OneHotEncoded format. *Holiday* feature describes if a specific date is a government-designated holiday. *Temperature* feature presents the current outdoor temperature. *Hour* feature is current hour of the day. Finally, *control* feature describes if building heating level has been overridden either by the automatic controller or through manual control of the heating level. Figure 6 presents some example features in the dataframe.

4) **Pipelines:** Pipelines are a different kind of preprocessing and modeling pipelines Currently automatic machine learning pipeline supports the following pipelines: *TF-FFNN_single* (Tensorflow Feed Forward Neural Network) with a single input for all the input features, *TF-FFNN_multi* with separate inputs for each input feature, *MLP_single* (multilayer perceptron regressor) with a single input for all the input features, *MLP_multi* with separate inputs for each input feature, *SVR* (Support vector regression), *Linear* (Linear regression)

5) **Baselines:** Simple baseline models to compare trained models in the evaluation. Currently, pipeline supports *Mean* (Mean value of the values), *Lag_4* (using 4 previous target values) and *Lag_24* (using 24 previous target values)
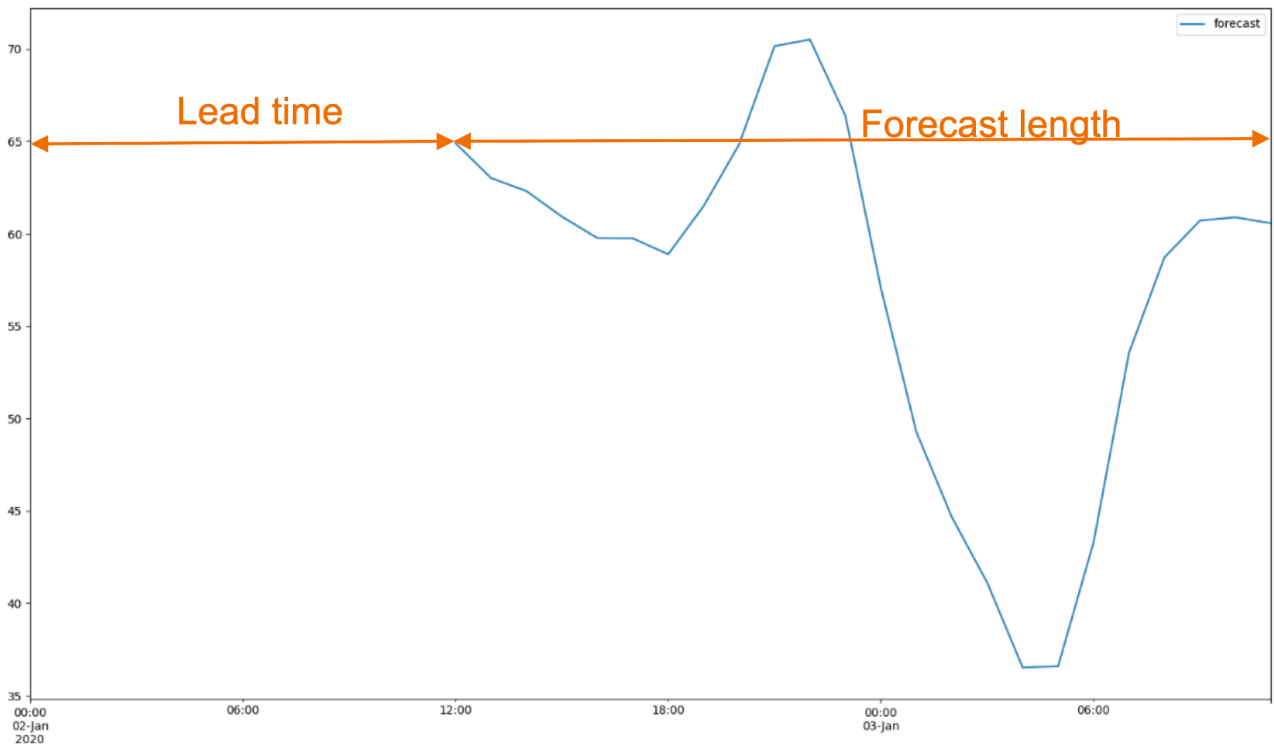
Figure 5: Time series forecasting terminology



Figure 6: Features in the dataframe.

Automatic forecasting pipeline produces *Forecaster class* that can be used in evaluation and online forecasting. It supports two different prediction output formats, namely multi-output format and single output format. Multi-output format is the default output format of the *Forecaster* class and it is implemented as a Pandas Dataframe. In this output format, each dataframe row is indexed with DatetimeIndex and contains multiple forecasts for the timestep. For example, using forecast length of 24 hours, each row contains 24 different forecasts made at different times. Figure 7 presents the multi-output format in Pandas dataframe. In this figure, rows are different timesteps and columns represent forecasts made at different times i.e. forecast_13 is a forecast made 13 hours ago.

Figure 7: Forecaster class multi-output format.

Single-output format is a *Forecaster* class output and is implemented as a Pandas Series and it is used for online forecasting. In this format, each row contains just a single forecast. Figure 8 presents the single-output format in Pandas series.



Figure 8: Forecaster class single output format.

## 5.3 Phase 2 models

The digital twin repository implemented consists of two types of digital twins: Digital twin of an apartment building community and a digital twin for a household. The building community digital twin, represented in section 5.3.1, consists of models for the following purposes: district heating demand forecasting, electricity demand forecasting, heating demand forecasting and indoor temperature modelling. The household digital twin is presented in section 5.3.2. It consists of prosumer load and flexibility forecasting models.

### 5.3.1 Digital twins for a building community

The Digital Twin of the building community is designed to model the baseline load profiles and flexibility of a building community (apartment building) that has a central heating and ventilation system providing flexibility for the consumer. The building has a hybrid heating system where the heat is provided with district heating (DH) and an exhaust air heat pump. This heating system supplies heat both for space heating and for heating of the domestic hot water. Only space heating is used for flexibility because we do not want to compromise

the comfort of the residents. Figure 9 depicts the mapping between the Digital Twin, measured (and non-measured) parameters and the residents of the building community.
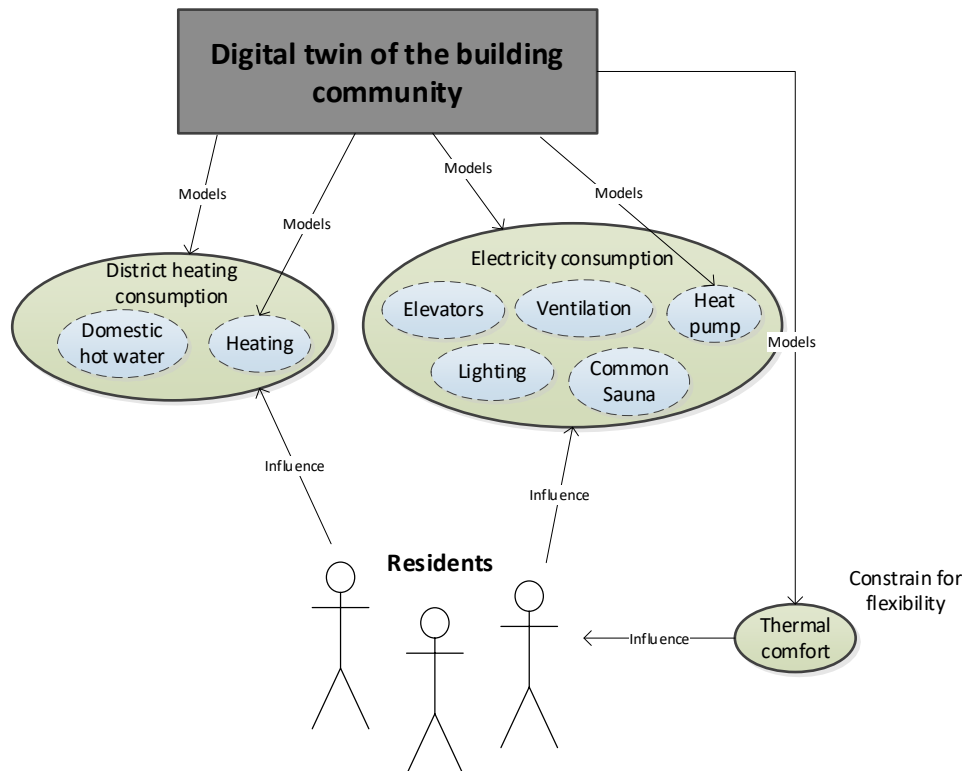


Figure 9: Conceptual representation of the interdependencies among the Digital Twin, measured parameters (green), non-measured parameters (blue) and the residents of the building community

The green colour indicates the parameters for which we have direct measurements available. As it can be seen, the total DH and electricity consumption are measured, as well as the average thermal comfort of the residents. The behaviour of the residents naturally effects both the DH (especially domestic hot water) and the electricity consumption (especially elevators and Sauna). The flexible asset to be modelled in phase 1, is the apartment heating, which is provided with a combination of DH and electricity (exhaust heat pump). The main constraint for the flexibility is the thermal comfort of the residents. The average temperature should not drop below 21.0 Celsius degree (short periods below 21.0 are allowed but should be corrected as soon as possible). Two different approaches for implementing the Digital Twin are studied in this phase of the project. Section 0 describes that approach 1, that is a minor update to the hybrid approach represented in the D3.1. In this approach machine learning is used for modelling the baseline consumption and physics-inspired hybrid models are used to modify the baseline energy consumption according to the law of energy conversion. Section 5.3.1.2 presents a new hybrid approach that utilises machine learning for modelling also the energy response to control signals. Both approaches use a simple physics-based hybrid model to model the indoor temperature during the demand response events.

In both approaches the Digital Twins use the same forecasting length, lead time, forecasting frequency and sampling rate parameters depicted below:

- Forecast length: 24 h
- Lead time: 0 min
- Data sampling rate: 60 min
- Forecast frequency: 60 min

5.3.1.1 **Approach 1**

The models of the building community Digital Twin in the approach 1 are presented in Figure 10. The Digital Twin consist of two main models: energy demand model and the indoor temperature model.
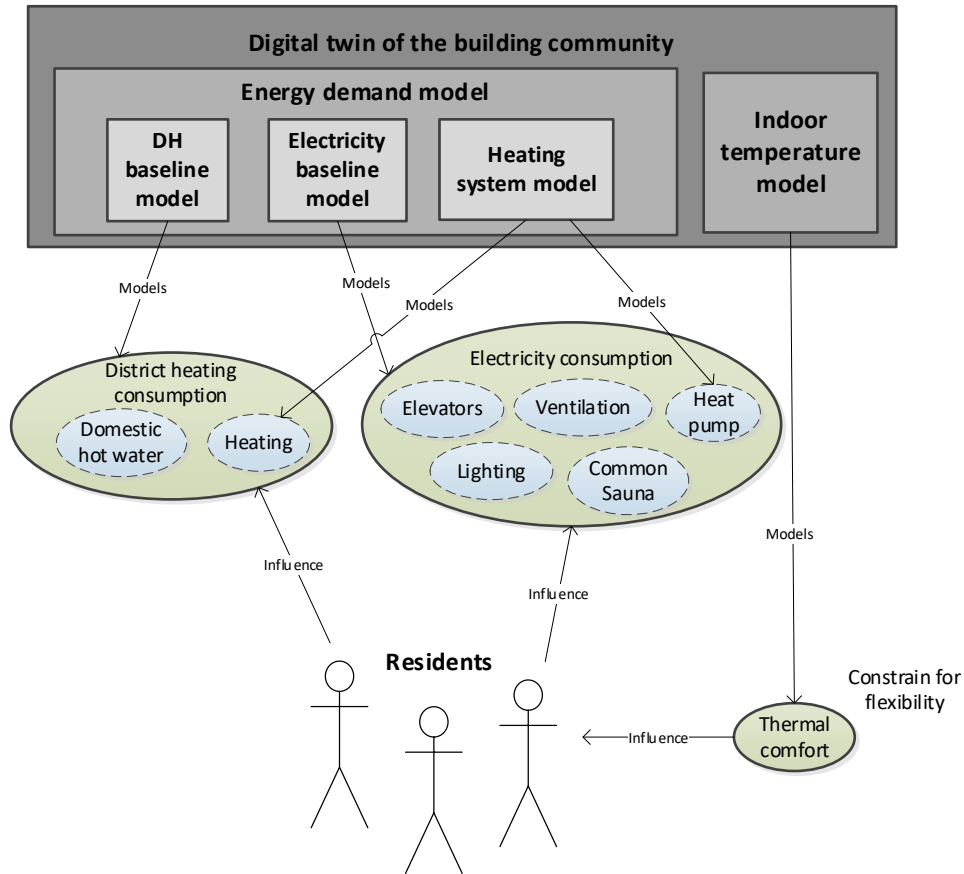
Figure 10: Conceptual view of the digital twin for the building community (approach 1).

The indoor temperature model predicts the average indoor temperature during demand response events. This model is relatively simple physics-based model with two parameters learned from data. The model is presented in more detail in section 5.3.1.1.1

The energy demand model produces electricity and district heating consumption forecast. The energy demand model is a hybrid model that consists of three main models: DH baseline model, electricity baseline model, and heating system model. The DH baseline and electricity baseline models are implemented purely with ML techniques. Section 5.3.1.1.2 describes the design and implementation of the models in more detail. The heating system model is a physics-inspired model with several parameters learned from data. This model is represented in section 5.3.1.1.3.

A formal representation of the energy demand model is presented in equations (1) and (2). Equation (1) represents the district heating output and equation (2) the electricity output.

$$E_{dh}(a) = \begin{cases} E_{dh_{base}} + Q_{pb}, & a = 0 \\ Q_{demand_{a=1}} - Q_{hp_{a=1}}, & a = 1 \end{cases} \tag{1}$$

$$E_{ele}(a) = \begin{cases} E_{ele_{base}}, & a = 0 \\ E_{ele_{base}} + E_{hp_{a=1}} - E_{hp_{a=0}}, & a = 1 \end{cases} \tag{2}$$

The $a$ is the control signal ($a = 1$ means heating is constrained, and $a = 0$ means normal operation). The $E_{dh_{base}}$ and $E_{ele_{base}}$ are the baseline energy consumptions provided by the machine learning models represented in section 5.3.1.1.2. The other parameters are provided by the heating system model. These parameters include:

- $Q_{pb}$: Heating payback representing the recoil effect (law of energy conversion) once the heating is returned to normal set point,

- $Q_{demand_{a=1}}$: Total heating demand (space heating + domestic hot water) when the heating is constrained.

- $Q_{hp_{a=1}}$: Heat pump energy production during the DR event (heating constrained).

- $E_{hp_{a=0}}$: Heat pump electricity consumption when the heating is not constrained.

- $E_{hp_{a=1}}$: Heat pump electricity consumption when the heating is constrained.

### 5.3.1.1.1  Indoor temperature model

The purpose of this model is to predict the indoor temperature when the amount of energy used for heating is limited. In the first phase, we assume that the heating energy is reduced to zero during DR events. This model thus predicts the temporal dimension of the flexibility (i.e., how many minutes can the heating be turned off). The modelling is based on applying Newton's law of cooling, represented in equation (3).

$$Q = hA(T - T_{env}) = hA\Delta T \tag{3}$$

where $Q$ is the rate of heat transfer out of the body (SI unit: watt), $h$ is the heat transfer coefficient (SI unit: W/m$^2$K), and $A$ is the heat transfer surface area (SI unit: m$^2$).

In the simple modelling approach, the building is treated as a lumped capacitance object with a uniform internal temperature $T$. As presented in equation (4), the internal energy $U$ stored into the building can be presented in terms of this uniform temperature, the heat capacitance $C$ of the building, and a reference temperature at which the internal energy is zero.

$$U = C(T - T_{ref}) \tag{4}$$

By differentiating $U$ with respect to time $t$ and applying the first law of thermodynamics we get:

$$\frac{dU}{dt} = C\frac{dT(t)}{dt} = -Q \tag{5}$$

The heat transfer $Q$ out of the building can be represented by Newton's law of cooling, and thus by combining (3) and (5), we get:

$$\frac{dT(t)}{dt} = -\frac{hA(T - T_{env})}{C} = -\frac{1}{\tau}(T - T_{env}) \tag{6}$$

The solution to this differential equation is:

$$T(t) = T_{env} + (T(0) - T_{env})e^{-t/\tau} \tag{7}$$

where $\tau = \frac{C}{hA} \Rightarrow \frac{C}{H}$ is the time constant of the building cooling. To calculate the time constant, we need to estimate the heat loss coefficient $H$ and the thermal mass $C$ of the building. It should be noted that with longer forecasts the outdoor temperature, $T_{env}$, can change. In this case the forecast for the indoor temperature needs to be executed in parts while updating the $T_{env}$ for each time period.

Section 5.1.1.5 describes how the heat loss coefficient and thermal capacitance parameters of the model are identified.

### 5.3.1.1.2  Baseline load forecasting models

The baseline load forecasting models consist of DH baseline and electricity baseline models. These total models are implemented with pure machine learning methods. This is because of two reasons. First, the behaviour of residents has a big impact on these total consumptions (e.g., how water demand, elevators, common sauna, also influences the space heating) which makes the ML methods a natural choice. Second, there is a large amount of hourly data available about the building's district heating and electricity consumption which makes it easy to develop accurate ML methods for these tasks. Root means square error (RMSE) is (equation 8) used as the error metric for evaluating the models. The results are also normalized with respect to minimum and maximum values, as formulated in equation (9) to provide a more easily interpretable percentual metric for the error.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{8}$$

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \tag{9}$$

Different machine learning models, including linear regression, ANN, and support vector regression were trained and evaluated for the given task with different hyperparameters and features using the automated modelling pipeline presented in section 5.2. A two-layer Artificial Neural Network with the following features was the best performing model in both tasks:

- Past energy consumption from the last week (input size: 168): Missing values in the data are replicated with mean value and the input is then scaled between 0, and 1.

- Weekday of the target period, represented separately for each hour (input size: 24). The input is one-hot encoded before.

- Hour of the target period, represented separately for each hour (input size: 24). The input is scaled between 0 and 1.

- Flag indicating weather, represented separately for each hour (input size: 24).

- Outdoor temperature for the target period (input size: 24). Weather forecast is utilized instead of measured values in the online version of the model. Missing values in the data are replicated with mean values and the input is then scaled between 0, and 1.

The average errors in the test set were 14.4 kW and 2.1 kW for DH and electricity respectively. The corresponding NRMSE errors were 7.8% and 7.6% for DH and electricity, respectively. Figure 11 and Figure 12 illustrate the electricity and district heating forecasts for sample periods during the pre-piloting phase.
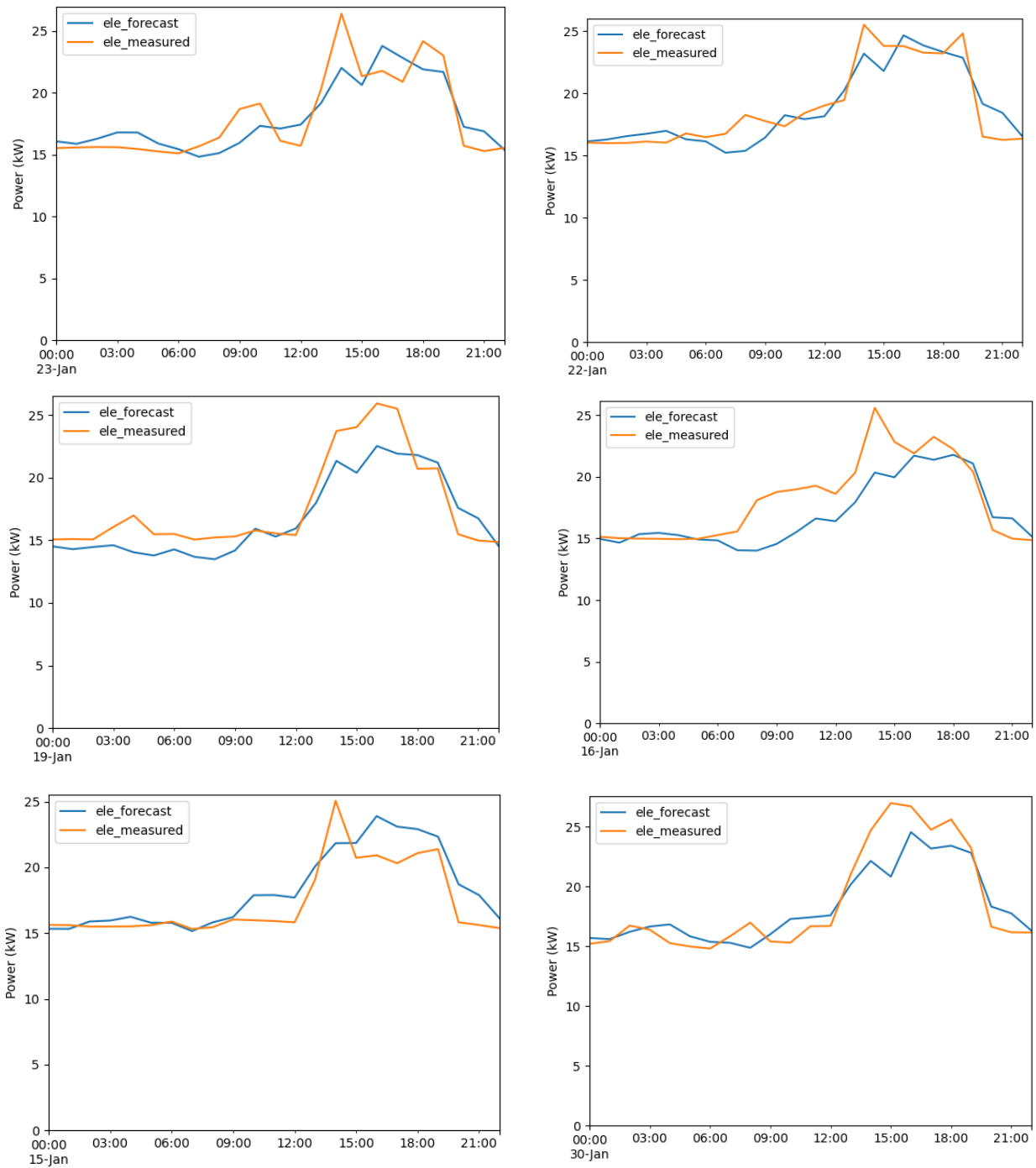
Figure 11: 24-hour electricity baseline load forecasts. Measured values are represented in orange and forecasts in blue.

Figure 12: 24-hour district heating baseline load forecasts. Measured values are represented in orange and forecasts in blue.

As can be seen from the results the model works quite well for the power consumption which is more stable. For district heating the behaviour seems slightly more unpredictable based on the figures (i.e., there are some peaks that are probably caused by irregular domestic hot water usage in the pilot building). Although the normalized error is still similar to the electricity.

### 5.3.1.1.3    Heating system model

The heating system model is designed to predict the response of the heating system during and after a DR event. It is used for modifying the baseline consumption forecast of the machine learning models as illustrated in equations (1) and (2). The outputs of the heating system model are the following:

- $Q_{demand_{a=1}}$: Total heating demand (space heating + domestic hot water) when the heating is constrained. The model is presented in equation (10).

- $E_{hp_{a=0}}$: Heat pump electricity consumption when the heating is not constrained. This model is represented in the equation (11).

- $E_{hp_{a=1}}$: Heat pump electricity consumption when the heating is constrained. This model is represented in equation (12).

- $Q_{hp_{a=1}}$: Heat pump energy production during the DR event (heating constrained). This model is represented in equation (13).

- $Q_{pb}$: Heating payback represents the recoil effect (law of energy conversion) once the heating is returned to the normal set point. This model is represented in the equation (14).

Next how the model produces these outputs is briefly elaborated.

$$Q_{demand_{a=1}} = uQ_{heating} + Q_{dhw} \qquad (10)$$

The $u$ is a learned parameter that specifies the portion of the heating demand that remains when the heating is constrained during the demand response event. Ideally the $u$ is zero (i.e., heating is fully turned off), but in practice the value 0.35 matches the data collected during the pre-pilot phase. $Q_{heating}$ is the space heating demand represented in equation (16). $Q_{dhw}$ is the domestic hot water demand presented in equation (17).

$$E_{hp_{a=0}} = \frac{Q_{hp_{a=0}}}{COP} \qquad (11)$$

$$E_{hp_{a=1}} = \frac{Q_{hp_{a=1}}}{COP} \qquad (12)$$

The heat pump's coefficient of performance is a common parameter in equations (11) and (12). The COP depends on the temperature delta but an average value (3.3) identified during the pre-piloting is used in this model. The heat pump's heat production is provided by equations (13) and (14).

$$Q_{hp_{a=1}} = \min(Q_{hp\_max}, Q_{demand_{a=1}}) \qquad (13)$$

$Q_{hp\_max}$ is the maximum heat production of the heat pump. This parameter is learned from data (50 kWh was identified based on the pre-piloting period). $Q_{demand_{a=1}}$ is given by the equation (10).

$$Q_{hp_{a=0}} = Q_{hp\_max} \qquad (14)$$

The heat pump's energy production is not measured separately which makes it difficult to estimate especially during the normal operation. The assumption presented by the equation (14) is that the heat pump is producing maximum power during normal operation. This assumption is valid for most of the time because the heating system is configured so that the heat pump is prioritized over district heating (i.e., the assumption is only valid when the total space heating and DHW heating demand is lower than the heat pump maximum production.). This could happen in the summertime, but the heat pump is turned off during summer period, because the district heating price drops dramatically during that period.

$$Q_{pb} = \min(Q_{debt}, Q_{pb\_max}) \qquad (15)$$

According to the law of energy conversion the heating system needs to compensate (payback) the energy deduced during the demand response event. $Q_{debt}$ specified the current heat debt. It is calculated as presented in (18). $Q_{pb\_max}$ represents the maximum payback energy during the sampling period (60-min). This parameter is identified from data.

$$Q_{heating} = H \times T_{out} + P_{T_0} \qquad (16)$$

The space heating demand model is presented in equation (16). The module has the following two parameters that are identified from data: $H$, and $P_{T_0}$. The building's heat loss co-efficient (H) and power demand at zero Celsius degrees ($P_{T_0}$) are identified from data as represented in section 5.3.1.1.5.

$$Q_{dhw} = E_{dh_{base}} + Q_{hp_{a=0}} - Q_{heating} \qquad (17)$$

The heating demand for domestic hot water is presented in equation (17). It is calculated based on the DH baseline forecast ($E_{dh_{base}}$), heat pump energy generation in normal mode ($Q_{hp_{a=0}}$) and the heating demand ($Q_{heating}$).

$$Q_{debt} = \begin{cases} Q_{debt} - Q_{pb}, & a = 0 \\ Q_{debt} + p \times u \times Q_{heating}, & a = 1 \end{cases} \tag{18}$$

Equation (18) presents how the heat debt of the building is calculated. When heating is not constrained ($a = 0$) the heat debt is reduced by the payback amount ($Q_{pb}$). When the heating is constrained, the debt is increased by the amount specified by $p \times u \times Q_{heating}$. Here the parameter p specifies the portion of energy that needs to be paid back. It is identified from data. In an ideal system the value would be 1.0. Here a value of 0.35 was identified based on the control testing.

#### 5.3.1.1.4  Parameter identification for physics-based models

**Energy signature method**

The energy signature (ES) method is a simple method for estimating the heat loss coefficient, *H*, of a building. The method is based on modelling the linear relationship between heat consumption and outdoor temperature. Typically, two linear equations are fitted to the data: one for the heating season and one for the intermediate season. **Error! Reference source not found.** illustrates an example of the ES method.
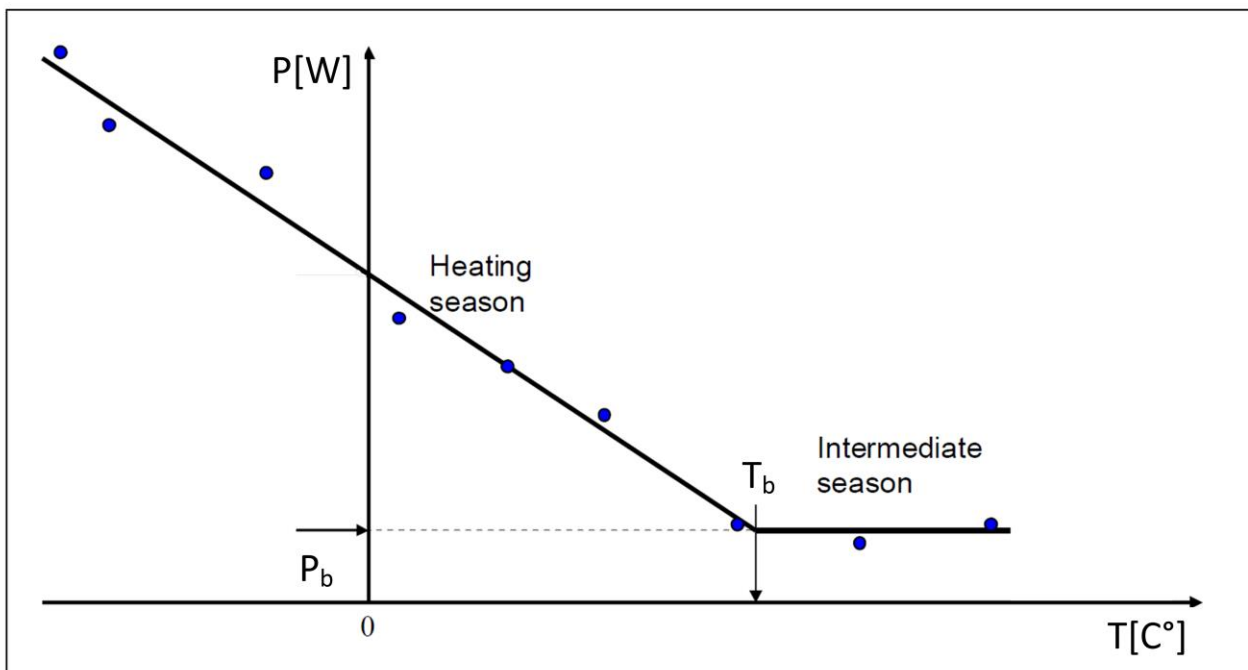


Figure 13: Example of the energy signature method.

T$_b$ refers to the temperature value between the heating season and the intermediate season. P$_b$ is the power used for domestic hot water (DHW) and domestic hot water circulation (DHWC). The heat loss coefficient of the building is the gradient of the line fitted to the heating season.

An important part of the ES method is sampling the data into reasonable resolutions so that heating dynamics are averaged out. Daily resolution was selected for the apartment building to be utilized in the pilot.

**Error! Reference source not found.** illustrates the ES plots for years 2015 - 2020. An extra challenge for the case study studied in the project is that heating in the apartment building is supplied by two sources: district heating and heat pump. To further complicate the challenge, the heat pump has been operating only part time due to technical issues. Because of this, two different regression lines can be identified from the heating

season: one with the heat pump and one without the heat pump. The heat pump has been also turned off during summer times because the price of district heating is so low that it is cheaper to use it for heating.

Data from year 2018 was used for fitting the regression line (and evaluating the building heat loss coefficient). In order to fit a regression line to the heating season data, the periods in which the heat pump was off and on have to be separated. To automate this a machine learning clustering algorithm, called a Gaussian mixture model, was utilized. In practice the clustering was implemented with scikit-learn. StandardScaler of scikit-learn was used for scaling the data before clustering. Only data below 10.0 Celsius degree (heating season) was utilized in the clustering. **Error! Reference source not found.** illustrates the results of the clustering as well as the two lines fitted to the heating season data.

The heat loss coefficient is the gradient of the line when the heat pump has been off. In this case the gradients are almost identical which indicates that the heat pump output is independent of the outdoor temperature. The $H$ is -4.14 kW/C° and the linear model of the space heating (DH) is presented in equation (19):

$$P_{\text{heating}} = -4.14 \left[ \frac{\text{kW}}{\text{C}°} \right] * T_{\text{env}} + 66.0 \, [\text{kW}] \qquad (19)$$
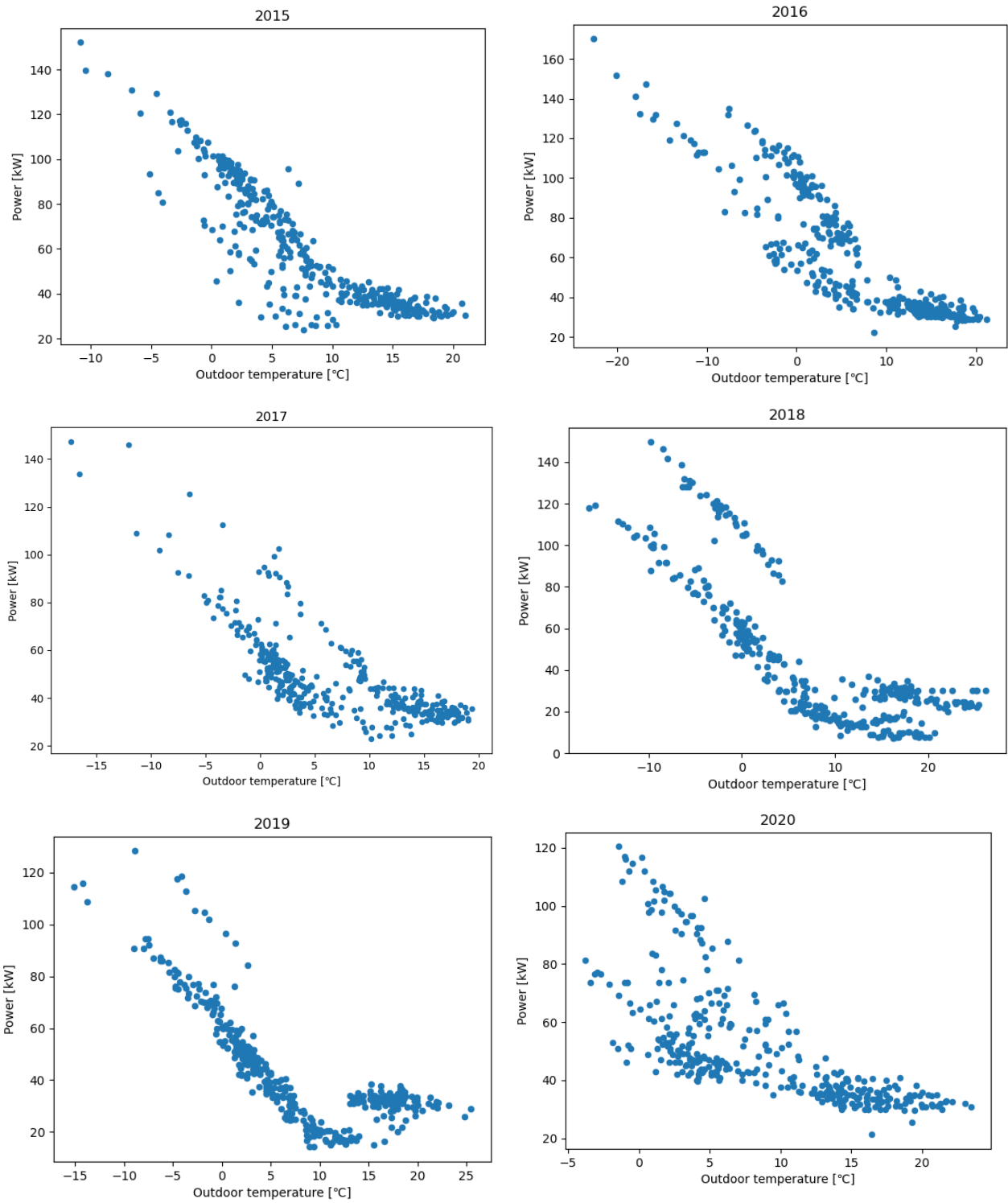
Figure 14: Daily average power versus the outdoor temperature for years 2015 - 2020.
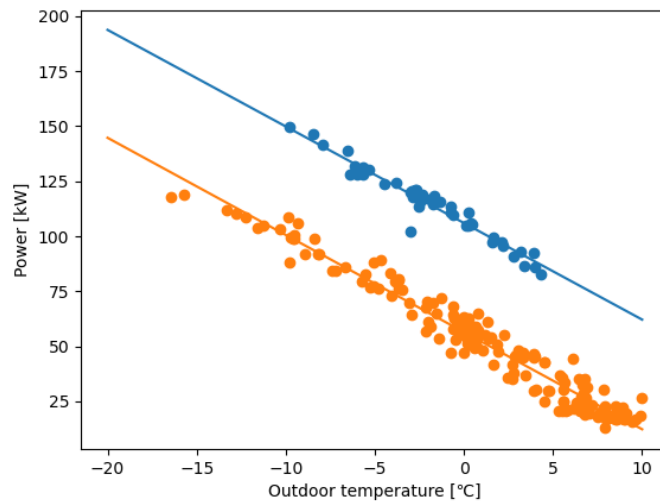
Figure 15: Application of the energy signature method for year 2018 by fitting separate lines for periods in which the heat pump has been on (orange) and off (blue). Data was clustered with a Gaussian mixture model algorithm.

**Estimating the thermal capacitance of the building**

The thermal mass (capacitance) of the building, *C*, can be calculated for example as defined in standards SFS-EN ISO 13786 or SFS-EN ISO 13790. High detail structural information is needed to perform detailed calculation of a building's thermal capacitance. The approach applied in the first phase modelling is based on utilizing typical values calculated for different building types, and fine-tuning these values based on data obtained from actual experiments. **Error! Reference source not found.** - **Error! Reference source not found.** present thermal capacitance values for different building and structure types that are typical in Finland.

Table 1: Typical thermal capacitance values per conditioned floor area for detached houses in Finland, including the furniture.

| Structure type | Example structures | $C/A$ [$Wh/(m^2 K)$] |
|---|---|---|
| Light | All walls and floors are lightweight materials. | 40 |
| Medium I | Base floor is concrete, all other walls and floors are lightweight materials. | 70 |
| Medium II | Exterior wall is concrete brick or massive timber, base floor is concrete and other floors lightweight materials. | 110 |
| Heavy | Walls are concrete or concrete bricks. Floors are concrete. | 200 |

Table 2: Typical thermal capacitance values per conditioned floor area for apartment buildings in Finland, including the furniture.

| Structure type | Example structures | $C/A$ [$Wh/(m^2 K)$] |
|---|---|---|
| Light | Base floor is concrete, all other walls and floors are lightweight materials. | 40 |
| Medium | Walls are lightweight materials. Floors are concrete. | 160 |
| Heavy | Floors and walls are concrete. | 220 |

Table 3: Typical thermal capacitance values per conditioned floor area for office buildings in Finland, including the furniture.

| Structure  type | Example structures | $C/A$ [$Wh/(m^2 K)$] |
|---|---|---|
| Light | Base floor is concrete, all other walls and floors are lightweight materials. | 70 |
| Medium | Walls are lightweight materials. Floors are concrete. | 110 |
| Heavy | Floors and walls are concrete. | 160 |

The pilot building is an apartment building with medium structure type. So 160 Wh/(m²K) is used as an estimate for C/A. The floor area, A, of the building is 4500 m2, which means that the initial estimate for *C* is 720 kWh/K. This parameter can be recalibrated based on measurements obtained from the building. The equation used for identification of C from data can be derived from the equation (6). Below the equation is represented in a format where the $\frac{dT(t)}{dt}$ is approximated with temperature change over 60 minutes (i.e., during a DR event).

$$\Delta T = -\frac{H}{C}(T - T_{env}) \tag{20}$$

From this we can solve the building's heat capacity as presented in equation (21).

$$C = -H\frac{T - T_{env}}{\Delta T} \tag{21}$$

$H$ is -4.14 kW/C° as presented above. $\Delta T$ is the average drop in indoor temperature during an hour of DR event where heating is constrained. $T - T_{env}$ represents the difference between indoor and outdoor temperatures. Average difference during the DR event is used for evaluating the $C$. Based on DR tests done in the pilot building between November 2021 and June 2021, corrected value for the C can be calculated. In those tests, indoor temperature measurements from seven different locations of the building were used. Mean value for the C based on DR tests was 390 kWh/K with standard deviation of 187 kWh/K.

#### 5.3.1.1.5    Integrated digital twin of the building community

The aforementioned models comprising the digital twin of the building community are implemented with Python programming language. The model takes as input the current indoor temperature, as well as the outdoor temperature and control commands (on/off) for the period to be forecasted. The outputs of the model include the electricity and DH consumed for space heating as well as the indoor temperature for the forecast period. Figure 16 illustrates three example forecasts evaluated against the data collected in the pre-pilot phase. In each example, a DR event is executed so that all the models of the digital twin need to be used (i.e., the physics models are used in addition to the baseline load profiles produced by the machine learning models). In the first two experiments, a single DR event is executed at 7:00-8:00 and 8:00-09:00, respectively. In the third example, two events are executed during the 24-hour forecast window at 7:00-8:00 and 12:00-13:00.

The left side of the column represents the district heating consumption and forecast. It can be seen that there is a clear drop in the DR consumption during the times when the DR event was executed. The model predicts the performance quite well also during the DR event. For the electricity consumption, there is no identifiable impact in the energy consumption caused by the DR event. This is because the heat pump starts to serve the DHW heating once the space heating is constrained. The model seems to predict this behaviour also well.
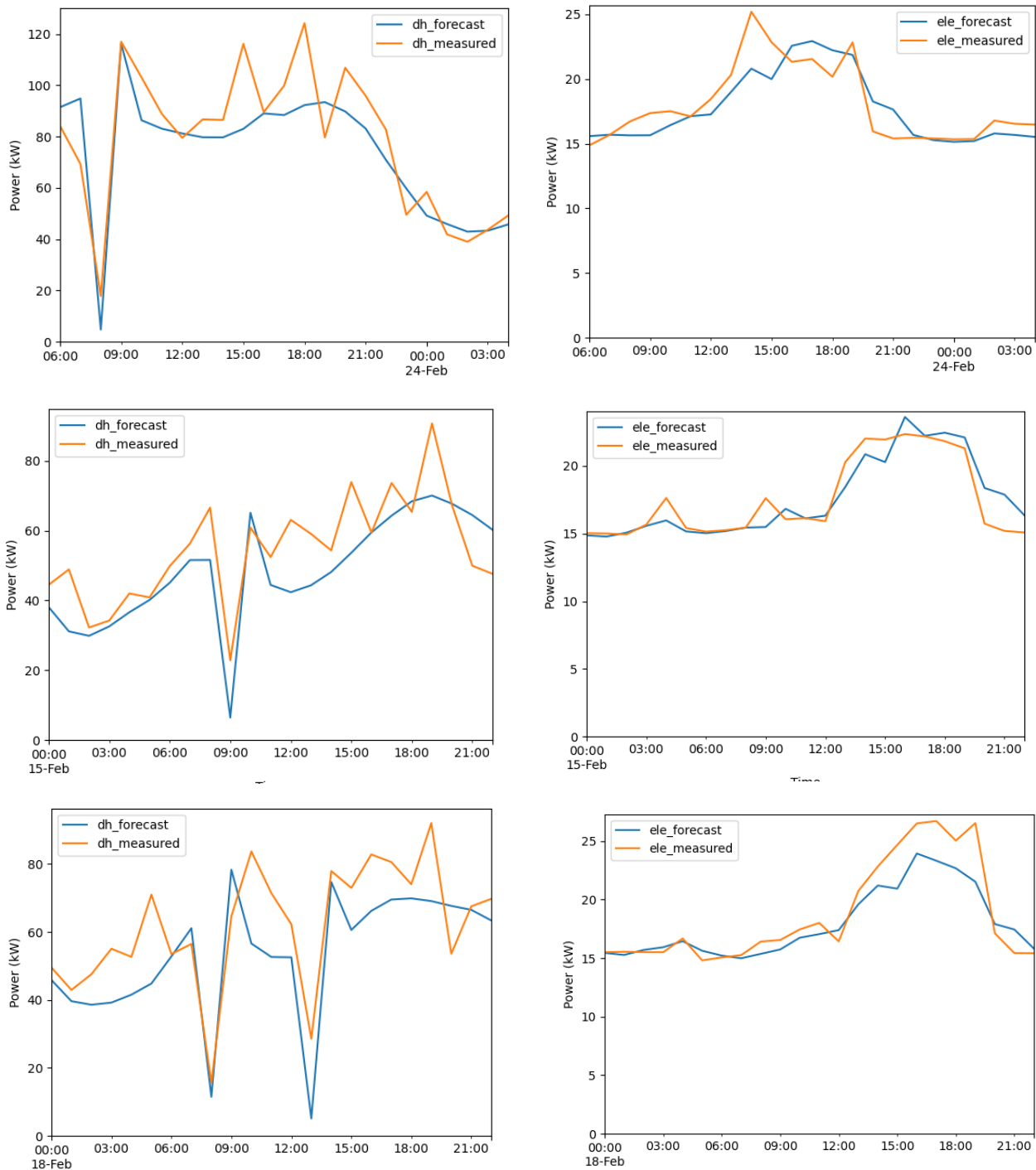
Figure 16: Forecasted response of the building energy consumption during a day with a DR event. The district heating forecast and consumption is presented in the left column. The electricity consumption of the building is represented on the right. The forecast is represented in blue and measured energy consumption in orange.

### 5.3.1.2 Approach 2

The above-mentioned approach 1 uses machine learning models to calculate baseline consumption for the both district heating and electricity and combines it with a physics-based model to model the behaviour of the building energy consumption during and after a demand response event. This kind of approach was selected mainly because several years of data about energy consumption during the building normal operation was available, so the baseline consumption could be modelled using accurate machine learning methods. However, before the start of the iFLEX piloting, no demand response tests were done in the pilot building.

Therefore, machine learning could not be used to forecast the behaviour of energy consumption during and after demand response tests and that is why these periods were complemented with physics-based models described in the previous section to model e.g. the rebound effect and how the district heating and electricity consumption behaves during demand response action.

During the phase 1, demand response control events have been executed several times of the week on different times of the day. Figure 17**Error! Reference source not found.** presents an example how district heating and electricity reacts of the district heating control test that has been executed between 10:00 – 11:00 9.2.2022. By running series of demand response tests, dataset now contains also samples from the periods, where HVAC system is either providing flexibility (space heating is turned off) or recovering from demand response events after space heating is turned on again.
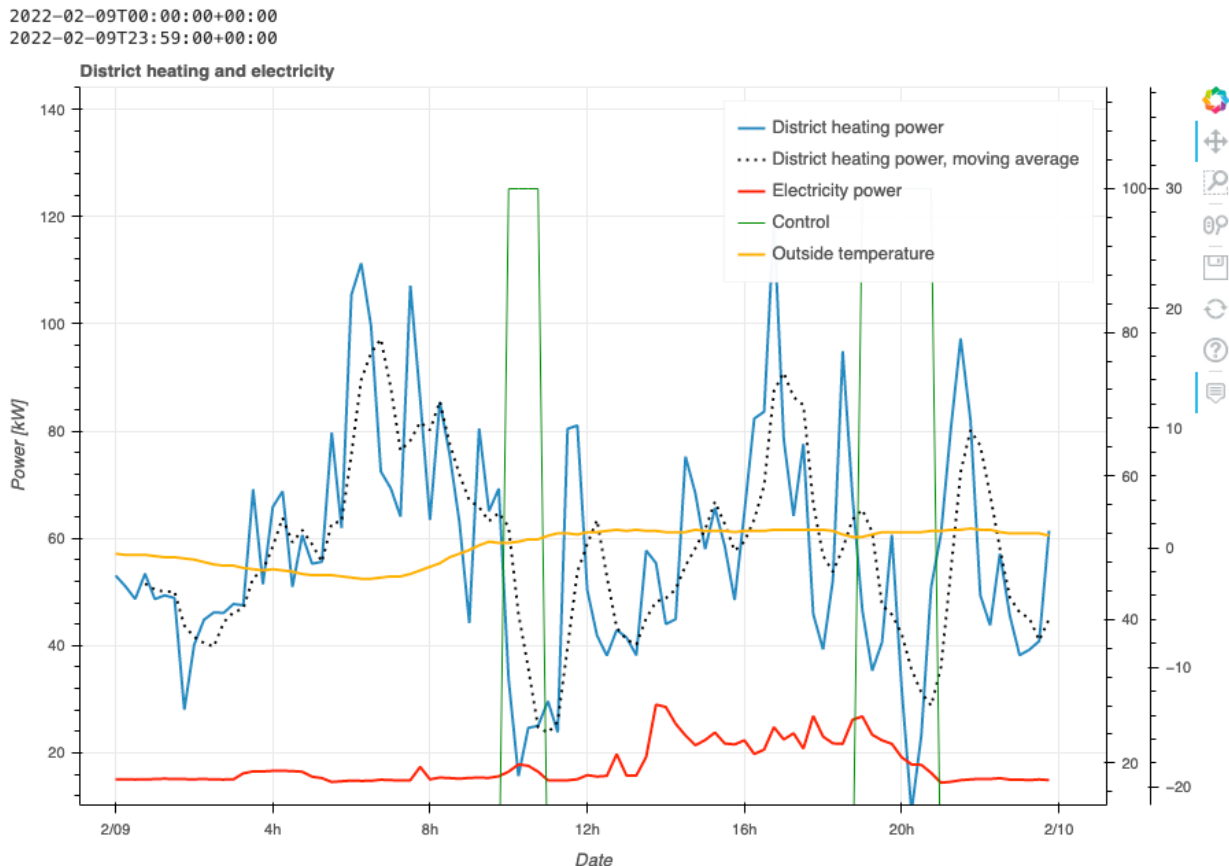


Figure 17. Demand response test on February 2022

Using the dataset collected during pre-pilot phase of the project, an approach (approach 2) to model building community energy system has been developed. Conceptual view of this approach is presented in Figure 18. Two main aspects of this approach are the following:

- District heating power and electricity power are fully modelled with machine learning. District heating consumption includes energy required to generate heating for the building and also heating of the domestic hot water. Behaviour of the residents has significant effect on the usage of the water and also on the required heating energy. For example depending the time of the day, taking a shower creates significant need for domestic hot water. Electricity consumption includes all the energy usage in shared spaces of the apartment building, where the most significant power consuption comes from heat pump, ventilation, sauna, lightning and elevators

- Indoor temperature during the demand response event is modelled using physics based greybox modelling, in the same way as the approach 1.
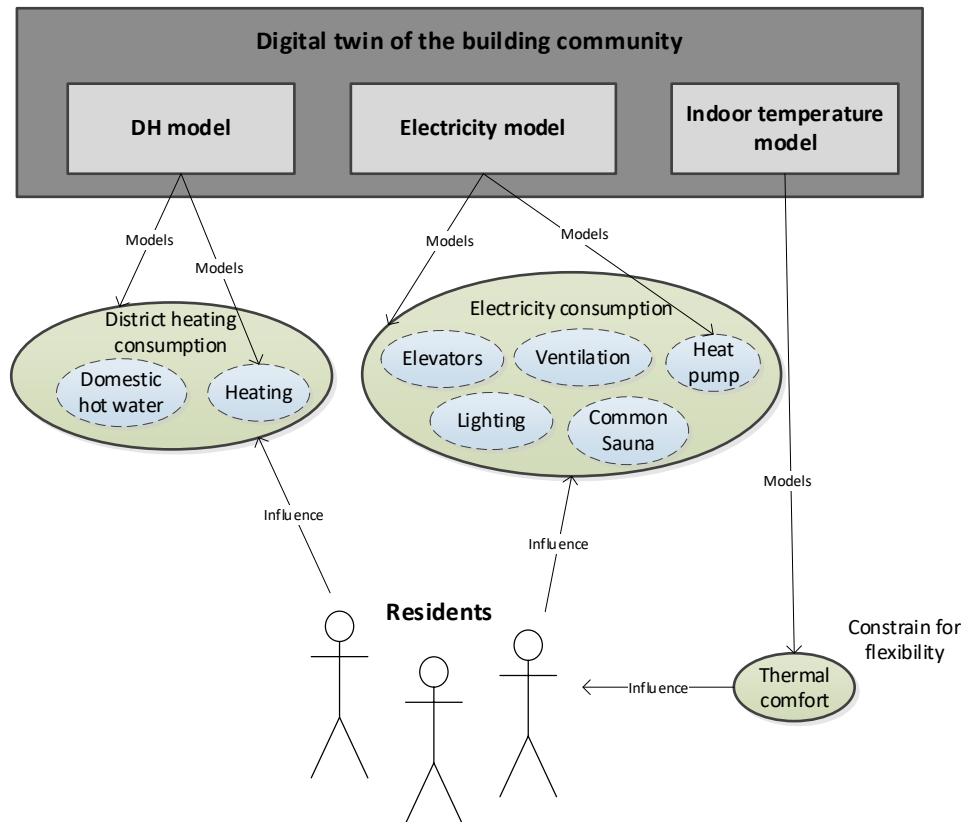
Figure 18.  Conceptual view of the digital twin for the building community (approach 2).

District heating and electricity models are created using automated machine learning pipeline presented in the section 5.2. Using this method, separate models are created for the targets i.e. district heating power and electricity power. Lead time (timesteps (data frequency) before the first forecast period) has been 0 hours and forecast length (the length of the forecast in data frequency timesteps) has been 24 hours. Various input features have been tested and the following presents one of the most promising input parameters:

- lagged target (168 historical values of the target, integer): Historical values of either district heating or electricity power measurements,

- weekday (24 future values, one for each forecasted step, OneHotEncoded): day of the week

- hour (24 future values, one for each forecasted step, integer): Current hour of the day

- temp(24 future values, one for each forecasted step, 8 historical values, integer): outdoor temperature

- holiday(24 future values, one for each forecasted step integer): government-designated holiday, True/False value

- control (24 future values, one for each forecasted step ,4 historical values, integer): presents the past control singals. Control point deviates the heating circuit water input temperature level so that no heating is provided to the heating system. 4 step window size has been tested in order to model the reaction of the building heating system during the demand response event and also model the rebound effect

Figure 19 illustrates the internal architecture of the model. Using above-mentioned input and output parameters, output size of the model is 24 and input size is 444. Machine learning model predicts both the baseline consumption for district heating and electricity, but also the down flexibility for both targets. Baseline consumption is calculated by setting control input to 100 and possible flexibility is calculated by setting the control input to 0. Indoor temperature model is developed originally for the approach 1 and is used also to calculate how long demand response actions can be made.
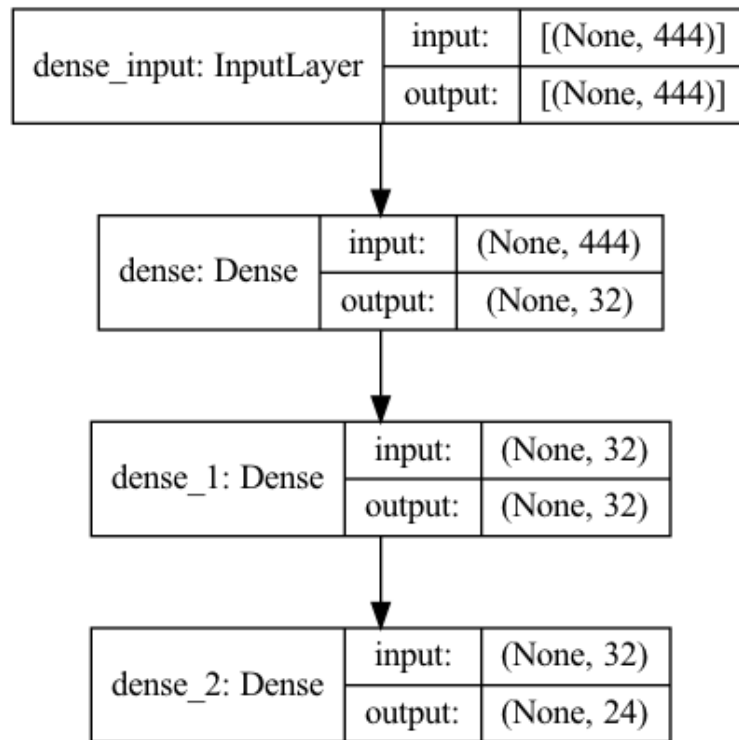
Figure 19: Model architecture

### 5.3.2 Digital twin of a household

The digital twin of a household consists in this initial phase of four models: a household thermal model, a household electricity model, a household flexibility model and an occupant model. Conceptual relationships between the models, measured data and household residents are presented in Figure 20. Presented models are build based on the measured data and their historical values. The models have some clear constraints related to household comfort influenced by the household residents. Some of the modelling relationships are denoted with dashed lines, they will be modelled in the second piloting phase. The digital twin model is subject for improvements in the next project phases.
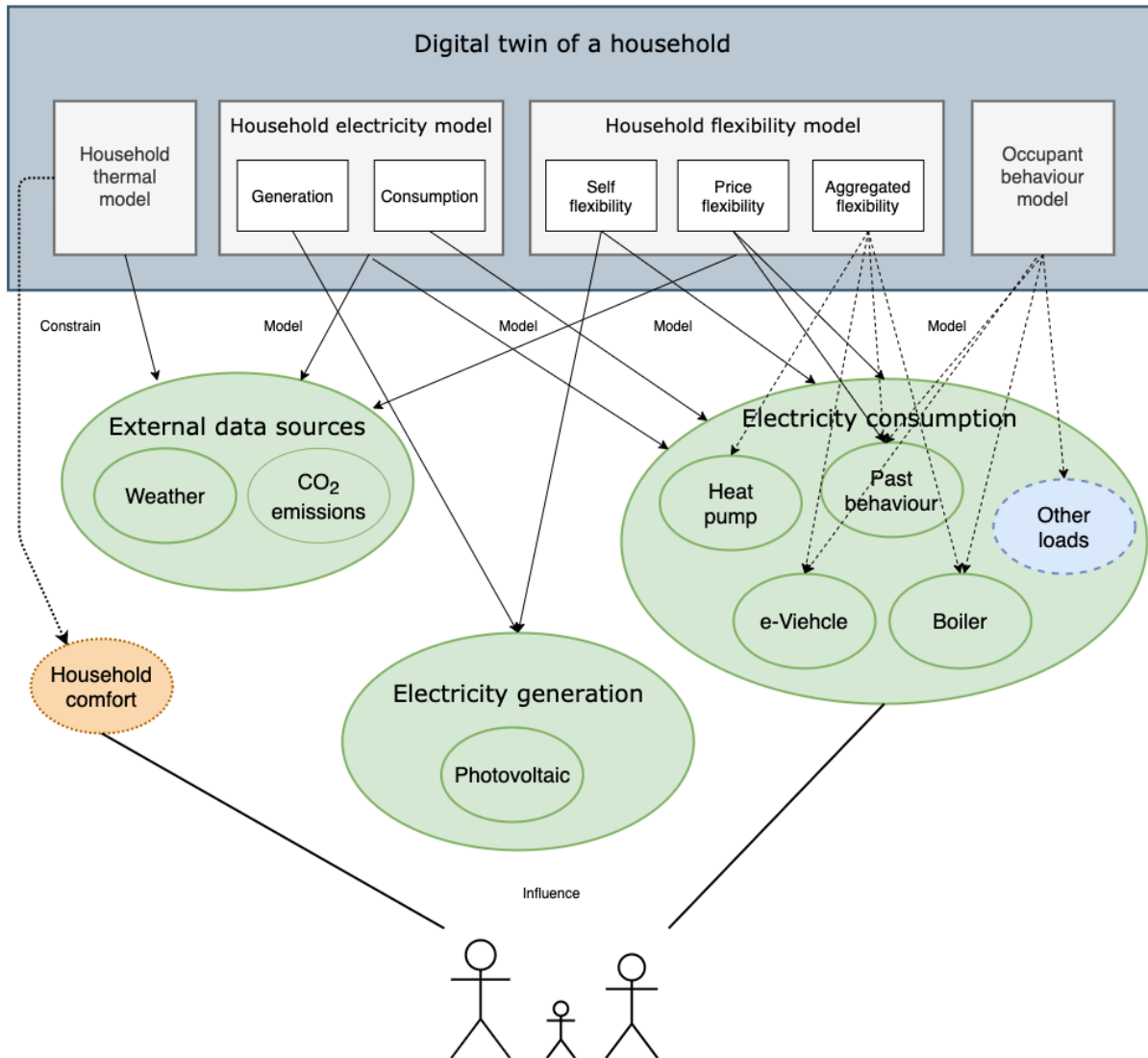
Figure 20: Household modelling, models in grey, measured parameters in green, constrains in orange, non-measured parameters in blue. Full arrows represent phase one focus.

The household thermal model aims to evaluate the thermal response of the household and the potential of the household thermal lag. Heating and cooling can present an important part of energy consumption of a typical household. When present in electricity consumption, they are roughly proportional to the difference between indoor and outdoor temperature. The thermal lag tells how the energy flows between the inside of the household and the outside. The lag depends on physical details of the thermal mass of a building or an apartment of the household. Thermal mass has the effect of dampening and delaying the transmission of heat and cold from the outside. Depending on the building characteristics, the delay can vary between a few hours to more than a day.

The household electricity model defines how the electricity in the household will be consumed and generated, if generation is present, in the future. These two models are essential for other models building. Both the consumption and generation are dependent on weather data. The prediction depends on weather data prediction which is assumed to be provided from external data sources.

The flexibility models define how much flexibility is available at a certain time period in the household for specific flexibility provisioning. Flexibility provisioning may be divided between self-provisioning and provisioning to external parties, for example, for participation in demand response (DR) programmes. Modelling of external provisioning can be based on information on participation in previous DR programmes or based on aggregation of household, through HEMS controlled appliances. Three models will be considered: self-flexibility, aimed at balancing between self generation and consumption, price flexibility, telling how the household responds in general to DR price signals and aggregated flexibility, defining an aggregated response of the household through home energy management system (HEMS) control. The self-flexibility can be defined

on generation and consumption data, price-based flexibility on the consumption forecast and past price events behaviour, and the aggregated flexibility on usage potential of the controlled appliances through the household HEMS.

Many large loads are still under direct household occupants' control like hair-dryers, blenders, TVs, ovens, etc. Such loads are denoted with other loads in Figure 20. Also, electric vehicles are considered under direct control though they could be controlled through automation. Defining an occupant behaviour model is a complex task due to the inherent unpredictability of the human behaviour. A simpler substitute can be defined based on pattern of usage within days, days of a week or seasons.

In the following sections a baseline for the introduced models will be presented. Some of the models will be addressed only at later piloting stages.

### 5.3.2.1 Data used for models building in the first phase

The data used for modelling in the first phase is of three sources:

- Smart metering data collected in a Critical Peak national project (KKT) in years 2017 and 2018. The data belongs to more than 700 pilot users and more than 14.000 control users in Celje region. The region covers more than 200 transformer stations. The data is accompanied with weather data in the same period. The smart metering data is provided on 15-minute intervals, the weather data is on 1h intervals. For the weather data temperature, radiation and precipitation measurements are provided,

- Smart metering data being collected in Use it Wisely (UiW) national project from autumn in year 2019 till end of year 2021. The data belongs to more the 700 pilot users and more then 10.000 control users in Celje region. The region covers more than 200 transformer stations. Besides the residential consumption also the generation at the prosumers is provided. The data includes industrial consumers at the substations as well. The region and the users are not the same as in the KKT project. The data is accompanied by weather data of same characteristics as in KKT project,

- HEMS data, collected at first phase pilot users. The piloting is planned to start in autumn and at that time more information will be available on which data will be collected and at which sampling rate.

### 5.3.2.2 Prosumer flexibility forecasting

The starting point of flexibility forecasting is the prediction of the response of prosumers to a price signal. The model which will be presented is a generalization of a price flexibility model as denoted in Figure 20.

The flexibility based on price signals is defined as the ability of the consumers to adapt their consumption according to the price of the energy they are consuming. The approach estimates harvested flexibility during a flexibility event. The flexibility event is defined as a time interval when an aggregator signals a price change – positive or negative deviation from standard price. Deviations are considered from the aggregator point of view. Positive deviation introduces higher price of energy and negative lower price. The higher price periods aim at lowering the peak in the period and lower price periods increasing the consumption at the period.

The modelling is based on past flexibility events. In 2017 and 2018 Elektro Celje (ELE), SCOM and JSI have participated in a national project called "Critical peak tariff" (CPT). The project has been granted by The Energy Agency in the Slovenian energy market[1] has introduced a positive critical peak tariff (PKKT) as an incentive for consumers. At the time of the PKKT the network fee was up almost 10-fold to normal tariffs and in the rest of the time was a bit lower – on the end, for both tariffs combined, the consumers were paying less than they would normally do in the same period. Over 700 households participated in the pilot. The pilot prepared 42 flexibility events which have been announced a day ahead based on consumption forecast of 209 transformer stations, participating in the pilot. The events were on average one hour long but some experimentation has been done with two-hour events. The modelling will be updated based on currently running project "Use it Wisely" (UiW) results. The project involves the same partners as the previous one. However, the PKKT incentive of the UiW supports a "negative" incentive as well – NKKT. During the NKKT periods, the network fee is much lower than normal. 3650 hours are available for NKKT and 100 hours for PKKT per year. Roughly the same number of households participates in the project as before, while their topological spread is similar – but different – to the CPT project.

The data used as input for the flexibility modelling is aligned across flexibility events. The events used as input lasted for one hour or two hours. Events are scheduled during different times, depending on a time of the year. For this reason, they were aligned at event start including with an hour and a half before and after one hour

---

[1] See the agency home page for more details: https://agen-rs.si/web/en/about-the-agency

event and an hour before and after two hours events. Combined, all events were four hours long. Sample two-hour event is presented in Figure 21. The flexibility event took place between 18:15 and 20:15. The event is marked with 1 in event column and intervals before and after events are marked with 0. All aligned events have the same event index.

| | Sample | event | event_index | event_day |
|---|---|---|---|---|
| **2018-03-22 17:15:00** | 69.0 | 0 | 1 | 1 |
| **2018-03-22 17:30:00** | 70.0 | 0 | 2 | 1 |
| **2018-03-22 17:45:00** | 71.0 | 0 | 3 | 1 |
| **2018-03-22 18:00:00** | 72.0 | 0 | 4 | 1 |
| **2018-03-22 18:15:00** | 73.0 | 1 | 5 | 1 |
| **2018-03-22 18:30:00** | 74.0 | 1 | 6 | 1 |
| **2018-03-22 18:45:00** | 75.0 | 1 | 7 | 1 |
| **2018-03-22 19:00:00** | 76.0 | 1 | 8 | 1 |
| **2018-03-22 19:15:00** | 77.0 | 1 | 9 | 1 |
| **2018-03-22 19:30:00** | 78.0 | 1 | 10 | 1 |
| **2018-03-22 19:45:00** | 79.0 | 1 | 11 | 1 |
| **2018-03-22 20:00:00** | 80.0 | 1 | 12 | 1 |
| **2018-03-22 20:15:00** | 81.0 | 0 | 13 | 1 |
| **2018-03-22 20:30:00** | 82.0 | 0 | 14 | 1 |
| **2018-03-22 20:45:00** | 83.0 | 0 | 15 | 1 |
| **2018-03-22 21:00:00** | 84.0 | 0 | 16 | 1 |

Figure 21: Sample flexibility event data

Flexibility forecast is done based on past flexibility events, consumer response and weather data. For the forecast a black box model is used. Linear, multilinear, Deep Neural Network (DNN) (LeCun, Bengio, & Hinton, 2015), Recurrent Neural Network (RNN) (Jordan, 1986) or gradient boosted tree (Natekin & Knoll, 2013) based models have been evaluated to model the flexibility relationship.

A simple linear model is presented in Figure 22. A dense neural network node presents a base for the linear model. Applying temperature and consumption data of the events' time slots to the model gives the results in Figure 23. From the figure we can recognise the thermal response of the households, indicating both the heating part in the left-hand side of the figure and the cooling dependency at temperatures higher than 23°C.
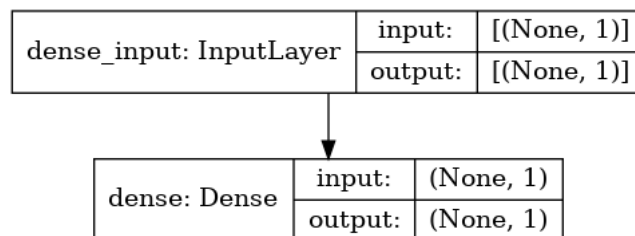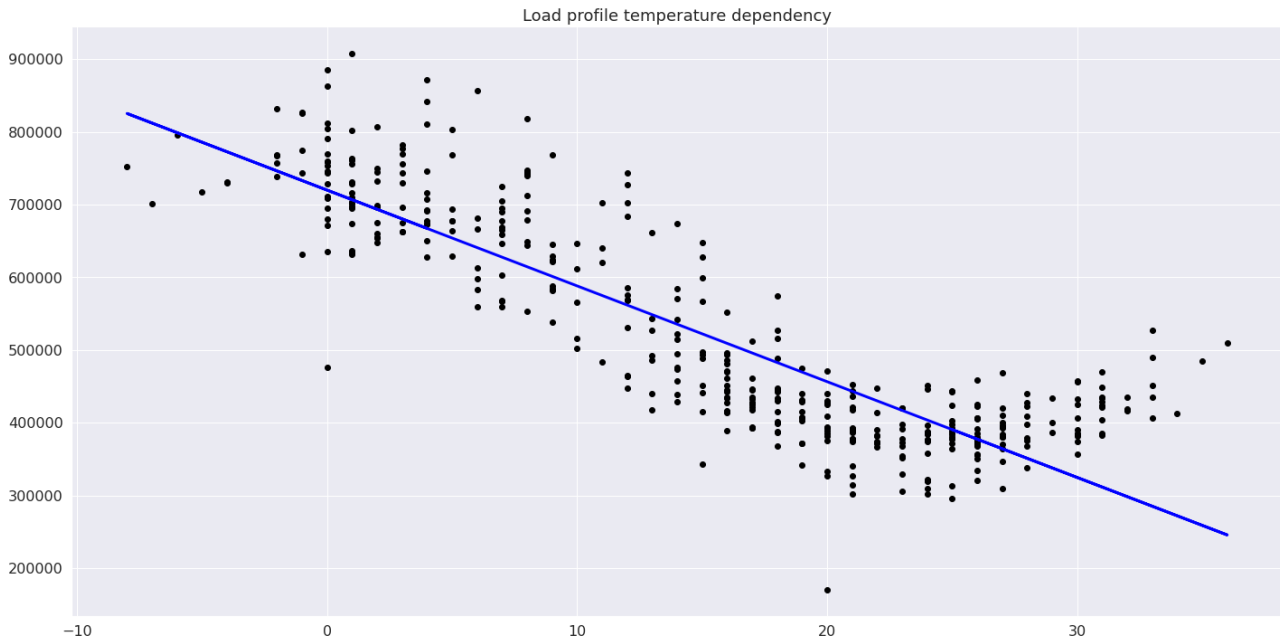


Figure 22: Simple linear model

Figure 23: Linear regression and temperature dependency of event consumptions

An example Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) stacked neural network model used is presented in Figure 24. The neural network uses dense pre-layer and five 128 nodes LSTM layers stacked one on another and connected in a ResNET (He, Zhang, Ren, & Sun, 2016) manner to prevent a vanishing gradient problem. Each LSTM layer is accompanied with a dropout layer.

For the extended flexibility events the consumption has been forecasted with a chosen model based on the data outside the events. For different models, the predictions are given in Figure 25. The black line presents the real consumption of the aligned events. The blue and red line present LSTM model, presented in Figure 24, prediction. The yellow and green lines are XGboost (Chen & Guestrin, 2016) predictions, green one is parameter optimized.

It can be seen from Figure 25 that the XGBoost prediction seems to be superior to LSTM model one. When the right model is selected, the flexibility can be calculated as a difference between predicted and real consumption on the flexibility events. Based on this data, in a similar manner as before, the new flexibility events and their flexibility can be predicted. Flexibility prediction is limited only to weather forecast availability. While doing the prediction, care needs to be taken to take into the account the number of consumers participating in the event. The approximate time of previous events should be taken into account for new events.

It has to be decided how to progress from common flexibility prediction to individual prosumer flexibility prediction. At first, new data from UiW project will be evaluated and the data used for modelling will be updated. Besides the PKKT incentive, also the NKKT incentive data will be evaluated in a similar manner to see how cheaper networking fees stimulate higher consumption in off peak hours. Then, the flexibility of an individual will be evaluated over all PKKT and NKKT events. The model will be built in a similar manner to common flexibility prediction models and evaluated on the events data. The model will be confirmed by aggregated models as well for the prosumers with HEMS systems available, already in the first phase. The price-based model is important since it allows to evaluate the flexibility of the prosumers on smart metering data only.
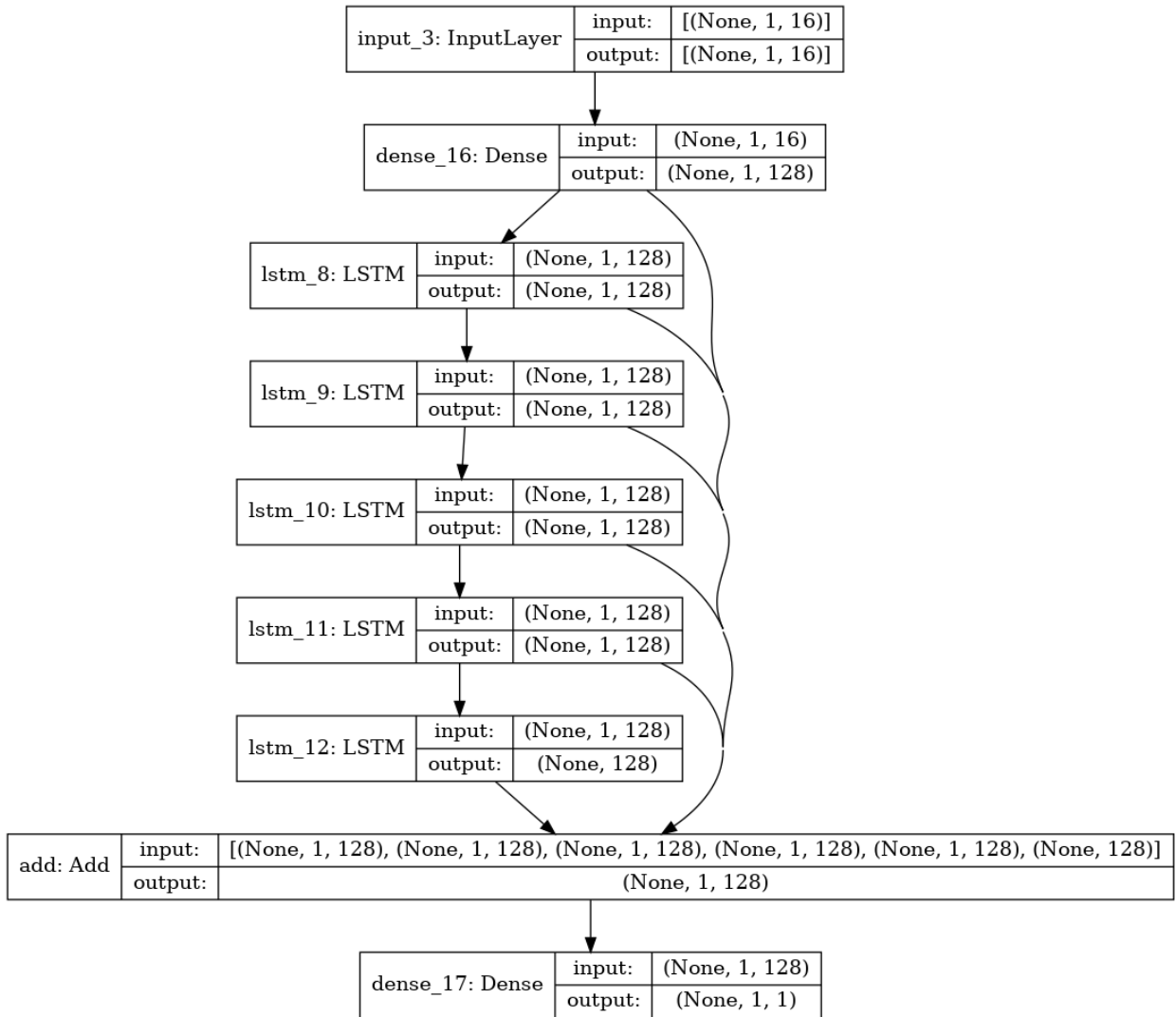
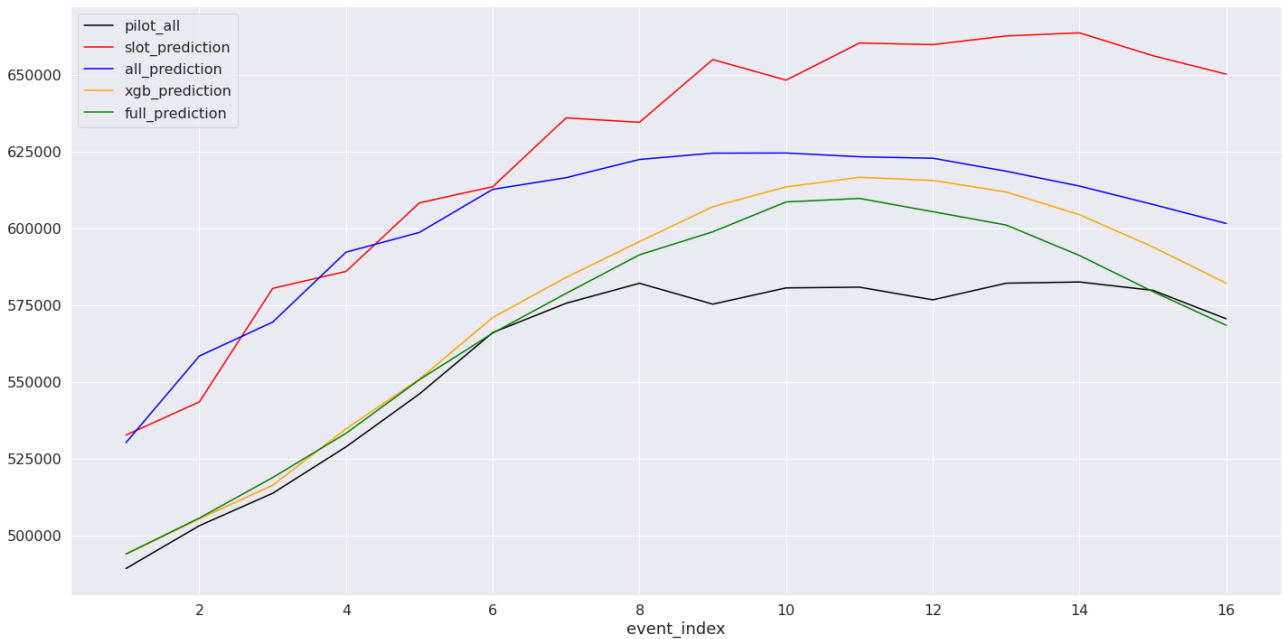Figure 24: LSTM stacked model with dense pre layer



Figure 25: Aggregated and averaged flexibility response, multiple models

5.3.2.3 **Prosumer load and generation forecasting**

Prosumer load and generation forecasting is one of the core services expected from iFLEX Assistant. The load forecast enables to assess implicit flexibility potential as has been discussed in 5.3.2.2 so it can be reported to flexibility services operators, allows for more optimal automation control and management, helps the end user to decide which flexibility services to participate in, etc. The generation forecasting helps to plan how to use self-flexibility to balance the household own consumption as well as to plan future consumption potentials.
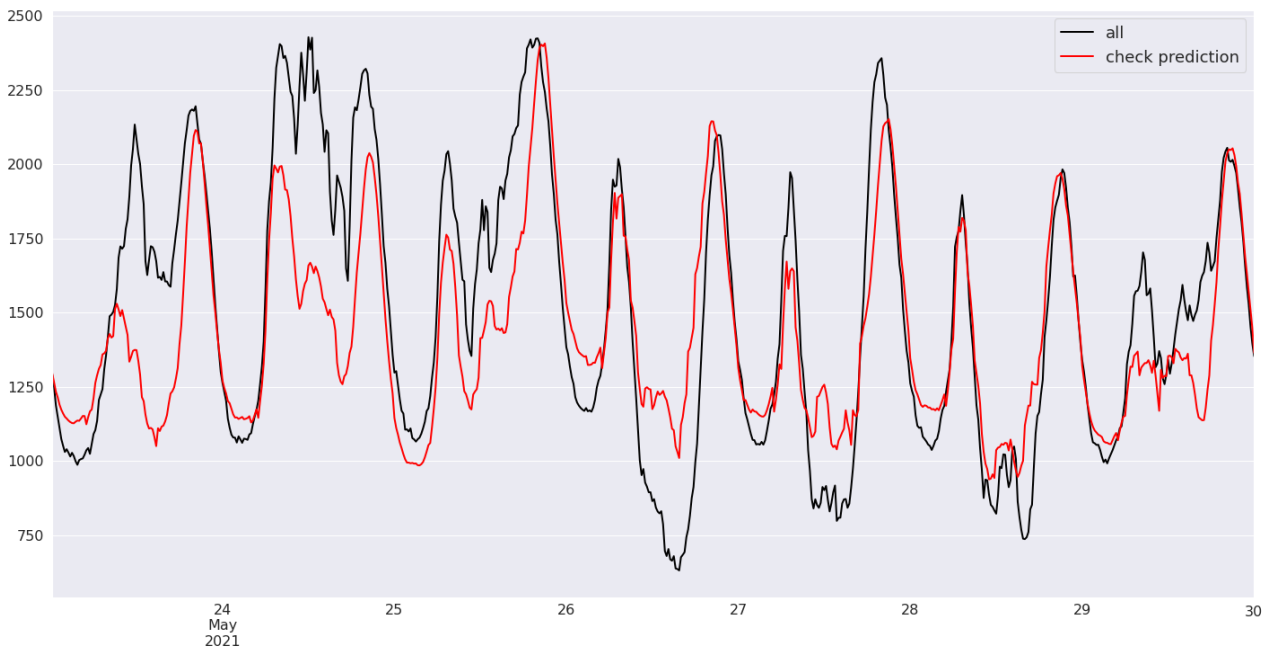


Figure 26: Stacked LSTM model consumption prediction

The baseline services already provided are based on forecasting the consumption of a group of pilot users in the CPT and UiW project. The consumption of more than 750 users is regularly forecasted for a week in advance to be able to select optimal days and time intervals for the flexibility events. For the week forecast, the key input parameter is the time of expected peak so that the flexibility event will match the peak. On overall year horizon it would be optimal to shave the highest peaks in the network for the KKT incentive to have the best effect. For this purpose, also the height and shape of the peak is important so the right peaks could be selected for shaving.

Two types of models have been used and evaluated for UiW project prediction: LSTM neural network based model and XGBoost model. The LSTM model is similar to the one in Figure 24. The stacked LSTM model with dense pre-layer and dropout layers provides in 512 node, 6 layer configuration, the result as is presented in Figure 26. The input vector consists of 183 variables based on statistical variations of inputs of previous consumptions, time based and weather parameters: temperature, radiation and precipitation. The prediction is provided for a week in advance, the variables are build in a way to allow the prediction of a whole week. A whole week is mostly not achievable since the metering data is D-1 and event data is not complete. The real forecasting span is then 5 days long. The weather forecast provided is for 7 days.

In Figure 27, a basic evaluation of five statistical parameters of the forecast is provided per day in percentage values: peak height error (blue), peak time error (orange), min value error (green), peak-to-peak error (red), standard deviation error of day data (violet) and mean error (brown). From Figure 26 and Figure 27 it can be seen that the peak time is quite well predicted. The peak height prediction error varies around and below 10%, with one bold missed prediction on 24th of May. Peek-to-peek and standard deviation errors are in the range of 20% and the mean value error is close to the peak error.
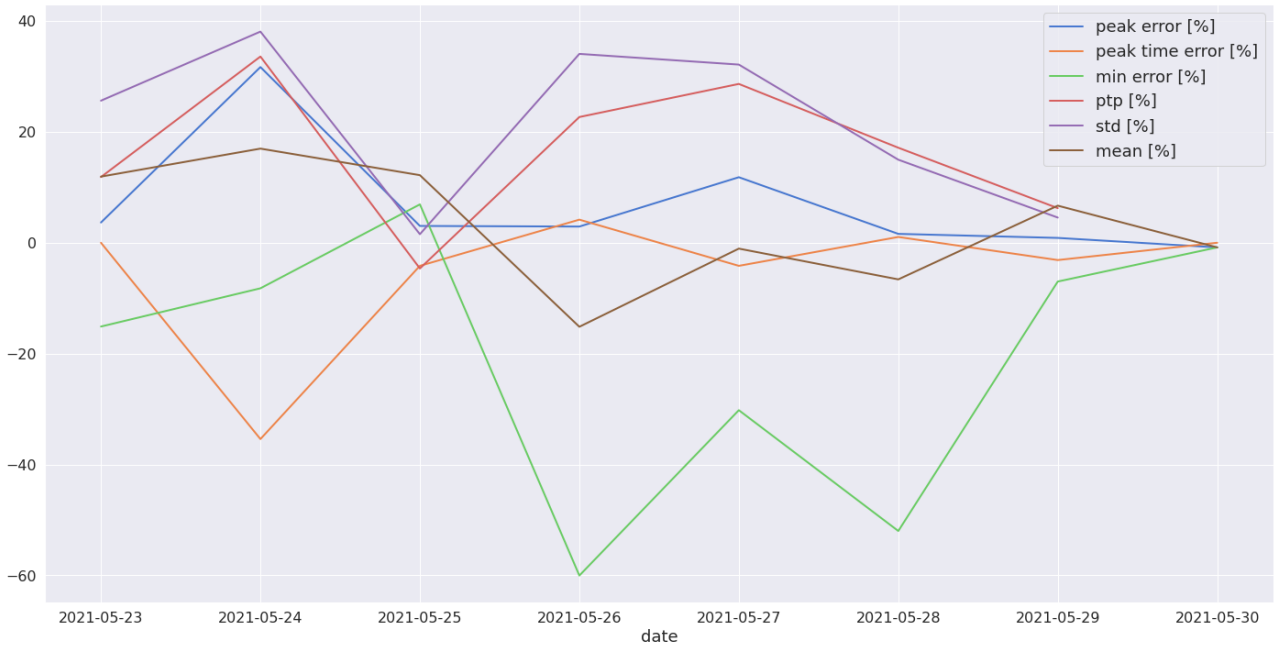
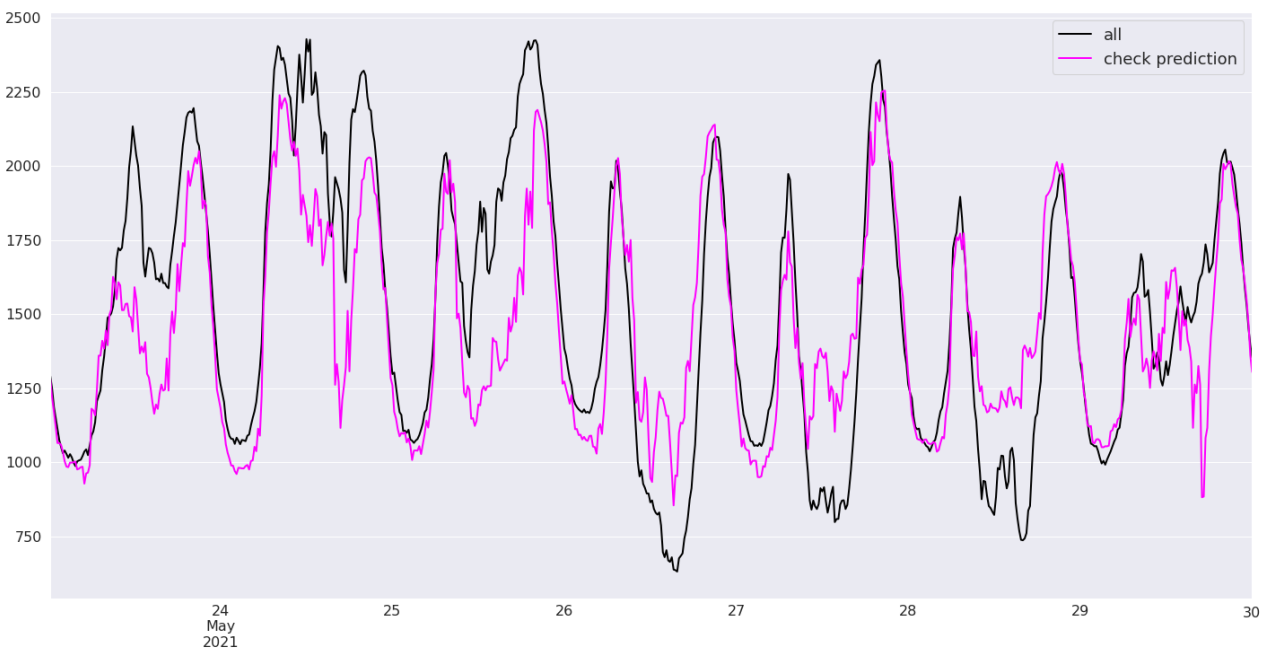Figure 27: Stacked LSTM model consumption prediction errors



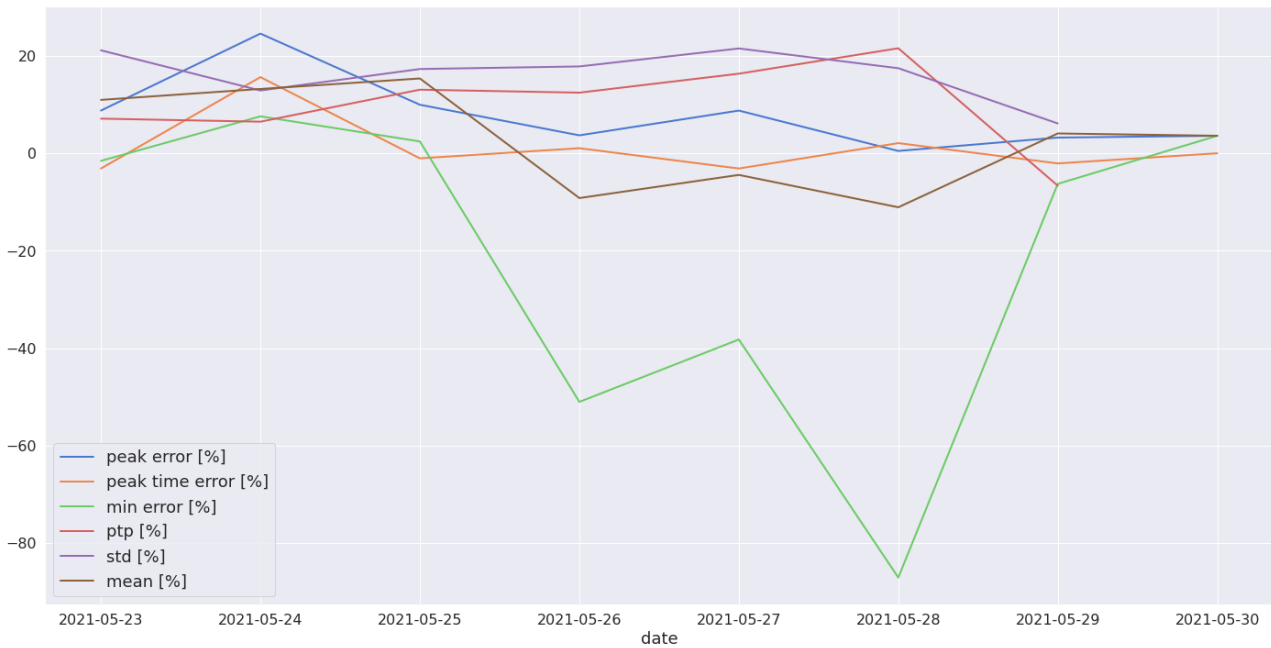Figure 28: XGBoost optimised consumption prediction model

Figure 29: XGBoost optimised consumption prediction model errors

In Figure 28 a XGBoost prediction of the same load is provided. The observation of the figure indicates that the model behaves better then the stacked LSTM model. Similar truth conveys the error report in Figure 30 which shows peak time error close to 0%, mean, peak and peek-to-peek error in the range of 10% and standard deviation error in the range of 20%. The minimum value has a significant error, at least in three prediction days.

The generation prediction is provided only for XGBoost model in Figure 30. The generation prediction of overall generation at all prosumers in the pilot group is not perfect. We will need to check it against single user prediction and try to predict the aggregated from there.
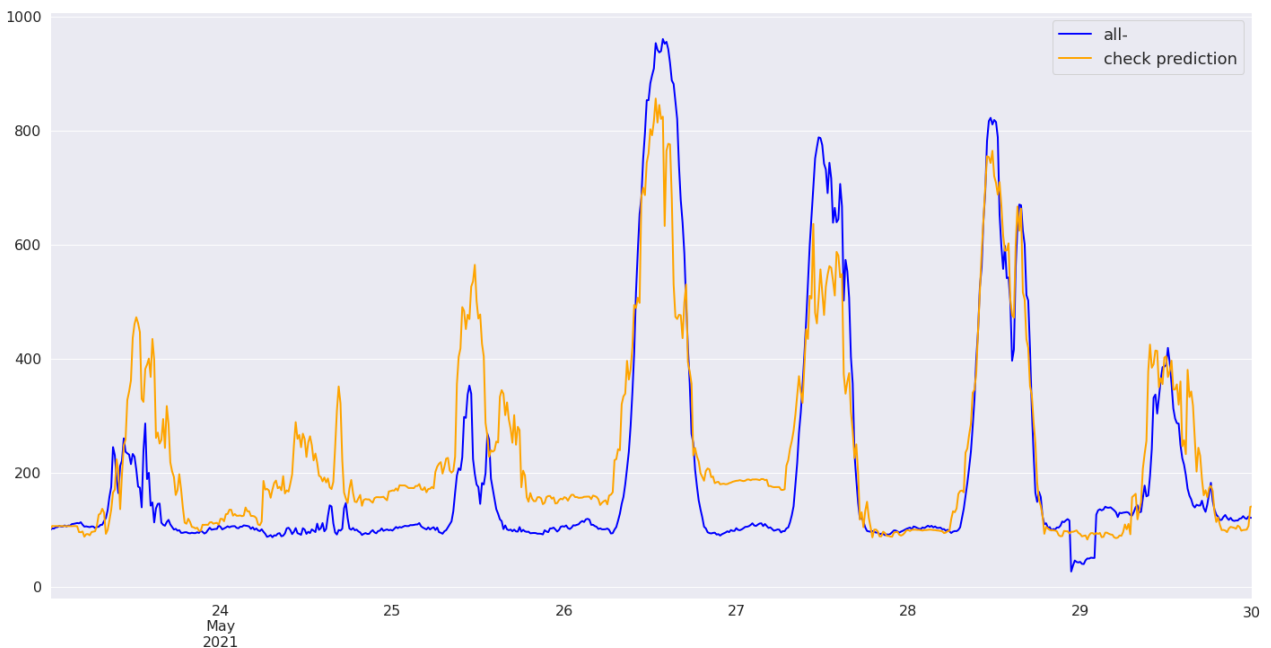


Figure 30: XGBoost generation prediction

The baseline shows that the predictions at a level of a few hundreds of consumers are possible, though not completely accurate. The predictions based on the data collected in 2020 and 2021 are even harder, since the patterns of household consumption usage are very diverse, due to COVID-19 pandemic. In the next period the predictions will need to be scaled down to a level of a single prosumer.

##### 5.3.2.4 **Thermal modelling**

The thermal modelling involves studying of the household thermal response to outside temperature and of the household thermal lag. In the first phase only the thermal response has been studied. In Figure 31 the thermal response of the piloting group of households in the CPT project is presented. The data in the figure is based on smart metering of more than 750 pilot users. In the figure, daily averages for year 2018 are presented with respect to the daily temperature average. The average daily consumption is in the range of 30kW to more than 70kW. The average daily temperature is in the range of -13 to +25°C. What can be seen in the figure is that both the thermal dependency for lower and higher temperatures exist. At lower temperatures, the consumption grows due to heating at some of the households. At higher temperatures the consumption also increases and the graph gets a typical "hockey stick" look, due to the cooling in some of the households. In contrast to the Finish example in **Error! Reference source not found.**, the bottom figure shows the increase of the consumption after the tipping point temperature of 18°C, which is a rough estimation given in (Borgeson, 2013). The thermal response observed on daily averages corresponds with the thermal response evaluated during flexibility events as presented in Figure 23.
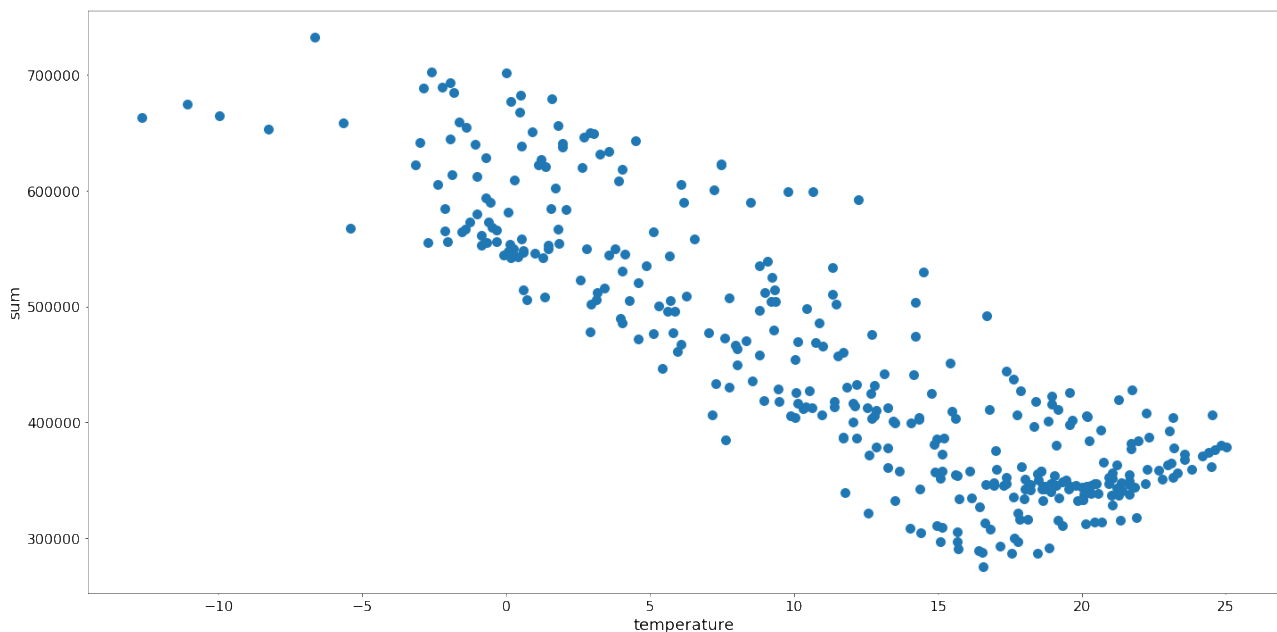


Figure 31: Thermal response of CPK project pilot group (2018)

The thermal response can be modelled with linear regression models as shown in the equation below. For basic modelling we have used a simple linear regression model as is provided by sci-kit learn, a Python toolkit for predictive data analysis[2]. The sci-kit Linear Regression module fits a linear model with coefficients $\beta = (\beta_0, …, \beta_n)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

$$y = b_0 + b_1 x_{i1} + … + b_n x_{in} + e_i \tag{8}$$

In the first modelling approximation only two parameters have been modelled, namely $\beta_0$ and $\beta_1$. Sample thermal responses are shown in Figure 32. The figures on the left hand side, from top to bottom:

- the upper figure presents clean, low temperature thermal response. The household uses electrical energy for heating only. The average daily consumption is up to 4kW maximum.

- the second figure presents a more scattered consumption with clear heating characteristics at low temperatures and cooling characteristics at high temperatures. In comparison to top figure example, more scattered consumptions could indicate an older house with more diverse thermal response.

- the third figure again indicates quite a scattered response. Two average consumption patterns are present at lower temperatures. The upper, indicating usage of electricity for heating and the lower one,

---

[2] See sci-kit learn home page for details: https://scikit-learn.org/stable/index.html

when some other kind of heating seems to be used. The higher temperature consumption response is weak.

- the last figure presents a response with no thermal dependency, where the consumption is the same regardless of the temperature.
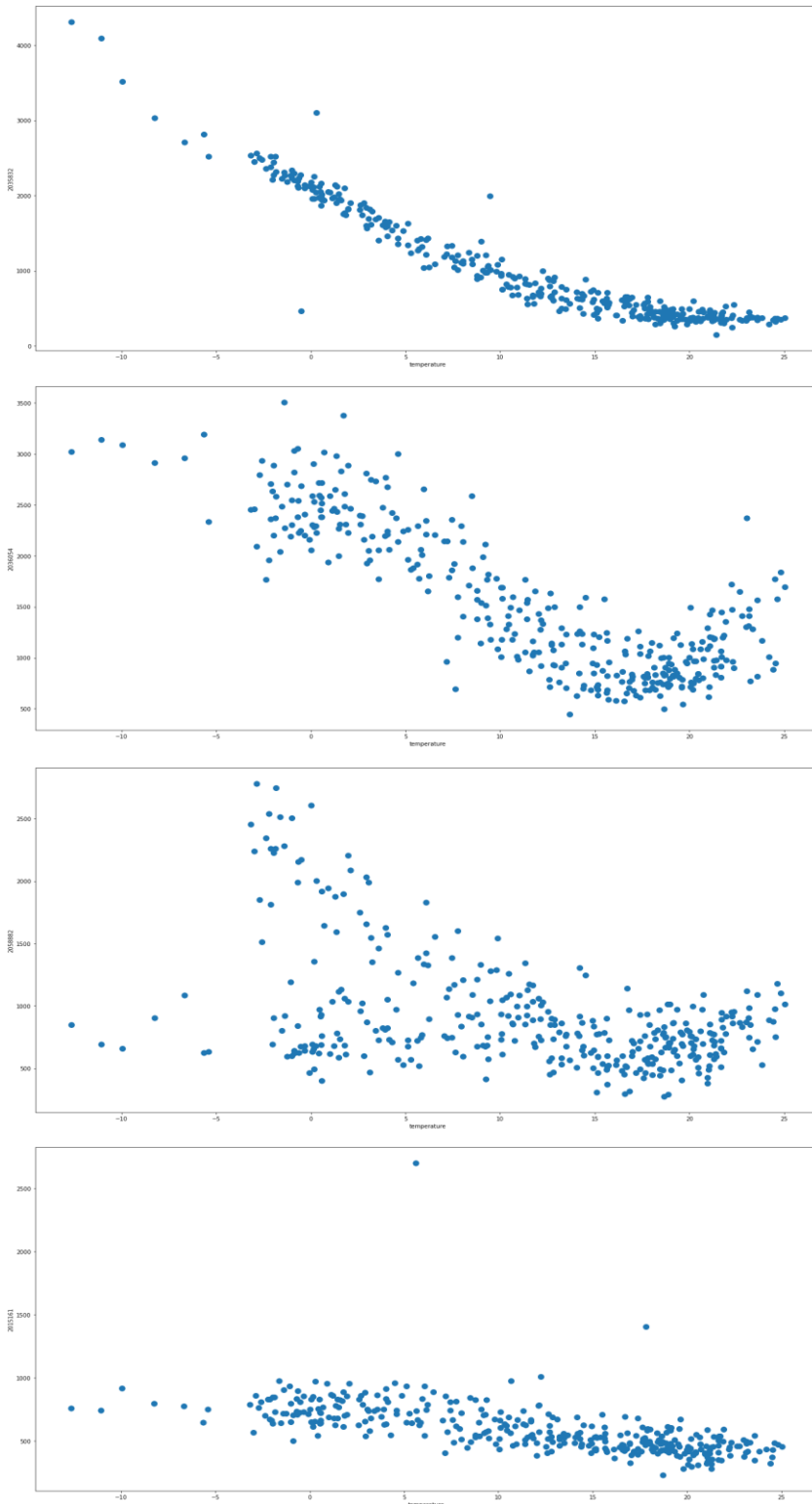


| | Intercept | Slope |
|---|---|---|
| 562 | 5548.11 | −263.64 |
| 266 | 4469.68 | −206.00 |
| 507 | 4711.19 | −202.05 |
| 613 | 3759.46 | −159.48 |
| 667 | 3843.38 | −152.12 |
| 623 | 3484.30 | −147.46 |
| 70 | 3387.03 | −141.22 |
| 596 | 3403.16 | −140.85 |
| 500 | 3686.79 | −138.50 |
| 589 | 3502.13 | −135.91 |
| 495 | 3109.77 | −130.75 |
| 605 | 3113.58 | −129.92 |
| 6 | 2922.78 | −124.50 |
| 143 | 2832.51 | −121.90 |
| 580 | 2985.39 | −121.75 |
| 38 | 2482.62 | −117.75 |
| 325 | 2933.40 | −116.86 |
| 469 | 2655.09 | −116.09 |
| 420 | 2777.33 | −112.35 |
| 719 | 2836.41 | −109.74 |
| 675 | 2868.74 | −107.34 |
| 160 | 2460.73 | −107.27 |
| 183 | 2658.02 | −106.70 |
| 188 | 2525.21 | −103.61 |
| 716 | 2500.81 | −102.48 |
| 362 | 2290.58 | −101.26 |
| 730 | 2453.79 | −100.83 |
| 751 | 2238.58 | −96.85 |
| 2 | 2251.27 | −96.08 |
| 595 | 2626.04 | −93.33 |

Figure 32: Individual households temperature dependencies

On the right-hand size of Figure 32, the top 30 $\beta_1$-parameter households are presented. The $\beta_1$ parameter denotes the slope of the linear regression line through the data. The top household uses exhibit an increase of 250W for every one degree Celcius less, according to the linear regression model. The $\beta_0$ parameter denotes an interception point with y axis, roughly indicating the top daily average consumption.
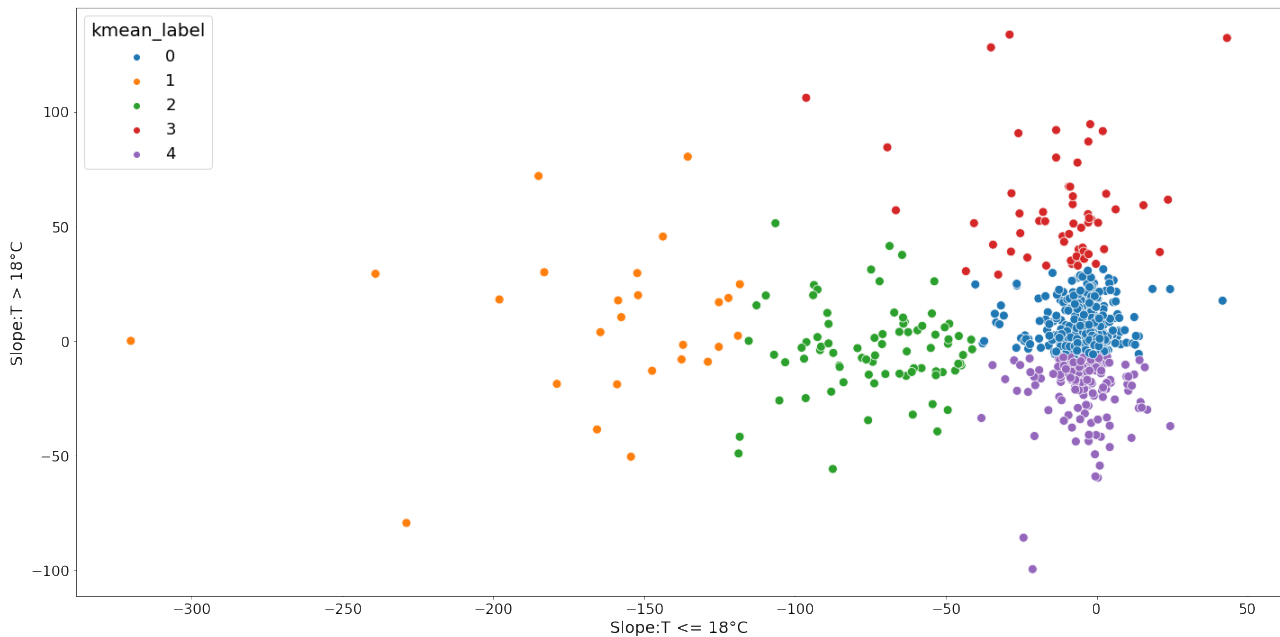


Figure 33: Clustering of the CPT piloting group per thermal response below vs. above tipping point (18°C) in year 2018

| kmean_label | Number | Share [%] | Consumption/year [kW] | Consumption [%] |
|---|---|---|---|---|
| 0 | 171 | 22.83 | 35184.98 | 21.50 |
| 1 | 25 | 3.34 | 14403.28 | 8.80 |
| 2 | 82 | 10.95 | 29407.64 | 17.97 |
| 3 | 411 | 54.87 | 66649.84 | 40.72 |
| 4 | 60 | 8.01 | 18028.97 | 11.02 |

Figure 34: Piloting users consumption cluster information

To further model the household consumptions, they have been split at the tipping point of 18°C. In this way, average daily consumptions at temperatures higher and lower than 18°C were grouped. Each group has been modelled with the linear regression strategy, employing a two-parameter model, as has been explained previously. The modelling has led to two group of parameters. For higher temperatures, the $\beta_{1h}$ parameter is expected to be slightly negative when no cooling is present and higher, when the household uses cooling systems. For lower temperatures, the $\beta_{1l}$ parameter is expected to be negative, since consumption should rise when the temperature is dropping. A few combinations of both $\beta_1$ parameter can be expected:

- $\beta_{1l} \approx 0$, $\beta_{1h} > 0$: the household uses electricity for cooling,

- $\beta_{1l} \approx 0$, $\beta_{1h} \approx 0$: the household is not thermal dependent,

- $\beta_{1l} < 0$, $\beta_{1h} \approx 0$: the household uses electricity for heating,

- $\beta_{1l} < 0$, $\beta_{1h} > 0$: the household uses electricity both for heating and cooling.

Based on these assumptions, Figure 33 presents the clustering of the households according to $\beta_{1l}$ (Slope ≤ 18°C) and $\beta_{1h}$ (Slope > 18°C). Clustering has been done with sci-kit learn K-means clustering algorithm. Five clusters were used. The orange cluster outpoints households with large heating dependency. The cooling dependency is either present or not. The green cluster indicates medium heating dependency and similar cooling dependency to the orange cluster. It could be that the cluster uses both electricity and other energy

sources for heating. The red and blue cluster exhibit similar low thermal dependency. The red one seems to be more constant in energy consumption and the blue one shows similar low heating dependency but even lower consumption at the temperatures above the tipping point. The violet cluster indicates low heating and high cooling dependencies. The households in this cluster seem to use alternative energy sources for heating exclusively and energy for cooling only. As such, they are a nice group with a potential for future energy installations (heat pumps, photovoltaics, etc.). Finally, the k-means clustering is not as efficient as envisioned. The clusters are not close to intended split in four clusters as were presented before. More experimentation is neded with additional parameters or different method of clustering.

In Figure 34, basic information about the clusters is provided. In the first column, the strength of the cluster is given. The red (3) and blue (0) clusters are the strongest in number. They present more than two thirds of the population. Their thermal response does not promise much potential in future DR programmes. The orange (1), green (2) and violet (4) clusters, show more potential, either for direct participation in flexibility services or as a potential to receive energy advice how to alter existing energy settings. Combined, these clusters consume almost 38% of the overall consumption.

### 5.3.2.5    Federated learning: practical example in PySyft[3]

In this simple example we have two data owners and a data scientist. Both data owners generate sensitive data. We pretend that the data was actually acquired through some measurements from data owners by adding some noise.

The data scientist wants to find out what the function that generated the data was, without ever getting access to actual data. Data scientist achieves this through federated learning.

Data used in this presentation is going to be quite simple - function generating data is linear:

$$y = f(\vec{x}) = [1,2,\dots,7] * \vec{x} + 42 \tag{9}$$

or equivalently

$$y = f(x_1, \dots, x_7) = 1 * x_1 + 2 * x_2 + \dots + 7 * x_7 + 42 \tag{10}$$

If data was centralized, the data scientist could build a simple neural network with only one linear layer. Resulting model would look something like [$w_1$, $w_2$, . . . , w8] and during training it should converge to some approximation of actual function coefficients [1, 2, 3, 4, 5, 6, 7, 42].

Without access to actual data, the data scientist describes the model and lets data owners train that model on their private data. After both data owners are done building the model, they send it back to the data scientist. Data scientist then combines both models by averaging individual weights. This combined model is a better approximation for the function that generated both data owner's data than any individual model.

Explicitly: let's say that the model trained on the first data owner's data is described by [$w_{1,1}$, $w_{1,2}$, . . . , $w_{1,8}$] and the model trained on the second data owner's data is described by [$w_{2,1}$, $w_{2,2}$, . . . , $w_{2,8}$]. The combined model, calculated by the data scientist is then described by

$$\left[\frac{w_{1,1} + w_{2,1}}{2}, \frac{w_{1,2} + w_{2,2}}{2}, \dots, \frac{w_{1,8} + w_{2,8}}{2}\right] \tag{11}$$

We can see that this technique is easily adapted to situations with more than two data owners.

To further simplify the procedure, we shall avoid using differential privacy techniques.

### 5.3.2.5.1    PySyft

Pysyft is a python library offering implementations of certain privacy tools/techniques: federated learning, differential privacy, secure multi-party computations and homomorphic encryption.

---

[3] PySyft, A library for computing on data you do not own and cannot see. See PySyft home page for more information: https://github.com/OpenMined/PySyft

#### 5.3.2.5.2    Code

To follow the example three notebooks need to be started: 'DataOwner1.ipynb', 'DataOwner2.ipynb' and 'Data-Scientist.ipynb'.

The following pages contain screenshots of the python notebooks. Under each screenshot are the names of the notebooks in which you have to enter the code from that screenshot.

##### 5.3.2.5.2.1    Import libraries

```
1  import syft as sy
2  import torch
3  import numpy as np
4  import random
```

Figure 35: Run at DataOwner1, DataOwner2 and DataScientist

##### 5.3.2.5.2.2    Launch Duet server 1 and send Duet Server ID to the Data Scientist and wait for him to connect

```
1  duet = sy.launch_duet()
```



Figure 36: At DataOwner1: Launch Duet server 1 and send Duet Server ID to the Data Scientist and wait for him to connect

#### 5.3.2.5.2.3 Launch Duet server 2 and send Duet Server ID to the Data Scientist and wait for him to connect

```
1  duet = sy.launch_duet()
```



Figure 37: At DataOwner2: Launch Duet server 2 and send Duet Server ID to the Data Scientist and wait for him to connect

#### 5.3.2.5.2.4 Connect to the first Data Owner's duet server and send back Duet Client ID

```
1  duet1 = sy.join_duet("96f8b5d791c69e2f78fb0f64d93bc1c6")
```



Figure 38: At DataScientist: Connect to the first Data Owner's duet server and send back Duet Client ID

#### 5.3.2.5.2.5 Connect to the second Data Owner's duet server and send back Duet Client ID

```
1 duet2 = sy.join_duet("bbd1e5490378c0bc36faf49be7e4e780")
```

# Duet

🎺 🎻 ♪♪♪ Joining Duet ♪♪♪ 🎻 🎹

♪♪♪ > DISCLAIMER: **Duet is an experimental feature currently in beta.**
♪♪♪ > **Use at your own risk.**

> ❤ **Love** Duet? Please consider supporting our community!
> https://github.com/sponsors/OpenMined

♪♪♪ > **Punching through firewall to OpenGrid Network Node at:**
♪♪♪ > **http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000**
♪♪♪ >
♪♪♪ > **...waiting for response from OpenGrid Network...**
♪♪♪ > **DONE!**

♪♪♪ > STEP 1: Send the following Duet Client ID to your duet partner!
♪♪♪ > Duet Client ID: 654a42e59af53249bcd205671bcbcc24

♪♪♪ > ...waiting for partner to connect...

Figure 39: At DataScientist: Connect to the second Data Owner's duet server and send back Duet Client ID

#### 5.3.2.5.2.6 Accept Data Scientist as a client by entering their Duet Client ID

```
1 duet = sy.launch_duet()
```

# Duet

🎺 🎻 ♪♪♪ Starting Duet ♪♪♪ 🎻 🎹

♪♪♪ > DISCLAIMER: **Duet is an experimental feature currently in beta.**
♪♪♪ > **Use at your own risk.**

> ❤ **Love** Duet? Please consider supporting our community!
> https://github.com/sponsors/OpenMined

♪♪♪ > **Punching through firewall to OpenGrid Network Node at:**
♪♪♪ > **http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000**
♪♪♪ >
♪♪♪ > **...waiting for response from OpenGrid Network...**
♪♪♪ > **DONE!**
♪♪♪ > Duet Server ID: **96f8b5d791c69e2f78fb0f64d93bc1c6**

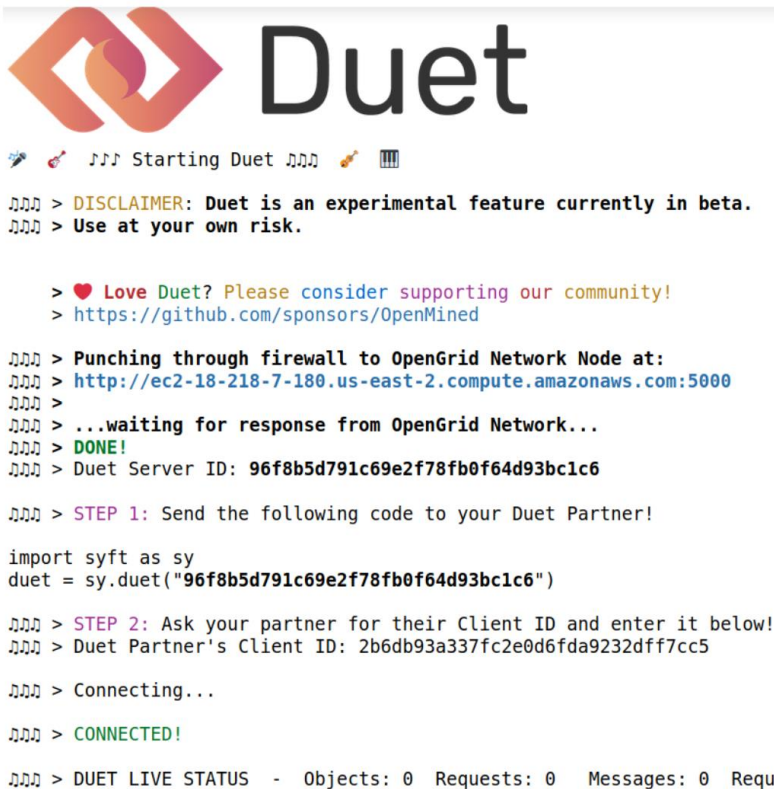♪♪♪ > STEP 1: Send the following code to your Duet Partner!

import syft as sy
duet = sy.duet("**96f8b5d791c69e2f78fb0f64d93bc1c6**")

♪♪♪ > STEP 2: Ask your partner for their Client ID and enter it below!

♪♪♪ > Duet Partner's Client ID: 2b6db93a337fc2e0d6fda9232dff7cc5

Figure 40: At DataOwner1 and DataOwner2: Accept Data Scientist as a client by entering their Duet Client ID

#### 5.3.2.5.2.7 Connection between Data Owner and Data Scientist successfully established



Figure 41: At DataOwner1 and DataOwner2: Connection between Data Owner and Data Scientist successfully established

#### 5.3.2.5.2.8 Define data generating functions

```python
def dot(x, y):
    r = 0
    for i in range(len(x)):
        r += x[i]*y[i]
    return r

#f(x1, x2, x3, ..., xn) = 1*x1 + 2*x2 + 3*x3 + ... + n*xn + 42
def f(x):
    W = [i for i in range(1, len(x)+1)]
    return dot(x, W) + 42

def f_noisy(x):
    return f(x) + random.random()

def gen_data(dim, n):
    datapoint = []
    data = []
    for i in range(n):
        datapoint = [50*random.random()-25 for _ in range(dim)]
        data.append(datapoint)
    return data

def gen_target(data):
    return [f_noisy(x) for x in data]
```

Figure 42: At DataOwner1 and DataOwner2: Define data generating functions

#### 5.3.2.5.2.9 Generate data using those functions - 1000 samples, 7 features

```
1  in_dim = 7
2  out_dim = 1
3  N = 1000
4
5  data = gen_data(in_dim, N)
6  target = gen_target(data)
7  data = torch.FloatTensor(np.array(data))
8  target = torch.FloatTensor(np.array(target).reshape(-1, 1))
```

```
1  print(data[0:5])
2  print(target[0:5])
```

```
tensor([[   7.5745,    1.5919,  -12.5249,  -20.7874,    5.2903,  -11.8149,    7.0543],
        [ -18.3937,   24.1931,   -5.9781,   -6.5116,   21.3468,   21.8862,   14.0414],
        [  14.9093,   15.5951,   24.8717,   -6.0548,  -17.8292,   -2.5904,   20.2570],
        [  17.0236,    6.3571,   13.9933,  -20.3199,   23.2245,   22.5704,   21.2761],
        [   3.4338,   -1.2499,    4.2079,   11.3048,  -24.1069,   17.5111,    5.2745]])
tensor([[-62.5342],
        [365.1499],
        [175.9734],
        [433.5750],
        [122.5972]])
```

Figure 43: At DataOwner1 and DataOwner2: Generate data using those functions - 1000 samples, 7 features

#### 5.3.2.5.2.10 Upload pointers to 'data' and 'target' to Duet server

Pointers on Duet server are memory addresses to Data Owner's data. They contain zero information about data they are pointing to.

```
1  data = data.tag("DO1 data")
2  data = data.describe("Dataset of " + str(N) + " samples, " + str(in_dim) + " features")
3  data_ptr = data.send(duet, pointable=True)
```

```
1  target = target.tag("DO1 target")
2  target = target.describe(str(N) + " ground truths, " + str(out_dim) + " features")
3  target_ptr = target.send(duet, pointable=True)
```

Figure 44: At DataOwner1 and DataOwner2: Upload pointers to 'data' and 'target' to Duet server

#### 5.3.2.5.2.11 Accept all requests

Duet client (Data Scientist) is able to get data that pointers are pointing to through requests. Requests can be manually accepted or denied. While developing we accept all requests for practical reasons.

```
1  duet.requests.add_handler(action="accept", print_local=True)
```

Figure 45: At DataOwner1 and DataOwner2: Accept all requests

#### 5.3.2.5.2.12 Define machine learning model

Data in this example is so simple we only need one linear layer. PySyft inherits largely from PyTorch so it's really similar to defining ML model in PyTorch. The main difference is that it works with pointers to objects instead of directly with objects.

```
1  in_dim = 7
2  out_dim = 1
```

```
1  class SyNet(sy.Module):
2      def __init__(self, torch_ref):
3          super(SyNet, self).__init__(torch_ref=torch_ref)
4          self.linear = self.torch_ref.nn.Linear(in_dim, out_dim)
5
6      def forward(self, x):
7          x = self.linear(x)
8          return x
```

Figure 46: At DataScientist: Define machine learning model

#### 5.3.2.5.2.13 Define train function

The main loop runs on our machine but actual work (computing loss, gradients) is done on the Data Owner's machine. This loop basically tells Data Owners machine what to do through pointers. As Data Scientists we

need feedback about losses, to see if training is actually converging. We do this through requesting the actual value of pointers to loss value. Requests require some back-and-forth communication and are bottlenecking whole process, that's why we only request every 100th loss value.

```python
def train(iterations, model_ptr, torch_ptr, optim_ptr, data_ptr, target_ptr):
    losses = []

    for i in range(iterations):

        optim_ptr.zero_grad()

        output_ptr = model_ptr(data_ptr)

        loss_ptr = torch_ptr.nn.functional.mse_loss(output_ptr, target_ptr)

        loss_item_ptr = loss_ptr.item()

        if i % 100 == 0:
            loss_value = loss_item_ptr.get(request_block=True)
            losses.append(loss_value)
            print("Epoch", i, "loss", loss_value)

        loss_ptr.backward()

        optim_ptr.step()

    return losses
```

Figure 47: At DataScientist: Define train function

### 5.3.2.5.2.14    Save pointers to 'data' and 'target' from both Data Owners from Duet.store

```python
duet1.store.pandas
```

| | ID | Tags | Description | object_type |
|---|---|---|---|---|
| 0 | <UID: 7cbfeb7326ec4ee4a0b651533aaa9961> | [DO1 data] | Dataset of 100 samples, 7 features | <class 'torch.Tensor'> |
| 1 | <UID: 4701e631039040c991d4959dfa035d0a> | [DO1 target] | 100 ground truths, 1 features | <class 'torch.Tensor'> |

```python
duet2.store.pandas
```

| | ID | Tags | Description | object_type |
|---|---|---|---|---|
| 0 | <UID: ba19a150cae948358f60bed1c31a39d2> | [DO2 data] | Dataset of 100 samples, 7 features | <class 'torch.Tensor'> |
| 1 | <UID: 330060b8e2ba41f6850a7b0fc36e34c2> | [DO2 target] | 100 ground truths, 1 features | <class 'torch.Tensor'> |

```python
data1_ptr = duet1.store[0]
target1_ptr = duet1.store[1]

data2_ptr = duet2.store[0]
target2_ptr = duet2.store[1]

print(data1_ptr)
print(target1_ptr)
```
```
<syft.proxy.torch.TensorPointer object at 0x7fdc49b3c3a0>
<syft.proxy.torch.TensorPointer object at 0x7fdc49b44970>
```

Figure 48: At DataScientist: Save pointers to 'data' and 'target' from both Data Owners from Duet.store

### 5.3.2.5.2.15    Get pointers to additional information needed in our train function

Firstly, we need pointers to base models. We do this by compiling our previously defined model locally and sending it to the Data Owner. The value returned by that process is a pointer to the remote model. We also need pointers to remote Torches and optimisers.

```
1  local_model1 = SyNet(torch)
2  remote_model1_ptr = local_model1.send(duet1)
3  remote_torch1_ptr = duet1.torch
4  remote_optim1_ptr = remote_torch1_ptr.optim.Adam(params=remote_model1_ptr.parameters(), lr=0.1)
```

```
1  local_model2 = SyNet(torch)
2  remote_model2_ptr = local_model2.send(duet2)
3  remote_torch2_ptr = duet2.torch
4  remote_optim2_ptr = remote_torch2_ptr.optim.Adam(params = remote_model2_ptr.parameters(), lr = 0.1)
```

Figure 49: At DataScientist: Get pointers to additional information needed in our train function

### 5.3.2.5.2.16    Run training loop on first Data Owner's data

```
1  iteration = 1500
2  losses = train(iteration, remote_model1_ptr, remote_torch1_ptr, remote_optim1_ptr, data1_ptr, target1_ptr)
```

```
Epoch 0 loss 32351.53515625
Epoch 100 loss 1088.52294921875
Epoch 200 loss 602.0419311523438
Epoch 300 loss 354.1318054199219
Epoch 400 loss 192.7604522705078
Epoch 500 loss 96.97334289550781
Epoch 600 loss 44.94887161254883
Epoch 700 loss 19.143888473510742
Epoch 800 loss 7.488239765167236
Epoch 900 loss 2.707026481628418
Epoch 1000 loss 0.9301570057868958
Epoch 1100 loss 0.33328601717948914
Epoch 1200 loss 0.1525404155254364
Epoch 1300 loss 0.10334277153015137
Epoch 1400 loss 0.09135233610868454
```

Figure 50: At DataScientist: Run training loop on first Data Owner's data

### 5.3.2.5.2.17    Run training loop on second Data Owner's data

```
1  iteration = 1000
2  losses = train(iteration, remote_model2_ptr, remote_torch2_ptr, remote_optim2_ptr, data2_ptr, target2_ptr)
```

```
Epoch 0 loss 28302.818359375
Epoch 100 loss 948.8074340820312
Epoch 200 loss 459.63250732421875
Epoch 300 loss 216.29408264160156
Epoch 400 loss 91.88968658447266
Epoch 500 loss 34.7899055480957
Epoch 600 loss 11.662283897399902
Epoch 700 loss 3.474759578704834
Epoch 800 loss 0.9547136425971985
Epoch 900 loss 0.28255370259284973
```

Figure 51: at DataScientist: Run training loop on second Data Owner's data

### 5.3.2.5.2.18    Send requests to get content of pointers to models trained on individual Data Owner's data

```
1  params1 = remote_model1_ptr.parameters().get(request_block=True)
2  params2 = remote_model2_ptr.parameters().get(request_block=True)
3
4  params1 = np.append(params1[0].detach().numpy(), params1[1].detach().numpy())
5  params2 = np.append(params2[0].detach().numpy(), params2[1].detach().numpy())
```

```
1  print(params1)
2  print(params2)
```

```
[ 0.99712676  2.0022209   2.9997318   4.002106    5.0015187   5.9974766
  7.0004635  42.5041     ]
[ 1.0008566   1.9986084   3.0013392   4.0008726   5.001893    5.9982076
  7.0012107  42.513268  ]
```

Figure 52: At DataScientist: Send requests to get content of pointers to models trained on individual Data Owner's data

### 5.3.2.5.2.19    Measure the error of individual trained model

As described in the previous section, both models should look close to [1, 2, . . . , 7, 42]. We define model error as its distance from [1,2,...,7,42].

```
1  exact_parameters = np.array([1,2,3,4,5,6,7,42])
2
3  err1 = np.linalg.norm(params1 - exact_parameters)
4  err2 = np.linalg.norm(params2 - exact_parameters)
5
6  print('Error1: ', err1)
7  print('Error2: ', err2)
```

```
Error1:  0.12579936909934394
Error2:  0.05482733131874097
```

Figure 53: At DataScientist: Measure the error of individual trained model

**5.3.2.5.2.20 Combine both models through averaging individual weights and calculate error of combined model**

```
1  combined_params = (params1 + params2) / 2
2
3  err3 = np.linalg.norm(combined_params - exact_parameters)
4
5  print(err3)
```

```
0.03669843264261282
```

Figure 54: At DataScientist: Combine both models through averaging individual weights and calculate error of combined model

**5.3.2.5.2.21 Combined model should be better than individual models**

Combined model error is smaller than individual model error

```
1  err3 < err1 and err3 < err2
```

```
True
```

Figure 55: At DataScientist: Combined model should be better than individual models

# 6   Conclusion

This deliverable has documented the intermediate work on the digital twin repository module of the iFLEX Framework. The digital twin module consists of several models supporting forecast and response modelling of the apartment buildings and prosumer households. In this phase, intermediate models were presented, as well the project baselines as a starting point of the project work. Experiences and data gained from the first phase of the project pilot have been used to improve the models from the initial versions. Also, in this phase of the project, multiple methods to model the buildings have been tried. The data and system information will help to improve the models and foster their application and usage in overall iFLEX framework solutions.

# 7 List of figures and tables

## 7.1 Figures

## 7.2  Tables

# 8   References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*.

Bengio, Y., Deleu, T., Rahaman, N., Ke, N. R., Lachapelle, S., Bilaniuk, O., … Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms. *ArXiv*.

Borgeson, S. D. (2013). *Targeted Efficiency: Using Customer Meter Data to Improve Efficiency Program Outcomes*. UC Berkeley Electronic Theses and Dissertations.

Chakhchoukh, Y., Panciatici, P., & Mili, L. (2011). Electric load forecasting based on statistical robust methods. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2010.2080325

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Chollet, F. (2018). *Deep Learning with Python and Keras. MITP-Verlags GmbH & Co. KG*.

Chou, J. S., & Bui, D. K. (2014). Modeling heating and cooling loads by artificial intelligence for energy-efficient building design. *Energy and Buildings*. https://doi.org/10.1016/j.enbuild.2014.07.036

d'Avila Garcez, A., & Lamb, L. C. (2020). Neurosymbolic ai: The 3rd wave. *ArXiv*.

De Wilde, P., Martinez-Ortiz, C., Pearson, D., Beynon, I., Beck, M., & Barlow, N. (2013). Building simulation approaches for the training of automated data analysis tools in building energy management. *Advanced Engineering Informatics*. https://doi.org/10.1016/j.aei.2013.05.001

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP.2013.6638947

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*. https://doi.org/10.1038/s41586-020-2649-2

He, K., Zhang, X., Ren, S., & Sun, J. (2016). ResNet. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://doi.org/10.1162/neco.1997.9.8.1735

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice. *Principles of Optimal Design*.

Jordan, M. I. (1986). *Serial order: A parallel distributed processing approach. ICS Report*.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). *Efficient and Robust Automated Machine Learning* https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Kannari, L., Kiljander, J., Piira, K., Piippo, J., & Koponen, P. (2021). Building Heat Demand Forecasting by Training a Common Machine Learning Model with Physics-Based Simulator. *Forecasting*. https://doi.org/10.3390/forecast3020019

Ko, C. N., & Lee, C. M. (2013). Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter. *Energy*. https://doi.org/10.1016/j.energy.2012.11.015

Koponen, P., Niska, H., & Mutanen, A. (2019). Mitigating the weaknesses of machine learning in short-term forecasting of aggregated power system active loads. In *IEEE International Conference on Industrial Informatics (INDIN)*. https://doi.org/10.1109/INDIN41052.2019.8972182

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python*

*in Science Conference.*

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics.* https://doi.org/10.3389/fnbot.2013.00021

Nielsen, A. (2020). *Practical Time Series Analysis: Prediction with Statistics & Machine Learning. Pesquisa Operacional.*

OpenAI. (2020). AlphaFold : a solution to a 50-year-old grand challenge in biology. *Https://Deepmind.Com/Blog.*

Ovdes, M. G., Souvent Ovdes, A., & Ovdes, N. S. (2020). Evaluating temperature-dependent consumers in a demand response program using machine learning. In *IEEE PES Innovative Smart Grid Technologies Conference Europe.* https://doi.org/10.1109/ISGT-Europe47291.2020.9248950

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research.*

Srivastava, A. K., Pandey, A. S., & Singh, D. (2016). Short-term load forecasting methods: A review. In *International Conference on Emerging Trends in Electrical, Electronics and Sustainable Energy Systems, ICETEESES 2016.* https://doi.org/10.1109/ICETEESES.2016.7581373

Turhan, C., Kazanasmaz, T., Uygun, I. E., Ekmen, K. E., & Akkurt, G. G. (2014). Comparative study of a building energy performance software (KEP-IYTE-ESS) and ANN-based building heat load estimation. *Energy and Buildings.* https://doi.org/10.1016/j.enbuild.2014.09.026

Zheng, J., Xu, C., Zhang, Z., & Li, X. (2017). Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network. In *2017 51st Annual Conference on Information Sciences and Systems, CISS 2017.* https://doi.org/10.1109/CISS.2017.7926112

## 9    Appendix: Digital twin Jira requirements

### [IF-68] Apartment building flexibility model

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | The flexibility model forecast part of the building energy consumption that is flexible and available to be used in flexibility services. |
| **Source:** | HLUC-3, PUC-6, PUC-8, PUC-9, PUC-10 |
| **Pilot Finland:** | Phase one |
| **Pilot Greece:** | Not applicable |
| **Pilot Slovenia:** | Not applicable |

**Description**

Define an apartment building flexibility model and forecast available flexibility in a specified time frame.

### [IF-67] Apartment building electricity model

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | Electricity consumption in the building together with the district heating model provides information on total energy consumption in the building. |
| **Source:** | HLUC- 3, PUC-8, PUC-10 |
| **Pilot Finland:** | Phase one |
| **Pilot Greece:** | Not applicable |
| **Pilot Slovenia:** | Not applicable |

**Description**

The apartment building electricity model provides forecast for electricity consumption in a building.

**[IF-66] Apartment building district heating model**

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | District heating supplies part of energy to the apartment building and is important to understand general thermal conditions in the building. The model is used in combination with electricity consumption model to forecast a total energy and electricity consumption in the building. |
| **Source:** | HLUC-3, PUC-8, PUC-10 |
| **Pilot Finland:** | Phase one |
| **Pilot Greece:** | Not applicable |
| **Pilot Slovenia:** | Not applicable |

**Description**

Create district heating model and provide a district heating forecast for a specified period.

**[IF-65] Household occupant behaviour model**

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | Many of the loads in the household are under direct consumer control. To be able to predict a general household consumption more accurate an occupant behavior model is needed. |
| **Source:** | PUC-4, PUC-5, PUC-6, PUC-8, PUC-10 |
| **Pilot Finland:** | Not applicable |
| **Pilot Greece:** | Not applicable |
| **Pilot Slovenia:** | Phase two |

**Description**

Define an occupant behavior model for better prediction of the household consumption.

## [IF-64] Household flexibility model

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | Flexibility forecast is needed for planning of various flexibility services, from self-balancing to implicit, price based, and explicit flexibility services. The flexibility can be reported to flexibility services management for better planning and optimization. |
| **Source:** | PUC-8, PUC-4, PUC-5, PUC-6, PUC-10 |
| **Pilot Finland:** | Not applicable |
| **Pilot Greece:** | Phase two |
| **Pilot Slovenia:** | Phase one |

**Description**

Create a household flexibility model. The model is able to provide a forecast of an available household flexibility in a specified time-frame.

## [IF-63] Household electricity model

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | A basic service for iFLEX Assistant. The service forecast the consumption and generation so the other iFLEX Assistant components could plan for and automate the household consumption as well evaluate future participation in flexibility services. |
| **Source:** | HLUC-1, PUC-4, PUC-6, PUC-8, PUC-10 |
| **Pilot Finland:** | Not applicable |
| **Pilot Greece:** | Phase two |
| **Pilot Slovenia:** | Phase one |

**Description**

Create a household electricity model being able to forecast the household consumption and generation.

## [IF-62] Household thermal model

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Dusan Gabrijelcic | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | |
|---|---|
| **Rationale:** | A thermal model is essential to understand the household consumption and evaluate its future flexibility potential. |
| **Source:** | PUC-5 |
| **Pilot Finland:** | Not applicable |
| **Pilot Greece:** | Not applicable |
| **Pilot Slovenia:** | Phase one |

**Description**

Create a household thermal model and potential thermal lag.

## [IF-105] Data collection from demand response tests

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Non-functional | **Priority:** | Major |
| **Reporter:** | Jussi Kiljander | **Assignee:** | Jussi Kiljander |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | None | | |

| | | | | |
|---|---|---|---|---|
| **Issue Links:** | **Depend** | | | |
| | depends | IF-106 | Machine learning based apartment buil... | Open |
| | depends | IF-66 | FN-DTR-04 Apartment building district... | Validated |
| | depends | IF-67 | FN-DTR-05 Apartment buiding electrici... | Validated |
| **Rationale:** | In order to train and evaluate forecasting models and how they work during and after demand response events, measurement data from pilot is required. Measurement data includes district heating and electricity power measurements. | | | |
| **Pilot Finland:** | Phase one | | | |
| **Pilot Slovenia:** | Not applicable | | | |
| **Pilot Greece:** | Not applicable | | | |

Activate demand response events in different times of the day and different days of the week to improve dataset used for forecasting model training

### [IF-106] Machine learning based apartment building district heating and electricity flexibility models

| | |
|---|---|
| **Status:** | Open |
| **Project:** | iFlex Project |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

| | | | |
|---|---|---|---|
| **Type:** | Functional | **Priority:** | Major |
| **Reporter:** | Jussi Kiljander | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | DigitalTwinRepository | | |

| | | | |
|---|---|---|---|
| **Issue Links:** | **Depend** | | |
| | is depended by | IF-105 Data collection from demand response ... | Open |
| **Rationale:** | Machine learning can potentially outperform physics based models when representative data is available | | |
| **Pilot Finland:** | Phase two | | |
| **Pilot Greece:** | Not applicable | | |
| **Pilot Slovenia:** | Not applicable | | |

**Description**

Create model that forecasts available flexibility on district heating and electricity using machine learning and indoor temperature using physics based model. Dataset for available flexibilty should contain flexibility tests.