Forecasting electricity consumption using the Groupe E dataset

extended version

Authors: HEPIA (Yohann Perez, John White), KTH (Carlo Fischione), Groupe-E (Teo Brigljevic)

September 2025

Introduction

This document outlines the development and experimentation of a prototype machine learning model designed to predict electricity consumption. It is an extended version of the “Machine Learning with Groupe E dataset” document of the last progress report.

The document follows this structure: First, it introduces the dataset used throughout the project and then outlines the preprocessing steps. Next, it describes the feature engineering phase, which extracts useful and potentially actionable information from the dataset. After that, he presents the implemented DVC pipeline and the MLflow integration, which facilitates model experimentation and its reproducibility.

Finally, it concludes with a brief discussion of the experimental results and the limitations of the dataset and infrastructure.

Structure of the Groupe E dataset

The `swarm_smi_p.csv` file is a comma-separated value (csv) dataset provided by Groupe E containing electrical consumption measurements recorded by 1157 metering devices. It follows the format where each column represents a measuring device, while each row corresponds to a timestamp.

The records are timestamped at 15-minute intervals, covering the period from January 3, 2021, to December 30, 2022, which in our csv file corresponds to 69601 measures per device.

However, some measures may be empty or equal to zero, indicating either missing data for certain devices at specific times or measurement failures. This could result from temporary interruptions, or other factors affecting data collection.

The dataset (Figure 1) is complemented by additional files that provide details about the measuring devices and the infrastructures they are connected to. These files contain metadata, such as the technical specifications of the devices, their locations, and the types of infrastructures they monitor.

By cross-referencing the `swarm_smi_p.csv` file with these additional datasets, it should be possible to obtain insight about energy consumption behaviors and identify patterns across different infrastructures.

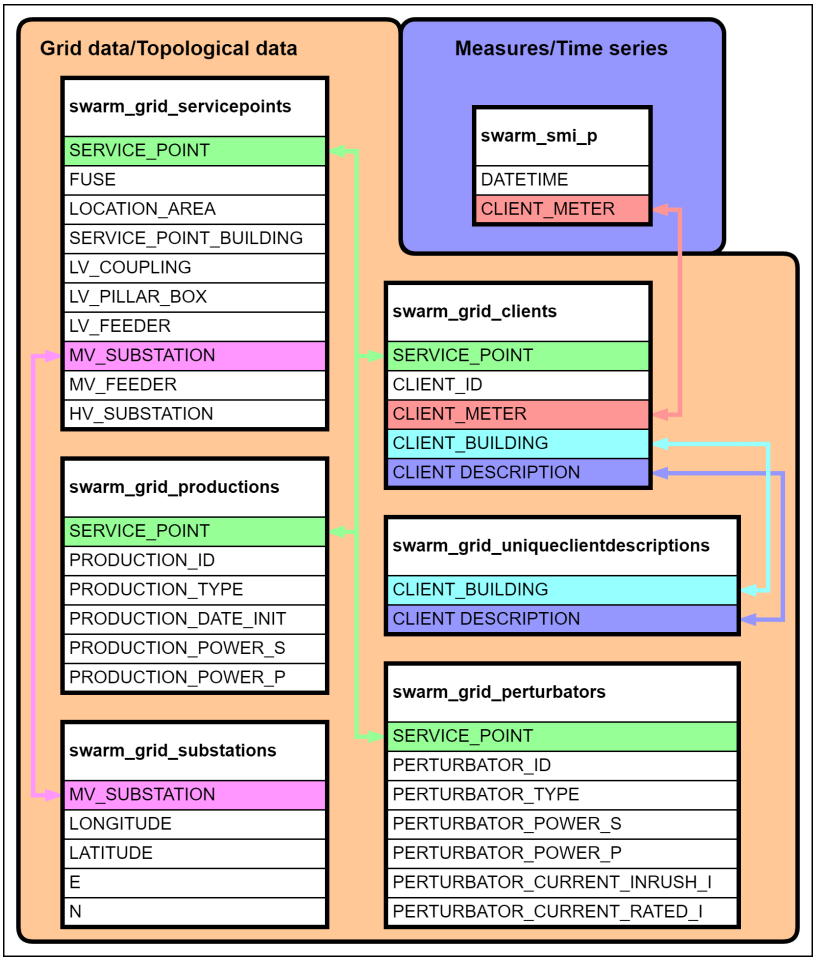


Figure 1 : Structure of the dataset as provided by Groupe E.

Data preprocessing

A data and device cleaning process has been implemented to ensure data quality and reliability. This process is based on three stages:

- **Elimination of incomplete data** : Devices with both more than 1% zero values and more than 8% missing values are excluded from the dataset. This is done to limit the impact of faulty devices and incomplete data.
- **Deletion of duplicate devices** : Duplicate devices are removed to avoid bias and redundancy errors
- **Device verification** : Devices that take measurements must only be cataloged once in the clients file. In other words, a device corresponds to a single entry in the clients file, and this must be verified.
- **Filling missing value** : Fill forward missing values, carrying the last valid observation forward, but only up to 4 consecutive missing values.

Creation of temporal series

To operate, the model needs a time series of 4 past values (lookback, in blue), then predicts a single 1 output value (horizon, in green). These lookback/horizon values are fully configurable, it can be any number. In this case it is designated as 4/1. As energy measurement equipment can break down or stop collecting data at any time, the measurements collected can be discontinuous: i.e. not present systematically for every quarter-hour. So the adopted approach consists of segmenting the data as soon as a missing value is detected, in order to avoid this discontinuity in the model's analysis of the time series.
In concrete terms, rather than interpolating these values or ignoring them, segmenting the sequences in such a way as to preserve continuity and coherence between them has been the solution adopted. In this way, each sub-sequence remains continuous and usable, without the introduction of biases associated with imputing values or approximating missing values.

Let's start with a normal perfect example to demonstrate how a series is segmented if it contains no missing values (Figure 2). Here is the series (in grey) and it will be segmented using 4 past values (in blue) and 1 output value (in green)

Creation of temporal series												
	1	2	3	4	5	6	7	8	9	10	11	12

Temporal series												
	1	2	3	4	5							
		2	3	4	5	6						
			3	4	5	6	7					
				4	5	6	7	8				
					5	6	7	8	9			
						6	7	8	9	10		
							7	8	9	10	11	
								8	9	10	11	12

Figure 2: Creation of temporal series

The result of this data preparation on this series of 12 measurements therefore returns 8 time series.

Now consider the segmentation of a series (in grey) with a missing value in 7th position (in light grey), see Figure 3. The segmentation is again performed by considering the 4 past values (in blue) to predict a single output value (in green).

Creation of temporal series												
	1	**2**	**3**	**4**	**5**	**6**	***NaN***	**8**	**9**	**10**	**11**	**12**
Temporal series	1	2	3	4	5							
		2	3	4	5	6						
								8	9	10	11	12

Figure 3 : Creation of temporal series when there is a missing value

The result of this data preparation on this series of 12 measurements therefore returns 3 time series. Figure 4 shows a training using a model training considering only time series as described above.

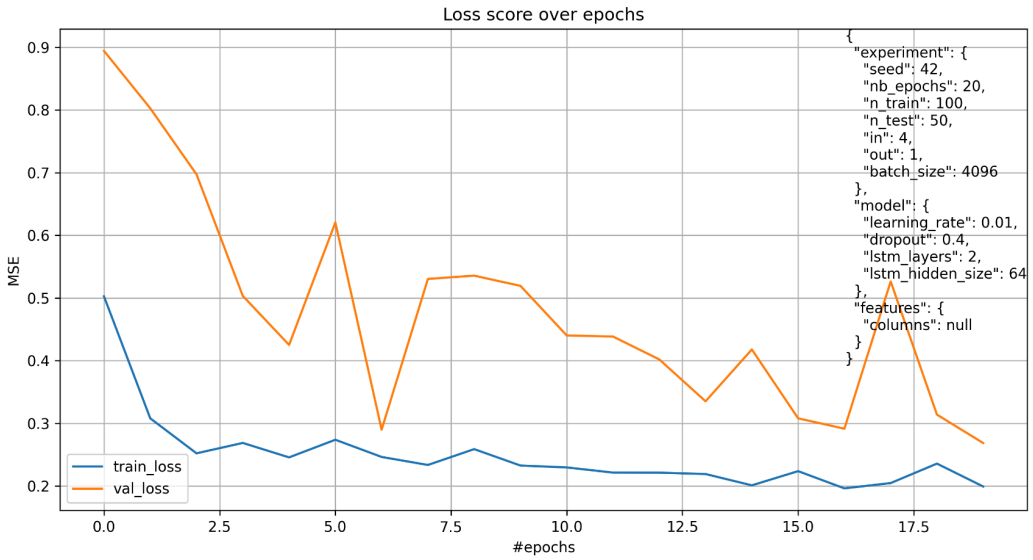


Figure 4: Loss curve considering only the time series.

Features engineering

The models presented in this document have all been trained on the same data.

Features are individual measurable characteristics of the data used by the model to make predictions. They serve as the input variables that help the model recognize patterns and relationships within the data. In our case, features are classified in the four following categories.

Device-related features

Three types of information are available that provide details on the type of dwelling in which the device is used.

- **LOCATION_AREA (4)**
  - ['Suburbaine', 'Rurale', 'Urbaine', 'Montagnarde']
- **CLIENT_BUILDING (16)**
  - ['Maison familiale', 'Immeuble locatif', 'Bâtiment industriel', 'Rural', 'Bâtiment administratif', 'Bâtiment commercial + habitation', 'Rural + habitation', 'Bâtiment industriel + habitation', 'Bâtiment commercial', 'Bâtiment administratif +

habitation', 'Bâtiment public + habitation', 'Place de sport', 'Bâtiment public', 'Elément isolé', 'Ouvrage militaire', 'Lieu de culte']

- **SERVICE_POINT_BUILDING (40)**
  - ['Chalet', 'Locatif', 'Villa', 'Ferme', 'Villa contigue', 'Habitation', 'NON RENSEIGNE', 'Locaux commerciaux et appartement', 'Villa jumelée', 'Halle industrielle', 'RCP', 'Garage et habitation', 'Rural', 'Restaurant et Habitation', 'Garage', 'Banque', 'Château', 'Boulangerie', 'Stand de tir', 'Commerce', 'Halle de sports', 'Chalet alpage', 'Café', 'Collège', 'Centre commercial', 'Dépôt', 'Restaurant', 'Atelier', 'Bureaux', 'Pompage', 'Halle de stockage', 'Eglise', 'Hangar', 'Bureau Communal', 'Week-end', 'Cave', 'Rural et habitation', 'Antenne', 'Cure', 'Station électrique']

Figure 5 shows the training results using a model that considers both time series and these device related features.

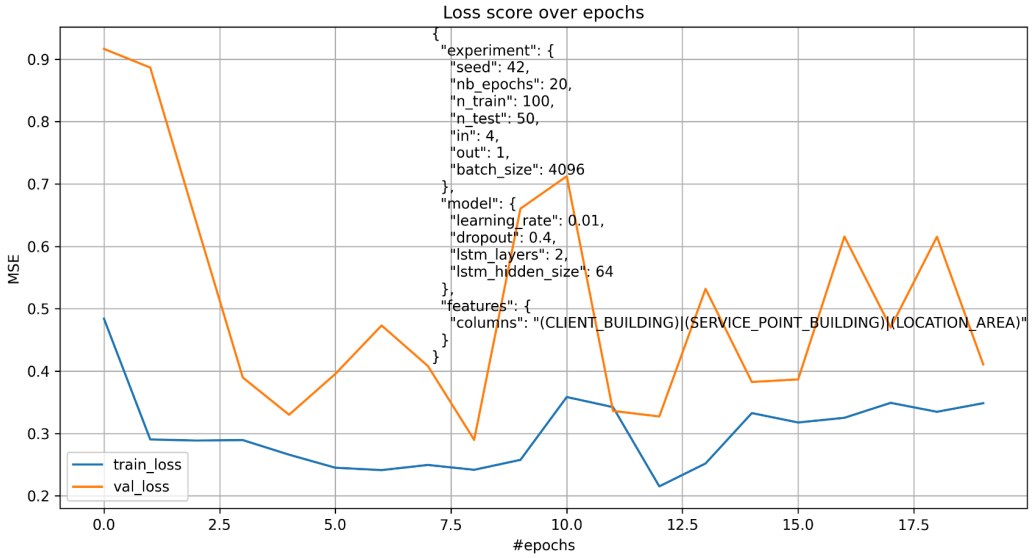


Figure 5: Loss curve considering the device related features.

Timestamps-based features

Energy consumption data is collected every 15 minutes and therefore have a timestamp value, enabling temporal metadata such as **month**, **day**, **hour** and **quarter-hour** to be extracted. This information can be used by the model to identify consumption trends.

The use of these features improves the model's understanding of seasonality, recurrence and patterns, and thus helps to optimize predictions. Figure 6 shows the training results of a model that considers time series and the timestamps based features. Figure 7 shows the resulting predictions and real values of energy consumption, using the mentioned model, for a single device.

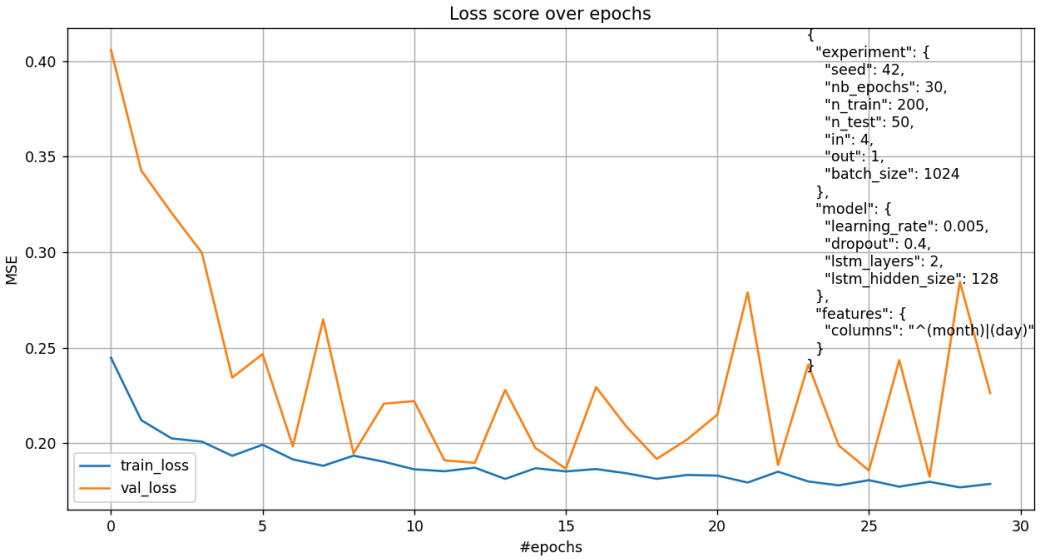


Figure 6: Loss curve considering the timestamp-based features.

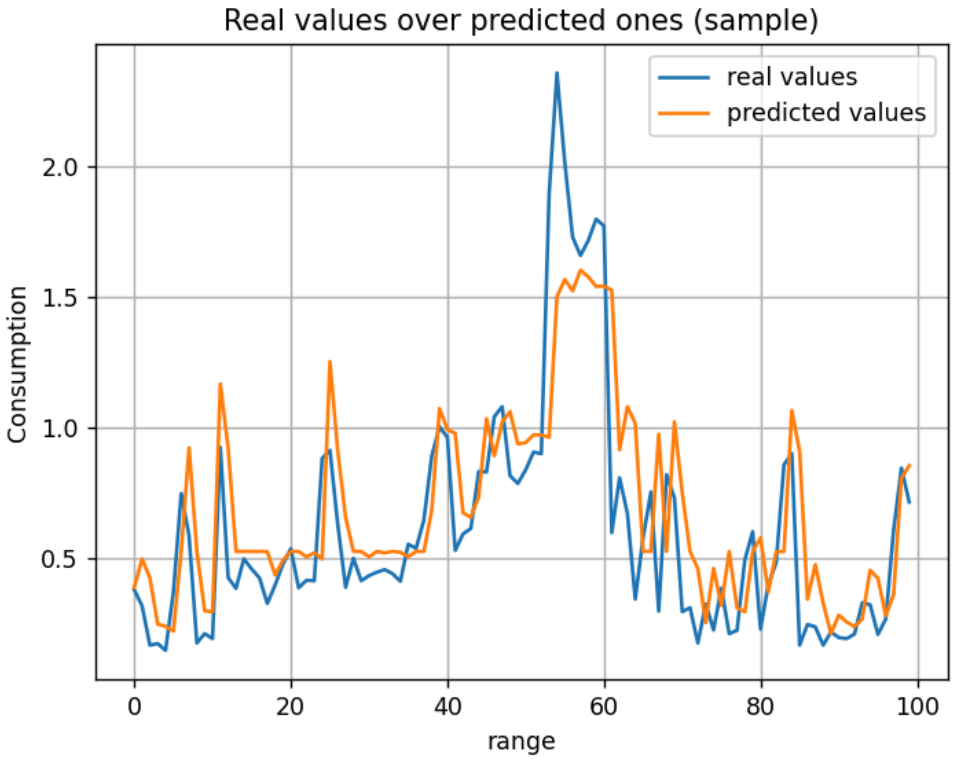


Figure 7: Real energy consumption values vs. predicted.

Statistics features

The values measured by the devices can be analyzed to extract statistical information that can be used to generate enriched features. This new information helps identify trends.
There are three types of statistics to exploit:

- **Average device consumption** : Devices are rated according to their average consumption: **low**, **normal** or **high**.
- **Seasonal activity** : seasons in which devices consume more than the general average are identified: **spring**, **summer**, **autumn**, **winter**.
- **Activity by season and time of day** : The times of day during the seasons when appliances consume more than the general average for the time of day are identified. A day is divided (for the time being) into five periods:
    - **Night** : 00:00 to 05:00
    - **Morning** : 05:00 to 10:00
    - **Noon** : 10:00 to 15:00
    - **Afternoon** : 15:00 to 20:00
    - **Evening** : 20:00 to 0:00

These statistics provide insight into consumers' daily electricity use habits.

Figure 8 gives a training result when the model considers the average device consumption statistics features and Figure 9 when considering the seasonal activity features.

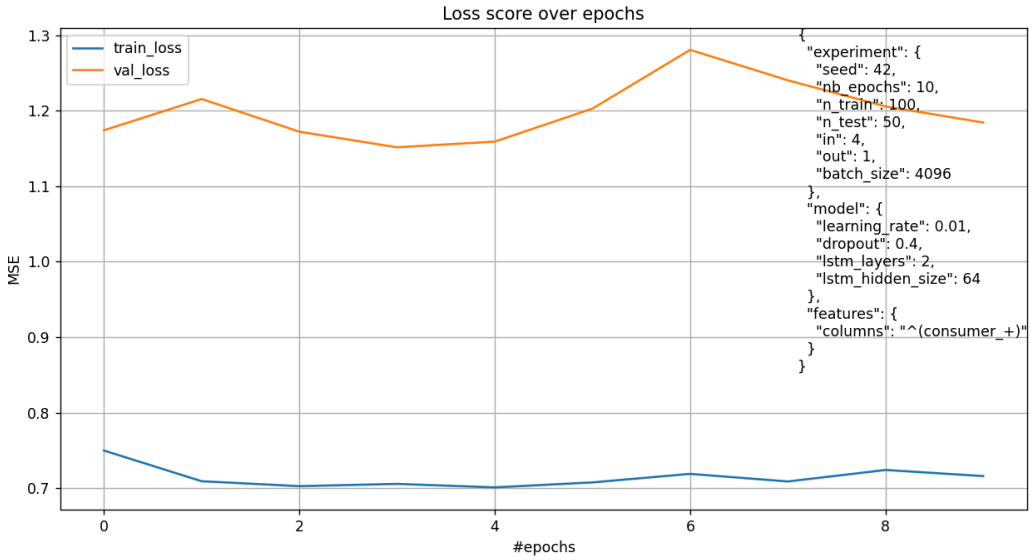


Figure 8: Loss curve considering the average device consumption statistics features.

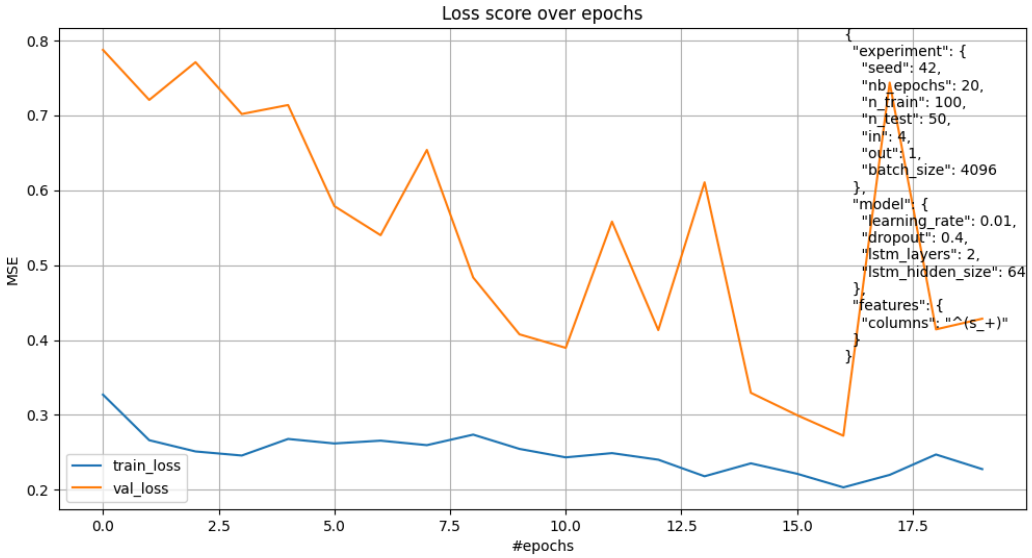


Figure 9: Loss curve considering the seasonal activity features.

Wavelet features - PyWavelets

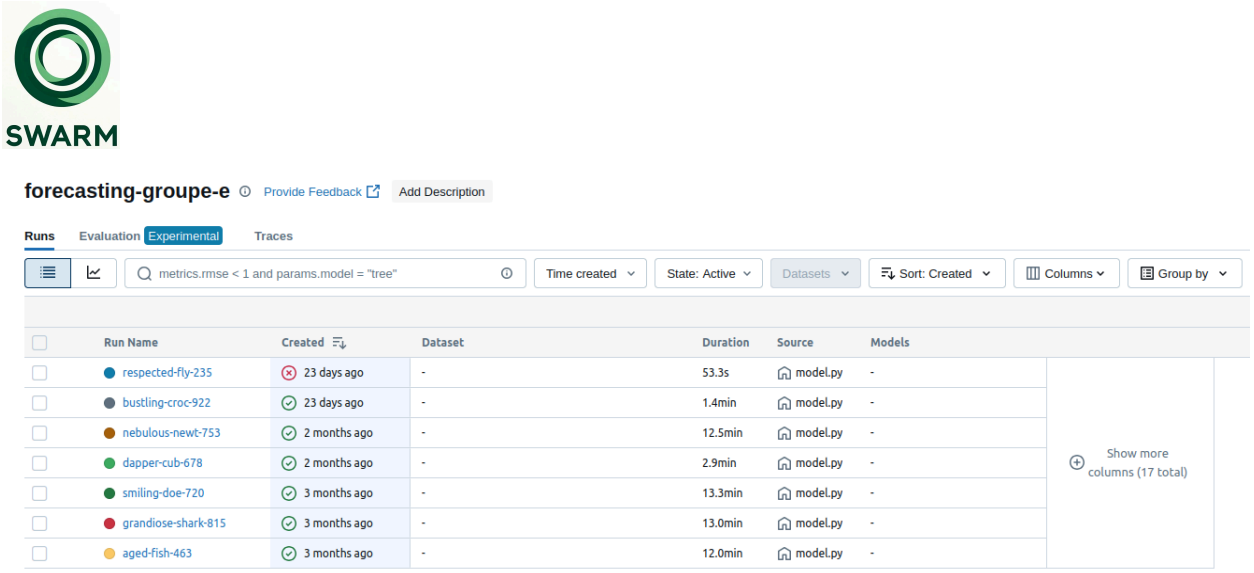
It is possible to decompose signals into discrete wavelets using the PyWavelets (pywt) library.

This decomposition produces, at each level, coefficients that represent different frequency components of the signal. From these coefficients, one can compute simple yet relevant statistics such as the mean, standard deviation, maximum, and minimum. These indicators provide a compact description of the signal across multiple scales.

These statistics can then be used as features to apply unsupervised learning methods, clustering, in order to classify and compare different signals based on their properties.

ML Flow integration

To track experiments, an MLflow server has been set up. The experiment configurations are logged, along with key metrics for each epoch : training loss, validation loss, and time. This information allows for efficient comparison of different runs and facilitates the analysis of model performance.



DVC pipeline

To ensure the reproducibility of experiments, a DVC pipeline has been implemented. It consists of five stages and can be easily executed using the dvc repro command. Each stage includes parameters that can be modified through the params.yaml file.

The first step involves cleaning the data, as described earlier. The second step is to select the number of devices to be used for training and testing. The third step focuses on creating features, following the explanations provided earlier in this document. The fourth step generates the time series, as previously detailed. Finally, the fifth and last step consists of configuring and training the model.

Structure of the params.YAML file

The parameters of each step of the dvc pipeline can be configured via a YAML file as shown in Figure 10. The file contains several keys corresponding to each step:

- “preprocessing"
- “selection"
- “feature_engineering"
- “preparation"
- “model"


```yaml
preprocessing:
  ratio_error: 1e-4 # Tolerance of number of 0 value
  ratio_missing_na: 0.08 # Tolerance of missing nan value

selection:
  seed: 0
  n_train : 100 # Number of train devices to be selected
  n_test: 50 # Number of test devices to be selected

feature_engineering:
  seed: 0

preparation:
  loopback: 4 # Number of past values to be considered
  horizon: 1 # Number of future values to be generated

model:
  seed: 0
  horizon: 1 # Number of future values to be generated !same as before
  features_to_use: '#' # re. to select features base on the text columns
  batch_size: 512
  lstm_hidden_size: 64
  lstm_layers: 2
  dropout: 0.2
  fc_hidden_size: 2
  learning_rate: 0.0001
  num_epochs: 5
```

Figure 10 : Sample of YAML file for the dvc experiment configuration.

Conclusion

Throughout this document, we have introduced the dataset, the cleaning process, and the feature extraction, with a limited view of how they are incorporated during model training. We have also seen how the prototype can be fully configurable, from the experiment setup to the model's hyperparameters. That said, although the entire prototype is conceptually functional, several factors still need to be taken into account.

It should be noted that the dataset contains uncertainties and requires substantial preprocessing. Indeed, many devices either have missing values, represented as NaN in the code, or errors indicated by zero values. In some cases, devices that are supposed to be used during the day show no actual power consumption during daytime, which is inconsistent.

As for the metadata, the textual information provided is not effectively usable, as it is either too vague or lacks specificity. Moreover, the data corresponding to the clients' unique descriptions cannot be linked to the measurement devices, as there is no existing connection that allows for such association.

However, determining statistics based on consumption habits, time of day, and periods of the year seems to be a good approach and can yield promising results. Similarly, creating a clustering of these devices based on these same data would be a valuable additional feature.

Regarding the loopback and horizon values, given the currently available resources, these can only be kept very small. Larger values would result in significantly longer computation times for the training stage.

One of the key benefits of this implementation is the integration of MLflow, combined with the DVC pipeline, which makes every experiment fully reproducible and traceable, including its configuration.