GENTE

Developing Edge Intelligence

And enhancing submetering device

SUMMARY

Improving forecasting for photovoltaic systems utilising edge devices.

Website genteproject.com

Contact hello@genteproject.com

Impressum

Internal Reference

Deliverable No.	D 5.2 (2023)
Deliverable Name	Edge-based DER forecasting and diagnostic algorithm, incl. privacy-preserving learning
Lead Participant	HSLU
Work Package No.	5
Task No. & Name	T 5.2
Document (File)	GENTE-D5.2-Developing Edge Intelligence-PU-P_R0
Issue (Save) Date	2024-02-15

Document status

	Date	Person(s)	Organisation
Authors	2024-01-12	Peter Allenspach	HSLU T&A DEEP
Verification by	2024-02-09	Alexander Nnamdi Ndife	Chalmers
Approval by	2024-02-16	Roman Lötscher	HSLU T&A DEEP

Document sensitivity

☐	**Not Sensitive**	Contains only factual or background information; contains no new or additional analysis, recommendations or policy-relevant statements
X	**Moderately Sensitive**	Contains some analysis or interpretation of results; contains no recommendations or policy-relevant statements
☐	**Sensitive**	Contains analysis or interpretation of results with policy-relevance and/or recommendations or policy-relevant statements
☐	**Highly Sensitive Confidential**	Contains significant analysis or interpretation of results with major policy-relevance or implications, contains extensive recommendations or policy-relevant statements, and/or contain policy-prescriptive statements. This sensitivity requires SB decision.

Disclaimer

About ERA-Net Smart Energy Systems

ERA-Net Smart Energy Systems (ERA-Net SES) is a transnational joint programming platform of 30 national and regional funding partners for initiating co-creation and promoting energy system innovation. The network of owners and managers of national and regional public funding programs along the innovation chain provides a sustainable and service oriented joint programming platform to finance projects in thematic areas like Smart Power Grids, Regional and Local Energy Systems, Heating and Cooling Networks, Digital Energy and Smart Services, etc.

Co-creating with partners that help to understand the needs of relevant stakeholders, we team up with intermediaries to provide an innovation ecosystem supporting consortia for research, innovation, technical development, piloting and demonstration activities. These co-operations pave the way towards implementation in real-life environments and market introduction.

Beyond that, ERA-Net SES provides a Knowledge Community, involving key demo projects and experts from all over Europe, to facilitate learning between projects and programs from the local level up to the European level.

www.eranet-smartenergysystems.eu

Abstract

This deliverable introduces an edge intelligence application aimed at forecasting the future production of solar panel installations by leveraging available data sources, including historical solar panel output and meteorological predictions. The objective of the AI-based model is to estimate the future power production of a PV array within a relatively short time frame (1 day) by utilizing past meteorological data and forecasts. The dataset used for the model originates from the 'Am Aawasser' PV array, complemented by daily grabs of hourly weather forecasts for the subsequent 7 days from the Meteoblue API. The analysis covers the period from January 13, 2023, to July 11, 2023, with intermittent data dropouts.

The findings indicate that a straightforward model relying solely on irradiation forecasts from a sophisticated weather model, scaling them by a certain constant factor to convert from W/m^2 to kW output by the PV array, may suffice for predicting solar power within a short timeframe (approximately 1 day). Notably, the investigation did not reveal any discernible enhancement in forecast accuracy with the adoption of a more intricate model, specifically the XGBoost algorithm.

Table of Contents

Impressum **1**
Abstract **3**
List of Abbreviations **5**
1. Introduction **6**
2. Up- and Downsides of Edge Intelligence **7**
3. Data Acquisition and Organisation **8**
3.1 Meteorological input data 8
4. Goal of model **9**
4.1 Why a Day-Ahead Forecast? 9
4.2 Loss Functions 10
5. Evaluation of different models **11**
5.1 Data processing 11
5.2 Forecast days 12
5.3 Data analysis 13
5.4 Data Selection & Preparations 17
5.5 Simple model 18
5.5.1 Automating with Linear Regression 19
5.6 XGBoost model 20
5.7 XGBoost model performance 22
6. Implementation on the Edge **26**
6.1 Target Setup 26
6.2 Simple Model 27
6.3 XGBoost Model 28
7. Conclusion **29**
References **30**

List of Abbreviations

API - Application Programming Interface

PV - Photovoltaic system

MSE - Mean Squared Error

MCU - Microcontroller Unit

1. Introduction

Within the GENTE project, both SmartHelio and Reengen developed submetering devices, bringing the intelligence to the edge. One benefit of increased computational power at the edge level, is the option to execute algorithms on site, in order to circumvent or at least minimise the data collection at a centralised location. There are other possible advantages, such as reduced latency, which are of no benefit in this use case.

As such, this deliverable investigates the development of an on-the-edge prediction algorithm to forecast the future power generated by PV – Arrays. The basis of this deliverable is an HSLU student's master's thesis, investigating different algorithms, suited for a PV Forecast. The thesis revealed that the inclusion of weather forecasts to the PV forecasting model greatly increased its accuracy, therefore laying the groundwork for the GENTE project, as similar algorithms were used.

With this first step taken, the contents of this deliverable focuses on the adjustment, optimisation and implementation of a PV forecasting model on an edge device.

2. Up- and Downsides of Edge Intelligence

Machine learning models are often run on powerful computers connected to the internet. This can lead to all kinds of potential problems - mostly related to the fact that those machines are not local but merely connected through the internet. With edge intelligence, it is possible to run models on a local system, often a lot smaller and more power-efficient. For example, an nRF52840 Arm Cortex-M microcontroller has an absolute maximum power consumption of 50mA@3.3V even with wireless connectivity enabled, which comes out to a maximum power draw of just 160mW - usually significantly lower. Compared to an x86-based system, which often draws tens to hundreds of watts in operation, a microcontroller is vastly more power efficient.

However, there are also problems with edge intelligence - if the edge device is not able to only use local data, it needs to connect to the internet somehow, which often requires an user to input, for example WiFi credentials along with a higher power draw compared to a system with no connection to the internet - as well as reliance on external systems. If a given server has a data source or machine learning model that exclusively uses output of another model running in the cloud for its input, running the additional model on an edge device might not be necessary, as the model could just be computed on that particular server or another server and be transmitted to the edge device thereafter. A significant upside however exists when this data can be locally collected and used, for example, for keyword detection used in voice recognition. Rather than needing to constantly record and upload audio and download the model results, utilising edge intelligence in this case can allow for a way more efficient data flow - keeping the data on the device at all times. For example, rather than running a full voice recognition engine, a more simple model could be used that only detects a certain keyword, which allows such a model to demand minimal processing power, and in turn power draw.

Another potential upside, should the edge device be connected to the internet, is that depending on the model's purpose, the model output can be dramatically smaller than a model input. For example, a face detection model can turn an input stream of video into a simple, text-based output that just contains an ID corresponding to a detected person. In the case of PV production prediction, this reduction in size isn't as large, as a few hundreds of bytes turn into an output of a few bytes. If the model input is not required elsewhere, this can lead to a drastic reduction in the amount of data needing to be transmitted.

This means that, for edge intelligence to be a useful technique, these conditions usually need to be met:

- Data collected locally
- Model output used locally or only (considerably smaller) model output sent out
- Model sufficiently small to run on constrained hardware

In the case of this work, this would mean that the boundary conditions for an edge-intelligence model to be preferable over a model run on a general computing device would be:

- Locally collected PV output data relevant to model input
- Model doesn't heavily rely on remote input data

3. Data Acquisition and Organisation

Data acquisition is a crucial part of this study, as the quality and reliability of the data directly affect the outcomes of our analysis. This chapter outlines the data sources, the methods employed for data gathering, and the rationale for our choices.

Initially, data from Hochschule Luzern (HSLU) experiments were considered for this study. This data was a part of an HSLU student's master's thesis[1], and collected from the SmartHelio prototypes mounted on solar panels on the roof of HSLU. However, it was decided to not use this data as:

- Temporal Scope: The HSLU data only spanned approximately one month, which limits the scope of any long-term analysis.
- Nature of Weather Data: The dataset comprised real weather data that was acquired retroactively, rather than using historical meteorological forecasts.

The chosen replacement dataset leverages data from 'Am Aawasser,' a local energy community in Buochs, Nidwalden, which is a testing site of the GENTE project. Am Aawasser contains a 124kWp solar power plant. The data logging interval is 15 minutes, resulting in 96 measurements per day, per measurement type.

3.1 Meteorological input data

For obtaining reliable meteorological predictions, the services of Meteoblue[2] were utilised. Meteoblue provides a range of meteorological metrics essential for the study, most notably forecasting solar irradiance data [W/m^2]. To ensure utilisation of the most current and applicable data, an Application Programming Interface (API) was utilised to make daily calls to Meteoblue's servers. This approach allows retrieval and collection of real weather forecasts, rather than relying on past, recorded (actual) weather data, which may not be indicative of future (uncertain) weather conditions. This is important because the weather information available on Meteoblue is updated constantly, and calling the API for past data returns historical recorded data (actual) rather than the historic predictions made for those days.

The acquired data through the API response is saved into individual files, one for each API call. This data offers forecast data up to 10 days ahead, providing numerous possibilities for conducting various types of analyses and predictions, as a single API call returns a forecast for 10 days. However, for the specific purposes of this study, only 1-day ahead forecasts were utilised in the training models. The rationale behind this choice is grounded in the observed degradation in forecast accuracy as one attempts to predict weather conditions further into the future. Previous evaluations have demonstrated a significant decrease in the accuracy of meteorological forecasts for periods extending beyond a single day. Selective focus on 1-day ahead forecasts in the training data aims to strike a balance between forward-looking analysis and data reliability. This approach is chosen to ensure accuracy of the results, as weather forecasts can change quite rapidly, and the timescale in which a forecast is needed is rarely more than a day in advance.

4. Goal of model

The goal of the model is to predict the future output of solar panels on a relatively short scale of 1 day using past meteorological data and meteorological forecasts. The model is designed to be as accurate as possible, with a pessimistic bias. Past solar panel data is available, but future output is not dependent on past data other than for scaling. Therefore, past time series data will not be used as a feature.

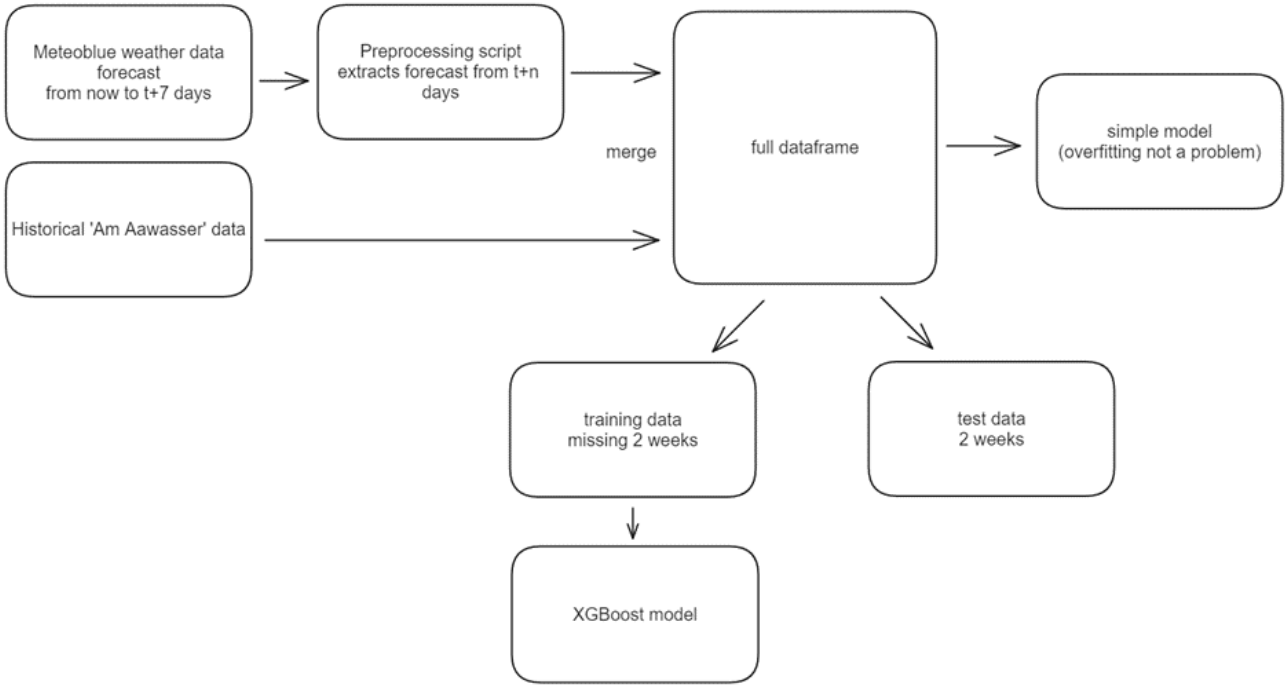

Figure 1 - Architecture of PV Forecasting Model

4.1 Why a Day-Ahead Forecast?

Solar power is an intermittent energy source, meaning that its output can fluctuate significantly depending on weather conditions. This variability can make it difficult to integrate solar power into the grid, and can also lead to problems for solar panel operators who need to be able to predict how much power their panels will generate.

A short-term (1 day) solar power forecast can help to address these challenges. By predicting how much solar power will be available the next day, grid operators can better integrate solar power into the grid and ensure that there is enough power to meet demand. LECs can also use a short-term forecast to plan their energy needs and avoid problems such as running out of power in the local battery banks or having to sell excess power back to the grid at a low price.

4.2 Loss Functions

There are multiple loss function concepts that can be used to train a ML model. Two possible loss functions are:

Average Error (over the 96 measurements of day): This loss function minimises the average error of the model's predictions over the entire day. This loss function J_{AE} is useful if backup power is available, such as battery buffers.

$$J_{AE} = \frac{1}{96} \sum_{i=0}^{96} P_{True}[i] - P_{Pred}[i]$$

Mean Squared Error (MSE): This loss function minimises the error of the model's predictions for each individual data point. This loss function J_{MSE} is useful if no backup power is available and peaks need to be predicted so that non-time-critical devices can be scheduled to run at peaks.

$$J_{MSE} = \frac{1}{n} \sum_{i=0}^{n} (P_{True}[i] - P_{Pred}[i])^2$$

The choice of loss function will depend on the specific needs of the user. For example, if the user has backup power available, they may prefer to use the Average Error loss function. If the user does not have backup power available, they may prefer to use the MSE loss function, which was used in this work to train the various models.

In this work, especially as the data available is not comprehensive, it was decided to focus mostly on standard error measures. However, if a future work with a more generalised model were to be made, it would be imperative to focus on the loss function, and employ one well-suited for the physical output the model is projecting.

5. Evaluation of different models

Multiple models were tested for solar power prediction in the HSLU student's master's thesis[1]. However, this testing was only numeric, meaning that the models were evaluated solely based on their error measure, rather than considering what influences a solar panel's output, allowing the model to optimise for inconsequential values, such as the solar panel's output on past days, which are irrelevant to forecasts other than for scaling purposes, as a solar panel has no 'memory'. Additionally, no future forecast data was available, so past measured data with noise applied was used. This approach was not ideal, as real forecasts are not just past data with noise - a forecast might be way more off because it predicts clear skies and a cloudy day happens. Therefore the decision was made to pursue the development of a newer (more lightweight) forecasting model using actual forecast data to train, which can be deployed onto an edge device.

Due to the complexity of the weather models provided by Meteoblue, it is desirable for the solar power prediction model to be relatively simple. This would allow the model to utilise the current weather model outputs as an input to calculate the predicted solar energy output. A simpler model is also preferable to run on an edge device, such as a device fitted directly to a solar panel, over a very complex model. If a complex model using only the weather model's output as input data could improve performance in respect to irradiance data, this would mean the weather model's predictions are imprecise and that model could be used to improve the underlying weather model.

Since a solar panel acts like a pyranometer (more incoming irradiance [W/m^2], more wattage output [W]), it is likely that the predicted solar power is highly correlated to pyranometer readings. This suggests that a simple model that uses pyranometer readings as input may be sufficient for predicting solar power.

5.1 Data processing

The selected data input streams for the new forecasting models were:

- 1 - day ahead weather forecast from MeteoBlue.

Data input was given from the 'Am Aawasser' PV array producing a measured peak output of 82kWp in a resolution of 15 minutes, along with hourly weather forecasts for the next 7 days, grabbed from the Meteoblue API every day at 14:00. Data from 2023-01-13 up to 2023-07-11 was analysed, limited by the amount of predictions scraped at that time.

As the solar weather forecasts from MeteoBlue have a temporal resolution of 1 hour, the PV data from Am Aawasser was downsampled to achieve the same resolution. This simplifies the architecture of the employed ML model and should increase forecasting accuracy.

The meteorological forecast from Meteoblue consisted, along with static metadata of the forecast location, of weather data described in Meteoblue's documentation[3], also visible as feature variables in Figure 4.

Past solar data points were deliberately not used as model inputs, as a solar panel's normal working output is only dependent on the amount of sunlight shining on it, not its past output. The only relevant information of a solar panel's past output is how much electricity it produces with a set amount of sunlight hitting it (measured in W/m^2). In a condition where a solar panel is partially shaded at some part of the day, potentially only during some seasons, or positioned in a way where it only gets direct sunlight during a part of the daylight hours, it might be beneficial for the model to have past solar data as inputs in order to learn that behaviour. However, since the available data did not contain these properties, such training could not be conducted and/or verified.

5.2 Forecast days

Utilising historical weather data rather than historical weather forecasts can lead to wrong conclusions - if historical data is used for training or analysis rather than forecasts, this leads to a marked difference between performance during analysis and actual usage, as in the real world, the wanted output relies on forecast data, and not historical data. Past weather data includes the actual weather at that time, while a forecast might predict the wrong weather, which majorly affects the output more than simple noise; What is in the past is known, and would not require a forecast.

With the periodically scraped data, it is possible to get past forecasts for n days. Most interest lies in 1-day ahead forecasts, so usually, n=1.

To allow for historical forecasts (meaning forecasts made in the past) to be used, a logging script was implemented, to collect historical forecasts, as Meteoblue does not offer this information. In this context, 'historical forecasts' means that forecasts were collected before they were replaced with actual weather data at that specific date. Note that the forecasts were always grabbed at 14:00; the forecast might be different (less, respectively more precise) in the morning or the evening, due to the time forecast varying from 9.5 to 33.5 hours.

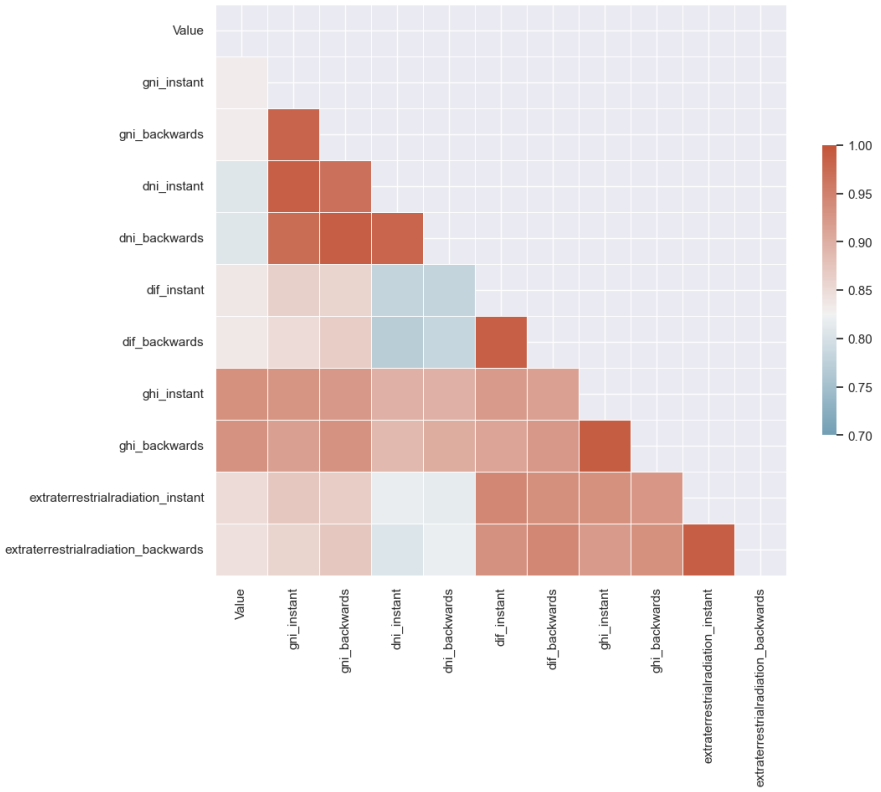

Figure 2 - Visualization of correlations between PV output ('Value') and various irradiance measures.

5.3 Data analysis

A correlation plot of all metrics from MeteoBlue and the PV output power was made to analyse the available data.

As it was suspected that the PV output power would heavily correlate with various irradiance measures from 1-day-ahead forecasts from Meteoblue, the correlation between the various irradiance measures and PV output was calculated, as shown in figure 2. Note that the color scale in the figure starts at 0.7 rather than the usual 0.0 to make the differences between the individual values more easily visible.

From figure 2, looking at the leftmost column, it is clear that, while all of the irradiance measures correlate heavily with the solar panel output (all above 0.7 correlation), the options correlating the most are `ghi_instant` and `ghi_backwards`, describing the global horizontal irradiation in the forecast location. This makes sense, as these values correlate to the amount of solar radiation received from the sun on a horizontal surface (W/m^2). While a solar panel mounted at an angle will receive a differing amount of radiation, the radiation received on a horizontal surface provides a good base level. Whether this correlation would hold up with other PV installations would have to be tested with different datasets from different locations. However, this data is not available at this time.

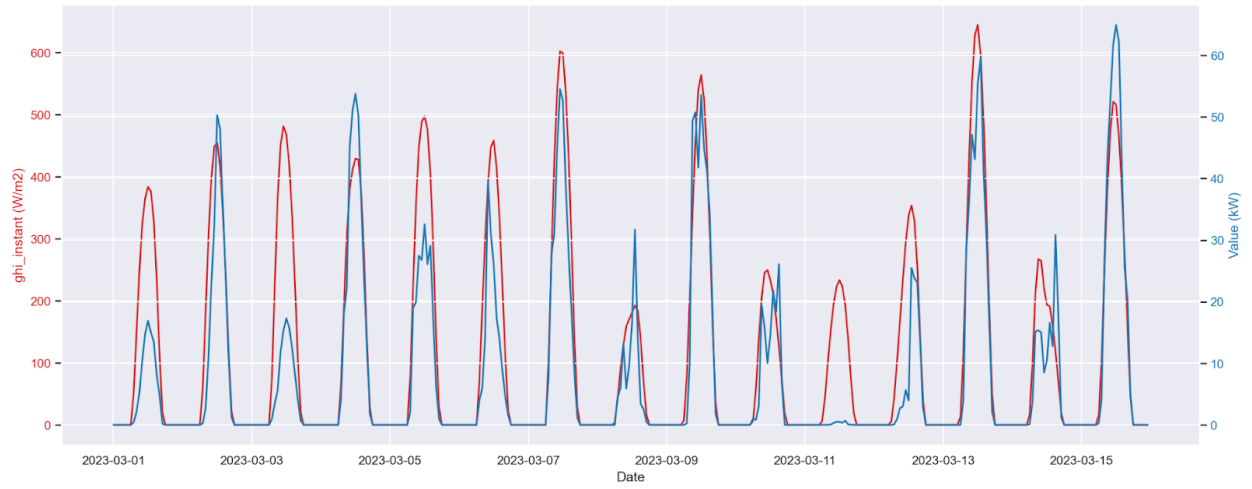

Figure 3 - Visualization of ghi_instant 1-day ahead forecasts (Global Horizontal Irradiance) and PV output ('Value'), showing a strong correlation between the two data series

Shown in figure 3 is the ghi_instant forecasts overlaid with the actual production of the PV array between 2023-03-01 to 2023-03-15 , showing extremely strong correlation. It also showed forecasts sometimes being off, such as on 2023-03-04, when a nice day was forecasted, but according to actual PV production it seemed to be overcast. At 2023-03-10 a day with spotty cloud coverage is also visible, identifiable by PV output fluctuating up and down throughout the day rather than following a curve.

Figure 4 shows the correlation between all the available meteorological data and the PV array output, confirming that indeed, the most strongly correlating values lie in the irradiance measures, with some correlation in the temperature and sunshine time fields, which make sense - as both higher temperatures and longer sunshine time imply more sunshine, and thus more solar panel output.

The pair plot in figure 5 confirms this assumption, showing a heavy correlation between ghi_instant 1-day ahead forecasts and PV array output, with some outliers where weather was better than forecast (values in top-left half) or worse than forecast (values in bottom-right half).

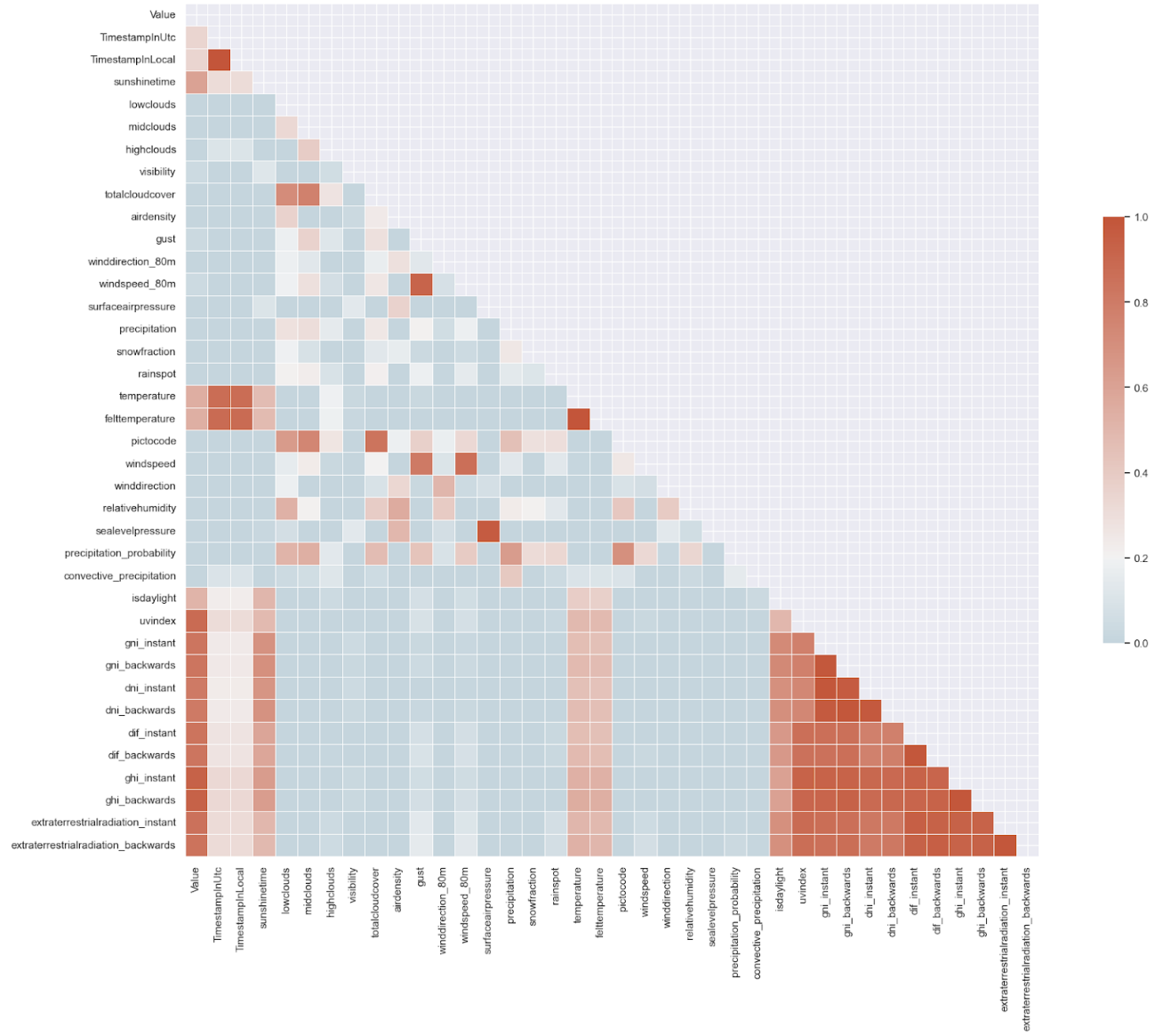

Figure 4 - Visualization of correlations between PV output ('Value') and weather data from 1-day-ahead forecasts from Meteoblue

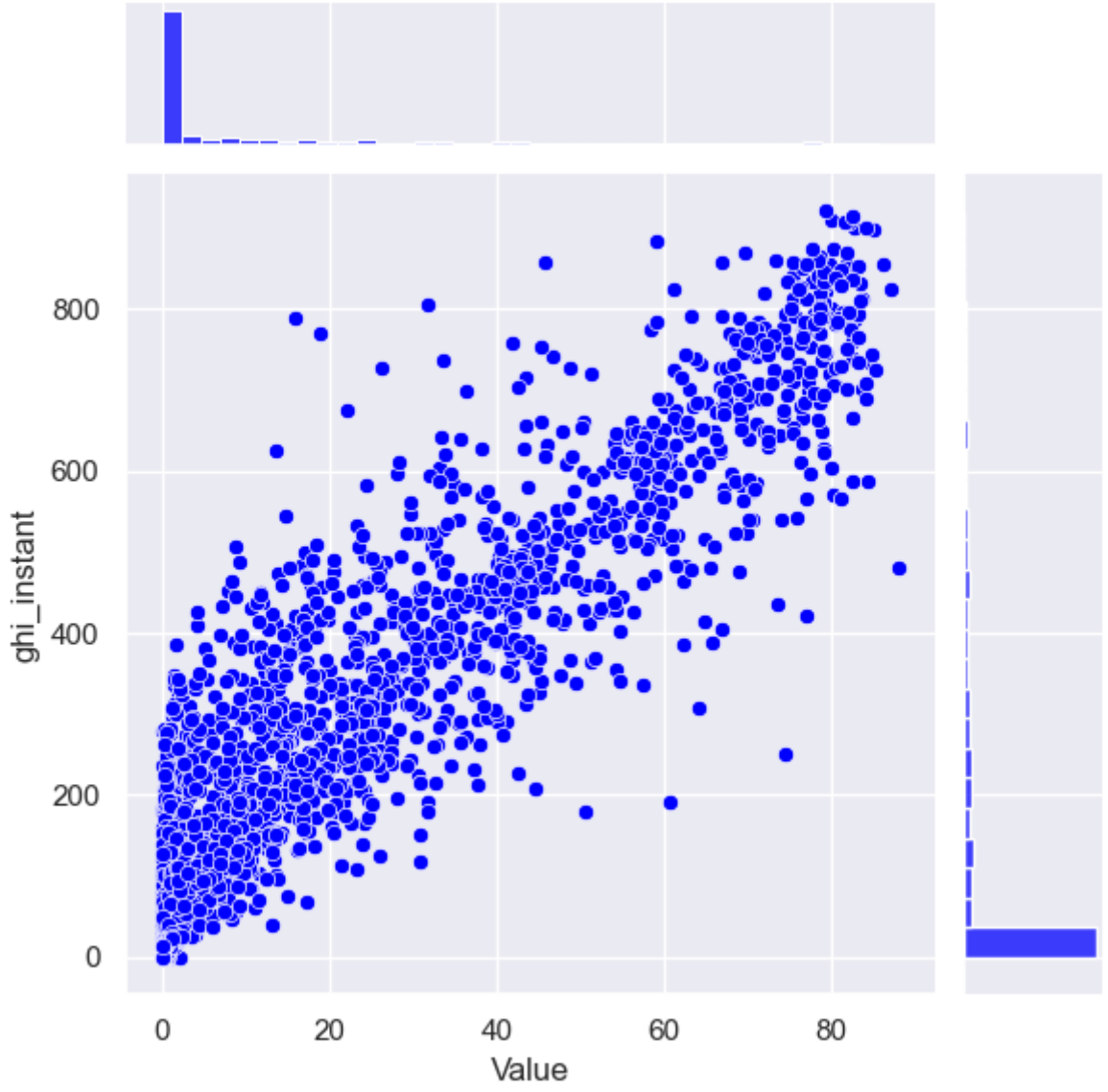

Figure 5 - Pair plot of ghi_instant forecasts and PV array output

5.4 Data Selection & Preparations

Based on the revelations from the data analysis section, it was decided to provide the model with the one-day-ahead 24 hour irradiance forecast, with input data at a temporal resolution of 1 hour. The target for this interval was the 24 hours actual output of the PV power system at the Am Aawasser site as shown in See Figure 6.

To prepare the data for training, both data types were normalised using the standard - normalisation function, so both data types had unitless values between 0 ... 1:

$$x_{norm} = \frac{x_{Data} - \mu_{x_{Data}}}{\sigma_{X_{Data}}}$$

Additionally, a Train - Test - Split was used to create both a training and test dataset, without any overlapping samples. As there is no hyperparameter-optimisation using this test data, splitting off an additional validation dataset is not needed. For this, a 2 week test - data snippet was taken out of the available 25 weeks of total data, resulting in a 23 weeks training dataset.

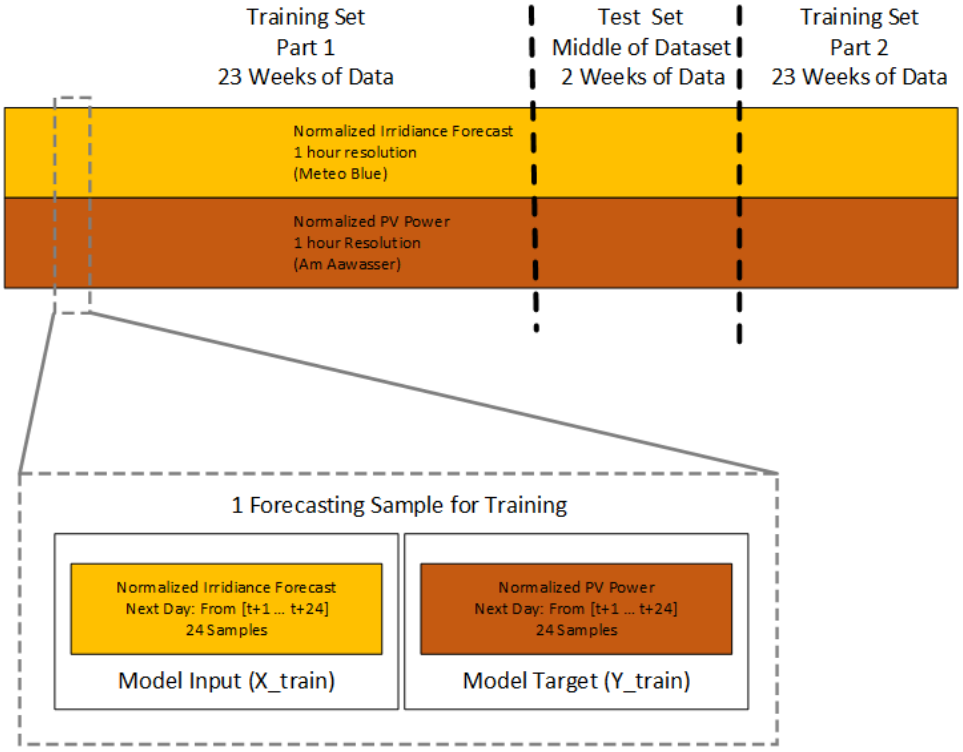

Figure 6 - Data Split & Data Preparation

5.5 Simple model

The results of data analysis imply that using a very simple model of only scaling the ghi_instant forecast might already be enough to predict the production of the PV array. First, this scaling was applied manually in order to try minimising the 'Mean Average Error' by calculating those error measures while stepping through the scaling at 0.1 steps, which was found to be at 11.5 for this specific array, applied in the following formula:

$$ghi\ [W/m^2]\ \cdot\ 11.5\ =\ pred_{output}\ [kWh/15\ min]$$

This result could also be achieved by various optimization techniques descending on a minima.

The prediction being in kWh is due to the fact that this is the way the PV array's output was measured. Rather than measuring instantaneous power, it displays the cumulative output of the last 15 minutes. The calculated value, and as such the output, could be changed to average kW over an hour by multiplying by 4.

The error measures for the simple model are shown in figure 7. Note that the error measures are percentual. This has up- and downsides: if absolute error measures were shown, all more-or-less forecast days would have errors close to zero, while generally wrong forecasts (cloudy day forecast, actual day sunny, as was the case in mid-january) would have high error measures. This stands alongside the fact that errors on overcast days would only show up as minimal, while errors on sunny days would be quite significant. In order to mitigate this behaviour, percentual error measures were used for display ($\epsilon_{\%}\ =\ \epsilon_{real}\ /\ x_{mean}\ \cdot\ 100$). This amplifies such coarse errors in forecasting while showing errors on overcast days as well as clear days in a fair comparison. However, such an error measure should not be used when training the model because of exactly this problem - an overcast day which was forecasted to be sunny would have MAPEs in the thousands of percent, which would coax the model to overfit to specifically those occurrences.

Figure 7 - Percentual error measures over the full data period with simple scaling model. Blank period has missing data

5.5.1 Automating with Linear Regression

Linear Regression allows a set of linear parameters (factors) for a corresponding set of input parameters to be optimised on a selected loss function.

A model was set up to be trained to find the optimal coefficients for the input parameter of ghi_instant. The MSE loss function was used for the optimisation step. The extracted coefficient for the input parameters turned out to be [11.14]. Note that these parameters are slightly different from previous results as in manual testing, because a different error measure - mean error - was used, and manual testing only changed the parameters in steps of 0.1.

To analyse whether it is worth it to include the other various available weather variables in a linear regression model, increasing its complexity, a model utilising all possible input features from Meteoblue was also trained to assess the importance of each and every input variable. For the purposes of comparison,the coefficients of this regression with minmax-normalized input variables is shown in figure 8.

Checking these results, at first glance it might appear as though temperature also plays a major role in the model's output; this however is just an artefact of the fact that the data input includes both a `temperature`

and a `felttemperature` variable, which are very strongly correlated, of which one gets multiplied positively, and one negatively.

This means, rather than those variables having a big effect on model output, the difference between felt and real temperature, which mostly is dependent on factors such as humidity, sunshine etc., has a minor impact on model output.

It is also important to note that even temperature itself is correlated to irradiance measures. As sunlight passes through the atmosphere, it heats up the air, increasing its temperature.

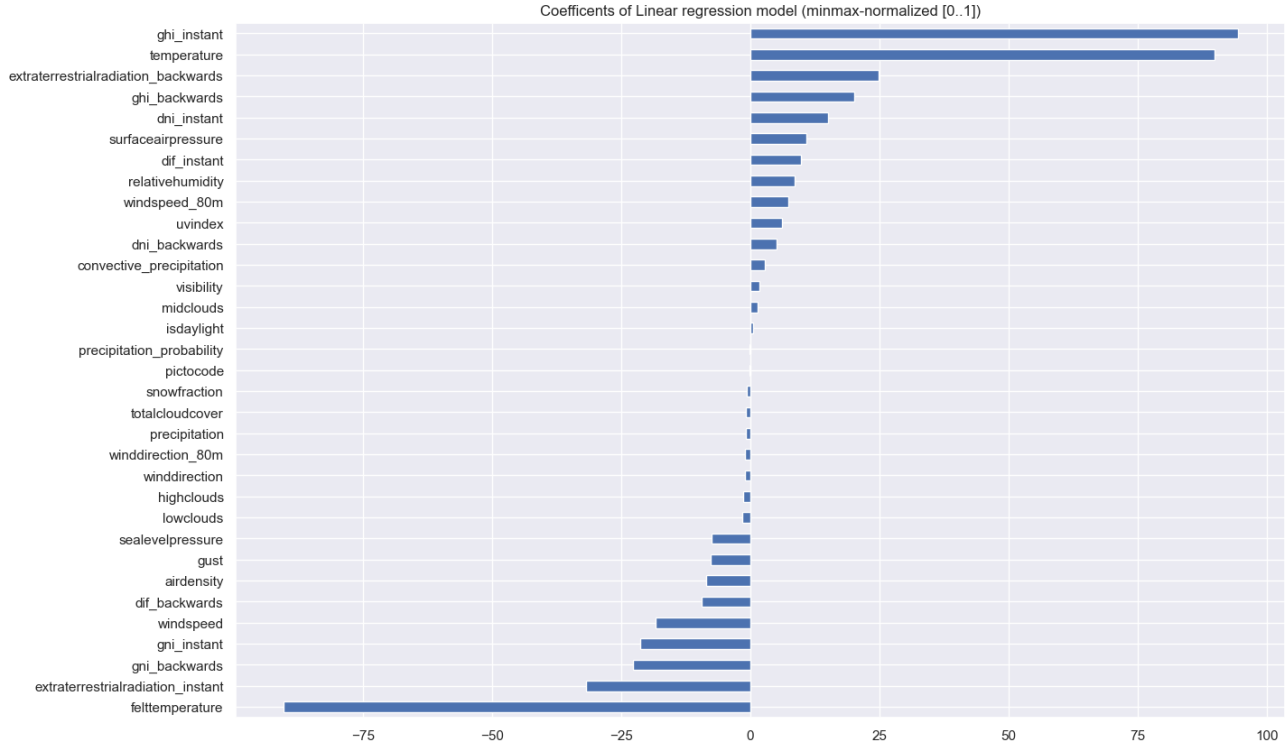

Figure 8 - Coefficients of the linear regression model with all weather variables as inputs.

5.6 XGBoost model

The HSLU student's master thesis [1] took a look at various traditional machine learning models, finding XGBoost to be the most performant one. While that work only was able to look at past historical data with added noise, the results are conclusive enough to rule out those various other models from performing better than an XGBoost model. As such, in this work, XGBoost was chosen as a representative comparative model to judge more complex models' performance against a simple model.

eXtreme Gradient Boosting (XGBoost), is a powerful machine learning algorithm that combines decision trees and gradient boosting to create accurate and efficient predictive models. It tends to outperform other gradient boosting algorithms. XGBoost is widely used for classification, regression, and ranking tasks in various domains like finance, healthcare, and natural language processing.

Performance analysis of XGBoost models in this use case will be discussed in the next section.

5.7 XGBoost model performance

Shown in figure 9 is the performance of various XGBoost models trained on available data on a given test period. Looking at the results, it is clear that their performance is essentially the same as the simple model which linearly scales the irradiance values. No clear trend of better predictions, however miniscule, is evident.

It should be noted that in figure 9, the first and last visible days were purposefully left in from the training dataset, showing a major discrepancy in XGBoost model performance between seen and unseen data, indicating a major overfit. However, trained models that did not show this overfitted behaviour had even worse performance in unseen data.

Further, the SHAP values of the XGBoost model's inputs are plotted in figure 10. SHAP values can be understood in a similar way to the coefficients of a linear model, showing how much a certain input variable has influence on the model's output. These values are shown as they more clearly show input variable's effects on model outputs than the three 'usual' feature importances, weight, gain and cover. The values in figure 10 show that, other than **ghi_instant** and **isdaylight**, the input features have a minimal effect on model output.

Overall performance over the test window, including both correctly and wrongly forecast days for various models is shown in figure 11, showing fairly small differences between the different models.

Figure 9 - Performance of various XGBoost models trained on available data compared to the simple baseline model.

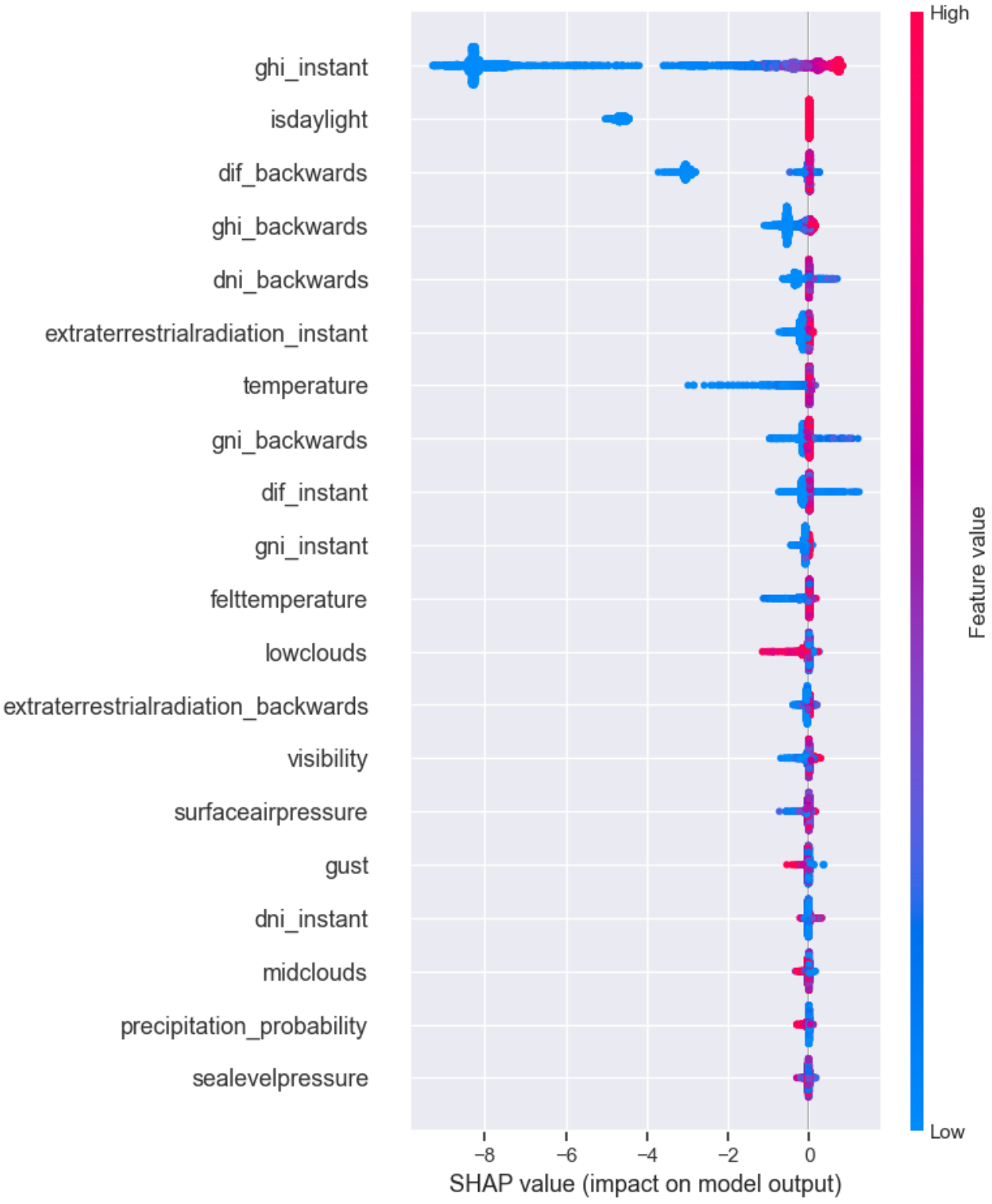

Figure 10 - SHAP values of a trained XGBoost model

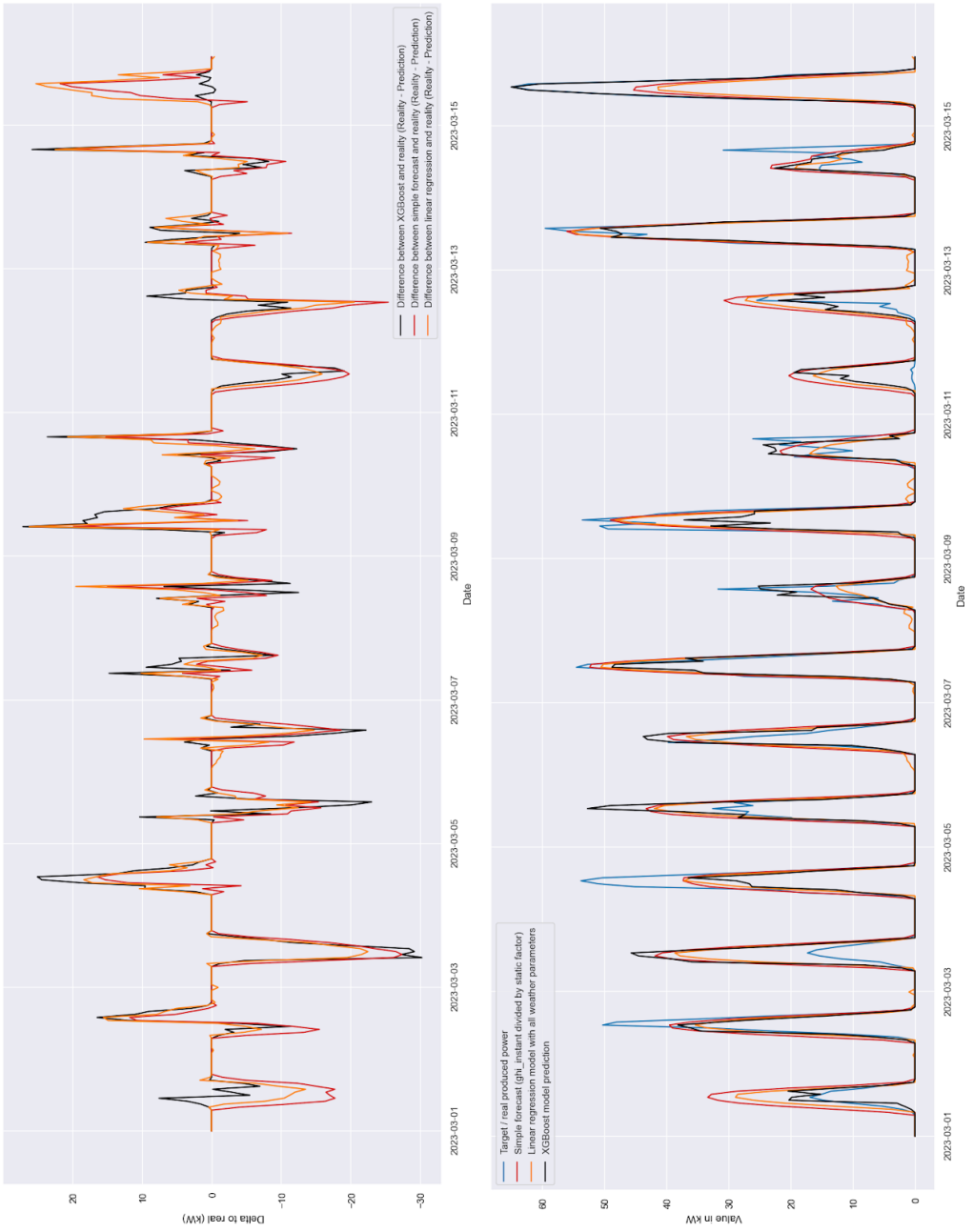

Figure 11 - Model outputs over the 2 week testing period

6. Implementation on the Edge

As was shown in the previous chapter, there is no requirement for local data to be used for forecasting future performance, once some data of the PV array is available, meaning that locally generated data - the PV array's output - is only used in model training. Model inference only utilises weather forecasts generated offsite.

However, a goal of the GENTE project is to have parts of its forecasting algorithms run on the edge in order to test whether AI-based models can be used in a low-computational environment.

In order to test this, the SmartHelio [4] platform with its Nordic nRF52840 [5] microcontroller was used as the target platform.

One challenge working with edge devices is their 'inability' to efficiently access (up- and download) data from the cloud. This is mainly due to the fact that these edge devices often are numerous and in locations where internet access, be it through WiFi or other gateways, is not easily accessible. In the case of SmartHelio's hardware, all communication would need to be done over the mobile network, and a SIM card and mobile service available. This generally means that the setup of such devices is often significantly more time-consuming than installing a service on an already existing server. Therefore, when edge computing devices are used, it is of great importance to process local data, to minimise the amount of cloud communication the microcontroller system has to do.

New developments in the GENTE project rendered the usage of pure edge intelligence for the forecasting of the PV power mostly irrelevant. Therefore the decision was made to research the on-edge-forecasting only as a minimal proof-of-concept, as the development of a more sophisticated algorithm would lead to no further gain in the overall project while still requiring an extensive amount of development time.

6.1 Target Setup

The nRF52840 microcontroller, present on an nRF52840 DK [5] development kit, shown in figure 12 (identical to the MCU setup on SmartHelio's hardware) was set up with nRF Connect; the recommended framework by Nordic for developing software for their chips. The board was connected to the host over USB, with program output shown through the integrated logging functionality.

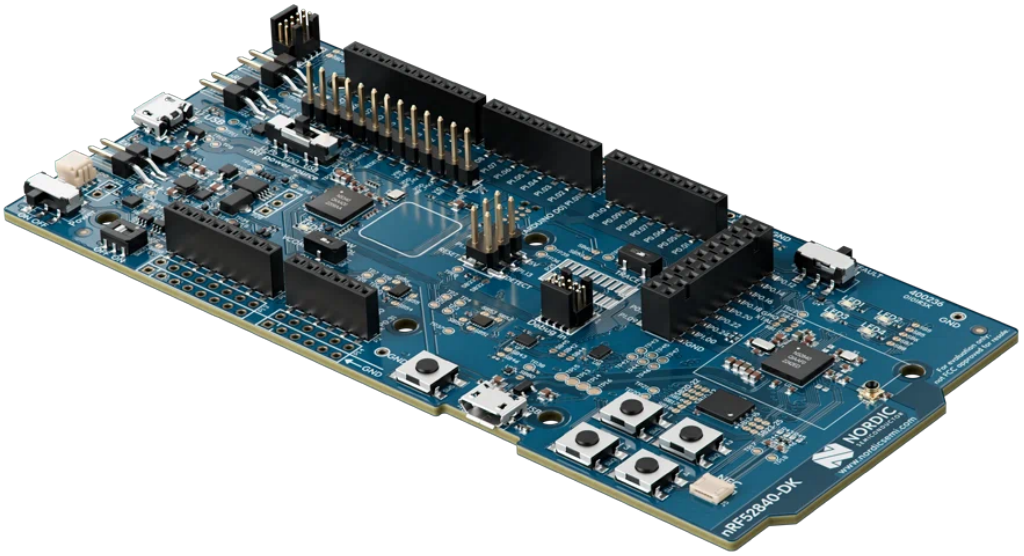

Figure 12 - nRF52840-DK development board (Source: [5])

6.2 Simple Model

During the development of the different AI models, research showed that more sophisticated models, such as an XGBoost model, lead to similar (or worse) results as a simple 1-coefficient linear regression model. Therefore the decision was made to follow the paradigm of Occam's razor[6] , and use a simple solution, rather than overcomplicate the developed system.

The resulting linear regression model can be summarised as:

```
prediction = ghi_instant * scale_factor;
```

6.3 XGBoost Model

Machine learning algorithms are inherently complex and require substantial computational resources compared to simple algorithms. This presents significant challenges when porting them to microcontrollers. Due to the limited processing power and memory capacity that is a constraint of MCUs, the porting of machine learning models to those platforms can be hard. Large models, which are often necessary for achieving high accuracy, are particularly problematic; they demand extensive memory - which microcontrollers often do not possess - and fast computation to provide inferences quickly - which, if not available, makes them take a long time.

In the case of an XGBoost model, libraries, namely treelite and tl2cgen, were identified with the capacity to translate an XGBoost model to an edge device. Utilising these libraries, it was possible to generate C code that contained the model and which would theoretically allow it to run on an embedded system. However, when trying to port it to the nRF52840, it was found that the generated code contained some incompatibility to the ARM C compiler (arm-none-eabi-gcc), which made it fail to compile. On an x86-based computer, it was however possible to compile and run the ported model, arriving at the same results as running the model directly in Python. Due to the fact that the XGBoost model was not shown to provide better performance than a simple scaling model, getting the ARM compiler to compile the model for use on the Cortex-M platform was not pursued further.

7. Conclusion

In conclusion, due to the availability of sophisticated and accurate weather predictions, it is of no use to try to squeeze out extra performance for PV-power forecasting by using a more complicated AI model with more parameters, the reason being that the weather model used as an input is optimised heavily, coupled with the fact that forecasted irradiance and actual PV power are strongly correlated.

There is a degree of uncertainty in every weather prediction; future events could cause drastic changes in the upcoming weather; a cloudy day occurring when a sunny day is forecasted will always lead to massive error function terms. Considering the fact that even state-of-the-art weather models have such vast uncertainty, a relatively simple model capable of running on the edge cannot be assumed to somehow magically lead to the possibility of predicting future weather more accurately.

Reasonable results were achieved with a linear scaling of the irradiance values, which is a decent roundabout number of solar production forecasting. As such an operation is as simple as can be, it is recommended to not run this code on a specific edge device dedicated to that task, but rather some kind of edge device or existing computer infrastructure that already is present on site and has an internet connection - some kind of server, such as the REENGEN Gateway[7] or similar, as grabbing the forecast and applying the scaling to it takes minimal time. Training for the linear scaling only has to be done once per site, using logs of previous solar output and comparing it to either past forecasts or historical weather data to scale. This could be automated if the device the model runs on also has access to the PV output, and as such a device logging and storing such information would be an ideal candidate to run that code. It would be possible to integrate the system in a way where, if the predictions were a certain amount inaccurate, more recent predictions could be automatically pulled from the meteorological API. However, it is not recommended to include this in the model itself.

If the weather model given as an input would not have included forecasts for irradiance data, it would be plausible to assume that a fairly complex model taking into account all kinds of other meteorological data could potentially be developed to forecast these irradiance values by itself.

A potentially interesting approach would be to have federated data collection with a vast network of collected weather data in various locations. This data could then be used to run a ‘democratic weather model’ on a distributed network of edge devices, using similar technologies as already exist with products such as Folding@Home[8], allowing predictions such as the ones used in this work from MeteoBlue without having to rely on corporate entities.

However, an approach such as this would be vastly different from the approaches considered here and would require considerable time and development resources along with a considerable amount of data from various geographical locations already collected.

References

[1] N. S. Dere, “Advanced energy forecasting for Local Energy Communities based on AI techniques.” Jan. 2023.
[2] Meteoblue AG, “Meteoblue,” meteoblue. Accessed: Dec. 19, 2023. [Online]. Available: https://www.meteoblue.com/
[3] Meteoblue AG, “Forecast Data · Technical Documentation.” Accessed: Oct. 30, 2023. [Online]. Available: https://docs.meteoblue.com/en/weather-apis/packages-api/forecast-data
[4] SmartHelio AG, “SmartHelio,” SmartHelio. Accessed: Dec. 19, 2023. [Online]. Available: https://smarthelio.com/
[5] Nordic Semiconductors, “nRF52840 DK.” Accessed: Nov. 08, 2023. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK
[6] “Occam’s razor,” *Wikipedia*. Dec. 06, 2023. Accessed: Dec. 19, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Occam%27s_razor&oldid=1188624085
[7] “Reengen Energy IoT Platform.” Accessed: Jul. 21, 2022. [Online]. Available: https://www.reengen.com/
[8] “Folding@home – Fighting disease with a world wide distributed super computer.” Accessed: Dec. 19, 2023. [Online]. Available: https://foldingathome.org?lng=en

FUNDING

This project has received funding in the framework of the joint programming initiative ERA-Net Smart Energy Systems' focus initiative Digital Transformation for the Energy Transition, with support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883973.