Light Learning Estimator Challenge

Develop a light-weight local predictor that shares a local forecast based on smart meter data.

Title	Light Learning Estimator (LLE) Challenge	
	A description of a 'challenge' to arrive at a continuous learning algorithm, which produces an accurate forecast for a household with light hardware.	
Version	21 Dec 2021	
Contact	Leon de Jong Hans Fugers	Leon.de.jong@ hans.fugers@

Lite Learning Estimator Challenge

Background of the challenge

Alliander shapes the energy future of the Netherlands

We stand for an energy supply that gives everyone access to reliable, affordable and sustainable energy under equal conditions. That's what we work on every day. By continuously improving our network, we prepare for the future. A future in which everyone can use, produce and share sustainable energy.

Alliander develops and manages energy networks. More than three million Dutch households and businesses receive electricity, gas and heat via our cables and pipelines. We manage more than 90,000 km of electricity grid and 40,000 km of gas grid and are proud that our grids are among the most reliable in the world. Our 7000 colleagues ensure that the light is on, the houses are warm and the businesses are running. We do this in the interests of society to keep energy reliable, affordable and accessible for everyone.

This assignment has been drawn up by the research center and directly contributes to the reliability of our energy grid. The Research Center for Digital Technologies is shaping the digital networking company of the future. Through this challenge we investigate how we can use small forecasts, namely those of the individual householdns, added together to make a better forecast and for the consumptionof tomorrow. With a better forecast, weare better able torespond to anyproblems. Through this challenge you contribute directly to the energy transition!

The Light Learning Estimator

The Light Learning Estimator **(LLE)** periodically makes a forecast for each individual household. These households provide cumulative insight into any problems at district level. This software should be able to operate on light weight hardware, for our test this will be a Raspberry Pi 4. The software generates a forecast of your and energy consumption and energy twice a day. When something changes in your use, for example, by purchasing solar panels or a charging station at home, the estimator adjusts within a few days and the forecast becomes accurate again.

The LLE is a continuous learning, predictive algorithm that runs on light hardware located at the ends of the network. The advantages for users and Alliander/Liander at a glance:

Advantages of Alliander:

- Better insight into expected energy consumption of households, and therefore district and the electricity grid
- And thus being able to better remedy malfunctions

Benefits for users:

- Privacy-sensitive data stays within the household's smart meter

In this challenge you will work to develop this piece of software. Of course, there is also something in return:

In addition to eternal fame, the first prize is €1,000,-. The deadline for submission is 6 February 2022, 23:59. Have fun!

Lite Learning Estimator Challenge

Contents

Background of the challenge 2
Outputspoints ..... **Fout! Bladwijzer niet gedefinieerd.**
Deadline challenge 4
Prize money 4
Starting material 4
Solution development 4
Handing in solution 5
Verification and results 5
Training dates 5
Usage data 5
Weather data 6
Format 6
Delivered product 6
Continuous learning 7
Review 7
Relevant test days 7
Analysis quality forecast 8
The winner 9
Special entry 9
Outcomes and feedback 9
Attachments 10
Appendix 2 : Format user data 10
Appendix 3 : Table applications houses 10
Appendix 4 : Format weather data 11
Appendix 6: Hardware and OS Raspberry Pi 12
Docker 12
Appendix 7: Details forecast and measurement 13
Details: forecast 13
Details: Call up historical readings 13

Basis for the challenge

The idea is that a device (agent) is connected to the smart meter of a residential house, which learns to make an accurate forecast of the household and adapts to changing conditions in energy use (continuous learning).

A starting point is that the winning solution in terms of algorithm and architecture are made public, In consultation this can be made public later in terms of timing to first publish about it in other media, with reference to this challenge.

Alliander remains the owner of the delivered entries at all times.

In addition, the winning solution must be able to run on limited hardware, in which we take the Raspberry Pi 4 as a starting point. In addition to testing the quality of the forecast, it is also examined whether the solution can run sufficiently fast on this Raspberry Pi.

The generic time frame for data is 5 min. So we expect a forecast per 5 minutes for the next 24 hours. The test is also done over 5 minutes. So at a higher frequency of input, it is averaged (or added up) over the 5 min.

Deadline challenge

The end date of the challenge is Sunday 6 February 2022, 23:59. Feedback on the winners will then take place on Monday 21 February.

Prize money

In addition to eternal fame, the Winner will also receive 1000 euros. You win by making the best forecasts in the assessment periods. Further details are below.

The second place receives 500 euros.

Starting material

The datasets of Smart Meter-data are covered by a Non-Disclosure Agreement (NDA) and may only be used for the development of a solution. The data must be deleted afterwards,as included in the processing agreement to be signed [see here why this is necessary].

We define the hardware specifications that will process the dataset. In case of questions and/or additional information, this will be provided to all participants.

Alliander provides a scoring mechanism for the s on the basis of which we will designate a winner.

Solution development

The parties develop a solution that generates predictions on the specified hardware.

The participant provides the solution in the form of a working Python script (ver 3.10), R-script (ver 3.6.0) or docker-compose, global description of the solution approach and installation instructions for test data, which we can run on the specified hardware and the own evaluations of the prediction with the scores.

We expect an input control of test data, which guarantees the correct processing of the delivered algorithm.

Submitting your solution

Solutions can be submitted in two ways:

- Upload the solution on your personal repo and give us access
- Mail to leon.de.jong@

Verification and results

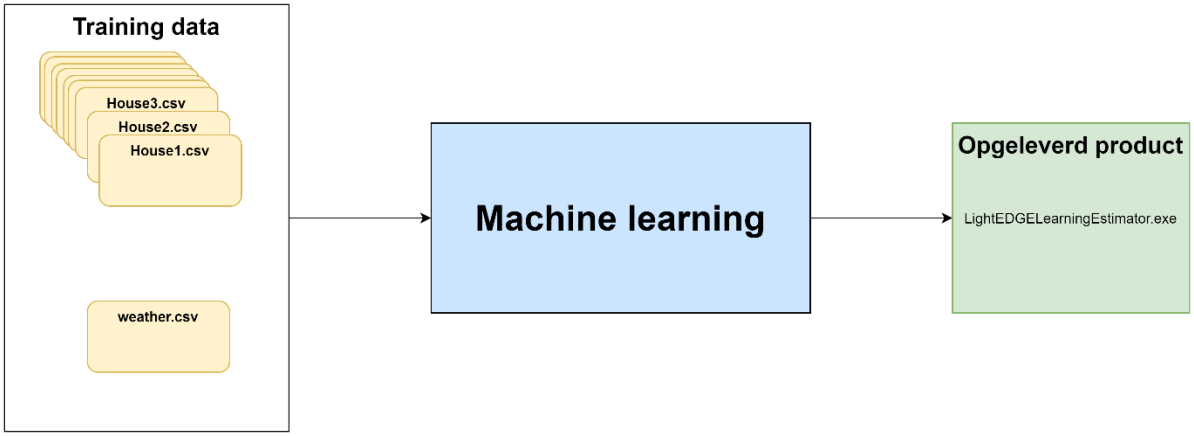
Alliander carries out a verification with a number of test scenarios in the form of test data, in which changes are included in some cases, namely the addition of solar panels **(PV:** photovoltaic)

If test data leads to errors, contact is sought to repair them within a few days. This is done a maximum of 2x.

After going through all test scenarios of all delivered solutions, these are evaluated and the results published to all participants and within Alliander.

Supplied data

After the Non-Disclosure Agreement (NDA, see Appendix 1) and processor agreement have been signed, you will have access to the data. The data consists of two sets: Energy data from households and weather data. Both datasets are about the same time period: 01-08-2018 to 31-08-2020. So these are 762 days (365+366+31).



Energy usage data

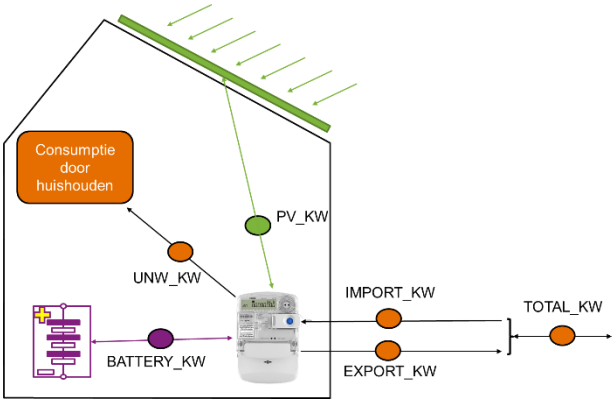
Usage data consists of minute values. They are datasets from multiple houses, and can be divided into three situations (with the columns belonging to each dataset below):

- Houses with Smart Meters (SM)
  `[timestamp, house, IMPORT_KW, EXPORT_KW, TOTAL_KW, UNC_KW]`

- Houses with Smart Meters and Solar Panels (SM and PV)
  `[timestamp, house, IMPORT_KW, EXPORT_KW, TOTAL_KW, UNC_KW, PV_KW]`

- Houses with Smart Meters, solar panels and a battery (SM, PV and Battery)
  `[timestamp, house, IMPORT_KW, EXPORT_KW, TOTAL_KW, UNC_KW, PV_KW, BATTERY_KW]`

The columns of the datasets therefore depend on the situation. In the picture below it is shown again:



Weather data

Weather data consists of a large dataset in hourly values, including temperature(, global radiation(J/cm$^{2)}$ and sunshine duration (in 0.1 hours, so 10= whole hour of sunshine). The definitions of all columns can be found in Appendix 3.

Format

Power data in kilowatts (kW):

- 60 CSV files, a separate file for each house
- Appendix 4 shows which house has which situation (SM, SM+PV, SM+PV+BATTERY)

Weather data

- CSV files

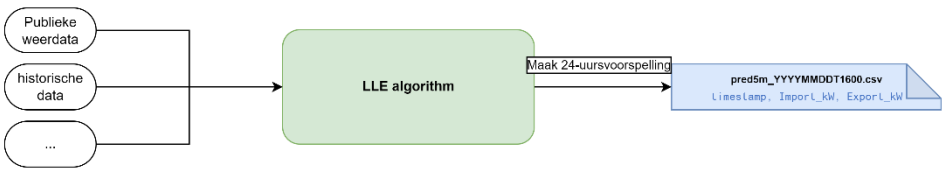
Delivered product

The delivered product must meet the following conditions:

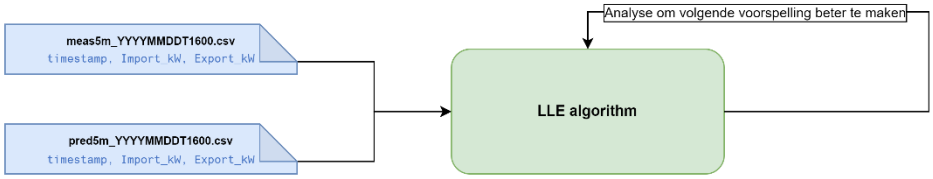
1. Every day at 04:00 and 16:00, the CSV-file for forecasts is created with the 24-hour forecast for Import and Export, including timestamps in blocks of 5 minutes. This forecast is made in 288 5-minute values, which predict how much kW is used on average during that period. This forecast is saved as a CSV file. Of course, weather forecasts, historical data, and whatever is needed for a good forecast can be included. Further details can be found under Details: forecast.

   Format CSV with headers:

```
Timestamp, Import_kW, Export_kW
2021-11-22T14:00:00+02:00, 0.123, 1.123
2021-11-22T14:05:00+02:00, 0.234, 0.231
```



2. The LLE continues to learn*(continuous learning),*so that in the event of a change in the energy pattern, it can make accurate forecasts again after a period of time. For example, if solar panels are purchased, both the patterns of `Import_kW` and `Export_kW` will change, LLE should be able to deal with this quickly. For example, the LLE can compare measurement data, i.e. the real energy patterns, with the forecast of that day(s). The actual energy patterns are stored in the same format as the forecasts.



Further details on in which folders etc. the files should be stored can be found in Appendix 7.

Continuous learning

Based on the quality of the previous forecasts, the self-learning algorithm can then sharpen new forecasts, so that the forecasts (on average) get better and better. Even if a change in `Import_kW` and `Export_kW` -patterns occurs, it is the intention that the algorithm can deal with this, and make a reasonable forecast within 7 days. The implementation of this is up to the participant.

Review

If your solution of the Light Learning Estimator has been handed in, we will test it as soon as possible. This takes place in the following way:

The assessment is done on the basis of Smart Meter-data from houses that have been withheld in advance. Some of these contain solar panels, others not or later. The delivered product is always fed the data of a single household to see how well the energy pattern of that specific household can be predicted.

After it has been connected, the measurement data of that day is fed in every 12 hours (at 04:00 and 16:00), after which the algorithm always makes a forecast and can learn on the basis of the actual measurement values. Historical weather data from that day is also available as a 'forecast' in the format in which it was also delivered. If you use smart other external sources for a good predictor, it is useful that you provide historical values from the period 01-08-2018 to 31-08-2020, so that we can also feed them in.

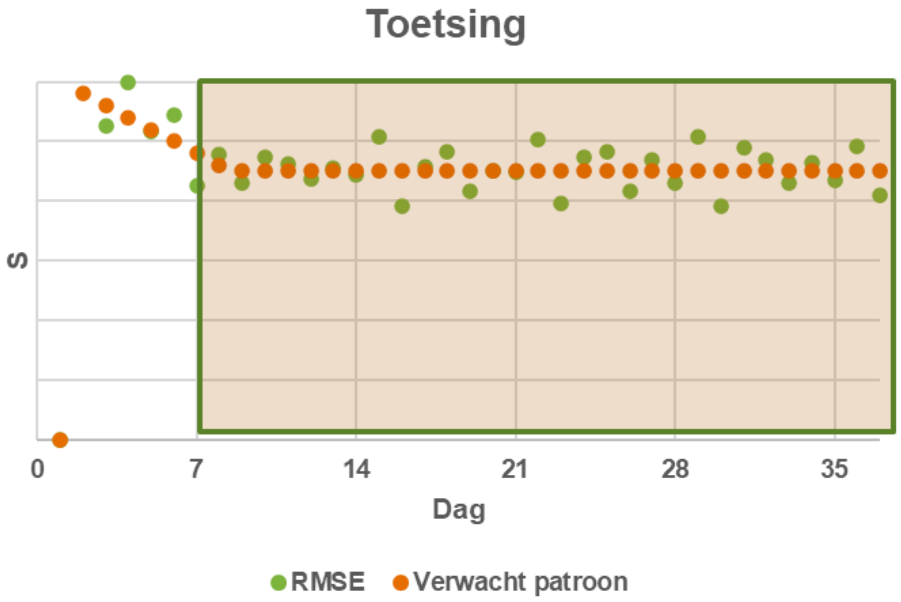
Then, for each forecast made by the LLE, the real values are tested. This is done using the Root Mean Square Error method. This results in a value for every day.

N.B. The sequence of events at 04:00 and 16:00 is as follows:

1. CSV with forecast is retrieved
2. CSV with measured values is fed and RMSE is determined
3. New forecast is made by submitted LLE algorithm

Relevant test days

First, the product is given 7 days to get used to the usage pattern, and then ends up in a 'test period of 30 days, in which the RMSE values of those days are looked at. This leads to 60 RMSE values (2 per day*30 days).



Then after x number of days the same thing is done again. This value x is predetermined, and ensures that the second assessment takes place just after solar panels are installed in some houses, to see if the LLE can cope with this change. This value x is therefore known to the jury in advance.

The average RMSE value is determined after the LLE has been tested on data from several houses, and the one who has the lowest RMSE value on average across the scenarios wins.

Analysis quality forecast

24 hours after the forecast is made, an evaluation takes place of how well the forecasts of `Import_kW` and `Export_kW` corresponded to reality. For each 5-minute value, the absolute difference between the forecast and reality is looked at. The RMSE formula is applied:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

This is done both for the forecast of `Import_kW` and `Export_kW` by calling up `meas5m_20211122T1600.csv` and `pred5m_20211122T1600.csv` and determining the difference.

A small working example with 7 random `Import_kW` -values (in the actual challenge this will of course be 288 per day for generation and 288 per day for consumption:

timestamp	IMPORT_KW	PRED_IMPORT_KW	delta
2021-11-22T14:00:00+02:00	1.08	1.12	0.04
2021-11-22T14:05:00+02:00	1.34	1.42	0.08
2021-11-22T14:10:00+02:00	1.45	1.23	0.22
2021-11-22T14:15:00+02:00	1.21	1.32	0.11
2021-11-22T14:20:00+02:00	1.32	1.33	0.01
2021-11-22T14:25:00+02:00	1.29	1.22	0.07
2021-11-22T14:30:00+02:00	1.28	1.25	0.03

		RMSE
		0.28

For `Import_kW` and `Export_kW`, an RSME value is kept separately.

The winner

The winner is the one with the lowest average score *RMSE*.

Of course, we're also testing the algorithm on a Raspberry Pi 4, and it should be able to *run* on it.

Special entry

It may be that someone finds a completely different way to make a more accurate forecast, for example, the use of other hardware or other publicly accessible (open) data.

An entry in this category must describe in sufficientdetail, where it deviates and what the improvement is that is achieved with it. By satisfactorye we mean that we can independently test the design on the basis of the description and supplied code.

The final score must then score at least 50% better than the competition.

Outcomes and feedback

Two weeks after the deadline, on 21 February, the winners will be announced.

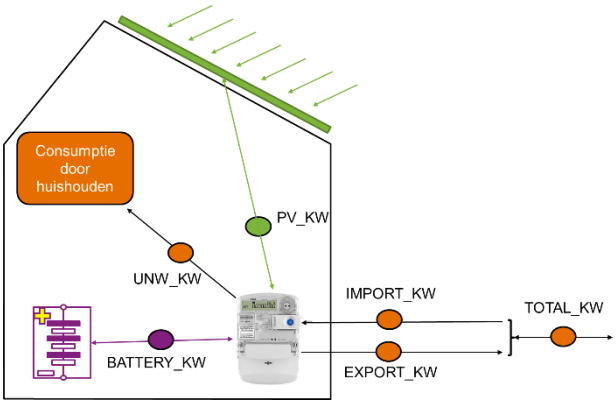
Attachments

Appendix 2 : Format user data

Column name	meaning	Unit/format
Timestamp	time	yyyy-mm-ddThh:mm:ss+02:00
House	Indicates which house it is	-
IMPORT_KW	Power from grid to SM	kW
EXPORT_KW	Power from SM to grid	kW
TOTAL_KW	IMPORT_KW – EXPORT_KW	kW
PV_KW*	Power generated by PV	kW
BATTERY_KW**	Power from battery to SM	kW
UNC_KW	Power used by household	kW

* Only for houses that get solar panels over time

**Only for homes that have a battery over time



Appendix 3 : Table applications houses

1 = 'house owns this application'

House	Smart meter?	PV panels?	Battery?
1	1	1	1
3	1	1	
4	1	1	
5	1	1	1
7	1	1	1
8	1		
9	1	1	
10	1	1	
11	1	1	
13	1	1	
15	1	1	1
16	1	1	1
17	1	1	
19	1	1	
20	1	1	
21	1	1	

24	1	1	
25	1	1	
26	1		
27	1		
28	1	1	
29	1	1	1
30	1		
32	1	1	
33	1	1	
34	1		
35	1	1	
36	1		
37	1		
38	1	1	
39	1		
40	1	1	
42	1		
43	1	1	
44	1	1	
45	1	1	
46	1	1	
48	1	1	
50	1	1	
52	1	1	
53	1		
56	1	1	
57	1	1	1
58	1		
59	1	1	
60	1	1	
61	1		
62	1		
63	1		
64	1	1	
66	1		
68	1		
69	1	1	
71	1		
72	1		
73	1		
75	1		
Total	57	37	7

Appendix 4 : Format weather data

Source: https://www.knmi.nl/nederland-nu/klimatologie/uurgegevens, weather station 278 – Heino, is 15 km from the houses

Appendix 6: Hardware and OS Raspberry Pi

We use a Raspberry PI4+ with 4 Gb memory and as OS the latest version of 'Raspberry Pi OS'

After installation, an update and upgrade is performed:

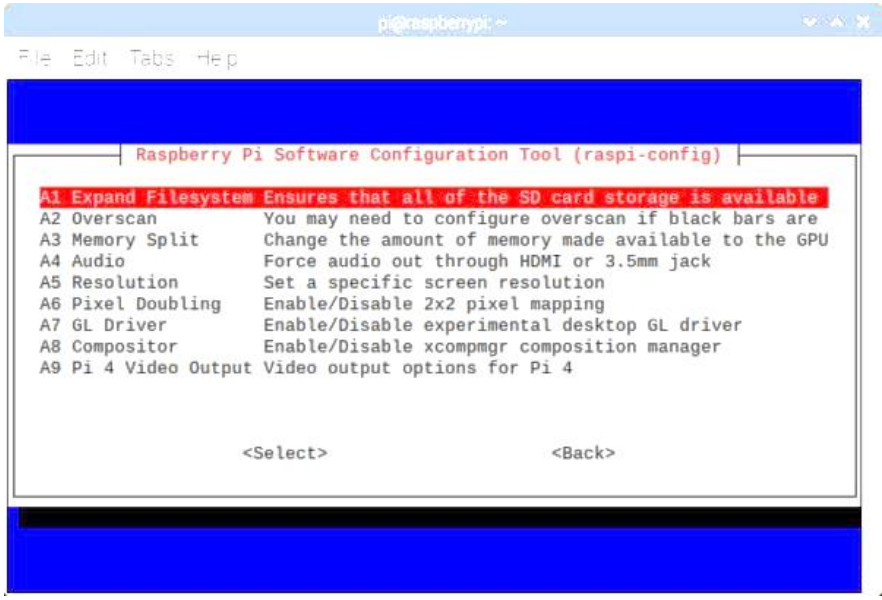
So in the terminal *: <ctrl><alt>-T*

```
$sudo apt-get update
$ sudo apt-get full-upgrade
```

Expand file-system to get more space

```
$ sudo raspi-config
```

Selecteer optie 7 Advanced Options
Selecteer then A1 Expand filesystem



Restart with

```
$sudo reboot
```

With the command

```
$DF-H
```

one sees the result of the space obtained.

For even more space :

```
$ sudo apt-get purge tungsten engine
$ sudo apt-get purge libreoffice
$ sudo apt-get clean
$ sudo apt-get autoremove
```

Docker

See also link URL

For the installation of docker download the installation script with:

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
```

Execute the script with:

```
$ sudo sh get-docker.sh
```

To make life easy for the current user and make sudo unnecessary for docker tasks:

```
$ sudo usermod -aG docker &USER$
```

The default RPI user - $USER$ can also be written as 'pi'.

Appendix 7: Details forecast and measurement

Details: forecast

Every day at 04:00 and 16:00 a forecast is made about the import and export values of the next 24 hours. The forecast makes a forecast of each time block of 5 minutes. This leads to 24*12 = 288 values for energy use(Import_kW)and feed-in (Export_kW).

This results in a series of values with time-stamp (start of the 5 min interval) and 2 float values that indicate the average Import and Export kW during that 5 min. The sum of these two values is therefore the net consumption by the household, which normally constitutes the planned load on the electricity grid.

To test these forecasts, these sequences must be stored in CSV files.

The file is named pred_<timestamp>. csv at the location: `/home/predictions`

For example, file name : `pred5m_20211122T1600.csv`

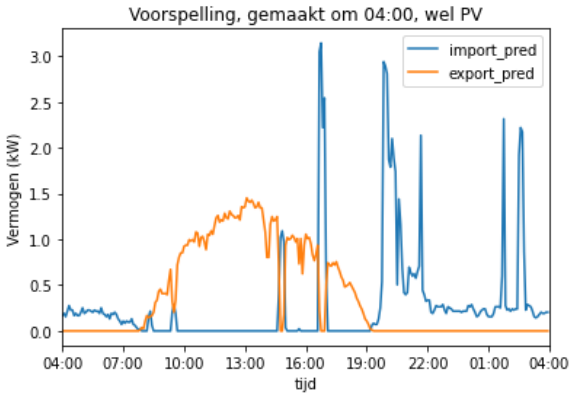
`Can be related to training data`

Format CSV with headers:

```
Timestamp, Import_kW, Export_kW
2021-11-22T14:00:00+02:00, 0.123, 1.123
2021-11-22T14:05:00+02:00, 0.234, 0.231
....
```

N.B. The timestamp the beginning of the interval on, so 2021-11-22T14:00:00+02:00 contains average Import and export (kW) between 14:00:00 – 14:04:59 on 22-11-2021.

Plotted sees the . for example, csv looks like this:



Details: Call up historical readings

To make the algorithm *continous learning,* it can be useful to test the forecast after 24 hours against the actual measurement data.

For each day, the measurement data can be found under the name meas5m_<timestamp>. csv at the location: `/home/measurements`

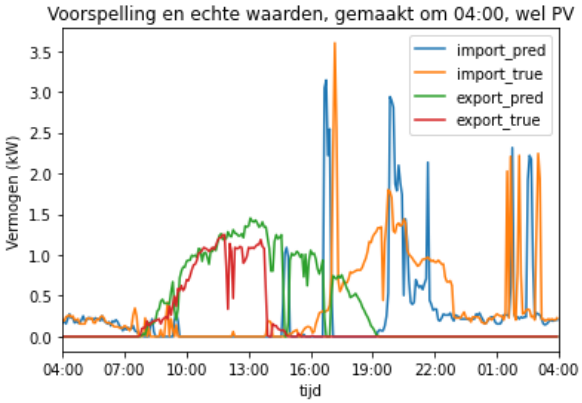
For example, file name : `meas5m_20211122T1600.csv`

Format of this CSV with headers:

```
Timestamp, Import_kW, Export_kW
2021-11-22T14:00:00+02:00, 0.123, 1.123.
2021-11-22T14:05:00+02:00, 0.234, 0.231.
```

While the forecast file is filled with 288 values at 04:00 and 16:00, a line is added to the measurement file every 5 minutes for 24 hours.

Below is an example of when `meas5m_20211122T1600.csv` and `pred5m_20211122T1600` would be plotted after 24 hours:



========== the end ==========