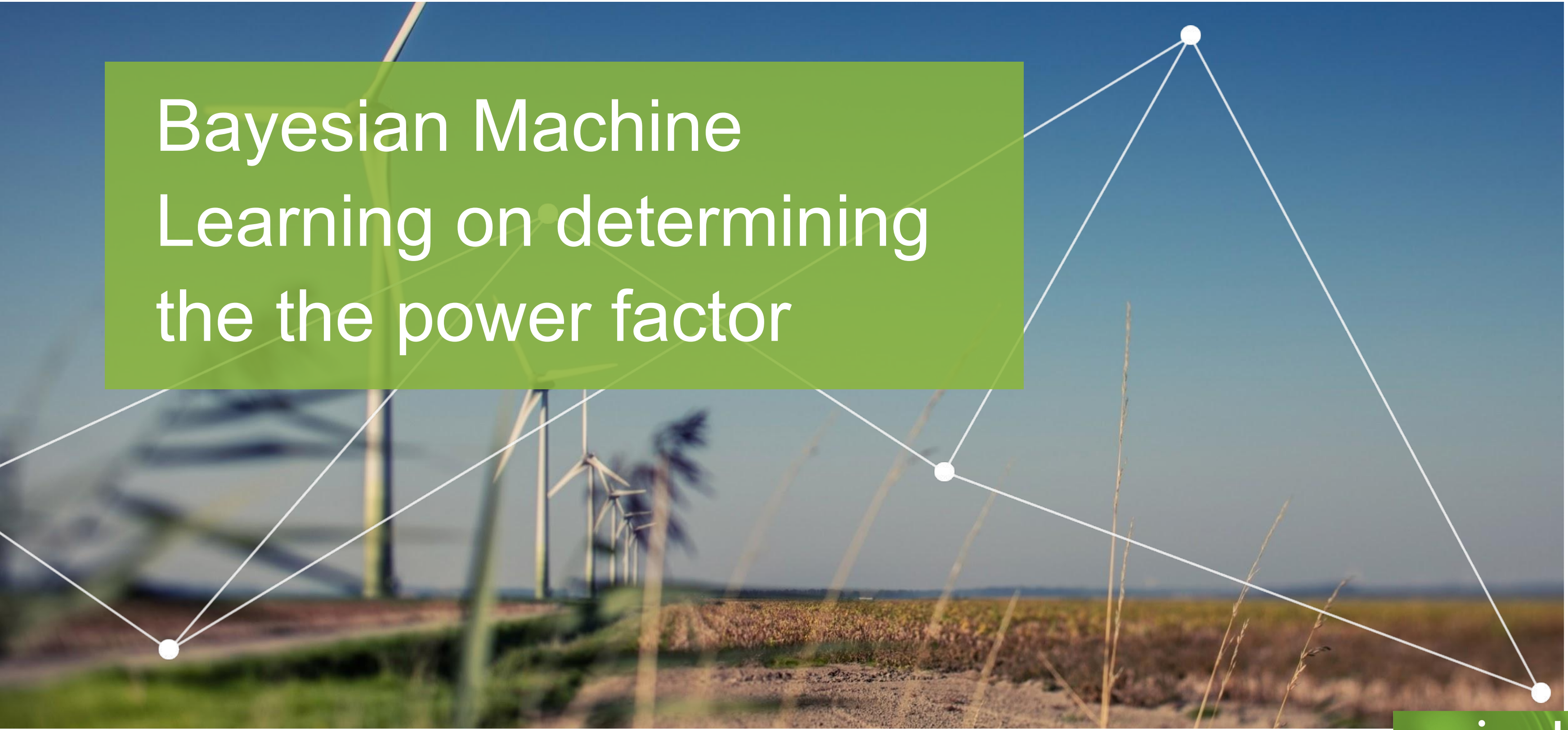
# Bayesian Machine Learning on determining the the power factor



David Teulings I Advanced Analytics I 03-04-2023

# What is there to be learned?

- Classically Energy flow: Producer → Consumer
  - Easy to maintain balance between demand and supply
  - General: Single direction of power flow

- With decentralised power generation
  - Power flow in Middespannings (MS) net becomes muddled
  - Hard to keep up with supply, demand, and capacity

  ⟶ Big bottleneck in the energy transition

- Current measurements insufficient to understand
  - Most MS substations only record absolute current (I) and voltages (U)
  - Look for a machine learning solution to learn the active power (P)

# What is there to be learned?

Quick recap

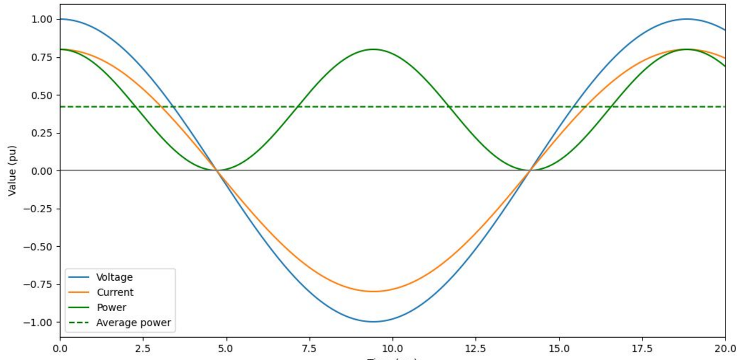
Alternating current grid:

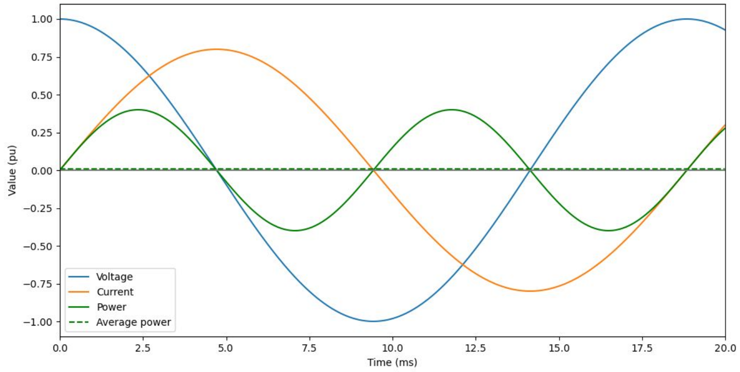
- S = U x I x √3
    - Apparent power (S)
    - Theoretical maximum

- S = √(P² + Q²)
    - Apparent power (S)
    - Hypotenuse of the active power (P) and the reactive power (Q).
    - P is the actually used/consumed power
    - Q is the non-used power → wasted

- P/S = Cos(ϕ)
    - P/S is the power factor [-1,1]
    - P and Q are determined by the phase angle (ϕ)

- If we can learn P from available data we are able to calculate the power factor at MS substations



# Quick overview of bayesian machine learning

## Why Bayesian machine learning?

- It allows for statistical models that can be updated with prior knowledge expressed as distributions
  - Observational data is combined with the prior and normalised to give a posterior prediction

- Great! Perhaps we can encode common constraints of the grid in the prior to help in predicting power flow:
  - E.g. Kirchhoff's current law ( $\Sigma(P) = 0$ )
  - $P \leq S$ (realistically always $P < S$)

$$\underbrace{\boxed{P(A|B)}}_{\text{posterior}} = \underbrace{\boxed{P(A)}}_{\text{prior}} \times \frac{\overbrace{\boxed{P(B|A)}}^{\text{likelihood}}}{\underbrace{\boxed{P(B)}}_{\text{marginal}}}$$

# Quick overview of bayesian machine learning

## Gaussian Processes

- I decided to try apply Gaussian Processes (GP) as the Bayesian machine learning method
  - Specifically GP regression
- GP Regression (GPR)
  - Nonparametric: Not limited by a function form
  - GPR → distribution over all admissible functions fitting the data
  - Can be described as an infinite-dimensional multivariate gaussian distribution

$$f(x) \sim GP(m(x), k(x, x'))$$

- Gaussian prior
  - Mean function: m(x)
  - Covariance function: k(x,x')

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix} \right)$$

- Training data
  - Gets applied to the covariance matrix to create the kernel matrix
  - Does not require a lot of data to train
- GPs allow for:
  - Interpolation of observations (useful for holes in the data)
  - Probabilistic predictions (confidence intervals)
- GP weaknesses
  - Computationally costly for high dimension feature spaces

# What I ended up doing

## Three uneven topics

- Main focus on Bayesian machine learning / Gaussian processes
- However two (sort of) separate ideas

1 Look at the possible use of flow networks for energy grids

- One master thesis seemed promising
    - "*Network flow models for power grids*", Francesca Wegner, 2014
- Required knowledge of wider energy flow in the grid (Outside of the scope of my internship/topic)

2 Look at the construction of custom loss functions for general machine learning models (XGBoost)

- Some physical laws can be encoded in the loss function
    - Loss: MSE + $\Sigma(P) = 0$?
    - Prevents 0 predictions, still accounts for some part of the physics

3 Before learning the actual power factor, first check if we can accurately predict P

- How should we present the data?
- How do we ensure the model understands physical limits (E.g. Kirchhoff's current law)?
- Which features do we add?
    - Geographical data?
    - Weather data?

- Actual meat of the internship!

# A tale of two datasets

Original MSE data:

- 1 year, 10 stations
- .Time series data (measurement / 5 min)
    - per station, per installation, per field

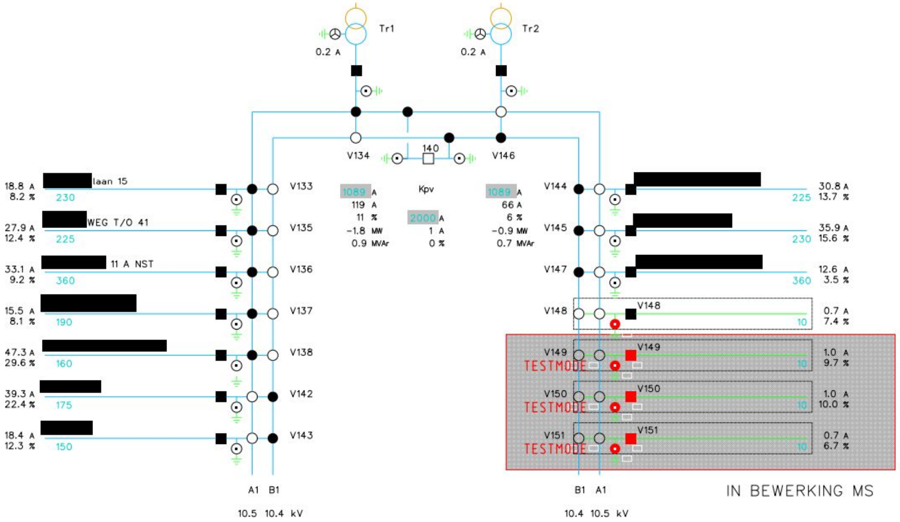
- Contains absolute current, voltage, active and reactive power
    - Unfortunately Kirchhoff's current law rarely held up (At worst 12 MW deviation, total ~35MW → ~ 30% deviation)

Decided to focus on Kirchhoff's current law as my primary physical law to incorporate

- How to combine spatial and temporal features?
- No good method found

Main features would be:

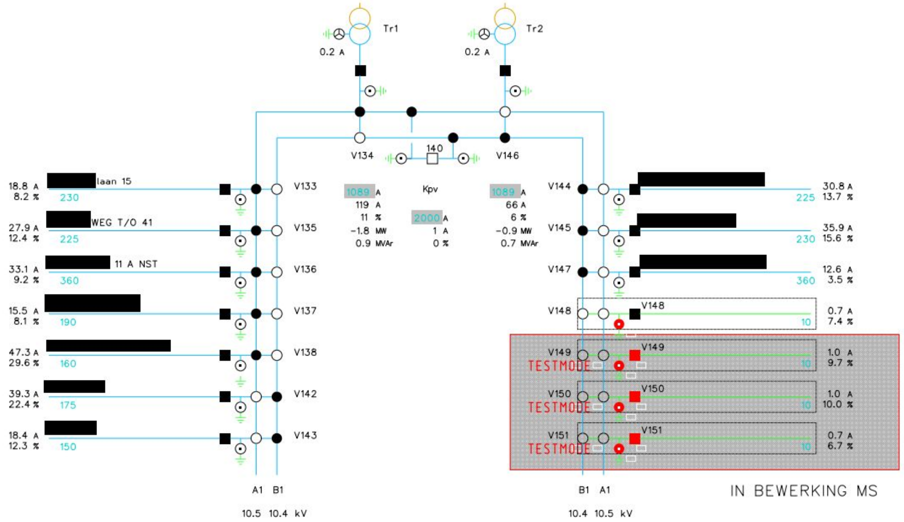
- Train: Current (I), voltage (U)
- Test: Active power (P), reactive power (Q)



# A tale of two datasets

Decided to use two datasets

- Time series data
  - Time relation through lagging data:
    - P at t, t-1, t-2, etc
- Spatial data
  - No (real) time relation between data points
  - Single data point = entire station
  - Amount of features: F x N
    - F = amount of fields per station
    - N = amount of measurements per field
- Big problem!
  - Variable size of stations results in different amount of features
  - → Do we train per station or in general?
    - Resulting in idea for other experiment



# Spatial data

| | DATUM_TIJD | STATION | I_0 | U_0 | I_1 | U_1 | I_2 | U_2 | I_3 | U_3 | ... | I_20 | U_20 | I_21 | U_21 | I_22 | U_22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | Tex | 0.000000 | 0.0105 | 76.790000 | 0.0105 | 44.380000 | 0.0105 | 52.980000 | 0.0105 | ... | 0.0 | 0.0000 | 0.00 | 0.0000 | 0.00 | 0.0000 |
| **1** | 0.0 | Hby | 2.080332 | 0.0105 | 25.878577 | 0.0105 | 2.743656 | 0.0105 | 9.843535 | 0.0105 | ... | 0.0 | 0.0000 | 0.00 | 0.0000 | 0.00 | 0.0000 |
| **2** | 0.0 | Dtn | 65.649983 | 0.0105 | 45.709995 | 0.0105 | 6.767125 | 0.0105 | 0.000000 | 0.0000 | ... | 0.0 | 0.0000 | 0.00 | 0.0000 | 0.00 | 0.0000 |
| **3** | 0.0 | HFDP | 0.000000 | 0.0210 | 0.000000 | 0.0210 | 127.780000 | 0.0210 | 126.570000 | 0.0210 | ... | 0.0 | 0.0000 | 0.00 | 0.0000 | 0.00 | 0.0000 |
| **4** | 0.0 | Grd | 0.000000 | 0.0105 | 56.730000 | 0.0105 | 57.390000 | 0.0105 | 0.000000 | 0.0105 | ... | 0.0 | 0.0105 | 22.19 | 0.0105 | 56.78 | 0.0105 |

Internship became more and more about data exploration than machine learning.

- What data to use (temporal vs spatial)?
- How to train (stations vs general)?
- Influence of different time of year?
- What/how to add knowledge about fields?

# Experiments

Machine learning models straight out of the box (scikit-learn)

- Gaussian process
- XGBoost

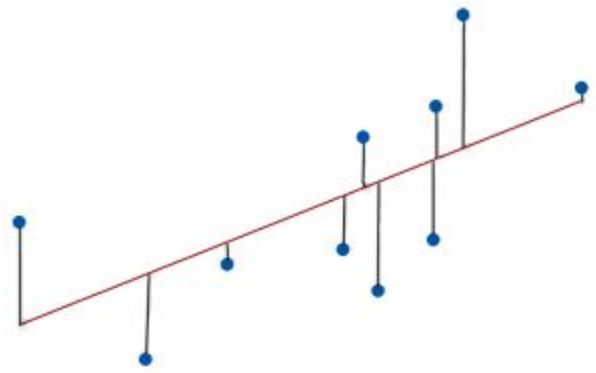
Experiment variables:

- Temporal or spatial data
- GP or XGBoost
- Train per station or on periods of 2 weeks
- Additional feature (Thanks to Jacco)
    - Treat all fields equal or differentiate between incoming and outgoing fields

Metrics used:

- MSE
- Accuracy on sign prediction

Difficult to determine when a prediction was good without visual inspection



# Results from experiments

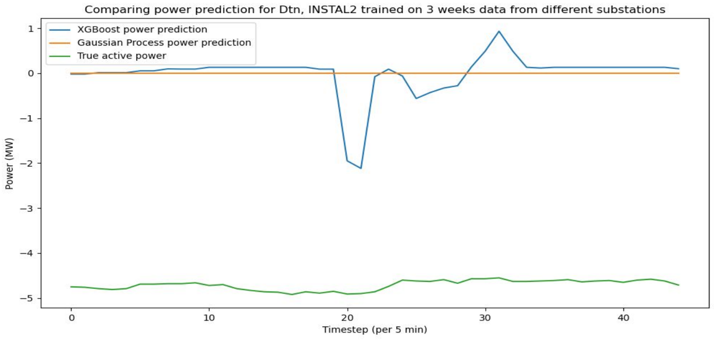
Metric wise:

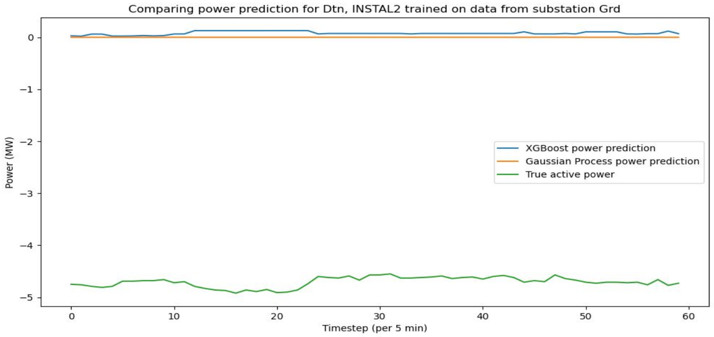
- Mixed results:
  - ~ 60-90% accuracy predicting sign (Promising)
    - Some outliers (50% or 95% accuracy)
  - ~ 5-10 MSE (painful)
    - Meaning an average error of ~√5 - √10 MW

Notice a slight improvement in performance on training on individual stations.

However with visual inspection we soon see the results

- GP acc: 0.82, mse: 6.48
- XGB acc: 0.63, mse: 12.17



# Conclusion

Disclaimers

- My code is still quite buggy/messy
  - Order of stations/fields does not match between spatial and temporal data
  - Splitting for cross validation between weeks and stations has inconsistencies

- Training for GPs had to be done on small sets due to intensive memory requirements
  - Therefore the training for XGBoost was limited to the same data size, to keep the comparison fair

- Error below obtained training GP on 2 weeks of data

```
MemoryError: Unable to allocate 75.7 GiB
```

Take-away messages

- Further research required for data sanity
  - Does the data match the expected values?
  - Often I found myself in situations where it did not

- GPs are powerful, but perhaps unwieldy for a grid wide scope
  - Perhaps look in to (GP) models for individual stations?

- Further research in to developing a loss function
  - XGBoost seems widely used in Alliander
  - Incorporating simple constraints such as Kirchhoff's current law in to a loss function and implementing it

- Further research on the best way to represent the temporal and spatial information of the grid
  - Graph Neural Networks?

- Incorporating concrete constraints (Such as sum of predictions = 0) in to GPs is very complex
  - "*Incorporating Sum Constraints into Multitask Gaussian Processes*" Philipp Pilar