46765 Machine Learning for Energy Systems

Assignment 1: Renewable energy trading in day-ahead and balancing markets
Code: `https://github.com/albertkjoller/ml-energy-systems`

Albert Kjøller Jacobsen (s194253)
Ilias Manolakis (s222834)
Tais Abrahamsson (s193900)
Denisa Enache (s222964)
Andrei Lavric (s222965)

11th of October, 2023

Contents

Contribution table

	Model development	**Coding**	**Results analysis**	**Report writing**
Albert	30%	20%	22%	20%
Ilias	15%	20%	15%	20%
Tais	10%	20%	25%	20%
Denisa	25%	20%	18%	20%
Andrei	20%	20%	20%	20%

Notation

Table 1: Mathematical conventions used throughout this project report

Notation	Type	Size	Description
$\boldsymbol{x}$	Numeric	M	A single input vector (or covariates) of M covariates.
$\boldsymbol{X}$	Numeric	$N \times M$	Input features (or covariates) with N being the number of data points and M being the number of attributes / covariates.
y	$\mathbb{R}$	1	Target variable associated with a single input point, e.g. $y \in \mathbb{R}$ for regression.
$\boldsymbol{y}$	$\mathbb{R}^N$	N	A collection of target variables.
$f(\boldsymbol{X})$	Function	N	The model, being either a classifier or a regression model that seeks to approximate $\boldsymbol{y}$, i.e. $f(\boldsymbol{X}) \approx \boldsymbol{y}$. Output is of size N.

1 Introduction

This assignment focuses on energy trading markets, mainly day-ahead and balancing markets, employing linear and non-linear regression with regularization techniques and utilizing real-world data, such as actual wind power, market prices, and climate data from 2021 to 2022. The primary goal of the assignment is to deploy two models for trading in these markets. Model 1 is created to implement regression to predict wind power, calculate revenue, and optimize trading models. On the contrary, Model 2 directly determines the optimal offering strategy using supervised learning.

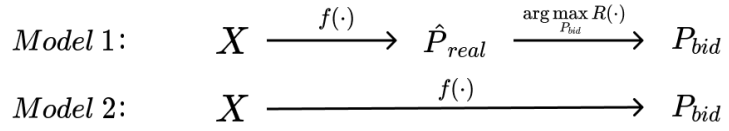

Figure 1.1: Overview of the two modeling approaches considered for determining the optimal offering strategy. *Model* 1 approaches the task by using the predicted power production, $\hat{P}_{real}$, to solve the optimization problem (see Section 1.1) for the decision variable, P_{bid}. Contrarily, *Model* 2 has direct access to historical offering strategies for which reason a supervised model, $f(\cdot)$ is directly applied to learn the mapping from input features, $\boldsymbol{X}$, to the target, P_{bid}. This was revised with Thomas before handing-in.

1.1 Formulating the optimization problem

Assumptions

1. The price is independent of what the level of production. (The wind farm is a price-taker) This means that the power bid by the wind farm does **not** have an effect on the prices, and that the prices for each hour will be static
2. It is assumed, that a perfect price forecast of both the balancing market, up and down-regulation market is given.

Objective function:

$$\max_{P_t^{bid}} \sum_{t=1}^{24} \lambda_t^D P_t^{bid} + \lambda_t^{\downarrow}(P_t^{bid} - P_t^{real})^+ - \lambda_t^{\uparrow}(P_t^{real} - P_t^{bid})^+ \tag{1.1}$$

$$s.t.\ 0 \leq P_t^{bid} \leq 1, \forall t \in [1..24] \tag{1.2}$$

where P_t^{bid} is the wind farm's owner bid, λ_t^D is equal to the energy price for the day-ahead market, $\lambda_t^{\downarrow}$ is defined as the price for down-regulation and $\lambda_t^{\uparrow}$ for up-regulation. Finally, the $(...)^+$ indicates that we are only looking at the positive errors for both up- and down-regulation. This is in line with the example of a two-price market presented in *Integrating Renewables in Electricity Markets by Morales et al.* [1]. To account for the $(...)^+$ this problem has been slightly reworked to instead consist of.

$$\max_{P_t^{bid}} \sum_{t=1}^{24} \lambda_t^D P_t^{bid} + b_t^{\downarrow}\lambda_t^{\downarrow}(P_t^{bid} - P_t^{real}) - b_t^{\downarrow}\lambda_t^{\uparrow}(P_t^{real} - P_t^{bid}) \tag{1.3}$$

$$s.t.\ 0 \leq P_t^{bid} \leq 1 \qquad \forall t \in [1..24] \tag{1.4}$$

$$b_t^{\downarrow} = 0 \vee b_t^{\downarrow} = 1 \qquad \forall t \in [1..24] \tag{1.5}$$

$$P_t^{bid} \geq P_t^{real} + \epsilon - M(1 - b_t^{\downarrow}) \qquad \forall t \in [1..24] \tag{1.6}$$

$$P_t^{bid} \leq P_t^{real} + M b_t^{\downarrow} \qquad \forall t \in [1..24] \tag{1.7}$$

Where $b_t^{\downarrow}$ is a binary indicator, that shows whether down-regulation is necessary for this bid, ϵ is a very small step-value necessitated by Gurobipy, since it does not allow for absolute inequalities and $M = 1+\epsilon$, since 1 is the upper bound of P_t^{bid}.

[1] Link to textbook: `https://link.springer.com/book/10.1007/978-1-4614-9411-9#bibliographic-information`

2 Data

The primary methodology employed in data acquisition entailed a meticulous exploration and assimilation of diverse information sources, with an emphasis on comprehending their varied format. An assessment of the data sources revealed data originating from DMI Climate data, day-ahead prices for price areas for and surrounding Denmark (i.e. DK1, DK2 and areas of Sweden, Norway and Germany) and up- and down-regulation prices with no further specification of price area. Additionally, a time series of normalized power production, $\boldsymbol{P}_{real}$, for a windfarm in Roskilde was provided as the modeling target. For combining these data sources, a specialized `DataProcessor` class was developed to centralize preprocessing and merging of the data sources - the main effort being aligning UTC timestamps of the individual sources.

As the climate data had a fine-grained spatio-temporal resolution for a variety of weather attributes - e.g. minute-based timestamps along with long- and latitudinal information of the weather station - all weather observations were aggregated as the average value per hour and municipality. This choice of averaging hours implies no specific handling of summer- and wintertime changing hours. Lastly, the weather information was restricted to "wind power"-related attributes - i.e. `mean_wind_speed`, `mean_wind_dir`, `max_wind_speed_10min` and `mean_wind_speed_3sec`. With a specific focus on predicting wind farm power generation in Roskilde, the weather information was restricted to include only meteorological data from the municipality of Roskilde. Similarly, price information was restricted to the DK2 price zone, yielding a dataset with a total of unique 17.512 data points (i.e. hourly values). This systematic approach ensured a refined dataset aligned with project objectives.

2.1 Data analysis

Now, taking a closer look at the data reveals an interesting finding related to the scope of this project. In Figure 2.1, it is depicted that the historical actual power production is subject to irregularities which are defined by rapid increases or decreases. The initial hypothesis considered large maximum wind speeds as the underlying reason for irregularities, yet, there is no immediate relation between these two time series as the wind speed does not reach any lower or upper limits in terms of the functionality of the wind farm. Another case could be related to the maintenance of mechanical or electrical parts of the windmill and as a result, the wind farm would stop working for a specific time until it is fully recovered. Finally, the most suitable justification for this case is associated with the curtailment. Sometimes, the electricity grid to which the wind farm is connected may have limitations. When the grid reaches its maximum capacity, it may require the wind farm to curtail its power production to avoid overloading the grid. Hence, this can lead to a sudden reduction in power output.

As such, validly capturing these irregularities in the power production would from a machine learning point-of-view either require 1) access to a maintenance plan or 2) a historical overview of curtailment. As to our knowledge, both are quite extensive to acquire for which reason, we hypothesize that the soon-to-be-presented models will not be able to capture these irregularities.

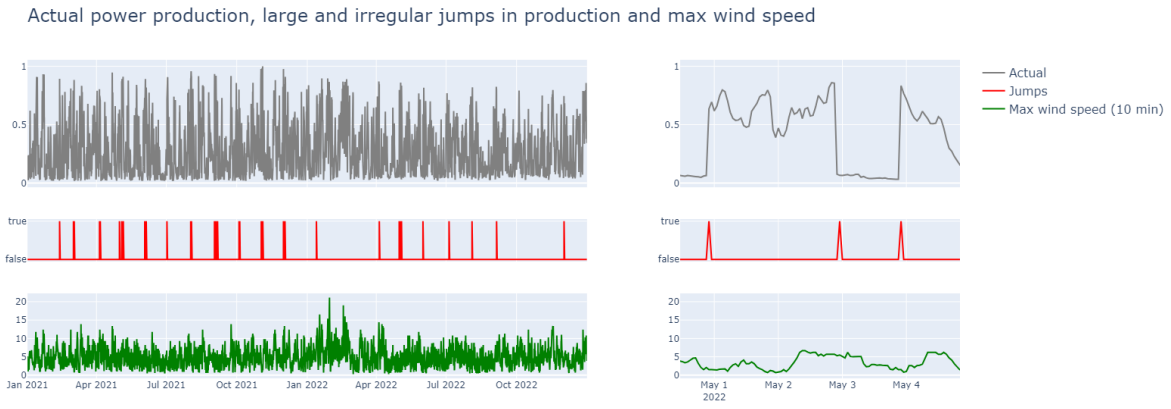

Figure 2.1: *Top:* the full power production time series (left) and a zoomed in version (right) that visualizes "jumps" in the power production series. *Mid:* indications of unusually large "jumps" in the power production series, defined by thresholding the difference between the series and its lagged version, i.e. $P_t - P_{t-1} \geq \tau$ with $\tau = 0.5$. *Bottom:* the maximum wind speed. There is no clear correspondence between high wind speeds and the "jumps" occurring in the power production series.

2.2 Constructing the target vector and data matrix

As previously mentioned, the normalized power production time series of length t was considered as the target attribute, i.e. the modeling goal, for which reason we define the target vector and data matrix as:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_t \end{bmatrix}^T, \qquad \boldsymbol{x}_i = \begin{bmatrix} 1 & \boldsymbol{x}_i^{time} & \boldsymbol{x}_i^{wind} & \boldsymbol{x}_i^{power} \end{bmatrix}^T$$

$$\boldsymbol{y} = \begin{bmatrix} P_{t=1}^{real} & P_{t=2}^{real} & \dots & P_{t=t}^{real} \end{bmatrix}^T$$

where the elements of $\boldsymbol{x}_i$ are defined by Eq. 2.1 - 2.2.

$$\boldsymbol{x}_i^{time} = \begin{bmatrix} \cos\left(\frac{2\pi \texttt{HourOfDay}}{24}\right) \\ \sin\left(\frac{2\pi \texttt{HourOfDay}}{24}\right) \end{bmatrix}^T \tag{2.1}$$

$$\boldsymbol{x}_i^{wind} = \begin{bmatrix} \texttt{max\_wind\_speed\_10min} \\ \texttt{max\_wind\_speed\_3sec} \\ \texttt{mean\_wind\_dir} \\ \texttt{mean\_wind\_speed} \\ \texttt{extreme\_wind} \\ \texttt{very\_extreme\_wind} \end{bmatrix}^T, \quad \boldsymbol{x}_i^{power} = \begin{bmatrix} \texttt{total\_power\_previous\_day} \\ \texttt{median\_power\_previous\_day} \\ \texttt{lowerCI\_power\_previous\_day} \\ \texttt{upperCI\_power\_previous\_day} \\ \texttt{total\_power\_previous\_week} \\ \texttt{median\_power\_previous\_week} \\ \texttt{lowerCI\_power\_previous\_week} \\ \texttt{upperCI\_power\_previous\_week} \end{bmatrix}^T \tag{2.2}$$

For the set of modified features mentioned in Eq. 2.2 perfect weather forecasting was assumed, for which reason the features of $\boldsymbol{x}_i^{wind}$ were constructed from the DMI data. Here, `extreme_wind` and `very_extreme_wind` were binary variables defined from being above the 75 and 95 quantiles, respectively, computed from all observations of `max_wind_speed_10min`. Similarly for $\boldsymbol{x}_i^{power}$, the `median`, `lowerCI` and `upperCI` were computed as the 50, 2.5 and 97.5 quantiles of the power production from either the `previous_day` or `predivous_week` dependent on the suffix of the attribute name. Lastly, as the hour of day is clearly periodic - hence, 02:00 is closer to 23:00 than to 12:00 - this is accounted for by $\boldsymbol{x}_i^{time}$ where hour of days are mapped to unique locations on the unit circle using a pair of sine/cosine features for each hour. The intuition is presented in Figure 2.2.

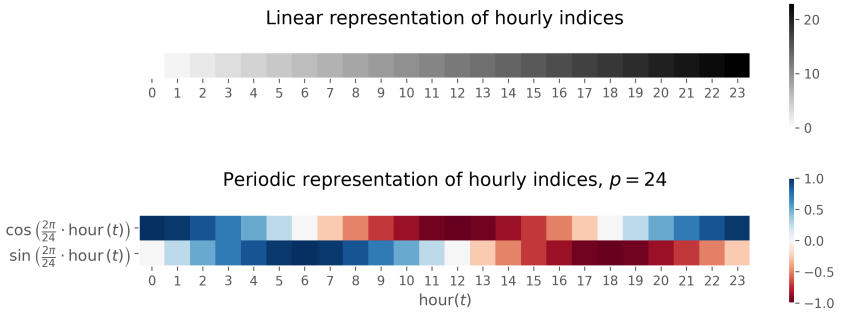

Figure 2.2: *Top:* Linear representation of hours of a day. The numerical difference between the first and last hour does not capture the underlying periodicity of a day. *Bottom:* Periodic representation of hours of a day. While preserving uniqueness of representations, the periodic representation rescales hours to $[-1; 1]$ and captures periodicity of a day.

All binary features - i.e. `extreme_wind` and `very_extreme_wind` - were one-hot-encoded resulting in a total of 18 features. Since the sine/cosine transformation of the daily hour maps to the interval $[-1; 1]$, all remaining continuous attributes were normalized to the interval $[-1; 1]$ using min-max normalization as presented in Equation 2.3 with $a = -1$ and $b = 1$:

$$a + \frac{(x - \min x)}{\max x - \min x} \cdot (b - a) \tag{2.3}$$

3 Methods

3.1 Linear Regression

As a baseline *linear regression* (LR) is considered. As the name indicates, linear regression considers a learnable linear mapping from an input vector, $\boldsymbol{x}$, to the continuous target attribute, y, i.e. $f_{\boldsymbol{\theta}}(\cdot) : \boldsymbol{x}^D \rightarrow \mathbb{R}$, where the term *"learnable linear"* relates to linearity of the model parameters, $\boldsymbol{\theta}$ that can be learned from data. In practice we define the function, $f(\cdot)$ with associated least-squares objective function as:

$$f_{\boldsymbol{\theta}} := \boldsymbol{\theta}^T \boldsymbol{x} = \hat{y} \tag{3.1}$$

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(y_i - f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_i)\right)^2 = \sum_{i=1}^{N} \left(y_i - \boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i\right)^2 \tag{3.2}$$

where the intercept, θ_0, has been absorbed in the parameter vector by adjusting $\boldsymbol{x}_i$ accordingly, i.e. $\tilde{\boldsymbol{x}}_i = \begin{bmatrix}1, & \boldsymbol{x}_i^T\end{bmatrix}^T$. Thus, the optimization task reduces to minimizing the objective function wrt. the decision variable, $\boldsymbol{\theta}$, i.e.:

$$\hat{\boldsymbol{\theta}}_{LR} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i\right)^2 \tag{3.3}$$

$$\hat{\boldsymbol{\theta}}_{LR} = \arg\min_{\boldsymbol{\theta}_{LR}} \left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right)^T \left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right) \tag{3.4}$$

$$\boldsymbol{X} = \begin{bmatrix}\tilde{\boldsymbol{x}}_0^T \\ \tilde{\boldsymbol{x}}_1^T \\ \vdots \\ \tilde{\boldsymbol{x}}_N^T\end{bmatrix} = \begin{bmatrix}1 & \boldsymbol{x}_0^T \\ 1 & \boldsymbol{x}_1^T \\ \vdots & \\ 1 & \boldsymbol{x}_N^T\end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix}y_0 \\ y_1 \\ \vdots \\ y_N\end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix}\theta_0 \\ \theta_1 \\ \vdots \\ \theta_{D+1}\end{bmatrix} \tag{3.5}$$

where 3.4 exploits the compact matrix form of the objective function.

There are now two approaches to solving the minimization objective; 1) using gradient descent or 2) from equating the gradient to zero and obtaining the closed form solution. Both requires an expression of the gradient which can be expressed as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = -2\boldsymbol{X}^T \boldsymbol{y} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\theta} \tag{3.6}$$

By equating the gradient to zero and reordering, the closed form solution immediately occurs as:

$$\hat{\boldsymbol{\theta}}_{LR} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{3.7}$$

Now, as the gradient is associated with the direction of maximum growth of the objective function, gradient descent relies on the idea of iteratively updating the parameters by stepping in the direction of the negative gradient, i.e.:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_{t+1}\left(\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\theta}_t - \boldsymbol{X}^T \boldsymbol{y}\right) \tag{3.8}$$

where the constant 2 (from Eq. 3.6) has been absorbed in the learning rate, α, that should decrease with time for ensuring convergence of the solution. As presented in Figure 3.1 using a subset of the available input features and a learning rate $\alpha = 0.001$, the gradient descent solution clearly converges slowly towards the closed form solution. Moreover, even if the solution has converged after approximately 500 steps, the algorithm does not stop until it reaches the total number of iterations 1200.

Though the gradient descent solution clearly is a valid approach to solving the minimization task for linear regression, the remaining part of this project report refers to the closed form solution as the baseline, since gradient descent in practice requires more computation time and could take long before converging.

3.2 Linear regression with polynomial features

As the name suggests, a linear regression model will in its simplest form be incapable of modeling non-linear trends. Yet, with its simplicity linear regression is rather intuitive as explanations of the underlying dynamics can directly be read off from the parameter coefficients.

A natural approach to extending linear regression for modeling non-linear trends is through the construction of a design matrix, $\boldsymbol{\Phi}$, that applies a transformation of the input features to a non-linear

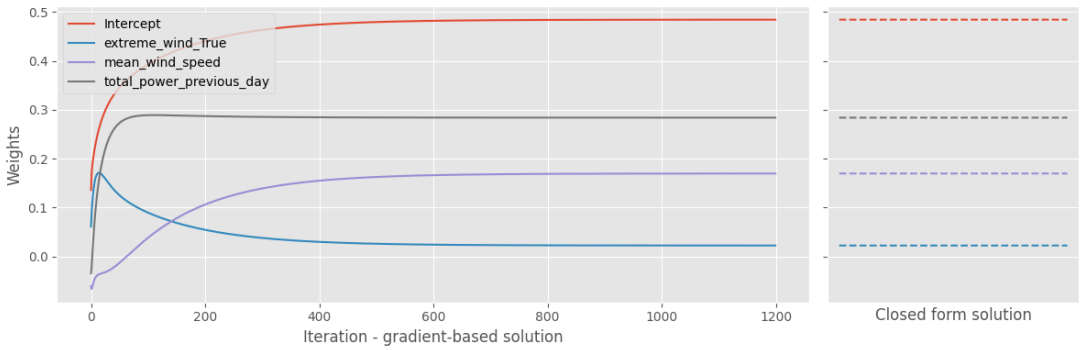

Figure 3.1: Trace of the parameter values, $\boldsymbol{\theta}_t$, for the gradient descent solution (Eq. 3.8) for 1200 iterations. The solution converges towards the closed form solution (Eq. 3.7).

representation. Many such design matrices exist, yet, this project report is restricted to considering polynomial design matrices of up to a specified order, p. The linear regression with capabilities of modeling non-linear trends is presented in Eq. 3.9:

$$\hat{\boldsymbol{\theta}}_{Polynomial} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{y} \tag{3.9}$$

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & \boldsymbol{x}_0^T & \left(\boldsymbol{x}_0^T\right)^2 & \cdots & \left(\boldsymbol{x}_0^T\right)^p \\ 1 & \boldsymbol{x}_1^T & \left(\boldsymbol{x}_1^T\right)^2 & \cdots & \left(\boldsymbol{x}_1^T\right)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \boldsymbol{x}_N^T & \left(\boldsymbol{x}_N^T\right)^2 & \cdots & \left(\boldsymbol{x}_N^T\right)^p \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \\ \theta_{p+1} \\ \vdots \\ \theta_{D\cdot p+1} \end{bmatrix}$$

We highlight that this model is still linear wrt. the parameters, $\boldsymbol{\theta}$, however able to capture polynomial trends.

The maximum order p was selected by training different models with multiple orders. The figure 4.3 shows that the best results were obtained using the first 2 orders, and as the order increases, the performance decreases.

3.3 LASSO and Ridge regression - regularization

When applying linear regression in real-world applications, two situations must be avoided, namely, underfitting and overfitting. While the first case relates to a too simple model, the second case is characterized by the model trying too hard to learn the available training data, resulting in a complex model that additionally learns the noise element of the training data, thereby not generalizing well to unseen data points although the training accuracy is high. Regularization-aware models such as LASSO and Ridge regression aim at combatting this undesired effect.

The key idea of regularization is to minimize the objective function described by Eq. 3.7 while ensuring that values of the learned parameters are as low as possible, thereby constraining the parameter vector to reduce the risk of learning behaviors that are too complex. For LASSO and Ridge regression the type of constraints introduced are in terms of the L1 and L2 norm, respectively.

For LASSO regression the penalty term forces some parameter values to go to zero, allowing some input variables to be effectively removed from the model. The regularization term, λ, is a hyperparameter that decides how much we want to penalize the flexibility of the model. As such, $\lambda = 0$ results in standard linear regression and implies that all features are considered relevant and it is equivalent to the linear regression model. Contrarily, increasing λ results in less non-zero parameters, thereby yielding a sparse model. The LASSO minimization objective is presented in Eq. 3.10 - one must note, that the L1 norm is non-differentiable at 0, meaning that subgradient methods must be exploited for estimating the parameter vector. As this was not presented in the curriculum, we consider this derivation out of scope for this

project.

$$\hat{\boldsymbol{\theta}}_{Lasso} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i\right)^2 + \lambda \sum_{i=1}^{M} |\theta_i| \tag{3.10}$$

The effect of introducing L1 regularization is considered in Figure 3.2. As overfitting is most apparent for data sets of lower size, only 2000 data samples are considered. On the left plot, the nature of the LASSO solution - namely, that more and more input variables are removed as λ increase - is apparent while the right plot visualizes the train and test RMSE for the same range of λ values. Considering the test RMSE, the optimal hyperparamter value is $\lambda = 0.00156$ which results in approx. half of the features having values close to 0.

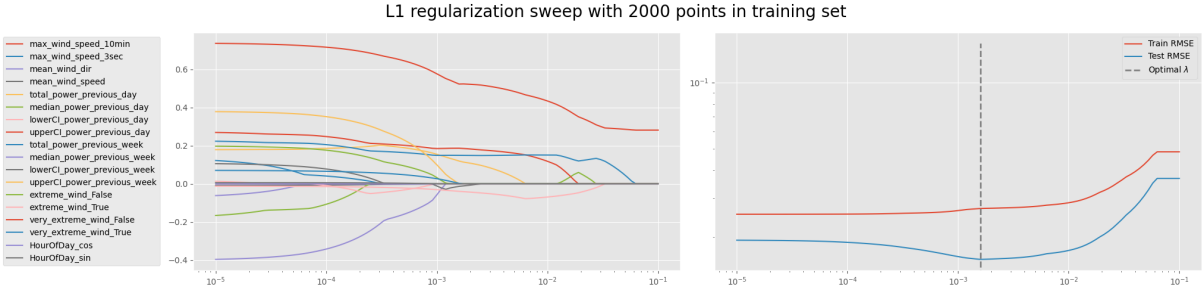

Figure 3.2: Parameter values and train/test errors (RMSE) as functions of the LASSO regularization parameter, λ, in the range $[10^{-5}, 10^{-1}]$. As the regularization value increases, sparse solutions are promoted (*left*) while for a certain range resulting in an optimal test RMSE.

For Ridge regression, the Euclidean norm, i.e. L2 norm, of the parameters is added as the constraint. As a direct consequence of this, the values of the coefficients tend asymptotically to 0, but do not reach this value. As such, the Ridge solution is not sparse. The minimization objective is given in Eq. 3.12.

$$J_{Ridge}\left(\boldsymbol{\theta}\right) = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{M} \theta_i^2 \tag{3.11}$$

$$\hat{\boldsymbol{\theta}}_{Ridge} = \arg\min_{\boldsymbol{\theta}} J_{Ridge}\left(\boldsymbol{\theta}\right) = \arg\min_{\boldsymbol{\theta}} \left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right)^T \left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right) + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta} \tag{3.12}$$

Following a similar approach as the derivation of the standard linear regression (Eq. 3.6 - 3.7), a closed form can be found. This is given in Eq. 3.13.

$$\hat{\boldsymbol{\theta}}_{Ridge} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda I\right)^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{3.13}$$

Figure 3.3 shows the importance of Ridge regularization. As for the LASSO case, a scenario with $|\boldsymbol{X}_{train}| = 2000$ consider, yet, now compared to a scenario exploiting the full training data. In the first scenario, the effect of the regularization term is stronger as the model is prone to overfitting when learning from limited amounts of data, whereas a larger data set will result in a model that generalizes better. For this visualization, the optimal regularization strengths were $\lambda = 17.38$ and $\lambda = 48.98$ with RMSEs of 0.01608 and 0.01563 for the first and second scenario, respectively. As expected, the optimal regularization strength and associated RMSE decrease with the increase of the training set size.

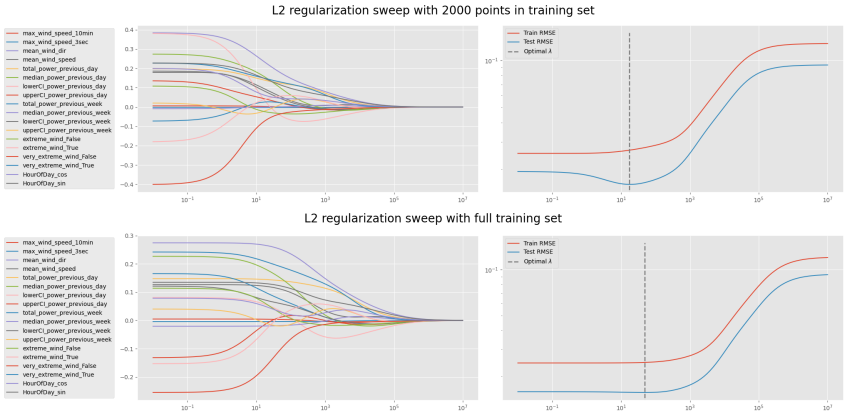

Figure 3.3: Parameter values and train/test errors (RMSE) as functions of the Ridge regularization parameter, λ. *Top:* with 2000 data points, the effect of Ridge regularization on the test error is clearly visible and thus counters overfitting. *Bottom:* using 80% of the data as the training set the regularization effect is less apparent.

To summarize, regularization increases the bias for the parameters, $\boldsymbol{\theta}$, while decreasing the variance, however, the optimal performance lies somewhere in between an unbiased and a variance-free model. This is formally known as the bias-variance trade-off. As such, hyperparameter tuning is required for properly introducing regularization in the modeling approach.

3.4 Kernel Ridge regression - exploiting local data patterns

While the previously mentioned models are *parametric models* - meaning that a parameter vector, $\boldsymbol{\theta}$, is learned and used for inference - Kernel Ridge regression is a *non-parametric* approach to linear modeling, that exploits local data patterns in the training set by learning a parameter vector, $\boldsymbol{\theta}$, for each inference data point through incorporation of a kernel function, $k(\cdot, x)$ that can be thought of as mapping distances between points. As such, the running time scales with the number of training- and test data points. The formulas considering Kernel Ridge regression are given in Eq. 3.14.

$$\boldsymbol{\theta}_i = \left(\boldsymbol{X}^T \boldsymbol{K}_i \boldsymbol{X} + \lambda I\right)^{-1} \boldsymbol{X}^T \boldsymbol{K}_i \boldsymbol{y} \tag{3.14}$$

$$\hat{y}_i = \boldsymbol{\theta}_i^T X$$

where $\boldsymbol{K}_i$ is the kernel matrix associated with the i'th inference data point such that

$$\boldsymbol{K}_i = \operatorname{diag} k\left(\boldsymbol{x}_i^*, \boldsymbol{X}\right) = \begin{bmatrix} k\left(\boldsymbol{x}_i^*, \boldsymbol{x}_1\right) & 0 & \cdots & 0 \\ 0 & k\left(\boldsymbol{x}_i^*, \boldsymbol{x}_2\right) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k\left(\boldsymbol{x}_i^*, \boldsymbol{x}_N\right) \end{bmatrix}$$

where $\boldsymbol{x}_i^*$ is the i'th test data point and N denotes the number of data points in the training set. Though many choices of kernel functions exists, this project restricts itself to the Gaussian kernel (Eq. 3.15) with hyperparameters κ and σ where $\kappa = 1$ is considered constant. A vectorized implementation of Kernel Ridge regression has been implemented in the associated code.

$$k\left(\boldsymbol{x}^*, \boldsymbol{x}_n\right) = \kappa \exp\left(-\frac{1}{2\sigma^2}\left(\boldsymbol{x}^* - \boldsymbol{x}_n\right)^T \left(\boldsymbol{x}^* - \boldsymbol{x}_n\right)\right) \tag{3.15}$$

3.5 K-means Ridge regression - learning local models

Unsupervised learning techniques such as K-means clustering are valid approaches to extracting patterns from unstructered information sources. As such, exploiting K-means for determining clusters in the data can be used for extracting local patterns, thus allowing for learning a set of local models. This is formalized in Eq. 3.16.

$$\hat{\boldsymbol{\theta}}_{K=c} = \left(\boldsymbol{X}_c^T \boldsymbol{X}_c + \lambda I\right)^{-1} \boldsymbol{X}_c^T \boldsymbol{y}_c, \qquad X_c = \{\texttt{KMeans}(x_i) = c\} \tag{3.16}$$

In other words, a ridge regression model is trained on the data contained in a cluster, c, for each cluster, $c \in \{1, 2, \cdots K\}$. This results in a K-set of models, $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_K\}$, that can be used in inference after having determined which of the K clusters the i'th data point, $\boldsymbol{x}_i^*$, belongs to. Notice, that the clusters are constructed from individual hours and not as stereotypical production days.

4 Evaluation approach

4.1 Two-layer cross validation

For evaluating model performance, a two-layer cross validation (CV) approach was considered. The main reason for this was to be able to 1) do hyperparameter optimization on regularization strengths, λ, polynomial order, p, kernel lengthscale, σ, and number of K-means clusters, K, while 2) obtaining the generalization error for all of the models considered. To do so two-layer rather than a repeated one-layer CV approach is a strict necessity as the repeated one-layer CV would introduce a bias in the generalization error by evaluating hyperparameters on the same validation split as is validated on in the generalization loop. This is accounted for in the setup of the two-layer CV, as is visualized in Figure 4.1. For all inner folds, a range of values for each hyperparameter was evaluated. The final ranges were determined in an iterative manner for being as precise as possible.

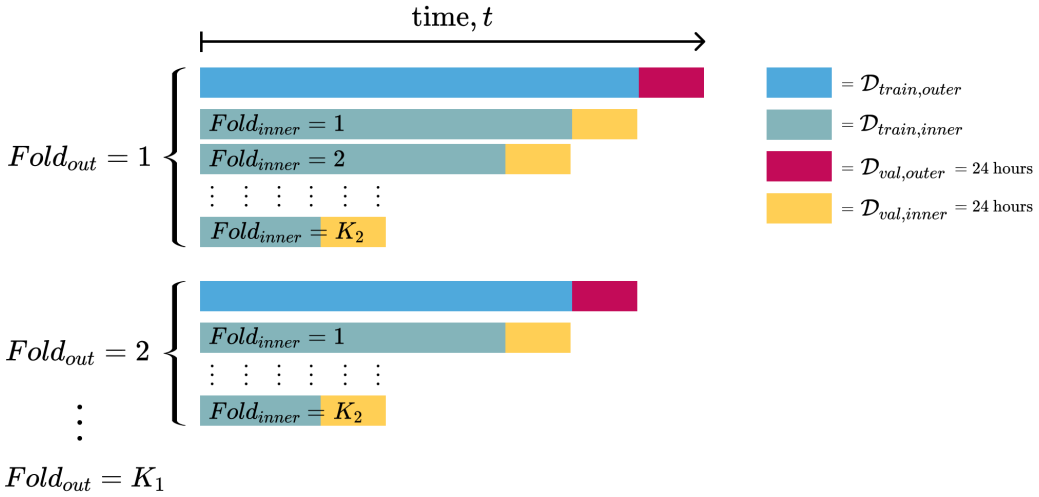

Figure 4.1: A visualization of the two-layer cross validation (CV) approach applied throughout the project. For each outer fold K_2 inner loops trains and evaluates a model for each hyperparameter value after which the best (on average) hyperparameter value is used for training the model in the associated outer fold. As such, using two-layer CV gives us the generalization error while *also* running hyperparameter optimization. For each outer fold, the validation set is shifted 24 hours in time. Throughout this project validation sets of sizes $|\mathcal{D}_{val,inner}| = |\mathcal{D}_{val,outer}| = 24$ hours is used while $|\mathcal{D}_{train,outer}|$ and $|\mathcal{D}_{train,inner}|$ is dependent on the value of $Fold_{out}$ and $Fold_{inner}$.

For maximum resemblance to the real-world modeling task at hand - namely, forecasting wind power prediction 24 hours in advance for bidding at the day-ahead market - all models were evaluated using the same specific outline of the two-layer CV, namely a *leave-one-day-out* approach. Thus, validation sets were selected as 24 hour intervals with corresponding training sets consisting of all previous (in terms of time) data points. Though this choice strongly relates to the overall modeling task it implies an increased computational cost for which reason the two-layer CV was restricted to $K_1 = K_2 = 14$ days. This will in practice result in a generalization error that does not capture longer periodic trends such as the yearly pattern in wind speeds, however, it was a necessity for being able to run all model comparisons in time. It will furthermore impact the suitable metrics.

4.2 Metrics

All models were optimized based on the Root Mean Squared Error (RMSE) score. Additionally, the Mean Absolute Error (MAE) and the R^2 score were used for comparing model performances. However, due to the evaluation outline considering 24 hour validation sets, the R^2 score will almost surely perform worse than random, resulting in negative R^2 scores. As for why this is the case, consider the definition

of the R^2 score:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{4.1}$$

$$SS_{res} = \sum_i \left(y_i - f\left(\boldsymbol{x_i}\right)\right)^2 \tag{4.2}$$

$$SS_{tot} = \sum_i \left(y_i - \mathbb{E}\left[\boldsymbol{y}\right]\right)^2 \tag{4.3}$$

As is seen, the R^2 score can be considered as a comparison between the model predictions and a mean-predicting baseline. Then, as the validation set consists of a 24 hour time series of wind power production (which is most likely not stationary) and since the residual error (MSE or SS_{res}) is unbounded, the model, $f(\cdot)$ will in general perform worse than random, possibly yielding unbounded negative R^2 scores.

4.3 Feature selection

The optimal set of features is determined by making use of the `sklearn`-function, `SelectKBest` which selects the K-set of most informative features according to the K highest scoring configurations based on a specified score function which was selected as the `mutual_info_regression`-score. This approach was preferred to forward- or backward feature selection due to the high computational cost of evaluation using 2-layer CV.

Figure 4.2 (*left*) shows that there are no clear distinction in model performance when varying the fraction of the training set used (where the full data of 17.517 data points is used), however, using 10% of the data seems to in general be slightly lower in average sense. Thus, $17.512 \cdot 0.1 \approx 2000$ data points were used for further model fitting (including Figure 4.2 (*right*)). In general, the RMSE decreases as the number of features increases, yet, the lowest score is reached when 14 features are considered (in the example fold). After that it increases slightly until the maximum number of features (18) is used. Using majority voting on the outer folds, the optimal number of features was determined to be 15. Furthermore, it appeared that the three features `HourOfDay_cos`, `HourOfDay_sin` and `very_extreme_wind_True` did not improve the model performance significantly for which reason they are not considered in future model fitting.

Table 2: Optimal number of features found applying `SelectKBest` on the outer folds.

Fold	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Majority
$N_{features}$	13	13	13	13	15	15	15	15	15	15	15	15	13	14	**15**

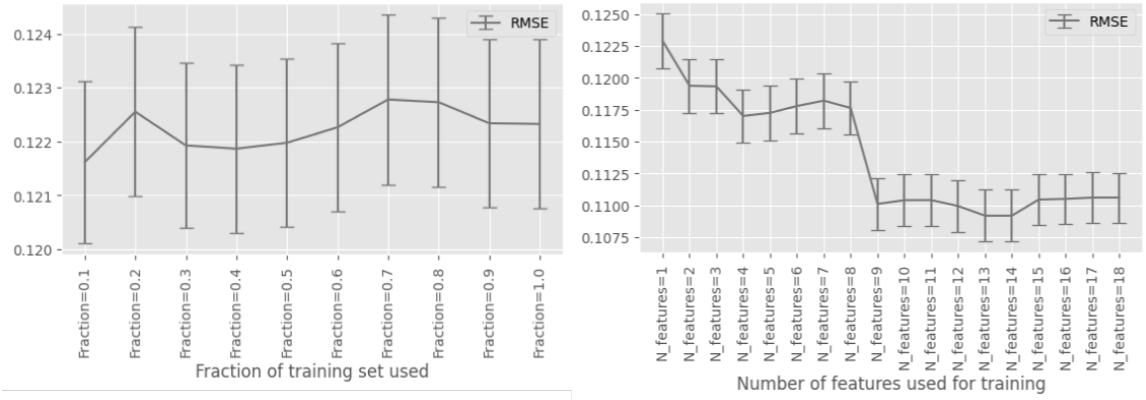

Figure 4.2: Visual comparison of the RMSE as a function of the training set size (*left*) and the number of features determined using `SelectKBest` (*right*) for an example fold of the CV.

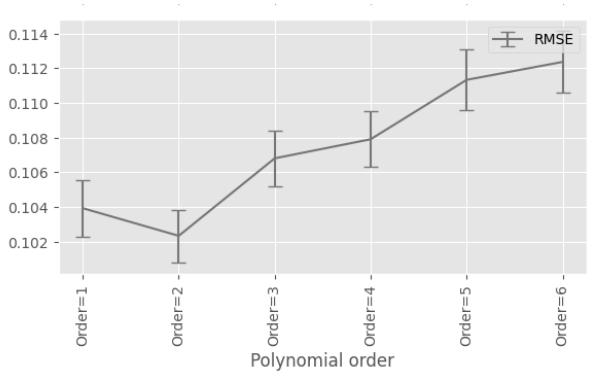

Figure 4.3: Visual comparison of the RMSE as a function of the polynomial order. The performance gains slightly from exploiting second-order polynomial terms, however, too complex polynomial data representations decreases test performance.

5 Results & Discussion

5.1 Model comparison

Table 3 summarizes the model comparison for all the outer folds. It includes the fold-wise RMSE along with the optimized hyperparameter for each considered model found from majority voting on the inner CV loop. As is seen, the optimal hyperparameter value varies for each outer fold, highlighting the benefit of exploiting a two-layer CV evaluation approach. Futhermore, the RMSEs seem to vary by quite a margin depending on which of the outer fold that is evaluated.

Table 3: Benchmark of different modeling approaches showing the RMSE for $K_1 = 14$ outer CV folds along with the optimal hyperparameter value - i.e. regularization strength, λ, polynomial order, p, kernel lengthscale, σ, or number of clusters, K - as defined from majority voting on the $K_2 = 14$ inner CV folds.

	Baseline	**Ridge**		**LASSO**		**Polynomial**		**Kernel Ridge**		**Kmeans**	
$Fold_{out}$	RMSE	RMSE	λ	RMSE	λ	RMSE	p	RMSE	σ	RMSE	K
1	0.222	0.232	90.0	0.216	0.006	0.221	2	0.222	3.1	0.233	4
2	0.135	0.116	70.0	0.123	0.005	0.107	2	0.131	3.5	0.122	4
3	0.204	0.104	70.0	0.154	0.002	0.118	2	0.189	3.5	0.090	4
4	0.071	0.122	70.0	0.076	0.002	0.119	2	0.075	3.1	0.188	4
5	0.161	0.083	70.0	0.127	0.002	0.082	2	0.155	3.1	0.064	4
6	0.098	0.126	70.0	0.120	0.005	0.118	2	0.095	2.7	0.163	2
7	0.078	0.065	70.0	0.066	0.005	0.072	2	0.079	2.7	0.058	2
8	0.089	0.070	70.0	0.082	0.005	0.070	1	0.089	2.7	0.067	2
9	0.096	0.076	70.0	0.074	0.005	0.076	1	0.093	2.7	0.061	2
10	0.088	0.087	70.0	0.094	0.005	0.087	1	0.097	2.7	0.130	2
11	0.162	0.169	70.0	0.175	0.005	0.169	1	0.160	2.7	0.154	2
12	0.233	0.218	70.0	0.222	0.005	0.218	1	0.274	2.7	0.233	2
13	0.101	0.105	90.0	0.112	0.005	0.105	1	0.103	2.7	0.106	2
14	0.054	0.079	90.0	0.057	0.006	0.079	1	0.055	2.7	0.081	2

Table 4 presents the generalization errors of each model in terms of the RMSE, MAE and R^2-score values reported with the empirical standard error of the mean (SEM). As is seen, all models except Kernel Ridge regression on average improves on the standard linear regression baseline with LASSO regression yielding the better results, however, by accounting for uncertainty (through the SEM), none of the models perform significantly better than the others. The fact that Kernel Ridge regression is on average worse than the baseline could be related to the definition of "locality" for high-dimensional data (i.e. 18 input features) that might not be properly captured by the Gaussian kernel. As such other kernel functions or a finer optimization grid for σ might yield different results.

Table 4: Generalization error obtained from averaging across predictions for all $K_1 = 14$ outer folds. The provided metrics are RMSE, MAE and the R^2-score along with the empirical standard error of the mean, i.e. $\frac{\sigma}{K_1}$.

E_{gen}	Baseline	Ridge	LASSO	Polynomial	Kernel Ridge	Kmeans
RMSE	0.128 ± 0.016	0.118 ± 0.014	0.117 ± 0.014	0.121 ± 0.014	0.131 ± 0.015	0.125 ± 0.016
MAE	0.111 ± 0.015	0.106 ± 0.008	0.106 ± 0.009	0.105 ± 0.006	0.106 ± 0.006	0.106 ± 0.005
R^2	-1.225 ± 0.561	-1.103 ± 0.373	-1.034 ± 0.288	-1.018 ± 0.247	-1.071 ± 0.222	-1.160 ± 0.243

5.2 Qualitative analysis

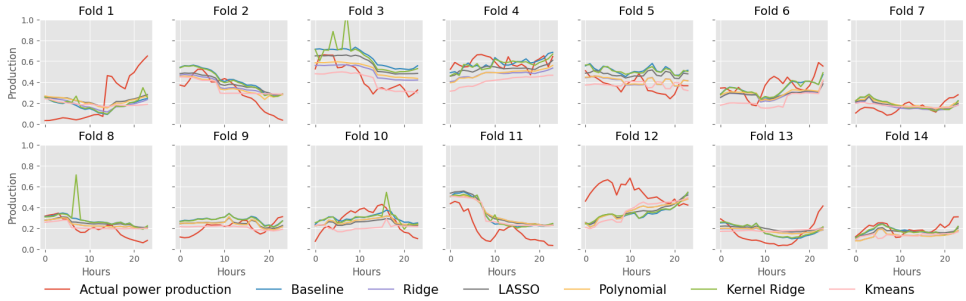

Figure 5.1: Visual comparison of the predicted power production for all considered models against the actual power production for each of the 24-hour intervals that the outer CV folds consist of.

Taking a further look at the predictions made on each outer fold, Figure 5.1 in general reveals that all models tend to capture the underlying trend of the actual power production. There are, however, certain exceptions. Fold 1 visualizes that none of the models are able of capturing the irregularities that occasionally occurs in the actual power production series (see Section 2), as expected. Furthermore, the model performances are distinguishable for Fold 3, 4 and 5 for which the actual power production seems to be relatively high compared to the remaining folds. This suggests, that the absolute production value of a day might influence what is learnt by the models and could be a motivation for leveraging more advanced unsupervised learning methods for extracting additional information e.g. by clustering stereotypical production days.

5.3 Revenue Calculation

By running the predicted power output through the optimization model, the following mean sum output with standard deviation is generated for each of the $K_1 = 14$ folds.

	Baseline	Ridge	LASSO	Kernel Ridge	Kmeans
$\hat{\mu}_{Rev} \pm \hat{\sigma}$	558.264 ± 148.5	545.639 ± 145.266	530.094 ± 148.181	558.171 ± 149.219	510.945 ± 150.715

Taking a look at these results show, that like in power prediction, the resulting optimal bids do not have a big variation in between.
Of course a certain variation is present wrt. which part of the data-set is used to test the model wrt. revenue calculation as is seen by the standard deviation.

6 Model 2: directly learning the optimal bidding strategy

While the theoretical basis remains the same, the case of directly learning a mapping, $f(\cdot)$, from a set of input features, $\boldsymbol{x}$, to the optimal bidding strategy, P^{bid}, requires a redefinition of the target vector and possibly also the data matrix. Thus, we redefine:

$$\boldsymbol{y} = \begin{bmatrix} P^{bid}_{t=1} & P^{bid}_{t=2} & \dots & P^{bid}_{t=t} \end{bmatrix}^T$$

To train model 2, we use the best model from the previous revenue calculation and attempt to re-train it using the power bid as a label instead.

The revenue calculation shows, that the highest revenue is estimated to be found when using the simple linear regression (of course with considerable variation).

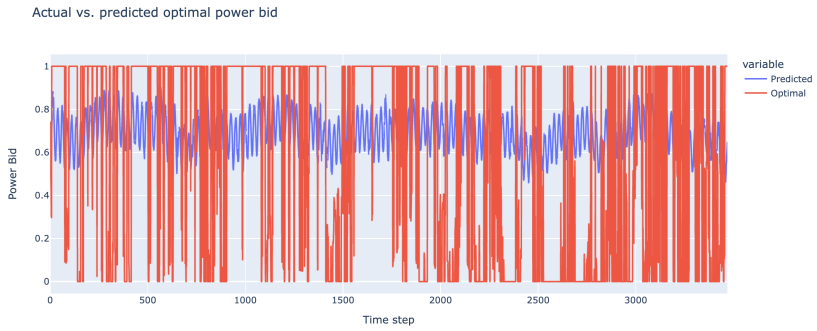

Figure 6.1: (The linear estimator cannot account for the sudden jumps in optimal power bid)

From figure 6.1 it is quite obvious that the linear estimator **cannot** account for the big jumps in optimal power-bid, Which also explains the large $\boldsymbol{RMSE}$. A more sophisticated model is thus necessary if an accurate bid is wanted.

$$\boldsymbol{RMSE} = 0.4664$$

7 General discussion

7.1 Predicted v. Actual Power Production

It is worth mentioning that the optimizer calculates and optimizes wrt. to the **predicted** power production instead of the **actual** power production. This necessarily means that for hours where over- and under-bidding is not gainful (see sec. 7.2), the model may have a higher **expected** revenue than others due to inaccuracies and not due to higher precision.

7.2 Three bidding scenarios

The optimizer assumes perfect information about day-ahead prices as well as both regulation prices.
If a price taker has access to these prices, there are really only 3 different solutions, which are given by the relation between these prices.

$$(\lambda^D > \lambda^{\uparrow}) \vee ([\lambda^D, \lambda^{\downarrow}] < -\lambda^{\uparrow}) \rightarrow P_{bid} = 1 \tag{7.1}$$

Here a bidder can gain revenue both from P_{actual} as well as any outstanding power deficit.

$$[\lambda^{\uparrow}, \lambda^D] < \lambda^{\downarrow} \rightarrow P_{bid} = 0 \tag{7.2}$$

Here a bidder has most to gain by selling at the up-regulation price, and thus bidding 0 in the day-ahead market.
Any other case can in this model be optimized via just bidding the actual power-prediction

$$P_{bid} = P_{actual} \tag{7.3}$$

8 Conclusion

For models at this level of complexity, it seems more gainful to guess the actual power-production for the future and calculate an accurate power-bid using this. Moreover the cases of inaccurate prediction of power-generation have been qualitatively analysed.
The theory behind different mathematical prediction models were explained and instances of them fitted to the data. It was shown that a gainful hyper-parameter adjustment could be found via 2-layer cross-validation for several of the models tested, and that very little variation was found in any of the models. wrt. accuracy after this adjustment.