Aalborg University

Master's Thesis

Mathematics-Economics

Analysis of Day-Ahead Prices and Influencing Factors in the Nordic Power Market

Anders Linnemann Nielsen

October 6, 2023

This document is typeset with LaTeX. Programming is performed in `Python`.

AALBORG UNIVERSITY
STUDENT REPORT

School of Engineering and Science
Mathematics-Economics
Skjernvej 4A
9220 Aalborg Øst
http://math.aau.dk

Title

Analysis of Day-Ahead Prices and Influencing Factors in the Nordic Power Market

Theme

Master's thesis

Project Period

02/06/2023 - 06/10/2023

Author

Anders Linnemann Nielsen

Supervisor

Esben Høg

Page Numbers

37

Abstract

This thesis analyses the performance of machine learning methods including Lasso, neural networks and random forests, on forecasting day-ahead prices in the Nordic power market. To obtain this, two benchmark models is presented for comparison to the machine learning methods, namely the ARIMA and the naïve model. The models compare against each other on 15 time series of day-ahead prices from the Nordic power market ranging from 2015 throughout September 2023.
Furthermore, the machine learning models is fed with wind and solar power production forecasts to improve on the effectiveness. The models is empirically compared by using MAE, MSE, MAPE and RMSE as performance metrics. Some of the main results is that neural networks perform very well, when it is tuned to not overfit the data. Furthermore Lasso gives stable results without extensive supervision.

Preface

This report is a 10th semester project in Mathematics-Economics and thus a master's thesis. It is written at the School of Engineering and Science at Aalborg University. The title of the project is *Analysis of Day-Ahead Prices and Influencing Factors in the Nordic Power Market*

The project is composed of numbered chapters with corresponding sections and subsections. Citations and external references are made using the Harvard system such that a source is refered to like so: [Last name, Year]. All programming is performed using Python. I would like to extend my appreciation to my supervisor Esben Høg.

Signature

Anders Linnemann Nielsen

Contents

1 Introduction

In the early 1990s the Nordic countries liberalised the power markets and introduced free competition [NordPool, 2023d]. The Nordic power market consists of four countries: Denmark, Finland, Norway and Sweden, which are linked through a common electricity grid. Day-ahead prices refer to the prices at which electricity is traded in advance for the following day's delivery [NordPool, 2023a]. Forecasting of the day-ahead prices are important for participants in the electricity market, including power generation companies, traders and consumers, as it helps them to make decisions regarding their operations, investments and consumption patterns [Kristiansen, 2014].

The day-ahead prices in the Nordic power market are determined by supply and demand. The supply and demand of electricity is influenced by factors such as weather conditions, fuel prices, generation capacity, transmission constraints and the bidding strategies of market participants. Thus the day-ahead prices are influenced by the same factors. Traditional forecasting models, such as autoregressive integrated moving average (ARIMA) models have been used to predict day-ahead prices in the Nordic power market. However, these models have limitations in capturing the complex and dynamic nature of the market, which can lead to inaccurate forecasts [Cifter, 2013].

In recent years, machine learning techniques, such as neural networks, decision trees, and random forest, have gained popularity in electricity price forecasting due to their ability to handle non-linear relationships and high-dimensional data. Although some studies have explored the use of machine learning techniques in the Nordic power market, there still seems to be a need for more research to evaluate the performance of these methods.

Therefore, the primary objective of this master's thesis is to investigate the application of machine learning techniques for forecasting the day-ahead prices in the Nordic power market. The thesis will explore the performance of different machine learning algorithms and compare them against each other. The study will also examine the impact from different variables, such as weather data, generation and transmission data, on the forecasting accuracy of the models.

The results of this study is significant for various participants in the electricity market, including investors, energy traders and researchers. By accurately forecasting the day-ahead prices, we can gain valuable insights into future electricity prices. These insights can help market participants optimize their operations, reduce costs and minimize risks. Ultimately, such insights might push for a transition towards a more sustainable and efficient energy system.

In the next section, the problem statement to be examined in this thesis will be formulated. The remainder of this thesis is structured as follows. Chapter 2 provides an introduction to the Nordic electricity market. Chapter 3 describes the theory and methodology used in the study. Chapter 4 presents the results and analysis of the forecasting models. Finally, chapter 5 and 6 concludes the study and discusses the findings and limitations.

1.1 Problem Statement

The main goal of this master thesis is to analyse the performance of machine learning methods, including Lasso, Neural Networks and random forest models against benchmark models, such as ARIMA and naïve models as suggested by Ludwig et al. [2015]. ARIMA models are commonly used in electricity price forecasting and thus make a good baseline for analysing the effectiveness of the machine learning models.

By comparing the models against benchmark models, I wish to find the advantages and disadvantages of each method. In the framework of the Nordic power market, this thesis will contribute to a deeper understanding of the strengths and weaknesses of machine learning.

Furthermore, this analysis will consider the impact of relevant variables on the performance of the models. Variables such as weather data and generation mix have a significant role in the determining of the day-ahead prices. By incorporating these variables in the models, I can evaluate their influence on the predictive performance and determine which variables contribute effectively to the forecasting process. Based on this the following research questions are presented.

1.1.1 Research Questions

How do different machine learning techniques, such as Lasso, neural networks, and random forests, compare in terms of forecasting accuracy for the day-ahead prices in the Nordic power market? Also how do they compare against benchmark models such as ARIMA models?

What are the respective strengths and weaknesses of the machine learning methods and benchmark models, when applied to forecasting in the Nordic power market?

What is the impact of including weather data and generation mix data on the effectiveness of machine learning models?

The Nordic Electricity Market 2

In order to be able to forecast day-ahead prices, we must first of all understand the nordic electricity market. Hence this chapter explore some of the dynamics in the nordic electricity market. This chapter also presents the data provider for this thesis, namely Nordpool, which is one of the biggest power exchanges in Europe.

The power markets is a crucial part of the energy sector, as they make it possible for producers and consumers to trade electricity at market determined prices. By balancing the supply and demand for electricity, the power market ensures that the electricity system is reliable and efficient.

As mentioned in the introduction, the European power markets have been subject to major changes in the past two decades. The liberalisation and integration of the power markets has led to increased competition and thus increased liquidity. One of the most significant markets is the short-term power market, which consists of the intraday and day-ahead markets. These short-term markets enables market participants to balance supply and demand in real-time, as they are able to adjust their production or consumption plans based on prices and forecasts.

2.1 Nord Pool

Nord Pool is the leading power exchange in Europe, and offers access to power trading in markets across 16 European countries. Currently 360 market participants distributed on 20 countries trade the markets from Nord Pool. Nord Pool offers two types of power markets: the intraday market and the day-ahead market [NordPool, 2023b].

The day-ahead market is where electricity is traded for the next 24 hours. Market participants submit bids to buy or sell the following day. At 10:00 CET every day, data on the network is published. This contains transmission capacities in the power grid. Market participants have until 12:00 to submit bids for the auction on electricity delivery for the next day. They are able to bid on hourly power from hour 1 to hour 24. Other power exchanges offers products of smaller intervals, for instance Epex offers 30-min day-ahead contracts in Great Britain. After gate closure at 12:00, a validation of the bids are done. Afterwards calculation of the market price is done. The calculation and validation is commonly done around 12:45, where the market price is published [NordPool, 2023a].

In the intraday market, electricity is traded for delivery on the same day. The intraday market is used to balance the system in real-time, so that supply and demand are constantly matched. This allow participants to adjust their positions throughout the day after the day-ahead market closes. With the increase of renewable energy being produced, there also follows an increased interest in

trading intraday due to the unpredictability of renewable energy sources [Winther, 2020]. In contrast to the auction format of the day-ahead market, intraday is a continous market with trading going on constantly, 365 days a year, up until one hour before delivery. The intraday markets of Nord Pool is devided into 15-minute, 30-minute and 60-minute products, that enable the market participants to balance their positions in greater detail [NordPool, 2023c].

Both the day-ahead and the intraday market are crucial for the European power markets. They provide market participants with an opportunity to hedge their positions, manage risks, and optimize their operations. Additionally, they help ensure a stable supply of electricity, which is quite important in our day and time.

2.2 Data

This section examine and make initial analysis of the data provided by Nordpool and ENTSO-E.

Day-Ahead data

Nord Pool is providing data on the day-ahead prices for the Nordic power market. Section 4.1 provides a more detailed elaboration on the process of obtaining the data. The day-ahead prices are quite different from stock prices or similar financial products, as the prices comes in bulks of 24 at a time. In this section, we take a look at day-ahead prices in the nordic power market and how it is to be interpreted. Nord Pool has provided day-ahead prices for seven countries, namely: Denmark, Estonia, Finland, Latvia, Lithuania, Norway and Sweden. Denmark is divided into two areas, DK1 and DK2. Sweden has four areas and Norway is divided into six areas although only five are used in the project, as NO3 was split into Trondheim and Kristiansund in 2022.

Figure 2.1 show a sample of the day-ahead prices provided by Nord Pool. It shows three series of day-ahead prices, for 3 different hours. The data is from DK2 in the period from 2015 through September 2023. Even though it is an example, the other series from different areas and at different hours are quite similar. The prices are volatile and especially the period after 2021 reflects an unstable market.

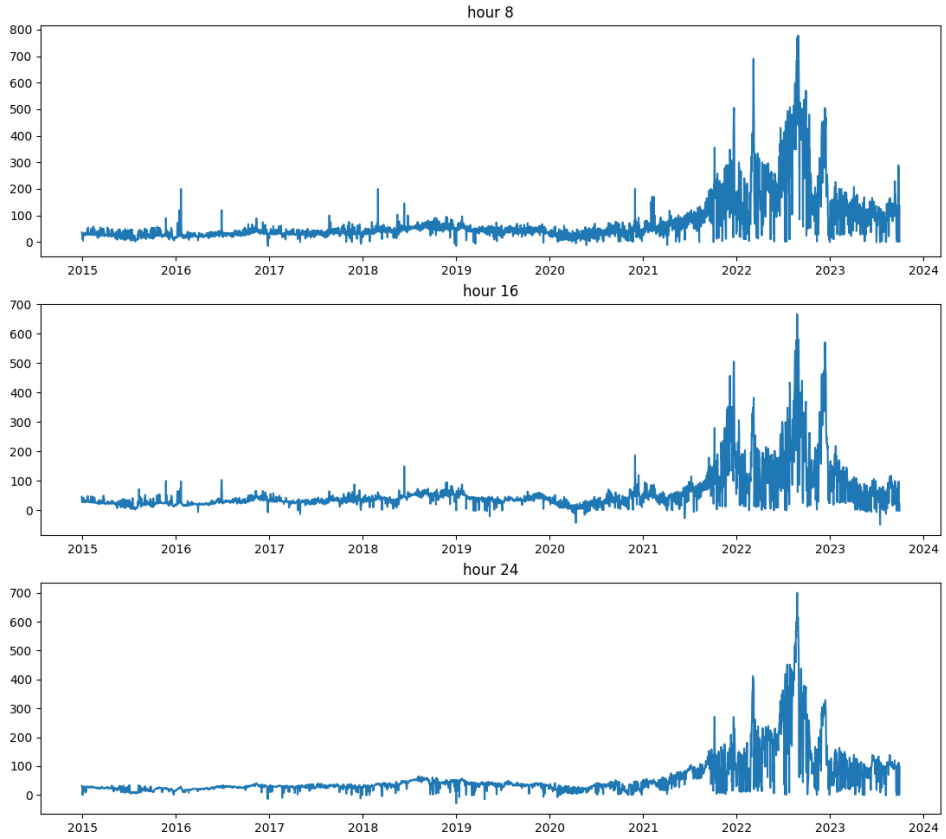

Figure 2.1: DK2 prices for hour 8, 16 and 24 from 2015 through september 2023.

Notice that the prices in hour 8 (the first picture in figure 2.1) in general seem to be higher than hour 16, which again is higher than hour 24. The high volatility from 2021 seems to have a peak mid 2022 and to be declining through 2023. To better understand the behavior of the day-ahead price, we can also look at the average hourly price for different days of the week and for different months of the year. Figure 2.2 shows the average hourly day-ahead price for the days of the week in DK2, and Figure 2.3 shows the average hourly price for the months of the year.

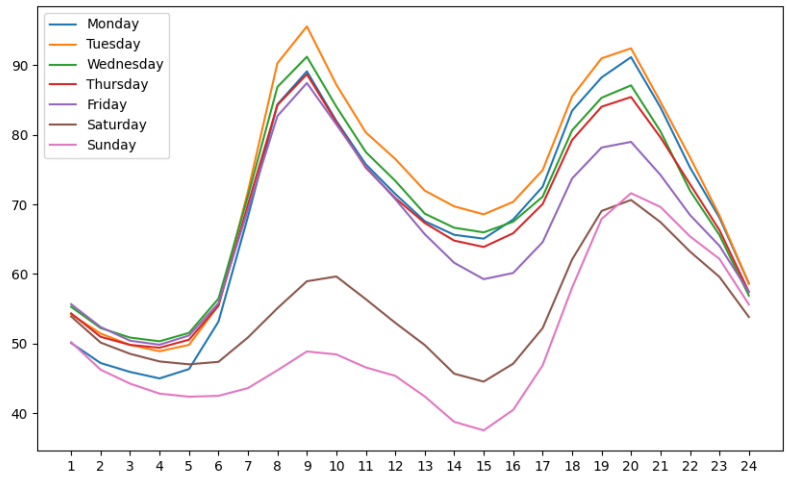

Figure 2.2: Average hourly day-ahead price for the days of the week

It is clear from figure 2.2 that the weekends have lower prices than weekdays and Sunday has the lowest prices. This is to be expected, as the demand of electricity should fall when people are of work and industries take time off.

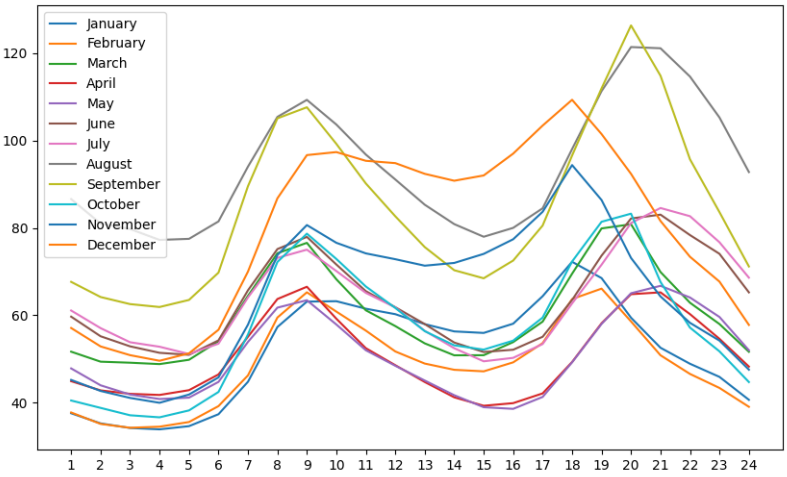

Figure 2.3: average hourly day-ahead price for the months of the year

In figure 2.3 we see that the summer months exhibit lower prices than the winter months. The lowest prices is April and May. Interestingly, the months with the highest prices are August and September. A theory for this could be that the highest ever general prices of electricity is around August and September 2022 and thus the average of these months is pushed up. This is seen in figure 2.1, where the biggest spike for hour 8 is on August 30, 2022. In section 3.1, variance stabilizing

transformations is investigated. These transformations might help to make the variance constant, and hopefully adjust for the irregularities the spikes of the prices make.

Furthermore, a daily pattern is also clear from the two plots above. The prices have a high around hour 9 and again around hour 20. There is a low at night where people surely are sleeping and again a higher low at around 15:00-16:00. O) ["EN].

Wind and solar forecasts

As explanatory variables for the machine learning models in this thesis, wind and solar forecasts from the ENTSO-E transparancy platform will be used. ENTSO-E is short for "the European Network of Transmission System Operators for Electricity", therefore, as the name states, it is the association for cooperation between the European transmission system operators (TSO) [ENTSO-E]. On the ENTSO-E transparancy platform, there is a wide range of different data on the European electricity system, including generation mix, power consumption and different forecasts. They provide a day-ahead forecast for the production of wind and solar power, which seem to fit this project well. The data is available from 2015 and onwards, in all nordic areas.

The first figure is a sample the power production from solar panels i DK1 from 2020 until September 2023

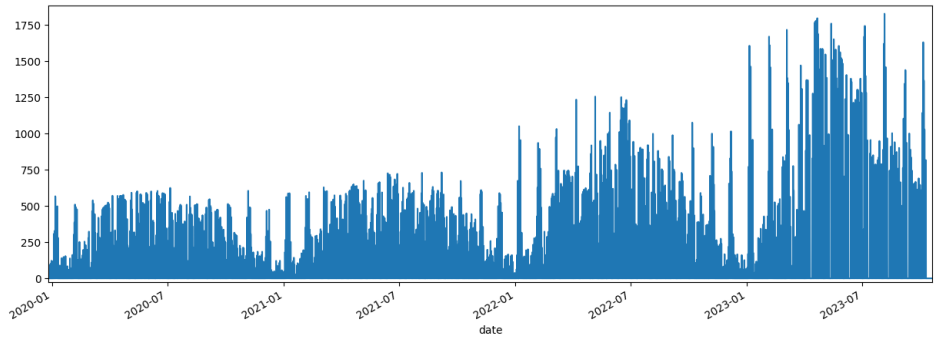

Figure 2.4: Solar power production in DK1 in MW.

In figure 2.4 a sample of the solar forecasts are displayed. It is observed that the production of solar power display strong seasonal trends which might fit well with the seasonality detected in the day-ahead prices in section 2.2. Furthermore, there is a general uptrend, which might be a result of more and more solar parks being built.

The following figure shows the power produced from offshore wind in DK1.

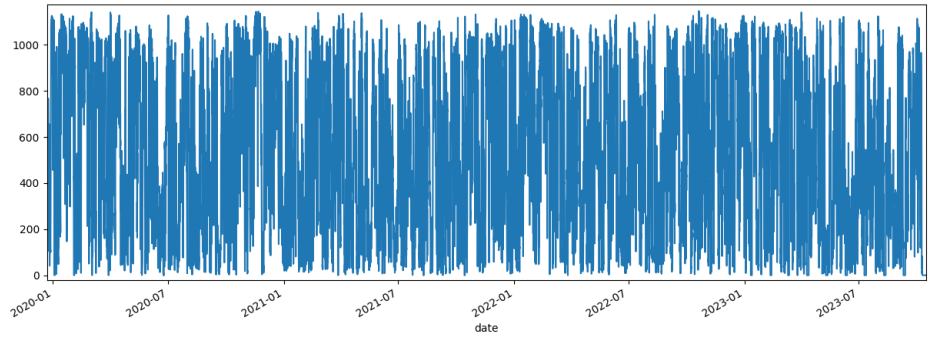

Figure 2.5: Offshore wind production in DK1 in MW.

The power production forecasts from figure 2.5 do not seem to have any systematic behaviour and almost resemble white noise. Although the data is unstructured, it might still prove to have important impacts on the day-ahead prices in the data analysis.

The next figure is the power produced from onshore wind in DK1.

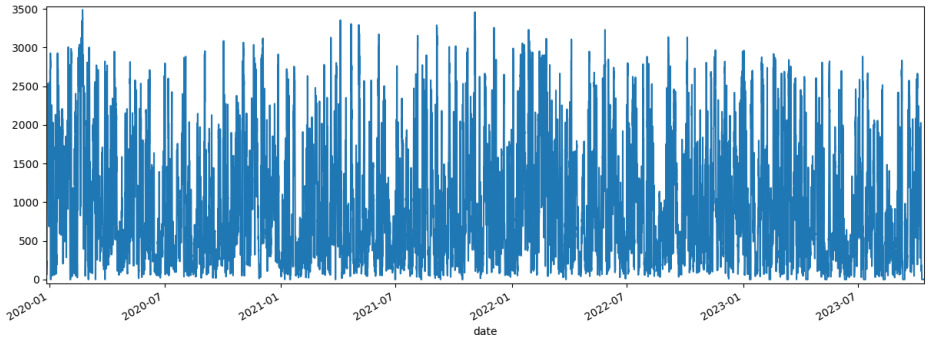

Figure 2.6: Onshore wind production in DK1 in MW.

As for the forecast for the offshore wind production, there is not much structure to the onshore wind production forcasts in figure 2.6. However, there is quite a difference in the power produced. The offshore production is peaking constantly around 1000 MW, and the onshore is frequently above 3000 MW, and also showing less constant variance.

Theory 3

As mentioned in the introduction, this thesis is comparing different machine learning methods in forecasting day-ahead prices. This chapter sets a framework for understanding the methods and concepts of the analysis. Another objective of this chapter is to uncover the strengths and limitations of the different methods. Hence we wish to expand and explore the research questions and gain a theoretical foundation for the empirical analysis that follows. First come in this chapter is the variance stabilizing transformations. As electricity prices have a quite a lot of inconsistent variance, it is a good idea to preprocess the data with variance stabilizing transformations, to try and meet the model assumptions of constant residual variance that might be. Then the three main machine learning methods of this thesis, Lasso, neural networks and random forest is investigated. In the end of the chapter, there is a section on evaluation metrics, that is used to compare the models.

3.1 Variance Stabilizing Transformations

In some cases, the variance of a dataset may not be constant. This can lead to issues in statistical analysis. One way to deal with this issue is to use variance stabilizing transformations, which transforms the data in a way that make its variance more consistent. In this section, two of these variance stabilizing transformations, namely the logarithmic transformation and the arcsine transformation, are investigated.

The logarithmic or log transformation is the most commonly used method of stabilizing variance. It is on the form

$$y_i = \log(x_i)$$

where x_i is the ith observation and y_i is the transformed value. This transformation has the properties that it compress the large values and enlarge the small values, thus stabilizing the variance and spikes of the data.

However, when the data exhibit close to zero or negative values, the log transformation might not be ideal. The log transformation is also not defined for zero and negative values. Schneider [2011] proposed to replace the log transformation with the area hyperbolic sine transformation (AHST), which is introduced on the next page.

The AHST is defined as

$$y_i = \sinh^{-1}\left(\frac{x_i - \xi}{\lambda}\right)$$

where $\sinh^{-1}$ is the inverse hyperbolic sine function, x_i is the ith price in the dataset, ξ is an offset and λ is a scale parameter. The $\sinh^{-1}$ function is defined as

$$\sinh^{-1}(x_i) = log(x_i + \sqrt{x_i^2 + 1}).$$

The asymptotic behaviour of the log transformation is kept in the AHST, since

$$\sinh^{-1}(x_i) = log(x_i + \sqrt{x_i^2 + 1}) \approx \text{sign}(x_i) \cdot log(2|x_i|) \quad \text{for} \quad |x_i| \to \infty.$$

The advantage of the AHST is that it behaves well where the log does not. By looking at the definition of the $\sinh^{-1}(x_i)$, the advantage compared to the log transformation is clear. The log transformation is undefined for $x_i \leq 0$. Notice that

$$\begin{aligned} \sqrt{x_i^2 + 1} &> \sqrt{x_i^2} \\ \Rightarrow x_i + \sqrt{x_i^2 + 1} &> |x_i| + x_i \geq 0. \end{aligned}$$

Thus $\sinh^{-1}(x_i)$ is defined for all $x_i \in \mathbb{R}$, as it is synonymous with the log of a positive value.

3.2 The Lasso

One of the objectives of this thesis is to analyse the performance that different variables, such as weather and power generation mix, adds to the modelling. For this reason, it is crucial to know which variables that improve forecasting, and those who do not. The Lasso (Least Absolute Shrinkage and Selection Operator) is a regression method which aim to find the linear relationship between a response variable and set of predictors. Thus it is particularly relevant to the analysis of machine learning methods for electricity price forecasting with weather generation and weather as explanatory variables.

The Lasso is an extension to the ordinary least squares (OLS), so the Lasso objective function is the OLS objective function, namely the residual sum of squares (RSS), with an additional penalty term. The OLS aim to minimize the RSS

$$\sum_{i=1}^{n} \Big(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\Big)^2.$$

The values $\hat{\beta}$ that minimize this sum is called the OLS estimator of β. The Lasso penalty term takes the form of an ℓ_1 norm defined as $||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$. Thus the Lasso objective function is

$$\sum_{i=1}^{n} \Big(y_i - (\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij})\Big)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda ||\beta||_1.$$

The coefficients $\hat{\beta}_\lambda^L$ that minimize this function

$$\hat{\beta}_\lambda^L = \underset{\beta_0,\beta_1,...,\beta_p}{\arg\min} \left\{ \sum_{i=1}^{n} \Big(y_i - (\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}) \Big)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

is the Lasso coefficients, where n is the number of observations, p is the number of predictors, y_i is the response variable for the ith observation, x_{ij} is the value of the $j$the predictor for the ith observation, $\beta_0, \beta_1, ..., \beta_p$ are the regression coefficients to be estimated, and $\lambda \geq 0$ is a tuning parameter to be determined.

The Lasso is very useful for sorting out irrelevant predictors, hence doing variable selection. If a predictor β_j is not effectively decreasing the RSS, the penalty term $\lambda \sum_{j=1}^{p} |\beta_j|$ will ensure that the given predictor is minimized. If the penalty parameter λ is sufficiently large, then some of the predictors will be forced to equal zero. However, if it is too large, the Lasso produce a model, where all the coefficients are zero, namely the null model. On the other hand, if $\lambda = 0$, the Lasso yields the least squares. Thus a good value for λ is important.

3.3 Neural Networks

Neural networks excel with large datasets and they are able to learn from a vast amount of data. Furthermore, neural networks are very flexible and capable of capturing nonlinear relationships in data. Although neural networks are considered "black box models", they are capable of doing feature selection. These properties seem to fit very well to the problem of this thesis. The day-ahead prices provided alone produces 24 data points for every day the last 10 years, not mentioning all the explanatory variables in the form of generation mix and weather data. Furthermore, it is assumed, there will be nonlinear relationships between response and explanatory variables in the data. It is also assumed that not all of the explanatory variables might be usable, why the feature selection property of neural networks is very beneficial. This section is based on [Aggarwal, 2015].

Neural networks simulate the human nervous system, which consists of neurons connected points called synapses. Learning occurs by adjusting the strength of these connections in response to stimuli. Artificial neural networks imitate this process, with nodes called neurons receiving input, performing computations, and passing them to other neurons. The computation function is determined by the weights on the input connections. Training data is analogous to the stimulus that incrementally adjusts these weights for better predictions. The networks effectiveness relies on its structure, which ranges from simple single-layer networks called perceptrons to complex multilayer networks.

Therefore the section is firstly introducing the perceptron as an introduction to neural networks. Then comes a subsection on multilayer neural networks and at last a subsection on backpropagation, which is the procedure of tuning the weights of the neural network.

3.3.1 The Perceptron

The most basic Structure of a neural network is referred to as the perceptron. The perceptron consists of two layers: the input layer and a single output node. In figure 3.1, we can see an illustration of the perceptron architecture.

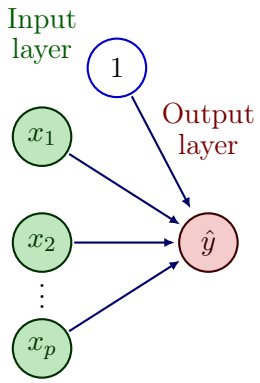

Figure 3.1: Network Diagram of the Perceptron with p input nodes.

The number of input nodes in the perceptron is equal to the dimensionality p of the underlying data. Each input node receives and transmits a single variable value to the output node. The input nodes do not perform any computation on the received values, they simply transmit them. On the other hand, the output node performs a mathematical function on its inputs.

In the basic perceptron model, the individual features in the training data is assumed to be numerical. If there are categorical attributes, they can be handled by creating separate binary inputs for each value of the categorical attribute. An example could be gender, where the feature is 1 if the subject is a female and 0 if it is a male. This binary representation of categorical attributes into multiple attributes allows them to be processed by the perceptron. For simplicity, we assume that all input variables are numerical.

Let $W = (w_1, \ldots, w_p)$ be the weights for the connections of the p different inputs in a data set of dimensionality p. The weights are represented by the arrows in figure 3.1. Additionally, a bias b represents the weight of the white node with the label "1" in the figure. The output $\hat{y}_i$ for the feature set $X_i = (x_{1i}, \ldots, x_{pi})$ of the ith data point is given by:

$$\hat{y}_i = \sigma\left(\sum_{j=1}^{p} w_j x_{ij} + b\right) = \sigma\left(W \cdot X_i + b\right) \tag{3.1}$$

Where, $\hat{y}_i$ is the perceptrons prediction for true value y_i and our goal is to learn the weights so that $\hat{y}_i$ approximates the true variable y_i as closely as possible.

In equation 3.1, σ is the activation function. In the basic perceptron, σ is often assumed to be a linear function, while other common activation functions include sigmoid, ReLU and tanh, each with their own characteristics and advantages.

$$\sigma_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma_{\text{ReLU}}(x) = \max(0, x)$$

$$\sigma_{\tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The functions is illustrated in figure 3.2 below.

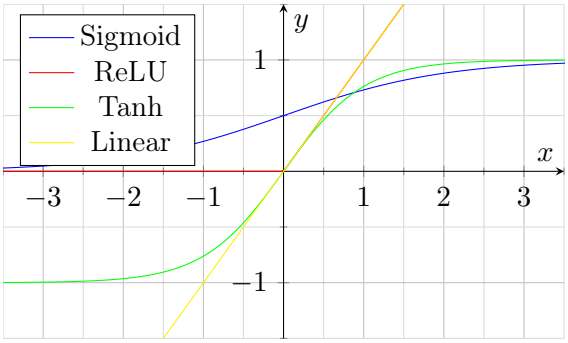

Figure 3.2: Different activation functions.

The basic perceptron algorithm starts with a random vector of weights. The algorithm then processes the input data features from X_i one by one and makes the prediction $\hat{y}_i$. The weights are updated based on the error value $(\hat{y}_i - y_i)$. When the data point X_i is inputted to the perceptron in the tth iteration of the algorithm, the weight vector W_t is updated as:

$$W_{t+1} = W_t + \eta(y_i - \hat{y}_i)X_i$$

Here, η is a parameter that controls the learning rate of the neural network. The perceptron algorithm iterates through all the training examples in the dataset, adjusting the weights incrementally, until it reaches convergence. These cycles or iterations through the training data is called epochs. Weight updates only occur when the error value $(y_i - \hat{y}_i) \neq 0$, as the incremental term becomes 0 when the predicted value matches the true value.

The choice of learning rate η is crucial. A high η leads to fast learning but may yield less than optimal solutions. Conversely, a smaller η ensures convergence to higher quality solutions but slows down the process. Typically, an initially large η is gradually reduced as weights approach the ideal values. This approach balances the advantage of large steps early on with the risk of getting less than optimal solutions at the end. In some cases, η is proportional to the inverse of the number of epochs completed so far. The training of the network is depicted in the following algorithm.

Algorithm 1 Perceptron Learning Algorithm

Initialization: Initialize weight vector W with random values;
repeat
  Receive next training tuple (X_i, y_i);
  $\hat{y}_i \coloneqq \sigma(w_1x_1 + \cdots + w_kx_k + b)$
  $W \coloneqq W + \eta(y_i - \hat{y}_i)X_i$
until ***convergence***;

3.3.2 Multilayer Neural Networks

The perceptron model is a basic form of a neural network with a single input layer and output layer. It learns a simple linear model based on a single output node. In contrast, multilayer neural networks, also known as *deep neural networks* or *deep nets*, have a hidden layer in addition to the input and output layers, thus allowing for more complex models. The hidden layer can consist of multiple layers with different connectivity patterns. An example of a multilayer neural network is illustrated in figure 3.4 below.

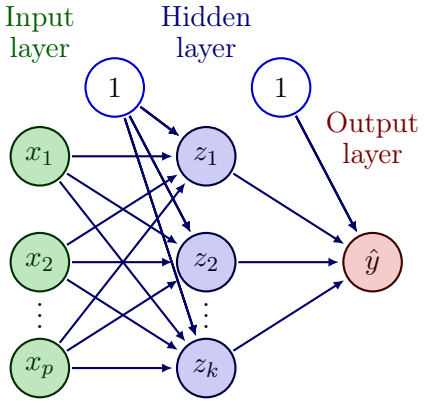

Figure 3.3: Network Diagram of the multilayer neural network with p input nodes and k nodes in the hidden layer.

In the multilayer neural network, it is not only the node in the output layer that is doing a calculation in the form of the activation function. The calculation is also done in the hidden layers, where the nodes take variables from the previous layer along with the corresponding weights and make the calculation:

$$z_j = \sigma\left(w_{0j}^{(1)} + \sum_{i=1}^{p} w_{ij}^{(1)} x_i\right), \quad j = 1, \ldots, k, . \tag{3.2}$$

Where $w_{ij}^{(1)}$ is the weights from the first layer. If there is more than one hidden layer, the nodes $z_j^{(2)}$ in the second layer is calculated as in equation 3.2, but with weights $w_{ij}^{(2)}$ and inputs $x_i = z_i^{(1)}$ and so forth for the following layers. Note that the number of nodes k in the hidden layers might vary from layer to layer. Just as in the perceptron, the multilayer neural network can use different activation functions, such as ReLU and sigmoid, to calculate the nodes in the hidden layer.

This study specifically investigate what is called feed-forward networks. These networks enable information to flow in a single direction, from the input layer through the hidden layers to the output layer. Other types of networks, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) allows for information to travel backwards and in loops in the system.

In a multilayer network, we can directly apply error calculation at the final layer by comparing the actual and predicted values like in the perceptron. However, this is not possible in the hidden layers as there is no target values. To update the weights throughout the network, we use backpropagation, which is introduced in the following section.

Backpropagation

This section is based on [Pajankar and Joshi, 2022]. Backpropagation involves propagating the error from the last layer to the first layer. This process includes forward propagation, where computations are performed layer by layer, and then weight updates using gradient descent. This iterative process is known as backpropagation. Consider a simplified neural network with one hidden layer.

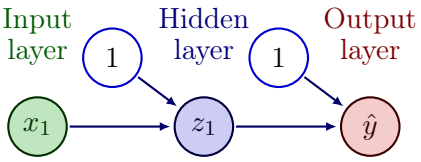

Figure 3.4: Network Diagram of the simplified neural network with a single hidden layer.

The output of the hidden layer, denoted as $\hat{y}$, is computed as follows:

$$\hat{y} = \sigma_2(w^{(2)}\sigma_1(w^{(1)}x + b_1) + b_2),$$

where σ_2 and σ_1 represents activation functions for the output layer and the hidden layer respectively. To update the weights, we compute the partial derivatives of the loss function with respect to $w^{(1)}$ and $w^{(2)}$. In this example, the mean squared error (MSE) is the loss function of choice:

$$J = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

By using the chain rule, the derivative with respect to $w^{(2)}$ is given by:

$$\frac{\partial J}{\partial w^{(2)}} = \frac{1}{n}\sum_{i=1}^{n} 2\left(\hat{y}_i - y_i\right)\frac{\partial \hat{y}_i}{\partial w^{(2)}}$$

Using the chain rule twice, the partial derivative $\frac{\partial \hat{y}_i}{\partial w^{(2)}}$ can be expressed as:

$$\frac{\partial \hat{y}_i}{\partial w^{(2)}} = \sigma_2'\left(w^{(2)}\sigma_1(w^{(1)}x_i + b_1) + b_2\right)\sigma_1\left(w^{(1)}x_i + b_1\right) \tag{3.3}$$

Similarly, the partial derivative with respect to $w^{(1)}$ is computed as:

$$\frac{\partial J}{\partial w^{(1)}} = \frac{1}{n}\sum_{i=1}^{n} 2\left(\hat{y}_i - y_i\right)\frac{\partial \hat{y}_i}{\partial w^{(1)}}$$

And $\frac{\partial \hat{y}_i}{\partial w^{(1)}}$ is given by:

$$\frac{\partial \hat{y}_i}{\partial w^{(1)}} = \sigma_2'\left(w^{(2)}\sigma_1(w^{(1)}x_i + b_1) + b_2\right)\left(w^{(2)}\sigma_1'(w^{(1)}x_i + b_1)\right)(x_i) \tag{3.4}$$

We have now computed some of the elements of equation 3.4 in the calculations from equation 3.3. The trick in backpropagation is to avoid doing unnecessary duplicate calculations. If there is more than one hidden layer, we just continue computing partial derivatives using the chain rule multiple times. At each iteration, the output values for σ_1 and σ_2 in what is called the forward pass. Then follows the backward phase, where the weights are updated using gradient descent.

3.4 Random Forests

Just as with the neural networks, random forests are very capable of handling large datasets, capturing nonlinear relationships in data and provide a measure of feature importance. However, where it can be challenging to interpret the feature importance in neural networks, random forest provide, a specific measure for which variables that have the most significant impact on the forecast. Also random forest is in general a more simple method that require less hyperparameters than neural networks. This section is based on [Aggarwal, 2015] and [Hastie et al., 2009]

Random forest is an algorithm that uses a collection of decision trees for the given dataset. Each decision tree in the "forest" each hold a vote for the output. Thus firstly this section will examine decision trees, then an expansion of the decision tree to regression trees. At last bootstrap aggregation (bagging) and its relation to random forests will be examined.

3.4.1 Decision Trees

Decision trees are classification models that use a hierarchical structure of decisions on feature variables to model a classification process. The decision tree consists of nodes, which correspond to split criteria, and branches, representing the possible outcomes. The nodes are organized in a top-down manner, where each node receives a subset of the data determined by the combination of split criteria from the nodes above it.

The initial node, known as the root node, is where the first split occurs. The subsequent nodes are referred to as decision nodes and the final nodes are called leaf nodes. The leaf nodes provide the final predictions or estimates generated by the decision tree model. Figure 3.5 illustrates the decision nodes as rectangles with the split criteria inside and the leaf nodes as red circles with the output. The root node is the green decision node.

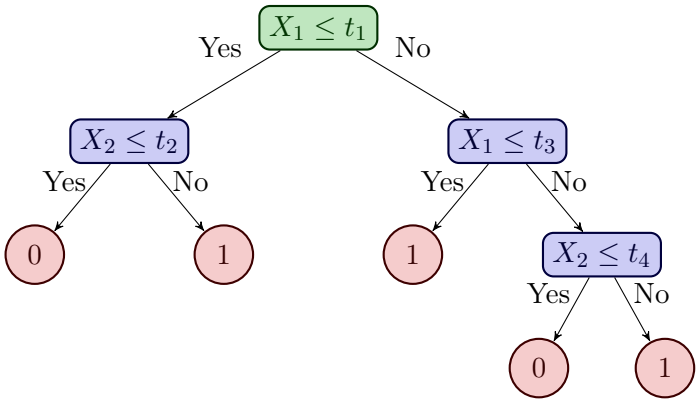

Figure 3.5: Example of a decision tree, with binary output.

In figure 3.5 t_1, t_2, t_3 and t_4 is values that the feature variables X_1 and X_2 is measured against in order to split the data set. The split criterion at a given node is a condition on one or more variables in the training data. The goal is to identify a split criterion that minimizes the mixing of the response variable in each branch of the tree. For example if the data has a binary response variable, the goal is to have a split criterion where there is as many "ones" in one branch and as many "zeros" in the other branch as possible.

When constructiong a decision tree, a number of splits is performed in a top-down fashion, with a goal of creating leaf nodes where the mixing of the response variable is minimized. The following pseudo code presents an algorithm for constructing a decision tree.

Algorithm 2 Decision Tree Algorithm

begin
  Create root node containing the data set;
  repeat
    Find best split using FSM
    Generate daughter nodes
  until ***Stopping criteria is reached***;
  Prune overfitting nodes
  Each leaf node is labelled with its dominant class
end

The following sections will explain the feature selection measures (FSM) used in the algorithm, as well as an explanation of the concept "stopping criteria".

Feature Selection Measure (FSM)

When building a decision tree, we face the problem of selecting the optimal split among several different options. This involves making choices from a set of features and considering different possibilities for splitting each feature. Thus, it becomes crucial to have tools to measure the effectiveness of these splits. The Gini index is one such tool, typically used to measure effectiveness of splits for categorical features. The Gini index is defined as

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2,$$

where S is a set of data points. k is the number of classes, in the example in figure 3.5 the classes is "0" and "1". p_j is the fraction of values from s that belongs to the jth class. The Gini index indicates a perfect split when $G(S) = 0$, as all the feature values must belong to the same class. In the opposing case where the feature values are evenly distributed between k classes, the Gini index takes the maximum value $1 - 1/k$.

The $G(S)$ from before explain the distribution of feature values in a single node. However, we want to look at a split of S into the sets $S_1, \ldots, S_r$, thus the r-way split Gini index is introduced

$$G_{split}(S_1, \ldots, S_r) = \sum_{i=1}^{r} \frac{n_i}{n} G(S_i).$$

Thus the Gini index for the split is a weighted average of the Gini indexes from the individual nodes. The weights n_i/n is the number of data points in S_i, divided by the number of data points in S, where $\sum_{i=1}^{r} n_i = n$.

Stopping Criteria

In the case of a binary or categorical response variable, a stopping criterion can be when each leaf node contains instances that belong to only one category. For a binary response variable, the algorithm stops when each leaf node contains only "ones" or "zeros". However, this approach can lead to overfitting, where the tree becomes overly specific to the training data and performs poorly on test data. In general, we prefer simpler models over complex ones. To avoid overfitting, we can prune the tree by converting some internal nodes into leaf nodes. There are different ways to decide which nodes to prune, and one approach is to penalize complex trees by number of nodes compared to the error.

In the next subsection, we will explore "Regression Trees" and their role in random forests.

3.4.2 Regression Trees

In the previous section, decision trees were discussed. They are good at classifying catagorical variables, but when dealing with numeric response variables, such as the day-ahead prices, a different approach is needed. This is where regression trees come into play. Regression trees are designed to model nonlinear relationships between the features and the response variable.

Consider the example in figure 3.5, we can look at the splits as regions. The first region, R_1, is bounded by $X_1 \leq t_1$ and $X_2 \leq t_2$, R_2 is bounded by $X_1 \leq t_1$ and $X_2 > t_2$, R_3 is bounded by $t_1 < X_1 \leq t_3$, R_4 is bounded by $X_1 > t_3$ and $X_2 \leq t_4$ and R_5 is bounded by $X_1 > t_3$ and $X_2 > t_4$. These regions are illustrated in figure 3.6 below.

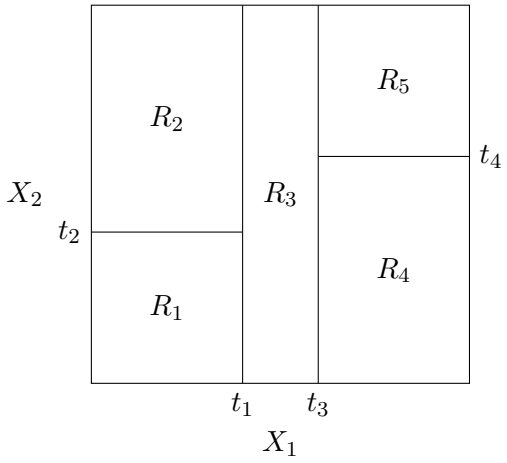

Figure 3.6: Partition of the data into regions.

In the case of a categorical response variables, each region is assigned a constant c_m as response variable. Where $m \in 1, \ldots, M$ is number of regions. Thus

$$f(X) = \sum_{m=1}^{M} c_m \mathbb{1}(X \in R_m).$$

When the response variable is a continuous variable, the best estimation of c_m is the average of y in R_m,

$$\hat{c}_m = \frac{1}{N_m} \sum_{i=1}^{n} y_i \mathbb{1}(x_i \in R_m),$$

where $N_m = \#\{x_i \in R_m\}$ is the cardinality of x_i in R_m. To find the best split, we start with all the data and consider a splitting variable X_j and a split value s. In this case the binary split is considered, hence we define the two regions

$$R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\}.$$

The aim is to find variables X_j and s that solves the following

$$\min_{j,s} \left(\min_{c_1} \left(\sum_{i|x_i \in R_1(j,s)} (y_i - c_1)^2 \right) + \min_{c_2} \left(\sum_{i|x_i \in R_2(j,s)} (y_i - c_2)^2 \right) \right). \tag{3.5}$$

The two inner minimizations is solved by the averages

$$\hat{c}_1 = \frac{1}{N_1} \sum_{i=1}^{n} y_i \mathbb{1}(x_i \in R_1) \text{ and } \hat{c}_2 = \frac{1}{N_2} \sum_{i=1}^{n} y_i \mathbb{1}(x_i \in R_2).$$

By calculating the inner part of equation 3.5 for each split between all inputs, the best split is found as the one with the lowest value. This is done for each splitting variable j, and thus the best pair (j, s) is found. Then we have a partition of the data into two regions and the splitting process can be repeated on the regions recursively until some stopping criteria.

One stopping criteria could be when there is no decrease in MSE. However, there would still be a chance of a missing a very good split later on. Thus the preferred strategy according to Hastie et al. [2009] is to stop the splitting process when some minimum size of the nodes is reached. This could be a node size of minimum five. Then the three is pruned using *cost-complexity pruning*, which is described below. First $T \subset T_0$ is defined as any subtree possibly obtained by pruning T_0. The leaf nodes is indexed by $m \in 1, \ldots, |T|$, where $|T| = \#\{\text{leaf nodes in } T\}$. Let

$$Q_m(T) = \frac{1}{N_m} \sum_{i=1}^{n} (y_i - \hat{c}_m)^2 \mathbb{1}(x_i \in R_m),$$

then we define the cost complexity criterion by

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|,$$

where $\alpha \geq 0$ is called the tuning parameter. For each α, there is a unique subtree T_α that minimizes $C_\alpha(T)$. Large values of α result in smaller trees and smaller values result in larger trees. Note that if $\alpha = 0$, then $T_\alpha = T_0$. To find T_α the weakest link pruning algorithm is used. To estimate α, we use cross validation.

3.4.3 Random Forests and Bagging

This section is based on Hastie et al. [2009].

Random forest is a powerfull machine learning method that is based on the concept of bagging. The aim of bagging is to reduce the variance of the model predictions by taking an average of multiple models that is trained on bootstrapped subsets of the data. Bagging is composed of the words bootstrap and aggregation. Bootstrapping is the procedure of randomly drawing datasets with replacement from the original dataset $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), \}$. We produce B bootstrapped datasets denoted Z^{*b} for $b = 1, 2, \ldots, B$, with the same sample size as the original dataset. This means that if the original dataset is of sample size N, we select N random datapoints where the chance of picking any point is $1/N$. After selecting a point for the bootstrap dataset, the probability of selecting any point in the future does not change. This is done B times to obtain the desired number of bootstrap datasets.

Furthermore in the tree growing process, only m features are selected at random for each tree, in order to decrease the correlation between the individual trees. The inventors of random forest suggest to use $m = \lfloor \sqrt{p} \rfloor$ for classification problems and $m = \lfloor p/3 \rfloor$ for regression [Hastie et al., 2009]. Where $\lfloor x \rfloor$ denotes the floor function of x.

The aggregation part of bagging consists of averaging the models fitted from the the bootstrapped datasets. The trees is created using the regression tree procedure from section 3.4.2. For each

bootstrap sample Z^{*b}, a model is fit, resulting in the prediction $\hat{f}^{*b}(x)$. The bagging estimate is then

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

As trees are known to be noisy, it is a great advantage to average them. Since the trees generated by bagging is identically distributed, we have that $\mathrm{E}[\hat{f}_{bag}(x)] = \mathrm{E}[\hat{f}^{*b}(x)]$ for any b. This means that the bias of the bagged trees is the same of that of the individual trees. Therefore the only way to improve, is by reducing variance. Due to the identical distributions of the trees, the variance of $\hat{f}_{bag}(x)$ is

$$\begin{aligned}
\mathrm{Var}\left(\hat{f}_{bag}(x)\right) &= \mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)\right) \\
&= \sum_{i,j=1}^{B}\frac{1}{B^2}\mathrm{Cov}\left(\hat{f}^{*i},\hat{f}^{*j}\right) \\
&= \sum_{b=1}^{B}\frac{1}{B^2}\mathrm{Var}\left(\hat{f}^{*b}(x)\right) + \sum_{i\neq j=1}^{B}\frac{1}{B^2}\mathrm{Cov}\left(\hat{f}^{*i},\hat{f}^{*j}\right) \\
&= \frac{1}{B^2}B\sigma^2 + \frac{1}{B^2}B(B-1)\rho\sigma^2 \\
&= \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2
\end{aligned}$$

The first term goes to zero as B increases and the size of the second term is decided by the correlation between the trees. Thus the aim of random forests is to reduce the variance by reducing the correlation between the trees. That is the reason for the random feature selection in the bootstrapping step of the random forest algorithm.

The random forest algorithm is displayed below:

Algorithm 3 Random Forest Algorithm

begin

1. For b=1 to B;
   a) Draw bootstrap sample Z^{*b} from Z.
   b) For each terminal node, until minimum node size is reached;
      i. Select m random features from the p features of the data.
      ii. Select the best split among the m features.
      iii. Split the node into two daughter nodes.
2. Return $\{\hat{f}^{*b}(x)\}_{b=1}^{B}$

The models prediction at x is then:

$$\hat{f}_{\text{rf}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

end

The algorithm concludes the random forest theory, a potent machine learning method. Some take-aways from random forest is that they are supposed to produce very accurate and robust predictions, while being preventative of overfitting. One problem might be that random forests can require a lot of memory and thereby be time consuming, as they are able to handle large datasets, and can use many decision trees in the process. These factors will be investigated in the application section.

3.5 Bias-Variance Tradeoff

This section is a short introduction to some of the hurdles that data analysts face. Logunova and Khaciyants [2023] have written an article about the topic, and this section has drawn inspiration from them. In machine learning, finding the right equilibrium between bias and variance is essential for model performance. This tradeoff describes the balance between a models simplicity and its ability to capture complex patterns in data.

Bias

Bias is the error that is caused by a model with too simplictic assumptions about the data and underlying distribution. High bias results in a model that underfits the data and fails to capture all of the patterns in the data. It translates to missing details that were important to the end result. Bias is calculated as:

$$Bias(\hat{f}(x)) = E[\hat{f}(x)] - f(x)$$

Where $\hat{f}(x)$ is the models prediction and $f(x)$ is the true underlying function.

Variance

Variance, quantifies a models sensitivity to small variations or noise in the training data. High variance models are too flexible, fitting the training data closely but failing to generalize to new data. It is like focusing too much on one task (fitting the training data) and forgetting the bigger picture. Variance is calculated as:

$$Var(\hat{f}(x)) = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

The Tradeoff

The bias-variance tradeoff is characterized by the following relationship:

- **High Bias, Low Variance:** Models oversimplify and underfit.
- **Low Bias, High Variance:** Models overfit and are too sensitive to noise.

The objective is to find the sweet spot, a model with reasonable bias and variance that captures essential patterns while maintaining generalization. The bias-variance tradeoff is important for precise and efficient models for the day-ahead price forecasting. In the data analysis these principles are key to assess the models performance, and finding efficient predictions in the Nordic power market.

3.6 Evaluation Metrics

When evaluating the performance of the models of this thesis, different metrics are used to compare the accuracy and predictive power of the models. In this section, we will discuss four evaluation measures: Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE).

Mean Absolute Error (MAE)

MAE measures the average absolute difference between the predicted values $\hat{y}_i$ and the actual values y_i. It provides a straightforward indication of how close the predictions are to the true values without considering the direction. The formula for MAE is given by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE)

MSE is another common evaluation metric that calculates the average of the squared differences between the predicted values $\hat{y}_i$ and the actual values y_i. It amplifies larger errors compared to MAE, making it more sensitive to outliers. The formula for MSE is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Absolute Percentage Error (MAPE)

MAPE measures the average percentage difference between the predicted values $\hat{y}_i$ and the actual values y_i. It is particularly useful when the magnitude of the data varies significantly. The formula for MAPE is given by:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \cdot 100$$

Root Mean Squared Error (RMSE)

RMSE is the square root of the MSE and represents the average magnitude of the errors in the same scale as the target variable. The formula for RMSE is given by:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

When selecting an evaluation measure, it is important to consider the specific given problem. While MAE and RMSE are widely used and provide great measures of model performance, MAPE offers additional insights into the relative error magnitude, and MSE captures the squared errors more prominently.

Application 4

4.1 Data

Nord Pool is the provider of the day-ahead data for this project. Through their FTP server data on day-ahead, intraday, system prices and more are available. In the day-ahead categori, data from Denmark, Estonia, Finland, Latvia, Lithuania, Norway and Sweden is available. The first data is available from 1992 where Nord Pool was founded and as more and more countries joined, more data becomes available. On that account, data from Latvia, which was the latest country to join, is available from 2013 and forward.

The data is scraped from the server using a python script, and stored in categorical csv files according to the region of the data. The files obtained are listed below.

- DK1
- DK2
- Estonia
- Finland
- Latvia
- Lithuania
- SE1
- SE2
- SE3
- SE4
- NO1
- NO2
- NO3
- NO4
- NO5

The solar and wind power production forecasts, are scraped from the ENTSO-E transparency web-side using another python script. The data is sorted into the same regions as listed above. The columns consists of "solar production", "onshore wind production" and "offshore wind production", all in MW.

4.2 Variance Stabilizing Transformation

The following analysis is done for each of the 15 areas, while DK1 is used as an example throughout the section. The first step in the data analysis is to apply the area hyperbolic sine transformation from section 3.1, to reduce the variance of the time series. The optimal values for ξ and λ are calculated using MLE. In the following figure, the day-ahead prices for DK1 is depicted together with the series after the transformation.

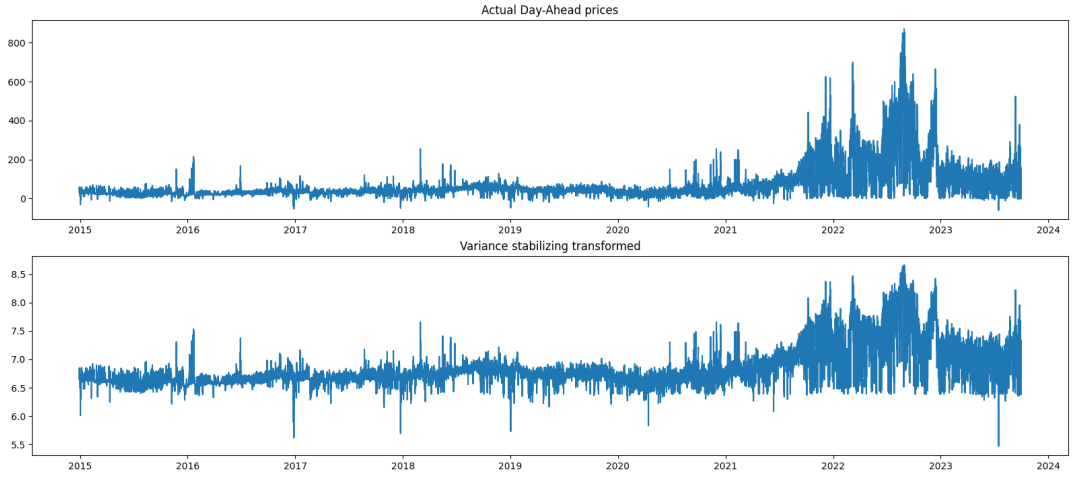

Figure 4.1: Difference between DK1 before and after variance stabilizing transformation.

It is clear from figure 4.1 that the transformation reduces variance in the series.

From the bottom plot in figure 4.1, we notice that there seems to be some seasonality in the data, which was also discovered in the initial data exploration section 2.2. We use a similar approach to Ergemen et al. [2016], to seasonally adjust the data by the following filtering:

$$Y_t = a_0 + a_1 t + a_2 cos\frac{2\pi t}{365.25} + a_3 cos\frac{2\pi t}{24} + D_t$$

Where D_t is a dummy variable to capture weekends. The deseasonalized data can be seen in the figure below.

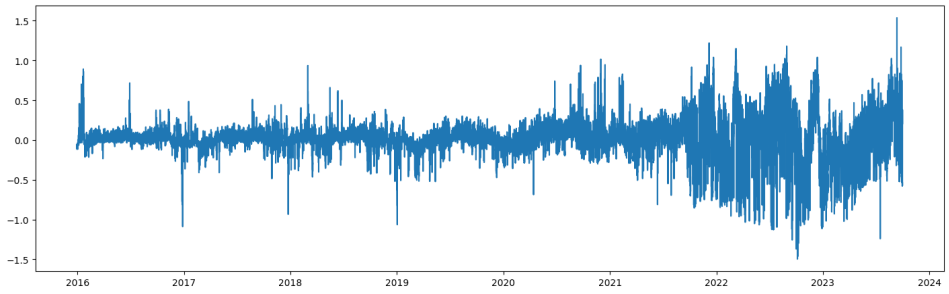

Figure 4.2: DK1 after deseasonalization.

The 15 time series were then tested for stationarity using the augmented Dickey-Fuller (ADF) test and none of them showed signs of any.

4.3 Benchmark models

The first models in this analysis, are the benchmark models. This is simple models that are used to compare to the machine learning models that is to come. In the machine learning section of the

analysis, it becomes crucial to have a large training data set, this is why we aim to forecast 2023 and use data from 2015 until 2023 as training data. To be able to compare the performance metrics against each other, the benchmark models will forecast prices in 2023 only as well.

Naïve model

The first model is the naïve model, which is based on the idea that the price of yesterday reflect the price of today. The naive model is thus given by $\hat{y_t} = y_{t-24}$. In the table below, the metrics for the model is depicted.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
mae	0.2104	0.2115	0.1963	0.1910	0.1940	0.1959	0.0870	0.1087	0.0830	0.0562	0.0789	0.1135	0.1129	0.1801	0.2070
mse	0.0787	0.3870	0.0707	0.0728	0.0683	0.0697	0.0174	0.0255	0.0146	0.0080	0.0135	0.0250	0.0248	0.0625	0.0781
mape	324.6174	308.7380	197.0617	245.5622	302.7896	413.9749	351.9505	322.3198	268.8192	193.6351	207.5732	354.3370	457.4803	1010.2757	454.7957
rmse	0.2805	0.6221	0.2659	0.2698	0.2614	0.2639	0.1319	0.1596	0.1206	0.0895	0.1162	0.1580	0.1574	0.2500	0.2795

Figure 4.3: Metrics for the naïve models.

From figure 4.3 it seems like NO4 is the area where it is most effective to guess the price of tomorrow as the price of today.

ARIMA

The next benchmark is the ARIMA models. To make the job of building these models easier, as there is 15 models to be built, the python package "pmdarima" offers an auto arima which have been used to construct the models. One problem is that there is no "rolling" forecast options to the ARIMA models, meaning in order to forecast for a year, the model either have to be fitted every single day with new parameters or fitted once and forecast a year in advance. Both strategies have been explored, but obviously the forecasts went to zero after only a couple of days, in the example of forecasting a year in advance, yielding a MAPE value of 100. The metrics for the rolling and fitting everyday strategy is shown in the figure below.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
mae	0.2012	0.2223	0.2613	0.2079	0.2542	0.2521	0.0735	0.1081	0.0754	0.0566	0.0697	0.1014	0.1025	0.1736	0.1996
mse	0.0674	0.2727	0.1091	0.0799	0.1053	0.1045	0.0137	0.0234	0.0109	0.0065	0.0116	0.0189	0.0191	0.0541	0.0682
mape	166.8590	194.7701	129.5089	144.9673	358.6020	305.8998	208.7126	142.0871	198.7947	149.3895	122.0652	199.3034	219.4527	400.6511	183.6982
rmse	0.2596	0.5223	0.3304	0.2827	0.3245	0.3232	0.1172	0.1530	0.1043	0.0807	0.1075	0.1376	0.1383	0.2327	0.2611

Figure 4.4: Metrics for the ARIMA models.

Again the NO4 area seem to be the easiest for the models to fit. The best fitting model for the NO4 data was an ARMA(2,2) model.

4.4 Machine Learning Models

In this part of the data analysis, the machine learning models is introduced. Furthermore we introduce exogenous variables to the analysis. Besides the exogenous variables onshore wind-, offshore wind- and solar power production forecasts, the day-ahead prices shifted 24 hours is also used. This makes sense as the 24 hourly day-ahead prices is published once a day, thus we can use the prices from today to predict the prices tomorrow. The exogenous variables are scaled to be between 0 and 1 to make sure that there is not a single variable with a potential great magnitude that dominates

the learning process. As previously mentioned, the data from the years 2015 until 2023 is used as training for all the machine learning models, and the test period is 2023 throughout September.

Lasso

The first machine learning model is the Lasso. There is a variety of great python packages to perform Lasso, and the one used in this analysis is the "sklearn LassoCV". As the name suggest, this package uses cross validation to choose the best model. An advantage to this model, is the speed. Compared to neural networks and random forest, Lasso is very quick. The performance metrics of the Lasso models is depicted below.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
mae	0.1919	0.1934	0.1976	0.1828	0.1929	0.1943	0.0838	0.1038	0.0795	0.0540	0.0775	0.1081	0.1075	0.1653	0.1843
mse	0.0612	0.2868	0.0646	0.0594	0.0627	0.0633	0.0154	0.0221	0.0126	0.0068	0.0123	0.0212	0.0207	0.0481	0.0577
mape	220.2287	212.4932	139.8814	179.8202	194.9726	274.2817	308.3466	275.9530	218.9955	161.4787	182.4026	279.9892	300.8244	643.3335	291.3938
rmse	0.2473	0.5355	0.2541	0.2437	0.2504	0.2516	0.1241	0.1486	0.1121	0.0826	0.1111	0.1454	0.1439	0.2193	0.2401

Figure 4.5: Metrics for the Lasso models.

Overall, the Lasso performs a bit better than the ARIMA model, but the Norwegian area NO1, where the models yield the lowest scores in general shows that the ARIMA is better. The Lasso seems to perform better than ARIMA in Estonia. Thus we take a look at the coefficints that Lasso suggest.

	Solar	Offshore wind	Onshore wind	Historical prices
NO4	0	0	-0.00454923	0.79106339
Estonia	0.00175021	0	-0.01716654	0.61635663

Table 4.1: Lasso coefficients for NO4 and Estonia

It is clear that there is very much weight on the historical prices, especially in NO4, where there is close to no information to gain from the other exogenious variables. The Estonia model seems to find more information from the power production forecasts, and might be a reason for the advantage against the ARIMA model. Below is plotted a sample of the Lasso forecasts in Estonia compared to the true values.

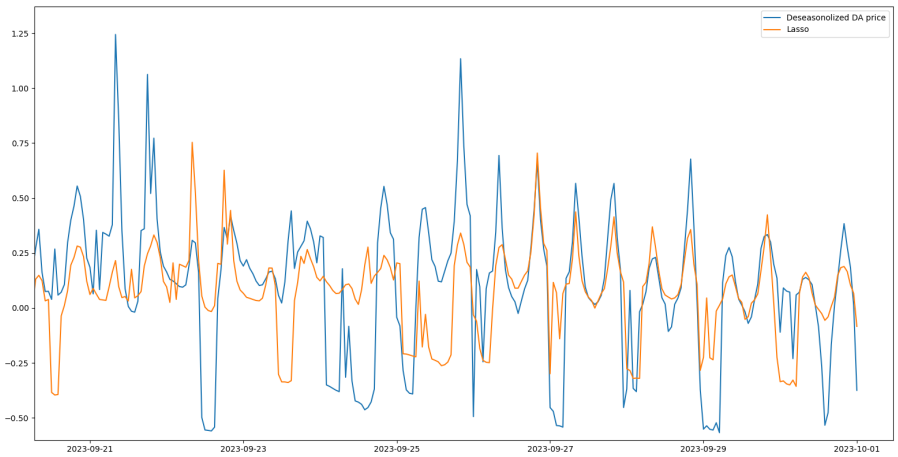

Figure 4.6: Forecast Estonia. Lasso forecast is the yellow line.

It looks like quite a good fit, but it is clear to see that the price from yesterday have a lot of weight.

Neural Networks

The second model is the neural network. The neural networks, compared to the Lasso, is a bit more challenging to build in python. The python packages are not as intuitive and obviously there is a lot of hyper parameter tuning that is not present in Lasso. The analysis make use of the "keras" package for neural network modelling. Another disadvantage is that they are quite time consuming to build. As there was 15 models to be built, the parameters were tuned to speed up the process. This meant a bigger batch size of 32 and less epochs at only 10. The latter did not seem to harm the model, as the loss function did not decrease extensively after 4-5 epochs. The networks consists of two hidden layers of 50 nodes each and uses the "adam" optimizer and MSE as loss function. In the figure below is the metrics from the neural networks displayed.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
mae	0.2029	0.1989	0.1975	0.2075	0.1847	0.1850	0.0988	0.1225	0.1466	0.1068	0.0813	0.1895	0.2025	0.1943	0.2178
mse	0.0691	0.2437	0.0653	0.0717	0.0601	0.0601	0.0188	0.0279	0.0312	0.0160	0.0138	0.0552	0.0671	0.0658	0.0777
mape	280.3415	264.0852	164.8066	296.6384	173.4273	313.1235	373.3961	317.3026	476.8088	364.3156	216.7527	748.6509	1063.6633	1150.9256	394.7908
rmse	0.2629	0.4937	0.2556	0.2677	0.2452	0.2451	0.1371	0.1669	0.1766	0.1265	0.1175	0.2349	0.2591	0.2564	0.2787

Figure 4.7: Performance metrics for the neural networks.

The neural networks is better than the ARIMA model in around half of the areas, but is not as efficient as the Lasso. Again, the machine learning model perform significantly better in Estonia. But to shake things up, let us take a look at the forecast of Latvia, where the machine learning models both outperform ARIMA.

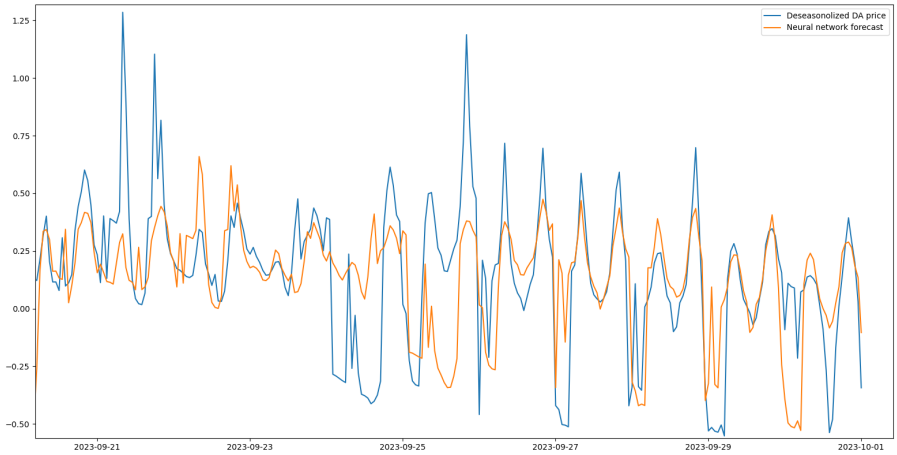

Figure 4.8: Forecast Latvia. Neural network forecast is the yellow line.

It is noticeable how the time series of the price in Latvia was almost identical with the one in Estonia. Obviously these countries lie side by side, but none the less, a majority of the prices are actually identical. From a visual inspection, the fit from the neural network is quite good and it does not seem to suffer as badly from dependence on the historical prices as the Lasso model.

Random Forest

Similarly to the Lasso model, python have some strong and intuitive packages for random forest modelling. Again sklearn have a great tool, the "RandomForestRegressor" package. The models uses 100 trees in the forest and squarred error as split criterion. As the inventors suggested, the max features to consider while looking at a split was $\lfloor p/3 \rfloor = \lfloor 4/3 \rfloor = 1$.

The metrics for the random forest is depicted below.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
mae	0.2056	0.2039	0.2192	0.2277	0.2109	0.2188	0.1035	0.1255	0.0885	0.0609	0.0991	0.1334	0.1343	0.2131	0.2408
mse	0.0711	0.2502	0.0795	0.0934	0.0760	0.0813	0.0220	0.0302	0.0158	0.0085	0.0202	0.0313	0.0329	0.0764	0.1132
mape	235.8713	212.2921	150.3926	267.5756	283.1996	361.3595	354.4453	281.4661	559.3627	209.1068	220.0404	340.5932	436.8162	816.6435	399.8118
rmse	0.2666	0.5002	0.2820	0.3056	0.2757	0.2851	0.1485	0.1738	0.1256	0.0924	0.1421	0.1771	0.1815	0.2764	0.3364

Figure 4.9: Performance metrics for the random forest.

Again as with the two other machine learning models, there was not a clear advantage to random forest compared to the benchmark model ARIMA. It is in the same areas where the machine learning models outperform the ARIMA model and vice versa. Although ARIMA performed better in NO4, it was also the area with the lowest metric values for random forest. NO4 is plotted below.

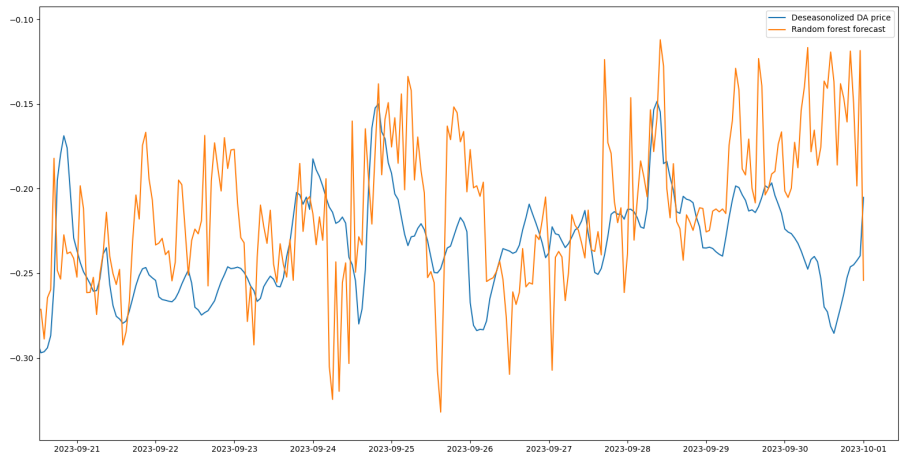

Figure 4.10: Forecast NO4. Random forest forecast is the yellow line.

In figure 4.10 it shows quite promenently how much the random forest overfits the actual data. The forecasts for the areas where the performance was better than the ARIMA was also plottet and inspected, and there were not the same form of overfitting. Thus the term benchmark seems to suit the ARIMA model well.

4.5 Comparison

In this section we look into the performances of the models quantified by the different metrics. Be aware that the strong colors suggest bad models, as the stronger the color, the higher the value.

The first metric is the MAE.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
naive	0.2104	0.2115	0.1963	0.1910	0.1940	0.1959	0.0870	0.1087	0.0830	0.0562	0.0789	0.1135	0.1129	0.1801	0.2070
ARIMA	0.2012	0.2223	0.2613	0.2079	0.2542	0.2521	0.0735	0.1081	0.0754	0.0566	0.0697	0.1014	0.1025	0.1736	0.1996
Lasso	0.1919	0.1934	0.1976	0.1828	0.1929	0.1943	0.0838	0.1038	0.0795	0.0540	0.0775	0.1081	0.1075	0.1653	0.1843
Neural net	0.2029	0.1989	0.1975	0.2075	0.1847	0.1850	0.0988	0.1225	0.1466	0.1068	0.0813	0.1895	0.2025	0.1943	0.2178
Random forest	0.2056	0.2039	0.2192	0.2277	0.2109	0.2188	0.1035	0.1255	0.0885	0.0609	0.0991	0.1334	0.1343	0.2131	0.2408

Figure 4.11: MAE values for all models

The first thing to notice from figure 4.11 is that the naïve model actually seems to fit the Estonia area the best according to MAE. This is quite surprising, as the machine learning methods was suggesting very good models in this particular area. The first thought was that the explanatory variables would add some information that the historic price did not, but the naïve model is only the historic price. Other than that, we notice that Lasso is very average through it all and neural networks and random forest perform significantly better outside of Norway and Sweden.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
naive	0.0787	0.3870	0.0707	0.0728	0.0683	0.0697	0.0174	0.0255	0.0146	0.0080	0.0135	0.0250	0.0248	0.0625	0.0781
ARIMA	0.0674	0.2727	0.1091	0.0799	0.1053	0.1045	0.0137	0.0234	0.0109	0.0065	0.0116	0.0189	0.0191	0.0541	0.0682
Lasso	0.0612	0.2868	0.0646	0.0594	0.0627	0.0633	0.0154	0.0221	0.0126	0.0068	0.0123	0.0212	0.0207	0.0481	0.0577
Neural net	0.0691	0.2437	0.0653	0.0717	0.0601	0.0601	0.0188	0.0279	0.0312	0.0160	0.0138	0.0552	0.0671	0.0658	0.0777
Random forest	0.0711	0.2502	0.0795	0.0934	0.0760	0.0813	0.0220	0.0302	0.0158	0.0085	0.0202	0.0313	0.0329	0.0764	0.1132

Figure 4.12: MSE values for all models

The MSE plot below is very similar to the MAE plot, but fortunately the naïve model is not the best in Estonia. The Lasso is again very average and the neural networks are still the best outside of Norway and Sweden.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
naive	324.62	308.74	197.06	245.56	302.79	413.97	351.95	322.32	268.82	193.64	207.57	354.34	457.48	1010.28	454.80
ARIMA	166.86	194.77	129.51	144.97	358.60	305.90	208.71	142.09	198.79	149.39	122.07	199.30	219.45	400.65	183.70
Lasso	220.23	212.49	139.88	179.82	194.97	274.28	308.35	275.95	219.00	161.48	182.40	279.99	300.82	643.33	291.39
Neural net	280.34	264.09	164.81	296.64	173.43	313.12	373.40	317.30	476.81	364.32	216.75	748.65	1063.66	1150.93	394.79
Random forest	235.87	212.29	150.39	267.58	283.20	361.36	354.45	281.47	559.36	209.11	220.04	340.59	436.82	816.64	399.81

Figure 4.13: MAPE values for all models

In the MAPE plot in figure 4.13, we see that the ARIMA model is very bright colored all the way through the areas, indicating the best overall model by this metric. The neural network is on the other hand not performing well, not even in the non Sweden and Norway areas.

	DK1	DK2	Estonia	Finland	Latvia	Lithuania	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4
naive	0.2805	0.6221	0.2659	0.2698	0.2614	0.2639	0.1319	0.1596	0.1206	0.0895	0.1162	0.1580	0.1574	0.2500	0.2795
ARIMA	0.2596	0.5223	0.3304	0.2827	0.3245	0.3232	0.1172	0.1530	0.1043	0.0807	0.1075	0.1376	0.1383	0.2327	0.2611
Lasso	0.2473	0.5355	0.2541	0.2437	0.2504	0.2516	0.1241	0.1486	0.1121	0.0826	0.1111	0.1454	0.1439	0.2193	0.2401
Neural net	0.2629	0.4937	0.2556	0.2677	0.2452	0.2451	0.1371	0.1669	0.1766	0.1265	0.1175	0.2349	0.2591	0.2564	0.2787
Random forest	0.2666	0.5002	0.2820	0.3056	0.2757	0.2851	0.1485	0.1738	0.1256	0.0924	0.1421	0.1771	0.1815	0.2764	0.3364

Figure 4.14: RMSE values for all models

The last plot is again very similar to the MAE and MSE plots. To wrap up this section, we can conclude that the ARIMA model performs particularly well in Norway and Sweden. The Naïve model is almost never preferable according to the metrics. Lasso seems to be an overall great model performance metric wise, in addition to it being very light computation wise and intuitive to use. The neural network performed very well in the areas besides Norway and Sweden, and the endless tuning possibilities for this model makes for an interesting model. The random forest did not seem able to outperform the naïve model, which was unexpected. It was only performing as good as the neural network in DK2.

Discussion 5

In this chapter the methods of the thesis will be discussed. Furthermore there will be elaborated on which other directions could have been pursued or will make sense to pursue in further research.

There is a lot of factors that impact the actions on the Nordic power market. The first factor is the level of consumption of power. Through this thesis, the method for dealing with this unknown factor has been to adjust for seasonality throughout the day, the week and the year. This method has proven to work well, as the consumption is, mostly based on time of the day and week. The focus in the thesis has not been on the consumption part, but rather the production. To further improve the results, an idea could definitely be to acquire forecasts and data about consumption patterns.

The next factor is production of power. The initial idea was to obtain weather data for the different areas as well as generation mix and use these to predict the power prices. But there is two types of historical weather data. As it turns out, it is very easy to obtain historical actual weather data, another matter is to acquire historical weather forecasts, that are free. Maybe the actual weather data from today can help predict the price of electricity tomorrow, but it did not seem to be worth the time. An idea could be to use time to contact the individual governmental weather institutes. For instance: the Danish meteorological institute have historical weather forecasts, though for Denmark only. Another possible solution is to pay for the forecasts.

Forecasts that on the other hand is accessible are the wind and solar production forecasts. From a Danes point of view these are some important variables, as a heavy part of Denmarks power production comes from wind. In retrospective it should have been investigated how important wind and solar power are in the other areas as well. The models of the thesis that were using wind and solar power, as exogenous variables, were good in most countries except for Norway and Sweden. This makes sense, as they both rely much more on hydro power than wind and solar. Thus for further research of especially those two countries, other power sources should be accounted for. ENTSO-E, the provider of the wind and solar forecasts, actually provide generation mix, which contain more than 20 power sources. The only problem is that they are also actual data, and therefor neglected for this thesis.

The models of the thesis all seem to have potential, and some good fits were found. An idea for the future would be not to span too wide. Three machine learning models might not be a wide span, but when combined with 15 areas it becomes a cumbersome task. It would have been interesting to further tune on the machine learning models, to see if there was a possibility to gain even more precision on the good forecasts or similarly fix those models who were not performing too well. For future research, a wish is to really go in depth with one or two models on a single or a few time series. A good idea would be to not wait until all the models are finished before comparing the performance metrics. One can easily have the idea that machine learning models always perform better than naïve and ARIMA models, but this was not always the case in this thesis.

It is also interesting to consider the models that beat the benchmark models in performance, all seemed to fit the actual value better. It sounds obvious, but it is interesting how similar models can model similar data, as the Nordic day-ahead prices are very different. Although the models were fit individually, some of the models were over fitting or had random noise in their the output, emphasizing the need for more parameter tuning.

Conclusion 6

The main goal for this master's thesis was to compare forecast accuracy between different machine learning methods and also compare them to benchmark models when forecasting day ahead prices in the Nordic power market. To do so, the Lasso, neural networks and random forests algorithms were introduced and fitted to 15 different time series. One of the results were, that the machine learning methods in general failed to beat the benchmark models in Norway and Sweden. On the other hand they seemed to have an advantage in the rest of the countries. To conclude whether the machine learning methods perform better than the benchmark models, further research into the complex connections in the power market is necessary. If the hypothesis, that Norway and Sweden do not rely significantly on solar and wind power is true, while the remaining countries do, then the models did outperform the benchmark models.

When fitting neural networks to the day-ahead prices, it becomes clear that it is important, to be aware of the behaviour of the output. Neural networks seemingly have an ability to find patterns in the data, and not rely too much on the price of yesterday. But if there is no pattern to be found, they are prone to be over fitting. If supervised correctly neural networks might have a strong potential. On the other hand, both Lasso and random forest find to be more intuitive and non-sensitive to hyperparameter tuning. The most robust model of them all was the ARIMA model. Measured on all the performance metrics ARIMA was a top contender in every country, and seems to be performing well under different conditions. The problem with ARIMA lies in its limitations. For instance it is not suitable for a longer time horizon and might be struggling with more complex patterns.

On the assumption that the reason for the lack of results for the machine learning models in Norway and Sweden, is that there is not a big enough impact factor from solar and wind power on the day ahead prices, it can be concluded that including power production forecasts have an impact on the effectiveness of machine learning methods in the Nordic power market. In future research, it is advisable to dig deeper into the underlying mechanisms and evidence for using power production forecasts in machine learning models for day-ahead price forecasting.

Bibliography

Aggarwal, **2015**. Charu C. Aggarwal. *Data Mining The Textbook*. Springer, 2015. ISBN 978-3-319-14141-1.

Cifter, **2013**. Atilla Cifter. *Forecasting electricity price volatility with the Markov-switching GARCH model: Evidence from the Nordic electric power market*. Elsevier, 2013.

ENTSO-E. ENTSO-E. *ENTSO-E Mission Statement*. URL `https://www.entsoe.eu/about/inside-entsoe/objectives/`.

Ergemen et al., **2016**. Yunus Emre Ergemen, Niels Haldrup and Carlos Vladimir Rodríguez-Caballero. *Common long-range dependence in a panel of hourly Nord Pool electricity prices and loads*. Energy Economics, 2016.

Hastie et al., **2009**. Trevor Hastie, Robert Tibshirani and Jorome Friedman. *The Elements of Statistical Learning*. Springer, 2009. ISBN 978-0-387-84857-0.

Kristiansen, **2014**. Tarjei Kristiansen. *A time series spot price forecast model for the Nord Pool market*. Elsevier, 2014.

Logunova and Khaciyants, **2023**. Inna Logunova and Alexey Khaciyants. *Bias-Variance Tradeoff in Machine Learning*. serokell blog, 2023.

Ludwig et al., **2015**. Nicole Ludwig, Stefan Feuerriegel and Dirk Neumann. *Putting Big Data analytics to work: Feature selection for forecasting electricity prices using the LASSO and random forests*. Journal of Decision Systems, 2015.

NordPool, **2023a**. NordPool. *Day-ahead market*, 2023a. URL `https://www.nordpoolgroup.com/en/the-power-market/Day-ahead-market/`.

NordPool, **2023b**. NordPool. *About us*, 2023b. URL `https://www.nordpoolgroup.com/en/About-us/`.

NordPool, **2023c**. NordPool. *Intraday trading*, 2023c. URL `https://www.nordpoolgroup.com/en/trading/intraday-trading/`.

NordPool, **2023d**. NordPool. *The Power Market*, 2023d. URL `https://www.nordpoolgroup.com/en/the-power-market/`.

Pajankar and Joshi, **2022**. Ashwin Pajankar and Aditya Joshi. *Hands-on Machine Learning with Python*. Apress, 2022. ISBN 978-1-4842-7921-2.

Schneider, **2011**. Stefan Schneider. *Power spot price models with negative prices*. MPRA, 2011.

Winther, **2020**. Christian Dahl Winther. *Visual guide to the power grid*. pages 123–133. The Visual Power Grid Company, 2020. ISBN 9788797195901.