University of Tartu

Faculty of Science and Technology

Institute of Mathematics and Statistics

Robert Pikmets

Forecasting intraday electricity prices on the Nord Pool using LASSO

Actuarial and Financial Engineering

Master's Thesis (30 ECTS)

Supervisor: PhD Raul Kangro

Tartu 2021

FORECASTING INTRADAY ELECTRICITY PRICES ON THE NORD POOL USING LASSO

Master's thesis

Robert Pikmets

Abstract

This thesis aims to forecast hourly intraday electricity prices on the Nord Pool's continuous intraday market Elbas. For this, an aggregate volume-weighted average price of all intraday transactions during the last 4 hours prior to each delivery hour is predicted for the Nordic and Baltic price areas. The main modelling technique used is the least absolute shrinkage and selection operator (LASSO). Two of the most common forecasting frameworks are compared, known as the univariate and multivariate frameworks in the electricity price forecasting literature. The LASSO estimated model set in the univariate framework is found to perform the best, beating the multivariate framework as well as simple benchmark models in terms of forecast accuracy. The best performing LASSO model achieves a MAE of 3.83 EUR/MWh and RMSE of 6.99 EUR/MWh in the out-of-sample test period, representing a 13.6% increase in forecasting accuracy compared to the best naive estimate.

CERCS research specialisation: P160 Statistics, operations research, programming, financial and actuarial mathematics.

Key Words: Intraday electricity market, electricity price forecasting, Nord Pool, LASSO, R (programming language).

NORD POOLI PÄEVASISESTE ELEKTRIHINDADE PROGNOOSIMINE LASSO ABIL

Magistritöö

Robert Pikmets

Lühikokkuvõte

Selle magistritöö eesmärk on prognoosida tunniseid päevasiseseid elektrihindu Nord Pooli päevasiseseks kauplemiseks mõeldud Elbas turul. Selleks ennustatakse Baltikumi ja Põhjamaade hinnaalade jaoks ühist kogusega kaalutud keskmist hinda, mis sisaldab tehinguid, mis leiavad aset nelja viimase tunni jooksul enne igat tarnetundi. Põhiline rakendatav meetod on LASSO regressioon. LASSO puhul võrreldakse ühise mudeli sobitamist kõikide tarnetundide jaoks (nn ühemõõtmeline juht) ning iga tarnetunni jaoks eraldi mudeli loomist (nn mitmemõõtmeline mudeldamine). Tulemustest selgub, et parima prognoosivõimega on ühemõõtmeline juht, mis saavutab võrreldes naiivse hinnaprognoosiga $13,6\%$ madalama prognoosivea valimivälises testperioodis.

CERCS teaduseriala: P160 Statistika, operatsioonianalüüs, programmeerimine, finants- ja kindlustusmatemaatika.

Märksõnad: Päevasisene elektriturg, elektrihinna prognoosimine, Nord Pool, LASSO, R (programmeerimiskeel).

Contents

Introduction

One of the unique characteristics of power markets is the fact that a balance of supply and demand of electricity is required at all times. The main mechanism for ensuring this equilibrium is the auction-based day-ahead market, on which the day-ahead prices for each hour of the following day are determined. Recent years have seen a global rapid expansion of electricity generation from renewable energy sources, such as wind and solar power. However, such energy sources are characterised by variability due to its dependence on weather conditions and can therefore be difficult to forecast ahead. Imbalances between supply and demand are more likely to emerge after the closing of the day-ahead market, which poses a challenge to the stability of the entire power grid.

One of the mechanisms for mitigating these imbalances is intraday trading, which allows market participants to buy and sell electricity very close to the delivery hour. Depending on the weather and market conditions, intraday trade prices can differ significantly from the day-ahead prices. Given the increasing importance of intraday markets, it becomes critical for market participants to be able to accurately forecast these intraday prices in order to aid their decision-making, boost profitability and also ensure the smooth functioning of the power system.

The purpose of the thesis is to develop a statistical method for forecasting an aggregate measure of intraday price on the Nord Pool, which is the power exchange that operates the intraday market Elbas in the Nordic and Baltic countries. More specifically, the thesis aims to predict the volume-weighted average of all Elbas trades that take place during the last 4 hours prior to the start of a given delivery hour. This 4-hour window aims to capture the latest and therefore most relevant price information and should give market participants enough time to act on their own imbalances or other opportunities that the market can potentially offer.

The geographical focus is set on the intraday trades of six countries on the Nord

Pool – Estonia, Latvia, Lithuania, Finland, Sweden and Denmark. Trades for which at least one of the counterparties, i.e. the buyer or the seller belong to the aforementioned areas, have been included in the data set.

Inspired by existing research on both day-ahead and intraday price forecasting, LASSO regression is applied as the main forecasting tool. The developed models utilise a large variety of market variables and the most up-to-date information in an attempt to offer market participants a way to improve their intraday price forecasting accuracy.

Most of the existing academic research is concentrated on forecasting day-ahead prices, which is why this thesis aims to contribute to the rather scarce literature of intraday price forecasting. Intraday price forecasting has only recently started to gather more attention, more so in the context of the German market. However, to the best of knowledge of the author of this thesis, no research has considered the Baltic price areas in the context of intraday price forecasting.

Firstly, a brief overview of the most relevant existing academic research on the topic of electricity price forecasting is presented in Chapter 1. Secondly, Chapter 2 provides the reader with a general background on structure and functioning of the electricity market in the Baltic and Nordic regions. Next, Chapter 3 outlines the process of data collection, pre-processing, aggregation and performs data exploration. In Chapter 4, the methodology and framework of forecasting is introduced and explained. Lastly, results and discussion is presented in Chapter 5.

1 Literature review

This chapter provides an overview of the most relevant existing literature on electricity price forecasting (EPF). Chapter 1.1 discusses the developments in use of statistical methods in day-ahead EPF. Chapter 1.2 focuses on the intraday price forecasting and Chapter 1.3 discusses explanatory variable selection in intraday EPF.

1.1 Day-ahead price forecasting

Lago et al. (2021) outline three main branches of methods for EPF – statistical, deep learning (or more generally, machine learning) and hybrid methods. According to the authors, most models in the statistical methods class rely on linear regression and represent the dependent (or output) variable, i.e. price $p_{d,h}$ for day d and hour h, by a linear combination of independent variables and usually also contain autoregressive terms.

In the field of statistical methods, Lago et al. (2021) argue that the most relevant key contribution in recent years has been the application of linear regression models with a large number of input features that utilize regularization techniques, such least absolute shrinkage and selection operator (LASSO) or its generalization the elastic net. LASSO regression will be further introduced in Chapter 4.3. While LASSO regression can be considered a machine learning technique by some authors, Lago et al. (2021) classify it as statistical as the underlying model is autoregressive.

One of the most notable research papers to apply these regularization methods in day-ahead EPF and achieve state-of-the-art results was authored by Uniejewski, Nowotarski, and Weron (2016), who were one of the first to develop LASSO-estimated models for a large set of predictors consisting of both autoregressive and exogenous variables in this context. This model was also included in the empirical study of Lago, De Ridder, and De Schutter (2018), in which 27 of the most promis-

ing methods for predicting prices on the Belgium day-ahead market were evaluated. While the authors found that in general, deep learning models outperform the more traditional statistical methods, they also show that models employing LASSO or elastic net regularization outperform all other statistical methods. Lago et al. (2021) argue that despite slightly lower forecasting accuracy of LASSO-models compared to deep neural network (DNN) models, their advantage of up to 100 times lower computational costs over DNN models makes them the best available option when fast decision-making and low complexity are of the highest priority.

1.2 Intraday price forecasting

So far, a vast majority of research and its applications have concerned day-ahead electricity prices (Marcjasz, Uniejewski, and Weron, 2020). When it comes to forecasting intraday electricity prices in European power markets, the literature is very scarce (Uniejewski, Marcjasz, and Weron, 2019). However, recent years have seen a shift in research focus to intraday price forecasting due to its increasing importance in balancing the demand and supply of electricity (Marcjasz, Uniejewski, and Weron, 2020).

Uniejewski, Marcjasz, and Weron (2019) were one the first to publish a research paper on intraday EPF in the context of a European power market. More specifically, they consider 12 different models for predicting the ID-3 Price index (a widely used measure for intraday electricity prices) on the German EPEX market. They find that for an appropriately chosen value of the complexity parameter, the LASSO model significantly outperforms alternative models.

However, Narajewski and Ziel (2020) also conduct an empirical study on intraday EPF on the German market, but with controversial results. They conclude that the German market for hourly intraday products is weak-form efficient, meaning in fact the most recent transaction price is the best predictor and none of the more

complex models managed to significantly outperform the naïve most recent value.

In response to Narajewski and Ziel (2020), Marcjasz, Uniejewski, and Weron (2020) publish another paper and show that it is in fact possible to build models that significantly outperform the naïve benchmark. The authors develop a parameter-rich model with four types of fundamental variables as inputs and show that the naïve forecast can be significantly outperformed by combining (forecast averaging) it with a prediction of a LASSO-estimated model.

Whereas the body of literature is starting to build up in the midst of fierce academic discussion regarding the German intraday electricity market, literature in the context of Nordic, and especially Baltic markets is very scarce. To the best of knowledge of the author of this thesis, there are only two works published on forecasting intraday prices in the Nordic area and none in the context of the Baltics. Kolberg and Waage (2018) use deep learning to predict the volume-weighted average Elbas price over the period of six hours ahead of a given hour of power delivery. In fact, this work is most closely related to the topic of this thesis, with some key differences including the choice of bidding areas, prediction time (4 hours vs 6 hours ahead of a delivery hour) and most importantly, the modelling technique (LASSO regression vs deep learning). Kolberg and Waage (2018) do actually include LASSO as one of the simpler benchmark models in their work, but the analysis is very limited as their research focus is clearly set on deep learning.

1.3 Variable selection in intraday EPF

Variable selection is a very important issue in EPF, and it may be even more crucial for intraday markets than for day-ahead markets because of the vast amount of data available (Uniejewski, Marcjasz, and Weron, 2019). The authors find in their intraday study that the most important explanatory variables are the most recent intraday price and the day-ahead price that corresponds to the same delivery hour.

In addition to the most recent intraday price and day-ahead price, Marcjasz, Uniejewski, and Weron (2020) also include lagged hourly consumption and its day-ahead forecast, lagged wind power generation and its day-ahead forecast and lagged photovoltaic generation and its day-ahead forecasts in the model to beat the naive benchmark (i.e. the most recent intraday price). Adding dummy variables as model features to capture seasonal effects is also common, e.g. day of the week and hour of the day as in Kolberg and Waage (2018).

2 Nordic-Baltic electricity market overview

The Nordic and Baltic electricity market can be divided into financial and physical power markets. This thesis sets the focus on the physical market, which contains day-ahead, intraday, and regulating power markets. (Spodniak, Ollikka, and Honkapuro, 2021)

Trading on the day-ahead and intraday markets in the Baltic and Nordic countries is provided to market participants by Nord Pool. In addition, Nord Pool also offers intraday trading to customers in UK, Germany, Poland, France, the Netherlands, Belgium and Austria. (Nord Pool, 2021b)

The power market is divided into many bidding areas (see Figure 1). The different bidding areas help to indicate constraints in the transmission systems and ensure that regional market conditions are reflected in the price. Due to bottlenecks in the transmission system, bidding areas may get different prices called area prices. When there are constraints in transmission capacity between two bidding areas, power will always go from the low price area to the high price area. (Nord Pool, 2021a)

2.1 Elspot market

The most important market by volume and liquidity is the day-ahead market, known as the Elspot market (Spodniak, Ollikka, and Honkapuro, 2021).

The primary role of the Elspot market is to establish a balance between production and consumption of electricity. This equilibrium is especially important in power markets because of the inability to store electricity and the high costs and potentially serious consequences associated with power outages. The day-ahead market at Nord Pool is an auction-based exchange for physical delivery of electricity. (Nord Pool, 2021d)

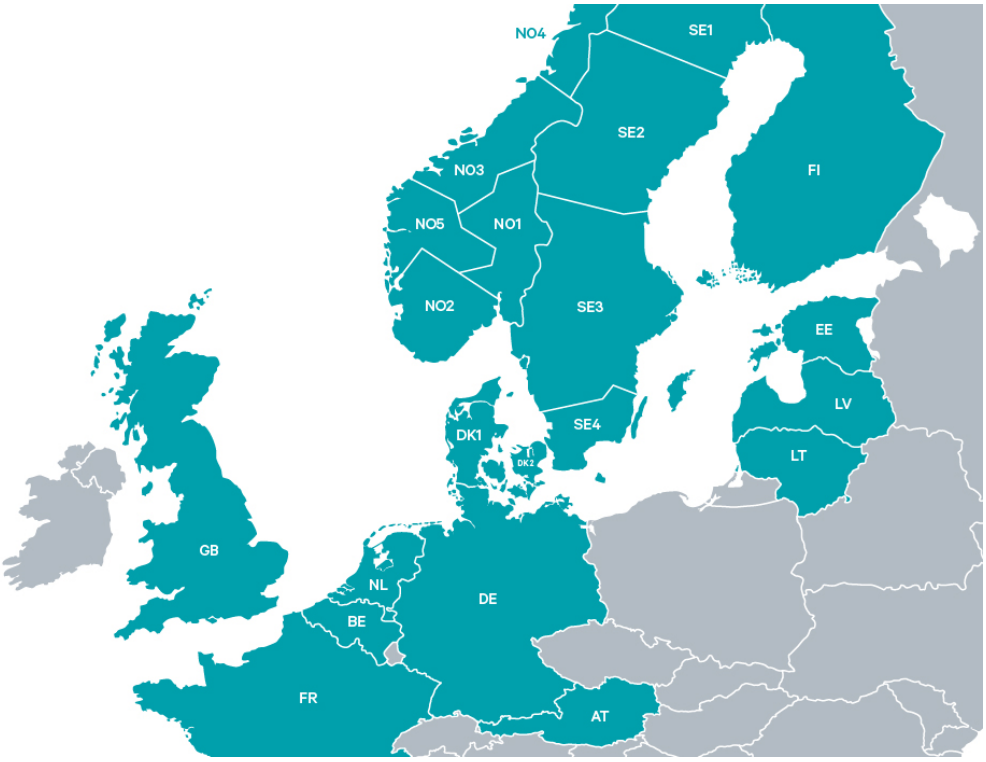

Figure 1: Nord Pool bidding areas. Source: Nord Pool (2021a)

Elspot follows a uniform price auction in which buyers and sellers submit their bids by 12:00 CET the day before for each hour of power delivery the following day. Information about the transmission capacities is made available for the day-ahead auction at 10:00 CET by Nord Pool. Nord Pool then sets the hourly prices so that equilibrium is expected between supply and demand in each of the following day's 24 delivery hours. The day-ahead prices are published shortly after the auction. Therefore, information about the next day's hourly prices is available to market participants 12 hours ahead of the first delivery hour (00:00–01:00 day ahead) and up to 36 hours ahead of the last delivery hour (23:00–00:00 day ahead). (Spodniak, Ollikka, and Honkapuro, 2021)

The theoretical Elspot price that balances overall supply and demand in a market with no transmission grid congestion is called the system price. The system price is used as a reference for trading in the financial electricity markets. However, in

practice, there are constraints on how much power can be transmitted between different areas. As a result, the Nord Pool market is divided into the bidding areas as seen on Figure 1. (Kolberg and Waage, 2018)

2.2 Elbas market

Elbas is a continuous market, on which market participants can trade every day around the clock until one hour before power delivery, and in some cases right up until the delivery hour (Nord Pool, 2021c). This allows market participants to address errors in their consumption and production forecasts and adjust the commitments to receive or deliver electricity made during the day-ahead auction (Spodniak, Ollikka, and Honkapuro, 2021). Therefore, the Elbas market complements the day-ahead market by providing an opportunity to adjust imbalances, i.e. deviations from the participants' day-ahead promises, closer to real time (Kolberg and Waage, 2018).

Elbas market opens at 14:00 CET the day before delivery (Spodniak, Ollikka, and Honkapuro, 2021), so market participants can start trading 10 hours before the first delivery hour and up to 34 hours before the last delivery hour of the next day. Market participants can place orders for 15-minute, 30-minute and hourly products. A product in this context is the amount of electricity to be physically delivered from the buyer to the seller during the given interval at the transaction price. Prices are set based on a first-come, first-served principle, where best prices come first, i.e. the highest buy price and lowest sell price. (Nord Pool, 2021c)

2.3 Regulating power market

Another alternative to the Elbas market for dealing with any imbalances emerging after closing of the Elspot auction, is the regulating, or balancing, power market. The regulating power market is run by local transmission system operators (TSOs). TSOs are neutral market members, who are responsible for the stable operation of

the electrical grid (Nord Pool, 2021e).

However, the Baltic and Nordic countries belong to two different synchronous areas, which are subject to different regulating power market rules. Finland, Sweden, Denmark and Norway form the Nordic synchronous area and Estonia, Latvia and Lithuania form the Baltic synchronous area (Scharff and Amelin, 2016). Due to data availability and differences in methods for balancing, this work only focuses on the Nordic regulating market as data regarding this area has far more impact on the thesis' objective.

The purpose of the regulating market is to secure a constant balance of supply and demand at all times. Electricity producers with flexible power generation can submit bids to the regulating market, which in case of imbalances are activated. Submission of regulating bids for market participants opens at 13:00 CET the day before and closes 45 minutes before the delivery hour. (Kolberg and Waage, 2018)

The clearing methodology is marginal pricing based on the most expensive up-regulation bid or the lowest down-regulation bid activated during the operation hour. Transmission system operators can order up- or down-regulation from the regulating energy market according to the power system requirements, where up-regulation can be achieved by increasing production or reducing consumption, and down-regulation by reducing production or increasing consumption. Additionally, the up- and down-regulation prices also serve as the basis for imbalance prices in the imbalance settlement process, which arises due to the difference between planned and actual physical power delivery. After the delivery hour, deviations between the consumption and production balance responsible bids and the actual amount of electricity provided/used are determined. Local TSOs serve as open suppliers for the balance responsible parties (BRPs) that are obligated to purchase or sell these imbalances to/from the TSO. (Spodniak, Ollikka, and Honkapuro, 2021)

2.4 Importance of the Elbas market

As the topic of the thesis is intraday price forecasting, the importance of the Elbas market should be further discussed. With the increasing amount of renewable energy production, interest in trading in the intraday markets is increasing, as it becomes more and more challenging for market participants to avoid imbalances due to the variable nature of renewable energy sources, such as wind and solar power. Being balanced on the network closer to delivery time is beneficial for market participants and for power systems alike by, among other reasons, reducing the need for reserves and associated costs. In addition, the intraday market is an essential tool that allows market participants to take unexpected changes in consumption and outages into account. (Nord Pool, 2021c)

The remainder of the chapter is based on the work of Scharff and Amelin (2016). There are several reasons as to why intraday trading can be considered beneficial by market participants. Firstly, it acts as a way to reduce imbalance costs to which market participants are subject to when supplying more or less electricity than previously scheduled. These imbalance costs can be an important incentive for all market participants to forecast production and consumption as accurately as possible as well as to trade based on these forecasts. Reducing imbalance volumes also helps to hedge against the uncertainty of the imbalances prices, which might be significantly less favourable than day-ahead prices.

Secondly, market participants are motivated by the possibility to optimise their own supply and demand schedules, e.g. by buying energy to cut down generation in their own power plant that would be more costly to run otherwise.

Finally, intraday trading can also be utilised to offer flexibility in own production or consumption to other market participants who are willing to pay more relative to the costs of running and rescheduling of the power plants.

The benefits of intraday trading from the power grid's perspective is that it can reduce the volume of activated balancing services. For example in such a case when errors related to variable renewable energy, such as wind power generation, can be mitigated shortly before the delivery hour. Here, intraday trading can be helpful because wind power forecasts updated on the day of the delivery hour are on average more accurate than the forecasts made the day before.

3 Data

This section gives an overview of all of the data used in this thesis, such as its original format and pre-processing steps required for turning it into suitable format for model development. Furthermore, some of the most interesting aspects of the data set are visualised and discussed in Chapter 3.4.

All data collection, manipulation and exploration was done using R software. Scripts developed for data collection and pre-processing have been presented in Appendix 2 and code for data exploration and visualisation is shown in Appendix 3.

3.1 Data collection and filtering

All of the necessary market data was provided by Nord Pool through its FTP server. The aim of data collection is to create a set of time series of hourly resolution for each of the model variables from 1st of January 2016 to 31st of December 2020. Including two leap years, this makes up a total of 43848 hours (observations) over 5 years of data. However, since some variables of the model require past information to be used, such as the autoregressive terms or forecast errors, the data collected actually starts from 25th of December 2015. A full explanation of the variables derived from data is provided in Chapter 4.6.

Due to the geographical focus of the thesis, data relevant only to the bidding areas of EE, LV, LT, FI, DK1-DK2 and SE1-SE4 is collected. Note that Norwegian areas (NO1-NO5) have been intentionally left out as they differ significantly from other areas in terms of its power generation mix, which is exceptionally reliant on hydro energy. As such, it is believed that the set of variables considered in this thesis is not suitable for modelling Norwegian intraday prices.

As a very first step, all relevant files and folders were simply copied from the server

to a local computer. Then, R scripts were developed to read data correctly into the R environment and filter out relevant data. The following data was collected:

- Elbas ticker data. This data is provided as daily CSV files and each file contains all intraday transactions between all Nord Pool bidding areas on a continuous basis for a given day. Each transaction contains the trade time, product code, price, quantity, currency, both sides' bidding area (i.e. buy and sell areas) and whether the transaction was cancelled or not. The structure of Elbas data actually changes mid-2018 to also allow for quarter-hourly products being traded, which has to be taken into account. Firstly, Elbas data is filtered to only contain data on hourly products and non-cancelled trades. Secondly, transactions are filtered such that one of the sides (i.e. buyer or seller) belongs to one of the bidding areas under focus. Then, for each transaction, the transaction time, price (EUR/MWh), quantity (MWh), buyer area, seller area, delivery hour and delivery date (derived from product code) is extracted.

- Operating data. This data is provided as weekly SDV[1] files for each of the relevant countries separately on an hourly basis. Operating data is filtered to contain the following variables for each available bidding area[2] and for each delivery hour:
  - Total consumption (MWh)
  - Day-ahead consumption prognosis (MWh)
  - Total production (MWh)
  - Day-ahead production prognosis (MWh)
  - Settled wind production (MWh) excluding LT, FI, SE1-SE4

[1]SDV files in this context are similar to CSV files, but data points are separated with semicolons. SDV files can be opened with a simple text editor software.

[2]For some areas, wind data was too sparse to impute the missing values and was omitted completely for the area. Excluded areas are indicated below.

  - Day-ahead wind production prognosis (MWh) excluding LT, FI

- Elspot prices. Elspot data i.e. the day-ahead hourly prices are provided as weekly SDV files, 52-53 files per year. Whereas the files contain prices in all local currencies as well, only prices in terms of EUR/MWh and for the relevant bidding areas are collected. In addition, day-ahead system prices are also collected.

- Regulating data. This data is provided similarly to Elspot prices, as weekly SDV files. However, regulating data is only available for the Nordic price areas (FI, DK, SE) due to reasons outlined in Chapter 2.3. Regulating data contains the following variables for each available bidding area and for each delivery hour:
  - Down-regulating price (EUR/MWh)
  - Up-regulating price (EUR/MWh)
  - Imbalance settlement price for consumption (EUR/MWh)
  - Imbalance purchase price for production (EUR/MWh)
  - Imbalance selling price for production (EUR/MWh)
  - Dominating regulation direction (1 = Up; -1 = Down; 0 = no regulation)

3.2 Pre-processing

As a first step, after reading in all the necessary data, it was necessary to convert everything to a tidy format, meaning there is only one row for each observation (each delivery hour). This is important for being able to use R's built-in libraries for creating LASSO-estimated models. Before this can be done for the continuous Elbas data, it has to be aggregated to an hourly level. The method of aggregation is explained in Chapter 3.3.

Secondly, when dealing with hourly data, there is often an issue with the practice of daylight savings time. In this context, it means that usually in March, when clocks are turned one hour forward, there will be a missing hour, so data is available for only 23 hours of that day. On the other hand, usually in October clocks are turned one hour back and there will essentially be 25 hours in a day. To tackle this issue, the example of Hinman and Hickey (2009) has been followed. The missing values in spring have been interpolated by taking the average of the two neighbouring values, whereas the "extra" hour in autumn is omitted. In case of further missing values in predictor variables, related to the quality of data, linear interpolation has been applied [3].

As a further note, it is worth mentioning that no outliers have been removed from the data. Short-lived and generally unanticipated price spikes can be considered a unique and important characteristic of the electricity market (Weron, 2014). Hence it is important to train robust models that are as able to predict these spikes as possible.

As a last step, all of the variables have to be merged into a single data structure in R, which has 43848 rows. After this starts the model building process.

3.3 VWAP calculation

As most likely several or even hundreds of Elbas trades have been settled for each hour of power delivery, a volume-weighted average price (VWAP) has to be calculated as a measure of intraday price for a given delivery hour in the data set. Following the example of Kolberg and Waage (2018), VWAP for a given delivery hour at time t is defined as

[3] As an exceptional case, day-ahead total production and wind prognoses for EE are missing for the entire day of 24.12.2019. Here it is assumed that the prognoses are equal to the actual values for these 24 hours.

$$VWAP_t = \frac{\sum_{i=1}^{n}(P_{i,t} \cdot V_{i,t})}{\sum_{i=1}^{n} V_{i,t}},$$

where $P_{i,t}$ is the price (EUR/MWh) and $V_{i,t}$ is the volume (MWh) for trade i for the delivery hour corresponding to time t. The total number of trades for time t is n.

As mentioned in Chapter 2.2, the Elbas market opens at 14.00 CET the day before delivery. Since the objective of this thesis is to predict the aggregate VWAP for the 4 hours preceding some delivery hour, it means that some trades for that hour have likely already taken place. This is valuable information and can be used as one of the predictor variables. Therefore, VWAP is split into near-VWAP and far-VWAP based on the decision point in time (when the prediction is made), which is 4 hours before the start of the delivery hour at time t. The output variable that is being forecasted, is near-VWAP. This approach is similar to Kolberg and Waage (2018), but with 4 hours as the dividing point instead of 6 hours before the start of delivery hour.

For an example, let's say we want to predict the near-VWAP of delivery hour 10, i.e. 09:00-10:00 today. The prediction for this is made at 05:00 and we aim to predict the volume weighted average price of the trades made during the period 05:00-09:00. As one of the predictor variables, far-VWAP can be used, which is based on trades made from 14:00 day before until 05:00 today. See Figure 2 for an illustration of this example.

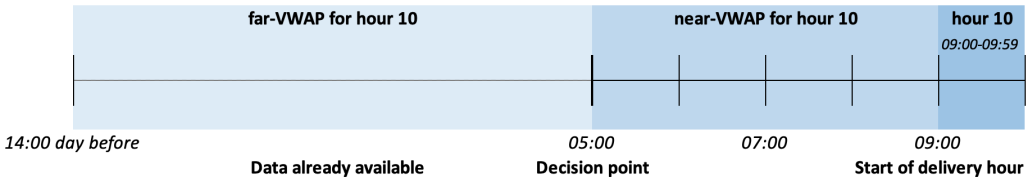

Figure 2: Illustration for the example of predicting near-VWAP for hour 10.

However, in some cases, it is also possible that no trades have taken place for some delivery hour, either in the far-VWAP period, near-VWAP period or both periods. In those situations, linear interpolation is used to derive the missing values, based on the two adjacent values of the corresponding variables for both the previous and next delivery hour. The fact that an aggregate near-VWAP is being predicted for the entire Nordic-Baltic region helps to mitigate this issue to a manageable level. In fact, the initial goal of the thesis was to attempt to predict near-VWAP for each bidding area separately, but the low volume of trades, especially in the years 2016-2018 put a stop to that idea.

3.4 Data exploration

As a first step, data from the continuous Elbas market is explored. The intraday trade price development throughout the day is perhaps best illustrated with an example of a single delivery hour on a random day in the data set. Figure 3 displays every trade with its price in EUR/MWh for delivery hour 22 (21:00-22:00) on 08.09.2019.

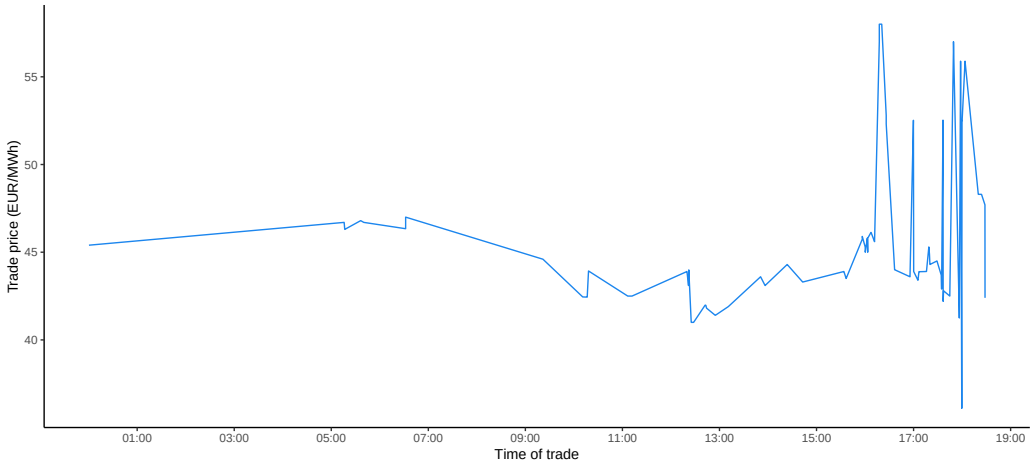

Figure 3: Elbas trades for delivery hour 22 of 08.09.2019.

Note that the trading activity seems to be lower in terms of number of trades when there is more time until the delivery hour. As time approaches the delivery

hour, trading activity increases due to need to eliminate unforeseen imbalances and trade prices seem to become more volatile. While this is an example of just a single observation out of 43848 hours in the data set, similar characteristics can be observed in case of other data points. This is one of the reasons why this type of continuous Elbas trade data is transformed to a time series of volume-weighted average price for each delivery hour in the data set.

Variability in trade price can be explained both by the inherent difference in price levels across bidding areas and by the fact that sudden spikes in electricity prices are rather common. The difference in mean trade prices across bidding areas is further illustrated on Figure 4. It can be observed that the average trade price tends to be higher in the Baltic bidding areas and lower in the Nordic areas with the largest difference being almost 17 EUR/MWh on average between LV and SE2.

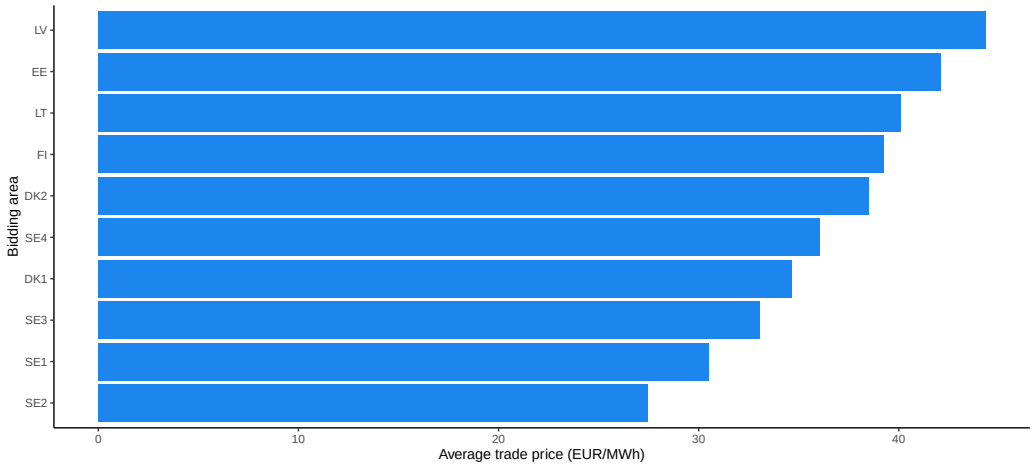

Figure 4: Average price per buying area in EUR/MWh

By looking at Figure 5, one can observe that throughout the observed years, trading activity on the Elbas market has increased both in terms of total number of trades and volume traded. While volume has been on a moderate uptrend, number of trades have multiplied across the years. This is likely due to increasing importance of the Elbas market and the fact that since June 2018, Elbas intraday market was extended to additional European areas besides just the Baltic and Nordic countries.

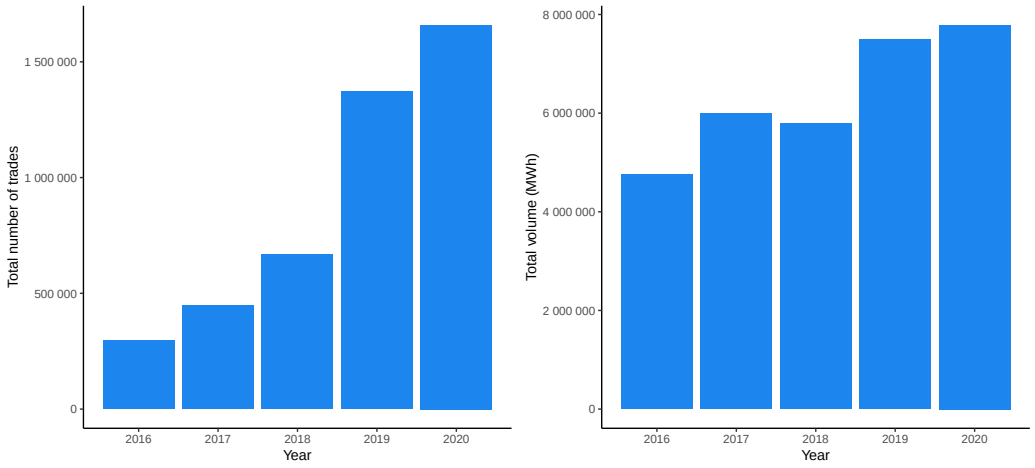

Figure 5: Left: Total number of trades in each of the years of the data set. Right: Total volume in MWh in each of the years.

In further analysis of the different bidding areas under consideration, both buy and sell volume is visualised by bidding area on Figure 6. Finland can be identified as one of the most active counterparties by volume in Elbas trading as it has bought the largest volume and sold the second largest volume of electricity in MWh. Baltic areas are among the smaller counterparties as naturally their economies and population consumes and produces less energy. Note that Elbas data under consideration can contain other buying or selling bidding areas as well, such as Germany. However, the focus is on the Baltic and Nordic countries and hence other areas have been excluded from the figures.

Next, it would be interesting to gain insight on which of the 24 daily delivery hours are most actively traded on the Elbas market. This has been illustrated on Figure 7. It can be seen that the number of trades tends to be lower for the delivery hours of early morning and starts to ramp up starting from delivery hour 9, which roughly corresponds to the start of regular office hours. The peak is reached at delivery hour 17, which roughly corresponds to the end of regular office hours.

However, all of the intraday trades on the Elbas market will be used to construct an aggregate measure of intraday price for the entire region during the 4 hours

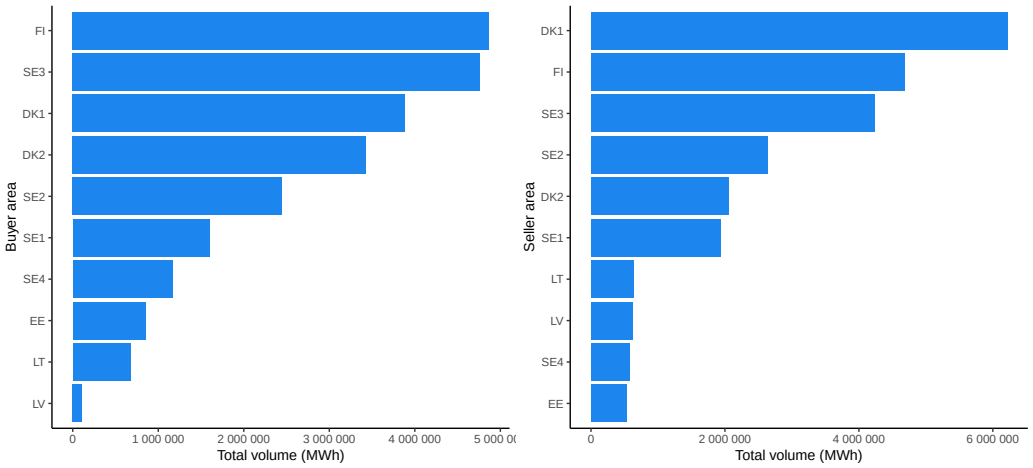

Figure 6: Left: Volume (MWh) bought on Elbas by bidding area. Right: Volume (MWh) sold on Elbas by bidding area.

prior to the delivery hour, which is the near-VWAP. One of the distinguishing characteristics of electricity prices and therefore, also near-VWAP, is strong daily seasonality. As an example, Figure 8 illustrates the development of near-VWAP during a randomly chosen week from the data set. It can be observed that each day, at least during the workdays, there seems to be 2 separate peak periods, of which the first one roughly coincides once again with the start of regular office hours and second one with the end of office hours.

The existence of these two daily peaks is further confirmed on Figure 9, which displays the average near-VWAP for each of the 24 delivery hours. Higher near-VWAP is associated with higher demand and vice-versa. On average, near-VWAP is highest for delivery hour 9 and lowest for delivery hour 4.

Lastly, a summary of descriptive statistics for near-VWAP is presented in Table 1. As will be explained in Chapter 4.2, the data set is divided into 3 parts - initial calibration set, validation set and test set. Summary statistics are presented for each of the data sets separately.

By far the biggest outlier of 701.77 EUR/MWh can be found in the test set. On

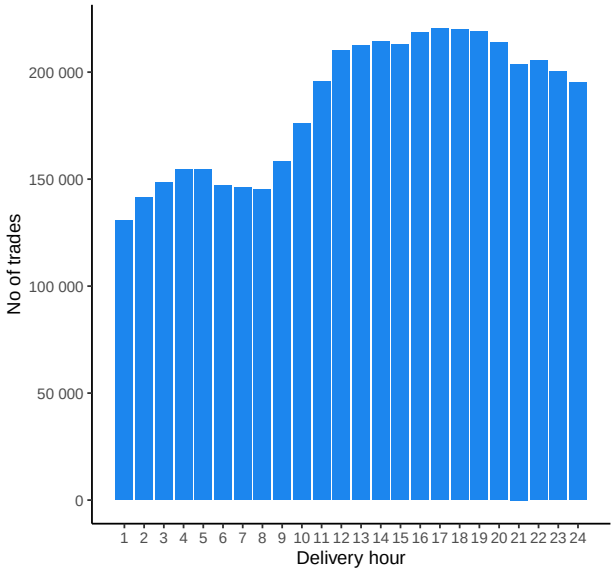

Figure 7: No of trades regarding each delivery hour.

further inspection, this outlier belongs to delivery hour 20 on 15.09.2020, following unexpected outages of several Swedish nuclear stations. Negative prices can be a result of increased share of cheap and variable renewable energy in the energy mix of the Baltic and Nordic markets, the lowest near-VWAP across the data sets being -48.09 EUR/MWh.

Table 1: Summary statistics for near-VWAP

Data set	**Min**	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**SD**	**NA's**
Initial calibration set	-22.12	22.12	28.09	29.71	34.92	221.60	12.55	11
Validation set	3.21	27.35	30.12	30.55	33.13	72.65	7.28	0
Test set	-48.09	25.86	35.42	35.82	45.35	701.77	17.43	56

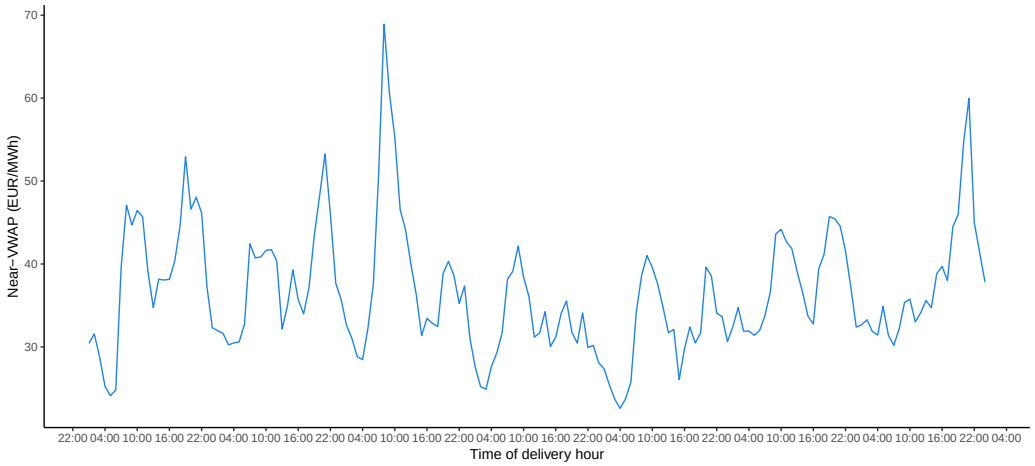

Figure 8: Near-VWAP (EUR/MWh) over a course of week from the first delivery hour of 02.09.2019 until the last delivery hour of 08.09.2019.

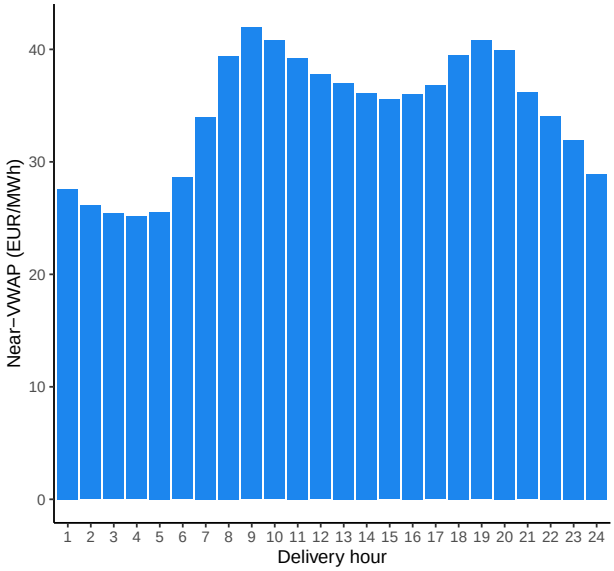

Figure 9: Average near-VWAP (EUR/MWh) per delivery hour.

4 Methodology

In this section the methodology used for forecasting intraday electricity prices will be introduced. Chapter 4.1 discusses two different commonly used frameworks for modelling electricity prices. Chapter 4.2 explains the rolling window scheme that is used for model calibration. Chapters 4.3 and 4.4 discuss LASSO regression and the method of choosing its complexity parameter. Chapter 4.5 introduces the error metrics used for model training and comparisons to benchmark models, which are discussed in Chapter 4.7. Chapter 4.6 describes all of the explanatory variables used in the LASSO-estimated models.

All model building and evaluation was done using R software. The relevant R scripts are presented in Appendix 4.

4.1 Univariate vs multivariate framework

In EPF literature, there exist two main methods for the representation of the price series – univariate and multivariate frameworks[4]. Modelling implemented in a multivariate fashion consists of separate models for each of the 24 delivery hours, whereas within a univariate framework, one large model is constructed to produce forecasts for each delivery hour using the same set of parameters (Ziel and Weron, 2018). Based on Ziel and Weron (2018), the multivariate framework can be formulated as

[4]Note that *univariate* and *multivariate* in this context is different from univariate and multivariate regression analysis as described in the more classical regression literature. In this thesis, whenever univariate or multivariate models or modelling is discussed, it is in the context of models set in the univariate or multivariate framework as defined in the EPF literature and Section 4.1.

$$\begin{cases} y_{d,1} = f_1(x_{d,1,1}, x_{d,1,2}, \ldots, x_{d,1,p}) + \varepsilon_{d,1}, \\ \quad \vdots \\ y_{d,24} = f_{24}(x_{d,24,1}, x_{d,24,2}, \ldots, x_{d,24,p}) + \varepsilon_{d,24}, \end{cases}$$

where $\varepsilon_{d,h}$ is the innovation (noise) term for day d and delivery hour h, $f_h(\cdot)$ are some functions of the explanatory variables $x_{d,h,j}$ and p is the total number of explanatory variables used. The j^{th} predictor belongs to a set of features introduced in Chapter 4.6.

The univariate framework is defined (Ziel and Weron, 2018) as

$$y_t = f(x_{t,1}, x_{t,2}, \ldots, x_{t,p}) + \varepsilon_t,$$

where ε_t is the innovation term at time t, and $f(\cdot)$ is some function of the explanatory variables $x_{t,j}$. Time t can be defined as $t = 24d + h$.

The fact that each load period (delivery hour) tends to display a rather distinct price profile, reflecting the daily variations in demand, supply, costs and operational constraints speaks to the advantage of the multivariate model. However, the disadvantage can be that the estimated set of models in the multivariate framework might not take into account the potentially important dependencies between the variables across different delivery hours. (Ziel and Weron, 2018)

Ziel and Weron (2018) perform an extensive empirical study on the two frameworks' predictive abilities and argue that the results are inconclusive – the multivariate models do not uniformly outperform the univariate models across all data sets, seasons of the year or hours of the day, and is sometimes outperformed by the latter. Hence, it is one of the objectives of this thesis to implement models both in the univariate and multivariate framework and compare their forecast accuracy in

the context of Nordic-Baltic intraday electricity market.

4.2 The rolling window scheme

Following the example of many research articles [5] on electricity price forecasting, a rolling window model calibration scheme is implemented. To account for seasonality in data, Uniejewski and Weron (2018a) advise that the model calibration window length should be a multiple of the weekly and annual periodicities, such as 364 or 728 days, corresponding to 1 and 2 years worth of data respectively. In this thesis, a 364-day rolling window scheme and been chosen, such as in Marcjasz, Uniejewski, and Weron (2020).

As a first step, the first 364 days of the data set is defined as the initial calibration window (see Figure 10). In this case, data from first hour of 01.01.2016 to last hour of 29.12.2016 is used for fitting the first model (models in case of multivariate structure). Whether this corresponds to 364 hours of data or $24 \cdot 364 = 8736$ hours of data depends on whether the model is univariate or multivariate. Using the fitted models, predictions for the 24 hours of 30.12.2016 are made. Next, the window is rolled forward by one day, all models are re-estimated and the next set of 24 predictions are made for 31.12.2016. This procedure is repeated until all forecasts in both the validation set and test set have been made. The validation set is used for model tuning and will be explained further in Chapter 4.4. Out-of sample test set will be used for calculating error metrics (Chapter 4.5) for measuring the model's predictive abilities.

4.3 LASSO regression

LASSO (least absolute shrinkage and selection operator) regression is a regularization method that shrinks the regression coefficients by imposing a penalty on their

[5] Uniejewski, Nowotarski, and Weron (2016), Narajewski and Ziel (2020), Marcjasz, Uniejewski, and Weron (2020).

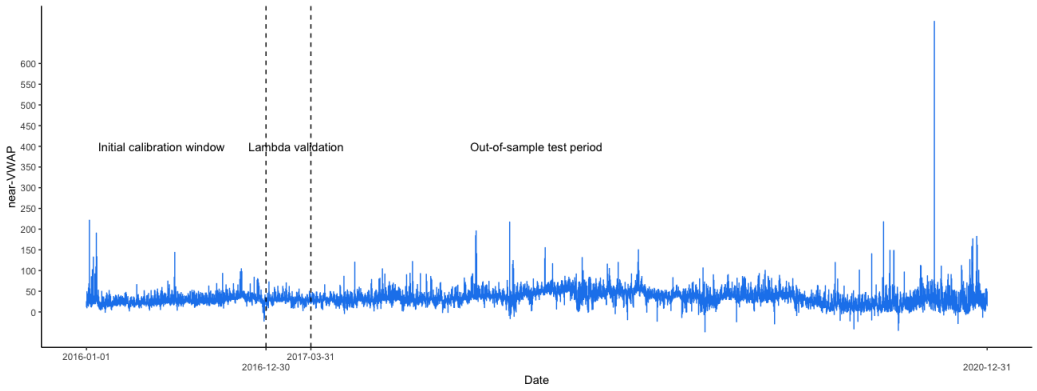

Figure 10: Initial calibration window, λ validation window and out-of-sample window

size. Hastie, Tibshirani, and Friedman (2009) define the LASSO estimate of model coefficients as

$$\hat{\beta}^{lasso} = \arg\min_{\beta}\{\frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\},$$

where N is the total number of observations in the data set.

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk toward zero. However, in the case of the LASSO, the penalty term has the effect of forcing some of the coefficient estimates to be exactly equal to zero when λ is sufficiently large. Hence, the LASSO performs variable selection. (Hastie, Tibshirani, and Friedman, 2009)

Before applying regularization methods, James et al. (2013) recommend standardising the predictors using the formula

$$\tilde{x}_{i,j} = \frac{x_{i,j}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \bar{x}_j)^2}},$$

where $x_{i,j}$ is the i^{th} value of the j^{th} predictor. The aim of this is for all the predictors to have unit variance and all be on the same scale.

LASSO models are developed using the *glmnet*[6] package in R. Note that the *glmnet* package performs predictor variable standardisation by default.

4.4 Choice of LASSO parameter λ

For choosing the optimal LASSO hyperparameter λ, a commonly used setup in machine learning literature is considered. Namely, the data set is divided into training set (i.e. the initial 364-day calibration window introduced in 4.2), validation set (91 days or roughly quarter of a year) and test set (1372 days) as illustrated on Figure 10. The validation set starts on 30.12.2016 and ends with the last hour of 30.03.2017. Multiple LASSO models are fit, each given a different λ parameter from a predefined grid of values as their performance is evaluated on the validation set.

Depending on the modelling framework, two different grids of exponentially increasing parameter λ values are considered. In case of univariate framework, the λ grid is defined as $\lambda_i = 10^{-\frac{31-i}{6}}, i = 1, 2, \ldots, 31$, resulting in 31 values on a log-scale ranging from 10^{-5} to 10^{0}. In case of univariate framework, the λ grid is defined as $\lambda_j = 10^{-\frac{10-j}{6}}, j = 4, 5, \ldots, 19$, resulting in 16 values on a log-scale ranging from 10^{-1} to approximately 31.6.

The reasons for this difference of grid length and magnitude of values is mainly due to limitation of computational resources. Firstly, the validation of multivariate framework models was found to be quite a bit more time-consuming, hence a shorter grid of 15 values is used. Secondly, by initial trial and error, it was found that the best-performing multivariate models (on the validation set) tend to use larger λ values.

[6] https://web.stanford.edu/~hastie/Papers/Glmnet_Vignette.pdf

For each corresponding grid value, RMSE (see Chapter 4.5 for more information on error metrics) is calculated for the entire validation period. In the univariate framework, a single optimal λ is chosen for all 24 delivery hours based on the lowest validation set RMSE. However, in the multivariate framework, it is possible to find an optimal λ value for each of the 24 delivery hours since each hour is modelled separately. These λ parameter values will then be used in the rolling window modelling scheme for making forecasts in the test set.

While choosing the window length and method for the calibration of λ, computational speed has to be weighed against the possible gains in accuracy. Uniejewski and Weron (2018b) argue that it is important to select a period of at least 60 days for the calibration process. A 91-day validation window confirms to this rule of thumb and also follows the example of Uniejewski, Nowotarski, and Weron (2016), who used the same window length and validation set approach as this thesis. An alternative for finding a fixed value of λ for the entire out-of-sample test set would be to recalibrate the parameter daily throughout the test period, which according to Uniejewski and Weron (2018b) does offer improved accuracy, but comes with heavy and perhaps impractical computational requirements.

4.5 Error metrics

In order to choose optimal values for the parameter λ and make comparisons between different models, it's important to define some error metrics. In the field of EPF, the most widely used metrics to measure the accuracy of point forecasts are the mean absolute error (MAE) and the root mean square error (RMSE) (Lago et al., 2021).

MAE and RMSE for the univariate framework are defined as

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

$$RMSE = \sqrt{\frac{1}{N_h} \sum_{i=1}^{N_h} (\hat{y}_i - y_i)^2},$$

where y_i and $\hat{y}_i$ are respectively the observed and forecasted price for the hour i, N_h is the total number of hours in the data set under observation.

For the multivariate framework, Lago et al. (2021) define MAE and RMSE as

$$MAE = \frac{1}{24N_d} \sum_{d=1}^{N_d} \sum_{h=1}^{24} |\hat{y}_{d,h} - y_{d,h}|,$$

$$RMSE = \sqrt{\frac{1}{24N_d} \sum_{d=1}^{N_d} \sum_{h=1}^{24} (\hat{y}_{d,h} - y_{d,h})^2},$$

where $y_{d,h}$ and $\hat{y}_{d,h}$ are respectively the observed and forecasted price for delivery hour h ($h = 1, \ldots, 24$) on day d, and N_d is the number of days in the data set under observation.

In this thesis, RMSE (lower RMSE is desirable) is taken as the main criterion in the model training process and comparison. RMSE is known to penalise bigger errors more severely, hence, it is hoped that RMSE-based decisions would result in more robust models that are able to handle the volatile nature of the electricity market better than its MAE-based counterparts.

However, a case could also be made for using MAE for training the models, if the model's user has access to some kind of expert knowledge at times of unforeseen price spikes or outages and could therefore rely on that, rather than model pre-

dictions. Furthermore, MAE can be considered more interpretable for forecasting users and according to Lago et al. (2021), is a more accurate representation of the underlying problem in most electricity market applications. For these reasons, MAE is also calculated and presented for the reader of this thesis.

4.6 Choice of explanatory variables

One of the main advantages of using LASSO is that a potentially very large set of predictors could be considered as it is able to perform feature selection.

Firstly, features to capture the daily and weekly seasonal effects on intraday prices are included – dummy variables for weekday $(1, \ldots, 7)$ and delivery hour $(1, \ldots, 24)$. This can be useful as each delivery hour tends to display a distinct profile in terms of price, consumption and production. Furthermore, day of the week can play a role as for example consumption profiles of workdays and weekends can differ significantly. Note that the delivery hour dummy variable is only required for the univariate model as each multivariate model only considers data for the same delivery hour.

Next, from the operating data (described in Chapter 3.1), last available errors (i.e. the difference between day-ahead prognosis and realised value) and prognoses of total production, wind production and consumption have been calculated and added as model features (up to 6 features per bidding area, 52 in total). Note that last available for errors here means the errors inherently have a 5-hour information lag, from the moment when the prediction in made until the last second of the delivery hour, which is forecasted. While for example the production prognosis has been made available the day before, the realised production value is not available at the time of decision - that is 5 hours prior to the end of that delivery hour. Therefore, for instance if we want to predict near-VWAP for delivery hour 12 (11:00-12:00), the last available production error is for delivery hour 7 (06:00-07:00), made available at 07:00am. It must be mentioned that this kind of lag

assumes almost instantaneous exchange of information and might be too short of a delay in practice. Depending on the actual speed of data exchange, this lag would have to be adjusted in a real-world setting. While the error is delayed, it is hoped that it acts as a proxy for the general level of quality of weather, production and consumption forecasts for the actual delivery hour.

For the Nordic bidding areas, all 6 variables of regulating data (as described in Chapter 3.1) have been added as well, which have to be delayed in a similar manner to errors of operating data for the same underlying reasons. This makes up a total of 42 input variables in addition.

Furthermore, day-ahead prices, determined at the Elspot auction on the previous day, should be a valuable source of information. Since the output variable near-VWAP is an aggregate volume-weighted price over several bidding areas, day-ahead prices for all the relevant areas have been included as predictor variables. In addition, the system price (theoretical equilibrium price in case of no congestion) is included, which makes up a total of 11 new variables.

Finally, and perhaps most importantly, 7 additional features are added to include information about past and last available near-VWAP as well as the available information about already settled intraday trades (at the time of prediction) for a given delivery hour. Past near-VWAP information is introduced as two autoregressive terms, at $d-1$ and $d-7$ (near-VWAP same hour one day ago and one week ago). Following the same logic of lagged errors of operating data, 5-hour lagged near-VWAP values and total transaction volume of the same period have also been included. Regarding the already settled trade information, far-VWAP and total volume during the far-VWAP period are included as predictors. But in addition to far-VWAP, a feature called latest-VWAP is engineered as well. The far-VWAP period can be up to 29 hours long, given the Elbas market opens at 14:00 the day before and the end of the far-VWAP window for delivery hour 24 is at 19:00

pm. However, it is reasonable to believe that as we move closer to the delivery hour in time, intraday transaction prices become more representative of the true near-VWAP value. Latest-VWAP aims to capture that information as it is the volume-weighted average price of the hour closest[7] to (but before of) the time of prediction. In some cases of low trading volume, it is possible that far-VWAP and latest-VWAP are in fact the same, but mostly new information is introduced.

Therefore, in total as much as 114 predictor variables are fit in the LASSO-estimated models. A comprehensive overview of all the model variables in a form of a table is provided in Appendix 1.

4.7 Simple benchmark models

To assess whether the use of this parameter-rich LASSO-estimated model is justified in practice, it should be compared to some simple benchmark models that require little time and resources to develop to see whether significantly improved forecasting accuracy is achieved.

As perhaps the most logical choices of such benchmark, values of far-VWAP and latest–VWAP variables for some delivery hour at time t could be considered as naive forecasts for the same hour's near-VWAP. For a comprehensive comparison, SE3 day-ahead price[8], 5-hour lagged near-VWAP and $d-1$ near-VWAP are also considered as simple benchmark models. The results are discussed in Chapter 5.1.

[7] To tackle the issue that some hours of intraday trading do not have any trades at all regarding some delivery hour, VWAP is calculated separately for each of the 5 hours prior to delivery hour, latest hour taking priority.

[8] SE3 day-ahead price was found to be the best simple benchmark in terms of MAE and RMSE in the work of Kolberg and Waage (2018).

5 Results and discussion

This section presents the results of benchmark models and LASSO models set in both univariate and multivariate framework, followed by discussion on forecast performance and feature selection performed by the LASSO models.

5.1 Benchmark model results

For making conclusions about forecasting accuracy, it is usually most important to compare model performance on completely unseen data, i.e. the test set. However, looking at validation error metrics can also provide useful information, hence MAE and RMSE are presented for both the validation set and test set in Table 2.

Table 2: Simple benchmark models results

	Validation		**Test**	
Benchmark	**MAE**	**RMSE**	**MAE**	**RMSE**
Far-VWAP	3.118	4.791	4.355	8.088
Latest-VWAP	3.288	5.947	4.507	8.231
SE3 day-ahead price	3.401	5.452	5.825	10.408
$d-1$ near-VWAP	5.419	7.702	8.961	15.034
$h-5$ near-VWAP	5.616	8.053	9.998	16.031

It turns out that in terms of test set RMSE, far-VWAP performs the best with MAE of 4.36 EUR/MWh and RMSE of 8.09 EUR/MWh. Second best, with comparable results, is the latest-VWAP benchmark. The fact that these two models perform the best, is in fact expected as they both are essentially versions of a naive[9] forecast for near-VWAP. By design, far-VWAP contains information about a larger number of trades and might be the reason as to why it seems to outperform latest-VWAP

[9]i.e. the most recent intraday price, which was introduced as one of the most important variables in Chapter 1.3.

by a small margin. The rest of the benchmark models demonstrate significantly lower forecasting accuracy in the test set. Moving forward, far-VWAP benchmark is chosen as the one to beat by the more complicated LASSO-estimated models.

In all cases, there is a noticeable difference in validation and test set performances, the error metrics being higher in the test set. This seems to indicate that perhaps the market conditions have become more volatile and simple benchmark models are not able to explain this increased variability.

5.2 LASSO-estimated models results and discussion

MAE and RMSE for both univariate and multivariate frameworks are presented in Table 3.

Table 3: LASSO models results

	Validation		**Test**	
Framework	**MAE**	**RMSE**	**MAE**	**RMSE**
Univariate	2.584	3.955	3.831	6.989
Multivariate	5.162	6.987	8.403	12.709

Optimal λ parameter based on the lowest RMSE on the validation set in the univariate framework is 0.01, i.e. the 19^{th} value on the grid of 31 values as defined in Chapter 4.4. In case of multivariate framework, optimal λ parameters were found for each of the 24 delivery hours, which are presented in Appendix 5. The most common λ across the hours is found to be approximately 4.64, i.e. the 11^{th} value on the grid of 16 values.

By looking at the error metrics, it is immediately clear that the univariate framework outperforms the multivariate framework by a considerable margin as its RMSE is 6.99 EUR/MWh, compared to multivariate's RMSE of 12.71 EUR/MWh.

Forecasting accuracy of the univariate framework is also superior to the best benchmark model in both the validation and test set and in terms of both MAE and RMSE. The improvement of the best performing LASSO model over the far-VWAP benchmark is 0.524 EUR/MWh in terms of MAE and 1.099 EUR/MWh in terms of RMSE, which is a roughly 13.6% decrease in RMSE. It can again be observed that the error metrics of the validation set are lower than in the test set, which is expected as the parameter λ is chosen such that it would minimise RMSE in the validation set, whereas the test set comprises of completely unseen data.

However, such a low level of performance of the multivariate framework is surprising and quite disappointing – it is even outperformed by 3 of the 5 considered benchmark models. Given this level of forecasting accuracy, it can be said that this framework in its current form is not suitable for use in practice. In fact, the results seem to indicate that it is a severe case of overfitting on the validation set. This is most likely due to the combination of several factors. Firstly, a much smaller number of observations is used to train each model[10], whereas the number of predictor variables (114) is relatively high. Some of these predictors are categorical, so the actual number of model coefficients to be estimated is even higher. Secondly, the multivariate framework was allowed to fit much larger values of λ due to the fact that these larger values were observed to achieve better validation set results. However, as it turns out that, these very large λ values diminished nearly all model coefficients to zero and achieved very poor test set results. Thirdly, it is likely that the validation set, which only covers the first quarter of 2017, has become less representative of the more recent times, i.e. the test set, due to changes in the market environment.

To test out whether the reasoning behind overfitting holds true, another approach in the implementation of the multivariate framework is considered. As a first step,

[10]Remember that 364 observations are used to train models in the multivariate framework compared to 8736 in the univariate framework.

20 of the most important explanatory variables in the univariate framework are determined, which is explained more in-depth in Chapter 5.3. Then, LASSO models are fit under multivariate framework using the same methodology as previously, but with two key differences - only these 20 most important features instead of the entire set of 114 features are used and a vector of smaller λ parameters is defined for the validation process. This new vector is defined as $\lambda_k = 10^{-\frac{19-i}{6}}, k = 1, 2, \ldots, 15$, resulting in 15 values on a log-scale ranging from 10^{-3} to approximately 0.22. In terms of magnitude of the values, they are similar to the ones originally defined for the univariate framework.

It turns out that these two adjustments greatly improve the forecasting results of the multivariate framework in the test set. In the validation set, MAE of 6.435 EUR/MWh and RMSE of 8.696 EUR/MWh is achieved, while the test set MAE is 3.917 EUR/MWh and RMSE is 7.178 EUR/MWh. The most commonly chosen λ is approximately 0.22, the full table is provided in Appendix 5. So while multivariate's test set results are still slightly worse than univariate framework results, they are now comparable and manage to beat all benchmark models. However, what is most curious in this case is that the validation set errors seem to be larger than those of the test set. Perhaps this confirms one of the initial suspicions, which is that the validation set and test set represent market conditions that have become too different. As a possibility for future research, a different approach to choosing the appropriate λ parameter could be considered.

While the LASSO model in the univariate framework can be considered best overall, it could also be interesting to evaluate performance across each of the daily 24 delivery hours. MAE and RMSE error metrics of the test set have been compared for the best baseline model (i.e. far-VWAP benchmark), univariate LASSO model and the best multivariate LASSO model per delivery hour on Figure 11.

As it can be seen, errors seem to be proportional to the average level of near-VWAP

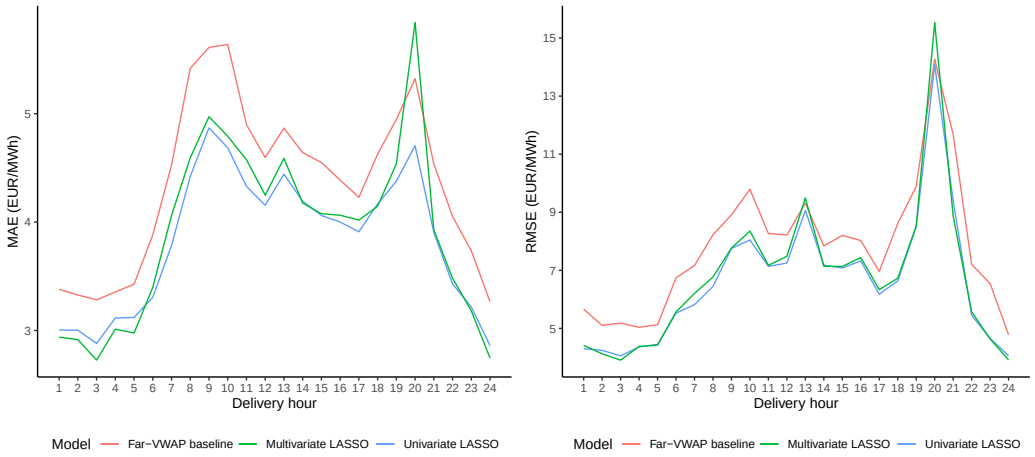

Figure 11: MAE and RMSE of the test set for each of the daily 24 delivery hours.

as illustrated on Figure 9. Furthermore, it looks as if the multivariate framework can perform slightly better in some very early morning (hours 1–5) or late night hours (hours 23-24), but in other hours, univariate is consistently more accurate. As further research, it could be interesting to experiment combining predictions for different delivery hours from both frameworks as a final forecast. As an additional note, it must be mentioned that the extreme outlier of near-VWAP, identified in Table 1, seems to have quite a significant effect on the entire RMSE of delivery hour 20.

5.3 Variable selection

As the test set consists of 1372 days and the LASSO model is calibrated daily in the univariate framework, 1372 models have been fitted as well. Each model can perform feature selection, so by looking at the most frequently used variables across the test set, one could gain important insight on the most important variables for predicting intraday prices on the Nord Pool. Top 10 of the most frequently used variables by the univariate LASSO model, accompanied by the number of occurrences, are the following:

- Wind production error of DK1: 1372
- Production error of SE2: 1372
- Latest-VWAP: 1372
- FI day-ahead price: 1372
- Far-VWAP: 1372
- $h - 5$ near-VWAP: 1370
- $d - 1$ near-VWAP: 1370
- Day-ahead wind production prognosis of LV: 1369
- Production error of SE1: 1367
- Imbalance price for consumption of DK1: 1358

It can be seen that there are 5 variables, which were included in the LASSO model for each and every one of the days in the test set. The fact that wind production error of DK1 was one of them makes sense - DK1 is one of the largest wind power generating areas on the Nord Pool and variable wind energy and its forecast errors has been described as one of the main factors for the increasing importance of intraday trading. Additionally, both of the variables used in the two best performing benchmark models are always included, i.e. latest-VWAP and far-VWAP.

On the other hand, top 10 of the least frequently used variables by the univariate LASSO model with the number of occurrences are the following:

- Down-regulating price of DK1: 282
- Down-regulating price of SE2: 291
- Imbalance selling price for production of SE2: 298

- Up as the dominating regulating direction of SE2: 311
- Up-regulating price of FI: 344
- Up as the dominating regulating direction of SE1: 415
- Day-ahead price of SE2: 482
- Down-regulating price of FI: 523
- Up as the dominating regulating direction of SE4: 544
- Imbalance settlement price for consumption of SE2: 555

It turns out that 9 out of 10 least used features belong to regulating data, pointing to the weakest predictive capabilities of information coming from the regulating market compared to all other sources of information. It is possible that all six types of regulating data are highly correlated with each other and therefore just the use of all of them is excessive.

The full list of variable frequency can be found in Appendix 6.

Conclusions

The recent shift of focus from the more traditional sources of electricity to variable renewable energy has brought along larger price fluctuations and challenges for keeping the system stable at all times and avoiding imbalance costs from the perspective of the market participants. The Elbas intraday market offers an opportunity to adjust commitments made on the day-ahead market and trade electricity very close to the delivery hour. Therefore, reliable and accurate intraday price forecasts can be crucial for market participants for making the best trading decisions.

The objective of the thesis was to provide a practical and robust statistical method for predicting an aggregate volume-weighted average intraday price over the last four hours prior to the delivery hour on the Nord Pool power exchange, limited to Estonian, Latvian, Lithuanian, Finnish, Danish and Swedish price areas. The thesis succeeds in demonstrating how LASSO regression in combination with a large variety of most recent information can be employed to generate forecasts that outperform the naive estimates for the intraday price. Of the two most commonly implemented forecasting structures, the so-called univariate framework, where each of the 24 delivery hours are forecasted based on the same set of model coefficients, has been found to achieve higher forecasting accuracy. Regarding the use of the multivariate framework, several potential pitfalls have been identified and addressed. Furthermore, it is hoped that the reader has been provided with useful insight of the functioning of the Nordic-Baltic electricity market and the most important explanatory variables for predicting intraday prices.

Future research could investigate alternative approaches to choosing an appropriate λ parameter for the LASSO regression or perhaps implement neural network based models known from the deep learning literature in the context of both Baltic and Nordic intraday electricity markets.

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer. URL: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf.

Hinman, Jennifer and Emily Hickey (2009). "Modeling and Forecasting Short-term Electricity Load Using Regression Analysis". In: URL: https://irps.illinoisstate.edu/downloads/research/documents/LoadForecastingHinman-HickeyFall2009.pdf.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An Introduction to Statistical Learning*. 2nd ed. Springer. DOI: https://doi.org/10.1007/978-1-4614-7138-7.

Kolberg, Johannes Krokeide and Kristin Waage (2018). "Artificial Intelligence and Nord Pool's intraday electricity market Elbas : a demonstration and pragmatic evaluation of employing deep learning for price prediction : using extensive market data and spatio-temporal weather forecasts". In: URL: https://openaccess.nhh.no/nhh-xmlui/handle/11250/2560898.

Lago, Jesus, Fjo De Ridder, and Bart De Schutter (2018). "Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms". In: *Applied Energy* 221, pp. 386–405. DOI: 10.1016/j.apenergy.2018.02.069. URL: https://linkinghub.elsevier.com/retrieve/pii/S030626191830196X.

Lago, Jesus, Grzegorz Marcjasz, Bart De Schutter, and Rafał Weron (2021). "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark". In: *Applied Energy* 293, p. 116983. DOI: 10.1016/j.apenergy.2021.116983. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306261921004529.

Marcjasz, Grzegorz, Bartosz Uniejewski, and Rafał Weron (2020). "Beating the Naïve—Combining LASSO with Naïve Intraday Electricity Price Forecasts". In: *Energies* 13.7, p. 1667. DOI: 10.3390/en13071667. URL: https://www.mdpi.com/1996-1073/13/7/1667.

Narajewski, Michał and Florian Ziel (2020). "Econometric modelling and forecasting of intraday electricity prices". In: *Journal of Commodity Markets* 19, p. 100107. DOI: 10.1016/j.jcomm.2019.100107. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405851319300728.

Nord Pool (2021a). *Bidding areas*. URL: https://www.nordpoolgroup.com/the-power-market/Bidding-areas/ (visited on 11/05/2021).

– (2021b). *Day-ahead and intraday power trading*. URL: https://www.nordpoolgroup.com/trading/ (visited on 11/05/2021).

– (2021c). *Intraday market*. URL: https://www.nordpoolgroup.com/the-power-market/Intraday-market/ (visited on 06/04/2021).

– (2021d). *Price formation*. URL: https://www.nordpoolgroup.com/the-power-market/Day-ahead-market/Price-formation/ (visited on 11/05/2021).

– (2021e). *The market members*. URL: https://www.nordpoolgroup.com/the-power-market/The-market-members/ (visited on 11/05/2021).

Scharff, Richard and Mikael Amelin (2016). "Trading behaviour on the continuous intraday market Elbas". In: *Energy Policy* 88, pp. 544–557. DOI: 10.1016/j.enpol.2015.10.045. URL: https://linkinghub.elsevier.com/retrieve/pii/S0301421515301713.

Spodniak, Petr, Kimmo Ollikka, and Samuli Honkapuro (2021). "The impact of wind power and electricity demand on the relevance of different short-term electricity markets: The Nordic case". In: *Applied Energy* 283,

p. 116063. DOI: 10.1016/j.apenergy.2020.116063. URL: https://linkinghub.elsevier.com/retrieve/pii/S030626192031494X.

Uniejewski, Bartosz, Grzegorz Marcjasz, and Rafał Weron (2019). "Understanding intraday electricity markets: Variable selection and very short-term price forecasting using LASSO". In: *International Journal of Forecasting* 35.4, pp. 1533–1547. DOI: 10.1016/j.ijforecast.2019.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207019300123.

Uniejewski, Bartosz, Jakub Nowotarski, and Rafał Weron (2016). "Automated Variable Selection and Shrinkage for Day-Ahead Electricity Price Forecasting". In: *Energies* 9.8, p. 621. DOI: 10.3390/en9080621. URL: https://www.mdpi.com/1996-1073/9/8/621.

Uniejewski, Bartosz and Rafał Weron (Aug. 2018a). "Efficient Forecasting of Electricity Spot Prices with Expert and LASSO Models". In: *Energies* 11, p. 2039. DOI: 10.3390/en11082039.

– (2018b). "Efficient Forecasting of Electricity Spot Prices with Expert and LASSO Models". In: *Energies* 11.8, p. 2039. DOI: 10.3390/en11082039. URL: http://www.mdpi.com/1996-1073/11/8/2039.

Weron, Rafał (2014). "Electricity price forecasting: A review of the state-of-the-art with a look into the future". In: *International Journal of Forecasting* 30.4, pp. 1030–1081. DOI: 10.1016/j.ijforecast.2014.08.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169207014001083.

Ziel, Florian and Rafał Weron (2018). "Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks". In: *Energy Economics* 70, pp. 396–420. DOI: 10.1016/j.eneco.2017.12.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S014098831730436X.

Appendix 1. Market variables

Table 4: Explanatory variables

Dataset	Variable	Unit	Lag
Elbas data	Near-VWAP one day ago	EUR/MWh	$d-1$
	Near-VWAP one week ago	EUR/MWh	$d-7$
	Far-VWAP	EUR/MWh	
	Far-VWAP volume	MWh	
	Last available near-VWAP	EUR/MWh	$h-5$
	Last available near-VWAP volume	MWh	$h-5$
	Latest-VWAP	EUR/MWh	
Elspot data (per area)	Elspot prices	EUR/MWh	
Operating data (per available area)	Consumption prognosis	MWh	
	Total production prognosis	MWh	
	Wind production prognosis	MWh	
	Consumption prognosis error	MWh	$h-5$
	Total production prognosis error	MWh	$h-5$
	Wind production prognosis error	MWh	$h-5$
Regulating data (per Nordic area)	Up-regulation price	EUR/MWh	$h-5$
	Down-regulation price	EUR/MWh	$h-5$
	Imbalance consumption price	EUR/MWh	$h-5$
	Imbalance prod. price (purchase)	EUR/MWh	$h-5$
	Imbalance prod. price (sell)	EUR/MWh	$h-5$
	Dominating regulation direction	1 = Up; -1 = Down; 0 = no regulation	$h-5$
Seasonal dummies	Hour of day	$1 \dots 24$	
	Day of week	$1 \dots 7$	

Appendix 2. R code for data collection and pre-processing

```r
library(tidyverse); library(lubridate); library(zoo)

#source for load_data function: https://stackoverflow.com/questions/23190280/
load_data <- function(path) {
  files <- dir(path, pattern = '\\.csv', full.names = TRUE)
  tables <- lapply(files, read_csv)
  bind_rows(tables)
}

##transform into tibble, change column names, filter to hourly products only, filter to buy or sell areas under consideration only and omit irrelevant columns
#for data after  13th of June 2018
prepare_data_post <- function(df, areas, columns) {
  new_df <- as_tibble(df)
  colnames(new_df) <- columns

  new_df <- new_df %>%
    filter((buyer %in% areas | seller %in% areas) & type %in% c("P60MIN", "PH")) %>%
    select(-c(currency, cancelled, id, type))

  new_df <- new_df %>%
    mutate(timestamp = ymd_hms(new_df$trade_time, tz="UTC"),
           delivery_day = ymd(substring(new_df$power_hour, 4, 11)),
           delivery_hour = as.numeric(substring(new_df$power_hour, 13, 14))
    ) %>%
    select(-c(trade_time, power_hour)) %>%
    relocate(timestamp, delivery_day, delivery_hour)

  return(new_df)
}

#for data before 13th of June 2018
prepare_data_pre <- function(df, areas, columns) {
  new_df <- as_tibble(df)
  colnames(new_df) <- columns
```

```
35
36    new_df <- new_df %>%
37      filter((buyer %in% areas | seller %in% areas), cancelled == 0) %>%
38      select(-c(currency, cancelled))
39
40    new_df <- new_df %>%
41      mutate(timestamp = ymd_hms(new_df$trade_time, tz="UTC"),
42             delivery_day = ymd(substring(new_df$type, 4, 11)),
43             delivery_hour = as.numeric(substring(new_df$type, 13, 14)),
44             product = substring(new_df$type, 1,2)
45      ) %>%
46      filter(product == "PH") %>%
47      select(-c(trade_time, type, product)) %>%
48      relocate(timestamp, delivery_day, delivery_hour)
49
50    return(new_df)
51  }
52
53  #calculate volume-weighted average price
54  calculate_vwap <- function(data) {
55    agg_data <- data %>%
56      group_by(delivery_day, delivery_hour) %>%
57      summarise(volume = sum(qty), vwap = round(sum(price*qty)/volume, 2))
58    return(agg_data)
59  }
60
61  far_vwap <- function(df, hours) {
62
63    new_df <- df %>%
64      mutate(end_time = ymd_h(paste(delivery_day, delivery_hour)),
65             interval = floor(difftime(end_time, timestamp, units = "hours")))
       %>%
66      filter(interval > hours)
67
68    new_df <- calculate_vwap(new_df)
69    return(new_df)
70  }
71
72  near_vwap <- function(df, hours) {
73
74    new_df <- df %>%
```

```
    mutate(end_time = ymd_h(paste(delivery_day, delivery_hour)),
           interval = floor(difftime(end_time, timestamp, units = "hours"))) %>%
    filter(interval <= hours)

  new_df <- calculate_vwap(new_df)
  return(new_df)
}

exact_vwap <- function(df, hours) {

  new_df <- df %>%
    mutate(end_time = ymd_h(paste(delivery_day, delivery_hour)),
           interval = floor(difftime(end_time, timestamp, units = "hours"))) %>%
    filter(interval == hours)

  new_df <- calculate_vwap(new_df)
  return(new_df)
}

#calculates the latest-vwap variable
latest_vwap_fn <- function(df) {
  latest_price_5 <- exact_vwap(df, 5)
  latest_price_6 <- exact_vwap(df, 6)
  latest_price_7 <- exact_vwap(df, 7)
  latest_price_8 <- exact_vwap(df, 8)
  latest_price_9 <- exact_vwap(df, 9)

  join_latest <- list(latest_price_5, latest_price_6, latest_price_7,
                      latest_price_8, latest_price_9) %>%
    reduce(full_join, by = c("delivery_day", "delivery_hour"))

  #more recent price info takes priority
  join_latest$vwap.x[is.na(join_latest$vwap.x)] <- join_latest$vwap.y[is.na(
    join_latest$vwap.x)]
  join_latest$vwap.x[is.na(join_latest$vwap.x)] <- join_latest$vwap.x.x[is.
    na(join_latest$vwap.x)]
  join_latest$vwap.x[is.na(join_latest$vwap.x)] <- join_latest$vwap.y.y[is.
    na(join_latest$vwap.x)]
  join_latest$vwap.x[is.na(join_latest$vwap.x)] <- join_latest$vwap[is.na(
```

```
    join_latest$vwap.x)]

  join_latest <- join_latest[,c(1,2,4)]
  names(join_latest)[3] <- "latest_vwap"
  return(join_latest)
}

create_vwap_df <- function(data, hours) {
  near_vwap_data <- near_vwap(data, hours)

  far_vwap_data <- far_vwap(data, hours)

  vwap_data <- full_join(far_vwap_data, near_vwap_data, by = c("delivery_day
    ", "delivery_hour"))
  names(vwap_data)[3:length(vwap_data)] <- c("far_vol","far_vwap", "near_vol
    ", "near_vwap")

  vwap_data$delivery_hour <- as.factor(vwap_data$delivery_hour)

  vwap_data <- vwap_data %>%
    ungroup() %>%
    mutate(h5_near_vwap = lag(near_vwap, 5),
           h5_near_vwap_vol = lag(near_vol, 5),
           d1_near_vwap = lag(near_vwap, 24),
           d7_near_vwap = lag(near_vwap, 168)) %>%
    select(-near_vol) %>%
    filter(year(delivery_day)>2015)
  return(vwap_data)
}

prepare_elbas <- function(data_2015, data_2016, data_2017, data_2018, data
    _2019, data_2020, price_areas, names1, names2){

  data_2015 <- prepare_data_pre(data_2015, price_areas, names1)
  data_2016 <- prepare_data_pre(data_2016, price_areas, names1)
  data_2017 <- prepare_data_pre(data_2017, price_areas, names1)
  data_2019 <- prepare_data_post(data_2019, price_areas, names2)
  data_2020 <- prepare_data_post(data_2020, price_areas, names2)

  #2018 first half data
  data_2018a <-  data_2018[1:which(is.na(data_2018), arr.ind=TRUE)[1]-1,]
```

```
148   data_2018a <- data_2018a[, colSums(is.na(data_2018a)) == 0]
149   data_2018a <- prepare_data_pre(data_2018a, price_areas, names1)
150
151   #2018 second half data
152   data_2018b <- data_2018[which(is.na(data_2018), arr.ind=TRUE)[1]:nrow(data
        _2018),]
153   data_2018b <- data_2018b[, colSums(is.na(data_2018b)) == 0]
154   names_2018b <- c("power_hour", "currency", "price","qty", "buyer",
155                    "seller", "cancelled", "trade_time", "type", "id")
156   data_2018b <- prepare_data_post(data_2018b, price_areas, names_2018b)
157
158   #merge 2018
159   data_2018 <- bind_rows(data_2018a, data_2018b)
160
161   elbas_data <- bind_rows(data_2015, data_2016, data_2017, data_2018, data
        _2019, data_2020)
162   return(elbas_data)
163 }
164
165 ##### Elbas data #####
166 setwd("/Users/Roobu/Documents/University of Tartu/Lo puto o /data/")
167
168 #before 13th of June 2018
169 names1 <- c("trade_time", "type", "currency", "price",
170             "qty", "buyer", "seller", "cancelled")
171
172 #after  13th of June 2018
173 names2 <- c("trade_time", "type", "id", "power_hour", "currency", "price",
174             "qty", "buyer", "seller", "cancelled")
175
176 #restrict areas to EE,FI,SE,DK, LV, LT
177 price_areas <- c("EE", "FI", "SE1", "SE2", "SE3", "SE4", "DK1", "DK2", "LV",
        "LT")
178
179
180 #2016-2020 data, assumes working directory has been set correctly
181 data_2015 <- load_data("2015")
182 data_2016 <- load_data("2016")
183 data_2017 <- load_data("2017")
184 data_2018 <- load_data("2018")
185 data_2019 <- load_data("2019")
```

```
data_2020 <- load_data("2020")

elbas_data <- prepare_elbas(data_2015, data_2016, data_2017, data_2018, data
    _2019, data_2020, price_areas, names1, names2)


###### VWAP variables ######
vwap_data <- create_vwap_df(elbas_data, 4)

join_latest <- latest_vwap_fn(elbas_data)
join_latest$delivery_hour <- as.factor(join_latest$delivery_hour)


####### Operating data #######
setwd("/Users/Roobu/Documents/University of Tartu/Lo puto o /data/
    operating")

sdv_fun <- function(filename, skip) {
  op_data <- read.csv(filename, skip = skip, header = FALSE,
                      as.is = TRUE, sep = ";", na.strings = "")
  op_data <- op_data %>%
    filter(V1 != "AL", V2 != "U") %>%
    mutate(delivery_day = dmy(V6)) %>%
    select(-c(V1, V3, V4, V6, V11, V33)) %>%
    relocate(delivery_day) %>%
    filter(year(delivery_day) %in% c(2015:2020))
  return(op_data)
}

load_data_sdv <- function(path, skip) {
  files <- dir(path, pattern = '\\.sdv', full.names = TRUE)
  tables <- lapply(files, sdv_fun, skip=skip)
  bind_rows(tables)
}

op_data_fun <- function(path, skip, flag = FALSE) {
  #specify folder
  setwd(path)
  #initiate dataframe with first year
  for (year in c(2016:2020)){
    if (flag == FALSE) {
```

```
      flag = TRUE
      first_df <- load_data_sdv(as.character(year), skip)
      next
    }
    new_df <- load_data_sdv(as.character(year), skip)
    first_df <- bind_rows(first_df, new_df)
  }
  return(first_df)
}

#for areas with all operating data available
tidy_sdv_wind <- function(op_data) {
  names(op_data)[5:length(op_data)] <- c(1:24) #delivery hours
  names(op_data)[2:4] <- c("code", "weekday", "area") #meaningful names

  #change data types
  op_data$weekday <- factor(op_data$weekday)
  op_data[5:length(op_data)] <- lapply(op_data[5:length(op_data)], as.
    numeric)

  #turn 24 delivery hour columns into one column
  op_data <- op_data %>%
    pivot_longer(cols = -c(1:4), names_to = "delivery_hour", values_to = "
    value") %>%
    group_by(code, area, delivery_day) %>%
    mutate(row = row_number()) %>%
    pivot_wider(names_from = code, values_from = value) %>%
    select(-row) %>%
    ungroup()

  op_data$delivery_hour <- as.factor(op_data$delivery_hour)

  #create lagged error columns
  #PE - Day-ahead production prognosis, P - Total production
  #WE - Day-ahead wind production prognosis, WS - Settled wind production
  #F - Total consumption, E - Day-ahead consumption prognosis
  op_data <- op_data %>%
    mutate(load_error = lag(E - `F`, 5), prod_error = lag(PE - P, 5), wind_
    error = lag(WE - WS, 5)) %>%
    select(-c(`F`, P, WS)) %>%
    filter(year(delivery_day) > 2015) %>%
```

```
263     pivot_wider(names_from = area, values_from = c("PE", "WE", "E", "load_
        error", "prod_error", "wind_error"), names_sep = "_")
264
265   return(op_data)
266 }
267
268 #for areas that do not have wind data available
269 tidy_sdv_nowind <- function(op_data, columns) {
270   names(op_data)[5:length(op_data)] <- c(1:24) #delivery hours
271   names(op_data)[2:4] <- c("code", "weekday", "area") #meaningful names
272
273   #change data types
274   op_data$weekday <- factor(op_data$weekday)
275   op_data[5:length(op_data)] <- lapply(op_data[5:length(op_data)], as.
        numeric)
276
277   #turn 24 delivery hour columns into one column
278   op_data <- op_data %>%
279     pivot_longer(cols = -c(1:4), names_to = "delivery_hour", values_to = "
        value") %>%
280     group_by(code, area, delivery_day) %>%
281     mutate(row = row_number()) %>%
282     pivot_wider(names_from = code, values_from = value) %>%
283     select(-row) %>%
284     ungroup()
285
286   op_data$delivery_hour <- as.factor(op_data$delivery_hour)
287
288   #create lagged error columns, no wind error
289   op_data <- op_data %>%
290     mutate(load_error = lag(E - `F`, 5), prod_error = lag(PE - P, 5)) %>%
291     select(-c(`F`, P)) %>%
292     filter(year(delivery_day) > 2015) %>%
293     pivot_wider(names_from = area, values_from = columns, names_sep = "_")
294
295   return(op_data)
296 }
297
298 #interpolate missing values linearly
299 missing_val <- function(df){
300   idx <- colSums(is.na(df)) != 0
```

```
301   df[, idx] <- na.approx(df[, idx])
302   new_df <- df %>%
303     mutate(across(where(is.numeric), round, 2))
304   return(new_df)
305 }
306
307 #read in all data
308 op_data_ee <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/estonia", 12)
309 op_data_fi <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/finland", 15)
310 op_data_dk <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/denmark", 19)
311 op_data_lv <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/latvia", 12)
312 op_data_lt <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/lt", 14)
313 op_data_se <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
      Lo puto o /data/operating/sweden", 33)
314
315 ##### EE #####
316 #24.12.2019 PE and WE are missing, assume prognosis equals realised values
      and derive PE and WE as such
317 where_christmas <- op_data_ee[op_data_ee$delivery_day == "2019-12-24"
      ,][3:4,]
318 where_christmas[,2] <- c("PE", "WE")
319 op_data_ee <- rbind(op_data_ee, where_christmas)
320
321 tidy_ee <- op_data_ee %>%
322   tidy_sdv_wind() %>%
323   missing_val()
324
325 ##### FI #####
326 tidy_fi <- op_data_fi %>%
327   filter(V2 %in% c("E", "F", "PE", "P")) %>%  # no wind variables
328   tidy_sdv_nowind(c("PE", "E", "load_error", "prod_error")) %>%
329   missing_val()
330
331 ##### DK #####
332 tidy_dk <- op_data_dk %>%
333   filter(V2 %in% c("E", "F", "PE", "P", "WS", "WE"), V7 != "DK")# omit
```

```
    regulating data for now

tidy_dk$V7[tidy_dk$V7 == "JY"] <- "DK1"
tidy_dk$V7[tidy_dk$V7 == "SJ"] <- "DK2"

tidy_dk <- tidy_dk %>%
  tidy_sdv_wind() %>%
  missing_val()

##### SE #####
tidy_se <- op_data_se %>%
  filter(V2 %in% c("E", "F", "PE", "P", "WE"), V7 != "SE") %>%
  tidy_sdv_nowind(c("PE", "E", "WE", "load_error", "prod_error")) %>%
  missing_val()

##### LV #####
tidy_lv <- op_data_lv %>%
  tidy_sdv_wind() %>%
  missing_val()

##### LT #####
tidy_lt <- op_data_lt %>%
  filter(V2 %in% c("E", "F", "PE", "P")) %>%
  tidy_sdv_nowind(c("PE", "E", "load_error", "prod_error")) %>%
  missing_val()


#merge tidy_ee, tidy_lv, tidy_lt, tidy_fi, tidy_dk, tidy_se
op_merge <- list(tidy_ee, tidy_lv, tidy_lt, tidy_fi, tidy_dk, tidy_se) %>%
  reduce(left_join, by = c("delivery_day", "weekday", "delivery_hour"))


###### Day-ahead price data ######
sdv_spot_all <- function(filename, skip) {
  spot_data <- read.csv(filename, skip = skip, header = FALSE,
                        as.is = TRUE, sep = ";", na.strings = "")
  spot_data <- spot_data %>%
    mutate(delivery_day = dmy(V6)) %>%
    relocate(delivery_day) %>%
    select(-c(V1,V2,V3,V4,V6,V12,V34)) %>%
    filter(year(delivery_day) %in% c(2015:2020), V8 == "EUR") %>%
```

```
    select(-V8)
  return(spot_data)
}

load_spot_sdv <- function(path, skip) {
  files <- dir(path, pattern = '\\.sdv', full.names = TRUE)
  tables <- lapply(files, sdv_spot_all, skip=skip)
  bind_rows(tables)
}

spot_data_fun <- function(path, skip, flag=FALSE) {
  #specify folder
  setwd(path)
  #initiate dataframe with first year
  for (year in c(2016:2020)){
    if (flag==FALSE) {
      flag = TRUE
      first_df <- load_spot_sdv(as.character(year), skip)
      next
    }
    new_df <- load_spot_sdv(as.character(year), skip)
    first_df <- bind_rows(first_df, new_df)
  }
  return(first_df)
}

tidy_sdv_spot <- function(spot_data) {
  names(spot_data)[4:length(spot_data)] <- c(1:24) #delivery hours
  names(spot_data)[2:3] <- c("weekday", "area") #meaningful names

  #change data types
  spot_data$weekday <- factor(spot_data$weekday)
  spot_data[4:length(spot_data)] <- lapply(spot_data[4:length(spot_data)],
    gsub, pattern = ",", replacement = ".")
  spot_data[4:length(spot_data)] <- lapply(spot_data[4:length(spot_data)],
    as.numeric)

  #turn 24 delivery hour columns into one column
  spot_data <- spot_data %>%
    filter((area %in% c("NO1", "NO2", "NO3", "NO4", "NO5", "FRE")) == FALSE)
     %>%
```

```
412     pivot_longer(cols = -c(1:3), names_to = "delivery_hour", values_to = "
        value") %>%
413     filter(year(delivery_day) > 2015) %>%
414     pivot_wider(names_from = area, values_from = value, names_glue = "{area
        }_spot")
415
416   spot_data$delivery_hour <- as.factor(spot_data$delivery_hour)
417   return(spot_data)
418 }
419
420 spot_data <- spot_data_fun("/Users/Roobu/Documents/University of Tartu/
        Lo puto o /data/elspot", 25)
421
422 tidy_spot <- spot_data %>%
423   tidy_sdv_spot() %>%
424   missing_val()
425
426
427 ###### Regulating data #######
428 regulating <- op_data_fun("/Users/Roobu/Documents/University of Tartu/
        Lo puto o /data/regulating", skip = 22)
429
430 tidy_regulating <- function(regu_data) {
431   names(regu_data)[5:length(regu_data)] <- c(1:24) #delivery hours
432   names(regu_data)[2:4] <- c("code", "weekday", "area") #meaningful names
433
434   regu_data <- regu_data %>%
435     filter(area %in% c("DK1", "DK2", "FI", "SE1", "SE2", "SE3", "SE4"))
436
437   #change data types
438   regu_data$weekday <- factor(regu_data$weekday)
439   regu_data[5:length(regu_data)] <- lapply(regu_data[5:length(regu_data)],
        gsub, pattern = ",", replacement = ".")
440   regu_data[5:length(regu_data)] <- lapply(regu_data[5:length(regu_data)],
        as.numeric)
441
442   #turn 24 delivery hour columns into one column
443   regu_data <- regu_data %>%
444     pivot_longer(cols = -c(1:4), names_to = "delivery_hour", values_to = "
        value") %>%
445     group_by(code, area, delivery_day) %>%
```

```
446     mutate(row = row_number()) %>%
447     pivot_wider(names_from = code, values_from = value) %>%
448     select(-row) %>%
449     ungroup()
450 
451   regu_data$delivery_hour <- as.factor(regu_data$delivery_hour)
452   regu_data$DD <- as.factor(regu_data$DD)
453 
454   regu_data <- regu_data %>%
455     pivot_wider(names_from = area, values_from = c("RO", "RN", "RC", "RP", "
      RS", "DD"), names_sep = "_") %>%
456     ungroup() %>%
457     mutate(across(4:45, lag, n=5)) %>%
458     filter(year(delivery_day)>2015)
459 
460   return(regu_data)
461 }
462 
463 regu_data <- regulating %>%
464   tidy_regulating() %>%
465   missing_val()
466 
467 #due to na.approx approximating DD factors as well, deal with one 0,5 value
      manually
468 regu_data[regu_data$DD_DK2 == "-0.5",]$DD_DK2 <- 0 #neutral value
469 regu_data <- regu_data %>% mutate(across(39:45, factor))
```

Appendix 3. R code for data exploration and visualisation

```r
library(ggplot2)

#Division into initial calibration window, validation set and test set
ggplot(merge_all3, aes(delivery_day, near_vwap))+
  geom_line(color = "dodgerblue2") +
  labs(x = "Date", y = "near-VWAP") +
  scale_x_date(breaks = as.Date(c("2016-01-01", "2016-12-30", "2017-03-31",
    "2020-12-31")),
               guide = guide_axis(n.dodge=2))+
  geom_vline(xintercept = as.numeric(as.Date("2016-12-30")), linetype=2)+
  geom_vline(xintercept = as.numeric(as.Date("2017-03-31")), linetype=2)+
  scale_y_continuous(breaks = seq(0, 600, by = 50)) +
  annotate("text", x=as.Date("2016-06-01"), y=400, label= "Initial
    calibration window") +
  annotate("text", x=as.Date("2017-03-01"), y=400, label= "Lambda validation
    ") +
  annotate("text", x=as.Date("2018-07-01"), y=400, label= "Out-of-sample
    test period") +
  theme_classic()


#An example of Elbas trades for a given delivery hour: 08.09.2019, hour 22
elbas_example <- elbas_data %>% filter(delivery_day == "2019-09-08",
    delivery_hour == 22)

library(scales)
ggplot(elbas_example, aes(x=timestamp, y=price))+
  geom_line(color="dodgerblue2")+
  scale_x_datetime(labels = date_format("%H:00"), date_breaks = "2 hours") +
  labs(x="Time of trade", y="Trade price (EUR/MWh)") +
  theme_classic()


#An example of two weeks of near-vwap
example_near_vwap <- merge_all3 %>%
  filter(delivery_day >= "2019-09-02", delivery_day <= "2019-09-08") %>%
  select(delivery_day, delivery_hour, near_vwap)

```

```
example_near_vwap$time <- ymd_h(paste(example_near_vwap$delivery_day,
                                      as.character(example_near_vwap$delivery_
    hour),
                                      sep = ","))

ggplot(example_near_vwap, aes(x=time, y=near_vwap))+
  geom_line(color = "dodgerblue2")+
  scale_x_datetime(labels = date_format("%H:00"), date_breaks = "6 hours") +
  labs(y = "Near-VWAP (EUR/MWh)", x="Time of delivery hour")+
  theme_classic()

#Near-VWAP average for each of the 24 delivery hours
near_vwap_per_hour <- merge_all3 %>%
  group_by(delivery_hour = factor(delivery_hour, levels = c(1:24))) %>%
  summarise(avg_near_vwap = mean(near_vwap))

ggplot(near_vwap_per_hour, aes(x=delivery_hour, y=avg_near_vwap))+
  geom_col(fill="dodgerblue2") +
  labs(x="Delivery hour", y="Near-VWAP (EUR/MWh)") +
  theme_classic()

#Total volume per buyer and seller bidding area
price_areas <- c("EE", "FI", "SE1", "SE2", "SE3", "SE4", "DK1", "DK2", "LV",
     "LT")
total_vol_buyer <- elbas_data %>%
  filter(year(delivery_day) > 2015, buyer %in% price_areas) %>%
  group_by(buyer) %>%
  summarise(total_volume = round(sum(qty), 0))
total_vol_buyer$buyer <- as.factor(total_vol_buyer$buyer)

total_vol_seller <- elbas_data %>%
  filter(year(delivery_day) > 2015, seller %in% price_areas) %>%
  group_by(seller) %>%
  summarise(total_volume = sum(qty))
total_vol_seller$seller <- as.factor(total_vol_seller$seller)

library(gridExtra)
buyer_plot <- ggplot(total_vol_buyer, aes(x=reorder(buyer, total_volume), y=
    total_volume)) +
  geom_col(fill="dodgerblue2")+
  coord_flip() +
```

```
  labs(y="Total volume (MWh)", x="Buyer area") +
  scale_y_continuous(labels = number) +
  theme_classic()

seller_plot <- ggplot(total_vol_seller, aes(x=reorder(seller, total_volume),
    y=total_volume)) +
  geom_col(fill="dodgerblue2")+
  coord_flip() +
  labs(y="Total volume (MWh)", x="Seller area") +
  scale_y_continuous(labels = number) +
  theme_classic()

grid.arrange(buyer_plot, seller_plot, ncol=2)


#Yearly total volume and no of trades
overall_volume <- elbas_data %>%
  filter(year(delivery_day) > 2015) %>%
  group_by(year = year(delivery_day)) %>%
  summarise(total_volume = sum(qty), trades = n())

yearly_vol_graph <- ggplot(data = overall_volume, aes(x=year, y = trades)) +
  geom_bar(stat = "identity", fill="dodgerblue2") +
  labs(x="Year", y = "Total number of trades") +
  scale_y_continuous(labels = number) +
  theme_classic()

yearly_trade_graph <- ggplot(data = overall_volume, aes(x=year, y = total_
    volume)) +
  geom_bar(stat = "identity", fill="dodgerblue2") +
  labs(x="Year", y = "Total volume (MWh)") +
  scale_y_continuous(labels = number) +
  theme_classic()

grid.arrange(yearly_vol_graph, yearly_trade_graph, ncol=2)


#No of trades across delivery hours
hour_trades <- elbas_data %>%
  filter(year(delivery_day) > 2015) %>%
  group_by(delivery_hour) %>%
```

```
  summarise(trades = n())

ggplot(data = hour_trades, aes(x=delivery_hour, y = trades)) +
  geom_bar(stat = "identity", fill="dodgerblue2") +
  labs(x="Delivery hour", y="No of trades") +
  scale_x_continuous(breaks = seq(1, 24, by = 1)) +
  scale_y_continuous(labels = number) +
  theme_classic()


#Mean elbas trade price per buyer area
area_prices <- elbas_data %>%
  filter(year(delivery_day) > 2015, buyer %in% price_areas) %>%
  group_by(buyer) %>%
  summarise(mean_price = mean(price))

ggplot(area_prices, aes(x=reorder(buyer, mean_price), y = mean_price)) +
  geom_col(fill="dodgerblue2") +
  coord_flip() +
  labs(x="Bidding area", y="Average trade price (EUR/MWh)") +
  theme_classic()


##### Summary statistics for near-VWAP
#on data that has not yet had its missing values imputed
training_set <- vwap_data %>% filter(delivery_day < "2016-12-30") %>%
    arrange(delivery_day, delivery_hour)
validation_set <- vwap_data %>% filter(delivery_day >= "2016-12-30",
    delivery_day < "2017-03-31") %>% arrange(delivery_day, delivery_hour)
test_set <- vwap_data %>% filter(delivery_day >= "2017-03-31") %>% arrange(
    delivery_day, delivery_hour)

summary(training_set$near_vwap)
summary(validation_set$near_vwap)
summary(test_set$near_vwap)
rbind(sd(training_set$near_vwap, na.rm = TRUE),
      sd(validation_set$near_vwap),
      sd(test_set$near_vwap, na.rm = TRUE))
```

```
#MAE/RMSE per delivery hour compared for best 1) benchmark 2) univariate and 3) multivariate
#baseline
baseline_matrix <- matrix(baseline1_errors_test, ncol = 24, byrow = TRUE)
#univariate
uni_matrix <- matrix(uni_test_set_errors, ncol = 24, byrow = TRUE)

hour_wise_metrics <- function(error_matrix){
  mae <- colSums(abs(error_matrix))/nrow(error_matrix)
  rmse <- (colSums(abs(error_matrix)**2)/nrow(error_matrix))**(1/2)
  return(cbind(mae,rmse))
}

rmse_df <- data.frame("delivery_hour" = c(1:24),
                      "baseline" = hour_wise_metrics(baseline_matrix)[,2],
                      "Univariate" = hour_wise_metrics(uni_matrix)[,2],
                      "Multivariate" = hour_wise_metrics(mv_test_results_small[[1]])[,2])

mae_df <- data.frame("delivery_hour" = c(1:24),
                     "baseline" = hour_wise_metrics(baseline_matrix)[,1],
                     "Univariate" = hour_wise_metrics(uni_matrix)[,1],
                     "Multivariate" = hour_wise_metrics(mv_test_results_small[[1]])[,1])

rmse_graph <- ggplot(rmse_df, aes(delivery_hour))+
  geom_line(aes(y=baseline, colour="Far-VWAP baseline"))+
  geom_line(aes(y=Univariate, colour="Univariate LASSO"))+
  geom_line(aes(y=Multivariate, colour="Multivariate LASSO"))+
  labs(x="Delivery hour", y="RMSE (EUR/MWh)", colour="Model")+
  scale_x_continuous(breaks = seq(1, 24, by = 1))+
  scale_y_continuous(breaks = seq(1, 16, by = 2))+
  theme_classic()+
  theme(legend.position = "bottom")

mae_graph <- ggplot(mae_df, aes(delivery_hour))+
  geom_line(aes(y=baseline, colour="Far-VWAP baseline"))+
  geom_line(aes(y=Univariate, colour="Univariate LASSO"))+
  geom_line(aes(y=Multivariate, colour="Multivariate LASSO"))+
  labs(x="Delivery hour", y="MAE (EUR/MWh)", colour="Model")+
  scale_x_continuous(breaks = seq(1, 24, by = 1))+
```

```
187   scale_y_continuous(breaks = seq(1, 6, by = 1))+
188   theme_classic()+
189   theme(legend.position = "bottom")
190 
191 grid.arrange(mae_graph, rmse_graph, ncol=2)
```

Appendix 4. R code for LASSO and benchmark models

```
library(tidyverse); library(lubridate); library(zoo); library(glmnet)

####### merging all data frames #######

vwap_op_merge <- left_join(op_merge, vwap_data, by = c("delivery_day", "
    delivery_hour"))

#missing value imputation
vwap_op_merge <- vwap_op_merge %>% missing_val()

merge_all <- left_join(vwap_op_merge, tidy_spot, by = c("delivery_day", "
    weekday", "delivery_hour"))
merge_all2 <- left_join(merge_all, regu_data, by = c("delivery_day", "
    weekday", "delivery_hour"))
merge_all3 <- left_join(merge_all2, join_latest, by = c("delivery_day", "
    delivery_hour"))
merge_all3 <- merge_all3 %>%
  missing_val()

variables3 <- merge_all3 %>%
  ungroup() %>%
  select(-delivery_day)

#define initial training set, validation and test set
calibration_window <- 364
validation_window <- 91
test_window <- nrow(variables3)/24-(calibration_window+validation_window)


###### Baseline models ######

calc_errors <- function(errors){
  mae <- sum(abs(errors))/length(errors)
  rmse <- (sum(errors**2)/length(errors))**(1/2)
  return(cbind(mae, rmse))
}

y_var <- merge_all3$near_vwap
```

```r
val_idx <- (24*calibration_window+1):(24*(calibration_window+validation_
    window))
test_idx <- (length(y_var)-24*test_window+1):length(y_var)

#baseline 1 - far-VWAP
baseline1_errors_val <- variables3$far_vwap[val_idx] - y_var[val_idx]
baseline1_errors_test <- variables3$far_vwap[test_idx] - y_var[test_idx]
baseline1_val <- calc_errors(baseline1_errors_val)
baseline1_test <- calc_errors(baseline1_errors_test)

#baseline 2 latest-VWAP
baseline2_errors_val <- variables3$latest_vwap[val_idx] - y_var[val_idx]
baseline2_errors_test <- variables3$latest_vwap[test_idx] - y_var[test_idx]
baseline2_val <- calc_errors(baseline2_errors_val)
baseline2_test <- calc_errors(baseline2_errors_test)

#baseline 3 SE3 price
baseline3_errors_val <- variables3$SE3_spot[val_idx] - y_var[val_idx]
baseline3_errors_test <- variables3$SE3_spot[test_idx] - y_var[test_idx]
baseline3_val <- calc_errors(baseline3_errors_val)
baseline3_test <- calc_errors(baseline3_errors_test)

#baseline 4 h-5 near-vwap
baseline4_errors_val <- variables3$h5_near_vwap[val_idx] - y_var[val_idx]
baseline4_errors_test <- variables3$h5_near_vwap[test_idx] - y_var[test_idx]
baseline4_val <- calc_errors(baseline4_errors_val)
baseline4_test <- calc_errors(baseline4_errors_test)

#baseline 5 d-1 near-vwap
baseline5_errors_val <- variables3$d1_near_vwap[val_idx] - y_var[val_idx]
baseline5_errors_test <- variables3$d1_near_vwap[test_idx] - y_var[test_idx]
baseline5_val <- calc_errors(baseline5_errors_val)
baseline5_test <- calc_errors(baseline5_errors_test)


###### LASSO models ######

freq <- 24 #univariate model window rolled forward by 24 hours
lambda_vec_longer <- 10**(-(31-c(1:31))/6) #from 0.00001 to 1, for
    univariate
lambda_vec_mv3 <- c(10**(-(10-c(4:19))/6)) #for initial multivariate
```

```
    framework
lambda_vec <- 10**(-(19-c(1:15))/6) # from 0.001 to 0.22, for adjusted
    multivariate framework


##### UNIVARIATE #####

#this function builds a univariate model, given a index for lambda vector
build_validation_model <- function(x_vars, y_var, calibration_window,
    validation_window, freq, lambda, lambda_vec){

  errors <- rep(NA, validation_window*freq)

  #roll the window by 24 (freq) hours each model calibration
  for (i in 0:(validation_window-1)) {
    #set train and validation indexes, which will be used to index x_vars
    and y_var
    train <- (1+freq*i):(freq*(calibration_window+i))
    validation <- (freq*(calibration_window+i)+1):(freq*(calibration_window+
    i+1))

    #train a Lasso model on a 364-day calibration window aka training data
    model <- glmnet(x_vars[train,], y_var[train], alpha = 1, lambda = lambda
    _vec[lambda])

    pred <- predict(model, s = lambda_vec[lambda], newx = x_vars[validation
    ,])
    errors[(1+i*freq):(i*freq+freq)] <- pred-y_var[validation]
  }
  return(errors)
}

#this function iterates over a given lambda vector and calls build_
    validation_model
validate_lambda <- function(df, calibration_window, validation_window, freq,
     lambda_vec) {

  start.time <- Sys.time()

  y_var <- df$near_vwap
  x_vars <- model.matrix(near_vwap~. , df)[,-1]
```

```
106
107   mae <- rep(NA,length(lambda_vec))
108   rmse <- rep(NA,length(lambda_vec))
109   error_matrix <- matrix(NA, nrow = validation_window*freq, ncol = length(
        lambda_vec))
110
111   for (lambda in 1:length(lambda_vec)) {
112     errors <- build_validation_model(x_vars, y_var, calibration_window,
        validation_window, freq, lambda, lambda_vec)
113
114     error_matrix[,lambda] <- errors
115     mae[lambda] <- sum(abs(errors))/length(errors)
116     rmse[lambda] <- (sum(errors**2)/length(errors))**(1/2)
117   }
118
119   end.time <- Sys.time()
120   time.taken <- end.time - start.time
121   print(time.taken)
122
123   return(list(error_matrix, as.data.frame(cbind(lambda_vec, mae, rmse))))
124 }
125
126 #this model takes in a given lambda value and calculates prediction errors
      on the test set
127 evaluate_test_set <- function(df, lambda, calibration_window, validation_
      window, freq) {
128
129   start.time <- Sys.time()
130
131   test_window <- nrow(df)/24-(calibration_window+validation_window)
132   y_var <- df$near_vwap
133   x_vars <- model.matrix(near_vwap~. , df)[,-1] #drops the intercept, added
        by glmnet automatically
134
135   predictions <- rep(NA, test_window*freq)
136   model_coefs <- matrix(nrow = test_window, ncol = ncol(x_vars)+1) #+1 for
        intercept
137
138   #roll the window by 24 (freq) hours each model calibration
139   for (i in 0:(test_window-1)) {
140     #shift forward by validation window, assume validation is done
```

```
    train <- (1+freq*(validation_window+i)):(freq*(calibration_window+
    validation_window+i))
    #first test day is 24 hours of 2017-03-31
    test <- (freq*(calibration_window+validation_window+i)+1):(freq*(
    calibration_window+validation_window+i+1))

    #train a Lasso model on a 364-day calibration window aka training data
    model <- glmnet(x_vars[train,], y_var[train], alpha = 1, lambda = lambda
    )
    model_coefs[(i+1),] <- matrix(coef(model))[,1]

    pred <- predict(model, s = lambda, newx = x_vars[test,])
    predictions[(1+i*freq):(i*freq+freq)] <- pred
  }
  errors <- predictions - y_var[(length(y_var)-24*test_window+1):length(y_
    var)]
  coef_names <- dimnames(coef(model))[[1]] #save coefficient names once

  end.time <- Sys.time()
  time.taken <- end.time - start.time
  print(time.taken)

  return(list(cbind(predictions, errors), model_coefs, coef_names))
}


###### MULTIVARIATE ######

#given an index for the lambda vector, builds the model
build_validation_model_mv <- function(x_vars, y_var, calibration_window,
    validation_window, freq, lambda, lambda_vec){

  errors <- matrix(NA, nrow = validation_window, ncol = freq)

  for (i in 0:(validation_window-1)) {
    train <- (1+1*i):(1*(calibration_window+i))
    validation <- (1*(calibration_window+i)+1):(1*(calibration_window+i+1))

    y_train <- y_var[train,]

    for (j in c(1:24)) {
```

```
177       x_train <- x_vars[x_vars$delivery_hour == j,][train,] %>% select(-
      delivery_hour)
178       x_train_matrix <- model.matrix(near_vwap~. , x_train)[,-1]
179 
180       x_val <- x_vars[x_vars$delivery_hour == j,][validation,] %>% select(-
      delivery_hour)
181       x_val_matrix <- t(as.matrix(model.matrix(near_vwap~. , x_val)[,-1]))
182 
183       y_var_train <- pull(y_train[,j]) #pull gets the vector from a tibble/
      dataframe
184 
185       #model is trained on the training set and prediction made on
      validation set
186       #lambda here is an index for the vector of lambda values (lambda_vec)
187       model <- glmnet(x_train_matrix, y_var_train, alpha = 1, lambda =
      lambda_vec[lambda])
188       pred <- predict(model, s = lambda_vec[lambda], newx = x_val_matrix)
189       errors[i+1,j] <- pred-pull(y_var[i+validation,j])
190     }
191   }
192   return(errors)
193 }
194 
195 #iterates over a vector of lambda values, calls build_validation_model_mv
      each time
196 validate_lambda_mv <- function(df, calibration_window, validation_window,
      freq, lambda_vec) {
197 
198   start.time <- Sys.time()
199 
200   y_var <- df %>% select(c("delivery_hour", "near_vwap"))
201 
202   y_var_mv <- y_var %>%
203     group_by(delivery_hour) %>%
204     mutate(row = row_number()) %>% #has to be done for unique distinguishing
       of each data point
205     pivot_wider(names_from = delivery_hour, values_from = near_vwap, names_
      prefix = "hour") %>%
206     select(-row)
207 
208   mae_matrix <- matrix(NA, nrow=length(lambda_vec), ncol=24)
```

```
  rmse_matrix <- matrix(NA, nrow=length(lambda_vec), ncol=24)

  #three-dimensional error array
  error_array <- array(NA, c(validation_window, freq, length(lambda_vec)))

  for (lambda in 1:length(lambda_vec)) {
    errors <- build_validation_model_mv(df, y_var_mv, calibration_window,
    validation_window, freq, lambda, lambda_vec)

    error_array[,,lambda] <- errors
    mae_matrix[lambda,] <- colSums(abs(errors))/nrow(errors)
    rmse_matrix[lambda,] <- (colSums(errors**2)/nrow(errors))**(1/2)
    print(lambda) #for keeping track when code is running

  }

  end.time <- Sys.time()
  time.taken <- end.time - start.time
  print(time.taken)

  return(list(mae_matrix, rmse_matrix, error_array))
}

#function parameter lambda has to be a vector of 24 indexes for lambda_vec
#assumes validation is done, calculates model errors on the test set
evaluate_test_set_mv <- function(df, lambda, calibration_window, validation_
    window, freq, lambda_vec) {

  start.time <- Sys.time()

  test_window <- nrow(df)/24-(calibration_window+validation_window)

  y_var <- df %>% select(c("delivery_hour", "near_vwap"))

  y_var_mv <- y_var %>%
    group_by(delivery_hour) %>%
    mutate(row = row_number()) %>% #has to be done for unique distinguishing
     of each data point
    pivot_wider(names_from = delivery_hour, values_from = near_vwap, names_
    prefix = "hour") %>%
    select(-row)
```

```
246
247 errors <- matrix(NA, nrow = test_window, ncol = freq)
248 model_coefs <- array(NA, c(test_window, 24,
249                            dim(model.matrix(near_vwap~. , df %>% select(-
    delivery_hour)))[2]))
250
251 for (i in 0:(test_window-1)) {
252   #shift forward by validation window, assume validation is done
253   train <- (1+1*(validation_window+i)):(1*(calibration_window+validation_
    window+i))
254   test <- (1*(calibration_window+validation_window+i)+1):(1*(calibration_
    window+validation_window+i+1))
255
256   y_train <- y_var_mv[train,]
257
258   for (j in c(1:24)) {
259     #filter x variables to only contain data for given hour j
260     x_train <- df[df$delivery_hour == j,][train,] %>% select(-delivery_
    hour)
261     x_train_matrix <- model.matrix(near_vwap~. , x_train)[,-1]
262
263     #test set
264     x_val <- df[df$delivery_hour == j,][test,] %>% select(-delivery_hour)
265     #transposing model matrix is necessary because R treats a single row
    dataframe
266     #as a vector and a bug arises, which is fixed with t() command
267     x_val_matrix <- t(as.matrix(model.matrix(near_vwap~. , x_val)[,-1]))
268
269     y_var_train <- pull(y_train[,j])
270
271     model <- glmnet(x_train_matrix, y_var_train, alpha = 1, lambda =
    lambda_vec[lambda[j]])
272     pred <- predict(model, s = lambda_vec[lambda[j]], newx = x_val_matrix)
273
274     errors[(i+1),j] <- pred-pull(y_var_mv[test,j])
275     model_coefs[(i+1),j,] <- matrix(coef(model))[,1]
276   }
277
278   coef_names <- dimnames(coef(model))[[1]] #coefficient names saved once
279   print(i) #keeping track
280 }
```

```
281
282   end.time <- Sys.time()
283   time.taken <- end.time - start.time
284   print(time.taken)
285
286   return(list(errors, model_coefs, coef_names))
287 }
288
289 ##### Univariate results #####
290
291 #validation results
292 uni_val_results <- validate_lambda(variables3, calibration_window,
        validation_window, freq, lambda_vec_longer)
293 #best lambda based on RMSE criteria
294 uni_lambda <- lambda_vec_longer[which.min(uni_val_results[[2]]$rmse)]
295 #test set results with previously found optimal lambda
296 uni_test_results <- evaluate_test_set(variables3, uni_lambda, calibration_
        window, validation_window, freq)
297
298 #validation set accuracy
299 uni_val_mae <- uni_val_results[[2]]$mae[which.min(uni_val_results[[2]]$rmse)
        ]
300 uni_val_rmse <- uni_val_results[[2]]$rmse[which.min(uni_val_results[[2]]$
        rmse)]
301
302 #test set accuracy
303 uni_test_set_errors <- uni_test_results[[1]][,2]
304 test_set_accuracy <- calc_errors(uni_test_set_errors)
305
306
307 ###### Multivariate results ######
308
309 #validation set, initial lambda vector
310 mv_val_results <- validate_lambda_mv(variables3, calibration_window,
        validation_window, freq, lambda_vec_mv3)
311
312 calc_mv_val_error <- function(mv_val_results) {
313   mv_val_rmse <- sum(apply(mv_val_results[[2]], 2, min))/24
314   mv_val_mae <- sum(apply(mv_val_results[[1]], 2, min))/24
315   return(cbind(mv_val_mae,mv_val_rmse))
316 }
```

```
317
318 calc_mv_val_error(mv_val_results)
319
320 #a vector of 24 indexes, each specifying the optimal lambda for all 24
        delivery hours of the day
321 mv_lambda <- apply(mv_val_results[[2]], 2, which.min)
322 #test set results
323 mv_test_results <- evaluate_test_set_mv(variables3, mv_lambda, calibration_
        window, validation_window, freq, lambda_vec_mv3)
324
325 #RMSE of multivariate test set
326 (sum(abs(mv_test_results[[1]])**2)/(24*nrow(mv_test_results[[1]])))**(1/2)
327 #MAE of multivariate test set
328 sum(abs(mv_test_results[[1]]))/(24*nrow(mv_test_results[[1]]))
329
330
331 ##### Adjusted multivariate framework #####
332 #less variables and smaller lambda values
333
334 #occurrence of predictors in univariate framework is counted on all test set
        days
335 coef_names <- uni_test_results[[3]]
336 model_coefs <- uni_test_results[[2]]
337 coef_count <- colSums(model_coefs != 0) #non zero coefficients
338 coef_percent <- coef_count/nrow(model_coefs)
339
340 coef_data <- as.data.frame(cbind(coef_names, coef_count, coef_percent))
341 coef_data$coef_count <- as.numeric(as.character(coef_data$coef_count))
342 coef_data$coef_percent <- as.numeric(as.character(coef_data$coef_percent))
343
344 #get top 20 variables, excluding intercept, which is the first element
345 top_sorted_coef <- coef_data[order(-coef_count),][2:21,]
346 #bottom 10 variables
347 bot_sorted_coef <- coef_data[order(coef_count),][1:10,]
348
349 top_sorted_coef_names <- as.character(top_sorted_coef$coef_names)
350 #manually rename factor variables, e.g. DD_FI1 to DD_FI
351 #because entire variable has to be chosen
352 top_sorted_coef_names[top_sorted_coef_names=="DD_FI1"] <- "DD_FI"
353 #last 3 are delivery_hour9, delivery_hour12, weekday7
354 #d7_near_vwap as 21st most frequent substitutes one of the delivery_hour
```

```
    variables
top_sorted_coef_names <- top_sorted_coef_names[1:(length(top_sorted_coef_
    names)-3)]
top_20_coef_names <- c(top_sorted_coef_names, "delivery_hour", "weekday", "
    d7_near_vwap")

#subset the main dataframe of x variables
variables3_small <- variables3[,top_20_coef_names]
variables3_small$near_vwap <- variables3$near_vwap #add near-vwap (output)
    as well

#smaller lambda values, from 0.001 to 0.215
mv_val_results_small <- validate_lambda_mv(variables3_small, calibration_
    window, validation_window, freq, lambda_vec)
#validation set errors
calc_mv_val_error(mv_val_results_small)

#determine best lambdas
mv_lambda_small <- apply(mv_val_results_small[[2]], 2, which.min)

#test set results
mv_test_results_small <- evaluate_test_set_mv(variables3_small, mv_lambda_
    small, calibration_window, validation_window, freq, lambda_vec)
#test set RMSE
(sum(abs(mv_test_results_small[[1]])**2)/(24*nrow(mv_test_results_small
    [[1]])))**(1/2)
#test set MAE
sum(abs(mv_test_results_small[[1]]))/(24*nrow(mv_test_results_small[[1]]))
```

Appendix 5. Validated LASSO λ parameters in case of multivariate framework

Table 5: LASSO parameter λ values for both the initial and adjusted model in the multivariate framework.

Delivery hour	Initial model λ's	Adjusted model λ's
1	4.642	0.215
2	4.642	0.215
3	4.642	0.215
4	4.642	0.215
5	4.642	0.215
6	4.642	0.215
7	4.642	0.001
8	21.544	0.215
9	14.678	0.215
10	14.678	0.215
11	14.678	0.001
12	14.678	0.215
13	10.000	0.215
14	6.813	0.215
15	6.813	0.215
16	4.642	0.215
17	6.813	0.215
18	10.000	0.215
19	6.813	0.215
20	10.000	0.001
21	10.000	0.215
22	6.813	0.215
23	6.813	0.215
24	4.642	0.215

Appendix 6. Full list of variable selection

Table 6: Frequency of variables used in univariate LASSO models in the 1372-day test period

Variable name	Count	Proportion
wind_error_DK1	1372	1.00
prod_error_SE2	1372	1.00
far_vwap	1372	1.00
FI_spot	1372	1.00
latest_vwap	1372	1.00
h5_near_vwap	1370	1.00
d1_near_vwap	1370	1.00
WE_LV	1369	1.00
prod_error_SE1	1367	1.00
RC_DK1	1358	0.99
DK1_spot	1352	0.99
load_error_DK2	1350	0.98
far_vol	1347	0.98
SP1_spot	1345	0.98
DD_FI1	1345	0.98
WE_DK1	1337	0.97
wind_error_DK2	1335	0.97
delivery_hour9	1328	0.97
delivery_hour12	1319	0.96
weekday7	1318	0.96
d7_near_vwap	1315	0.96
load_error_SE4	1313	0.96
prod_error_EE	1301	0.95
load_error_FI	1296	0.94
WE_EE	1290	0.94
delivery_hour14	1287	0.94
RS_DK1	1286	0.94
load_error_SE2	1271	0.93

Variable name	Count	Proportion
E_DK2	1269	0.92
PE_SE2	1269	0.92
prod_error_FI	1266	0.92
weekday6	1265	0.92
load_error_LV	1265	0.92
DK2_spot	1264	0.92
WE_SE3	1263	0.92
load_error_DK1	1260	0.92
E_FI	1259	0.92
weekday4	1252	0.91
delivery_hour18	1252	0.91
delivery_hour3	1246	0.91
delivery_hour13	1243	0.91
PE_SE3	1238	0.90
wind_error_EE	1232	0.90
RC_SE1	1232	0.90
load_error_SE3	1231	0.90
RS_DK2	1229	0.90
load_error_EE	1227	0.89
prod_error_LT	1225	0.89
delivery_hour11	1220	0.89
delivery_hour8	1210	0.88
prod_error_LV	1208	0.88
prod_error_DK2	1208	0.88
load_error_LT	1205	0.88
weekday3	1201	0.88
WE_SE1	1196	0.87
delivery_hour5	1190	0.87
delivery_hour4	1187	0.87
delivery_hour16	1185	0.86
prod_error_SE3	1176	0.86

Variable name	Count	Proportion
RC_DK2	1169	0.85
PE_LV	1167	0.85
delivery_hour22	1166	0.85
delivery_hour10	1164	0.85
wind_error_LV	1163	0.85
PE_FI	1151	0.84
PE_EE	1147	0.84
delivery_hour24	1144	0.83
RP_FI	1141	0.83
prod_error_DK1	1138	0.83
h5_near_vwap_vol	1137	0.83
PE_LT	1132	0.83
load_error_SE1	1129	0.82
DD_DK10	1125	0.82
delivery_hour23	1117	0.81
SE4_spot	1105	0.81
PE_DK2	1103	0.80
RC_FI	1103	0.80
weekday2	1095	0.80
PE_SE1	1094	0.80
delivery_hour20	1093	0.80
delivery_hour19	1090	0.79
weekday5	1089	0.79
RC_SE4	1073	0.78
DD_SE40	1073	0.78
E_DK1	1072	0.78
delivery_hour21	1071	0.78
delivery_hour7	1067	0.78
delivery_hour6	1052	0.77
DD_DK21	1042	0.76
PE_DK1	1037	0.76

Variable name	Count	Proportion
SE1_spot	1032	0.75
delivery_hour15	1029	0.75
RS_SE3	1029	0.75
delivery_hour17	1021	0.74
prod_error_SE4	1021	0.74
E_SE1	1014	0.74
WE_SE2	1012	0.74
RP_SE4	1005	0.73
DD_DK11	1000	0.73
DD_DK20	990	0.72
LT_spot	977	0.71
RP_DK2	976	0.71
WE_DK2	969	0.71
E_SE2	956	0.70
E_SE4	947	0.69
DD_FI0	934	0.68
RP_SE1	885	0.65
SE3_spot	879	0.64
E_LT	878	0.64
RP_DK1	873	0.64
delivery_hour2	867	0.63
RP_SE3	860	0.63
RO_DK1	846	0.62
DD_SE10	822	0.60
RO_SE1	813	0.59
PE_SE4	812	0.59
RC_SE3	811	0.59
RO_SE3	789	0.58
RS_SE4	783	0.57
LV_spot	755	0.55
RN_SE3	754	0.55

Variable name	Count	Proportion
RO_DK2	747	0.54
DD_SE31	741	0.54
DD_SE30	706	0.51
DD_SE20	702	0.51
E_LV	701	0.51
E_SE3	686	0.50
RN_DK2	684	0.50
RN_SE1	647	0.47
EE_spot	643	0.47
RO_SE4	643	0.47
RS_SE1	636	0.46
RS_FI	634	0.46
RO_SE2	586	0.43
WE_SE4	571	0.42
RN_SE4	561	0.41
E_EE	560	0.41
RP_SE2	557	0.41
RC_SE2	555	0.40
DD_SE41	544	0.40
RN_FI	523	0.38
SE2_spot	482	0.35
DD_SE11	415	0.30
RO_FI	344	0.25
DD_SE21	311	0.23
RS_SE2	298	0.22
RN_SE2	291	0.21
RN_DK1	282	0.21

Explanations for shorthand variable names in Appendix 5:

- RN – Regulating market: Down-regulating

- RO – Regulating market: Up-regulating
- RC – Imbalance price for consumption (used in settlement)
- RP – Imbalance price for production (purchase)
- RS – Imbalance price for production (sell)
- DD – Dominating regulation direction.
- E – Day-ahead consumption prognosis
- WE – Day-ahead wind production prognosis
- PE – Day-ahead production prognosis
- prod_error – Last known difference between production prognosis and actual production
- load_error – Last known difference between consumption prognosis and actual consumption
- wind_error – Last known difference between wind production prognosis and actual wind production
- area_spot – Day-ahead price for that area

Non-exclusive licence to reproduce thesis and make thesis public

I, Robert Pikmets,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, Forecasting intraday electricity prices on the Nord Pool using LASSO, supervised by Raul Kangro.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Robert Pikmets
25/05/2021