Høgskolen i Østfold

Time Series Data Prediction Of The Norwegian Energy Market

Bachelor Thesis

Khayam Nami, Isak Bamiani, Kjell Randby Kristensen, Ole Kristian Eriksen Nysted

Avdeling for Informasjonsteknologi
Høgskolen i Østfold
Halden
August 13, 2023

Abstract

The escalating electricity prices in Norway and Europe have created a pressing need for accurate electricity price predictions. These predictions are crucial in optimizing production and consumption, reducing waste, improving carbon efficiency, and ensuring affordable electricity for consumers. However, existing approaches to electricity price prediction need help accurately capturing supply and demand's complex dynamics. Therefore, there is a solid motivation to explore innovative solutions that leverage advanced techniques such as machine learning to improve the accuracy of electricity price forecasts.

The reliance on traditional regression analysis for electricity price prediction has limitations in capturing the nonlinear relationships and time-dependent patterns inherent in energy markets. To address this problem, this research aims to investigate and compare the performance of different machine learning algorithms in generating 24-hour forecasts of energy market prices in Norway (Bidding Zone NO1 - Østlandet). The objective is to identify the most effective algorithms that can accurately predict electricity prices, considering factors such as energy demand, generation, export, and market trends. Additionally, this study utilizes a comprehensive dataset derived from publicly available data provided by ENTSO-E, enabling a thorough analysis of bidding zone 1 and its neighboring zones.

To tackle the problem mentioned above, we employed a range of machine learning algorithms, including linear regression (LR), XGBoost, support vector regression (SVR), and random forest regressor (RFR), for electricity price prediction. These algorithms were trained and evaluated using our dataset, derived from publicly available ENTSO-E data. The evaluation metrics used to compare the models included mean absolute error (MAE), mean absolute percentage error (MAPE), mean squared error (MSE), and root mean squared error (RMSE). The LR method emerged as the best-performing algorithm, closely followed by XGBoost. In comparison, SVR and RFR performed relatively worse compared to the other models.

In conclusion, this research project addressed the challenge of accurate electricity price prediction in the Norwegian energy market. Through analyzing and comparing different machine learning algorithms, we identified the LR method

as the most effective approach for forecasting electricity prices, with an MAE of 2.7 and an MSE of 21.7, closely followed by XGBoost. Furthermore, by leveraging our dataset, we obtained valuable insights into the complex dynamics of the Norwegian energy market. Using advanced machine learning techniques contributes to developing more sustainable, cost-efficient, and reliable energy markets.

Keywords— Electricity price, Time series data prediction, Linear regression, XGBoost, Random forrest regressor, Support vector regression, ENTSO-E, Feature selection, Data acquisition

Acknowledgments

Throughout the project, the supervisor has been an invaluable source of guidance and support, playing a crucial role in maintaining the momentum and success of the endeavor. Their unwavering assistance and expertise have proven to be instrumental in structuring the thesis and establishing well-defined milestones, which have greatly facilitated our progress and enabled us to meet demanding deadlines.

Furthermore, the supervisor's commitment to ensuring the project's success has been evident in their consistent availability and responsiveness. Whether it was addressing our queries, providing timely feedback, or offering guidance during unforeseen challenges, their dedication to our project has been unwavering.

By actively involving themselves in the project's development, the supervisor has motivated and inspired us to keep high standards and a sense of accountability within the team. Their consistent encouragement and belief in our abilities have driven our progress, pushing us to strive for excellence in every aspect of our work.

Prerequisites

It is recommended to possess a foundational knowledge of Machine Learning to fully comprehend the contents of this thesis. However, for the convenience of readers, Chapter 2 provides an elucidation of fundamental Machine Learning concepts. This chapter serves as a suitable resource for those seeking to understand the material presented in this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Group

Kjell Randby Kristensen studies Computer Science with a specialization in the field of Machine Learning. He has a Bachelor's in International Marketing and has worked as a market analyst. He has an interest in using Machine Learning and data science to analyze data and gain insight into different markets and subject areas, and to solve strategic problems related to different subject areas.

Khayam Nami (Project Leader) studies Computer Science with a specialization in the field of Machine Learning. He has a partially finished degree in Information Systems with a specialization in Business Intelligence and Software Engineering. His interests lie in data science, machine learning and trying to understand how everything is connected through hidden threads to solve puzzles and challenges.

Isak Bamiani studies Bachelor's in Computer Engineering with a specialization in Machine Learning and mobile programming. Isak is passionate about programming and statistics. After graduating, he would like to work as a data engineer or a back-end developer.

Ole Kristian Eriksen Nysted studies Computer Science with a specialization in Machine Learning. His interest is in using Artificial intelligence (AI) to understand the data in question. He finds financial markets particularly interesting, where AI is used to predict prices. He has an interest in robotics as well, which he has worked with for COOP Norge in their fully autonomous warehouse.

1.2 Client

Our client for the project is **Smart Innovation Norway**. The company is a non-profit research and innovation company, which works for the green shift and new sustainable workplaces (Smart Innovation Norway AS, n.d.). SIN (Smart Innovation Norway) offers new technology and business models to Norwegian municipalities, publicly owned ports, and energy companies. Their main goal is to act as the bridge from idea to reality, from theory to practice.

The company has 70+ employees, and provides services within; applied research, innovation, commercialization in sustainable energy, applied artificial intelligence, digital entrepreneurship,

smart cities, and social and behavioral innovation [60]. Their mission is to strive for social, economic, and environmental sustainability by putting the UN's 17 sustainability goals at the top of the agenda [59].

SIN has extensive experience with the smart cities concept. They have delivered many concrete projects (Smart Innovation Norway AS, n.d.), results of our project and other similar projects can be decisive in digitization of cities. Finding the most relevant ML model to predict electricity prices is quite revealing for research and innovation within IoT and smart cities. This can be seen as an important step towards a smarter and more flexible everyday life for Norwegian citizens and industry.

In this project, our contact point is Surender Redhu. He holds a Doctor of Philosophy (Ph.D.) in Electrical Engineering from the Indian Institute of Technology Kanpur (IITK), India. He is currently developing artificial intelligence solutions for renewable energy markets. Surender is also exploring the role of the internet of things in improving demand-side response and flexibility in energy markets for several EU projects at SIN.

1.3 Project Description

With record-high electricity prices in Norway and many parts of Europe , the ability to predict electricity prices has never been more important [49]. Electricity is not like most other commodities in that it is an extremely perishable commodity as it must be consumed at the same time as it is generated, it is therefore important that there is a constant balance between production and consumption [61]. Both production and consumption are affected by factors such as temperature which can affect the need for indoor heating or air conditioning, rainfall which determines the availability of hydropower, and gas prices as gas is both used for direct heating and electricity production. [9, 49]. As a result, both supply and demand for electricity vary a lot and can lead to unstable prices. By predicting electricity prices, electricity market stakeholders (producers and consumers) can regulate their production and demand, which minimizes waste, maximizes carbon efficiency, and gives consumers access to lower price electricity [49].

Time series data prediction has traditionally been performed through regression analysis; however, the use of ML algorithms to perform time series data prediction has become more common [65]. It is therefore of interest to explore how different ML algorithms perform on energy market price prediction.

The purpose of this thesis is to compare the performance of different algorithms on time series data prediction (24 hour forecasts) of energy market prices in Norway (Bidding zone NO1 - Østlandet) in the year 2022. An aggregate dataset consisting of energy market prices, time of price, and external factors will be created. This dataset will be preprocessed to optimize the performance of the different algorithms. After which 4-5 traditional regression ML algorithms will be chosen based on research about time series data prediction ml algorithms and will be used to train models on the dataset. When the models have been trained, they will be tested based on several performance metrics. Finally, the models will be compared based on their performance, and a conclusion will be drawn about which ones perform best.

Once the analysis of the selected traditional ML regression algorithms has been conducted, 2-3 deep learning regression algorithms may be selected for further analysis and comparison if sufficient time remains for the project period. These algorithms will be used to train and test models on the same data as the traditional ML regression algorithms selected. The models will then be compared and a conclusion will be drawn about which specific algorithms and algorithm categories perform best. The algorithms will be implemented in the Python ecosystem with Jupyter Notebook files.

1.4 Objectives

Main objective

Gain insight into which of our trained ML regression models are best suited to predict time series data in the Norwegian energy market (Bidding zone NO1 - Østlandet).
To achieve the main objective, we set up the three following sub-objectives:

Sub-objective 1

Gain insight into which of our trained traditional ML regression models are best suited to predict time series data in the Norwegian energy market (Bidding zone NO1 - Østlandet).

Sub-objective 2

Gain insight into which our trained deep learning regression models are best suited to predict time series data in the Norwegian energy market (Bidding zone NO1 - Østlandet).

Sub-objective 3

Gain insight into whether our traditional ML regression models or our deep learning regression models perform better at time series data prediction of the Norwegian energy market (Bidding zone NO1 - Østlandet).

1.5 Risk Assessment

Risk analysis and management in ML projects is a continuous process. It is an important step for ensuring a low probability of risk occurrence and predicting uncertainties in this project to improve the chance of successful project completion. Early risk identification and analysis are to secure achieving the main objectives. Risk analysis will be conducted regularly after achieving key milestones in the project timeline. A simple risk impact can vary for each of the main objectives. Known risks such as poor problem-solution, are readily apparent at this stage of the project, and some risks will take more time to uncover [37]. Lack of understanding of our goals caused by lack of experience is also considered in this risk evaluation for this project. In the worst-case scenario, we would need to change the contract with our client (SIN). Creating a risk assessment matrix (Figure 1.1) enables us to identify, analyze and manage potential risks. This matrix consists of a grid with two primary dimensions: probability and risk severity. probability represents the

probability or frequency of a risk event occurring, while risk severity refers to the severity or impact it could have on the project. By assigning ratings(low. medium, high, extreme) to risks based on these dimensions, the matrix provides a visual representation of the overall risk profile.

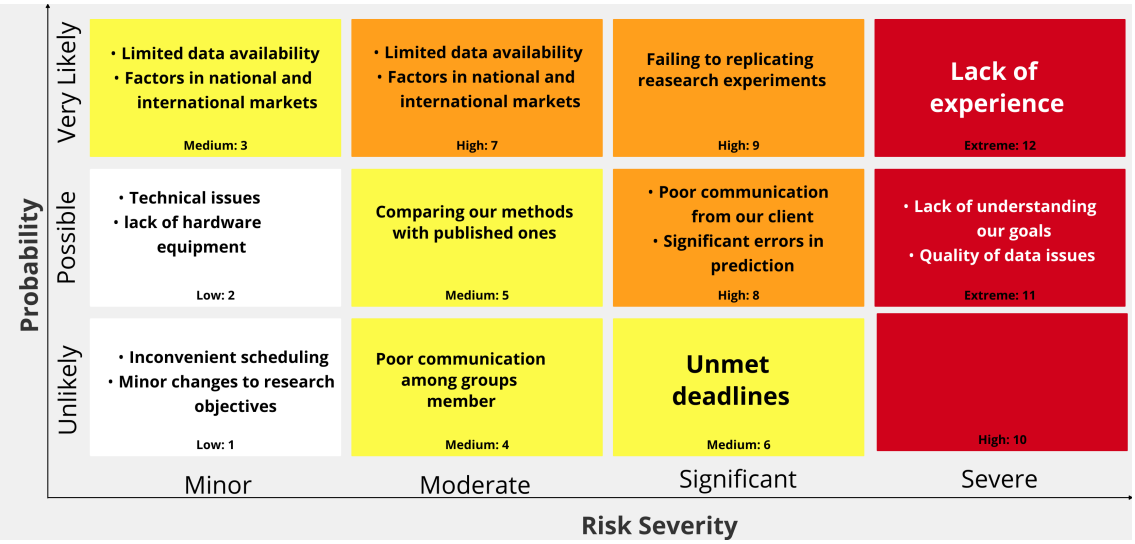

Figure 1.1: Risk Assessment Matrix

Risks with the highest impact and probability, within the next month or about 3 months:

- Data loss or corruption: In this case, it could be losing or working with corrupted or mismanaged data
- Quality of data issues: The data or the creation of the datasets can be of poor quality which can significantly impact the accuracy of the predictions/forecasting.
- Market volatility: Energy markets can be very volatile, and this volatility has increased during Covid and furthermore after the invasion of Ukraine and the impact of gas delivery from Russia to Europe
- Limited data availability: Limited availability of legitimate data that one can use may impact the scope and the accuracy of the research

Risks with the lowest impact and probability, within the next month or about 3 months:

- Technical issues: There may be technical issues due to a lack of equipment or malfunctioning version control
- Inconvenient scheduling: Scheduling is a major part of a group project and most of the time only requires adjustments and rarely impacts the timeline or outcomes.
- Minor changes to research objectives: There may be minor changes to the research project because of its scope and the time available.

The possible challenges for our thesis can be described in three main categories.

1. Challenges around electricity marked mechanism in Norway and Europe:

Storing electricity on a large scale is not possible. This results in an electricity market, where production and consumption levels must match all the time. Good price predictions are difficult since bidding strategies used by participants are complicated. There are different auction formats i.e., uniform-price and pay-as-bid auction [78]. Regulations around interconnectors in the EU will affect the market. EU aims at 15% interconnection by 2030 [25], this means that every member of the EU should have the infrastructure needed to allow the transport of 15% of national-produced electricity to neighboring countries. An increase in the number of interconnectors between EU members will lead to greater price convergence and lower prices.[20]

2. Challenges with ML projects:

Our ML models may have significant errors in prediction, which can be caused by an inability to exploit data characteristics related to BZNO1 or capture global trends affecting the day-ahead market at BZNO1. Electricity prices are determined by a wide range of factors e.g., transmission congestion, market participant behaviors, generation outages, fossil energy market, demand, political situations, and weather [1]. Hidden relationships between these factors in national and international markets are difficult to integrate into ML models.

Quality and quantity of data are quite decisive in getting the best accuracy and evaluation [62]. Data quality is ensured since all data is collected from the European Network of Transmission System Operators for Electricity (ENTSOE). We decided to only collect and analyze the data for the year 2022. Data from the year 2021 will not be representative and reliable considering how much market flow has changed in one year. The Ukraine war has an impact on energy prices and gas availability.
ML algorithms also have an element of randomness (Stochastic), it can be challenging and dependent on data. Key elements to prevent the impact of mentioned risk are being realistic and having good-quality data. We do not want to have very high expectations. The desired goals may take much more than one semester to achieve.

3. Challenges around the literature review :

Comparing new methods with published ones becomes very challenging because researchers have to re-implement methods from the literature. As a result, comparisons with state-of-the-art methods are often avoided, and new methods are usually compared with simple and easy-to-implement methods.
List of challenges:

- Data availability and quality: The quality and availability of data will impact the validity and reliability of the results from the research. Lack of access to high-quality data can and will limit the scope and accuracy of the research. This is one of the challenges with re-implementing methods from published research.

- Reproducibility: Reproducing results from previous studies or replicating experiments can be challenging, particularly when the data from previous research is not available. This can be further compounded by a lack of transparency in reports on research methods and results.

- Resource constraints: Due to the nature of the project and the resources available, one cannot always use the same methods as state-of-the-art due to a lack of resources. This can be either in funding or specialized equipment.
- Time constraint: The biggest challenge for this thesis project is the lack of time available and what can be done with the number of people available. This constraint can be worsened if there is a case of somebody suddenly not being available or delays in work happens.

4. Risk assessment for teamwork: potential challenges and solutions

it's important to identify potential risks that could negatively impact the team's progress and success. We explain some of the key risks related to teamwork and suggest actionable solutions to help mitigate them.
List of challenges:

- Lack of commitment: If one or more team members lose interest or leave the project, it could slow down progress and impact the quality of the work.
- Communication problems: If there are issues with communication within the group, it can cause misunderstandings and mistakes, which could impact the quality and timeliness of the project.
- Unequal workload: If some team members take on more work than others, it could cause tension and negatively impact motivation and productivity.
- Poor communication from our client side: If there are issues with communicating effectively with the client, it can lead to misunderstandings, missed deadlines, and an overall negative impact on the project. It may happen that the client does not follow the contract and agreed terms.

To address these risks, it's important to:

- Set clear expectations and goals for each team member and make sure everyone is on the same page
- Establish regular communication channels and check-ins to ensure everyone is aware of progress and any issues that arise.
- Make sure that each team member is contributing equally and that work is distributed based on individual strengths and skills.
- establish clear communication channels with the client and ensure that all parties involved are aware of project timelines, expectations, and deliverables.

1.6 Method

In order to produce the deliveries needed to answer the research objectives, we will use the following methods which are based on the Model Development Life Cycle outlined by Tarek Amr [6]:

- Research traditional ML regression algorithms that are being used for time series data prediction then choose 4-5 of these to be used to create models.

- Research what preprocessing and hyperparameters should be done to achieve the best performance with each of the chosen algorithms.
- Research what external factors have an impact on the Norwegian and European energy markets. This information will then be used in the exploratory data analysis to get a better understanding of the data and to select features for the aggregated dataset.
- Aggregate a dataset with electricity prices and external features.
- Clean the data and prepare the data for exploratory data analysis.
- Conduct an exploratory data analysis of the dataset to gain insight from the data.
- Preprocess the dataset for each algorithm based on the insight gained from the exploratory data analysis and the conducted research on ML regression algorithms for time series data prediction.
- Split the dataset into training data and test data so that we can test the performance of the final models with previously unseen data, and validate our results.
- Train each model and tune the hyperparameters according to the individual algorithms, to achieve the highest performance.
- Test the final models while using metrics that will allow for performance comparison of the different models. These metrics should be chosen based on the research conducted on the ML regression models for time series prediction.
- Perform all the previous tasks with 2-3 deep learning regression algorithms for time series prediction if sufficient time remains of the project period at this stage.

1.7 Thesis Organization

Chapter 1: Introduction

The introduction chapter sets the stage for the entire thesis. It will provide an overview of the thesis, including the background and motivation behind our research, the problem statement we are addressing, and the research questions we aim to answer. We will also explain the scope and limitations of our study and provide an outline of the thesis.

Chapter 2: Machine Learning Concepts

In chapter two, we will cover fundamental ML concepts that are relevant to our research, such as supervised and unsupervised learning, regression and classification, model evaluation metrics, feature selection, and engineering. This chapter will serve as a theoretical foundation for the ML techniques we will use in the later chapters.

Chapter 3: Literature Review

Chapter three will be a literature review, where we will provide an overview of existing research on energy price prediction and ML methods for this task. We will identify the limitations and challenges of current approaches and demonstrate how our research can contribute to this field.

Chapter 4: Machine Learning Algorithms

Chapter four will cover the machine learning algorithms used in our project, where we will explain how each of them work, how they are selected and which evaluation metrics are used to to analyze there performance. This information will be relevant to understanding the implementation and the performances of each model on our data.

Chapter 5: Data Acquisition

Chapter five will cover data acquisition, where we will explain the sources of our data, how we collected and preprocessed it, and how we ensured data quality. This chapter will provide the necessary context for the data analysis and modeling that we will perform later.

Chapter 6: Exploratory Data Analysis

In chapter six, we will perform exploratory data analysis to gain insights into our data, such as identifying correlations, visualizing data, and detecting outliers. This chapter will set the stage for the implementation and performance evaluation of our models in the following chapters.

Chapter 7: Implementation

Chapter seven will describe the implementation of our ML models, including the methodology and model selection, the implementation and tuning of the models, and parameter optimization. This chapter will provide a detailed description of the modeling process and the methods used to develop our predictive models.

Chapter 8: Performance Evaluation

In chapter eight, we will evaluate the performance of our models using various metrics, cross-validation, and validation methods, and compare different models to determine the most effective approach.

Chapter 9: Discussion

Chapter nine will be dedicated to the discussion of our research findings, highlighting the interpretation of our results, the limitations of our study, and making recommendations for future research. This chapter will help readers understand the implications of our research and identify opportunities for future studies.

Chapter 10: Further Development

Chapter ten will explore further development opportunities for our research, such as proposed future directions, additional methods for improving model performance, and potential application areas for the models developed in this study.

Chapter 11: Conclusion

Finally, in chapter eleven, we will provide a summary of our research findings, highlight the contribution of our study to the field of energy price prediction, and discuss the implications of our research for future research and practice. This chapter will help readers understand the significance of our research and its potential impact on the field.

Chapter 2

Machine Learning Concepts

2.1 Introduction

In this chapter, we will introduce the machine learning concepts used in our study to predict electricity prices in bidding zone NO1. The goal of our project is to accurately predict electricity prices, which is a complex task.
By the end of this chapter, readers will have a better understanding of the machine learning concepts used in our study. This knowledge will lay the foundation for the subsequent chapters, where we will describe the data preparation, model development, and performance evaluation stages of our study.

2.2 Machine Learning Concepts

2.2.1 Time series data prediction

Time series data prediction also referred to as time series forecasting is a task that involves predicting future values of a chosen variable based on its historical observations over time [32]. Time series data can be characterized by their sequential nature, where data points collected are at regular or irregular time intervals, often exhibiting complex patterns such as trends, seasonality, and noise [19].

2.2.2 Statistical methods

Statistical methods involves the act of executing a study including planning, designing, collecting necessary and relevant data to analyse and draw interpretations in a meaningful manner so that it can be reported and the findings be researched further [4].

2.2.3 Artificial intelligence

The term Artificial Intelligence (AI) denotes the capacity of machines to accomplish tasks that typically necessitate human cognitive abilities such as learning, reasoning, perception, and decision-making, by simulating human intelligence through programming [56].

2.2.4 Machine learning

Machine learning, a subset of artificial intelligence, is concerned with designing models and algorithms that learn from data to make predictions or choices without direct programming [38]. Its primary goal is to facilitate the capability of computer systems to adapt and improve their performance as they interact with novel data continuously, thus streamlining the knowledge acquisition process and reducing human intervention [34].

2.2.5 Artificial neural network

An Artificial Neural Network (ANN) is a machine learning model inspired by the human brain's structure and function [38]. It is a computational network composed of layers of interconnected nodes called neurons, which process and transmit information. The basic structure of an ANN consists of an input layer, a single or multiple hidden layers, and an output layer. Each neuron in the input layer receives input data, which is then processed and transmitted through the hidden layers to the output layer, where a final output is generated. During model training, the weights of the neuron connections are adjusted to minimize the error between the actual values and the predicted output.

2.2.6 Deep learning

Deep learning is a subfield of machine learning that focuses on training multi-layer artificial neural networks to learn hierarchical representations of data [38]. The key advantage of deep learning models is their ability to automatically learn data representations without requiring explicit feature engineering. By learning these representations from large and complex datasets, deep learning models can often outperform traditional machine learning models.

2.2.7 Supervised learning

Supervised learning is a subfield of machine learning where algorithms learn to map input features to output labels or targets based on a labeled dataset, which contains pairs of input-output examples [28]. The goal of supervised learning is to generalize and make accurate predictions on unseen data by minimizing the difference between predicted and actual output values, typically using a loss function. Supervised learning encompasses various algorithms, including linear and logistic regression, support vector machines, decision trees, and neural networks [31]).

2.2.8 Unsupervised learning

Unsupervised learning is a category of machine learning techniques that focus on discovering hidden patterns, structures, or relationships within a dataset without the guidance of explicitly labeled training data [29]. In contrast to supervised learning, where the algorithm is provided with input-output pairs, unsupervised learning algorithms seek to discern underlying patterns and structures in the data by analyzing the input features alone [64]. This type of learning is particularly useful for scenarios where labeled data is scarce or costly to obtain.
Unsupervised learning has broad applications across various domains, including anomaly detection, customer segmentation, feature extraction, and data compression [27]. These algorithms are valuable for gaining insights into complex datasets, especially when the relationships between data points are not immediately apparent.

2.2.9 Reinforcement learning

Reinforcement Learning (RL) is a branch of machine learning that aims to train intelligent agents to make decisions through interacting with an environment in order to accomplish a specific goal [63]. In RL, an agent learns to select the best actions by receiving feedback in the form of rewards or penalties, based on the outcomes of its actions [44]. The agent's goal is to learn a policy, which is a set of rules connecting states to actions, that maximizes the expected total reward over time [7]. Reinforcement learning has been successfully used in various applications, such as robotics, game playing, recommendation systems, and self-driving cars [35].

2.2.10 Regression problems in machine meaning

Regression problems in machine learning involve predicting a continuous target variable based on input features or predictors [46]. In these problems, the goal is to learn a mapping from the input features to the continuous output, which can be achieved through various algorithms, such as linear regression, polynomial regression, support vector regression, or neural networks [31].

2.2.11 Classification problems in machine Learning

Classification problems in Machine Learning are a type of Supervised Learning Problem where the goal is to predict a categorical or discrete output variable based on one or more input variables[26]. The input variables, also known as features or predictors, are used to train a model that can make predictions on new data. Classification problems can be divided into two categories: Binary Classification and Multiclass Classification. In Binary Classification, the model predicts one of two possible outcomes, i.e, whether or not an object is of that class. In Multiclass Classification, the model predicts one of several possible classes based on its characteristics, i.e., which class type the object is.

2.2.12 Machine learning algorithm

A machine learning algorithm is a computational approach that uses math and statistics to learn from data, allowing for the automatic creation of models that can predict or make decisions [5]. The goal of these algorithms is to find patterns, structures, or connections within the data, which can then be used to draw conclusions, make predictions, or classify new, unseen examples [45].

2.2.13 Machine learning model

A machine learning model is a computer program that can find patterns to learn from and make decisions based on newl, previously unseen data to make decisions ot predictions [79].

2.2.14 Data preprocessing in machine learning

Data preprocessing is the conversion of unprocessed data into a format suitable for machine learning models [70]. This includes various techniques, such as data cleaning, feature scaling, and handling missing values.

2.2.15 Data cleaning

Data cleaning is a process in data preprocessing that involves detecting and correcting or removing errors in the data[70]. These errors include missing values, outliers, duplicates, and inconsistent data.

2.2.16 Handling missing values

Dealing with missing values is a crucial preprocessing step in machine learning because missing or incomplete data can greatly affect the performance and dependability of predictive models [15]. Missing values can occur for various reasons, such as data entry mistakes, equipment failures, or the lack of certain attributes for specific examples. Addressing missing values is particularly important in real-world applications where incomplete data is common and can result in biased or incorrect model predictions [8].

2.2.17 Feature engineering

Feature engineering involves selecting and transforming raw data into features that are relevant to the machine learning task at hand [70]. The aim is to improve the performance of the ML model by providing it with meaningful input features. Feature engineering can involve various techniques, such as scaling, normalization, dimensionality reduction, etc.

2.2.18 Feature scaling

Feature scaling is the process of standardizing the scale of the input features, typically between 0 and 1 or -1 and 1 [70]. The goal of feature scaling is to ensure that all input features contribute equally to the machine learning model and to improve its performance.

2.2.19 Feature normalization

Normalization is a technique used to rescale the data distribution of a set of observations, transforming it into a normal distribution [70]. The main objective of normalization is to ensure that the data meets the statistical assumptions of a given machine learning or statistical analysis model. Normalization is beneficial when the data is not normally distributed, as this may lead to incorrect statistical inferences.

2.2.20 Dimensionality reduction

Dimensionality reduction is the process of reducing the number of input features in an ML model [12]. Dimensionality reduction aims to simplify the data and remove noise, redundancy, and irrelevant information, which can improve the ML model's performance, reduce overfitting, and speed up the training process.

2.2.21 Lagged features

The creation of lagged features is an essential concept within time series analysis used to forecast future values of a given time series based on its past values [19]. The lagged features are created by

shifting the time series data forward or backward by a specified number of time steps, resulting in new features representing the historical behavior of the time series at different time intervals. By analyzing the relationship between the current and lagged features, the models can learn to identify trends and patterns in the time series data, which can lead to better forecasting performance.

2.2.22 Outliers in data

An outlier in data is a data point or observation that significantly deviates from the general pattern or distribution of the rest of the dataset [2]. Outliers can be the result of measurement or data entry errors, sampling variability, or genuine variations in the underlying process. In the context of machine learning and statistical analysis, outliers can have a considerable impact on the performance and accuracy of predictive models, as they can lead to biased or skewed estimations of the underlying relationships in the data [14]. Identifying and handling outliers is an essential step in the data preprocessing stage, and various techniques, such as outlier detection algorithms or robust statistical methods, are employed to mitigate their effects on the analysis [81].

2.2.23 Data leakage

Data leakage refers to a situation in which information is unintentionally transferred from the training dataset to the testing dataset in a machine learning model, resulting in an overly optimistic evaluation of the model's performance during testing [10]. Data leakage can occur when information from the testing dataset is used to prepare the training dataset or when features not available during the prediction phase are included in the model training. Data leakage can lead to models that perform poorly when deployed in real-world applications.

2.2.24 Correlation

Correlation is a statistical measure that describes the relationship between two variables [48]. It indicates how much one variable is related to another, and the strength and direction of that relationship. Correlation coefficients range from -1 to 1, with negative values indicating a negative relationship, positive values indicating a positive relationship, and a value of 0 indicating no relationship at all.

2.2.25 Numerical data

Numerical data, also known as quantitative data, in machine learning refers to data that can be represented by numerical values and can be subjected to mathematical operations [51].

2.2.26 Categorical data

Categorical data is data that consists of categories or groups. This data is often non-numerical and represents qualities or characteristics that cannot be measured using numbers [72]. Categorical data is commonly used in statistics, machine learning, and data analysis. Categorical data can be divided into two types: nominal and ordinal. Nominal data has no order or ranking, while ordinal data has a specific order or ranking.

2.2.27 Training data

Training data, in the context of machine learning, refers to a curated set of observations or examples utilized to train and develop a predictive model. This dataset typically consists of input features (independent variables) and corresponding output labels or target values (dependent variables) that represent the ground truth or desired outcomes [38]. The training data plays a pivotal role in shaping the model's ability to generalize and make accurate predictions on unseen or new data.

2.2.28 Validation data

Validation data is a portion of the dataset that is held out during training and is used to evaluate the performance of the model on unseen data [28]. The purpose of validation data is to monitor the progress of the model and to tune the hyperparameters of the model.

2.2.29 Testing data in machine learning

Testing data in machine learning refers to a subset of the dataset that is held out and not used during the training process, serving as an independent sample for evaluating the performance of a trained model [28].

2.2.30 Overfitting and underfitting

Overfitting and underfitting are both problems that can occur when building AI models. The main goal of building AI models is to have a model that generalizes well to new data and predicts the unseen data/future. To build a good machine learning model, we need to find the right balance between complexity and simplicity [36].

Overfitting happens when a model is too complex, and fits the training data too closely [36]. We could say in this case our model has memorized the training data, and it doesn't generalize well to new data presented. There are many reasons for an overfitted model, such as model training for too long, implementing too many hyperparameters, skewed dataset(data quality issue), and when training data is not representative. An example of overfitting in the real world would be trying to learn a book by heart instead of understanding the concepts.
Underfitting happens when a model is too simple and doesn't capture the patterns in the training data well enough [36]. This means that the model doesn't fit the training data closely enough, and it also doesn't generalize well to new data. Underfitting can happen when a model doesn't have enough parameters or isn't trained for long enough. It's like only reading the first chapter of a book and trying to understand the entire story.

2.2.31 Signal in machine learning

In machine learning, the term "signal" generally refers to the meaningful information or patterns present in the data that can be used to build accurate models or make predictions [28]. The signal is often embedded within or obscured by noise, which represents random fluctuations, errors, or irrelevant information that can negatively impact the performance of machine learning algorithms [14]. Extracting and isolating the signal from noise is a crucial step in the data preprocessing and feature selection phases of the machine learning pipeline, as it can significantly improve the quality of the model and enhance its generalization capabilities [42].

2.2.32 Noise in machine learning

Noise in machine learning is the existence of unimportant, random, or incorrect information in the input data that can negatively impact the learning process and the performance of predictive models [68]. Noise can come from different sources, like data entry mistakes, measurement errors, or natural variability in the underlying processes. The presence of noise can cause overfitting, which occurs when the model learns to capture not only the real patterns in the data but also the noise, leading to poor performance on unseen data [80].

2.2.33 Seasonality in data

Seasonality in data refers to the presence of predictable and recurring patterns or fluctuations in the data that are observed over regular time intervals, such as days, weeks, months, or years [13]. These patterns are often associated with natural, social, or economic phenomena, like weather changes, holidays, or sales trends. In machine learning, accounting for seasonality is essential when analyzing time series data or developing forecasting models, as it helps improve the accuracy and interpretability of the models.

2.2.34 Trends in data

Trends in data refers to emerging patterns, behavior or changes in the data distribution that have the potential to impact model performance and generalization over time [77]. Analyzing trends in data is essential for identifying shift in underlying data patterns, which can lead to the development of more robust and and adaptive machine learning models.

2.2.35 Stationarity in data

Stationarity in machine learning refers to a property of time series data where the underlying statistical characteristics, such as mean, variance, and autocorrelation, remain constant over time [13].

2.3 Summary

In this chapter, we provided an introduction to the ML concepts utilized in our study, which aims to accurately predict electricity prices in bidding zone NO1. Understanding the complexities involved in this task, we delved into various fundamental concepts that form the basis of ML.
We began by exploring the time series data prediction concept, which involves forecasting future values based on historical observations. Recognizing the sequential nature of time series data, we discussed complex patterns such as trends, seasonality, and noise. Statistical methods played a crucial role in our study, encompassing data planning, design, and analysis to draw meaningful interpretations. We emphasized the significance of statistical techniques in providing a structured framework for extracting insights from the collected data.

Artificial intelligence (AI) emerged as a broad concept encompassing machines' ability to mimic human cognitive abilities such as learning, reasoning, perception, and decision-making. AI and ML offer a promising approach to addressing the challenge of accurate electricity price prediction.

ML, as a subset of AI, took center stage in our study. We emphasized that ML algorithms enable machines to learn from data and make predictions without explicit programming. This capability allows the development of predictive models that can identify patterns and relationships within the data.

Artificial neural networks (ANNs) were introduced as machine learning models inspired by the structure and function of the human brain. ANNs consist of interconnected nodes called neurons, which process and transmit information. During model training, the weights of neuron connections are adjusted to minimize prediction errors. Deep learning, a subfield of machine learning, focuses on training multi-layer neural networks to learn hierarchical representations of data. The advantage of deep learning models lies in their ability to automatically learn data representations without explicit feature engineering, often outperforming traditional machine learning approaches.

We explored the distinction between supervised and unsupervised learning. Supervised learning involves mapping input features to output labels based on labeled data, while unsupervised learning aims to discover hidden patterns and structures within unlabeled data. These learning approaches have diverse applications and are essential in tackling different prediction tasks.

Reinforcement learning emerged as a branch of machine learning concerned with training intelligent agents to make decisions based on interactions with an environment. Agents learn through receiving feedback through rewards or penalties, and their goal is to maximize the expected total reward over time.

Furthermore, we covered various topics related to data preprocessing in machine learning. These included data cleaning, handling missing values, feature engineering, feature scaling, feature normalization, dimensionality reduction, lagged features, outliers in data, data leakage, correlation analysis, and the treatment of numerical and categorical data.

We also discussed the significance of data partitioning into ML training, validation, and testing sets. Overfitting and underfitting were addressed as common challenges in model training, emphasizing the need to balance model complexity and generalization performance. Lastly, we explored the notions of signal and noise in machine learning and the concepts of seasonality, trends, and stationarity in data. Understanding these concepts is crucial for analyzing and interpreting the patterns and characteristics of the data in our study.

This chapter provided readers with a solid foundation for comprehending the subsequent chapters by covering these fundamental machine-learning concepts. In the following sections, we will delve into the specific details of algorithms, our study's data preparation, model development, and performance evaluation stages, leveraging the knowledge gained from these concepts.

Chapter 3

Literature Review

3.1 Introduction

Due to the importance of electricity price forecasting numerous studies have investigated various ML and statistical methods to improve the accuracy of electricity price forecasting [67]. This literature review aims to provide an overview of the existing research on electricity price prediction on time series data, including the methods used, their performance, and the challenges encountered. By synthesizing the findings of previous studies, this review intends to identify the current state-of-the-art in electricity price forecasting and highlight the directions for future research in this field.

3.2 Comparing XGBoost, Random Forest, And SVR Classifiers For Electricity Price Forecasting

In their study, Albahli et al., analyzed daily spot electricity price data from 2003-2018 in Ontario using an enhanced machine learning model to predict electricity prices by comparing the overall performance of XGBoost, Random Forest, and Support Vector Regression (SVR) classifiers [3]. The authors evaluated the model performance with several metrics, including Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE). They discovered that the XGBoost model, with a 70%/30% split, could predict electricity prices with an MSE of 15.66 and MAE of 3.740%, resulting in a potential 25.32% reduction in electricity costs. The accuracy of their proposed technique was 91%, while the accuracy of benchmark algorithms RF and SVR was 89% and 88%, respectively. In conclusion, the study demonstrated the effectiveness of the enhanced machine learning models, specifically the XGBoost classifier, in predicting electricity prices with high accuracy. The authors also suggested exploring neural networks and clustering as further extensions to their work.

3.3 Comparing Deep Convolutional Neural Network And ARIMA Models For Electricity Price Forecasting

In their study, Hsu-Yung Cheng, Ping-Huan Kuo, Yamin Shen, and Chiou-Jye Huang proposed a deep convolutional neural network model for short-term electricity price forecasting [17]. The

authors used hourly spot electricity price data from the New York Independent System Operator (NYISO) market from 2014-2016 to evaluate the performance of their model. The data were randomly split into training and testing sets with a 70%/30% ratio. They compared the accuracy of their proposed model with a traditional autoregressive integrated moving average (ARIMA) model and evaluated the performance using several metrics, including MAE, MAPE, and RMSE. The authors found that their proposed deep convolutional neural network model outperformed the traditional ARIMA model in all performance metrics. Specifically, their proposed model achieved a lower MAE of 2.64, a lower MAPE of 3.93%, and a lower RMSE of 3.59 compared to the ARIMA model. The results suggest that the deep convolutional neural network model captures electricity prices' complex relationships and non-linear patterns more effectively. The study also conducted a sensitivity analysis to investigate the impact of different factors on electricity price forecasting, such as seasonality, time of day, and temperature. The authors found that including these factors improved the performance of the deep convolutional neural network model, indicating that incorporating external factors into the model can enhance the prediction performance.

3.4 Deep Learning And Statistical Models With A Combination Of CNN And LSTM Models

In their study, Dhruv Aditya Mittal et al., propose a DL model combining LSTM (long short-term memory) and CNN (convolutional neural network) techniques to forecast electricity prices in Australia [43]. The dataset used for the task contains 35K price data points from 1st January 2000 to 22nd April 2020 with a half-hourly frequency rate in Australia's New South Wales region. Data distribution for training, validation, and testing is taken to be 60%, 20%, and 20%, respectively.

The proposed model is compared with other state-of-the-art models. The comparison involves DL models such as a simple convolution neural network model with perceptrons, a simple LSTM model with perceptrons, and statistical models such as ARIMA and Prophet. The proposed model has 5173 learnable parameters and a window size 10, which means the dataset is partitioned into 10 subsections. The model includes a 1D convolution layer with 3 filters and a kernel size of 3, as well as a ReLu (Rectified Linear Unit) activation function to apply kernel filters to matrices and assign them weights.

The model's accuracy and efficiency are evaluated using three parameters, which are also compared with other deep learning and statistical models. These parameters include RMSE (Root Mean Squared Error) for the standard deviation of predicted errors, MSE (Mean Squared Error) for squared deviation, and MAPE (Mean Absolute Percentage Error) for the mean deviation percentage.

Model	MSE	RMSE	MAPE
ARIMA	290.96	17.05	12.89
Conv1D with Perceptron	264.85	16.274	9.76
LSTM with Perceptron	234.48	15.313	10.01
Conv1D with LSTM and Perceptron	191.85	13.881	8.9
Prophet	270.86	16.45	10.90

Figure 3.1: Performance Evaluation

Figure 3.1 shows that the worst-performing models were Prophet and ARIMA models. However, these models perform well when forecasting for large forecasting intervals. Authors noted that deep learning methods show more promising results for short-term forecasting (2-3 hours). They concluded that the proposed hybrid model is able to map the random behavior of the time series and gives reliable forecasts.

3.5 Comparing Machine Learning Regression Algorithms

In their study, Sedat Orene et al. proposed four state-of-the-art models to forecast electricity prices of Victoria City in Australia [52]. These methods are Decision Tree, Random Forest, Gradient boosting, and Linear Regression. These models are evaluated using metrics such as Mean Absolute Error (MAE) and determination constant. In this research, the dataset consists of 2016 days between 1st January 2015 and 6nd October 2020.

The dataset was obtained from the Kaggle platform, with features such as date, the demand for electricity, minimum and maximum temperature, solar exposure, and rainfall. The dataset is partitioned into three distinct segments, which are training, validation, and test sets. These sets are divided into ratios of 70%, 10%, and 20%, respectively. A comparison of mentioned models shows that the Gradient boosting model has the best performance with an MAE of 31.47, and the determination constant value is 0,49. The next best-performed model is Random Forest, with an MAE of 32.68. As part of the results in this paper, it is mentioned that it is easier to predict electricity prices in cold weather since the result has lower errors. The authors also mentioned the use of more metrics for future works.

3.6 Technical Indicators And Prediction For Energy Market Forecasting

In a study by Catherine McHugh, Sonya Coleman, and Dermot Kerr, they discussed that technical analysis is a tool used in stock market trading to capture trends over time and predict share price movement [39]. The Integrated Single Electricity Market (ISEM) in Ireland has led to the design of novel technical indicators for electricity price forecasting. These indicators help energy traders observe market price trends and decide when to buy or sell electricity. Technical indicators can be split into three types: trend, oscillator, and momentum. This work presents eight novel technical indicators specific to energy trading. These indicators are derived from standard financial trading

indicators but are not identical. The focus is on price indicators that improve day-ahead accuracy.

Hourly electricity price data from the ISEM day-ahead market were used to calculate the technical indicators. The calculated technical indicators were used as training inputs for machine learning models to forecast future electricity prices. The data was split 85% for training and 15% for model testing. A persistence model was created as a baseline to predict the test set historical electricity price data.

The model accuracy of all the persistence models during training and testing ranged between 73% and 80%. The RMSE was used to evaluate overall model accuracy for the test set. The Random Forest model generated the lowest RMSE value during model training and Gradient Boosting provided the lowest RMSE value during model testing with unseen data.

The machine learning algorithms were trained using the technical indicators as inputs and the actual price as output. Predictive performance was compared using the testing data. Overall model accuracy ranged from 84% to 93%. The Random Forest algorithm performed best when observing the training results, but XGBoost had the lowest RMSE overall and is considered the optimal algorithm in the comparison.
Both the model training and testing output exhibited a well-fitted model compared to the baseline model. The visual output shows a close fit between the actual price and the predicted price. The RMSE also demonstrates the overall accuracy of the models using technical indicators. The XGBoost model showed promising results during model testing and was further tested using additional technical indicators input data to predict the next day's electricity price values. The forecasted price had a good fit with the actual price resulting in an overall RMSE of 5.75.

The significance of each technical indicator was analyzed through feature importance. The feature importance score for each technical indicator varies in terms of the model's performance. For Gradient Boosting, Random Forest, and XGBoost, there were three significant indicators above 0.1. MAD (Moving average deviation), PR (Percentage range), and PPCMA (Percentage price change moving average) were important for all three techniques, suggesting that they are good indicators for electricity price prediction. ADX (Average directional movement index), MACD (Moving average convergence/divergence), and RSI (Relative strength index) scores were not important for any of the three techniques and were considered insignificant.

3.7 Electricity Price Forecasting: A Review Of The State-of-the-art With A Look Into The Future

Electricity price forecasting is crucial to energy management, especially for market participants such as producers, distributors, and consumers. Accurate price forecasting enables optimal decision-making, risk management, and efficient allocation of resources. In the study by Rafał Weron, he provides a comprehensive review, and an in-depth analysis of the current state-of-the-art in electricity price forecasting methodologies and offers a glimpse into the future of this field. This literature review will explore the key insights and contributions of Weron's work to the domain of electricity price forecasting [71].

3.7.1 Methodologies and techniques

Weron [71] presents a thorough examination of various methodologies employed in electricity price forecasting, categorizing them into four main groups: (1) statistical methods, (2) computational intelligence techniques, (3) hybrid methods, and (4) other approaches. Statistical methods include time series models, regression models, and stochastic processes. Computational intelligence techniques involve artificial neural networks, support vector machines, and other machine learning approaches. Hybrid methods combine various techniques to improve forecasting accuracy. Finally, other approaches encompass agent-based models, game theory, and equilibrium models.

1) Statistical Methods:
Statistical methods are based on statistical theory and are widely used in electricity forecasting due to their interpretability and robustness. Weron describes several statistical methods, such as:

- Time Series Models: These models, including ARIMA, GARCH (Generalized Autoregressive Conditional Heteroskedasticity), and SARIMA (Seasonal ARIMA), are designed to capture the temporal dependencies in time series data.
- Regression Models: Regression techniques, such as Linear Regression, Logistic Regression, and Quantile Regression, aim to model the relationships between electricity prices and relevant explanatory variables, which may include demand, supply, weather conditions, and other factors.
- Stochastic Processes: Stochastic models, like the mean-reverting jump-diffusion model and regime-switching models, account for the random and dynamic nature of electricity prices.

2) Computational Intelligence Techniques:
Computational intelligence techniques are data-driven methods that can automatically adapt and learn from data. These methods have gained popularity in electricity price forecasting due to their flexibility and ability to handle complex relationships. Weron discusses several computational intelligence techniques, including:

- Artificial Neural Networks (ANNs): ANNs are biologically inspired models that can learn complex non-linear relationships between inputs and outputs. They are highly customizable and have been successfully applied to electricity price forecasting.
- Support Vector Machines(SVMs): SVMs are supervised learning models that can perform classification and regression tasks. In electricity price forecasting, SVMs can model non-linear relationships and have demonstrated competitive performance
- Other Machine Learning Approaches: Weron also acknowledges the potential of other machine learning techniques, such as Decision Trees, Random Forests, and Gradient boosting machines, in electricity price forecasting.

3) Hybrid Methods:
Hybrid methods combine various techniques to improve forecasting accuracy by capitalizing on the strengths of individual methods while mitigating their weaknesses. Weron notes that hybrid models can involve the combination of statistical methods and computational intelligence techniques or the integration of multiple models within a single framework, such as ensemble learning or model stacking.

4) Other Approaches:
In addition to the previously mentioned categories, Weron presents several other approaches that have been employed in electricity price forecasting:

- Agen-Based Models: These models can simulate the interactions of individual agents within a market, capturing the complex dynamics of electricity markets and allowing for the exploration of various market scenarios.
- Game-Theory: Game-theoretic models can be employed to analyze strategic interactions between market participants to predict their behavior, which can influence the prices.

Overall, Weron provides a comprehensive overview of various methodologies and techniques used in electricity price forecasting, highlighting the strengths and weaknesses of each approach and their potential applications in different forecasting contexts

3.7.2 Challenges and issues:

The author highlights the challenges and issues associated with electricity price forecasting, including the non-stationary nature of electricity prices, the presence of seasonal and calendar effects, the impacts of regulatory and market changes, and the limited availability and quality of data. Weron emphasizes the importance of addressing these challenges in developing effective forecasting models and suggests potential solutions, such as incorporating exogenous variables, using high-frequency data, and applying advanced preprocessing techniques.

3.7.3 Conclusions

Rafał Weron's detailed review provides useful information on the latest methods for predicting electricity prices and emphasizes the challenges and possibilities in this area. As the electricity market keeps changing, dependable and precise forecasting methods will be more and more important for those involved. Weron's research sets the stage for additional studies and progress, leading to better prediction models and approaches that will help make energy management and decision-making more efficient.

3.8 Using Machine Learning For Long-Term Electricity Price Forecasts

In their research, Yousefi et al. [76] delve into the application of machine learning techniques to enhance the precision and dependability of long-term electricity price predictions. They assess a range of machine learning strategies, scrutinize their effectiveness, and discuss the associated challenges and prospects. This literature review delivers a straightforward and easily digestible overview of the study's aims, methodologies, results, and implications.

3.8.1 Objective

The main goal of the study is to see how well various machine learning techniques perform in forecasting long-term electricity prices. The authors emphasize that accurate forecasts are

important for many people involved in the electricity industry, such as power generators, consumers, regulators, and policymakers because they help with planning investments, managing risks, and creating policies.

3.8.2 Methodology

The authors apply different machine learning methods, among these are Linear Regression, Support Vector Regression, artificial neural networks, and extreme learning machines to predict long-term electricity prices. They use past electricity prices along with other factors, such as energy demand, fuel costs, and temperature, as inputs for their models. They also use data improvement techniques like normalization and feature selection to make the models more effective. To check how accurate the models are, they measure their performance using methods like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

3.8.3 Findings

The results of the study show that machine learning techniques can give more accurate long-term electricity price forecasts than traditional time series models. The ELM and ANN methods perform the best in terms of MAE and RMSE, showing that it has the potential for forecasting long-term electricity prices. However, the authors point out that the best model to use depends on the specific data and how far into the future the forecast needs to be.

3.8.4 Challenges

The study mentions several challenges with using ML (Machine Learning) techniques for long-term forecasting of electricity prices, such as the quality and availability of data. Despite the challenges, the research points to machine learning techniques having the potential to improve long-term forecasting of electricity prices and help market participants to make better decisions

Wrapping up, this study contributes to the expanding research on ML applications in long-term electricity price predictions. The results reveal that methods like ELM and ANN can enhance forecast accuracy, aiding decision-making for market players. Nevertheless, further research is required to tackle the challenges and limitations of these methods, striving for even more precise, dependable, and comprehensible long-term electricity price forecasting models.

3.9 Using SVM To Forecast Energy Prices With The Impact Of Fossil Energy Prices

Ali Shiri, Mohammad Afshar, Ashkan Rahimi-Kian and Behrouz Maham found that accurate prediction of electricity prices is crucial for competitive markets, but conventional time series methods often overlook other influential factors [58]. This study proposes a novel approach using support vector machines (SVM) that incorporates significant variables such as oil and natural gas prices to improve predictions. The model is based on German electricity prices and considers variables including German BaseLoad Electricity Price, GASPOOL daily price, Net-Connect-Germany (NCG) daily price, and West Texas Intermediate (WTI) daily price. The SVM model is run on various combinations of these variables, and the performance is evaluated using Mean Square Error (MSE). The study covers the period from 2010 to the end of 2012, including a rare

event where the electricity price went negative for three days due to high power plant shutdown costs. The study includes two models, one for normal situations (2012a) and one for negative prices (2012b), with 16 different modes in total. The model has a training and prediction phase, with a radial basis function (RBF) chosen as the kernel function. The study concludes that WTI data is the most beneficial variable to include in the model and that the SVM algorithm has higher prediction accuracy than artificial neural networks (ANN). Precise electricity price prediction can assist private electricity markets in decision-making and future production strategies by considering the many effective parameters that influence the market.

3.10 Summary Literature Review

Electricity price forecasting is essential in the energy industry, as it helps people make intelligent decisions about producing, using, and trading electricity. However, predicting electricity prices can be challenging because the market constantly changes.
This literature review looks at research on predicting energy prices using historical data and discusses the methods used, how well they work, and the challenges they face.

Generally, most studies have used machine learning-based methods, like Support Vector Machine (SVM) and Long Short-Term Memory (LSTM), to predict short-term electricity prices in different markets. These methods have shown promising results when using the right features and data of higher quality, and these methods offer consistently low rates of error in predictions.
Besides machine learning, some studies have used statistical methods, which include AutoRegressive Integrated Moving Average(ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity(GARCH), to forecast electricity prices. These models look at how prices have changed and try to find patterns to help make predictions.

Some other research has focused on predicting long-term forecasting using models like Artificial Neural Networks(ANN) and Extreme Learning Machines(ELM). While these models show potential for improving long-term forecasts, they still face data quality, complexity, and evaluation challenges. Based on this, one can understand that these models would be unfit for our work due to the lack of higher-quality data since we can only get reliable data from 2022. These concerns will be talked about further in the challenges and limitations chapters.

In conclusion, this literature review highlights the progress made in machine learning about forecasting energy prices and related tasks to such an undertaking. Another thing to note is that one will need a high amount and quality. Some of the types of statistical methods like ARIMA can be used for our project along with ANN, though to reiterate, this will be very challenging because of the restrictions we face because of the restricted amount of data we can use.

Chapter 4

Machine Learning Algorithms

4.1 Introduction

In this chapter, we will introduce the algorithms linear regression, random forest, support vector regression and XGBoost used in our study to predict electricity prices in bidding zone NO1. The goal of our project is to accurately predict electricity prices, which is a complex task influenced by numerous factors such as weather, market demand, and supply conditions. Traditional methods used in price prediction have limitations, hence the need for a more robust and accurate solution.

4.2 Linear Regression

Linear regression is a statistical technique that models the relationship between a dependent variable and one or more independent variables [54]. It assumes that there is a linear relationship between the independent and dependent variables. The goal of linear regression is to find the best-fit line that represents the relationship between the variables. Each variable is given a coefficient of what impact they have on the target. This is a good method to use when the target value is expected to be a linear combination of the feature values.

OLS' coefficient estimations rely on the features' independence. The design matrix approaches singularity when features are highly correlated and the columns of the design matrix X have a roughly linear relationship. As a result, the least-squares estimate becomes very vulnerable to random errors in the observed target, leading to a huge variance. Multicollinearity can occur, for instance, when data are gathered without the use of an experimental design.

Linear regression is used for numerical data and is commonly used to predict continuous variables such as stock prices, housing prices, and temperature. It is also used in fields such as finance, economics, and social sciences to analyze relationships between variables.

Linear regression is based on the statistical principles of ordinary least squares (OLS), which minimizes the sum of the squared errors between the predicted and actual values. The mathematical problem of OLS is formulated in Equation (4.1). Here, the OLS method finds the best-fit line that minimizes the distance between the predicted and actual values of the dependent variable.

$$\min_{w} \|Xw - y\|_2^2 \tag{4.1}$$

Ridge regression is another linear regression algorithm which tries to address some of the problems of OLS by adding a penalty to the coefficients' size. Ridge takes a parameter α which controls the amount of shrinkage of the coefficients. When α is large, the coefficient becomes more robust to collinearity. The mathematical problem of Ridge regression is formulated in Equation (4.2)

$$\min_{w} \|Xw - y\|_2^2 + \alpha \|w\|_2^2 . \tag{4.2}$$

Lasso is a linear model that projects sparse coefficients. Due to its propensity to favor solutions with fewer non-zero coefficients, it is helpful in some situations by lowering the number of features that the provided solution depends on. Lasso and its variations are essential to the study of compressed sensing because of this. It is capable of recovering the precise set of non-zero coefficients under specific circumstances. It is mathematically a linear model with an additional regularization term the l1-norm - shown in Equation (4.3).

$$\min_{w} \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha \|w\|_1 . \tag{4.3}$$

Both l1 and l2-norm regularization of the coefficients were used to create the linear regression model known as ElasticNet. When learning a sparse model, similar to Lasso, with few non-zero weights, the regularization properties of Ridge are maintained. We can change the convex combination of l1 and l2 by modifying the l1 ratio parameter. Elastic-net is useful when numerous features are connected to one another. While Lasso is more likely to select one at random, Elastic-net is more likely to select both. Ridge's stability under rotation can be partially inherited by Elastic-Net as a result of the trade-off between Lasso and Ridge. The mathematical problem of ElasticNet is formulated in Equation (4.4).

$$\min_{w} \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha\rho \|w\|_1 + \frac{\alpha(1-\rho)}{2} \|w\|_2^2 . \tag{4.4}$$

4.3 Decision Tree

A decision tree is a popular ML algorithm for regression and classification tasks [12]. It models decisions and their possible consequences in a tree-like structure. The tree is constructed through recursive partitioning of the input space into progressively smaller regions, using a set of if-then-else rules associated with each node. At each node, a decision is made based on the values of one of the input features, and the data is split into subsets according to this decision. The process halts when a specific stopping criterion is met, such as reaching a maximum tree depth or having a minimum number of samples in each leaf node.

The goal of building a decision tree is to find the most informative features and the best splitting criteria at each node. This is typically done using a measure of impurity, such as Gini impurity

or entropy. The impurity measures the degree of disorder in a set of samples, and the split that minimizes impurity is chosen. This process is repeated recursively for each subset of data until the stopping criterion is met.

The resulting decision tree can be used for prediction by traversing the tree from the root node to a leaf node, following the decision rules at each node based on the feature values of the input. The leaf node reached represents the predicted class label in a classification task or the predicted value in a regression task.

4.4 Random Forest

Random Forests (RF) is an ensemble algorithm that combines multiple decision trees to make predictions [12]. It is based on the idea of bagging (bootstrap aggregating) and feature randomness. Bagging involves training multiple decision trees on different subsets of the training data, sample selection is independent, and the selected sample is returned to the dataset before the following selection is made. Each decision tree is trained independently, and during prediction, the results of all trees are combined to make a final prediction.

RF introduces an additional level of randomness through the random selection of a subset of features at each node of the decision tree. This random feature selection ensures that different trees are built using different subsets of features, reducing the correlation between the trees and leading to more diverse predictions.

To make predictions using RF, each decision tree in the ensemble independently predicts the output and the final prediction is the result of combining the different decision tree predictions, often through majority voting for classification or averaging for regression.

RF has several advantages, including being able to handle high-dimensional data, capturing complex interactions between features, and providing estimates of feature importance. RF tends to be robust, have high accuracy, and perform well on large datasets.

4.5 Support Vector Machine (SVM)

Support Vector Machines (SVM) is a machine learning algorithm used for classification and regression analysis [54]. It is based on the concept of finding the best separating hyperplane that maximizes the margin between the two classes in the data. SVM works by mapping the input data into a high-dimensional feature space, where a linear decision boundary can be found.

SVM can be used for both linear and non-linear classification problems, and it is especially effective when dealing with high-dimensional data. It is commonly used in areas such as image classification, text classification, and bioinformatics.

The main statistics used in SVM are the distance between data points and the margin between the classes. SVM uses a kernel function to map the data points into a higher dimensional space, where the margin between the classes can be maximized. The choice of kernel function can have a significant impact on the performance of the SVM algorithm.

Support Vector Regression(SVR)

In this section we present an implementation of Support Vector Regression (SVR) for time series forecasting. The proposed method involves transforming time series data into a supervised learning problem, followed by training and evaluating the SVR Model.
Support Vector Regression, shortened to SVR is a supervised ML algorithm used for regression tasks [54]. It's a branch-off from Support Vector Machine (SVM) which is primarily designed and used for classification problems. The main purpose of SVR is to find a function that gives a best approximation for the relationships between input features and target values while trying to minimize the prediction error.

How does it work?

Support Vector Regression or SVR works by fitting a hyperplane that best represents the data so that errors between predicted valued and actual values are minimized [54]. This approach has some similarities to linear regression algorithms their approach to errors differ. With linear regression the approach is to minimize the sum of squared errors. SVR on the hand has as aim to keep errors within a certain margin/threshold. This is known as an epsilon-sensitive tube. SVR has as an objective to find a balance between between maximizing the margin width (The distance between hyperplane and support vector), while also minimizing penalized errors.

Summarizing this one can see that Support Vector is a ML algorithm that is useful for various regression tasks with it's support for various kernel functions such as linear, polynomial, radial basis and sigmoid. This allows for one to model complex and nonlinear relationships between input features and target values.

4.6 Ensemble Learning Model

Ensemble learning is a powerful technique in machine learning that involves combining multiple estimators or models to improve the accuracy and generalizability of machine learning models. As a result of aggregating the predictions ensemble method learns a meta-estimator [36]. By leveraging the strengths of several models, we can create a more accurate and robust solution that is much better than what any individual model could achieve on its own. There are different types of ensemble methods, such as Bagging, Boosting, and Stacking, each with its own approach to combining the models' predictions.

In Bagging (Figure 2.1) we want to train multiple instances of the same model on different subsets of the data and combine their predictions using voting or averaging.

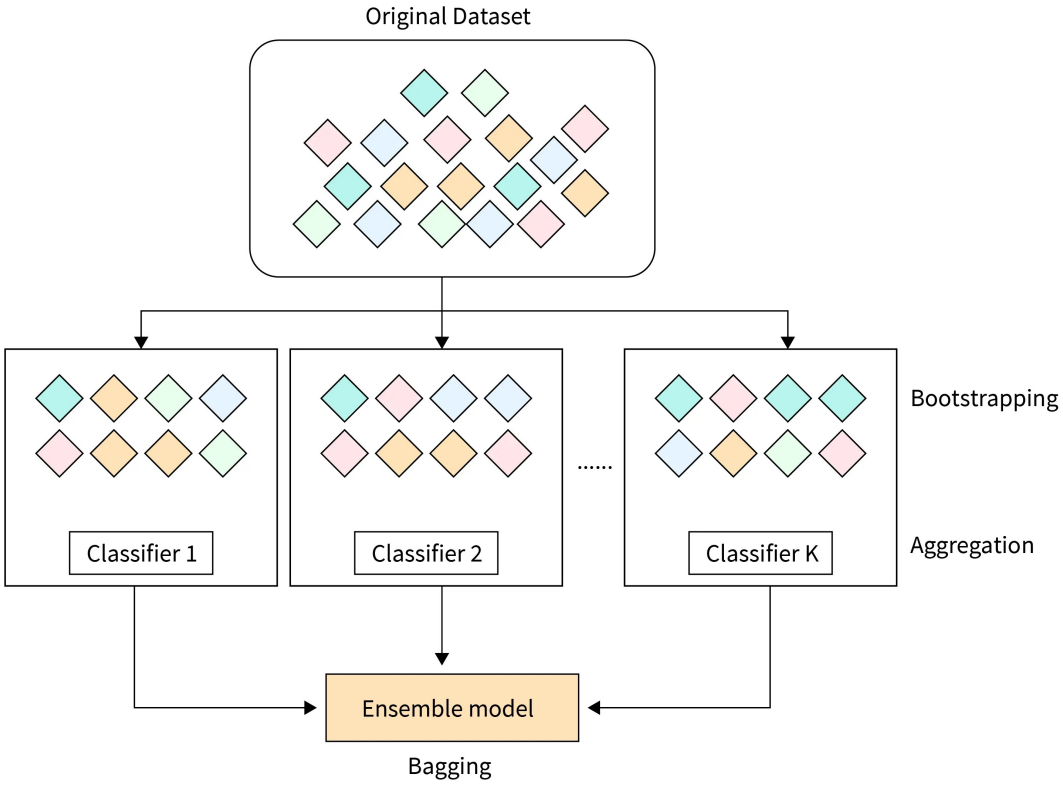

Figure 4.1: Bagging Steps

Boosting involves training a sequence of weak models and combining them with each one focusing on the errors made by the previous models (Figure 2.2). There are different types of boosting algorithms, such as AdaBoost and Gradient Boosting.In boosting weak models are used as base models, which are also called weak learners. Weak learners in a binary classification context would be a model predicting slightly better than random chance, also more than 50% accuracy.

As Gautam Kunapuli [36] states, “It turns out that “garnering wisdom from a council of fools” works surprisingly well in machine learning. This is the underlying motivation for sequential ensembles of weak learners.”

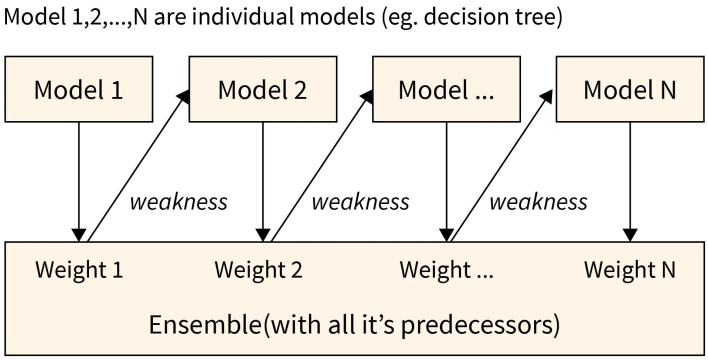

Figure 4.2: Boosting Steps

Stacking trains several models and combines their predictions using a meta-model/learner (Figure2.3). With stacking, we train several models to solve similar problems and use their combined output to build a new model with improved performance. Ensemble learning is especially effective in situations where individual models have high variance, meaning their predictions can be highly sensitive to small changes in the input data.

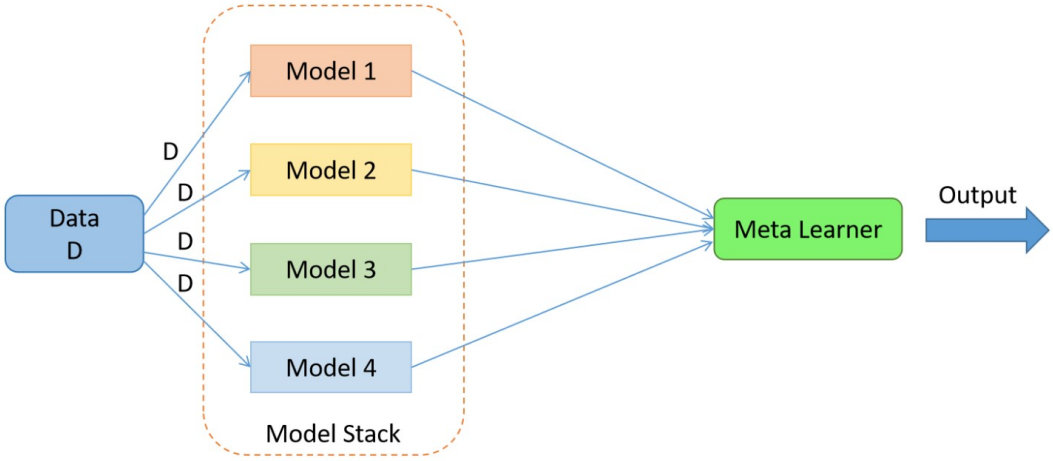

Figure 4.3: Stacking Steps

4.7 Gradient Boosting

Gradient boosting is an ensemble learning technique used for both classification and regression problems. It works by seeking to find an additive model that minimizes a given loss function, such as the squared error for regression, using a weak learner such as regression trees [36]. The loss function will output how bad your model predictions are, if the predictions are perfect then the output is zero. The ensemble of models in gradient boosting is typically referred to as an additive

model because each new model does not disturb the functions created by the previous models but instead imparts its own information to bring down the errors. This allows gradient boosting to create highly accurate predictive models that can capture complex patterns and relationships in the data.

First, the algorithm makes an initial guess of the response, for example, by taking the mean of the response variable in the regression. Then, the gradient or the residual (Example shown in Figure 2.4) of the loss function is calculated, and a new weak learner is trained to predict these residuals. Weak learners are weak models that barely can perform prediction and need improvements.

Then the current model is added to the previous model, and the process continues with a new iteration, where the residuals are recalculated based on the combined model. This iterative process continues until a user-specified number of iterations is reached. The key idea behind gradient boosting is that it corrects the errors made by the previous weak learners by adding a new weak learner at each iteration. This is similar to AdaBoost, another ensemble method that also trains a new weak estimator at each iteration to improve the performance of the previous estimator.

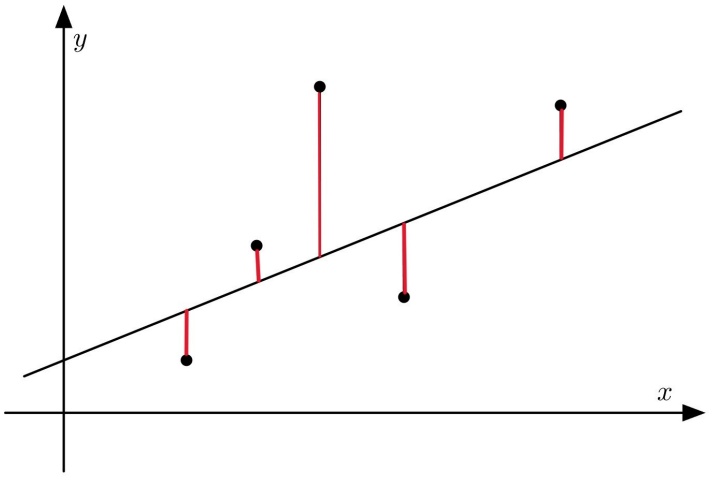

Figure 4.4: Residuals are marked in red color for a linear regression prediction.

4.8 XGBoost

XGBoost (Extreme Gradient Boosting) is a popular machine learning algorithm that is used for classification and regression tasks. It is an implementation of gradient boosting. The basic idea behind XGBoost is to iteratively train a series of decision trees on the data, where each subsequent tree attempts to correct the errors of the previous tree.

Specifically, XGBoost builds an initial tree and then computes the errors (residuals) of each data point. The second tree is then built to predict the residuals of the first tree, and so on. In each iteration, the tree is built to minimize the overall error of the previous trees.

4.8.1 Math behind Gradient Boosting in XGBoost

Suppose we have a dataset with two features (X_1 and X_2) and a target variable y. Our goal is to build a regression model that can predict y based on the features X_1 and X_2.

Step 1: Initialize the model

In the first step, we initialize the model by setting ($F_0 = 0$) . This means that our initial prediction for y is always 0. We can represent this as:

$$F_0(x) = 0 \tag{4.5}$$

Step 2: Fit a new model to the residuals

In the second step, we fit a new model to the residuals of the previous model. The residual (r_1) is the difference between the actual target value y and the predicted value of the previous model (F_0):

$$r_1 = y - F_0(x) \tag{4.6}$$

We can then fit a new model h_1 to the residuals r_1. Let's say we choose a decision tree as our base learner for h1, and we fit this tree to the residuals r_1. The output of h_1 is the predicted residuals, which we can represent as:

$$h_1(x) = \text{predicted residuals} \tag{4.7}$$

Step 3: Update the model with the new information

In the third step, we update the model with the new information from h_1. We do this by adding h_1 to F_0:

$$F_1(x) = F_0(x) + h_1(x) \tag{4.8}$$

The updated model $F_1(x)$ now predicts y more accurately than $F_0(x)$ because it takes into account the residuals that $F_0(x)$ was not able to capture. We can repeat this process for any number of iterations until we achieve the desired level of accuracy. For a better understanding of the mathematics behind gradient boosting, we look at one example. Suppose we have the following dataset:

x_1	x_2	y
1	2	3
2	3	5
3	4	7
4	5	9
5	6	11

Table 4.1: Example Dataset

Step 1: Initialize the model

$$F_0(x) = 0 \tag{4.9}$$

Step 2: Fit a new model to the residuals, we calculate the residuals r_1 as follows:

$$r_1 = y - F_0(x) = [3,5,7,9,11] - 0 = [3,5,7,9,11] \tag{4.10}$$

We then fit a decision tree h1 to the residuals r_1:

$$h_1(x) = \text{predicted residuals} \tag{4.11}$$

We can represent the output of h_1 in the table 4.2:

x_1	x_2	predicted residuals
1	2	1
2	3	2
3	4	2
4	5	2
5	6	2

Table 4.2: Decision tree $h1$ output

Step 3: Update the model with the new information: We update our model by adding h_1 to F_0:

$$F_1(x) = F_0(x) + h_1(x) \tag{4.12}$$

We can represent F1(x) as follows:

x_1	x_2	$F_1(x)$
1	2	1
2		

Table 4.3: Updated model F(x)

4.8.2 Key features in XGBoost

One of the key strengths of XGBoost is its ability to handle large datasets with a large number of features and to effectively deal with missing data. It achieves this by using a variety of techniques, such as regularization, to prevent overfitting and reduce the impact of noisy or irrelevant features.

- **Faster tree construction :**

  XGBoost constructs decision trees using an approximate greedy algorithm which is essentially a technique for selecting the best-split point in a decision tree in a computationally efficient manner. Traditional decision tree algorithms exhaustively search all possible split points for

each feature, which can be computationally expensive for datasets with many features and large amounts of data. XGBoost's approximate greedy algorithm is designed to be faster and more efficient by only considering a subset of the possible split points. Specifically, the algorithm first sorts the data points for each feature and then considers only a discrete set of split points at the quantiles of the sorted values. This reduces the search space and allows for faster computation [16]. However, because the algorithm only considers a subset of possible split points, it may not always find the optimal split. To mitigate this issue, XGBoost uses a technique called 'second-order approximation' that uses a Taylor expansion of the loss function to estimate the gain of each possible split point. This approximation allows XGBoost to evaluate the quality of each possible split point and select the best one quickly and accurately. This technique allows XGBoost to build decision trees much faster than traditional algorithms while still achieving high accuracy.

- **Regularization :**

  In ML, regularization is a technique that is used to prevent a model from overfitting and improve its general performance. XGBoost, a popular algorithm used in ML, implements two types of regularization techniques: L1 regularization (also known as Lasso regularization) and L2 regularization (also known as Ridge regularization). To prevent overfitting in its decision trees, XGBoost combines these two regularization techniques. The amount of regularization applied can be controlled by adjusting the hyperparameters lambda (for L2 regularization) and alpha (for L1 regularization) in the XGBoost algorithm. To find the best values for these hyperparameters, you can use cross-validation or other hyperparameter tuning techniques [16].

  L1 and L2 both add a penalty term to the loss function. A loss function in ML is a mathematical function that measures the difference between the predicted output of a model and the actual output. The main objective is to reduce the value of the loss function. This can also help reduce the complexity of the model and prevent overfitting.

  In L1 regularization penalty term is proportional to the absolute value of the model's coefficients. This penalty encourages the model to select only the most important features and set the coefficients of less important features to zero. On the other hand, L2 regularization implements a penalty term that is proportional to the square of the model's coefficients. This encourages the model to spread the coefficient values across all the features, instead of relying heavily on just a few features.

- **Parallel processing :**

  Parallel processing is an important capability of XGBoost that enables the algorithm to leverage multiple processors and distributed computing resources. It dramatically speeds up the training process on large datasets, which can otherwise be very time consuming and computationally demanding. To achieve parallel processing, XGBoost splits the training process into multiple parallel jobs that can be run simultaneously on different cores or machines. This approach distributes the workload across multiple processors, allowing XGBoost to handle large datasets much more efficiently [16].

  XGBoost uses a technique called parallel tree learning to implement parallel processing, where the algorithm trains multiple decision trees in parallel, with each tree using a subset of the training data. Once all the trees are trained, they are combined to form the final ensemble model. To configure the parallel processing feature, XGBoost provides various

parameters, such as the number of threads to use and the type of parallelism to use, such as multi-node, multi-GPU, or distributed processing. The optimal configuration depends on the specific hardware and dataset used, and it can be determined through experimentation and benchmarking.

- **Handling missing data :**

  Missing values present many challenges since ML algorithms require complete data to predict properly. XGBoost is different than other traditional ML algorithms, It has a built-in feature that allows it to handle missing data effectively [74].

  For a dataset containing missing values, XGBoost assigns each missing value to a default direction in the decision tree. During the tree construction process, the algorithm evaluates the direction of the split and assigns the missing value to the left or right child node of the split based on whether the split condition is true or false for the missing value. XGBoost also includes a parameter called "missing" that allows the user to specify a custom value that represents missing data in the dataset. This can be useful when the missing data is represented by a value other than NaN or Null.

  One important thing to mention is that XGBoost does not impute missing values. Instead, it treats missing values as a separate category and incorporates them into the decision-making process. This means that missing data can potentially have a significant impact on the model's performance, and it's essential to carefully consider how to handle missing data before using XGBoost.

- **Early stopping :**

  Early stopping is a technique used in XGBoost to prevent overfitting and improve the efficiency of the training process. The idea behind early stopping is to monitor the performance of the model during the training process and stop the training when the performance on a validation dataset stops improving. In XGBoost, early stopping is implemented by specifying a validation dataset (or multiple validation datasets) using the eval-set parameter during training.

  The eval-metric parameter is also specified to determine the metric used for evaluation (e.g., RMSE for regression problems, AUC (Area Under the ROC Curve) for binary classification problems) [73]. During the training process, XGBoost periodically evaluates the performance of the model on the validation dataset using the specified evaluation metric. If the performance on the validation dataset does not improve for a specified number of rounds (determined by the early-stopping-rounds parameter), the training process is stopped, and the model with the best performance on the validation dataset is returned.

  By using early stopping, XGBoost can prevent overfitting and save computation time by stopping the training process early if the model has already reached its optimal performance. This can lead to better generalization performance on unseen data and more efficient training of the model.

4.8.3 XGBoost's parameters

Here are the hyperparameters for XGBoost [75] along with a brief explanation of each:

- **learning_rate**: Controls the step size in the gradient descent optimization algorithm. A lower learning rate may improve model accuracy but will require more training iterations.

- **max_depth**: Controls the maximum depth of a tree. A deeper tree can model more complex relationships, but may overfit the training data.
- **n_estimators:**

  The number of boosting iterations to perform. More iterations will generally lead to a better model, but will also increase training time.
- **subsample:** The fraction of training instances to use for each boosting iteration. This can be used to prevent overfitting and improve model generalization.
- **colsample_bytree:** The fraction of features to use for each tree. This can be used to reduce overfitting and improve model generalization.
- **gamma:** Controls the minimum loss reduction required to make a split. A higher value will lead to fewer splits and a simpler tree, but may underfit the data.
- **reg_alpha:** L1 regularization term on weights. Can be used to prevent overfitting and encourage sparsity.
- **reg_lambda:** L2 regularization term on weights. Can be used to prevent overfitting and encourage generalization.
- **min_child_weight:** Controls the minimum sum of instance weight (hessian) needed in a child. This can be used to prevent overfitting and improve model generalization
- **max_delta_step:** Maximum delta step we allow each tree's weight estimation to be. This can be used to prevent overfitting.
- **tree_method:** The algorithm used to build decision trees. This can be set to 'auto', 'exact', 'approx', or 'hist'. The default is 'auto', which will choose the best algorithm based on the data and other parameters.
- **scale_pos_weight:** Controls the balance of positive and negative weights in the dataset. This can be used for imbalanced classification problems.
- **objective:** The loss function to be minimized. Common options include 'reg:squarederror' for regression problems and 'binary:logistic' for binary classification problems.
- **scale-pos-weight :** The evaluation metric used to measure model performance. Common options include 'rmse' for regression problems and 'logloss' for classification problems
- **scale-pos-weight :** If the validation metric does not improve after this many rounds, training will stop. This can be used to prevent overfitting and speed up training.

These are the most commonly used hyperparameters in XGBoost, but there are many others that can be adjusted for specific use cases.

4.9 How To Select An Algorithm

Identify the problem: First, identify the type of problem you are trying to solve. Is it a classification or regression problem? Is it a supervised or unsupervised learning problem?

Analyze the data: Next, analyze the characteristics of the data. Consider the size of the dataset, the number of features, and the distribution of the data. Some algorithms may perform better on large datasets, while others may be more suited to datasets with fewer features.

Identify the algorithm options: Based on the type of problem and data characteristics, identify the algorithms that are appropriate for the task. For example, if you are working on a classification problem, you might consider logistic regression, decision trees, random forests, or support vector machines.

Evaluate the algorithms: Once you have identified the possible algorithms, evaluate their performance on the data. This can involve testing the algorithms on a subset of the data and comparing the results. You can also use cross-validation to get a more accurate assessment of performance.

Choose the best algorithm: Finally, choose the algorithm that performs best on the data. Keep in mind that the best algorithm may not always be the most complex or the newest one available. It's important to choose an algorithm that is both accurate and interpretable.

4.10 Evaluation Metrics

Overview of the evaluation metrics used to determine the precision of our machine learning models:

Mean Squared Error (MSE) is a metric used in regression problems to measure the average squared difference between the predicted and actual values [54]. It is a good metric to use when the target variable has a continuous range of values.

$$MSE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y})_i^2. \tag{4.13}$$

Root Mean Squared Error (RMSE) is a popular metric used to evaluate regression models [54]. It measures the square root of the average squared difference between the predicted and actual values. Like MSE, RMSE measures the deviation of predicted values from actual values, but it is expressed in the same units as the target variable. The lower the RMSE, the better the model is at predicting the target variable.

$$RMSE(y,\hat{y}) = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y})_i^2}. \tag{4.14}$$

Mean Absolute Error (MAE) is a metric that measures the absolute difference between the predicted and actual values [54]. MAE is similar to RMSE, but it is less sensitive to outliers because it measures the absolute difference instead of the squared difference. The lower the MAE, the better the model is at predicting the target variable.

$$MAE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|. \tag{4.15}$$

Mean Absolute Percentage Error (MAPE) is a metric that measure the average absolute percentage difference between the predicted and actual value [54]. MAPE can produce infinite or undefined values if the actual value is 0. A smaller MAPE indicates a more accurate forecasting method.

$$MAPE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{|y_i - \hat{y}_i|}{max(\varepsilon, |y_i|)}. \tag{4.16}$$

4.11 Summary Machine Learning Algorithms

In this chapter, we have given an explanation of how each of the algorithms LR, RF, SVR and XGBoost used in our study works, how they are selected and what metrics is used to evaluate their performance. This should have provided a better understanding of how they are implemented and what makes them the best suited options for our data as well as how it effects their performance.

Chapter 5

Data Acquisition

Dataset acquisition refers to the process of obtaining data from various sources for use in data analysis, machine learning or other data-related tasks. This process involves the collection of data from different sources like databases, sensors, experiments or other primary sources.

The creation of the dataset involves using using the raw data gathered through the acquisition in the previous step to create a new dataset. Dataset creation require defining the scope, variables, formats, as well as ensuring the data is representative and relevant for the intended use case. This chapter represents what we did to create a dataset that is representative, relevant and has a high degree of accuracy of the current situation in the marked.

As part of this research project, the goal was to analyze energy price patterns over a period. Initially, the client had indicated that they would provide the necessary data for analysis. However, as time went on, it became apparent that we would need to create the dataset ourselves.

One of the main challenges we faced was that the data was not readily available in a single dataset. Instead, we had to combine data from multiple sources available in entso-e, a European energy transmission system operator. Careful selection and curation of data was necessary to ensure that it met the project's requirements.

Furthermore, the data was not immediately fit for analysis. We had to preprocess and clean it to ensure that it was of high quality and suitable for our needs. This involved addressing issues such as missing data, inconsistencies, and errors.

The resulting dataset consists of 26,304 rows and 57 columns. It represents a valuable resource that will be analyzed in the following chapters to generate insights and support our research findings.

Creating this dataset was a challenging task that required a great deal of effort and attention to detail. However, it was an essential step towards achieving the project's goals. The data would not have been available without combining multiple datasets and preprocessing, which demonstrates the importance of careful data collection and preparation in research projects such as this one.

5.1 Dataset Importance

Properly collected, organized and labeled datasets for training high-performance ML models are a scarce resource. Large technology companies in Norway tend to have access to abundant energy market data which is not open source and therefore not available for our thesis. The unavailability of data can possibly diminish a model's accuracy and prediction dynamics. Many aspects of a ML algorithm can be influenced by choice of the dataset, e.g. fraction of test, train and validate data. The test-train ratio affects the algorithm's performance and its robustness [69].

Creating a high-quality dataset is a critical step in any ML project. The responsibility of collecting data typically falls on the data science team. And it involves collaboration with other stakeholders such as domain experts, data analysts, and IT professionals. It is worth mentioning that none of us has any domain expertise about the day-ahead electricity market in Norway and Europe. If data collection and acquisition are not done correctly, it can lead to severe consequences for our project. Poor-quality data can result in inaccurate and unreliable models that fail to produce good results.

The challenges of our data acquisition are many. At the beginning of our thesis, we were told that our client would provide the dataset needed. However, three weeks later we were informed that data acquisition was a part of our thesis. We had to adjust to a new plan and make progress with the data acquisition. Below are some data quality aspects we took into consideration :

- **Quality:**

  Quality is the most important property of any given dataset. ML models will learn and execute algorithms based on the data we feed them. Any data imperfection such as imbalanced, underrepresented, and inconsistent data may result in biased and useless models [57]. Imbalanced data is often associated with classification techniques when some classes are over-represented which causes the model to overlook or discard the minority classes.

  When collecting data for a machine learning regression model, it's important to clearly understand the problem you're trying to solve and determine which variables are relevant to the problem. Feature values in a dataset should be consistent and must be available for all instances/data points. To remove any errors, inconsistencies, or missing values after collecting data, we need to perform data cleaning, data normalization, feature selection, and removing anomalies and missing values. All these processes are completed in chapters 5 and 6 in this report.

- **Quantity:** Determining the right volume of data required for our ML and DL models is a difficult task for us, our solution is to collect as much data as possible. At first, we chose only the year 2022. After reading several research and master thesis reports on forecasting electricity prices, and trying to follow their direction, we understood our data is insufficient. Based on our research, the recommended training data for a forecasting time series model is often one year. Our data in total was one year, our research suggested the need for at least three years of data (2020-2022) for each bidding zone.

- **Authorization & Credibility:** Relying on published research on a similar problem statement is always helpful. On the other hand, usually there is a lack of transparency for authorization and credibility around datasets. Often, datasets are restricted and difficult to access, or there is no access to information on the reliability of the author(s) and the methodology employed in the research.

- **Documentation:** Evaluating a dataset is done by studying the dataset documentation. Most datasets have a list of definitions, ranges of valid values, standard formats, and methodology descriptions to explain why and how these particular records are collected. Documentation can help to decide whether the data is fit for the problem or not.

5.2 Data Source

The data used in the time series data prediction was collected from The European Network of Transmission System Operators for Electricity (ENTSO-E)[21] - shown in Figure 5.1. ENTSO-E is the association for the cooperation of the European transmission system operators (TSOs)[22], and has developed a platform to enable transparent sharing of data with all participants within the European transmission system. ENTSO-E collects and shares data about electricity generation, transportation and consumption in the pan-European market. ENTSO-E's role within the market lends credibility to the quality of the data. Since the data is provided by the same institution, the collection frequency of the data is mostly standardized which reduces the amount of reprocessing needed for the aggregation of the dataset.

Figure 5.1: ENTSO-E

5.3 Dataset Selection

Datasets were gathered from zone NO1 (Norway Østlandet) and the neighboring zones NO2, NO3, NO5, and SE3, since these are the zones with the most impact on zone NO1 as they have a direct cable connection to zone NO1 as seen in figure below. These datasets span three years from the start of 2020 to the end of 2022. The selection of features was based on SIN's guidance.

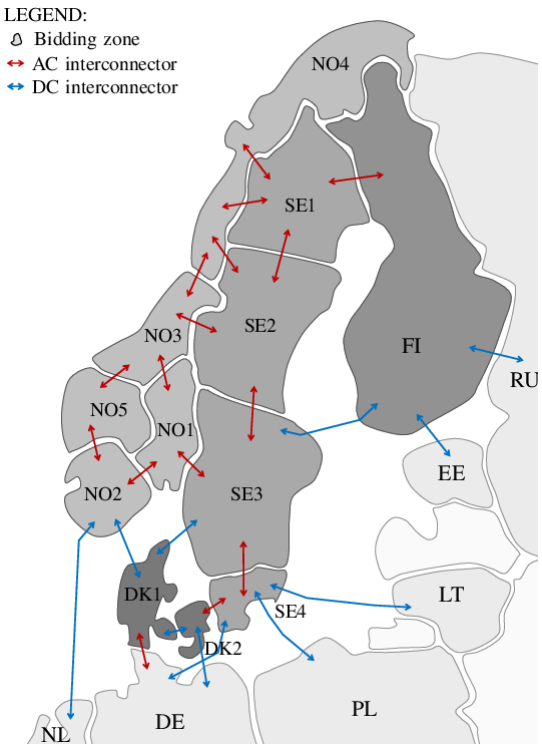

Figure 5.2: Map of interconnectors in scandinavia

5.4 Price Datasets

The price dataset contains pricing data of spot (day-ahead) market prices in the currency Euro per megawatt-hour (MWh) [**?**].

	MTU (UTC)	Day-ahead Price [EUR/MWh]	Currency	BZN|NO1
0	01.01.2020 00:00 - 01.01.2020 01:00	31.77	EUR	NaN
1	01.01.2020 01:00 - 01.01.2020 02:00	31.57	EUR	NaN
2	01.01.2020 02:00 - 01.01.2020 03:00	31.28	EUR	NaN
3	01.01.2020 03:00 - 01.01.2020 04:00	30.72	EUR	NaN
4	01.01.2020 04:00 - 01.01.2020 05:00	30.27	EUR	NaN

Figure 5.3: Table of original price dataset for zone NO1.

Figure 5.3 displays the first five rows of the original price dataset for zone NO1. The price datasets contains the four columns; "MTU (UTC)", "Day-ahead Price [EUR/MWh]", "Currency", and "BZN|NO1" (zone). The "MTU (UTC)" column holds the datetime range represented by two datetimes separated by a dash symbol. The "Day-ahead Price [EUR/MWh]" column contains the currency in the column name. The "Currency" column contains the currency unit of the day-ahead

price. The column "BZN|NO1" represents the bidding zone of the day-ahead price, but the rows in this columns only contain Not a Number values (NaN values).

	MTU (UTC)	Day-ahead Price [EUR/MWh] BZN|NO1	Day-ahead Price [EUR/MWh] BZN|NO2	Day-ahead Price [EUR/MWh] BZN|NO3	Day-ahead Price [EUR/MWh] BZN|NO5	Day-ahead Price [EUR/MWh] BZN|SE3
0	01.01.2020 00:00 - 01.01.2020 01:00	31.77	31.77	28.45	31.77	28.45
1	01.01.2020 01:00 - 01.01.2020 02:00	31.57	31.57	27.90	31.57	27.90
2	01.01.2020 02:00 - 01.01.2020 03:00	31.28	31.28	27.52	31.28	27.52
3	01.01.2020 03:00 - 01.01.2020 04:00	30.72	30.72	27.54	30.72	27.54
4	01.01.2020 04:00 - 01.01.2020 05:00	30.27	30.27	26.55	30.27	26.55

Figure 5.4: Table of processed and aggregated price datasets from all zones.

Figure 5.4 displays the first five rows of the processed and aggregated dataset of day-ahead prices in all zones. The rows containing the currency unit and the bidding zone has been dropped, and the information has been added to the column name of the day-ahead price. This was done to remove redundant information and to reduce the amount of data that ML algorithms have to process when training on this data. The datasets were merged based on the values of the time column "Time (UTC)" which aligns all measurements of day-ahead price from different zones to the corresponding time.

5.5 Load Datasets

The load dataset contains data about the power consumption in each zone per MW., and has 3 columns[23]. Figure 5.5 displays the first five rows of the original load dataset for zone NO1. The load dataset cointains the three columns; "Time (UTC)", "Day-ahead Total Load Forecast [MW] - BZN|NO1" and "Actual Total Load [MW] - BZN|NO1".

	Time (UTC)	Day-ahead Total Load Forecast [MW] - BZN|NO1	Actual Total Load [MW] - BZN|NO1
0	01.01.2020 00:00 - 01.01.2020 01:00	4316.0	4333.0
1	01.01.2020 01:00 - 01.01.2020 02:00	4270.0	4250.0
2	01.01.2020 02:00 - 01.01.2020 03:00	4209.0	4167.0
3	01.01.2020 03:00 - 01.01.2020 04:00	4214.0	4145.0
4	01.01.2020 04:00 - 01.01.2020 05:00	4250.0	4222.0

Figure 5.5: Table of original total load dataset for zone NO1.

Figure 5.6 displays the first 5 rows of the processed and aggregated dataset of load data for all zones. The column for day-ahead forecast of load has been dropped, because the access to the actual load. The datasets were merged based on the values of the time column "Time (UTC)" which aligns all measurements of load from different zones to the corresponding time.

MTU (UTC)	Actual Total Load [MW] - BZN|NO1	Actual Total Load [MW] - BZN|NO2	Actual Total Load [MW] - BZN|NO3	Actual Total Load [MW] - BZN|NO5	Actual Total Load [MW] - BZN|SE3
01.01.2020 00:00 - 01.01.2020 01:00	4333.0	4139.0	3016.0	1950.0	9350
01.01.2020 01:00 - 01.01.2020 02:00	4250.0	4114.0	2945.0	1926.0	9158
01.01.2020 02:00 - 01.01.2020 03:00	4167.0	4030.0	3032.0	1913.0	8980
01.01.2020 03:00 - 01.01.2020 04:00	4145.0	4032.0	2988.0	1894.0	8914
01.01.2020 04:00 - 01.01.2020 05:00	4222.0	4032.0	2848.0	1901.0	8964

Figure 5.6: Table of processed and aggregated total load datasets from all zones.

5.6 Energy Generation Datasets

The energy generation datasets include energy production by each production type in MW. The datasets for each zone have 23 columns where "Area" is the zone. "MTU" is the time interval of the generation period. The 21 other columns contains the distinct energy production types. However, each zone does not generate energy from every production type. Therefore we drop the columns which do not have any data or "n/e" (not expected). For the columns which have data, but have rows that contain "n/e", we replace "n/e" with 0. In the actual generation dataset we keep; "Fossil Gas", "Hydro Run-of-river and poundage", "Hydro Water Reservoir", "Other", "Waste", and "Wind Onshore" from all of the Norwegian zones. Bidding zones NO2, NO3 and NO5 include "Hydro Pumped Storage Aggregated" in addition. While NO3 is the only zone containing data in "Other renewable". In the Swedish zone SE3 the columns containing data is: "Fossil Gas", "Hydro Water Reservoir", "Nuclear", "Other", "Solar", and "Wind Onshore". For each bidding zone we simultaneously drop "Area" and include the zone in the column headers as it is easier to compare in the aggregated dataset. The aggregated actual generation dataset contains 34 columns and 26304 rows.

	Area	MTU	Biomass - Actual Aggregated [MW]	Fossil Brown coal/Lignite - Actual Aggregated [MW]	Fossil Coal-derived gas - Actual Aggregated [MW]	Fossil Gas - Actual Aggregated [MW]	Fossil Hard coal - Actual Aggregated [MW]	Fossil Oil - Actual Aggregated [MW]	Fossil Oil shale - Actual Aggregated [MW]	Fossil Peat - Actual Aggregated [MW]	...	Hydro Run-of-river and poundage - Actual Aggregated [MW]	Hydro Water Reservoir - Actual Aggregated [MW]
0	BZN|NO1	01.01.2022 00:00 - 01.01.2022 01:00 (UTC)	0	n/e	n/e	0	n/e	n/e	n/e	n/e	...	878	321
1	BZN|NO1	01.01.2022 01:00 - 01.01.2022 02:00 (UTC)	0	n/e	n/e	0	n/e	n/e	n/e	n/e	...	897	330
2	BZN|NO1	01.01.2022 02:00 - 01.01.2022 03:00 (UTC)	0	n/e	n/e	0	n/e	n/e	n/e	n/e	...	885	275
3	BZN|NO1	01.01.2022 03:00 - 01.01.2022 04:00 (UTC)	0	n/e	n/e	0	n/e	n/e	n/e	n/e	...	890	269
4	BZN|NO1	01.01.2022 04:00 - 01.01.2022 05:00 (UTC)	0	n/e	n/e	0	n/e	n/e	n/e	n/e	...	892	300

Figure 5.7: Table of original actual generation dataset for BZN|NO1.

	MTU (UTC)	Fossil Gas - BZN|NO1	Hydro Run-of-river and poundage - BZN|NO1	Hydro Water Reservoir - BZN|NO1	Other - BZN|NO1	Waste - BZN|NO1	Wind Onshore - BZN|NO1	Fossil Gas - BZN|NO2	Hydro Pumped Storage Aggregated - BZN|NO2	Hydro Run-of-river and poundage - BZN|NO2	...	Hydro Run-of-river and poundage - BZN|NO5	Hydro Water Reservoir - BZN|NO5	Other - BZN|NO5
0	01.01.2020 00:00 - 01.01.2020 01:00	25.0	727.0	1130.0	0.0	0.0	149.0	4.0	0.0	317.0	...	53.0	3553.0	0.0
1	01.01.2020 01:00 - 01.01.2020 02:00	25.0	729.0	1115.0	0.0	0.0	153.0	4.0	0.0	316.0	...	52.0	3326.0	0.0
2	01.01.2020 02:00 - 01.01.2020 03:00	25.0	731.0	1087.0	0.0	0.0	153.0	4.0	0.0	316.0	...	52.0	2915.0	0.0
3	01.01.2020 03:00 - 01.01.2020 04:00	25.0	731.0	1058.0	0.0	0.0	153.0	4.0	0.0	314.0	...	53.0	2594.0	0.0
4	01.01.2020 04:00 - 01.01.2020 05:00	25.0	731.0	1030.0	0.0	0.0	155.0	4.0	0.0	313.0	...	52.0	2392.0	0.0

Figure 5.8: Table of processed and aggregated actual generation datasets from all zones.

5.7 Import And Export Datasets

The cross-border physical flow dataset represents import and export [MW] for each bidding zone at a given hour (UTC time standard). In order to make it clear that features in this dataset represent the

	Time (UTC)	BZN|NO2 > BZN|NO1 [MW]	BZN|NO1 > BZN|NO2 [MW]
0	01.01.2022 00:00 - 01.01.2022 01:00	1242.0	0.0
1	01.01.2022 01:00 - 01.01.2022 02:00	1222.0	0.0
2	01.01.2022 02:00 - 01.01.2022 03:00	1679.0	0.0
3	01.01.2022 03:00 - 01.01.2022 04:00	1668.0	0.0
4	01.01.2022 04:00 - 01.01.2022 05:00	1697.0	0.0

Figure 5.9: Table of original cross-border physical flow dataset

cross-border physical flow from a zone to another, "CBF" is used as an abbreviation in the feature names. Then "BZN|NOX > BZN|NOY [MW]" was renamed to "CBF BZN|NOX > BZN|NOY [MW]", where X and Y mean respectively the zone exporting from and the zone exporting to. The next step was to merge datasets (NO1 and neighboring zones) for the years 2020-2022 based on the column "MTU (UTC)".

	MTU (UTC)	CBF BZN|NO2 > BZN|NO1 [MW]	CBF BZN|NO1 > BZN|NO2 [MW]	CBF BZN|NO3 > BZN|NO1 [MW]	CBF BZN|NO1 > BZN|NO3 [MW]	CBF BZN|NO5 > BZN|NO1 [MW]	CBF BZN|NO1 > BZN|NO5 [MW]	CBF BZN|SE3 > BZN|NO1 [MW]	CBF BZN|NO1 > BZN|SE3 [MW]
0	01.01.2022 00:00 - 01.01.2022 01:00	1242.0	0.0	371.0	0.0	594.0	0.0	1065	0
1	01.01.2022 01:00 - 01.01.2022 02:00	1222.0	0.0	383.0	0.0	676.0	0.0	927	0
2	01.01.2022 02:00 - 01.01.2022 03:00	1679.0	0.0	362.0	0.0	436.0	0.0	748	0
3	01.01.2022 03:00 - 01.01.2022 04:00	1668.0	0.0	324.0	0.0	345.0	0.0	918	0
4	01.01.2022 04:00 - 01.01.2022 05:00	1697.0	0.0	339.0	0.0	412.0	0.0	826	0

Figure 5.10: Table of aggregated Cross-border physical flow dataset

5.8 Water Reservoirs And Hydro Storage Plants Datasets

The dataset contains an aggregated weekly average filling rate of all water reservoir and hydro storage plants in the form of MWh (Megawatt hour) per bidding zone [24].

The data for each zone consists two columns. Column 1 denotes the week of the year and column 2 comprises of "Stored Energy Value Water Reservoirs and Hydro Storage Plants[MWh]" with the included zone [24].

Due to the data being measured at a frequency of weeks and not hourly frequency like the other data being used in this project there is now a need to perform resampling and interpolation to predict the values that fall within the range of the datapoints we have. Further measures to make the data more accurate there is also need to add the datapoint from week 52 of 2019 and week 1 of 2023.

Afterwards we create a range of number according to the amount of weeks in the datasets from week 52 of 2019 to week 1 of 2023. This amounts to 159 weeks and as such we will generate a column with 159 rows of int values in range of 1-159.

This a copy of this range of numbers will be added to the dataset of different zones and will be used as the "Week_Number" instead of the previous "Week" column which will be dropped. The datasets of the different zones are then merged on the basis of their "Week_Number"

Next step is to generate a range of dates in the frequency of 7 days from one point to the next one with the conditions of starting date being "2019/12/23" and end date "2023/01/02". This will create 159 data points that we can concatenate with the merged date of the different zones. These ranges indicate the start date of the weeks of the data and the new column will be called "Week_start_date"

Next step is to perform an interpolation through re-sampling by doing an up-sampling. This is done by increasing the frequency of the data of the data to hourly which will generate all new rows of data between the data points of "Week_start_date" and fill them with NaN(Not a Number) with the first filled row being "2020-01-01 00:00:00". Next point is to predict the values that fall within the range of datapoints we have available. This is done through a numerical analysis called "Cubic Hermite Spline Interpolation".
Next step is to "cut" the tails so that the data starts at "2020-01-01 00:00:00" and ends at "2022-12-31 23:00:00"

5.9 Summary Data Acquisition

In summary, this Chapter discusses the process of acquiring and creating a dataset that is representative, relevant, and has a high degree of accuracy of the situation in the electricity market. The importance of curated and labeled datasets for training high-performance ML models is highlighted, along with challenges in our data acquisition, such as determining the right volume of data, authorization and credibility issues, data quality, and structure.

The data source used in this thesis is the European Network of Transmission System Operators for Electricity (ENTSO-E), which collects and shares data about electricity generation, transportation, and consumption in the pan-European market. The dataset is collected from zone NO1 and neighboring zones: NO2, NO3, NO5, and SE3, which have the most impact on zone NO1. The dataset spans three years from the start of 2020 to the end of 2022, and features were selected based on SIN's guidance.

The dataset contains five types of data: price datasets, load datasets, energy generation datasets, import/export datasets, and water reservoirs and hydro storage plants datasets. The price dataset contains pricing data of spot (day-ahead) market prices in the currency Euro per MWh. The load datasets contains data about the power consumption in each zone per MW. The energy generation datasets include energy production by each production type in MW. The import/export datasets contain data about the import and export between the selected zones. The water reservoirs and hydro storage plants datasets contain data about the average filling rate of all water reservoirs and hydro storage plants in MWh. The datasets were processed and aggregated based on the values of

the time column "Time (UTC)". The Final dataset for zone NO1 and the neighbouring zones has 26304 rows (instances) and 57 columns(attributes) (two of which is datetime for start and end of time period).

Chapter 6

Exploratory Data Analysis

The goal of the Exploratory Data Analysis (EDA) is to make sure that the data is well understood, cleaned, and preprocessed so that the data is ready for model training. This involves gaining insight into the structure, distribution, feature correlation, and missing values. This information is the basis of data preprocessing, data cleaning, and feature selection.

6.1 Dataset Structure

The dataset is a time series dataset consisting of 26304 observations of the target feature "Day-ahead Price" in zone NO1, and 57 features that can be used to train ML models to predict the electricity price in zone NO1. The observations are on an hourly interval and span from the start of 2020 to the end of 2022. The features can be sorted into 15 feature categories that are generally shared amongst the five bidding zones with some exceptions:

- **Start/end MTU (UTC):** Start or end date and time of the interval for the measured values in the UTC time zone.
- **Day-ahead Price:** The day-ahead price of electricity measured in the currency Euro per megawatt-hour (MWh) for the specified bidding zone.
- **Actual Total Load:** The total power consumption of the specified zone per megawatt (MW).
- **Fossil Gas:** Electricity production measured in MW from power plants using fossil gas to generate electricity for the specified bidding zone.
- **Wind Onshore:** Electricity production measured in MW from sources using wind to harness energy on land (onshore as opposed to offshore) for the specified bidding zone.
- **Hydro Water Reservoir:** Electricity production measured in MW from hydro water reservoirs for the specified bidding zone. "Hydro water reservoirs" are hydroelectric power plants where the water used in electricity production comes from water stored in a reservoir above the power plant.
- **hydro-pumped Storage Aggregated:** Electricity production in MW from hydro-pumped Storage Aggregated for the specified bidding zone. "Hydro-pumped storage" is hydroelectric generation from plants where water is pumped from a reservoir below the plant to a reservoir above the plant, and is used for load balancing on the power grid.

- **Hydro Run-of-river and Poundage:** Electricity production measured in MW from Hydro Run-of-river and poundage for the specified bidding zone. "Run-of-the-river" is hydroelectric generation from plants without any significant water storage. "Poundage" is hydroelectric generation from plants with no water storage.
- **Solar:** Electricity production measured in MW from sources using radiant light and/or heat from the sun to harness energy for the specified bidding zone.
- **Nuclear:** Electricity production measured in MW from nuclear power plants in the specified bidding zone.
- **Other:** Electricity production measured in MW from sources that do not fall under any of the other electricity sources in the dataset for the specified bidding zone.
- **Other Renewable:** Electricity production measured in MW from sources that do not fall under any of the other electricity sources in the dataset and that are considered renewable electricity production for the specified bidding zone.
- **Waste:** Electricity production measured in MW from sources that use waste to produce electricity.
- **Cross-border Physical Flow of Electricity (CBF BZN|x > BZN|y):** Cross-border physical flow of electricity in MWh, from zone x to zone y measured in MWh. Because the focus of this dataset is bidding zone NO1, all cross-border physical flow of electricity features involve zone NO1 and the feature category can therefore be divided into two sub-categories:
  - **Import to Zone NO1 from Neighbouring Zones.**
  - **Export from Zone NO1 to Neighbouring Zones.**
- **Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh]:** Stored energy in water reservoirs and hydro storage plants measured in MWh.

6.2 Feature Trend Investigation

In this section, we analyze the graphed trends by each feature category.

6.2.1 Day-ahead price

Figure 6.1 displays the day-ahead price of electricity for each bidding zone measured in the currency Euro per MWh on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend line for the different zones aligns with each other with reasonably similar peak and trough prices. Figure 6.1 illustrates how the price increases from the start of 2020 to the end of 2022 with significant shifts to higher prices between each year. Furthermore, the fluctuation in the prices increases significantly over time, with a significant shift to higher price volatility each year. The Shifts to higher volatility align with the shift to higher prices. The large difference in the data between the different years may lead to significantly worse performance from a model trained on this data. Dropping the data from years 2020 and 2021 might therefore be considered to increase the potential performance of any model trained with this data.

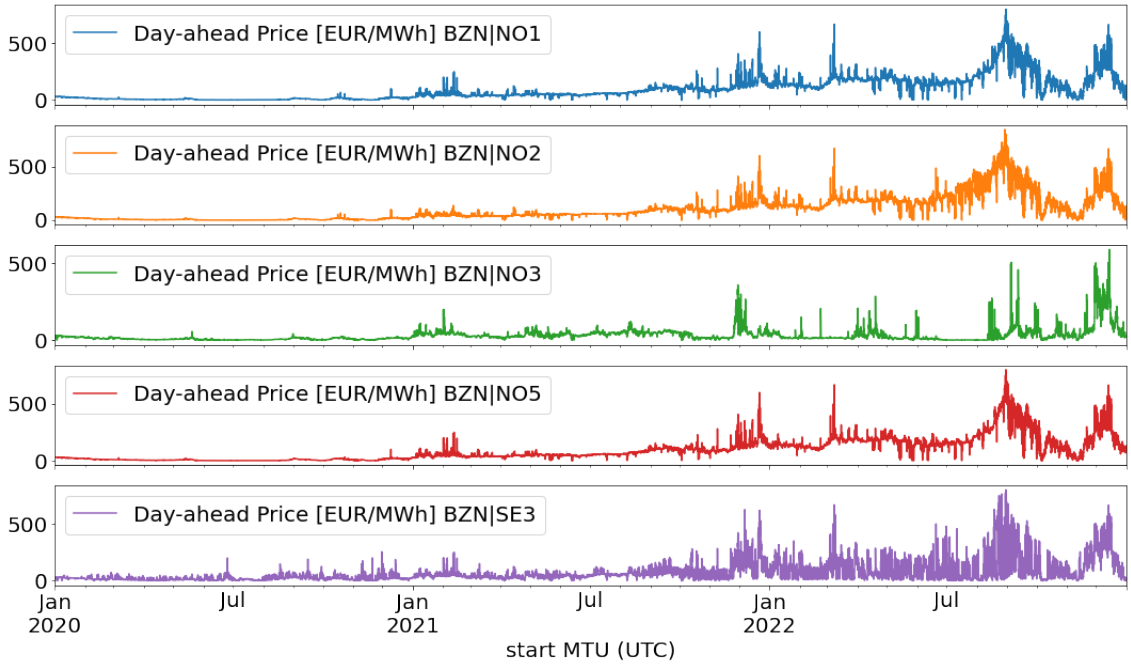

Figure 6.1: Day-ahead price for all zones.

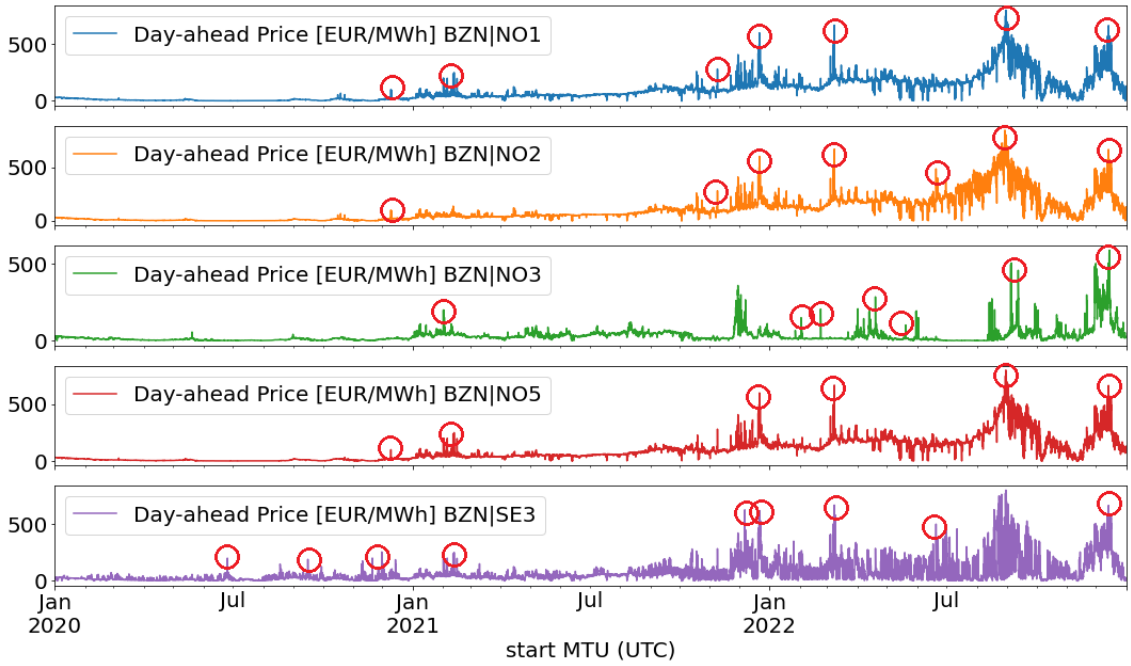

Figure 6.2: Day-ahead price for all zones with some of the outliers highlighted.

Figure 6.2 highlights some of the price spikes from Figure 6.1. These spikes are outliers that can significantly alter the statistics of the trend line by functioning as noise and might reduce the performance of a model trained on this data. These outliers should be handled before proceeding to succeeding parts of the EDA.

6.2.2 Total load

Figure 6.3 displays the Total load measured in MW for each bidding zone on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend line for the different zones aligns with each other and shows seasonality; peaking at the start of each year and bottoming in the middle of the year. The bidding zone with the highest load is SE3 peaking at 15000 MW followed by NO1 peaking at 7500 MW, NO2 at 7000 MW, NO3 at 4000 MW, and NO5 at 3000 MW. Lastly, there are some outliers in the trend lines that should be handled.

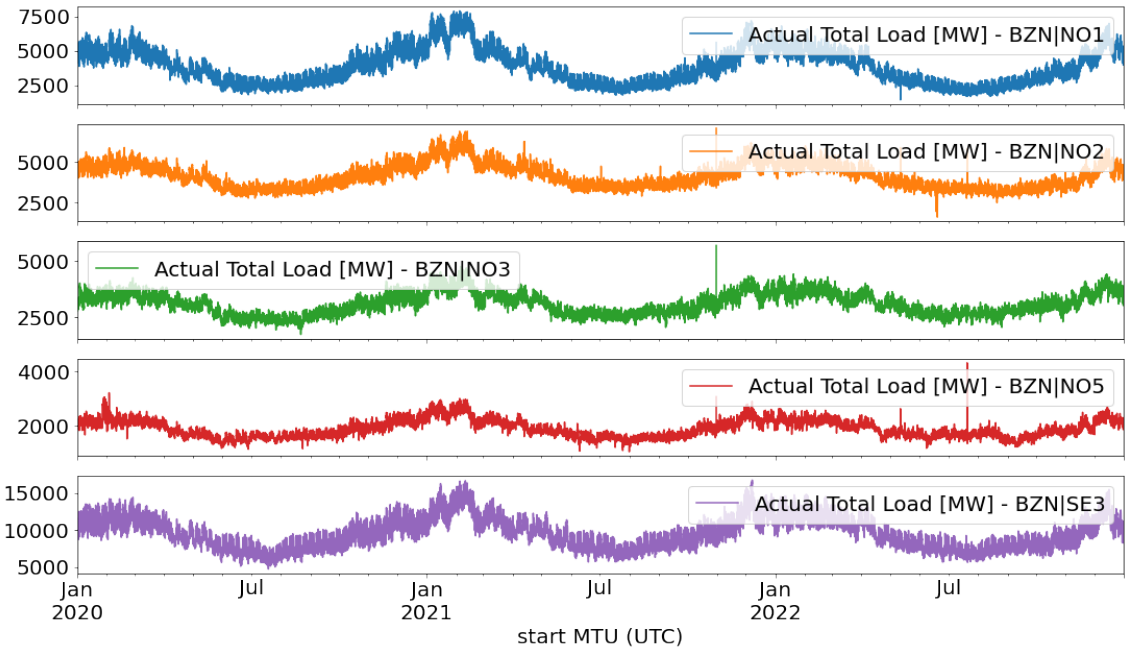

Figure 6.3: Total load all zones.

6.2.3 Electricity production stemming from fossil gas

Figure 6.4 displays the electricity production stemming from fossil gas for each bidding zone measured in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. A lot of the fossil gas electricity production data is missing from bidding zones NO1, NO2, NO3, and SE3. The missing values are too many to be handled by imputing new values, and will therefore have a low variance in the data. The low variance introduces little useful information to the model and will function as noise which reduces the performance of any model trained on the data. The fossil gas feature for bidding zones NO1, NO2, NO3, and SE3 should therefore be removed from the dataset during feature selection.

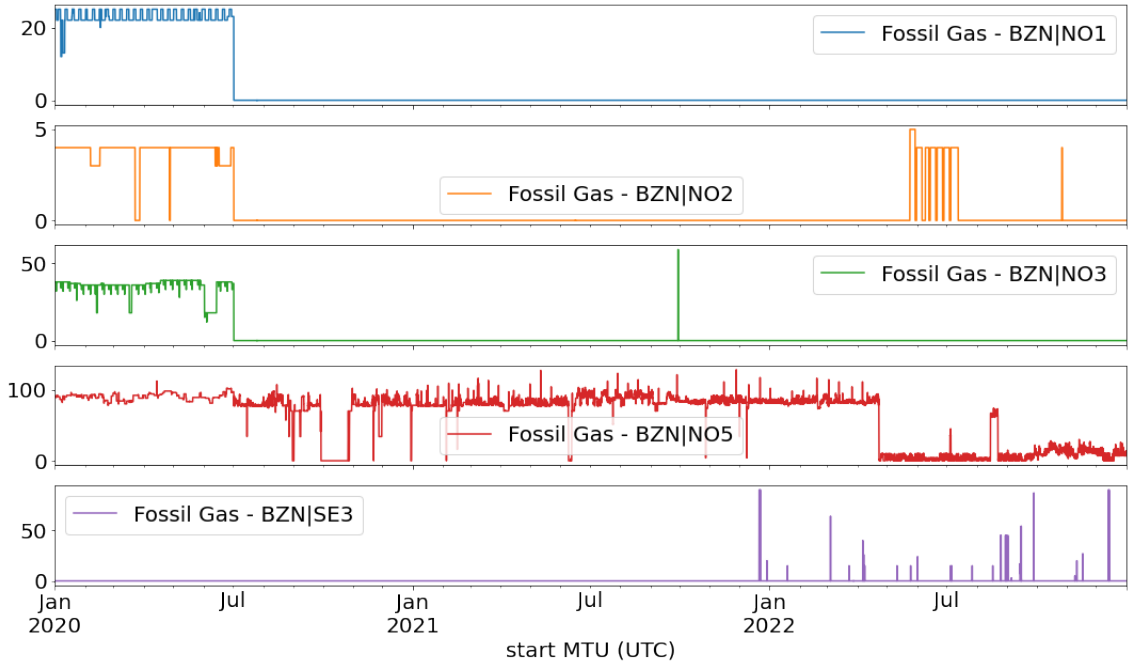

Figure 6.4: Electricity production stemming from fossil gas for all zones.

6.2.4 Electricity production stemming from wind onshore

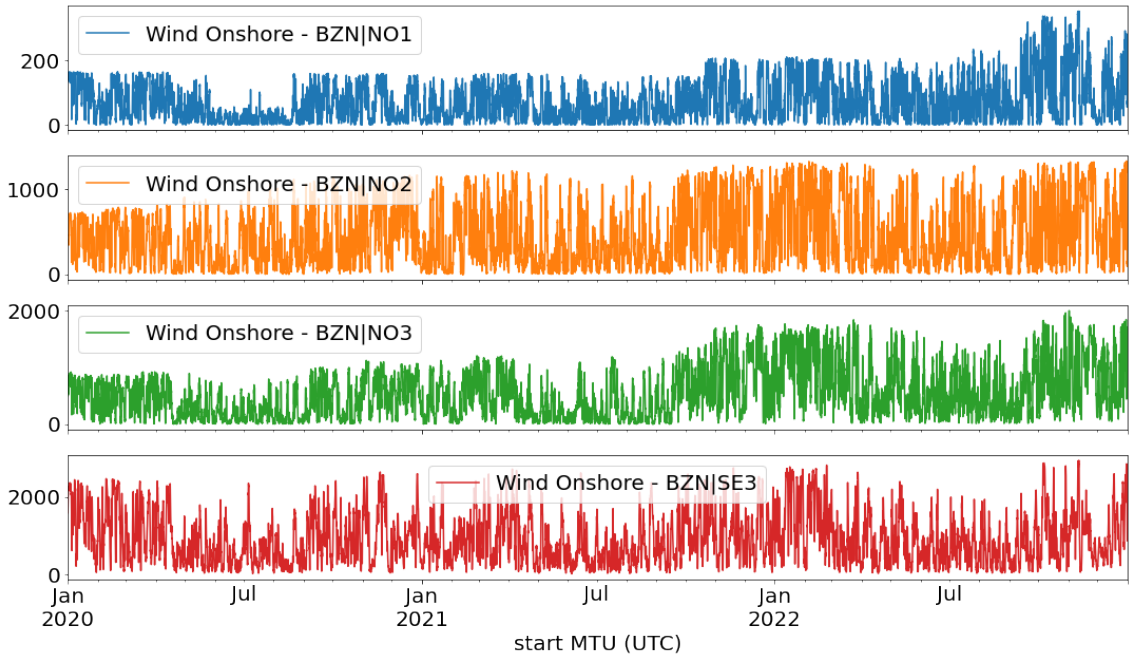

Figure 6.5: Electricity production stemming from wind onshore all zones.

Figure 6.5 displays the electricity production stemming from wind onshore for each bidding zone measured in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones vaguely align with each other, and the variation in production is very high for all zones. Wind production is highest in zones NO3 and SE3 which both peak at around 2000 MW, followed by NO2 at 1000 MW and NO1 at 350 MW.

6.2.5 Electricity production stemming from hydro water reservoir

Figure 6.6 displays the electricity production stemming from the hydro water reservoir for each bidding zone in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones vaguely align with each other. Bidding zone NO2 has the highest production peaking at about 10000 MW followed by NO5 at 6000 MW, NO3 at 3000 MW, NO1 at 2000 MW, and SE3 at 1800 MW. The production data for SE3 is missing pre-2022, therefore either the pre-2022 data should be dropped or the feature should be dropped during feature selection.

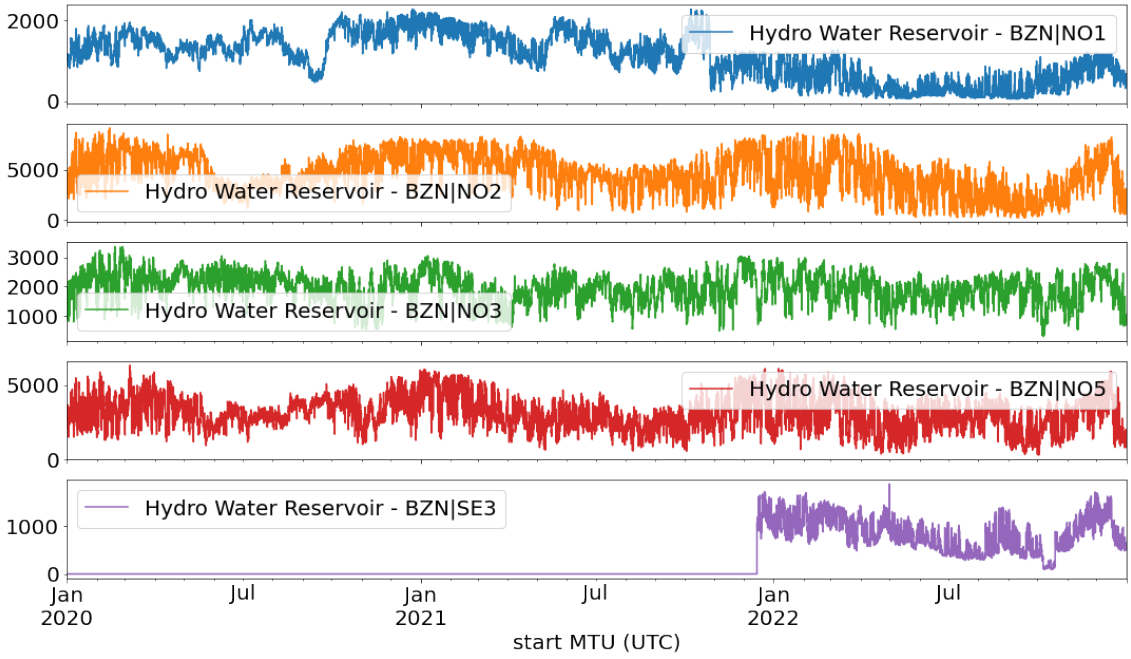

Figure 6.6: Electricity production stemming from hydro water reservoir for all zones.

6.2.6 Electricity production stemming from hydro-pumped storage aggregated

Figure 6.7 displays the electricity production stemming from hydro-pumped storage aggregated for each bidding zone in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones vaguely align with each other, and the variation in production is very high for all zones. The production is highest in zone NO2 peaking at 1000 MW followed by NO5 at 600 MW, and NO3 at 200 MW. The

data pre-July 2020 is missing, which means that either the pre-2021 data has to be dropped or the features have to be dropped during feature selection.

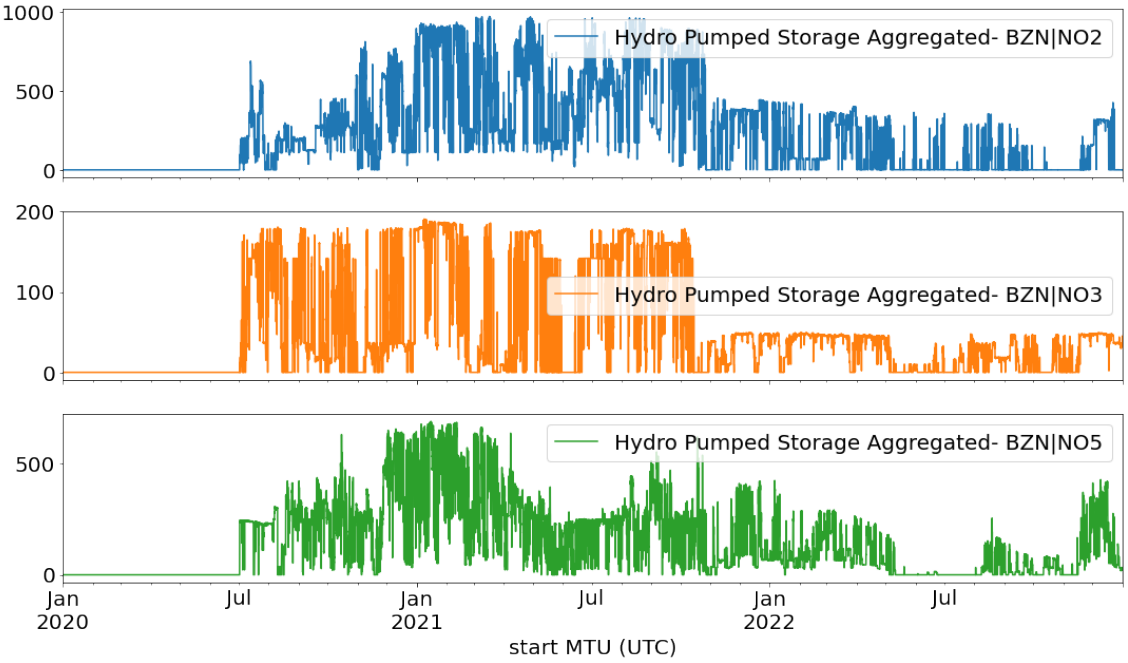

Figure 6.7: Electricity production stemming from hydro-pumped storage aggregated for all zones.

6.2.7 Electricity production stemming from hydro run-of-river and poundage

Figure 6.8 displays the electricity production stemming from hydro run-of-river and poundage for each bidding zone in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones match each other. The production is highest in zone NO1 which peaks at 2000 MW followed by zones NO2 and NO3 at 1000 MW and NO5 at 500MW.

6.2.8 Electricity production stemming from solar

Figure 6.9 displays the electricity production stemming from solar for bidding zone SE3 in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The data is missing pre-2022, therefore either the feature should be dropped under feature selection or the data pre-2022 should be dropped. Since the dataset only contains data for one year, we can not identify seasonality; however, the production seems to be higher during the summer months peaking during summer at 400 MW and being at its lowest mid-winter. This is reasonable since there is less sun in Nordic countries during the winter months.

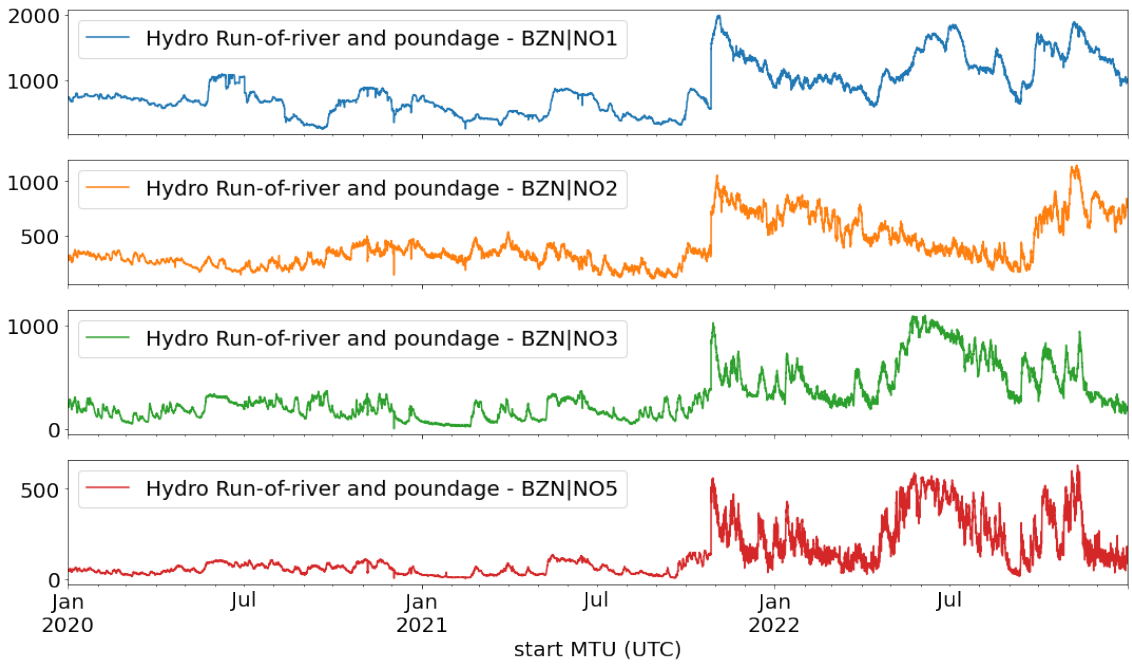

Figure 6.8: Electricity production stemming from hydro run-of-river and poundage for all zones.

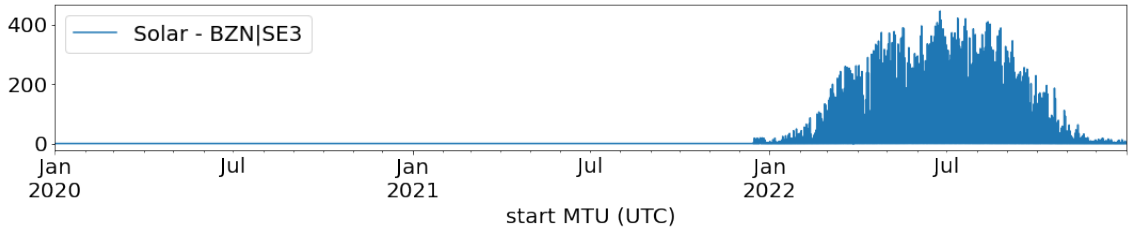

Figure 6.9: Electricity production stemming from solar zone SE3.

6.2.9 Electricity production stemming from nuclear

Figure 6.10 displays the electricity production stemming from nuclear for bidding zone SE3 in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. Energy production from "nuclear" in zone SE3 is missing before 2022. Either the data before 2022 should be dropped, or the feature should be dropped during feature selection. The data also contains an outlier which increases the peak value from around 7000 MW to 1000 MW. This outlier should be handled to reduce its impact on models trained using this data.

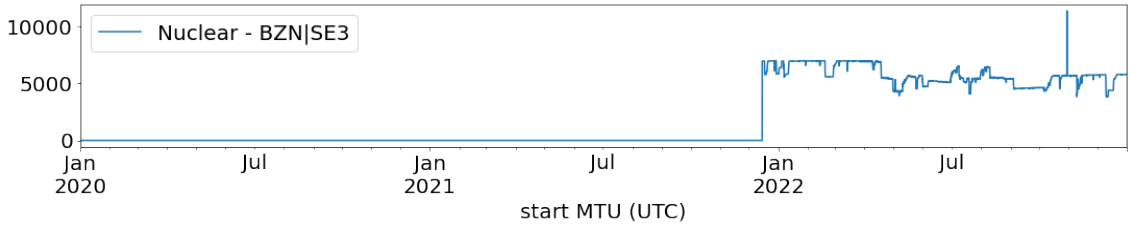

Figure 6.10: Electricity production stemming from nuclear zone SE3.

6.2.10 Electricity production stemming from other and other renewable

Figure 6.11 displays the electricity production stemming from "other" and "other renewable" for each bidding zone in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. Most of the data from the "other" feature is missing from zone NO1, NO2, NO3, and NO5, and this data should therefore be dropped. The bidding zone SE3 has data for the feature "other" for the year 2022, and can therefore be kept if only the data from 2022 is used. The peak production from "other" for bidding zone SE3 is 1200 MW.

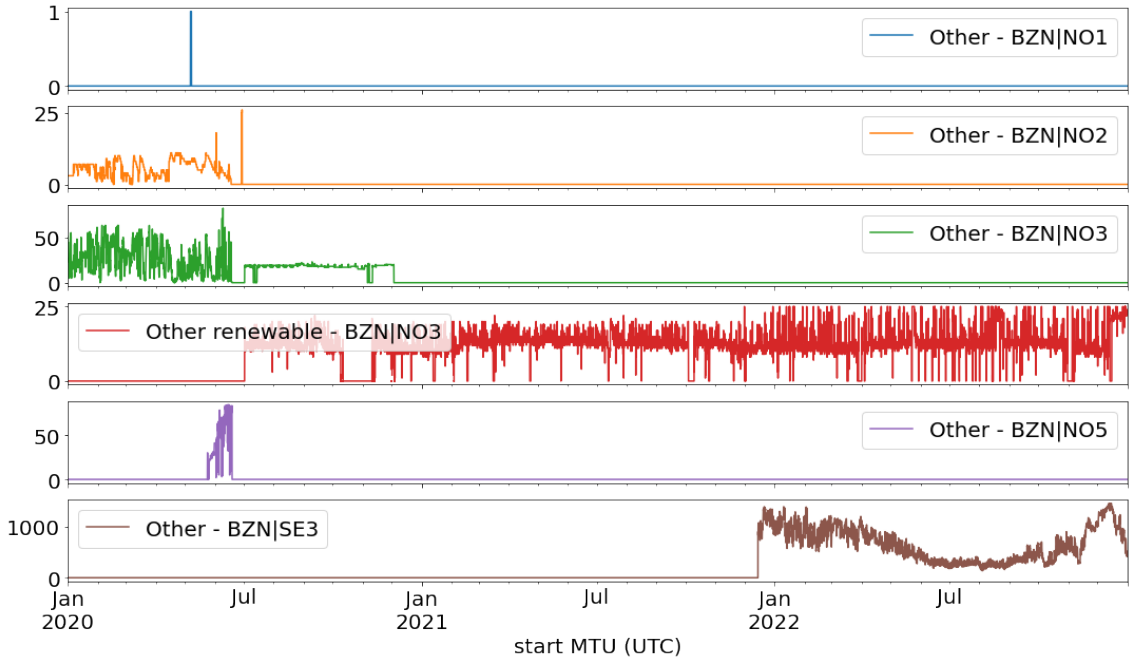

Figure 6.11: Electricity production stemming from "other" and "other renewable" all zones.

For the feature "other renewable" there is only data for bidding zone NO3. The data before mid-2020 is missing, therefore the data should only be included if the data for 2020 is dropped from the dataset. The feature also contains a lot of outliers which should be handled to reduce the noise trained by models using this data. The peak production from "other renewable" for bidding zone NO3 is 25 MW, which amounts to 2% of the production from "other" for bidding zone SE3.

6.2.11 Electricity production stemming from waste

Figure 6.12 displays the electricity production stemming from waste for each bidding zone in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines for bidding zones NO2 and NO5 align with each other. The remaining bidding zones do not have trend lines that align with each other. The data for the electricity production from waste is missing before mid-2020 for all zones, and for the first half of 2021 for zone NO1. Either the years including the missing values have to be dropped or the features must be dropped during feature selection. The bidding zone with the highest production is zone NO5 peaking at 20 MW followed by NO2 at 15 MW, and zones NO1 and NO3 at 5 MW. The data also contains outliers which will need to be handled.

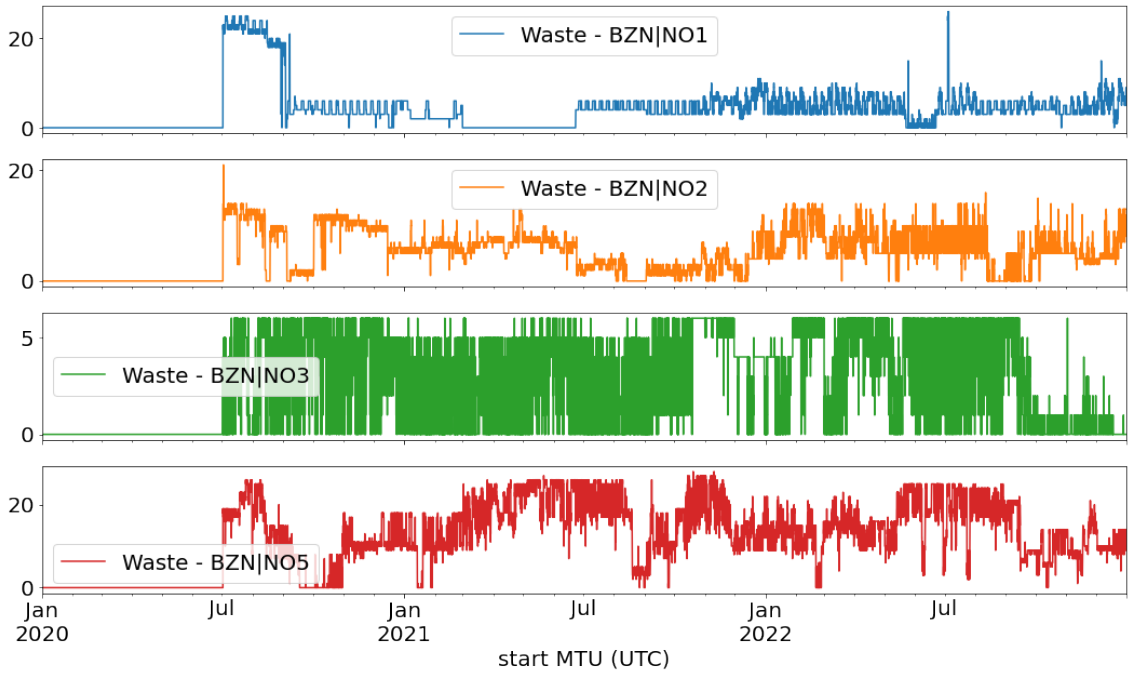

Figure 6.12: Electricity production stemming from waste all zones.

6.2.12 Cross-border physical flow of electricity

Figure 6.13 displays the cross-border physical flow of electricity from each of the neighboring zones to NO1 in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones vaguely align with each other, and the variation in the cross-border physical flow of electricity is very high for all zones. Zone NO1 is missing all values for the first months of 2020, which means that the data for 2020 should be dropped, or the feature should be dropped during feature selection. The bidding zone with the highest cross-border physical flow to zone NO1 is NO5 which peaks at 3000 MW followed by NO2 at 2500 MW, SE3 at 2000 and NO3 at 800 MW. The data also contain outliers which should be handled.

Import to zone NO1 from neighbouring zones

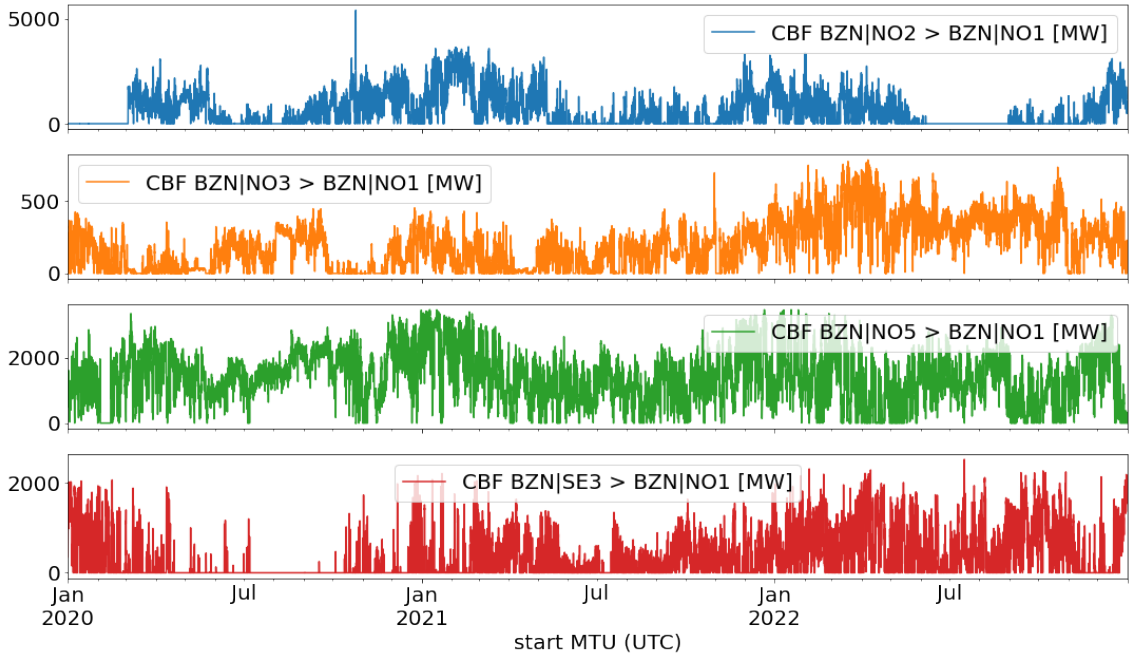

Figure 6.13: Import to zone NO1 from neighbouring zones.

Export from zone NO1 to neighbouring zones

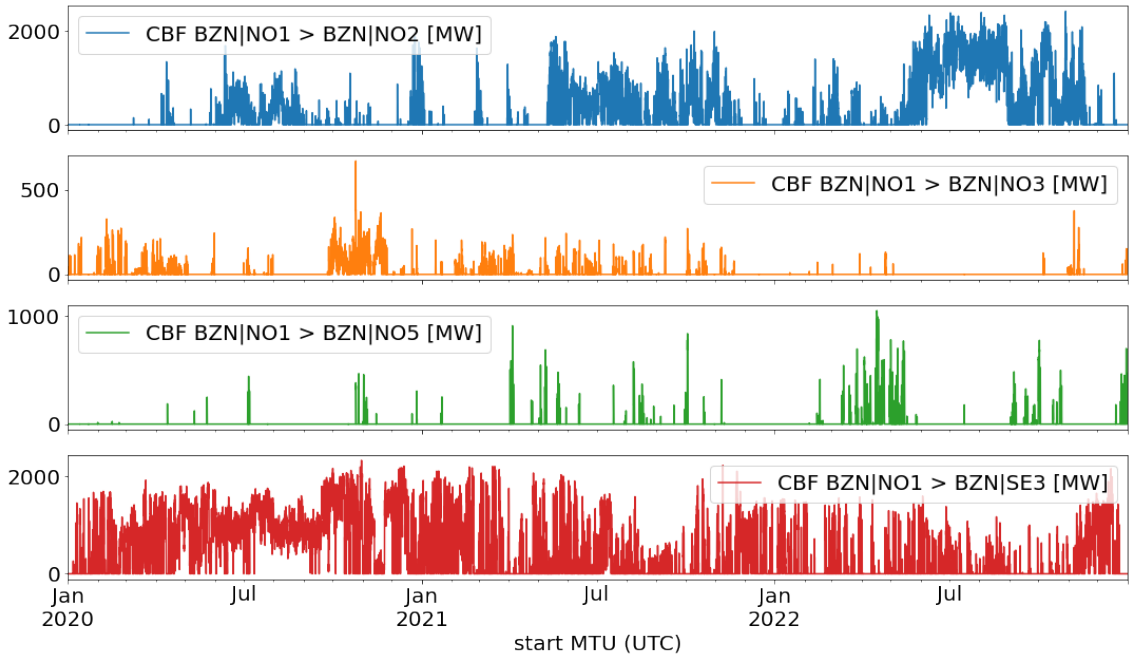

Figure 6.14: Export from zone NO1 to neighbouring zones.

Figure 6.14 displays the cross-border physical flow of electricity from NO1 to each of the neighboring zones in MW on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. Most of the electricity from zone NO1 is exported to zones SE3 and NO2 with peaks at around 2000 MW, followed by NO5 at around 1000 MW, and NO3 at 250 MW. The data for export to bidding zone NO3 contains a spike that doubles the maximum value of that graph, it should therefore be handled to reduce the impact it has on models training on the data.

6.2.13 Stored energy value in water reservoirs and hydro storage plants

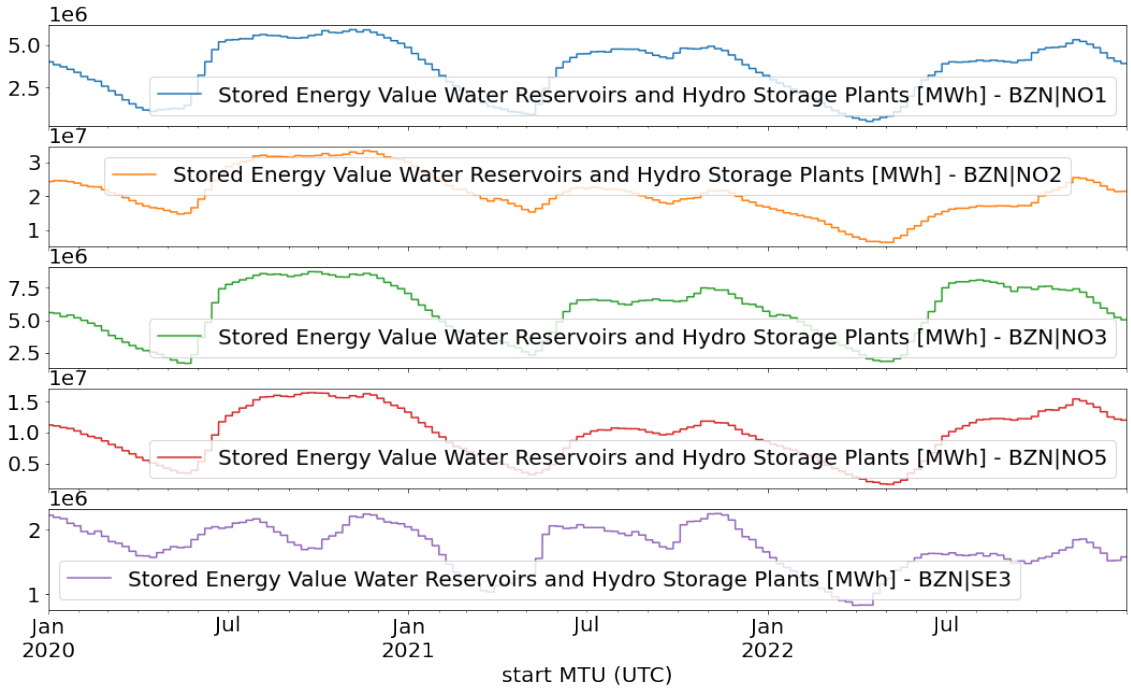

Figure 6.15: Stored energy value in water reservoirs and hydro storage plants all zones.

Figure 6.15 displays the stored energy value in water reservoirs and hydro storage plants for each bidding zone in MWh on the y-axis, and the date and time of measurement on the x-axis over the period from the start of 2020 to the end of 2022. The trend lines from the different zones vaguely align with each other, and have a stepping pattern caused by the values being measured with a weekly interval. The zone with the highest amount of stored energy is NO3 with a peak of 7.5 MWh followed by NO1 at 5 MWh, NO2 at 3 MWh, SE3 at 2 MWh, and NO5 at 1.5 MWh. The data exhibits seasonality where the stored energy is at its lowest at the beginning of the year and its highest from mid-year to the end of the year. This seasonality will be lost when only training on a single year of data.

6.2.14 Summary of feature trend investigation

The target feature price in zone NO1 shows a significant increase in price and volatility of price from 2021 to 2022, this might have a negative impact on the performance of models.
Several of the features have so many missing values and zero values that they would introduce more noise than signal when training models, and should therefore be dropped. The features that should be dropped are "Fossil Gas" for all zones except for NO5, and "Other" for zones NO1, NO2, NO3, and NO5 (this does not include "Other renewable").
A lot of the data seems to be missing prior to 2021, particularly electricity production stemming from "Waste". There are also a lot of data missing prior to 2022, particularly from zone SE3. This might also lead to a decreased performance from models trained on the three years of data.
Several of the features exhibit some seasonality which might be lost when only using a single year to train models. The combination of a significant increase in price and volatility of price, as well as many features that are missing a lot of values pre-2021 and pre-2022, means that training on the three years of data is likely to lead to poorer performance than just training on the data from the year 2022. We have made the decision to retain only the data from 2022.
The data contains outliers that function as noise when training the model. The outliers should therefore be handled to improve the performance of the model.

6.3 Dropping Instances From 2020 And 2021

As we discussed in Chapter 4 "Data Acquisition" the quality of the data is the most important factor in how biased ML model predictions are. Several features have inexplicable trends and sudden changes in values throughout the year 2020 (Figure 6.14). Based on Subsection 6.2.14 "Summary of feature trend investigation", it is concluded that inconsistency in our data is mainly caused by too many zero values in the year 2020, missing values also contribute to this data quality problem.

Looking at the plotted features in Figures 6.14 and 6.15, we find out that two of the biggest generation units in SE3 are "Nuclear" and "Hydro Water Reservoir", which do not have any value/data for 2020 and 2021. SE3 bidding zone data from 2020 and 2021 can be dropped and the dataset gets limited to 2022 only. This is also sensible because the Ukraine war started in 2022 and changed electricity market dynamics in Europe, in 2022 we see huge spikes in day-ahead prices more often compared to the years before. The reason for having 3 years of data was mainly based on our conducted research into electricity price forecasting ML models. We had an understanding that at least 1 year of testing data is required for successfully forecasting electricity prices. The reason for choosing 1 year is that one ML model can learn seasonally from a whole year and predict the seasonality and patterns for coming weeks and days. Knowing data for 2020 and 2021 has a low quality and will decrease model performance, the decision we made was to only focus on the year 2022.

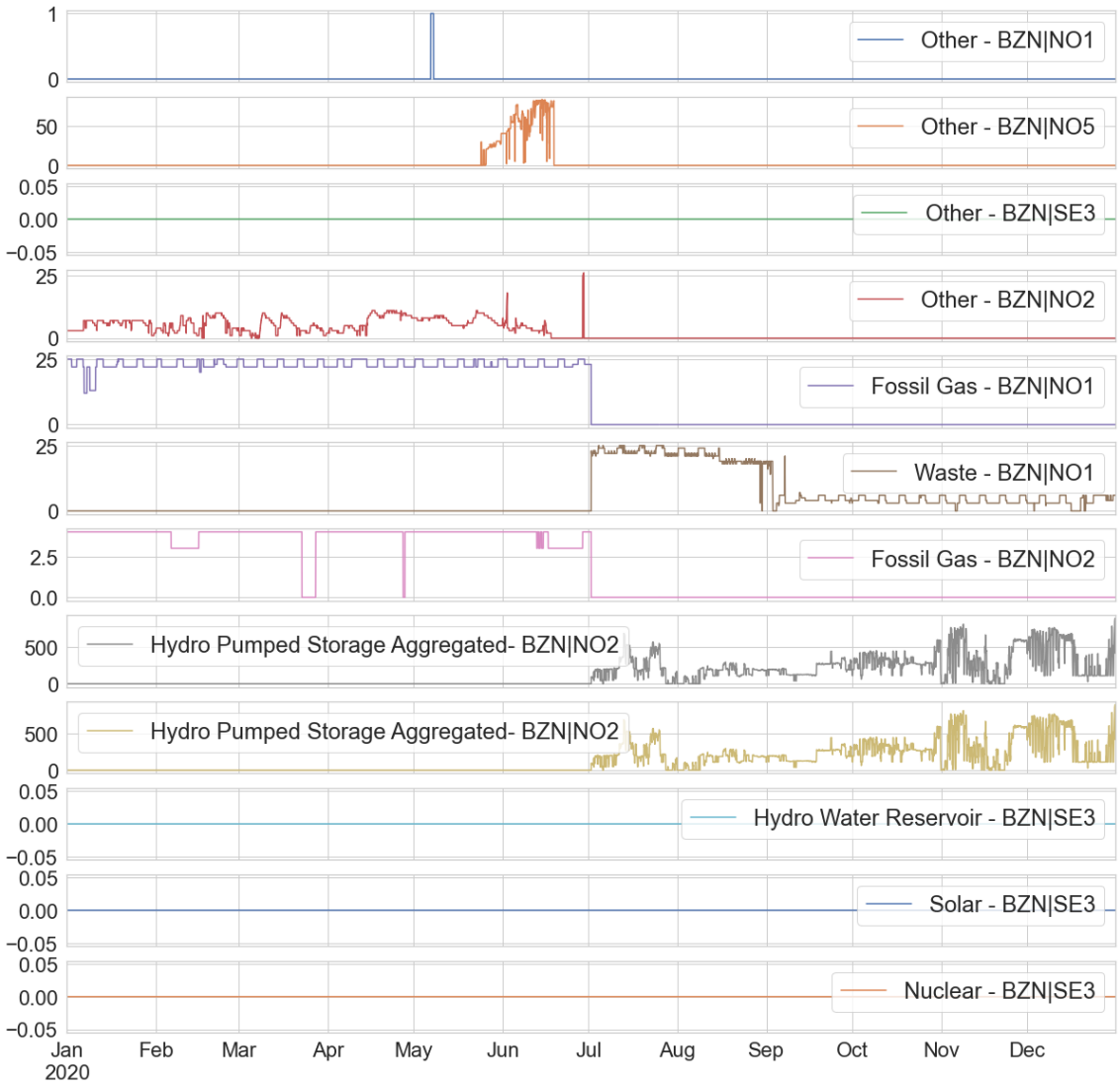

Figure 6.16: Low quality features 2020.

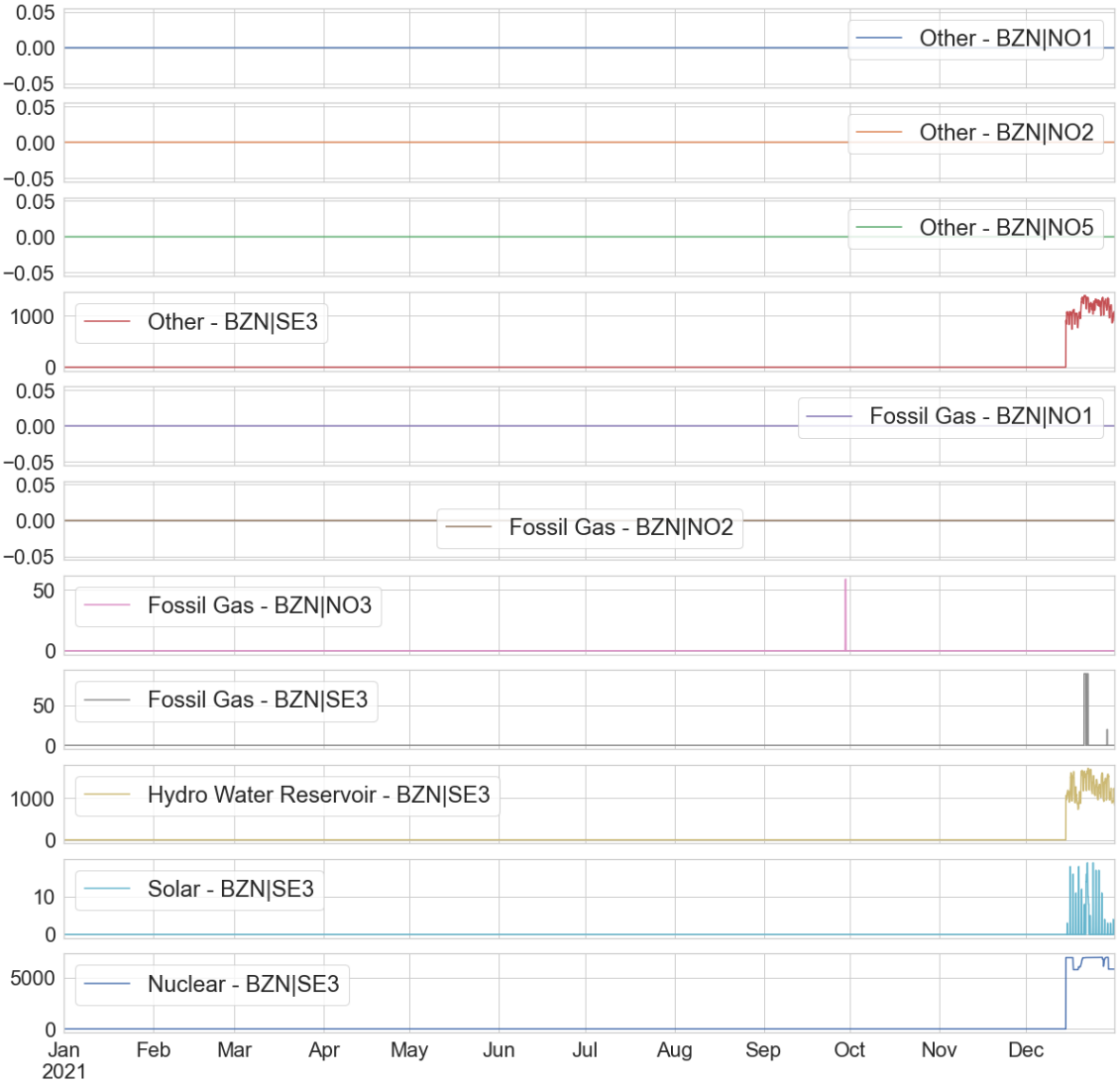

Figure 6.17: Low quality features 2021.

6.4 Handling Outliers

Under the feature trend investigation, we identified that the data had outliers. In this section, we will handle the existing outliers.

A common method to identify outliers is the standard deviation. However, we are dealing with time series data which has trends, and the standard deviation method would not be able to identify local outliers within the trends. We will therefore use a method that compares each value to its neighboring values to identify if the value is an outlier.

Another commonly used method for handling outliers is to drop the rows containing outliers, but since we are dealing with time series data, we cannot drop any rows. We will therefore calculate a new value that fits the trend line of the dataset, which will result in less noise in the dataset.

We have chosen to use the moving average method which identifies outliers based on the difference between the actual value and the average change in data series over time (the moving average). We remove the outliers by setting the value to NaN.

```
# handling outliers thorugh moving average method:

# creating a copy of the dataframe for moving average operations
df_ma = df_2022.copy()

# creating a list of numeric column names
numeric_column_names = df_ma.select_dtypes(include=np.number).columns.tolist()

# looping through all numeric columns
for column_name in numeric_column_names:

    # calculating the rolling mean of the price for NO1
    # using a window of 7 days * 24 hours = 168
    rolling_mean = df_ma[column_name].rolling(window=168).mean()

    # absolute difference between actual value and the rolling mean
    diff = abs(df_ma[column_name]-rolling_mean)

    # calculating the mean and standard deviation of the difference
    mean_diff = diff.mean()
    std_diff = diff.std()

    # setting threshold  for the outlier to 3 standard deviations
    # corresponding to a range including approximately
    # 99.7% of data in a gaussian distribution
    threshold = 3 * std_diff

    # identify the outliers
    outliers = diff[diff>threshold].index.tolist()

    # replace outlier values with nan values
    for i in outliers:
        df_ma.loc[i, column_name] = np.nan
```

Figure 6.18: Replacing outliers with Nan values in the dataset.

We can then impute the Nan values through linear interpolation which calculates a linear value based on the neighboring values.

```
# imputing missing values through linear interpolation
for col in missing_values_column_names:
    df_ma[col] = df_ma[col].interpolate(option='linear')
```

Figure 6.19: Replacing NaN values with linear interpolated values in the dataset.

To visualize the processing done on the dataset we can plot the day-ahead prices for all zones before and after handling outliers:

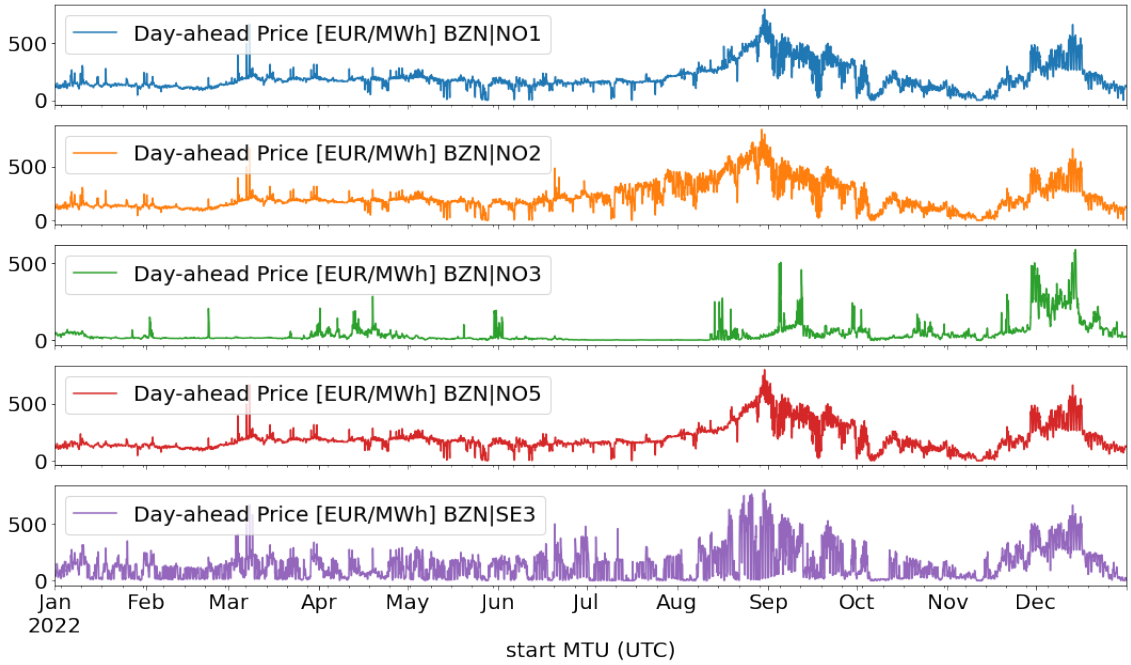

Figure 6.20: Day-ahead prices for all zones before handling outliers.

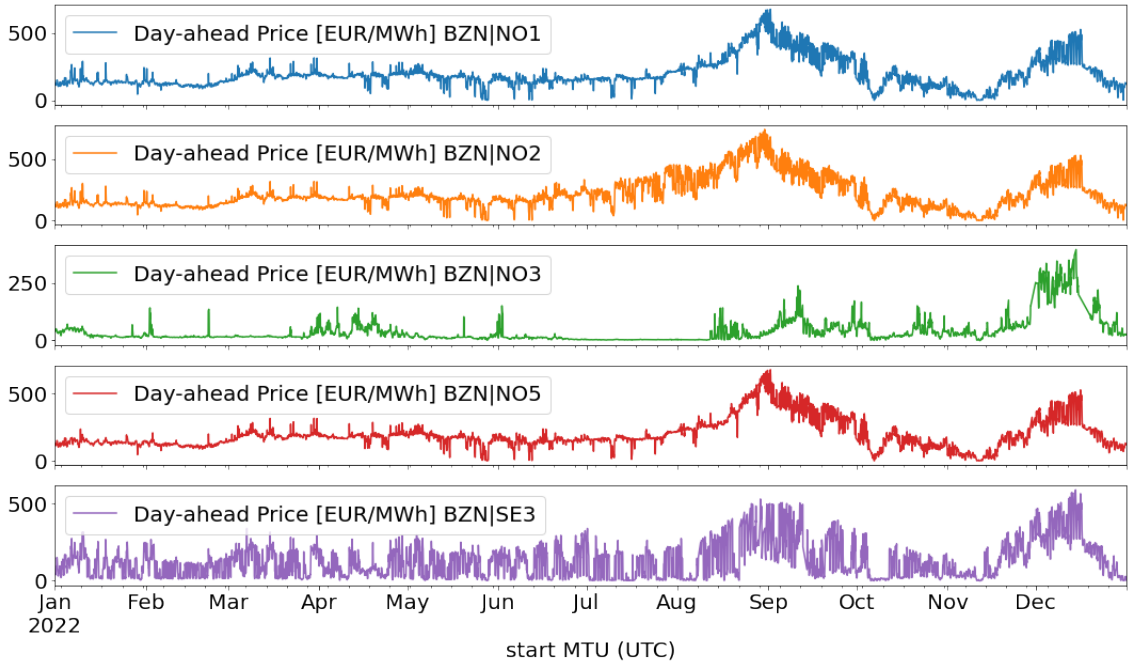

Figure 6.21: Day-ahead prices for all zones after handling outliers.

We can see that a lot of the outliers have been removed and replaced by a value closer to the trend line.

6.5 Feature Correlation Investigation

Correlation investigation is the process of analyzing the relationship between two or more variables and assessing the strength and direction of their association. Statistical methods are commonly employed to measure the extent to which changes in one variable are linked to changes in another variable. The main objective of correlation investigation is to identify patterns and associations that can aid in comprehending the underlying relationships between variables and aid in prediction or decision-making.

Feature correlation investigation, on the other hand, involves examining the relationship between various features or variables in a dataset to determine if they are related to each other, and if so, to what extent. It is a critical step in data analysis and machine learning as it aids in identifying which features are essential and which ones can be disregarded or removed to enhance the model's accuracy.

In feature correlation investigation, statistical methods like Pearson correlation coefficient, Spearman correlation coefficient, and Kendall's Tau are typically employed to quantify the degree of correlation between features. Scatter plots, heat maps, or correlation matrices are often used to visualize the results of this investigation to identify patterns and trends in the data. In this case, the Pearson correlation coefficient method is utilized to determine the correlation levels.

6.5.1 Table: correlation levels & ranges

Correlation Level & Ranges (X = Feature)	Number Of Features in Range	Percentage of total feature correlations (1540 correlations)
No correlation: X > 0 & X < 0.1	184	11.948%
Little Correlation: X > 0.1 & X < 0.3	448	29.090%
Medium Correlation: X > 0.3 & X < 0.5	378	24.545%
High Correlation:X > 0.5 & X < 0.7	298	19.350%
Very High Correlation:X > 0.5 & X < 1	232	15.064%

6.5.2 Interpreting table "correlation levels & ranges

By looking at the correlation one can observe that the 2 biggest levels of categories are "Little Correlation" and "Medium Correlation" with the former at 448 features and the latter at 378 features. These two count for the majority of features with a combined percentage of 53.635% of the total amount of features.

Other things to observe are that the upper categories "High Correlation" and "Very High Correlation" are very even with a difference of only 55 features falling into the categories. This is in contrast to the low ranger which consists of "No Correlation" and "Little Correlation" which has a difference of 264 features difference.

6.5.3 Correlation investigation by features

Table analysis: day-ahead price [EUR/MWh]

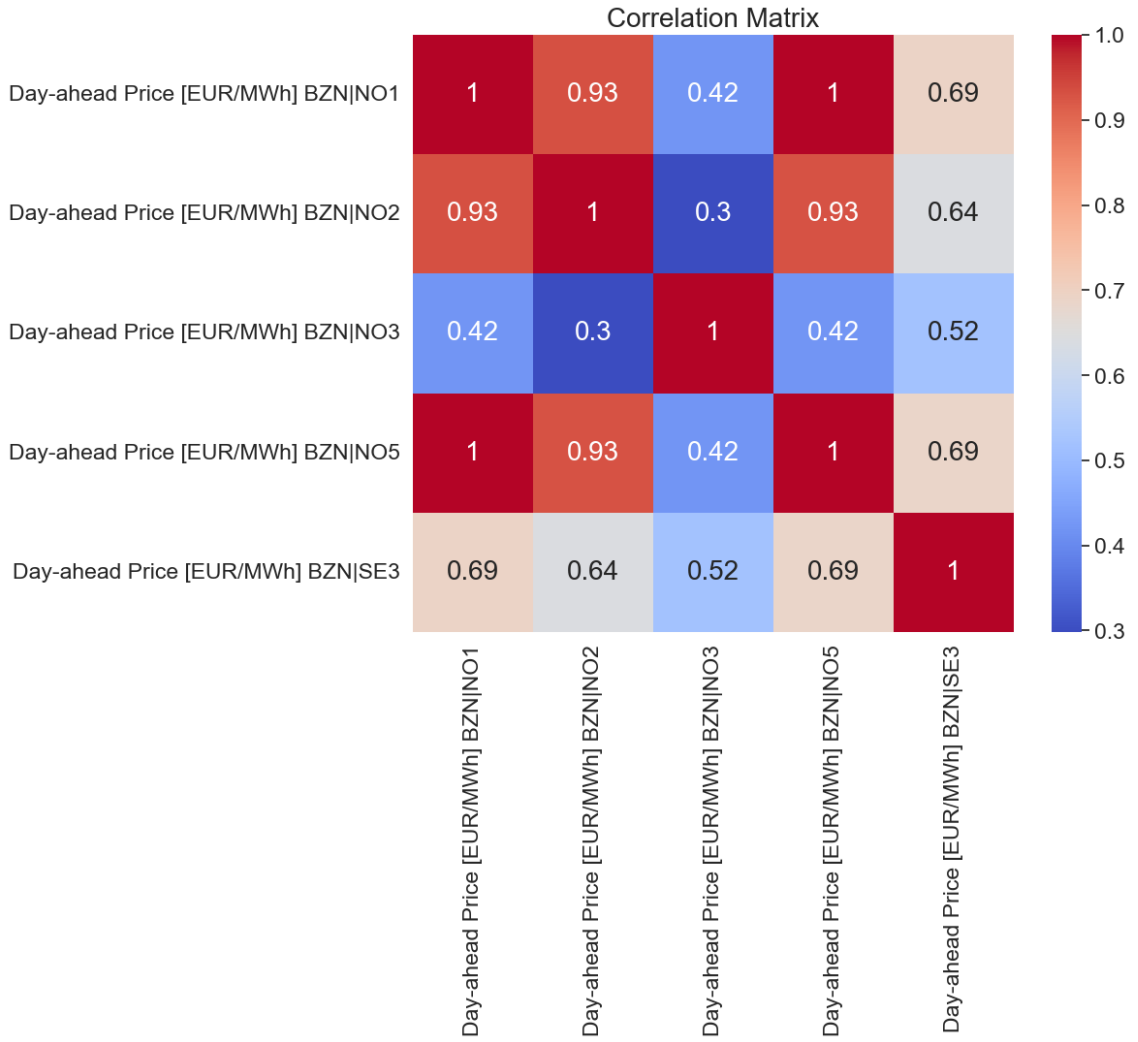

Figure 6.22: Day-ahead prices for all zones after handling outliers.

It can be observed that certain zones like NO3 have a medium correlation level with the prices of the other zones. This pattern can also be observed with SE3 which has medium correlation levels in the range of [0.52 - 0.69]. Another thing that can be observed is that NO1 and NO5's correlation levels are basically equivalent.

Table analysis: actual total load [MW] - BZN

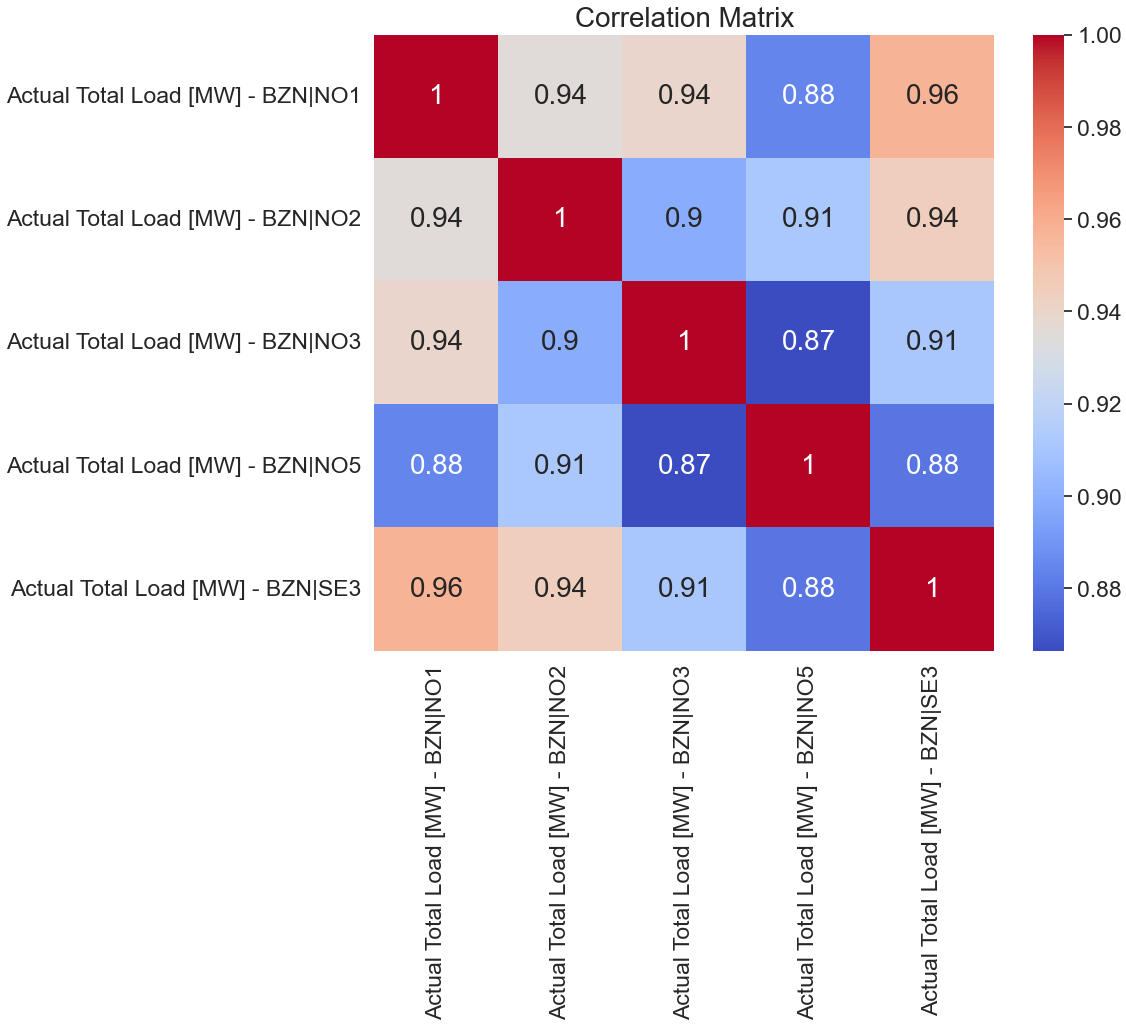

Figure 6.23: Actual total load [MW] - BZN.

Figure 6.23 demonstrates a strong correlation between the total load of the power network across all zones, regardless of the correlation pairs considered. The range of correlation coefficients between the total load in BZN and the total loads in other zones ranges between 0.87 and 0.96. This finding suggests that changes in total load in one zone are closely reflected in the total load across the entire network. Therefore, it can be hypothesized that the total load of the power network changes synchronously across all zones and in the same relative quantity to their respective zones.

Table analysis: hydro run-of-river and poundage - BZN

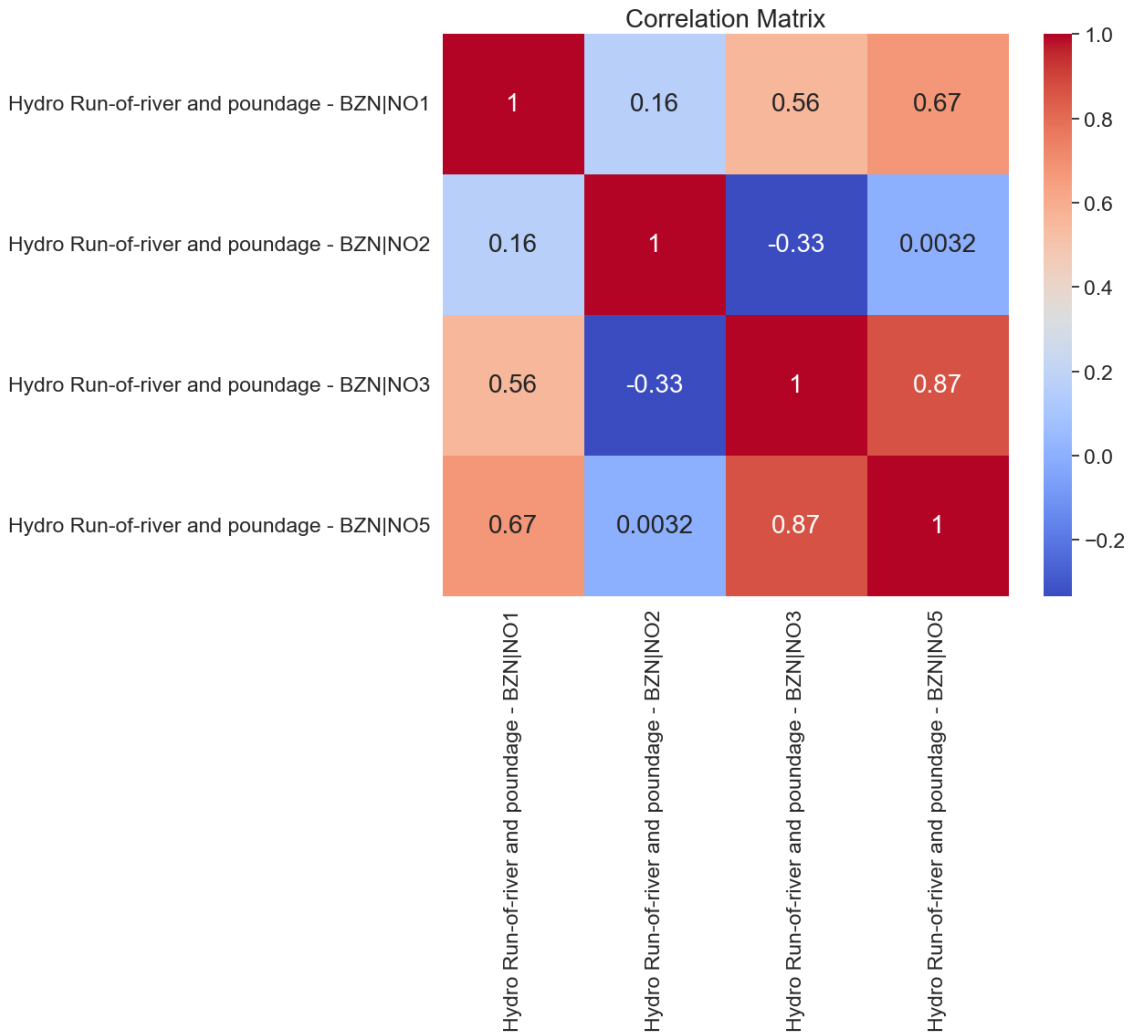

Figure 6.24: Day-ahead prices for all zones after handling outliers.

Figure 6.24 depicts the correlation levels with a wide range of values from 0.0032 to 0.87, irrespective of their positive or negative nature. The pair comprising of zones NO2 and NO5 exhibits the lowest correlation level.
The low correlation level can be attributed to various factors, such as the absence of running rivers and waterfalls in these zones. Maybe even the lack of infrastructure to harness the potential energy. This suggests that the energy generation in these areas may not be significantly affected by seasonal variations and weather conditions. Further investigation is necessary to identify and examine the potential factors contributing to the observed correlation levels among the zones.

Table analysis: hydro Water reservoir - BZN

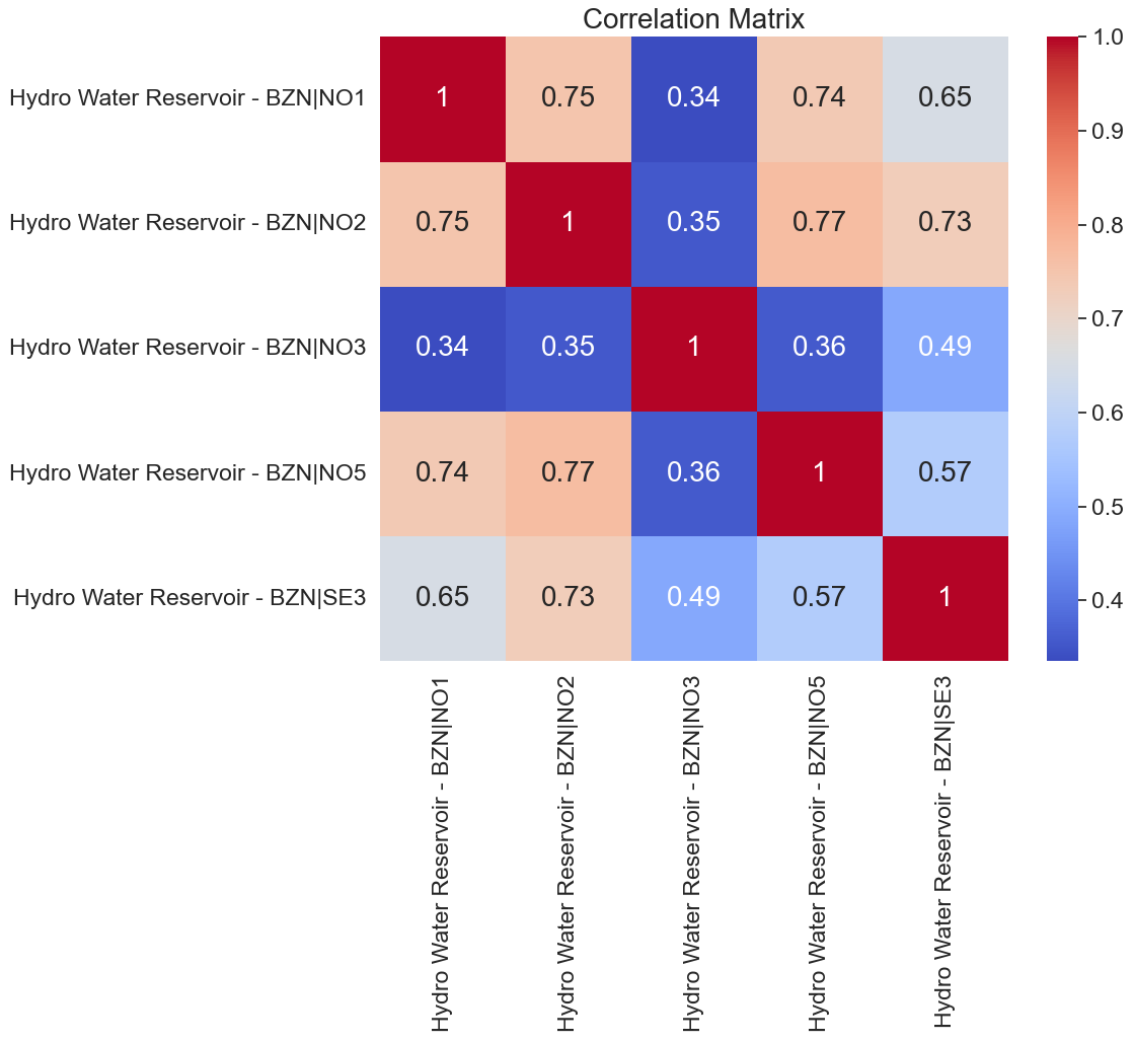

Figure 6.25: Day-ahead prices for all zones after handling outliers.

Figure 6.25 depicts a range of values that goes from 0.34 to 0.77. This can be interpreted as there being a weak to strong correlation between some of the zones with zone NO3 being the weakest link.

Zone NO3 exhibits a low correlation level with all the other zones, with the highest correlation level observed with SE3 at 0.49 and the remaining correlation values ranging between 0.34 to 0.36. These findings suggest that the correlation among the water reservoirs is not significant and that there are likely other factors influencing the water levels in these reservoirs.

Table analysis: waste - BZN

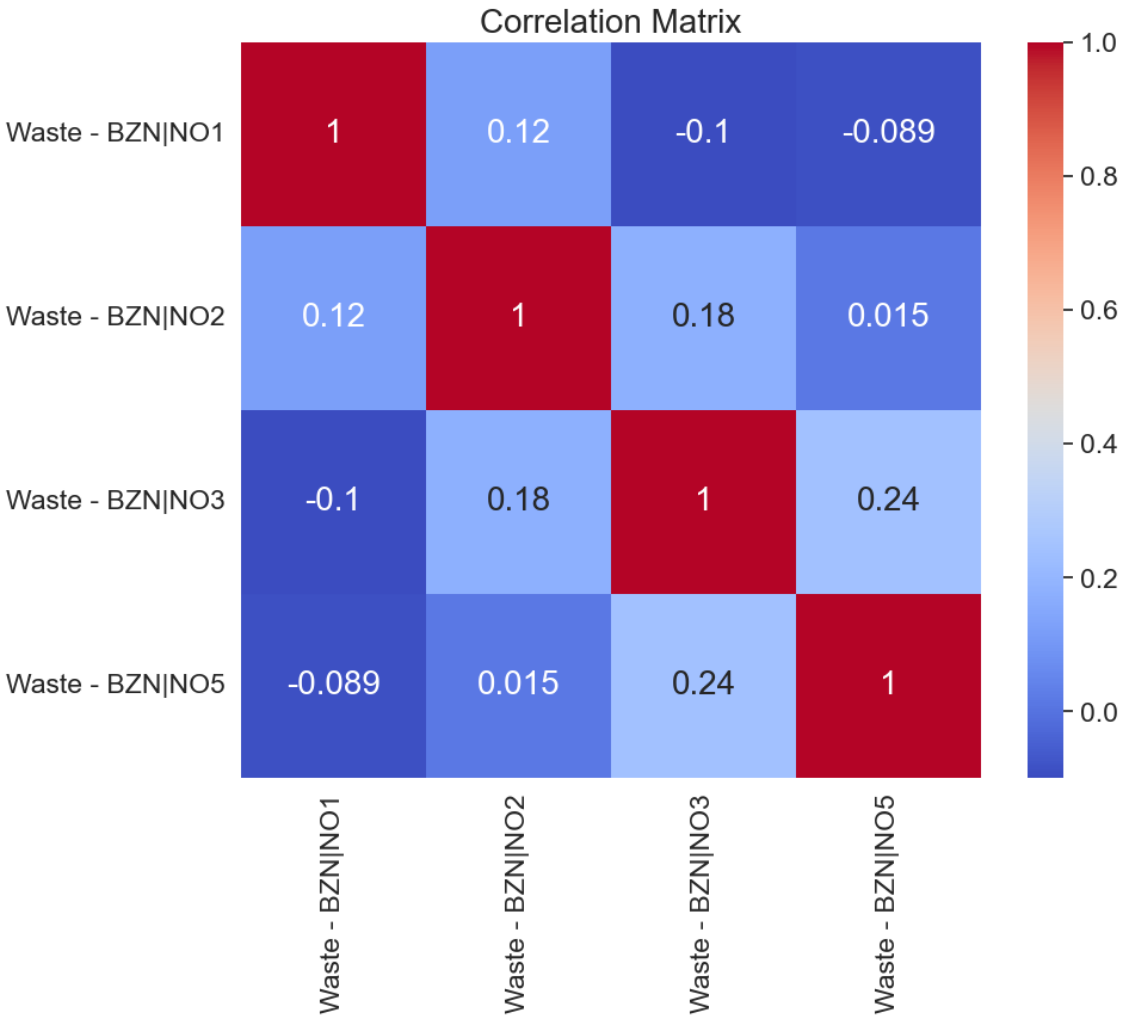

Figure 6.26: Day-ahead prices for all zones after handling outliers.

The results obtained from Figure 6.26 reveal a range of absolute values ranging from 0.015 to 0.24. This finding suggests that there is negligible to no correlation among the different zones when it comes to the generation of energy from waste. Therefore, it can be concluded that the energy generation from waste in a particular zone does not necessarily impact or correlate with the energy generation in another zone. Further research is required to identify potential factors influencing energy generation from waste in different zones.

Table analysis: CBF BZN

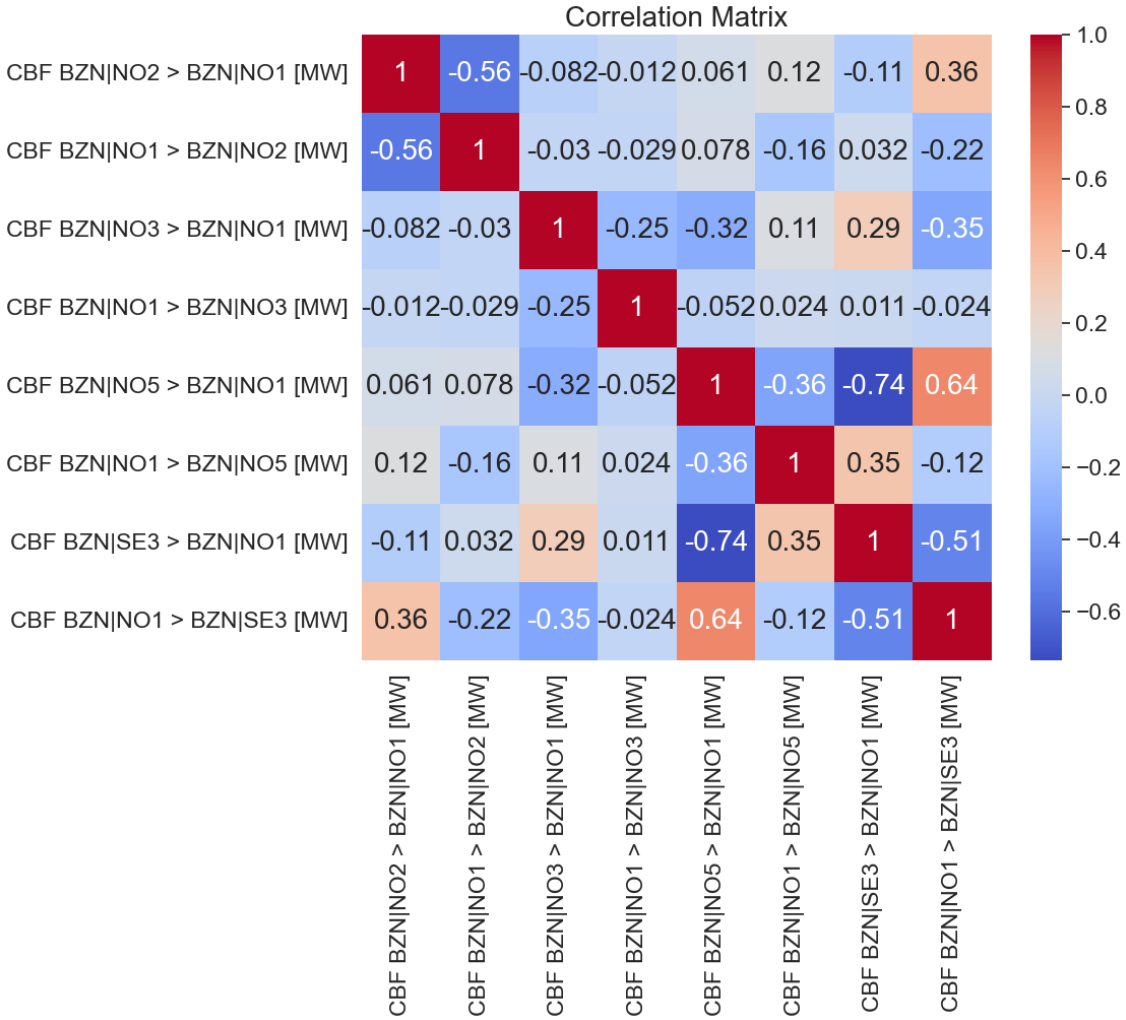

Figure 6.27: Day-ahead prices for all zones after handling outliers.

The analysis of Figure 6.27 reveals a wide range of correlations, ranging from 0.011 to 0.74. Negative correlations are observed between SE3 and NO1, as well as between NO5 and NO1. These correlations suggest that high energy transfer to either one of these zones from NO1 results in a reduction of energy transfer to the other zone. Moreover, the highest positive correlation is at 0.29, indicating little to no positive relationships among the zones. These findings suggest a strong inverse relationship among the transfers of different zones.

Table analysis: stored energy value water reservoirs and hydro storage plants [MWh

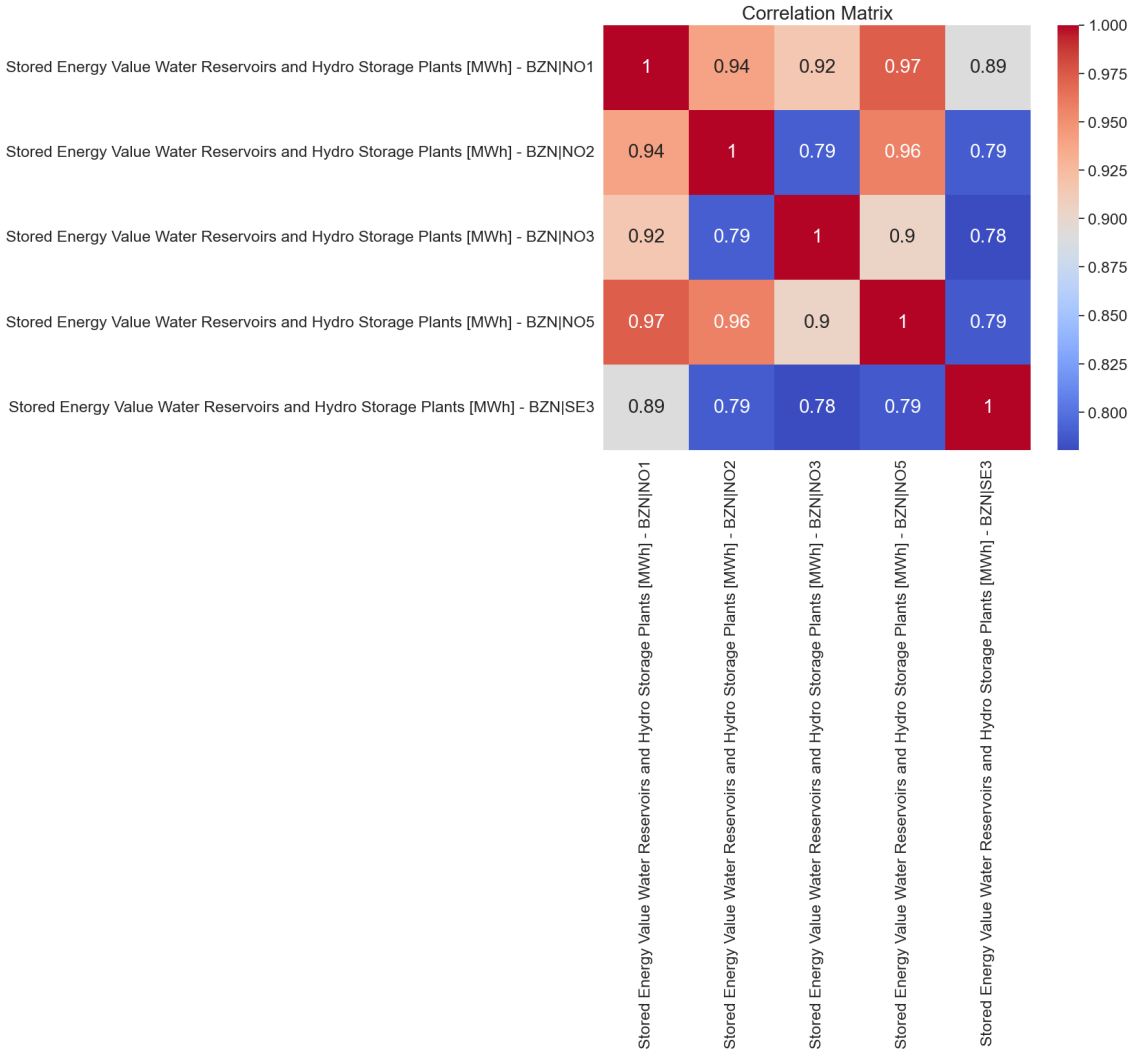

Figure 6.28: Day-ahead prices for all zones after handling outliers.

Upon analyzing Figure 6.28, it is apparent that there is a consistently high correlation between the zones, with the lowest correlation at 0.78 and the highest at 0.97. This suggests that there is a synchronous relationship between the stored energy levels of the water reservoirs and hydro storage plants across all the zones. Hence, changes in energy levels in one zone are likely to be reflected in the energy levels in other zones. It is possible that this synchronous relationship is due to similarities in weather patterns across the zones. This relationship can be observed in the figure in chapter 6.2.13 "Stored Energy Value Water Reservoirs and Hydro Storage Plants".

6.6 Feature Selection

Feature selection is an important step in the machine learning process that involves selecting the most relevant features from the dataset to use to build a model [41]. This process can help

reduce model complexity, improve model accuracy, and increase interpretability. In this subchapter, we will explore different techniques for feature selection, including removing features with low variance and those with high correlation. We will also discuss the benefits and challenges of feature selection.

6.6.1 Features with low variance

After dropping the data for 2022 and 2021, we discovered multiple features in the 2022 data that have very low variance (Figure 6.29). This means that there are certain features we use to train our ML model with the training data that add complexity and dimensionality without increasing accuracy. Consequently, estimating the target value "Day-ahead Price [EUR/MWh] BZN|NO1" may result in very small changes in either a positive or negative direction. Low variance is caused by having many zero values and few distinct values, which affect the value distribution and variation. By removing these features, we simplify the modeling process, which may also improve performance by focusing on the more important features.

By testing the output of a function we created that drops features based on a threshold for a variance on a simple linear regression model, we found that a threshold of 8 gave the best performance for that model. Therefore, we dropped the following features due to low variance:

- “Fossil Gas” for zones: NO1, NO2, NO3 and SE3.
- “Other” for zones: NO1, NO2, NO3, and NO5 (this does not include “Other renewable”).
- "Waste" for zones: NO1 and NO3

This confirms what we found from the trend investigation.

Figure 6.29: Low variance features 2022.

6.6.2 Features with high correlation

Multicollinearity is a phenomenon where two or more independent variables in a regression model are highly correlated with each other [18]. This can cause issues in regression models, such as unstable and unreliable coefficients, decreased statistical power, and difficulty in interpreting the results.

By removing one of the highly correlated features, we can reduce the multicollinearity in the model and improve its performance. This can lead to more accurate and stable coefficient estimates, better model interpretability, and improved predictive performance. Additionally, removing redundant features can reduce the complexity and overfitting of the model, making it more efficient and generalizable.

We created a function that selects features based on a threshold for the correlation pairs and chooses the one with the lowest correlation to the target feature to drop from the data frame. The function's output is tested on a simple linear regression model to determine the optimal threshold.

However, the results between a threshold of 0.70 and 0.9, which is commonly regarded as indicating a very high correlation, showed no distinguishable difference. Therefore, we decided to choose 0.9 to keep most features, in case a properly tuned model might be able to find complex relationships between the features and the target. As a result, we dropped the features for each pair, as seen in Table 6.1.

Dropped	Kept
Day-ahead Price [EUR/MWh] BZN|NO2	Day-ahead Price [EUR/MWh] BZN|NO5
Actual Total Load [MW] - BZN|NO1	Actual Total Load [MW] - BZN|NO2
Actual Total Load [MW] - BZN|NO3	Actual Total Load [MW] - BZN|NO1
Actual Total Load [MW] - BZN|NO3	Actual Total Load [MW] - BZN|NO2
Actual Total Load [MW] - BZN|NO1	Actual Total Load [MW] - BZN|NO5
Actual Total Load [MW] - BZN|NO2	Actual Total Load [MW] - BZN|NO5
Actual Total Load [MW] - BZN|SE3	Actual Total Load [MW] - BZN|NO1
Actual Total Load [MW] - BZN|SE3	Actual Total Load [MW] - BZN|NO2
Actual Total Load [MW] - BZN|SE3	Actual Total Load [MW] - BZN|NO3
Other - BZN|SE3	Actual Total Load [MW] - BZN|NO1
Hydro Water Reservoir - BZN|NO5	CBF BZN|NO5 > BZN|NO1 [MW]
Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO2	Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO1
Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO1	Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO3
Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO1	Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO5
Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO2	Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO5
Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO5	Stored Energy Value Water Reservoirs and Hydro Storage Plants [MWh] - BZN|NO3

Table 6.1: As seen in the correlation analysis "Day-ahead Price [EUR/MWh] BZN|NO2", "Actual Total Load [MW]" and "Stored Energy Value Water Reservoir and Hydro Storage Plants [MWh]" has a high correlation between them. Furthermore, we see that "Other - BZN|SE3" and "Actual Total Load [MW] - BZN|NO1" has a high correlation as well as "Hydro Water Reservoir - BZN|NO5" and "CBF BZN|NO5 > BZN|NO1 MW". The features under-dropped are removed because of their lower correlation to the target feature.

6.7 Summary EDA

In summary, when conducting exploratory data analysis it is important to first collect and prepare the data by checking for any missing values, outliers, or errors. Once the data is cleaned and

formatted properly, various statistical and graphical techniques were used to identify patterns, relationships, and trends in the data. Additionally, it is important to keep in mind the context and purpose of the analysis, as this will help guide the choice of techniques and interpretation of the results. The goal of exploratory data analysis is to gain insights and identify potential areas for further investigation.

In Section 6.2 "Feature Trend Investigation" it is noted that the energy market data in zone NO1 exhibits a significant increase in price and volatility from 2021 to 2022. Additionally, there are several features with many missing or zero values that should be dropped. The seasonality of several features may also be lost when training models on a single year of data. Therefore, it is suggested that training on the three years of data may lead to poorer model performance. The data also contains outliers that can be handled through the moving average method. The moving average method identifies outliers based on the difference between the actual value and the average change in the data series over time. Outliers are removed by setting the value to NaN. Linear interpolation is then used to impute the NaN values based on the neighboring values.

While investigating trends as a part of EDA it was discovered that the data from 2020 and 2021 had low-quality features with inexplicable trends and sudden changes in values. Additionally, the lack of data for the two biggest generation units in SE3 further supported the decision to drop the data from 2020 and 2021 and limit the dataset to 2022 only. This was a sensible approach given the changing market dynamics caused by the Ukraine war, which resulted in huge spikes in day-ahead prices more often compared to the years before. Although having 3 years of data was initially considered important for electricity price forecasting ML models, dropping the low-quality data from 2020 and 2021 may ultimately improve model performance.

As part of our exploratory data analysis process, we conducted a correlation investigation to analyze the relationships between variables and assess the strength and direction of their association. Specifically, we conducted a feature correlation investigation to examine the relationship between various features or variables in the dataset to determine if they are related to each other and to what extent. We utilized the Pearson correlation coefficient which is a statistical method to quantify the degree of correlation between features. And used visualization tools such as scatter plots, heatmaps, or correlation matrices to identify patterns and trends in the data. While interpreting the results, we noted that correlation does not necessarily imply causation, and it is important to exercise caution and investigate further before making any causal claims based on correlation data. Overall, correlation investigation is a valuable tool for identifying patterns and relationships between variables, aiding in prediction and decision-making for creating machine learning models.

By conducting feature selection as described in Section 6.6, the last part of exploratory data analysis, we can gain insights into which features are most informative and should be included in the model. This process can also help us to identify any issues with the data, such as missing or erroneous values. The process of feature selection helps to eliminate irrelevant or redundant features, reducing the dimensionality of the data and improving model performance. Feature selection is a powerful tool that can improve model performance, reduce overfitting, and speed up training time.

The trend investigation, dropping low quality and low variance instances, handling of outliers, features correlation investigation and feature selection aim to improve the performance of models

trained on our energy market dataset by reducing noise and improving data quality. Ultimately conducted EDA will improve model performance.

Chapter 7

Implementation

The successful implementation and training of ML models rely not only on the choice of algorithms themselves but also on carefully considering and applying appropriate preprocessing techniques and hyperparameter tuning on a per-algorithm basis. These crucial steps are vital in enhancing model performance and ensuring accurate and reliable predictions. In this Chapter, we explore the tools, libraries, and methodologies employed to train the selected ML models. Each algorithm requires specific preprocessing steps and hyperparameter tuning techniques to unlock its full potential. Therefore, we aim to provide a comprehensive overview of the functionalities of these tools and libraries while highlighting their significance in achieving optimal model performance.

Throughout this Chapter, we will outline the specific preprocessing steps undertaken for each algorithm, tailored to its unique requirements. Additionally, we will discuss the hyperparameter tuning process, which involves fine-tuning the models' key parameters to maximize their effectiveness. By reviewing these processes on a per-algorithm basis, we aim to provide readers with a clear understanding of the considerations and decisions involved in successfully implementing these ML algorithms. Figure 7.1 illustrates the ML model training process that has to be performed when implementing each of the algorithms. The process is repeated until the improvement of the model performance halts or is prohibitively resource intensive.

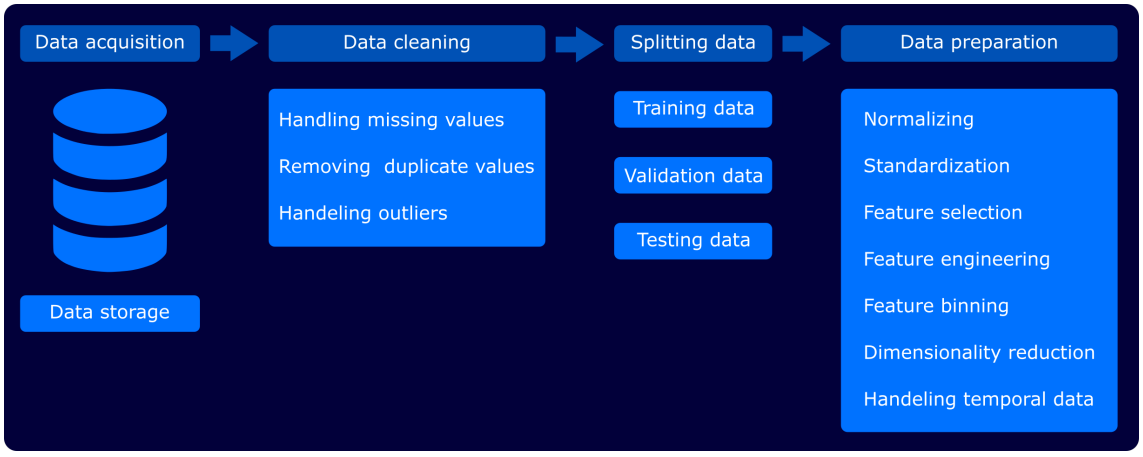

Figure 7.1: ML model training process.

7.1 Tools

- **Python** is a widely-used programming language with significant popularity in the field of ML due to its extensive libraries and frameworks designed specifically for building and training ML models [55]. Python provides a user-friendly syntax and a vast ecosystem of libraries such as NumPy, pandas, and scikit-learn, which offer powerful tools for data manipulation, numerical computing, and model development. These libraries enable researchers and developers to efficiently preprocess and analyze data, build predictive models, and evaluate their performance.

  In our research project, we made effective use of Python to handle various stages of the ML pipeline, including data acquisition, data preprocessing, and training our ML models.

- **Jupyter Notebook** is an interactive web-based tool widely used in data analysis, scientific computing, and ML [66]. It provides an environment where users can create and share documents containing live code, equations, visualizations, and explanatory text. Jupyter Notebook supports multiple programming languages, including Python, R, and Julia.

  Jupyter Notebook allows researchers and data scientists to combine executable code, computational output, and explanatory content in a single document. It provides a versatile platform for developing ML models, as it allows users to write, run, and modify code snippets interactively, facilitating the exploration and experimentation process.

  One of the significant advantages of Jupyter Notebook is its ability to display code output and visualizations in line with the code cells. This feature enhances the readability and interpretability of the code, making it easier for researchers to present their findings and share their work with others. Additionally, Jupyter Notebook supports the integration of rich media, including images, videos, and interactive visualizations, which further enhances the presentation capabilities of ML projects. Jupyter Notebook is particularly useful for ML tasks as it enables users to combine code with explanatory text, allowing them to document the step-by-step process of model development, data preprocessing, and evaluation. This documentation aspect makes it beneficial for academic projects, as it provides transparency and reproducibility of the research.

  In our research project, we utilized Jupyter Notebook as a development environment for our machine learning model coding in Python. Jupyter Notebook's interactive interface allowed us to test and iterate our code in real-time, providing us with instant feedback on our changes. We leveraged Jupyter Notebook's rich text formatting capabilities to document our code, ensuring readability and reproducibility.

- **GitHub** is an online platform and version control system that facilitates collaboration and sharing of code among individuals or teams working on software development projects [30]. It allows multiple developers to work collectively on the same codebase, enabling efficient collaboration and coordination.

  GitHub serves as a centralized repository where developers can store and manage their code. It provides features that enable seamless collaboration, such as version control, issue tracking, and code review. These features are especially valuable when working on complex projects or when multiple people are involved in the development process.

  In our research project, we leveraged the power of GitHub to streamline and enhance our code collaboration process for data acquisition, data prepossessing, and ML model training.

By utilizing GitHub as our central repository, we were able to effectively collaborate on code development and ensure seamless integration of our contributions.

7.2 Libraries

- **Numpy** is a widely-used Python library that provides powerful mathematical and numerical operations for efficient data manipulation and analysis [50]. It can handle multi-dimensional arrays and matrices, making it well-suited for scientific computing and data analysis tasks.

  In our project, we have leveraged the capabilities of Numpy to perform various mathematical operations on our dataset.

- **Pandas** is a popular open-source Python library extensively utilized for the effective handling and manipulation of structured data, providing efficient data analysis tools [40]. Pandas introduces two primary data structures: Series and DataFrame. The Series is a labeled, one-dimensional array capable of accommodating diverse data types, while the DataFrame is a tabular, two-dimensional data structure with labeled rows and columns. These data structures empower researchers and analysts to proficiently manage, manipulate, and analyze data. The library encompasses a wide range of functions and methods, serving purposes such as data cleaning, exploration, wrangling, and transformation. Pandas facilitates researchers in managing missing data, implementing filters, executing aggregation operations, merging datasets, and efficiently conducting statistical computations. Moreover, Pandas is recognized for its seamless integration with other Python libraries, including NumPy, Matplotlib, and Scikit-learn, establishing a robust ecosystem for data analysis and ML endeavors. It offers effortless interoperability with these libraries, enabling researchers to preprocess and prepare data for ML models while facilitating the visualization of outcomes.

  In our research project, the dataset is imported and organized into structured NUmDataFrames, serving as the foundation for subsequent data prepossessing and ML model training.

- **Matplotib** is a widely-used open-source library in Python that provides comprehensive and customizable plotting capabilities for creating visualizations and graphs [67]. It is a fundamental tool for data visualization in various fields, including scientific research, data analysis, and ML. Matplotlib allows researchers and data scientists to create a wide range of visualizations, including histograms, scatter plots, line plots, bar charts, and more. It provides fine-grained control over every aspect of the visualization, such as axes, labels, titles, colors, and styles. This flexibility enables researchers to create publication-quality visualizations that effectively convey insights and patterns in the data.

  In our research project, Matplotlib is used to plot the prediction vs actual electricity prices so that the performance can be evaluated.

- **Scikit-Learn** (or sklearn) is a widely utilized open-source Python library for machine learning, renowned for its extensive collection of tools and algorithms designed to tackle diverse machine learning tasks [47]. Leveraging the capabilities of prominent Python libraries like NumPy, SciPy, and Matplotlib, Scikit-learn emerges as a comprehensive and potent resource catering to the requirements of both researchers and practitioners in the machine learning domain.

  With Scikit-learn, researchers gain access to a broad spectrum of machine learning algorithms encompassing classification, regression, clustering, dimensionality reduction, and model

selection. These algorithms are implemented through a uniform and intuitive interface, empowering researchers to seamlessly experiment with various techniques and models.

Facilitating a cohesive framework, the library equips researchers with capabilities for data preprocessing and transformation, dataset partitioning for training and testing, model evaluation, and hyperparameter tuning. In addition, advanced functionalities such as feature extraction, cross-validation, and model persistence augment the overall ML workflow, further enriching the research process.

In our research project, Scikit-Learn functions and ML algorithm implementations are used to implement ML algorithms, preprocess data, partitioning of data, and hyperparameter tuning.

7.3 Random Forest Regressor

The scikit-learn library offers an easy-to-use implementation of the Random Forest Regressor with customizable parameters for fine-tuning the model's performance and generalization capabilities. We utilized the sklearn implementation of the RFR for training our RFR ML model [54]. The official documentation of scikit-learn provides comprehensive details on the RFR's usage and parameters, serving as a valuable resource for understanding and utilizing this powerful algorithm in our project.

7.3.1 Initial data preprocessing

The initial stage of implementing the ML algorithm is to perform initial data preprocessing to prepare the data for training a baseline model. A baseline model is an unoptimized model trained on the raw data without significant preprocessing or hyperparameter tuning. Its purpose is to provide a benchmark performance level that can be used to evaluate the effectiveness of subsequent model improvements. By minimizing the preprocessing performed during this stage, we can establish a more accurate baseline performance level for the model, which can help identify fundamental issues or limitations with the data that need to be addressed before proceeding with further processing. The initial data preprocessing for the baseline RFR model was limited to splitting the data into training, validation and testing data, and to create lagged features.

Train validation test split

Time series data often have temporal dependencies between observations. In other words, the current values of a time series are often related to past values, and this relationship is what we want to capture with our model. Splitting the data in chronological order ensures that the model is trained on past data and scored on future data, which is more representative of how the model behaves in the real world. This study used the train_test_split function from the sklearn library to split the data into three subsets; Training, Validation, and Testing. Specifically, the first 80% of the data were assigned to the training subset, and then 10% and 10% were assigned to the validation and test subsets, respectively.

Creating lagged features

Lagged features play a valuable role in time series forecasting as they offer insights into the past values of the target variable or other features. By including these historical values as input features,

the forecasting model becomes capable of capturing patterns and dependencies that evolve over time. This integration of past data enables the model to make more precise predictions by taking into account the inherent temporal characteristics of the data. Ultimately, incorporating lagged features empowers the model to better understand and leverage the dynamics and trends present in the time series data, leading to improved forecasting accuracy.

In this study, we perform multi-step forecasting, predicting the price at each of the 24 following periods. We can perform multi-step forecasting through direct forecasting or recursive forecasting. In direct forecasting, we train the model to predict all 24 periods into the future for each time step. In recursive forecasting, we only train the model with a one-step-ahead forecast and use the trained model repeatedly to achieve the desired numbers of predicted periods. The main drawback of recursive forecasting, instead of direct multi-step forecasting, is that it can accumulate errors over time and lead to increasingly inaccurate predictions as the forecast horizon becomes longer. Therefore, we use direct multi-step forecasting with lagged features for the RFR model in this study. To create a lagged dataset, we shift the target variable forward by one or more time steps, using the resulting series as the target variable for that time step. We also include the values of the predictors at the previous time steps as features. This way, the model can learn to use past predictor values to predict future target variable values.

```python
# Function to create lagged dataset with
# 24 lag features for each input feature
def create_lagged_dataset(df):
    # creating a copy of the dataframe
    lagged_df = df.copy()

    # Adding lagged features for target variable
    lagged_df['Day-ahead Price [EUR/MWh] BZN|NO1'] = \
        lagged_df['Day-ahead Price [EUR/MWh] BZN|NO1'].shift(-1)

    # Dropping the last row containing NaN values
    lagged_df.dropna(inplace=True)

    # Creating a dataframe with lagged features with
    # 24 steps for each of the original features
    lagged_df = pd.concat([lagged_df.shift(i) for i in range(24)], axis=1)

    # Removing the NaN rows that have been created
    # in the beginning of the dataset
    lagged_df.dropna(inplace=True)

    return lagged_df
```

Figure 7.2: Function used to create a 24-hour lagged dataset.

Figure 7.2 displays the function "create_lagged_dataset" which was written to create lagged features based on a horizon of 24 hours. The function returns a lagged dataset where each row contains the predictors and the target feature for a specific hour. The target variable for each row is the value 24 hours in the future, and the predictors for each row are the values of the predictors for the previous 24 hours.

7.3.2 Hyperparameter tuning

The RFR from sklearn has several important hyperparameters that significantly influence its performance [54]. One crucial hyperparameter is the number of estimators "n_estimators", which determines the number of decision trees in the ensemble. Increasing the number of estimators can improve the model's accuracy but may also increase computational complexity. Another vital hyperparameter is "max_depth", which defines the maximum decision tree depth in the RFR. Adjusting this parameter can control the model's complexity and ability to capture intricate patterns in the data. Additionally, "min_samples_split" and "min_samples_leaf" dictate the minimum number of samples required to split an internal node and to form a leaf node, respectively. These hyperparameters regulate the tree's depth and prevent overfitting by enforcing a minimum number of samples in each split. Lastly, "max_features" controls the number of features used when searching for the best split. It influences the diversity and randomness within the ensemble. Properly tuning these hyperparameters is crucial to optimize the RFR's performance and ensure accurate predictions for a given task and dataset.
The hyperparameter tuning process involved defining a dictionary of hyperparameters to search. Figure 7.3 displays the dictionary of hyperparameters used for tuning the RFR model.

```
# Define a dictionary of hyperparameters to search
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
}
```

Figure 7.3: Dictionary for hyperparameter tuning of RFR.

In this thesis, hyperparameter tuning for the RFR model was performed using the "RandomizedSearchCV" function from scikit-learn. This function selects n number of hyperparameter combinations from a provided hyperparameter dictionary. We performed the random search with 50 iterations.

To handle the time series nature of the data, rolling origin cross-validation was used, which involves training the model on a rolling window of historical data and testing it on the following observation. The primary benefit of using the rolling origin cross-validation method is that it maintains the temporal order of the data, which is essential for accurate time series forecasting. This approach ensures that the model is evaluated on data more similar to the future data it will predict.

7.4 XGBoost Regressor

This section presents detailed explanations for building multivariate forecasting models based on the direct one-step approach. The goal of our models is to predict future values of the target variable, "Day-ahead Price EUR/MWh BZN|NO1" based on historical data using both 1-hour and 24-hour lagged features. The models are described as multivariate because a set of time series input features is used for forecasting. The direct one-step forecasting approach involves directly estimating the

value at the next time step without considering the previous predictions as inputs.

We employed the XGBoost algorithm, using the "XGBRegressor" class from scikit-learn library to train our one-step direct forecasting XGBoost model. XGBoost is an ensemble of decision trees that combines multiple weak predictive models to create a strong predictive model. It iteratively optimizes a loss function to learn the relationships between the input features and the target variable. In section 4.8 of this thesis, we explained in great detail how XGBoost works.
The models use lagged features from the historical data to make predictions for the next time step. By training the models on a subset of the data and validating it on another subset, we can assess their performance in terms of accuracy and generalization ability.

Equation for direct one-step forecasting

The specific form of the direct one-step forecasting equation will vary depending on the chosen forecasting technique, such as linear regression, neural networks, etc. The equation provided below serves as a general representation of the direct one-step forecasting approach, where the model directly predicts the value at the next time step based on the available input features. A general equation [33] can be represented as:

$$\hat{y}_{t+1} = f(X_t) \tag{7.1}$$

Where:
- $\hat{y}_{t+1}$ is the predicted value at time step $t+1$.
- $f(\cdot)$ represents the forecasting model that maps the input features X_t at time step t to the predicted value $\hat{y}_{t+1}$.

7.4.1 Initial data prepossessing

Before training and building time series forecasting models, it is crucial to perform initial data preprocessing tasks such as feature engineering, creating lagged dataset, and indexing our time series data. These steps play a vital role in enhancing the model's performance and the quality of predictions.

Indexing and changes in feature names

The indexing dataset enables the model to understand the temporal nature of the data and identify trends, seasonality, and other time-dependent patterns. Indexing involves establishing a proper time index, and ensuring that data points are ordered chronologically. We use "start MTU (UTC)" feature as the index, then sort the data frame to reorder them in the correct (ascending) order as shown in Figure 7.4.

```
# Set the index of the DataFrame to 'start MTU (UTC)'
df = df.set_index('start MTU (UTC)')

# Convert the index to datetime
df.index = pd.to_datetime(df.index)

# Sort the DataFrame based on the index
df = df.sort_index()
```

Figure 7.4: Code for indexing and sorting the dataset

After indexing the dataset, we want to remove all symbols ("[", ">", "]", ">") from each feature name. Otherwise, the following error will be shown when training the XGBoost model:

"ValueError: feature names must be string"

Creating lagged features

As mentioned in the beginning of this section we have experimented with both 1-hour and 24-hour lagged features. The logic of model training and target predicting is precisely the same for both of these scenarios, only the number of input features differs. Our original dataset contains 36 features, and the 1-hour lagged experiment results in 72 (2×36 = 72) features in total. The 24-hour lagged experiment results in 900 (24 × 36 = 900) features in total. The number of input features is 71 and 899 respectively for each. Table 7.1 shows an example of a 24-hour lagged feature. We can find the values for "electricity price" at the timestamp 2022-01-02 00:00:00 moved to "lag24" at the timestamp 2022-01-02 23:00:00. In our dataset this operation is implemented for each of the 36 original features.

	electricity price	lag_1	lag_2	...	lag_{24}
2022-01-02 00:00:00	47.6	NaN	NaN	...	NaN
2022-01-02 01:00:00	63.5	47.6	NaN	...	NaN
2022-01-02 02:00:00	45.6	63.5	47.6	...	NaN
2022-01-02 03:00:00	30.2	45.6	63.5	...	NaN
⋮	⋮	⋮	⋮	⋮	⋮
2022-01-02 23:00:00	34.8	30.2	45.5	....	47.6
⋮	⋮	⋮	⋮	⋮	⋮

Table 7.1: 24 lagged features based on electricity price feature

Using the 3 functions shown in Figures 7.4 and 7.5, we are able to create lagged features for the dataset(Figure 7.4) and get a list of names of created lagged features(Figure 7.5).

```python
def create_lagged_features(dataframe, lag_hours):
    # Initialize an empty list to store the lagged features
    lagged_features = []

    # Iterate over each column in the dataframe
    for col in dataframe.columns:
        # Generate lagged features for the current column
        for i in range(1, lag_hours+1):
            # Create the name of the lagged feature
            col_name = '{}_lag_{}'.format(col, i)

            # Shift the values of the current column and rename the series with
            # the lagged feature name
            lagged_features.append(dataframe[col].shift(i).rename(col_name))

    # Concatenate the lagged features along the columns axis
    lagged_features_df = pd.concat(lagged_features, axis=1)

    # Concatenate the original dataframe with the lagged features dataframe, drop
    # rows with missing values
    new_dataframe = pd.concat([dataframe, lagged_features_df], axis=1).dropna()

    # Return the new dataframe with lagged features
    return new_dataframe
```

Figure 7.5: Function for creating lagged features

Inline comments in Figures 7.5 and 7.6 explain the implementation for each function. The "create_lagged_features" function helps to create a new data frame that includes both the original features and the lagged features, which can be used as input for our direct one-step forecasting models based on the XGBoost algorithm.

```python
def get_lagged_feature_names(features, lag_hours):
    lagged_features = []
    for col in features:
        for i in range(1, lag_hours+1):
            col_name = '{}_lag_{}'.format(col, i)  # Generate lagged feature name
            lagged_features.append(col_name)  # Add the lagged feature name to the list
    return lagged_features

def get_lagged_target_name(target, lag_hours):
    lagged_target = []
    for i in range(1, lag_hours+1):
        col_name = '{}_lag_{}'.format(target, i)  # Generate lagged target name
        lagged_target.append(col_name)  # Add the lagged target name to the list
    return lagged_target
```

Figure 7.6: Functions for getting lagged feature's names

Feature engineering

We can transform our data into more meaningful and informative features by creating new features based on time. Features "hour" and "dayofweek" is created at this step to capture important characteristics and dependencies, potentially leading to improved accuracy and generalization. By analyzing the relationship between "hour" and "Day-ahead Price NO1" shown in Figure 7.7,

we arrive at the conclusion that the average everyday prices are at the highest in the mornings and late evenings, and at the lowest around noon in the year 2022.

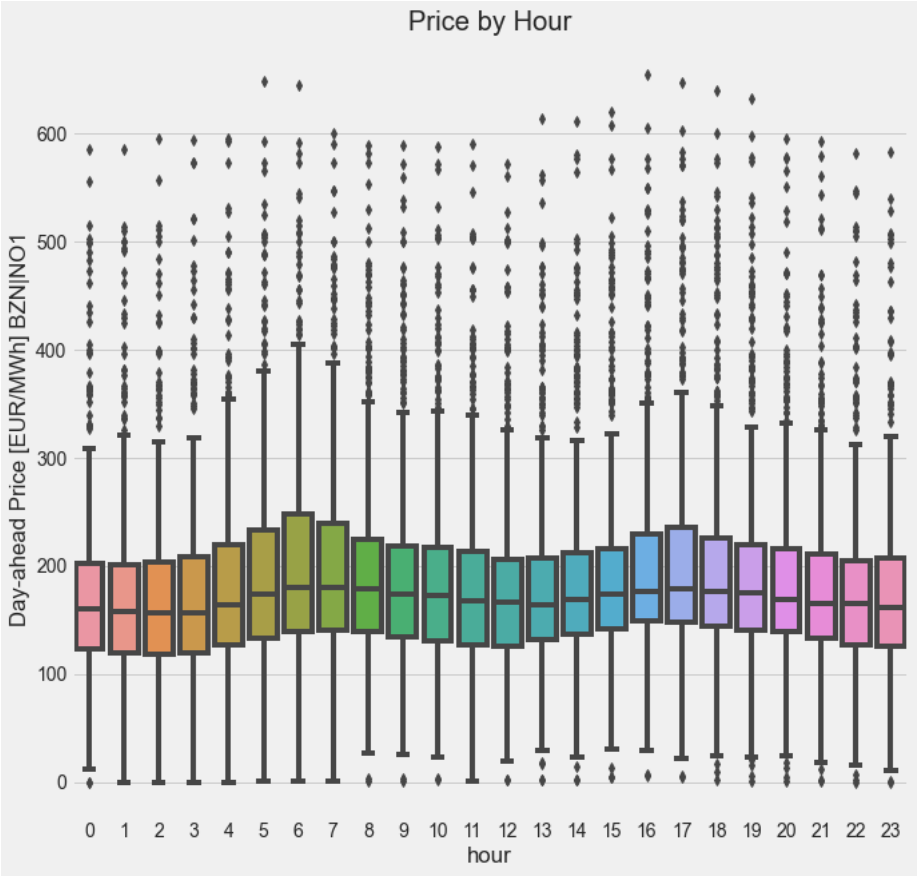

Figure 7.7: Price by hour Box plot

7.4.2 Time series cross-validation for training

For training our model we implement Time Series Cross-Validator from the Sklearn library. Figure 7.8 shows the code used for model training.

Here's a breakdown of what the code does:

1. The TimeSeriesSplit object is created with 5 splits and no gap between each split. This object will be used for cross-validation.
2. The code enters a loop that iterates over each fold in the time series cross-validation.
3. Inside the loop, the training and validation data are extracted from the original dataset (df) based on the train and validation indices provided by times-series.split (df).
4. The training dataset is split into features (X-train) and the target variable (y-train).
5. The validation dataset is split into features (X-valid) and the target variable (y-valid).
6. The XGBoost regressor is initialized with default hyperparameters.

7. The model is trained using the fit() function, passing the training features and target variables. The validation set is also provided to the eval-set parameter to monitor the model's performance during training.

8. The test dataset is prepared by extracting the features (X-test) and the target variable (y-test). X-test will be used as input data to our models to predict the y-test (Day-ahead price NO1).

```
times_series = TimeSeriesSplit(n_splits=5, gap=0)
fig, axs = plt.subplots(5, 1, figsize=(15, 15), sharex=True)
fold = 0

for train_idx, val_idx in times_series.split(df):
    train_data = df.iloc[train_idx]
    test = df.iloc[val_idx]
    val_data, test_data = train_test_split(test, test_size=0.5, shuffle=False)

    # Train dataset
    X_train = train_data[all_lagged_feature_names]
    y_train = train_data["Day-ahead Price _EUR/MWh_ BZN|NO1"]

    # Validation set, used for eval_set() method in the XGBoost library
    X_valid = val_data[all_lagged_feature_names]
    y_valid = val_data[TARGET]

    # Define the XGBoost regressor
    default_reg = xgb.XGBRegressor(
        learning_rate=0.1,
        n_estimators=1000,
        max_depth=5,
        min_child_weight=1,
        gamma=0.1,
        subsample=0.8,
        colsample_bytree=0.8,
        nthread=4,
        eval_metric='rmse',
        seed=27
    )

    # Fit the model on the training data
    model = default_reg.fit(
        X_train, y_train,
        eval_set=[(X_valid, y_valid)],
        verbose=False
    )

    # Test dataset, used for predicting
    X_test = test_data[all_lagged_feature_names]
    y_test = test_data[TARGET]
```

Figure 7.8: Code for training model

We visualize the training based on 5 folds using 5 subplots, each subplot shows the train, validation, and test dataset portion in different colors at each fold (Figure 7.9). The values on the left side

along the y-axis are "Day-ahead price NO1" and the values on the x-axis are hours. "Day-ahead price NO1" does not have any important role in this plot, one feature was chosen for being able to visualize the splitting and training process. This process applies to every feature in the dataset. The training set, validation set, and testing set are 7300, 730, and 729 hours respectively at the last fold. After finishing training the XGBoost model, the next step is to predict the x-test set by using .predict( ) function for the "default-reg". The forecasting results are explained in Chapter 8 Performance Evaluation.

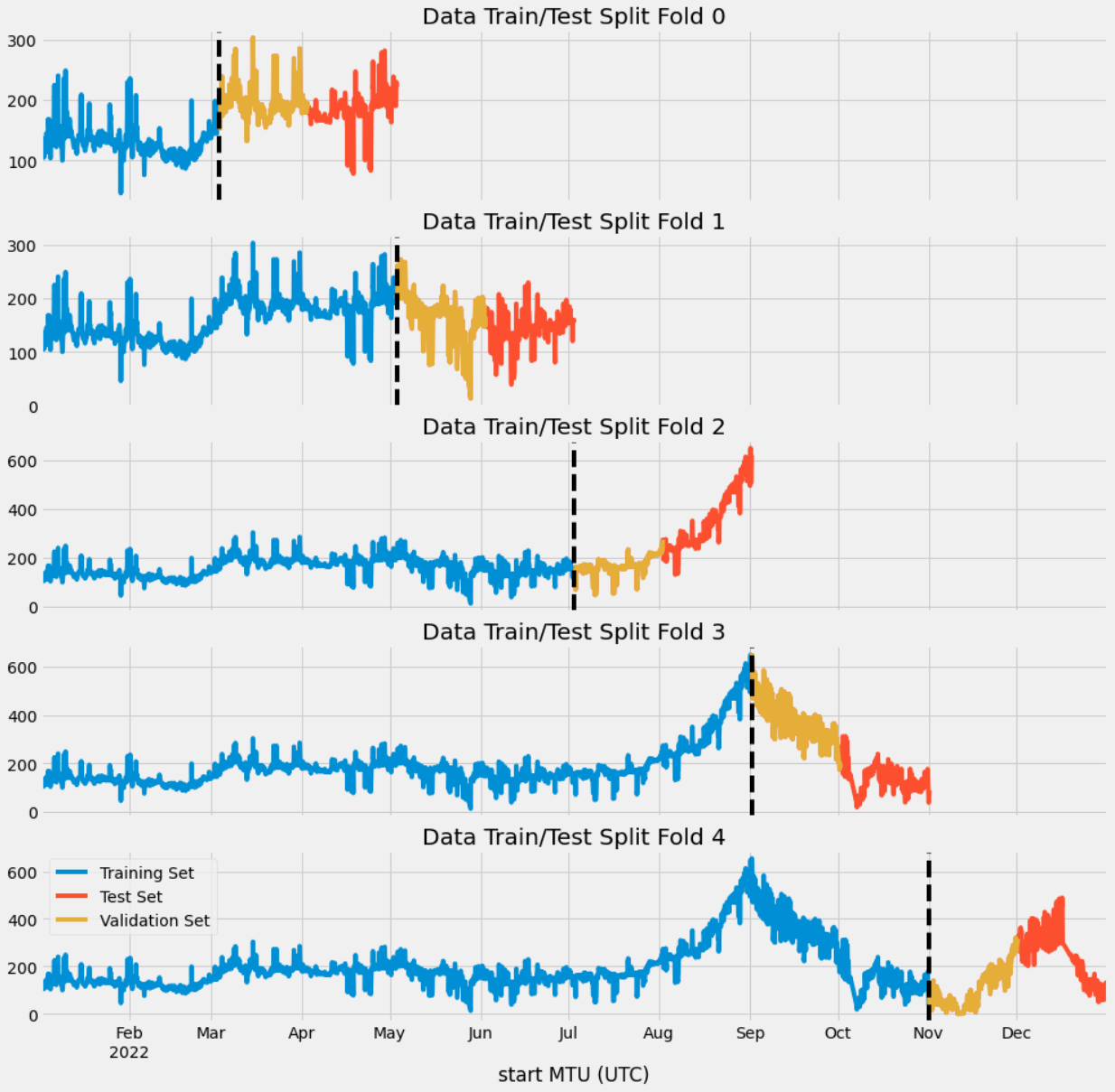

Figure 7.9: 5 Fold time series split

7.4.3 XGBoost hyperparameter tuning

By performing hyperparameter tuning, we efficiently explore a subset of the hyperparameter space and identify the best hyperparameters for our XGBoost models. Figure 7.10 shows a parameter grid used for hyperparameter tuning.

The parameter grid is a dictionary that specifies the values to be used for different hyperparameters in our XGBoost models. Each key in the dictionary represents a hyperparameter, and the corresponding value is a list of possible values to try. The idea behind using a parameter grid is to systematically explore different combinations of hyperparameter values to find the best configuration for the model. By specifying multiple values for each hyperparameter, the random search algorithm will try all possible combinations and evaluate the performance of the model using each combination.

```
# Define the hyperparameter grid
param_grid = {
    'n_estimators': [100, 300, 500 , 700, 1000],  # Number of trees in the forest
    'learning_rate': [0.1, 0.05, 0.01, 0.005, 0.001],  # Learning rate
    'max_depth': [3, 4, 5, 6, 7],  # Maximum depth of each tree
    'subsample': [0.7, 0.8, 0.9, 1.0],  # Subsample ratio of the training instances

     # Subsample ratio of columns when constructing each tree
    'colsample_bytree': [0.7, 0.8, 0.9, 1.0],


    # Minimum loss reduction required to make a further partition on a leaf node
    'gamma': [0, 0.05, 0.1, 0.15, 0.2],

    'reg_alpha': [0, 0.05, 0.1, 0.3, 0.5],  # L1 regularization term on weights

    'reg_lambda': [0, 0.05, 0.1, 0.3, 0.5],  # L2 regularization term on weights

     # Minimum sum of instance weight (hessian) needed in a child
    'min_child_weight': [1, 2, 3, 4, 5],

    # Minimum loss reduction required to make a further partition on a leaf node of the tree.
    'reg_gamma': [0, 0.1, 0.2, 0.3, 0.4],

}
```

Figure 7.10: Hyperparameter grid

Random search and grid search

Two popular approaches for performing hyperparameter tuning are random search and grid search. The following comparison and conclusion are made based on our reading [11] on hyperparameter tuning techniques for regression problems.

Random search is a technique where different combinations of hyperparameters are randomly sampled from a defined search space. It has some advantages. For example, it works well with large hyperparameter spaces because it explores various combinations in a random manner. This can help quickly identify good hyperparameter configurations, especially if the search space is well-defined. Random search is also useful when we do not know the exact impact of each hyperparameter in advance. However, it has some limitations. Since it randomly samples hyperparameter

combinations, there is no guarantee that it will adequately explore all important hyperparameters. It might require more iterations to find the optimal set of hyperparameters compared to grid search.

On the other hand, grid search involves systematically trying out all possible combinations of hyperparameters within a predefined grid. This approach has its own set of advantages. First, it guarantees that all combinations within the defined grid will be explored, leaving no stone unturned. Grid search is particularly useful when we have prior knowledge about the impact of specific hyperparameters on the model's performance.
However, grid search also has some drawbacks. It can be computationally expensive, especially if the search space is large or the training time is long. Additionally, if many hyperparameters are irrelevant or have little effect on the model's performance, grid search may not be efficient as it exhaustively tries all combinations.

In conclusion, random search offers flexibility and efficiency for large or unpredictable search spaces, while grid search is more suitable when the impact of specific hyperparameters is known. Both techniques were tried and the random search outperformed the grid search. Because random search can be advantageous due to the unpredictable nature of time-dependent data. By exploring random combinations of hyperparameters, it has a better chance of finding effective configurations that are well-suited to the specific time series data. However, it might take more iterations to find the best hyperparameters compared to grid search.

```
# Perform random search
random_search = RandomizedSearchCV(
    xgb_regressor,
    param_distributions=param_grid,
    n_iter=10,  # Number of parameter settings that are sampled
    scoring='neg_mean_squared_error',  # Scoring metric for evaluation
    cv=5,  # Number of cross-validation folds
    verbose=1,
    random_state=42
)

# Fit the model with random search
random_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = random_search.best_params_
print("Best hyperparameters:", best_params)

# Evaluate the best model on the test set
best_model = random_search.best_estimator_
print(best_model)
```

Figure 7.11: Random search

For implementing random search (Figure 7.11), first, an instance of "RandomizedSearchCV" is created by specifying the XGBoost regressor and a parameter grid that defines the ranges of values to sample from. The number of parameter settings to sample "n-iter", the scoring metric for

evaluation "scoring", the number of cross-validation folds "cv", verbosity level, and random seed are also specified.
The random search is then performed by calling the fit() method on the created "RandomizedSearchCV" object, passing the training data. This fits the XGBoost regressor with different randomly sampled parameter settings and evaluates their performance using cross-validation.

After the random search is complete, the best hyperparameters found during the search are retrieved using the "best-params-" attribute of the random search. Additionally, the best model obtained from the random search can be accessed using another attribute of the random search named ".best-estimator-".

Finally, the best hyperparameters and the best model are printed to the console, providing the optimized hyperparameter values and the corresponding trained model that performed the best according to the evaluation metric used, which is NMSE (negative mean squared error). NMSE, as the name implies, is the negative value of the MSE, MSE is explained in Section 4.10 Evaluation Metrics. The model with the lowest NMSE is considered the best fit for our dataset and hyperparameters. The hyperparameter tuning results are explained in Section 8.5 XGBoost Results.

7.5 Support Vector Regression

This section presents a high-level overview of the implementation process of the Support Vector Regression (SVR) model for time series forecasting. The process comprises transforming the time series data into a supervised learning problem, training the SVR model, evaluating its performance, and finally deploying the model for forecasting future data points.

7.5.1 Initial data prepossessing

Preparing the data for SVR involves several essential steps to ensure the model's efficiency and accuracy. The dataset for this study has been introduced in Chapter 5 "Data Acquisition", and the target feature for our SVR model is "Day-ahead Price [EUR/MWh] BZN|NO1". This feature represents the day-ahead energy price, which is the primary forecasting target in this study.

Train validation test split

The dataset was divided into three segments: training, validation, and testing. This division is crucial in model development and evaluation. The majority of the data, 80%, was allocated for training the model. The validation set, comprising 10% of the data, was used for tuning the model parameters. The remaining 10% was reserved for the final evaluation of the model's performance on unseen data. The data was shuffled randomly prior to the split to ensure the uniformity of data distribution in each set.

Creating lagged features

Time series data is inherently sequential, meaning that the order of observations is critical, and often, past observations influence future ones.
When working with machine learning models like SVR, which don't naturally handle temporal

dependence, we need to engineer features that encapsulate this sequential information. This is where lagged features come into play.
Lagged features are simply previous time steps of the target variable, used as input variables. For instance, if we're predicting the day-ahead energy price, a lagged feature might be the energy price from one hour ago, or one day ago, depending on the specific problem and data.

In this study, we created 24 lagged features. This means each input sample to our model includes the energy prices from the preceding 24 hours. The choice of 24 lags allows the model to capture daily patterns in the data, which are critical for predicting the day-ahead price.

By using lagged features, we're essentially treating our time series problem as a supervised learning problem, where the model learns to map a set of input features (the lagged observations) to an output (the next time step). This enables the SVR model to effectively learn from the past and make accurate future predictions.
However, it's important to note that the choice of the number of lagged features (lag order) is problem-specific and should be determined carefully. A lag order that's too small might not capture enough historical information, while a lag order that's too large may make the model overly complex and computationally expensive. In this case, a lag order of 24 was found to be effective, but for other problems, a different lag order may be optimal.

Hyperparameter tuning

Hyperparameters are crucial components of machine learning models as they dictate the structure of the model and the way learning occurs. In the case of Support Vector Regression, we focus on several key hyperparameters: the kernel type, the penalty parameter C, the gamma coefficient for the RBF kernel, and the epsilon parameter in the loss function.

The kernel type is pivotal in SVR as it determines the type of transformation applied to the data for learning the optimal boundary. There are several types of kernels, such as linear, polynomial, and radial basis function (RBF). Each kernel has its strengths and is suited to different types of data. In this study, we considered multiple kernel types and chose the one that performed best on the validation set.

The regularization parameter, denoted as C, is instrumental in balancing the dual objectives of minimizing the error on the training set and enhancing the model's generalizability on unseen data. A smaller value of C results in a smoother decision boundary, whereas a larger C prioritizes the correct classification of all training instances, allowing the model more liberty in choosing more samples as support vectors.

The gamma parameter is specific to the RBF kernel and defines how much influence a single training example has. In other words, it determines the shape of the decision boundary. A small gamma will produce a more flexible decision boundary, while a large gamma will create a more complex decision boundary.

The epsilon parameter defines the epsilon-tube within which no penalty is associated with the training loss function. This parameter can affect the number of support vectors used to construct the regression function. Tuning this parameter can help reduce the model's sensitivity to noise in

the training data.

In the context of our study, the optimal values for these hyperparameters were determined through a systematic grid search. This exhaustive process ensured that our SVR model was optimally configured for our specific task, thereby maximizing its predictive performance.

The subsequent chapter will delve into the details of training the SVR model, assessing its performance, and deploying it for future forecasting.

7.6 Linear Regression

7.6.1 Feature engineering

For our linear regression models we implement 7 new features:

1. Temporal features

   Month is added as a feature to capture seasonal variations present in the data.

   Day is included to allow the model to capture trends within the months.

   Hour is included to capture the trends of the days which is particularly useful as we know electricity prices tend to be lower at off-peak hours.

   Weekday let us differentiate patterns based on the day of the week. This could be useful to look for variations between weekdays and weekends.

2. Technical indicators

   Moving average deviation (MAD): This feature indicates the deviation of the current value from its moving average. This feature helps the model capture volatility.

   Percentage range (PR): The PR indicator represents the range between the highest and lowest values within each day. This feature captures the relative variability within each 24-hour period.

   Percentage price change moving average (PPCMA): This indicator calculates the percentage change between the previous 24-hour period and the current. It provides an insight into the relative price movement and helps in capturing momentum or trend reversal patterns.

```
# Extract the year, month, day and hour from the 'ds' column
data['month'] = data['ds'].dt.month
data['day'] = data['ds'].dt.day
data['hour'] = data['ds'].dt.hour
data['weekday'] = data['ds'].dt.weekday

# Technical indicators
# Moving average deviation
data['MAD'] = data['y'].rolling(window=24).mean() - data['y']
# Percentage range
data['PR'] = (data['y'].rolling(window=24).max() - data['y'].rolling(window=24).min()) / data['y'].rolling(window=24).mean()
# Percentage price change moving average
data['PPCMA'] = data['y'].rolling(window=24).mean().pct_change()
```

Figure 7.12: Implementation of new features.

7.6.2 Data scaling

To guarantee equitable feature influence, quick convergence, and proper regularization, the data is scaled using "StandardScaler" from scikit-learn. "StandardScaler" balances the influence of the features by standardizing them to have zero mean and unit variance, which also promotes convergence and allows for fair comparison of the coefficients and assessment metrics. Regression model efficacy and interpretability are improved.

Other scaling techniques including normalization and min-max scaling was also tested. However, the use of standardization alone performed the best. The standard scaler is fit to the training data and used to transform both training and testing. The column names are extracted for later use as the scaler returns a "ndarray" also known as an N-dimensional array.

```
# Scaling the data with standard scaler
Sc = StandardScaler()

cols = X_train.columns
# Fitting scaler to training data and transforming
X_train = Sc.fit_transform(X_train)
# Transforming testing data
X_test = Sc.transform(X_test)
```

Figure 7.13: Fitting scaler to training data and transforming both training and testing.

7.6.3 Data splitting

The data is split using scikit-learn's train_test_split with shuffle set to false as the price moves fluently through the data points. The current value is often dependant on the past values and shuffling would disrupt this natural time dependency. There would also be leakage when future observations end up in the training set. The data is split into training 90% and testing 10%. Validation is skipped because scikit-learn's gridsearchCV uses kFold cross-validation which is used in the implementation of these models.

7.6.4 Lagging features

To get a good representation of how the models will do in a real-life scenario the features are shifted so that the model does not have data points representing the actual time it is trying to predict. It would not make sense to give the model the price for NO5 for the time period it is trying to predict for the reason that it would just mimic the changes in price from NO5 to NO1.
To perform direct multi-step forecasting, the features are lagged to create a target for each hour the next 24-hour period. The implementation for the LR models is the same as seen in Figure 7.1.

7.6.5 Feature selection

Lasso and ElasticNet offer an automated approach to feature selection by applying the regularization techniques, effectively shrinking some coefficient to zero. This property makes them well suited for identifying the most important features from our dataset. This enhances the interpretability of our models by focusing on a reduced set of more important features simultaneously reducing the training time and overcoming multicollinearity. ElasticNet combines the L1 (Lasso) and L2 (Ridge)

regularization terms letting it capture both groupings of correlated predictors as Lasso does and handle situations where there are many correlated predictors as Ridge does. The features of these feature selection techniques help prevent overfitting while improving the generalization capability of the models.

The implementation of both these techniques is done by creating a pipeline that contains the function "SelectFromModel" which takes an estimator as input. In this case, it is the tuned models. The second part of the pipeline is a model that is used to test what threshold the coefficients should be higher then to be kept. The pipeline is fed through "GridSearchCV" to find what threshold gave our model the best performance.

```
# Create a pipeline with feature selection
pipeline = Pipeline([('select', SelectFromModel(Lasso())), ('lasso', Lasso())])

# Define a range of threshold values to test
thresholds = np.arange(0.1, 1.0, 0.1)

# Create a grid search to find the best threshold value
grid_search = GridSearchCV(pipeline, {'select__threshold': thresholds}, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# Print the best threshold value
print("Best threshold: {:.2f}".format(grid_search.best_params_['select__threshold']))
```

Figure 7.14: Finding best threshold for selecting features.

```
# Change X_train and test to only contain the selected features
X_train_selected = grid_search.best_estimator_.named_steps['select'].transform(X_train)
X_test_selected = grid_search.best_estimator_.named_steps['select'].transform(X_test)
```

Figure 7.15: Changing input data to selected features.

Both of the models got the best results with a threshold of 0.9, meaning that Lasso only kept 28 features where the significant coefficients were designated the price in NO5 and one of the technical indicators MAD. While ElasticNet has the same threshold it also grants more features high coefficients and therefore kept 276 of the 1008 original features. Where some of those high coefficients were given to: "Day-ahead Price [EUR/MWh] BZN|SE3", "Actual Total Load [MW] - BZN|NO5", "Hydro Run-of-river and poundage - BZN|NO1", "Hydro Run-of-river and poundage - BZN|NO2", "Waste - BZN|NO2", "Other renewable - BZN|NO3", "Hydro Run-of-river and poundage - BZN|NO5", "CBF BZN|NO1 > BZN|NO2 [MW]", "CBF BZN|NO1 > BZN|SE3 [MW]", "MAD" and "PR".

7.6.6 Hyperparameter tuning

The parameters for all LR models were found using a combination of exhaustive search (GridSearchCV) and changing the parameters to test for each iteration to values closest to the optimal parameter of the previous iteration.

For these models, the only tune-able parameters are the regularization terms. That means scikit-learn's LR model OLS has no hyperparameter tuning necessary. Scikit-learn's Ridge has one parameter, the complexity parameter $\alpha \geq 0$ that controls the amount of shrinkage making the

coefficients more robust to collinearity. The parameter was explored from 0.001 to 1000 in different iterations using exhaustive search as seen in Figure 7.16.

```
params = {
    'alpha': [0.01, 0.1, 0.5, 0.8, 0.9, 1, 1.1, 1.5,  2, 10, 20, 50, 100, 1000]
}

ridge = Ridge()

ridge_regressor = GridSearchCV(ridge, params, scoring='neg_mean_squared_error', cv=5)

ridge_regressor.fit(X_train_selected, y_train)

print(ridge_regressor.best_params_)
```

Figure 7.16: Hyperparameter tuning of ridge regression.

The α parameter of scikit-learn's Lasso controls the amount of sparsity among the coefficients by multiplying the L1 term. The parameter was explored between 0.001 and 1 as seen in Figure 7.17. Lasso additionally has a parameter "max_iter" which was set to 100000 to let the model converge, meaning that there is no more improvement for the model at each iteration of training.

```
params = {
    'alpha': [0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 0.7, 1],
    'max_iter': [100000]
}

lasso = Lasso()

lasso_regressor = GridSearchCV(lasso, params, scoring='neg_mean_squared_error', cv=5)

lasso_regressor.fit(X_train_selected, y_train)

print(lasso_regressor.best_params_)
```

Figure 7.17: Hyperparameter tuning of lasso regression.

Scikit-learn's ElasticNet takes to hyperparameters: α and "l1_ratio". Where α is a constant which multiplies the penalty terms and l1_ration being a convex combination of the L1 and L2 term. When "l1_ratio" = 0 the penalty is an L2 penalty while "l1_ratio" = 1 means the penalty is an L1 penalty. α was searched between 0.001 and 10 while "l1_ratio" was tested between 0.01 and 1. Additionally ElasticNet's "max_iter" was set to 100000 as well for convergence as seen in Figure 7.18.

```
# Create an Elastic Net Regression model with the optimal hyperparameters
params = {
    'alpha': [0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 1, 1.1, 10],
    'l1_ratio': [0.01, 0.1, 0.5, 0.8, 0.9, 1],
    'max_iter': [100000]
}

elastic_net = ElasticNet()

elastic_net_regressor = GridSearchCV(elastic_net, params, scoring='neg_mean_squared_error', cv=5)

elastic_net_regressor.fit(X_train_selected_e, y_train)

print(elastic_net_regressor.best_params_)
```

Figure 7.18: Hyperparameter tuning of elasticNet regression.

7.7 Summary Implementation

This chapter provides a comprehensive overview of how each algorithm was implemented, the environment they were implemented, and the tools used.

The implementation of the machine learning algorithms for time series prediction of electricity market prices in this study relied on the utilization of the Python programming language. Specifically, the Jupyter Notebook environment was employed to facilitate the development and execution of the code. Python was chosen due to its widespread adoption in the field of ML and its extensive ecosystem of libraries for data analysis and model training.

To leverage the capabilities offered by Python, several widely-used Python libraries were incorporated into the implementation process. These libraries provided crucial functionalities for data preprocessing, feature engineering, model selection, and evaluation. Notable libraries employed included NumPy, Pandas, and Scikit-learn, among others. Each library played a vital role in enhancing the efficiency and effectiveness of the implemented algorithms.

Furthermore, to promote collaboration and version control among the research team, a GitHub repository was utilized as a central hub for code sharing and synchronization. GitHub allowed multiple researchers to work concurrently on the project, facilitating efficient collaboration and ensuring the integrity and traceability of the implemented code.

By harnessing the power of Python, leveraging Jupyter Notebook, and utilizing various Python-based code libraries, this implementation phase achieved a robust and efficient framework for training and evaluating machine learning models. The utilization of a collaborative platform like GitHub fostered seamless teamwork and code management throughout the research process.

The preprocessing techniques and parameter tuning strategies used were chosen to ensure that the models are well-suited to handle the time-dependent nature of the data, enabling the models to achieve the optimal accuracy and reliability of their predictions. This includes the creation of lagged features that enable multi-step forecasting. Various preprocessing steps were undertaken to tailor the data according to the requirements of each algorithm. Additionally, the parameters of

the models were carefully tuned to optimize their performance, taking into account the available parameter options specific to each algorithm.

Chapter 8

Performance Evaluation

In this Chapter, we delve into a thorough performance evaluation of different ML models for day-ahead price prediction in the wholesale energy market. We explore the performance of RFR (Random Forest Regression), SVR (Support Vector Regression), XGBoost (Extreme Gradient Boosting), and LR (Linear Regression) models, with a particular focus on multi-step forecasting time series models. Only the XGBoost models are based on the one-step forecasting technique.

We begin by closely examining the RFR model. We evaluate its baseline performance, where default parameters are used and 24-hour lagged features are employed. Our assessment involves analyzing the model's behavior on both the training and test datasets. We uncover indications of overfitting in the baseline RFR model, suggesting the need to enhance its ability to generalize predictions beyond the training data.

Moving forward, we turn our attention to the SVR model. By examining the baseline SVR model, trained with default parameters, we assess its performance on the training, validation, and test datasets. However, we observe significant challenges in accurately forecasting future values within the energy market using SVR, as suggested by the high error values.

Next, we explore the XGBoost model, which offers promising forecasting capabilities. We evaluate both 1-hour lagged and 24-hour lagged one-step forecasting models. We begin by analyzing the performance of the base XGBoost model, which is trained using default hyperparameters, on the training and validation datasets. Subsequently, we assess the tuned XGBoost model, which undergoes hyperparameter tuning, on the combined train and validation sets and evaluate its performance on independent test data.

Lastly, we delve into the evaluation of LR models, encompassing Ordinary Least Squares (OLS), Ridge Regression, and Lasso Regression.

We employ 4 evaluation metrics to assess the performance of RF, SVR, XGBoost and LR, focusing on both one-step and multi-step forecasting time series models. Throughout our evaluation, we carefully analyze the accuracy, generalization capabilities, and limitations of each model. Our primary objective is to determine the most suitable model for accurate and reliable day-ahead price predictions.

8.1 Experiment Environment

To evaluate our time series forecasting models, specifications for each computer system for building and running each model were done, as displayed in Table 8.1. Model evaluation is not heavily dependent on hardware. One can perform evaluation tasks with modest hardware as long as the computer system meets the minimum requirements for running the necessary software. All these computer systems have Solid State Drives (SSDs) that provide faster data access compared to traditional Hard Disk Drives (HDDs).

Model	**OS Name**	**CPU**	**RAM**	**Motherboard**
RF	Microsoft Windows 10 Pro	AMD Ryzen 5 5600X, 6-Core Processor, 2 logical processors	64 GB	HP 8643 (Edoras) B360
XGBoost	Microsoft Windows 11 Home	AMD Ryzen 7 3700X 8-Core Processor, 3.59 GHz, 8 cores, 16 logical processors	16GB	HP 8643 (Edoras) B360
SVM	Microsoft Windows 10 Pro	AMD Ryzen 5 5600X, 6-Core Processor, 2 logical processors	32GB	ASUS ROG Strix B550-F
LR	Microsoft Windows 11 Home	Intel Core i7 9700f 3.0 Ghz, 8 cores, 8 logical processors	16GB	Rog STRIX B360-F

Table 8.1: PC environments for 4 implemented models.

8.2 Performance Metrics

Performance metrics: RMSE, MAPE, MSE, and MAE are used to evaluate model performance and compare each optimal model with the respective base model performance. These metrics are widely used and the most popular metrics in research papers. The formula for each of them is explained in Section 4.10. since each metric has its strengths and limitations. We will use multiple evaluation metrics to gain a comprehensive understanding of the model's performance and choose the best model.

Among 4 performance evaluation metrics, MSE measures the average squared difference between the predicted and actual values. RMSE measures the square root of the average squared difference between the predicted and actual values. MAE measures the absolute difference between the predicted and actual values. MAPE measures the average absolute percentage difference between the predicted and actual values.

8.3 RFR Results

8.3.1 Baseline RFR model results

The RFR baseline model was trained with the default parameters and 24 lagged features. This model had a training process that lasted 395 seconds until completion. Predictions were made with the training data and the test data so that the performance could be compared to check for overfitting, and the results are visualized in Table 8.4.

	MAE	MAPE (%)	MSE	RMSE
Training Data	2.4419	1.41	18.1545	4.2608
Testing Data	20.9706	8.93	926.6677	30.4412

Table 8.2: RFR baseline model results for MAE, MAPE, MSE, and RMSE.

This model scored 2.44 on MAE, 1% on MAPE, 18.15 on MSE, and 4.26 on RMSE with the training data. The model scored 20.97 on MAE, 9.0% on MAPE, 926.67 on MSE, and 30.44 on RMSE with the test data. Upon analyzing the model's performance on both the training and test datasets, it is evident that the model is exhibiting overfitting tendencies as it is performing significantly better on the training set when compared to the test set. This is evident from the fact that the model is producing lower error values on the training set compared to the test set. Moreover, it is noteworthy that the absolute error values on the test set are quite high, with the MAE being 20.97 and RMSE being 30.44, which clearly implies that the model's predictions are deviating substantially from the actual values.

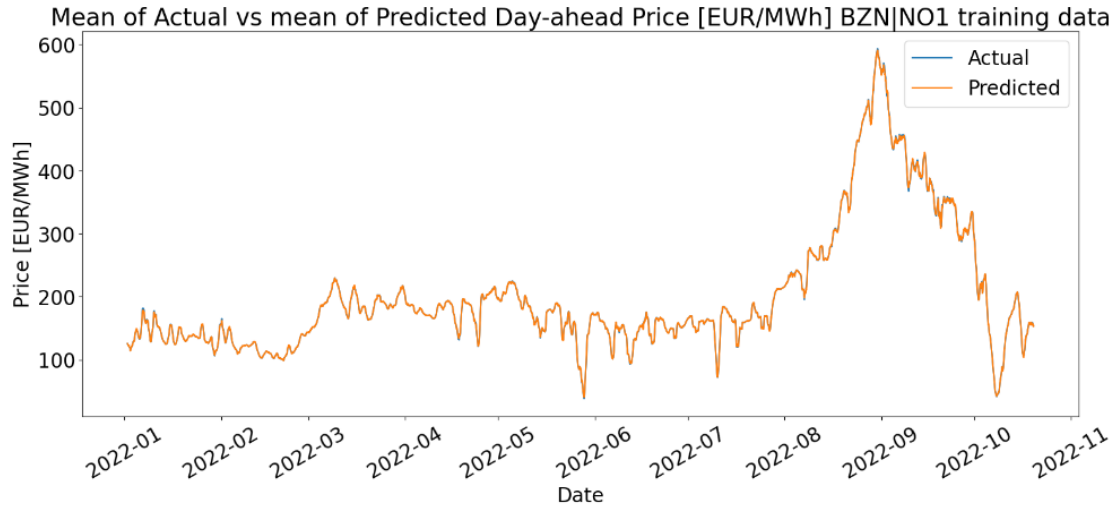

Figure 8.1: Graph of baseline RFR model predictions and actual values with training data.

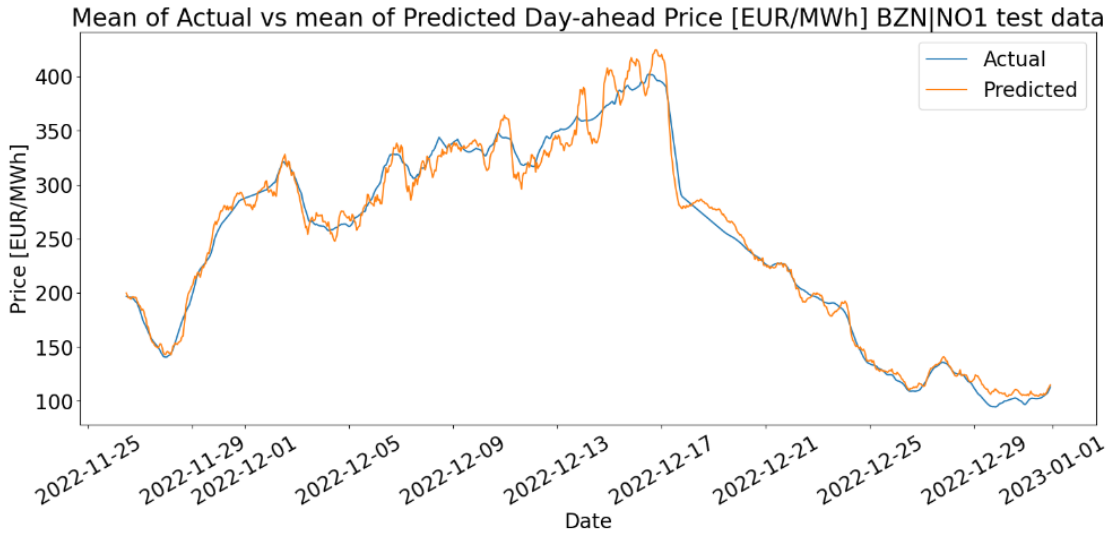

Figure 8.2: Graph of baseline RFR model predictions and actual values with test data.

Figures 8.1 and 8.2 provide a visualization of the overfitting of the model. Specifically, these figures display the mean of the predicted values plotted against the mean of the actual values for the training data and test data, respectively. In Figure 8.1, which depicts the training data, the two plots for the mean of actual and mean of predicted values are closely aligned. However, in Figure 8.2, which depicts the test data, the two plots deviate more, indicating that the model is overfitting on the training data and failing to generalize well to the test data.

8.3.2 Tuned RFR model results

The optimal hyperparameters identified during hyperparameter tuning were "n_estimators" set to 500, "min_samples_split" set to 2, "min_samples_leaf" set to 1, "max_features" set to log2, and "max_depth set" to 30. The tuned RFR model had a training process that lasted 395 seconds until completion. The results are visualized in Table 8.3.

	MAE	MAPE (%)	MSE	RMSE
Training data + Validation data	2.72	4%	19.64	4.43
Testing data	49.62	19%	4225.75	65.01

Table 8.3: RFR tuned model results for MAE, MAPE, MSE, and RMSE.

The results visualized in table 8.3 displays a score of 4.43 on MAE, 2.72 on MAPE, 0.04 on MSE, and 19.64 with the training data and validation data. The tuned RFR model scored 20.97 on MAE, 0.09 on MAPE, 926.67 on MSE, and 30.44 on RMSE with the test data.

These results suggest that while the tuning process improved the model's performance on the training data, it did not generalize well to the unseen test data. The tuned model's performance on the test data is noticeably worse than that of the baseline model, with significantly higher MAE, MAPE, MSE, and RMSE values.

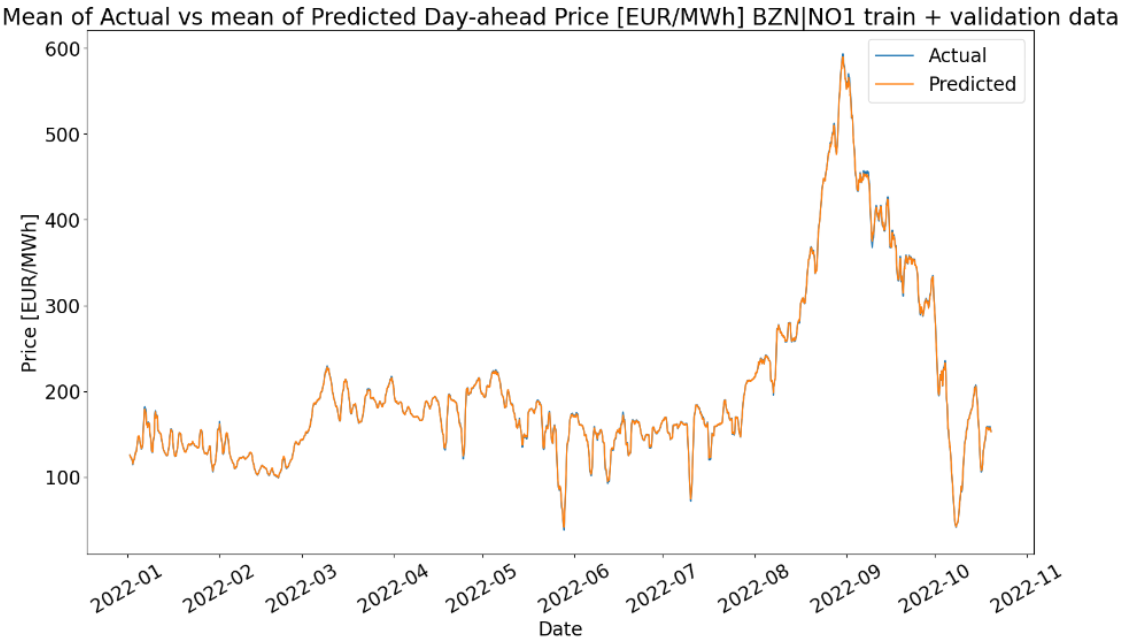

Figure 8.3: Graph of tuned RFR model predictions and actual values with training and validation data.

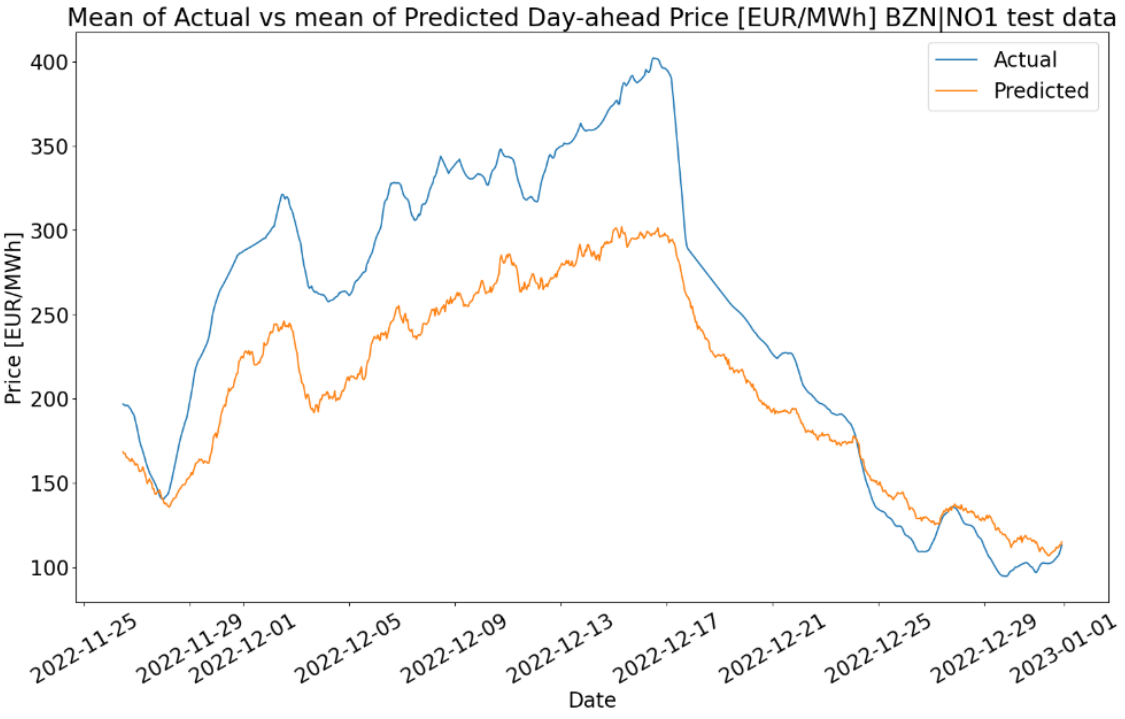

Figure 8.4: Graph of baseline RFR model predictions and actual values with test data.

Figures 8.3 and 8.4 provide a visualization of the increased overfitting of the model. These figures display the mean of the predicted values plotted against the mean of the actual values for the

training and validation data, and the test data, respectively. In Figure 8.3, which depicts the training and validation data, the two plots for the mean of actual and mean of predicted values are closely aligned. However, in Figure 8.4, which depicts the test data, the two plots deviate significantly, indicating that the model is overfitting the training and validation data and failing to generalize well to the test data. The gap between actual vs. predicted prices increases from the baseline model to the tuned model, indicating that the overfitting increases.

This performance is not entirely unexpected, RFR models have a tendency to overfit, particularly in cases where the data is characterized by significant noise or volatility which the exploratory data analysis identified the presence of. Furthermore, RFR struggles to capture sudden changes or short-term fluctuations in highly volatile datasets. The exploratory data analysis identified larger and sudden shifts in electricity prices, which makes RFR less likely to generalize on the data.

8.4 SVR Results

8.4.1 Baseline SVR Model Results

The baseline model employed default parameter settings for the SVR, serving as a foundation for assessing the benefit of tuning. The performance of the model on the training, validation, and test datasets was assessed using four metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

The results of the baseline model were as follows:

	MAE	MAPE (%)	MSE	RMSE
Training data	108.2472	328.9059%	21993.1941	148.3010
Testing data	106.4051	280.7304%	21340.4368	146.0836
Validation data	114.1476	44.9164%	1336.6663	36.5604

Table 8.4: SVR baseline model results for MAE, MAPE, MSE, and RMSE.

Despite the simplicity of this model, it yielded a reasonable baseline from which the tuned model could be compared.

8.4.2 Tuned SVR Model Results

Results and interpretations

Subsequent to the evaluation of the baseline model, a more sophisticated approach was pursued, employing a rigorous process of hyperparameter tuning to optimize the performance of the SVR model.

The results of the tuned model were as follows:

	MAE	MAPE (%)	MSE	RMSE
Training data	31.0795	92.7259%	1334.7892	36.5347
Testing data	32.1826	84.9603%	1417.0182	37.6433
Validation data	106.4051	280.7304%	21340.4368	146.0836

Table 8.5: SVR tuned model results for MAE, MAPE, MSE, and RMSE.

The tuned SVR model demonstrated a marked improvement across all performance metrics compared to the baseline model, reflecting the benefits of a refined and systematic model-tuning approach.

While these results are promising, it is important to interpret them with caution. The improvement observed in the training and validation set performance suggests that the model is learning the underlying pattern effectively. However, the generalizability of the model to unseen data, as reflected by the test set performance, is the ultimate measure of its predictive power. Given the complex and often unpredictable nature of energy markets, further investigation and possibly more advanced modeling approaches might be needed to capture the full dynamics of the 'Day-ahead Price [EUR/MWh] BZN|NO1' effectively in the future.

8.4.3 Understanding the results

MAE

The MAE (Mean Absolute Error) quantifies the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The smaller the MAE, the better the model's performance.

In the baseline model, the training, validation, and test data had MAE values of 108.2472, 114.1476, and 106.4051, respectively. This shows that the model, on average, was off by approximately 108 to 114 units for the day-ahead price.

In contrast, the tuned model's MAE was significantly lower, approximately 31 units for all three datasets. This significant decrease in the MAE indicates that the tuned model is considerably more accurate in its predictions.

MSE and RMSE

MSE (Mean Squared Error) and RMSE (Root Mean Squared Error) are two other metrics that measure the average squared difference between the estimated values and the actual value. RMSE is the square root of MSE. They are particularly helpful in situations where we want to penalize larger errors.

The MSE and RMSE values for the baseline model were quite high, especially compared to the tuned model. For instance, the test set's RMSE decreased from 146.0836 in the baseline model to 37.6433 in the tuned model. The substantial decrease in these values indicates that the tuned model is less prone to large errors in prediction.

MAPE

MAPE (Mean absolute percentage error) expresses the forecast errors as a percentage, and it's commonly used to compare the accuracy of different forecasting methods on a single dataset.
For the baseline model, the MAPE values were quite high across all datasets, particularly the training set with a MAPE of 328.9059%. The tuned model, however, was able to bring the MAPE values down significantly, suggesting a much better predictive accuracy relative to the size of the actual values.

Though it was brought down there is still an extremely high rate of error with MAPE which can be due to several untested factors.

- **Inadequate model**
  The chosen model itself may have not be capable of capturing the underlying patterns in the data.
- **Inadequate amount of data**
  The the possibility of there not being enough data to train the model effectively.
- **Volatility of data**
  Because of the volatility of the data there is a possibility that more data is needed to train the model and make it able to capture the underlying patterns and trends.

To sum up, the tuned model's superior performance on all these metrics indicates that it is more effective in predicting the day-ahead price compared to the baseline model. It suggests that the hyperparameters chosen for the tuned model indeed offer a more accurate representation of the underlying data. However, as always in predictive modeling, care must be taken not to over-interpret the results. Real-world factors can often introduce unanticipated variability, and the model's predictions should be used as part of a broader set of decision-making tools.

However, the high rates of failure may mean that SVR may not be the algorithm that works best for predicting future values with the data available.

8.5 XGBoost Results

This section contains evaluations of our models built by implementing the XGBoost regressor. We are evaluating how well our models predict the target feature "day-ahead price NO1", for a given set of data (input features). Every evaluation is done using 4 metrics mentioned in Section 8.2. These models are built based on the direct one-step forecasting approach explained in Section 7.4.

We experimented with both 1-hour and 24-hour lagged features. For both of these experiments, the evaluation is conducted in two main steps. First, we start by evaluating a base model built by default XGBoost hyperparameters. After conducting hyperparameter tuning we evaluate the tuned model trained on a combination of train and validation sets. This combination may result in better generalization and help score better results when predicting the test set.

8.5.1 Baseline XGBoost model results: 1-lagged features

The base model is the first model built using the default XGBoost regressor hyperparameters, which are explained in Subsection 7.4.2.

Evaluation of predictions on the train data

After splitting the dataset into training, validation, and testing sets and training the model on the training data (7300 hours), we begin evaluating the predictions on the already-seen training data. There are several important reasons for doing this, such as detecting overfitting, establishing performance benchmarks, and identifying areas for improvement.

Figure 8.5 displays the actual "Day-ahead price NO1" in the green plot and the predicted value in the blue plot. This figure indicates excellent predictions on the training data. Figure 8.6 illustrates the difference between the actual and predicted values, with differences shown on the y-axis as "Price Difference [EUR/MVh]". The differences are extremely small, with only a few peaks exceeding a 1 Euro difference.

Table 8.6 contains the performance results for this model during the "Base model training" phase. The low values for RMSE, MAE, MSE, and MAPE demonstrate that it is a very good model to serve as the base model—a reference point against which we can compare the performance of other models. Comparing the results of the base model to those of the validation and test sets helps us identify potential overfitting and underfitting issues in our model.

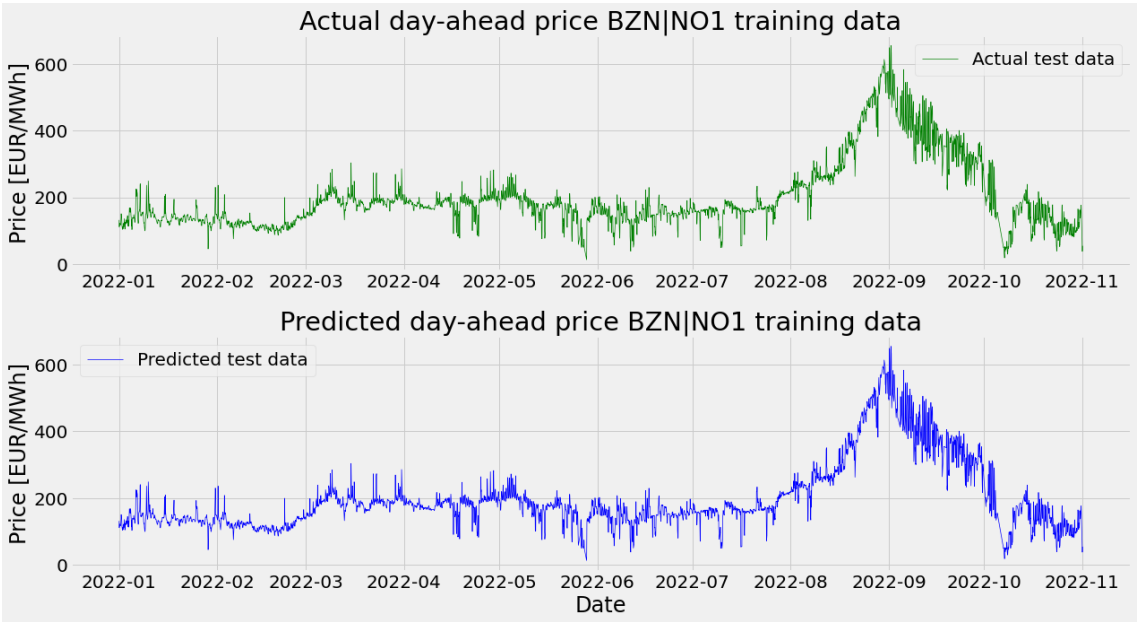

Figure 8.5: Actual and predicted day-ahead price NO1 training dataset 1-hour lagged features.

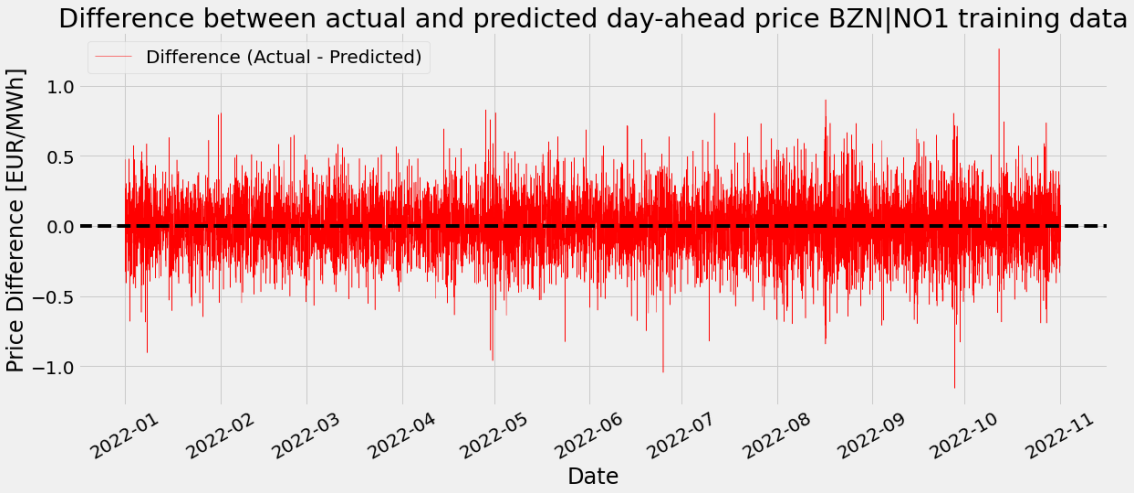

Figure 8.6: Difference between actual and predicted day-ahead price NO1 using training data.

Evaluation of predictions on the validation data

After evaluating the predictions on the training data, we proceed to assess the model's predictions on the validation set. This evaluation serves as an unbiased assessment of our trained base model and is crucial for hyperparameter tuning and avoiding overfitting during training.

By evaluating the model on the validation data, we obtain a more reliable estimate of its generalization performance on unseen data. We experiment with different hyperparameters and use the validation data to predict and choose the best hyperparameters before applying the model to the test dataset. To prevent overfitting, we closely monitor the performance of the base model on the validation data throughout training and halting the training process when the validation performance starts to deteriorate. This achievement is accomplished by specifying "eval.set = [(X-valid, y-valid)]" and setting "verbose=True" when fitting the model with the training data during the time series cross-validation split into 5 folds.

Figure 8.7 illustrates the actual "Day-ahead price NO1" in the blue plot and the predicted "Day-ahead price NO1" in the red plot. These plots indicate very accurate predictions on the validation data, with the worst predicted days occurring from 2022-11-11 to 2022-11-13.

Table 8.6 presents the performance results for this model during the "Base model validation" phase. In comparison to the base model, all four evaluation metrics exhibit significantly higher values, indicating substantial deviations between the predicted and actual values. The MAPE value of 311% reflects a high level of inaccuracy and suggests that further improvements are required through hyperparameter tuning.

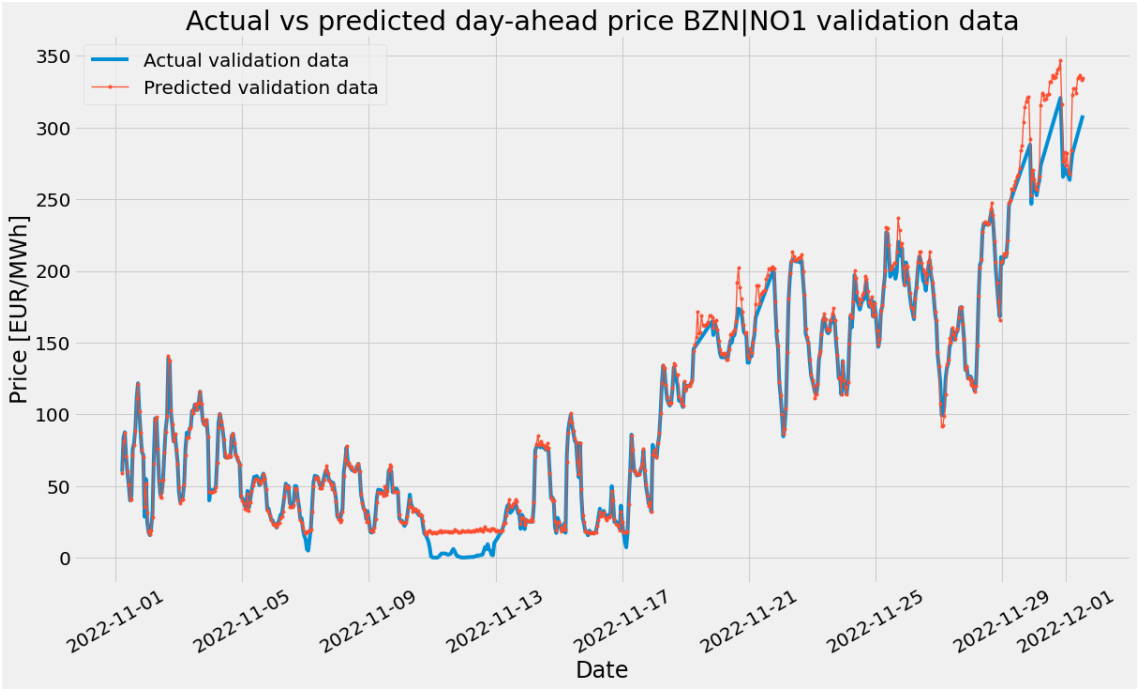

Figure 8.7: Actual and predicted day-ahead price NO1 using the validation dataset.

8.5.2 Tuned XGBoost model results: 1-lagged features

The tuned model represents the optimal model obtained through a random search that identified the best hyperparameters. These hyperparameters are the same for both the 1-hour lagged data and the 24-hour lagged data. Further details regarding the best hyperparameters can be found in Subsection 8.5.4.

Evaluation of predictions on the train and validation sets

Figure 8.10 displays the actual "Day-ahead price NO1" in the green plot and the predicted value in the blue plot when predicting the combined train and validation sets. These plots demonstrate highly accurate predictions on the validation data. Training the tuned model on the combined datasets allows the model to capture trends over a longer time period, resulting in improved generalization and performance.

Figure 8.11 illustrates the differences between the actual and predicted values for the "Day-ahead price NO1" feature in the combined dataset. The differences are negligible, as indicated on the y-axis labeled "Price Difference [EUR/MVh]".

Table 8.6 provides the performance results for this tuned model during the "Tuned model training" phase. Compared to the "Base model training," the performance of the "Tuned model training" is on average 10% better. Notably, the "Tuned model training" achieves an impressively low MAPE value of 0.08%, indicating a very high level of accuracy. These results are not entirely unexpected since the model undergoes a second training phase on the already-seen data (train set and validation set).

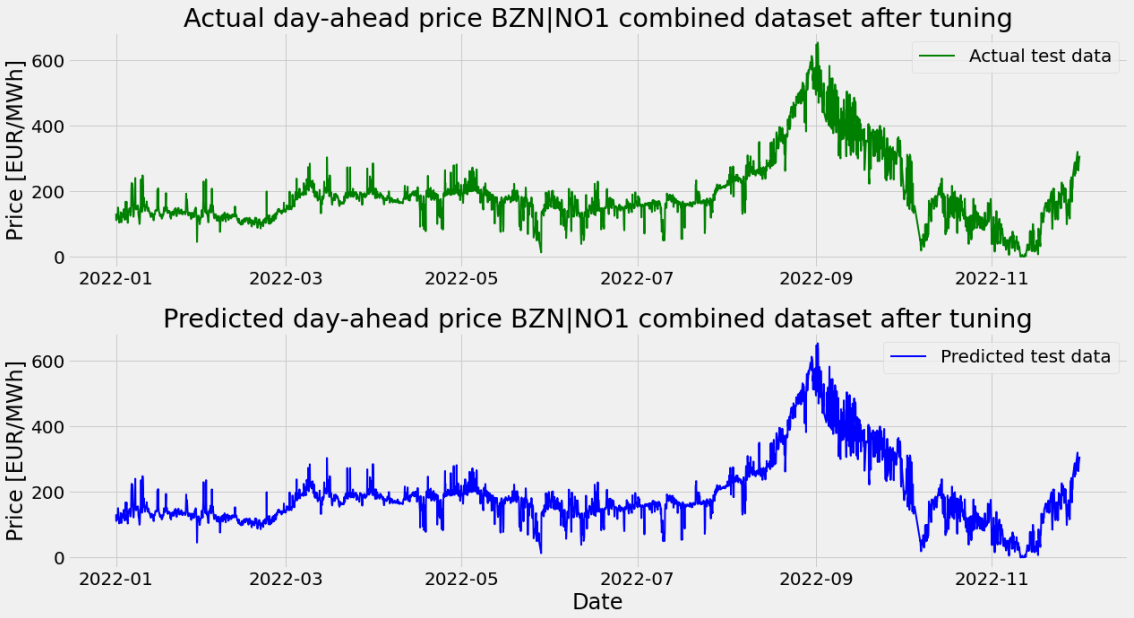

Figure 8.8: Actual and predicted day-ahead price NO1 validation dataset 1-hour lagged features.

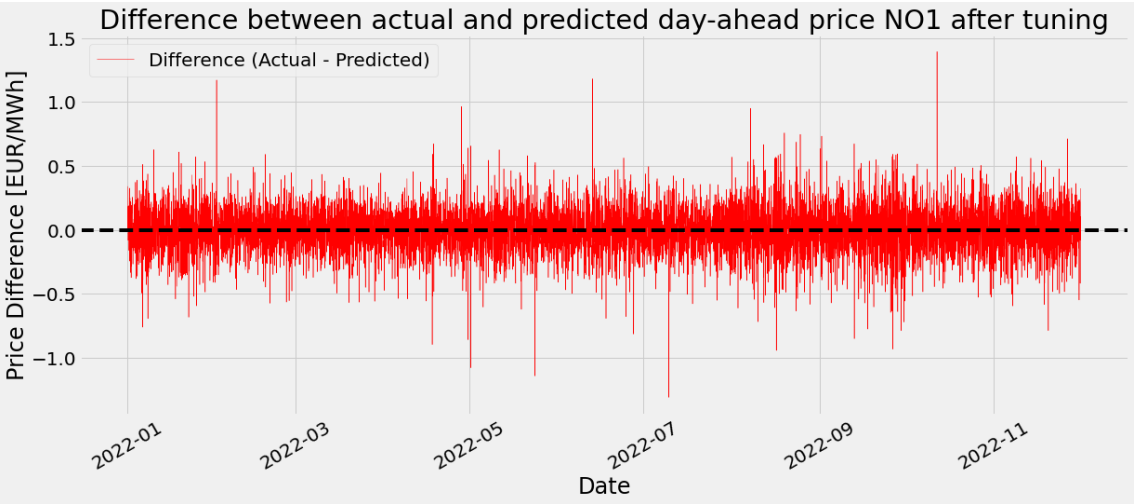

Figure 8.9: Difference between actual and predicted day-ahead price NO1 using combined data.

Evaluation of predictions on the test data

Figure 8.10 showcases the actual "Day-ahead price NO1" in the blue plot and the predicted "Day-ahead price NO1" in the red plot. The test data consists of 730 hours, predominantly in December 2022. The plots reveal highly accurate predictions on the test data, with the worst predicted days occurring on 2022-12-29 and 2022-12-30.

Table 8.6 presents the performance results for the tuned model, demonstrating its overall superiority to the base model. The tuned model achieves lower values for RMSE, MSE, and MAPE, indicating enhanced accuracy and performance. However, it is worth noting that the base model has a slightly lower MAE. Considering all evaluation metrics, the tuned model exhibits an improvement over the base model in terms of predictive accuracy.

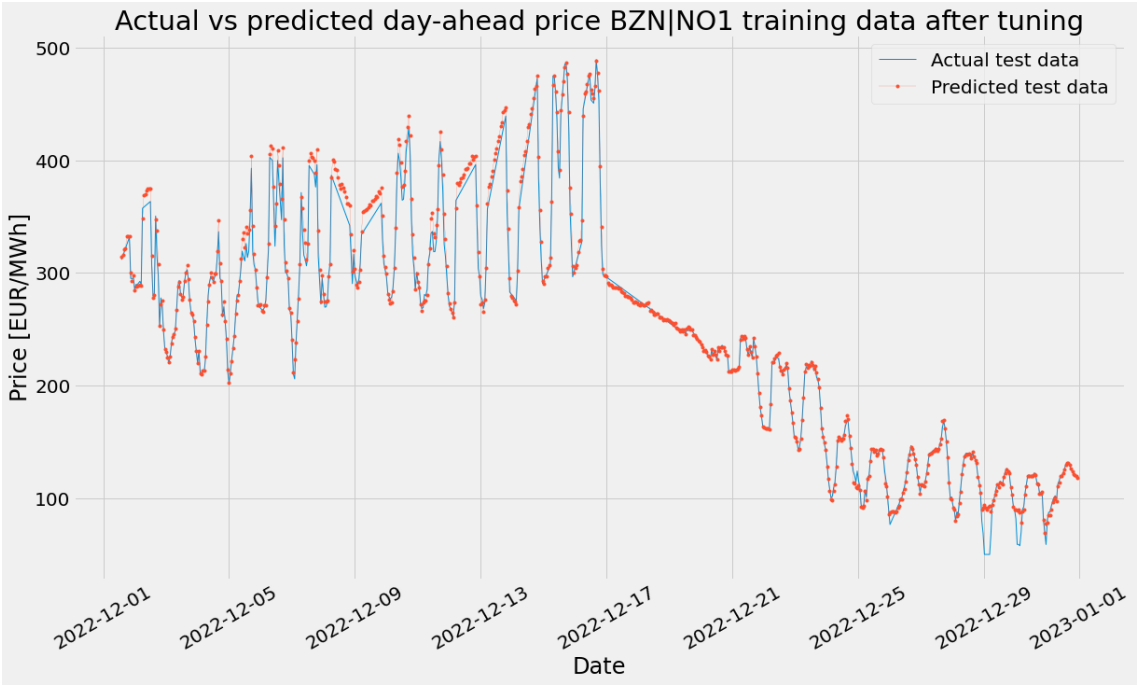

Figure 8.10: Actual day-ahead price and predicted day-ahead price NO1 for the test dataset.

Data	RMSE (1-hour)	MAE (1-hour)	MSE (1-hour)	MAPE (1-hour)
Base Model Training	0.21	0.16	0.04	0.10%
Base Model Validation	9.13	4.7	83.4	311%
Tuned Model Training	0.18	0.14	0.03	0.32%
Tuned Model Testing	8.19	4.9	67.1	2.5%

Table 8.6: Model's Performance for 1-hour lagged dataset.

Run time

Table 8.7 provides the prediction time for each model. An interesting observation is that the "Tuned Model Training" exhibits a shorter prediction time compared to the "Base Model Training." This is noteworthy considering that the tuned model utilizes more data. It suggests that our model does not require extensive training on already trained data (Train set) to achieve improved performance.

Model	Run time (seconds)
Base Model Training	0.06
Base Model Validation	0.01
Tuned Model Training	0.02
Tuned Model Testing	0.01

Table 8.7: Model's execution time on 1-hour lagged dataset.

8.5.3 Baseline XGBoost model results: 24-lagged features

The base model is the first model built using default XGBoost regressor hyperparameters. These hyperparameters are explained in Subsection 7.4.2.

Evaluation of predictions on the train data

As explained in Subsection 8.5.1, for models using 1-hour lagged features, we follow a process of splitting the data and training the model on the train data. Subsequently, we evaluate the predictions on the train data. Figure 8.11 presents the actual "Day-ahead price NO1" in the green plot and the predicted value in the blue plot. Figure 8.12 demonstrates that the differences between the actual and predicted values are extremely small, as depicted on the y-axis labeled "Price Difference [EUR/MVh].

Table 8.8 provides the performance results for this model during the "Base model training" phase. The model exhibits low values for RMSE, MAE, MSE, and MAPE, indicating high accuracy and making it a very good candidate to be used as the base model.

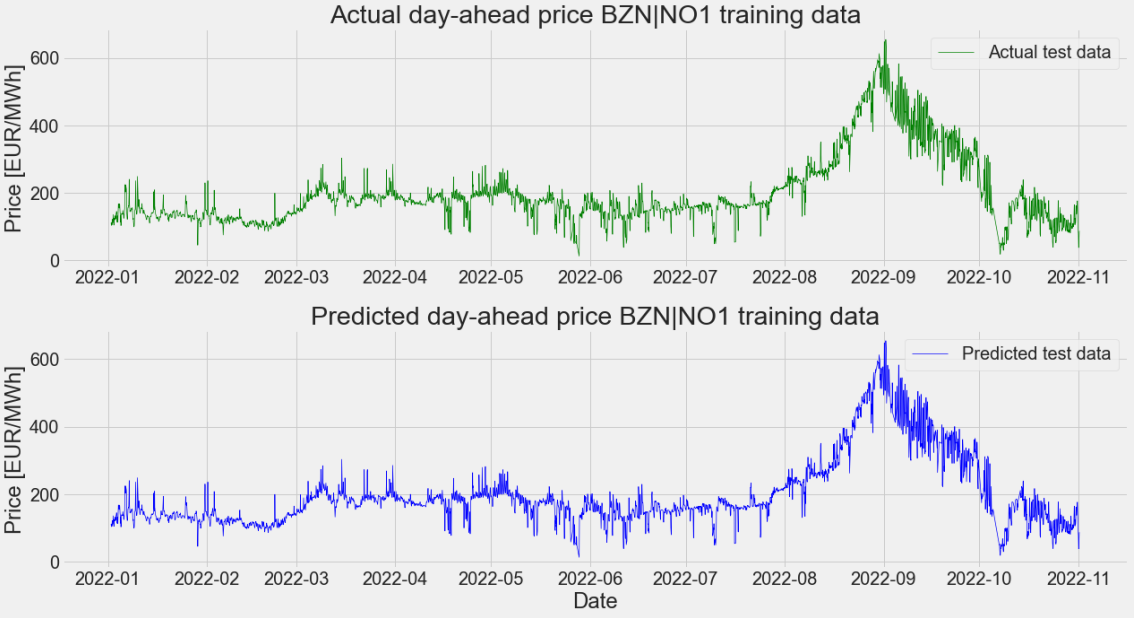

Figure 8.11: Actual and predicted day-ahead price NO1 training dataset 24-hour lagged features.

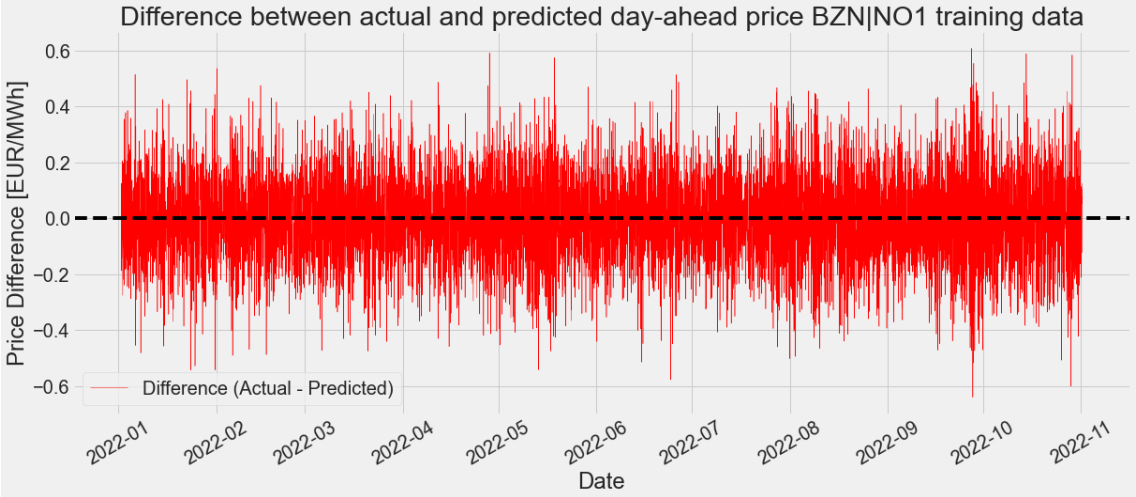

Figure 8.12: Difference between actual and predicted day-ahead price NO1 using training data.

Evaluation of predictions on the validation data

After evaluating the predictions on the train data, we proceed to evaluate the model's predictions on the validation set. The validation set serves as an unbiased evaluation of our trained base model and is crucial for hyperparameter tuning and avoiding overfitting during training, as discussed in Subsection 8.5.1, specifically in "Evaluating predictions on the validation data".

Figure 8.13 illustrates the actual "Day-ahead price NO1" in the blue plot and the predicted "Day-ahead price NO1" in the red plot. In comparison to the 1-hour lagged prediction on the validation model (Figure 8.7), the 24-hour lagged model exhibits better performance when predicting the validation data. This improvement can be attributed to having access to more historical data, which enhances the model's ability to capture trends. We can further validate these findings by referring to Table 8.4, where we observe that the RMSE and MSE evaluation metrics are 20% better, while the MAE and MAPE metrics are slightly worse. Overall, we can conclude that the base model's validation predictions yield better scores when our direct one-step XGBoost forecasting model is trained on a 24-hour lagged dataset.

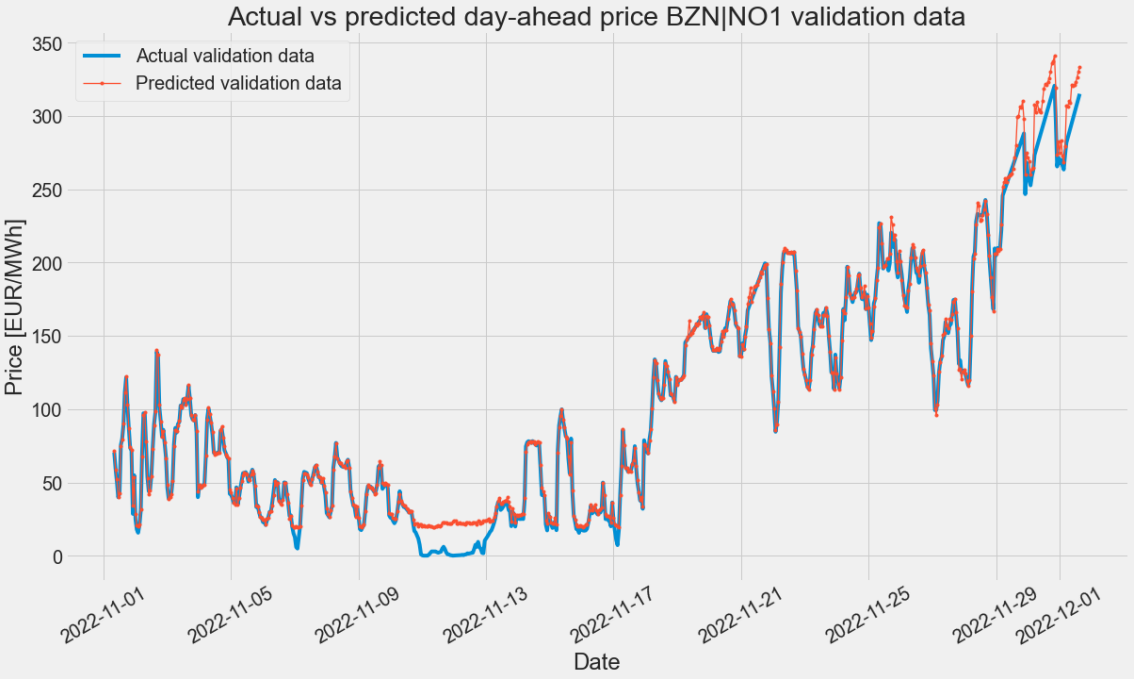

Figure 8.13: Actual and predicted day-ahead price NO1 before tuning, using the validation dataset.

8.5.4 Tuned XGBoost model results: 24-lagged features

The tuned model is the optimal model built by performing a random search and finding the best hyperparameters these parameters are exactly the same for both 1-hour lagged data and 24-hour lagged data. The best hyperparameters are explained in Subsection 8.5.4.

Evaluation of predictions on the train and validation set

Figure 8.14 displays the actual "Day-ahead price NO1" in the green plot and the predicted value in the blue plot when predicting the combined train and validation sets. Additionally, Figure 8.15 represents the difference between the actual and predicted values for the "Day-ahead price NO1" feature in the combined dataset. The y-axis labeled "Price Difference [EUR/MVh]" demonstrates that the differences between the actual and predicted values are insignificant. These plots indicate highly accurate predictions on the combined dataset, suggesting a good model fit.

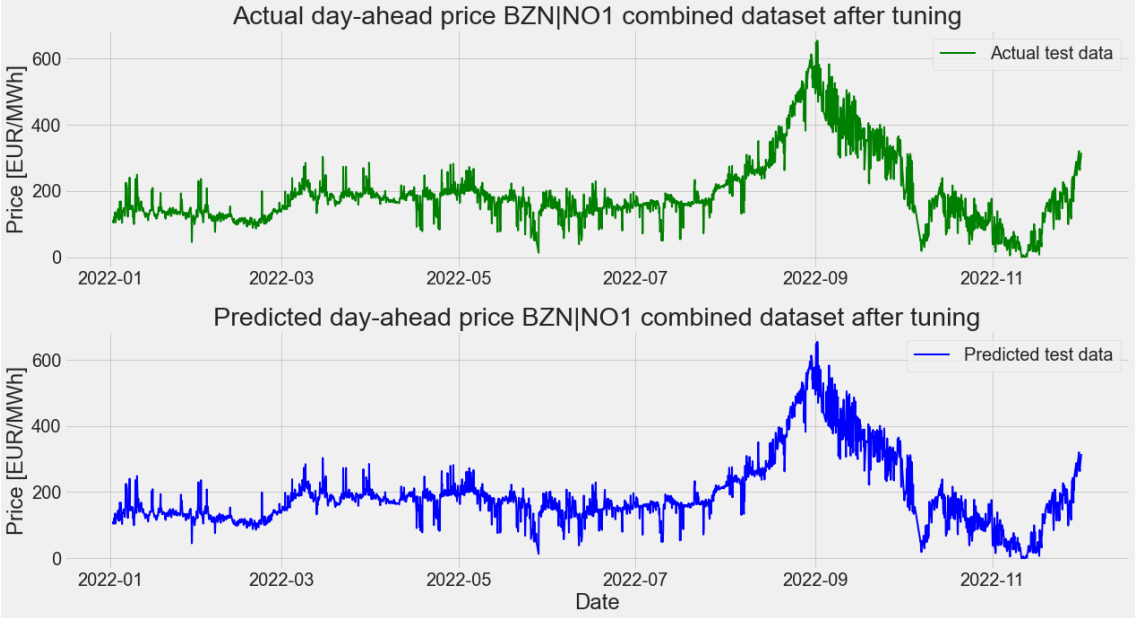

Figure 8.14: Actual and predicted day-ahead price NO1 validation dataset 24-hour lagged features.

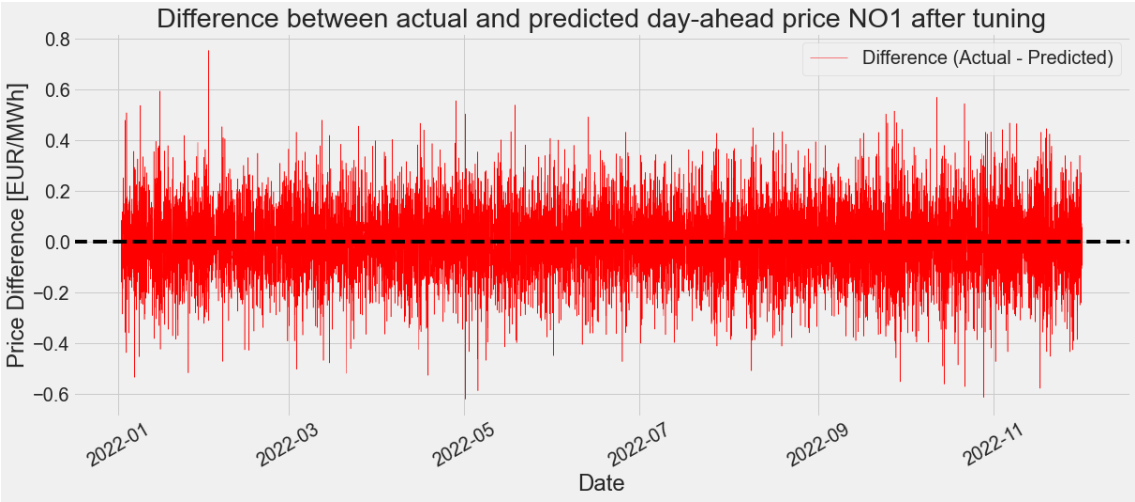

Figure 8.15: Difference between actual and predicted day-ahead price NO1 using combined data.

Evaluation of predictions on the test data

Figure 8.16 illustrates the actual "Day-ahead price NO1" in the blue plot and the predicted "Day-ahead price NO1" in the red plot. By analyzing Table 8.8 and Figure 8.16, we can make a comparison between the error and accuracy of both the "Base Model Validation" and the "Tuned Model Testing".

In terms of accuracy, the "Base Model Validation" exhibits higher error values (RMSE, MAE, MSE) and a significantly higher Mean Absolute Percentage Error (MAPE) of 371% compared to

the "Tuned Model Testing." Conversely, the "Tuned Model Testing" generally demonstrates lower error values and a lower MAPE of 2.6% in comparison to the "Base Model Validation." Therefore, the "Tuned Model Testing" showcases slightly better accuracy than the "Base Model Validation".

Regarding the error comparison, both models display a discrepancy between the predicted and actual values. The "Base Model Validation" yields higher error values with an RMSE of 7.46, MAE of 3.9, and MSE of 55.7, while the "Tuned Model Testing" exhibits an RMSE of 7.6, MAE of 4.8, and MSE of 57.7. Additionally, the "Base Model Validation" has a significantly higher MAPE of 371% compared to the "Tuned Model Testing" with a MAPE of 2.6%. Overall, the "Tuned Model Testing" demonstrates slightly lower error values and a smaller MAPE when compared to the "Base Model Validation.

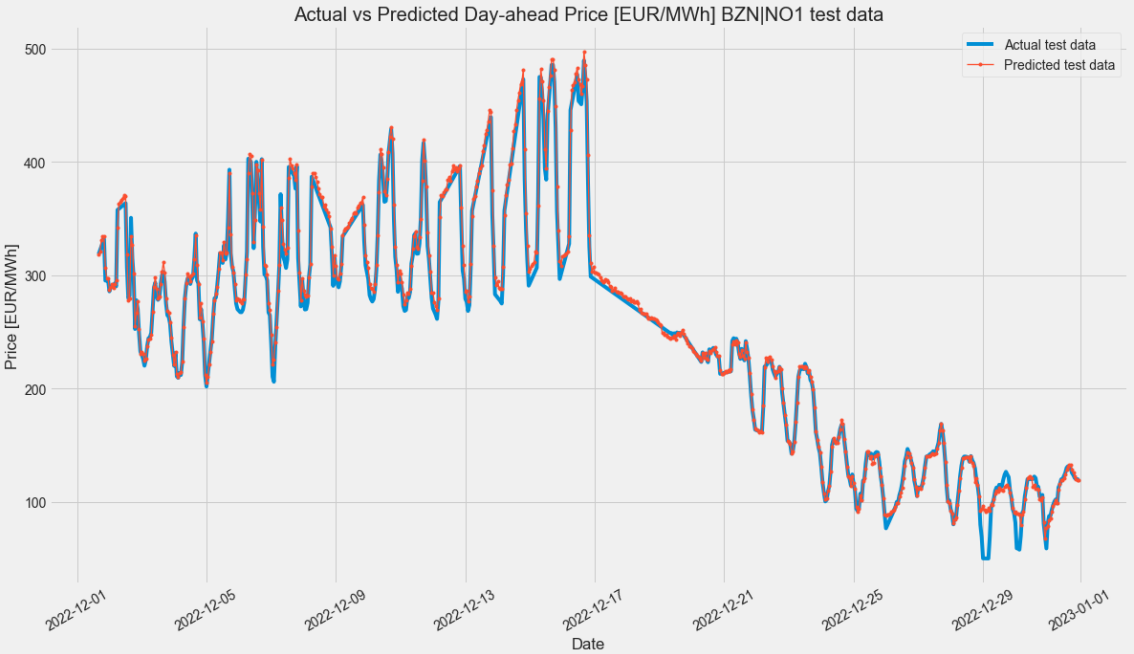

Figure 8.16: Actual and predicted day-ahead price NO1 after tuning, using the test dataset.

Data	RMSE (24-hour)	MAE (24-hour)	MSE (24-hour)	MAPE (24-hour)
Base Model Training	0.16	0.13	0.027	0.079%
Base Model Validation	7.46	3.9	55.7	371%
Tuned Model Training	0.15	0.11	0.02	0.17%
Tuned Model Testing	7.6	4.8	57.7	2.6%

Table 8.8: Model's Performance for the 24-hour lagged dataset.

Run time

Table 8.9 presents the prediction times for each 24-hour lagged model. The execution times are quite impressive, considering our dataset comprises 900 features and over 8500 instances. The XGBoost algorithm demonstrates one of its strengths by achieving low execution times, which is beneficial for efficient and fast predictions.

Model	Run time (seconds)
Base Model Training	0.18
Base Model Validation	0.03
Tuned Model Training	0.15
Tuned Model Testing	0.03

Table 8.9: Model's execution time on the 24-hour lagged dataset.

8.5.5 Hyperparameter tuning results

In this section, we will evaluate the hyperparameter tuning performed for both the 1-hour lagged dataset and the 24-hour lagged dataset, by implementing the random search technique. The random search was conducted with 10 iterations and 5-fold cross-validation, ensuring robustness and reducing the risk of overfitting. The performance of the tuned model was evaluated using the negative mean squared error metric, as explained in Section 7.4.3 XGBoost hyperparameter tuning.

The hyperparameter tuning process for the 1-hour lagged dataset took 2 minutes, while for the 24-hour lagged dataset, it took 20 minutes (Table 8.10). It is worth noting that the optimal model obtained from the tuning process has a reduced number of "n-estimators" compared to the base model, from 1,000 to 700. However, there were no other significant changes, and the results (Tables 8.7, 8.9) indicate slightly better performance after conducting hyperparameter optimization.

- Best hyperparameters are:
  - `subsample: 0.8`
  - `reg-lambda: 0.1`
  - `reg-gamma: 0.4`
  - `reg-alpha: 0.5`
  - `n-estimators: 700`
  - `min-child-weight: 4`
  - `max-depth: 6`
  - `learning-rate: 0.1`
  - `gamma: 0.1`
  - `colsample-bytree: 0.8`

Dataset	Run time (minutes)
1-hour lagged dataset	2
24-hour lagged dataset	20

Table 8.10: Random search hyperparameter tuning time.

8.5.6 RMSE results and interpretations

In this part we explain the interpretation of RMSE results from both the 1-hour lagged data and the 24-hour lagged data. We can evaluate the performance of the regression-based model by comparing its RMSE scores with the RMSE score of the baseline model. If the RMSE score of the model is significantly lower than that of the baseline, it indicates that the model is performing better.

Another approach to assess the model's performance is by comparing the RMSE value to a specific threshold. This threshold is determined by taking 10% of the response range, which represents the span of values in both the predicted values and the target variable (actual values). It provides an understanding of the range within which the model aims to predict. Figure 8.17 displays the implementation of this technique.

If the RMSE value is smaller than 10% of the response range, it suggests that the model's predictions align well with the data. On the other hand, if the RMSE value is equal to or greater than 10% of the response range, it indicates that the model's predictions significantly differ from the actual values.

We applied this method to evaluate all the RMSE results of our one-step direct forecasting models based on the XGBoost regressor. We can conclude that all the RMSE values are smaller than 10% of their corresponding response range values, indicating that the models perform well within the expected range.

```
response_range = max(predicted_values) - min(target_variable)

if rmse < 0.1 * response_range:
    print("The RMSE value of", rmse, "indicates a good fit for this regression model.
          The response range is:", response_range)
else:
    print("The RMSE value of", rmse, "indicates poor performance for this regression model.
          The response range is:", response_range)
```

Figure 8.17: Response range code snippet.

8.5.7 Comparison between 1-hour and 24-hour lagged models

To compare the performance of the 1-Hour and 24-hour lagged models, we can analyze Table 8.6 and Table 8.8. The "Tuned Model Testing" using the 24-hour lagged dataset demonstrates improved performance in terms of lower RMSE, MAE, MSE, and MAPE compared to the 1-hour lagged dataset. This indicates enhanced accuracy, precision, and overall predictive capability when utilizing the 24-hour lagged dataset.

The improved performance of the tuned model with the 24-hour lagged dataset can be attributed to several factors. Firstly, the 24-hour lagged dataset provides a broader temporal context, capturing comprehensive patterns and dependencies that span multiple hours. This allows the model to better understand the relationships between input features and the target variable.

Additionally, the 24-hour lagged dataset includes relevant historical information that impacts the target variable's behavior over a day, enabling the model to make more accurate predictions.

The dataset may also contain stronger signals and less noise compared to the 1-hour lagged dataset, as it aggregates data over a longer time span, reducing the impact of random fluctuations and short-term variations.

The improved performance emphasizes the importance of selecting an appropriate lagged dataset, considering the temporal context, and incorporating sufficient historical information to accurately capture long-term dependencies and trends.

8.6 Linear Regression Results

8.6.1 Ordinary least squares results

OLS has no hyperparameters to tune and therefore is tested on all variations of the dataset and its selected features. The training time is quite short, taking less than 2 seconds on the full dataset, and it converges even faster with fewer data. The algorithm complexity of OLS is $O(n_{samples}n^2_{features})$. From the results seen in the table below (8.11), an interpretation can be made that there seems to be less overfitting as the number of features shrinks. While the models trained on features from Lasso and ElasticNet have fairly similar testing results, the training results of the OLS model trained on features selected by Lasso resemble the actual testing results more closely than the other two models. This is especially visible when looking at the training and test results from the model trained on all data, where the training results are the best while the testing results are the worst.

Ordinary Least Squares results			
Evaluation metrics	All data	Features selected by Lasso	Features selected by ElasticNet
Training			
MAE	1.92998	1.67458	1.97711
MAPE	7.66515 %	1.48488 %	8.82282 %
MSE	11.76846	32.83399	14.29164
RMSE	3.43051	5.73009	3.78042
Testing			
MAE	3.88777	2.45351	3.58953
MAPE	1.91949 %	1.63258 %	1.91038 %
MSE	30.33589	29.30303	27.69405
RMSE	5.50780	5.41322	5.26251

Table 8.11: Ordinary least squares results of models trained on the different combination of features.

Figure 8.18 shows the actual price plotted against the mean of predictions done by the OLS model trained on the full dataset. It can be seen from the figure that the predictions have a quite good fit with the unseen data.

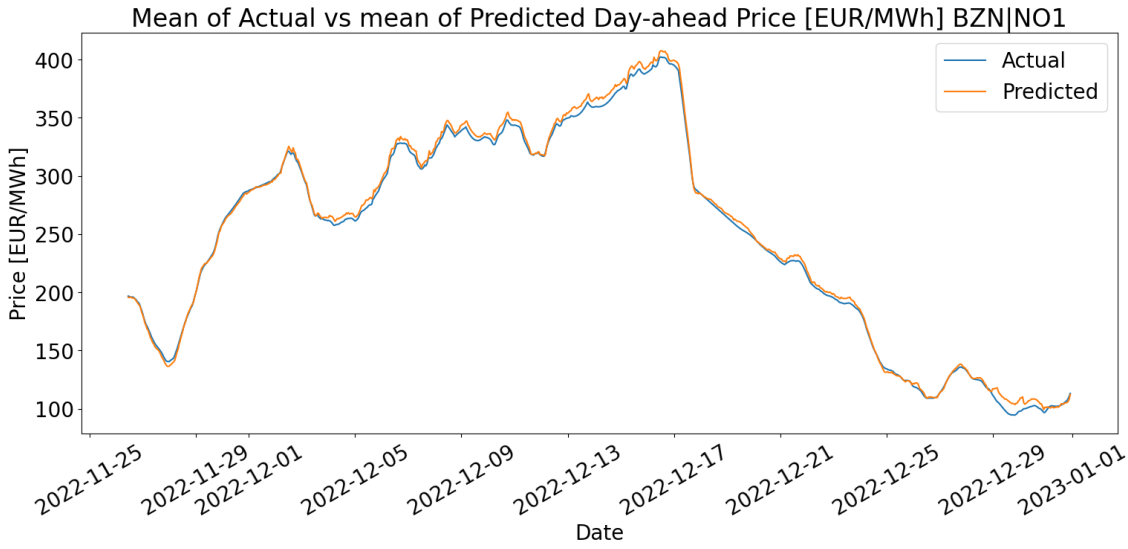

Figure 8.18: Graph of baseline OLS actual data vs mean of predicted.

8.6.2 Ridge regression results

Ridge Regression results			
Evaluation metrics	Untuned, on all data	Tuned, on all data	Tuned, on features selected by Lasso
Training			
MAE	1.93064	1.92898	1.67458
MAPE	7.68534 %	7.66967 %	1.48487 %
MSE	11.83975	11.78062	32.83399
RMSE	3.44089	3.43229	5.73009
Testing			
MAE	3.81943	3.87065	2.45345
MAPE	1.88955 %	1.91209 %	1.63256 %
MSE	29.65322	30.16075	29.30287
RMSE	5.44547	5.49188	5.41321

Table 8.12: Results of Ridge regression by the different models.

Ridge, with its L2 norm, generalizes better than OLS on the full data as seen in Table 8.12. However, the improvement is not significant, and it still shows signs of overfitting the training data. When considering the features selected by Lasso, there is no difference compared to OLS. Ridge also has a very similar training time of less than two seconds for the full training dataset, with the same algorithm complexity of $O(n_{samples}n^2_{features})$. Figure 8.21 displays the actual price of the unseen data against the mean price predicted by the tuned ridge model trained on features selected by Lasso at each time step.

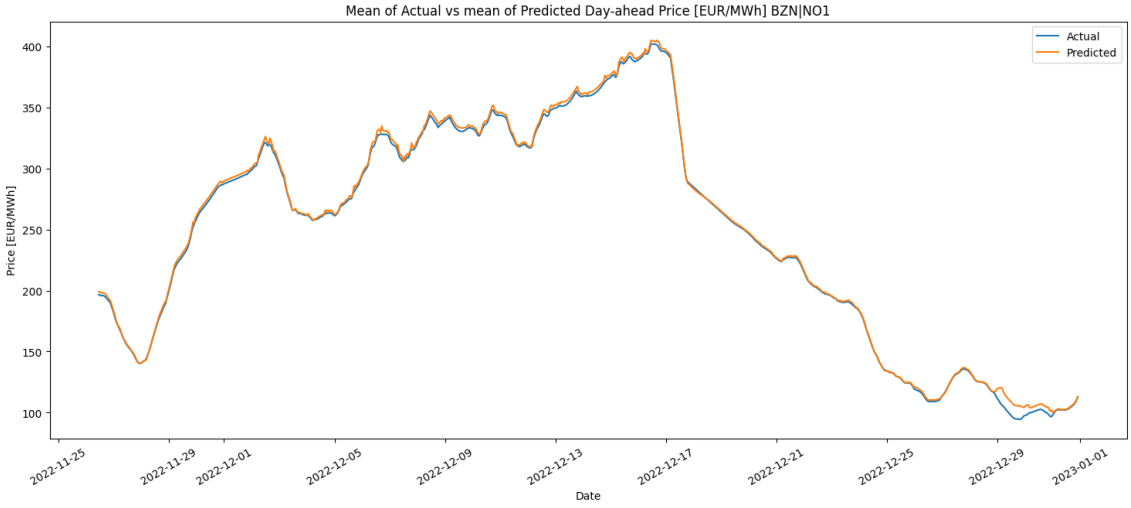

Figure 8.19: Graph of tuned Ridge model trained on features selected by Lasso.

8.6.3 Lasso regression resuts

Lasso Regression Results			
Evaluation metrics	Untuned, on all data	Tuned, on all data	Tuned, on features selected by Lasso
Training			
MAE	1.79584	1.87013	1.67085
MAPE	4.21441 %	5.56927 %	1.48071 %
MSE	36.91302	16.07989	32.83670
RMSE	6.07560	4.00997	5.73033
Testing			
MAE	1.70010	2.69565	2.44726
MAPE	1.46350 %	1.53035 %	1.62967 %
MSE	29.78844	21.74546	26.28737
RMSE	5.45787	4.66320	5.41178

Table 8.13: Lasso results by the different models.

All Lasso models perform well, as can be seen in Table 8.13. The untuned model trained on all data shows good generalization, with all evaluation metrics indicating better performance on the testing data than on the training data. This could be a sign of underfitting. However, it still performs adequately. This is because when the model is untuned, α is set to one. But when tuned, it prefers a value of 0.001, indicating that it does not heavily utilize the regularization term.

The second model, which is tuned on all the data, exhibits the best performance on the test data, with the lowest MSE observed. It also generalizes decently, with one of the metrics showing better results on the test data than on the training data. However, there might be a slight indication

of overfitting.

The last model, which is tuned and trained on the features selected by itself, shows relatively poorer results. However, two of the metrics, MSE and RMSE, perform better on the testing data than on the training data, suggesting better generalization compared to the other LR models.

These models have a training time of at most 3 minutes, which is significantly longer than OLS and Ridge. This is due to their complexity, which is $O(n^3_{features} + n^2_{features} n_{samples})$.

8.6.4 ElasticNet regression results

ElaticNet Regression Results		
Evaluation Metrics	untuned, on all data	Tuned, on features selected by Elasticnet
Training		
MAE	1.95607	1.95607
MAPE	7.93242 %	8.25846 %
MSE	12.58637	14.57974
RMSE	3.54772	3.84834
Testing		
MAE	3.56461	3.48247
MAPE	1.81501 %	1.87468 %
MSE	27.31957	26.58498
RMSE	5.22681	5.15606

Table 8.14: Results of ElasticNet regression by the different models.

Both ElasticNet models are observed to be overfit compared to the other LR models, as shown in Table 8.14. Despite the presence of two regularization terms, MAE and MSE nearly doubled from the training data to the testing data. This behavior is intriguing because the best hyperparameter for "l1_ratio" was found to be 0.9, indicating a strong preference for the L1 regularization norm. However, the models perform closer to the Ridge models than the Lasso models.
In Figure 8.20, the graph illustrates the mean of the predicted values against the mean of the actual values of the test data. Comparing this graph to the others of the LR models, a resemblance can be observed. Initially, all models exhibit a good fit, but it deteriorates over time. Between the 29th and 30th of December, all models display a common anomaly where they predict considerably higher prices than the actual values, deviating from their overall fit.
Due to the combination of both regularization terms, ElasticNet is the most computationally complex LR model, taking up to 12 minutes to compute.

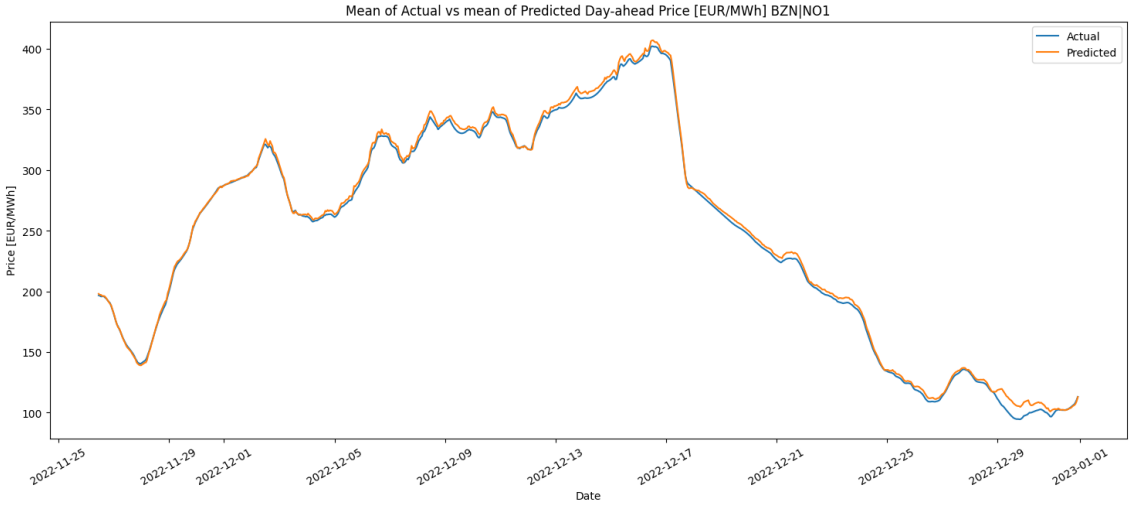

Figure 8.20: Graph showing the actual price against the predicted price of the model trained on features selected by ElasticNet.

8.7 Summary Performance Evaluation

We conclude the Evaluation chapter by addressing the issue of overfitting observed in most models. We analyze Table 8.15 and Figure 8.21, which provide visualizations of the best models based on random forest, support vector regression, extreme gradient boosting, and linear regression.
While our models exhibit excellent performance on the training set, they encounter difficulties when applied to the validation and test sets, except for the case of the LR models, which show mostly acceptable generalizations. This discrepancy indicates a clear case of overfitting. In simpler terms, the models have become too specialized in capturing the idiosyncrasies and intricacies of the training data, which hampers their effectiveness when encountering new, unfamiliar data.

Based on these findings, we draw the following final observations: The Lasso "tuned on all data" model stands out as the best performer among the evaluated models. With an MAE of 2.7, it achieves the smallest average prediction error compared to the actual values. This suggests that, on average, the predicted values are only off by 2.7 units. The low MAPE of 1.5% indicates a relatively small average percentage error in the predictions, implying a high level of accuracy. The MSE of 21.74 further supports its strong performance, as it represents the average squared difference between the predicted and actual values. Moreover, the RMSE of 4.66, derived from the square root of the MSE, provides an estimate of the average absolute error in the model's predictions. The low RMSE indicates that the Lasso "tuned on all data" model demonstrates accurate and precise predictions. Based on these metrics, we can conclude that this model has successfully captured the underlying patterns and relationships in the data.

The XGBoost 24-hour lagged model, while not as accurate as the Lasso model, still performs relatively well. It has an MAE of 7.6, indicating a slightly higher average prediction error compared to the top-performing model. However, considering the scale of the predicted values, an MAE of 7.6 may still be acceptable in certain applications. The MAPE of 2.6% suggests a reasonably small average percentage error, further supporting the model's accuracy. Although the MSE of

57.7 is higher compared to the Lasso model, it still represents a relatively low average squared difference between the predicted and actual values. The RMSE of 4.8, although slightly higher than the Lasso model, indicates a low average absolute error in the predictions. We can conclude that the XGBoost 24-hour lagged model captures important temporal patterns by considering the lagged values, contributing to its relatively accurate performance.

Model	MAE	MAPE	MSE	RMSE
Lasso tuned on all data	2.7	1.5%	21.74	4.66
XGBoost 24-hour lagged	4.8	2.6%	57.7	7.8
SVR	32.18	84.9%	1417	37.6
Random Forest	49.62	19%	4225.75	65.01

Table 8.15: Best performed models ranked from best to worst.

Moving on to the SVR model, we observe higher values across all metrics, indicating lower accuracy compared to the top-performing models. The MAE of 32.18 implies a larger average prediction error compared to both the Lasso and XGBoost models. The MAPE of 84.9% suggests a significantly higher average percentage error, which may be attributed to the model's limitations in capturing complex relationships in the data. The MSE of 1417 reflects a considerably higher average squared difference between the predicted and actual values, indicating a higher overall error. The RMSE of 37.6 confirms a larger average absolute error in the predictions. We can assume that the SVR model struggles to capture the underlying patterns and relationships accurately, possibly due to its linear nature or the lack of suitable kernel functions to capture the data's non-linearity.

Lastly, the Random Forest model exhibits the least accurate performance among the evaluated models. With an MAE of 49.62, it shows a significantly larger average prediction error compared to the other models. The MAPE of 19% indicates a higher average percentage error in the predictions, suggesting limitations in capturing the complexities of the data. The MSE of 4225.75 represents a substantial average squared difference between the predicted and actual values, indicating a larger spread of errors. Furthermore, the RMSE of 65.01 confirms a higher average absolute error in the predictions. Despite its lower accuracy, we can assume that the Random Forest model may still provide insights or be suitable for certain applications that do not require high precision.

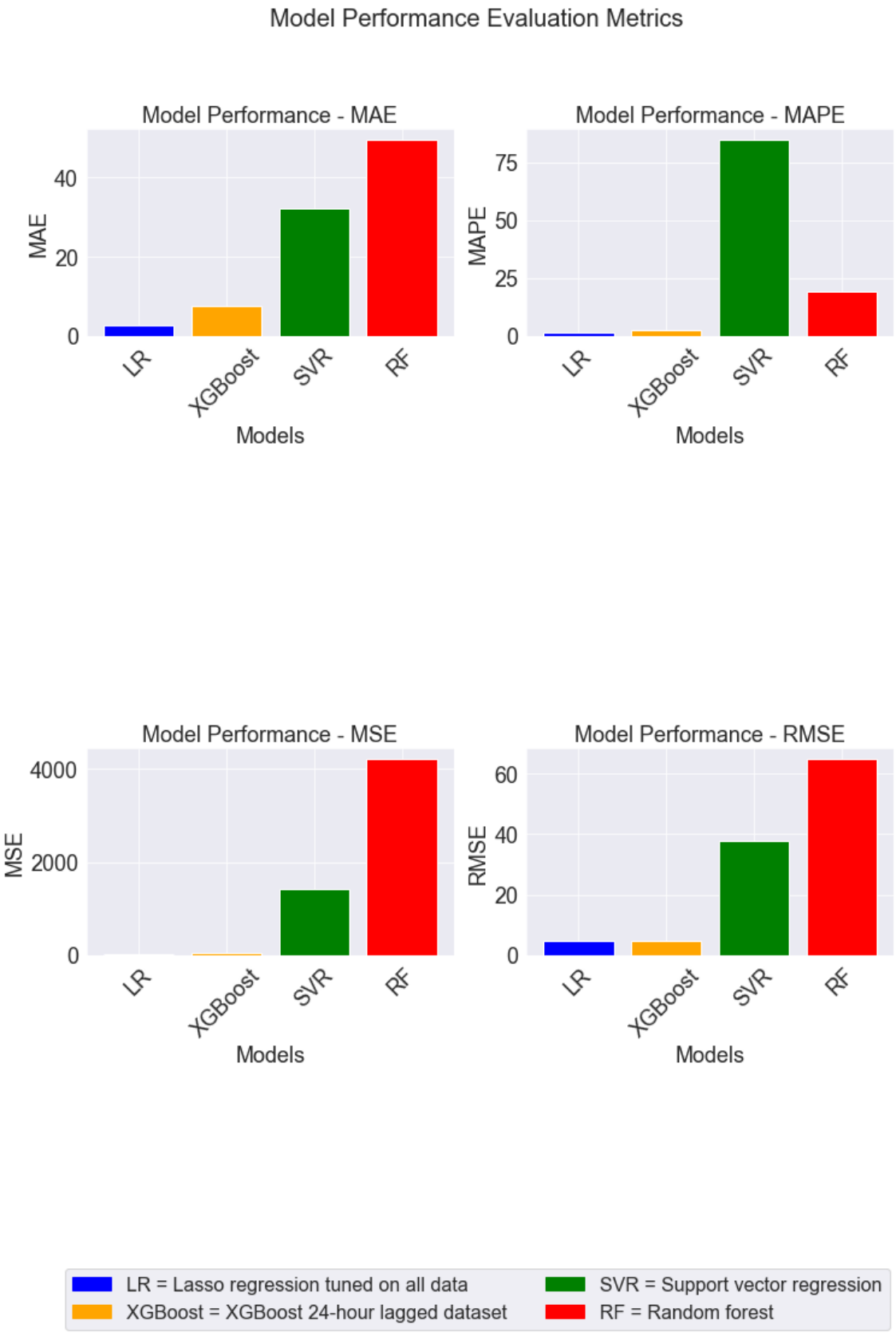

Figure 8.21: Comparing model's performances.

Chapter 9

Discussion and Challenges

9.1 Interpretation Of Results

The Lasso model, after being tuned on all available data, emerges as the top-performing model across all evaluation metrics. It exhibits the lowest values for MAE, MAPE, MSE, and RMSE, indicating superior predictive performance. Following closely behind is the XGBoost model, which also demonstrates strong overall performance with relatively low errors. In contrast, both the Random Forest and SVR models show higher errors and metric values compared to the top two models. The SVR model, in particular, exhibits lower accuracy, while the Random Forest model performs the least accurately among the evaluated models.

It's important to note that all models, except for "lasso with $\alpha = 1$", exhibit clear signs of overfitting when tested with unseen data. However, it's crucial to consider that this interpretation is based solely on the four evaluation metrics. To provide a comprehensive comparison, a deeper analysis of the internal mechanisms of each model would be necessary. Additionally, it's important to mention that there is currently no established benchmark model in the machine learning community that can be deemed both credible and reliable for this specific dataset.

Professional datasets typically undergo rigorous peer review and validation processes within the scientific or professional community, ensuring the dataset's integrity, credibility, and reliability. In contrast, our dataset lacks such formal validation processes, which should be taken into account when interpreting the results.

To improve the robustness of future analyses, it is recommended to consider additional validation techniques, such as cross-validation or external validation with independent datasets. These steps would help mitigate the risks of overfitting and enhance the credibility and reliability of the predictive models.

9.2 Comparison with Existing Literature

Our work shares a similar focus with the study conducted by Albahli et. al.[3], which utilized machine learning models such as XGBoost, Random Forest, and Support Vector Regression (SVR) for electricity price prediction [3]. They applied these models on daily spot electricity price data from Ontario and discovered that the XGBoost model achieved a high degree of accuracy.

However, our study diverges from their findings in significant ways. In our project, although the XGBoost model performed reasonably well, it was the Lasso tuned model that stood out as the best performer. This discrepancy could be attributed to differences in the datasets used, model tuning, validation strategies, or inherent differences in the electricity markets studied.

Another point of divergence is the issue of overfitting. While Albahli et al.[3] reported a high degree of accuracy on their dataset, our models, except for the Linear Regression ones, struggled with overfitting. This discrepancy indicates the complexities associated with model generalization, particularly in the context of electricity price prediction.

Despite these differences, our research complements Albahli et al.'s[3] work by emphasizing the importance of model validation and the persistent challenge of overfitting. Our study underscores the fact that while machine learning models can achieve high accuracy on training data, they may encounter difficulties when applied to new, unfamiliar data.

In line with Albahli et al.'s[3] suggestion of exploring neural networks for future work, our study also identifies the potential of deep learning models for electricity price prediction. Unfortunately, due to time constraints, we could not explore these models in the present study.

In conclusion, while our study shares the goal of predicting electricity prices more accurately with existing literature, it also offers its own unique insights. We discovered that the Lasso tuned model outperformed others in our experiments, and we noted the common challenge of overfitting. The constraints of our project meant we couldn't explore deep learning models, even though they show promise for future studies. Overall, our work contributes to ongoing efforts to tackle the complexity of electricity market prices.

9.3 Factors Affecting Model Performance

When one does forecasting there are several factors that may affect a models performance. Understanding these factors is important in helping to diagnose issues with a model

9.3.1 Quality and Quantity of Data

The quantity and quality of data used for training the model signifincantly affect the forecasting accuracy. More data usually improves the model's ability to understand the underlying patterns in the time series. Meanwhile clean and noise-free data can provide accurate trends and seasonality to the model. Due to the restricted quantity of data this has impacted the performance of the different models, in particular SVR which requires a high amount data to catch underlying trends.

9.3.2 Feature selection

The selection of relevant and meaningful features is crucial in machine learning. Including irrelevant features can introduce noise that confuses the model, while excluding important features can mean the model doesn't learn critical patterns in the data. Similarly, if there's multicollinearity (high correlation between predictor variables), it can be difficult for the model to determine the effect of each feature independently, which can negatively affect model performance and interpretability.

9.3.3 Seasonality and Trends

Presence of trends and seasonality in time series data can affect model performance. Some models handles trend and seasonality better than other. If a model is not particarly suited for that it may affect the performance.

9.3.4 Stationarity

A considerable number of models are predicated on the assumption of stationarity in a time series, meaning that the characteristics of the series are invariant concerning time. Breaching this assumption may result in suboptimal predictive outcomes. Methods like differencing or transformations may be employed to render the series stationary.

9.3.5 Model Hyperparameters

Hyperparameters control the learning process of the model. The choice of hyperparameters can have a significant impact on model performance. As an example, in the case of Support Vector Regression, hyperparameters like the kernel type, C (regularization parameter), gamma (kernel coefficient), and epsilon (margin of tolerance) can all significantly influence the model's ability to learn from the data and make accurate forecasts.

9.3.6 Hyperparameter Tuning

The predictive accuracy of the model can be greatly influenced by its hyperparameters. These include elements such as the learning rate involved in gradient descent, regularization parameters, and others.

9.3.7 Outliers

Outliers can significantly impact the performance of a time series forecasting model. Depending on the model, outliers can either be removed or the model can be made robust to handle these outliers.

9.3.8 Lag Selection

The choice of lag can have a significant effect on the model's performance, especially in autoregressive models. The autocorrelation function can be used to guide the selection of lags.

9.3.9 External factors

Models learn patterns in the training data, but they can't anticipate external factors that aren't represented in the data. For example, market fluctuations, policy changes, or other events can significantly affect energy prices, but if these factors aren't reflected in the data, the model won't be able to account for them, potentially leading to inaccurate predictions.

9.4 Generalizability and Scalability

In machine learning the ability of models to generalize and scale is crucial. The ability to generalize refers to the model's capacity to perform accurately on unseen data. This is particularly important in time series forecasting, where the ultimate goal is often to make future predictions based on past patterns. However, it's worth noting that generalization is heavily dependent on the assumption that the future will, to some extent, resemble the past. Therefore, if new unseen factors, such as significant changes in market behavior, technological advancements, or novel policy implementations, come into play, the model's accuracy might be compromised.

On the other hand, scalability refers to the model's capacity to handle an increase in workload, such as a larger dataset or more complex features. This is critical when applying the model to real-world scenarios where the data can be of enormous volume, velocity, and variety. The model should be able to handle this large-scale data efficiently without significantly degrading performance. However, certain models and techniques might be computationally expensive, limiting their scalability.

Furthermore, scalability is not only about handling larger datasets but also about adapting to more complex features or higher-dimensional data. Some models might require feature engineering or specific data preprocessing steps that can become increasingly challenging and time-consuming with more complex data. Therefore, considering scalability during model selection and design is crucial for long-term success.
Due to the nature of the dataset and it's limited size and low dimensionality makes the models not very applicable for real-wrold scenarios or larger-scale systems.

9.5 Unexpected Challenges

9.5.1 Lack of Dataset from Client

One of the most significant challenges faced during this project was the lack of a ready-to-use dataset from the client. Initially, the project plan was built around the assumption(due to contract) that the client would supply a complete and prepared dataset that could be directly fed into the analysis pipeline. The absence of such a dataset from the client was a substantial setback, which led to unforeseen alterations in the project execution.

Firstly, the task of data acquisition was unexpectedly thrust upon the project team. This process involves collecting, validating, and importing data from various sources. Depending on the nature of the data, this can be a time-consuming and complex process requiring specialized skills and resources. In some cases, there are also legal and privacy considerations when gathering data, further adding to the complexity of the task.

Following the acquisition, the data curation process became another major hurdle. This step involved the cleaning, transformation, and integration of the acquired data. Data curation is crucial to ensuring the accuracy, completeness, and consistency of the dataset, which in turn affects the reliability of the subsequent data analysis. Unfortunately, this process also turned out to be a resource-intensive and time-consuming task.

These unexpected steps of data acquisition and curation consumed about 1.5 months out of the total 5 months allocated for the project. This constituted a significant portion of the project timeline, leading to a considerable readjustment of the work schedule and project milestones.

Despite these hurdles, the experience provided the team with a valuable opportunity to gain a deep understanding of these crucial stages in the data science pipeline. It underscored the importance of proper communication and alignment of expectations between data scientists and clients, especially regarding data readiness, at the beginning of any data-driven project.

9.5.2 Discommunicaiton Between Group and Client

Two months into the project, as per instructions received from HiØ, the communication link with our client, Smart Innovation Norway, was suspended for the remainder of the project. This development, complied with in its entirety, introduced an extra layer of complexity to a project already filled with multifaceted challenges.

Maintaining continuous interaction with the client in data-focused projects of this scale is typically crucial. It often facilitates a deeper understanding of the problem space, ensures model assumptions align with practical realities, and tailors the data analysis process to meet client-specific objectives.

Moreover, the project was further complicated by the absence of a preprocessed dataset. In standard scenarios, the client's guidance could have been an invaluable asset in addressing these data-associated issues.

Given these constraints, the project's progression and execution were indisputably affected to an even higher level than it was previously.

9.5.3 Impact on Deep Learning Objectives

The unexpected necessity for data acquisition and curation, which consumed about 30% of the total project time, had a consequential impact on the planned deep learning objectives. Deep learning models are notably data-hungry and computationally intensive, requiring substantial time for model training, tuning, and evaluation.

Given the time consumed in acquiring and curating the data, the scope for developing and refining deep learning models was significantly reduced. This imposed constraint meant that there was insufficient time to explore a range of deep learning architectures, perform comprehensive hyperparameter tuning, or conduct exhaustive model validation and testing.

Furthermore, the limited time also restricted the ability to reiterate the model development process based on the results of initial analyses. Such iterative refinement is a crucial component of deep learning projects, often leading to substantial improvements in model performance.

In conclusion, the unexpected challenges associated with the dataset not only affected the immediate project timeline but also impinged upon the depth and breadth of the deep learning analysis that could be conducted.

9.6 Objectives

The unexpected requirement for data acquisition and curation resulted in significant repercussions on the project's sub-objectives, specifically those concerning the implementation and evaluation of machine learning. In addition this forces us to drop objectives pertaining to deep learning.

Sub-objective 1: Traditional ML Regression Models

The delay in starting data analysis due to data acquisition and curation left less time for implementing and evaluating various traditional machine learning regression models. The reduced timeframe meant a narrower range of models could be explored, limiting the scope and potentially the effectiveness of our model selection. Furthermore, the time constraints also limited the extent of hyperparameter tuning and validation tests that could be performed, which could impact the reliability of the insights gained about the best-suited models for predicting time series data in the Norwegian energy market (Bidding zone NO1 - Østlandet).

Sub-objective 2: Deep Learning Regression Models

The impact on the deep learning objectives, however, was severe. Given that deep learning models require substantial time for training, optimization, and validation, the unexpected diversion of resources towards data acquisition and curation essentially precluded the application of deep learning within the available project timeline. Therefore, we were unable to gain insights into which of our trained deep learning regression models would be best suited for predicting time series data in the Norwegian energy market (Bidding zone NO1 - Østlandet).

Sub-objective 3: Model Comparison

Due to the inability to implement deep learning models, the third sub-objective could not be achieved as initially planned. We intended to compare the performances of traditional ML models with deep learning models. However, given that deep learning was precluded due to time constraints, we could only carry out comparisons among the traditional machine learning models implemented.

In conclusion, the unexpected demands of data acquisition and curation led to significant deviations from the project's initial objectives. This experience underscores the importance of contingency planning and efficient time management in data science projects, especially when dealing with complex models like deep learning.

9.7 What could be done better

The process of this research project has highlighted several areas where enhancements could potentially have led to a more comprehensive and accurate set of findings.

9.7.1 Developing a Comprehensive Contingency Plan

While it is common practice to plan for unforeseen circumstances, this project underscored the need for a robust contingency plan, specifically centered around data-related issues. Provisions for potential scenarios such as data unavailability, poor data quality, or unexpected data preprocessing

needs should be an integral part of the project plan. This would ensure that project timelines and objectives remain realistic and achievable, even when confronted with unexpected challenges.

9.7.2 Enhancing Skill Sets for Efficient Data Processing

An area for team improvement could be enhancing our skills and proficiency in data acquisition and curation techniques. Advanced knowledge and usage of automated data cleaning and preprocessing tools could have saved time and allowed for more focus on model development and analysis. This is an area where continued learning and professional development can directly impact project efficiency.

9.8 Summary Discussion

This project tackled various challenges affecting machine learning model performance and achieving project objectives. Key influences on model performance included data quality and quantity, feature selection, seasonality, trends, stationarity, model hyperparameters, hyperparameter tuning, outliers, lag selection, and unrepresented external factors. The generalizability and scalability assessments revealed limitations due to the characteristics and size of the dataset.

Unexpected hurdles, notably in data acquisition and curation, significantly affected the project timeline, constraining the implementation of deep learning objectives. This led to a restricted exploration of traditional machine learning regression models, completely excluding the application of deep learning models. Thus, the comparison was confined to traditional machine learning models.

Compared to existing literature, specifically the study by Albahli et al[3]. that utilized machine learning models for electricity price prediction, our work shares similarities but also presents significant divergences. While XGBoost was the best performer in their study, our research found the Lasso tuned model to outshine the others. Moreover, our models, unlike Albahli et al.'s[3], grappled with overfitting, demonstrating the complexities of model generalization in electricity price prediction. Our work complements their research, underscoring the importance of model validation and the persisting overfitting challenge, and similarly recognizes the potential of deep learning models for future studies.

Reflecting on areas for improvement, there's a critical need for a robust contingency plan for unexpected data-related issues and enhancing the team's proficiency in data acquisition and curation techniques. By addressing these areas, more realistic project timelines can be ensured, and the focus can be redirected towards model development and analysis, overcoming some of the encountered limitations.

Chapter 10

Further Development

This chapter explores the possibilities and avenues for further development in the time series data prediction of electricity market prices using machine learning (ML) models. Building upon the insights gained from our research paper, we explore potential areas of improvement, propose novel methodologies, and suggest future directions for researchers and practitioners. By identifying key areas for advancement, we aim to contribute to the ongoing progress and enhance the accuracy and reliability of electricity market price predictions.

10.1 Expanding dataset time span for seasonal trend detection

Our research utilized a time series dataset with a relatively short period of only one year. Future studies should consider employing ML models on datasets spanning several years to provide more comprehensive insights into price fluctuations and capture larger seasonal trends. By encompassing multiple years data, these models would be better equipped to identify and predict yearly and monthly fluctuations in electricity prices. This broader temporal context would enable a more accurate representation of seasonal patterns and facilitate improved forecasting capabilities.

10.2 Investigating Deep Learning Algorithms for Handling Volatility

Although our research paper focused on ML algorithms for electricity price prediction, we did not incorporate deep learning methods into our analysis. Deep learning algorithms, such as recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, have successfully captured complex temporal dependencies in various domains [53]. Therefore, future research efforts could investigate the viability of deep learning algorithms in handling the highly volatile nature of electricity price data, particularly in bidding zone NO1. The volatile nature of electricity prices in bidding zone NO1 poses challenges for traditional ML algorithms to generalize and train models that effectively capture the underlying patterns. Deep learning models, with their ability to capture intricate temporal dependencies, offer a promising avenue for improving predictions in such volatile environments. By exploring deep learning algorithms, researchers can evaluate their potential to effectively model and predict electricity prices, potentially achieving more accurate and robust results.

10.3 Incorporating additional environmental features

While our dataset included several important features impacting the availability and pricing of electricity in bidding zone NO1, we acknowledge the potential for incorporating additional environmental features to enhance predictive performance. For instance, current weather forecasts, gas prices, or currency exchange rates can influence electricity prices. By including these features in the dataset, ML models can achieve better predictive performance by capturing the interplay between environmental factors and electricity market dynamics. Exploring the integration of such features and their impact on prediction accuracy can be a fruitful area for future research.

10.4 Accounting for market dynamics and events

Various market dynamics and events, such as changes in energy policies, regulatory interventions, or unexpected disruptions in supply or demand, influence electricity market prices. Future studies can explore incorporating market indicators and event data into ML models to capture the impact of these factors on electricity prices. The models can provide more nuanced and accurate predictions by integrating market-specific information.

10.5 Considering regional and local factors

Electricity markets can exhibit significant regional and local variations in price patterns. Future research can focus on developing ML models that account for these regional or local factors by incorporating location-specific data, such as transmission constraints, generation mix, or demand profiles. Predictions can be further improved by tailoring the models to specific regions or localities, leading to more accurate and localized insights.

10.6 Collaboration, Data Sharing, and Addressing Data Gaps

To advance the field of electricity price prediction using ML, collaboration and data sharing among researchers and industry stakeholders is essential. However, it is important to acknowledge that data acquisition can pose challenges, and under our data acquisition process, a large amount of data contains significant gaps of missing values. Future efforts should focus on establishing platforms and initiatives that facilitate data sharing and collaborative research and address data gaps and missing values. By pooling resources, knowledge, and expertise, researchers can collectively work towards developing innovative approaches to handle missing data and improve the accuracy and robustness of prediction models. This collaborative effort could enable a more comprehensive understanding of electricity market dynamics and contribute to developing practical solutions for addressing the complexities and uncertainties associated with missing data.

10.7 Summary Further Development

In this chapter, we have identified several key areas for further development in time series data prediction of electricity market prices using ML models. By expanding the dataset to include a

longer time span, exploring the potential of deep learning algorithms, and incorporating additional environmental features, researchers can advance the accuracy and applicability of electricity price predictions. Furthermore, conducting rigorous evaluations and comparisons of different modeling approaches will enable the identification of the most effective techniques for addressing the unique challenges of electricity market data. These efforts contribute to the ongoing progress in ML-based electricity price prediction and offer valuable insights for stakeholders in the energy sector, supporting informed decision-making and fostering the transition toward a more sustainable and efficient energy landscape.

Chapter 11

Conclusion

This thesis aimed to evaluate and compare different algorithms' performance in predicting the day-ahead (24-hour forecast) of electricity prices in bidding zone NO1 (Østlandet Norway). A dataset including energy market prices, time of price, and external factors for bidding zone NO1 and neighboring zones were aggregated, and the data was cleaned. Due to large amounts of missing data prior to 2022, the dataset was limited to the year 2022. Four regression ML algorithms were selected, and models were trained on the dataset. No deep learning models were trained due to time constraints caused by scope creep from having to acquire data, aggregate, and clean the dataset.

Of the implemented ML models, Lasso Regression closely followed by XGBoost demonstrated the best performance in predicting electricity prices, outperforming other models such as RF and SVR. The findings from the implementation of the ML models for predicting electricity prices in this study indicate a prevalent issue of overfitting across all models. This overfitting phenomenon, characterized by models performing well on the training data but failing to generalize effectively to unseen test data, can largely be attributed to the highly volatile nature of electricity prices observed during the period from which the training data was sourced. Furthermore, the relatively short length of the dataset, not spanning over several years and failing to capture yearly and monthly seasonality, may have also contributed to the overfitting phenomenon observed.

The findings of this study make a valuable contribution to the existing knowledge in the field of electricity price prediction. By comparing the performance of different ML algorithms in predicting day-ahead electricity prices, this research provides insights into their effectiveness and highlights the challenges posed by highly volatile energy markets. The identification of overfitting as a significant issue underscores the increasing importance of more powerful models like deep learning in addressing the challenges posed by the highly volatile nature of electricity prices, guiding future research efforts in developing more robust and adaptive models that can effectively capture trends and generalize from such data.

Deep learning algorithms have captured complex temporal dependencies in various domains previously [53]. Therefore, future research efforts should investigate the viability of deep learning algorithms in handling the highly volatile nature of electricity price data, particularly in bidding zone NO1.

Bibliography

[1] Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32:896–913, 2016.

[2] C. C. Aggarwal and C. C. Aggarwal. Applications of outlier analysis. *Outlier Analysis*, pages 373–400, 2013.

[3] S. Albahli, M. Shiraz, and N. Ayub. Electricity price forecasting for cloud computing using an enhanced machine learning model. *IEEE Access*, 8:200971–200981, Nov 2020.

[4] Z. Ali and S. B. Bhaskar. Basic statistical tools in research and data analysis. *Indian journal of anaesthesia*, 60(9):662–669, Sep 2016. PMID: 27729694.

[5] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, Cambridge, MA, 2014.

[6] T. Amr. *Hands-On Machine Learning with scikit-learn and Scientific Python Toolkits: A practical guide to implementing supervised and unsupervised machine learning algorithms in Python*. Packt Publishing, Limited, 2020.

[7] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[8] G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied artificial intelligence*, 17(5-6):519–533, 2003.

[9] G. D. Bella, M. Flanagan, K. Foda, S. Maslova, A. Pienkowski, M. Stuermer, and F. Toscani. Natural gas in europe: The potential impact of disruptions to supply. Retrieved January 13, 2023, from `https://www.imf.org/-/media/Files/Publications/WP/2022/English/wpiea2022145-print-pdf.ashx`, 2022.

[10] B. Bengfort, R. Bilbro, and T. Ojeda. *Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning*. O'Reilly Media, Inc., 2018.

[11] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[14] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[15] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[16] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[17] H.-Y. Cheng, P.-H. Kuo, Y. Shen, and C.-J. Huang. Deep convolutional neural network model for short-term electricity price forecasting. *arXiv preprint arXiv:2003.07202*, 2020.

[18] J. I. Daoud. Multicollinearity and regression analysis. *Journal of Physics: Conference Series*, 949(1):012009, dec 2017.

[19] J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.

[20] Directorate-General-Energy-EU. Quarterly report on european electricity markets. *Market Observatory Energy EU*, pages 10–25, 2021.

[21] ENTSO-E. dashboard.

[22] ENTSO-E. Entso-e mission statement.

[23] ENTSO-E. Total load - day ahead / actual.

[24] ENTSO-E. Water reservoirs and hydro storage plants, Year not provided. [Accessed: March 2, 2023].

[25] European-Commission. Electricity interconnection targets. `https://energy.ec.europa.eu/topics/infrastructure/electricity-interconnection-targets_en`, 2021. [Accessed: February 4, 2023].

[26] A. G'eron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[27] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[28] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[30] S. Guthals, P. Haack, and M. McCullough. *GitHub For Dummies*. For Dummies, 2020.

[31] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[32] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[33] A. G. Hyndman, Rob J. *Forecasting : principles and practice*. OTexts, 1 edition, 2014.

[34] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[35] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[36] G. Kunapuli. *Ensemble Methods for Machine Learning*. Manning, 2023.

[37] T. Lavanya, N. Malarvizhi. Risk analysis and management, 2020.

[38] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[39] C. McHugh, S. Coleman, and D. Kerr. Technical indicators and prediction for energy market forecasting. pages 1254–1259, 2020.

[40] W. McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 2017.

[41] J. Miao and L. Niu. A survey on feature selection. *Procedia Computer Science*, 91:919–926, 2016. Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016).

[42] P. Mitra, C. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, March 2002.

[43] D. A. Mittal, S. Liu, and G. Xu. Electricity price forecasting using convolution and lstm models. pages 1–4, 2020.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[45] M. Mori. Lattice isomorphisms between projection lattices of von neumann algebras. 8:e49, 2020.

[46] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[47] A. C. Müller and S. Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016.

[48] NIST/SEMATECH e-Handbook of Statistical Methods. Correlation coefficients. `https://www.itl.nist.gov/div898/handbook/eda/section3/corcoef.htm`, 2012.

[49] Nord Pool AS. White paper answering the energy crisis – a power market perspective. Retrieved January 13, 2023, from `https://www.nordpoolgroup.com/4a62f8/globalassets/download-center/whitepaper/nord-pool-white-paper-september-2022.pdf`, 2022.

[50] NumPy Contributors. NumPy: the absolute basics for beginners.

[51] R. Olu-Ajayi, H. Alaka, H. Owolabi, L. Akanbi, and S. Ganiyu. Data-driven tools for building energy consumption prediction: A review. *Energies*, 16(6), 2023.

[52] S. Orenc, E. Acar, and M. S. Özerdem. The electricity price prediction of victoria city based on various regression algorithms. pages 164–167, 2022.

[53] M. Pavicevic and T. Popovic. Forecasting day-ahead electricity metrics with artificial neural networks. *Sensors (Basel, Switzerland)*, 22(3):1051, 2022.

[54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[55] S. Raschka and V. Mirjalili. *Python Machine Learning*. Packt Publishing, 2017.

[56] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2010.

[57] M. Santos. Data quality issues that kill your machine learning models, 2021. Accessed: March 2, 2023. `https://towardsdatascience.com/data-quality-issues-that-kill-your-machine-learning-models-961591340b40`.

[58] A. Shiri, M. Afshar, A. Rahimi-Kian, and B. Maham. Electricity price forecasting using support vector machines by considering oil and natural gas price impacts. pages 1–5, 2015.

[59] Smart Innovation Norway AS. About smart innovation norway. Retrieved May 22, 2023, from `https://smartinnovationnorway.com/en/om-oss/`, 2023.

[60] Smart Innovation Norway AS. Our services. Retrieved May 22, 2023, from `https://smartinnovationnorway.com/en/vare-tjenester/smarte-byer-og-samfunn/vare-tjenester/`, 2023.

[61] Statnett. How the power system works. Retrieved January 14, 2023, from `https://www.statnett.no/en/about-statnett/get-to-know-statnett-better/how-the-power-system-works/#:~:text=Power%20is%20an%20extremely%20perishable,This%20is%20called%20instantaneous%20balance.`, November 2018.

[62] Stexnext. Risks in machine learning projects and how to avoid them, 2020.

[63] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[64] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding, 2020.

[65] Y. Tong, J. Liu, L. Yu, L. Zhang, L. Sun, W. Li, X. Ning, J. Xu, H. Qin, and Q. Cai. Technology investigation on time series classification and prediction. *PeerJ Computer Science*, 8:e982, 2022.

[66] D. Toomey. *Jupyter Cookbook: Over 75 recipes to help you overcome your difficulties with data analysis using Jupyter Notebook*. Packt Publishing, 2018.

[67] J. VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2016.

[68] B. Venkatesh and J. Anuradha. A review of feature selection and its methods. *Cybernetics and information technologies*, 19(1):3–26, 2019.

[69] K. Vu. How to create a dataset for machine learning. `https://www.kdnuggets.com/2022/02/create-dataset-machine-learning.html`, 2022. [Accessed: March 2, 2023].

[70] M. Walker. *Data Cleaning and Exploration with Machine Learning: Get to grips with machine learning techniques to achieve sparkling-clean data quickly*. Packt Publishing, 2022.

[71] R. Weron. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, 30(4):1030–1081, 2014.

[72] L. Winner. *Statistical Methods for Categorical Data Analysis*. Academic Press, 2012.

[73] XGBoostDevTeam. Python api reference. `https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.training`, 2021.

[74] XGBoostDevTeam. Xgboost documentation. `https://xgboost.readthedocs.io/_/downloads/en/release_1.4.0/pdf/`, 2021.

[75] XGBoostDevTeam. Xgboost parameters. `https://xgboost.readthedocs.io/en/latest/parameter.html`, 2021. [Accessed: April 20, 2023].

[76] A. Yousefi, O. A. Sianaki, and D. Sharafi. Long-term electricity price forecast using machine learning techniques. In *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, pages 2909–2913. IEEE, 2019.

[77] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe. Concept drift detection and adaptation with hierarchical hypothesis testing. *Journal of the Franklin Institute*, 356(5):3187–3215, 2019.

[78] B. W. Yueting Yu. Bidding and investment in wholesale electricity markets: Pay-as-bid versus uniform-price auctions. *13th Toulouse Conference on The Economics of Energy and Climate*, pages 2–10, 2022.

[79] C. Zhang, O. Vinyals, R. Munos, and S. Bengio. A study on overfitting in deep reinforcement learning, 2018. [Accessed: February 25, 2023].

[80] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[81] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.