ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Comparative Study of Traditional Statistical Models and Deep Learning for Load Forecasting: SARIMAX vs. NBEATSx in the Swiss Power Grid

Semester Thesis

M. Steilen

December 12, 2024

Supervisor: Dr. M. Kalisch

Seminar for Statistics, ETH Zürich

Acknowledgements

I would like to sincerely thank Dr. Markus Kalisch for his time and support during this semester project. His valuable feedback and genuine interest in enabling students' research interests are greatly appreciated and serve as a role model for academic growth and curiosity.

Code and Data Availability

Code Access The code and data necessary to reproduce the experiments conducted in this thesis are publicly available in a GitHub repository [1]. The repository contains all scripts and notebooks used for data preprocessing, model training, evaluation, and forecasting. Each script is documented with comments to clarify the functionality and flow of the code. Data sets used in this research are included in the repository, ensuring that all necessary components are accessible for reproducibility. Calculations were carried out using Python 3.10.15 and a `requirements.txt` file has been provided in the GitHub repository for installation of the respective package versions.

Reproducibility To promote reproducibility, the code follows best practices by providing a `requirements.txt` file that lists all required Python packages and their versions, and by setting random seeds in all relevant scripts.

[1] `https://github.com/MathiasSteilen/semesterthesis-load-forecasting-switzerland`

Abstract

Short-term consumption forecasting is essential for maintaining the balance of demand and supply in the power grid. As a financial incentive, Switzerland's transmission system operator uses bidirectional penalties to pass on costs arising from grid balancing with control energy to the responsible balance group managers. Given the sharp rise in balancing costs in recent years, effective forecasting models are more crucial than ever for improving grid stability and minimisation of balancing energy. This study investigates whether advanced methods such as NBEATSx, LightGBM and Deep Neural Networks can outperform traditional time series models like SARIMAX, Ridge Regression and Seasonal Naive in the context of load forecasting. The analysis of results is split into two parts: Evaluation using traditional loss metrics (RMSE, MAE, MAPE) and evaluation based on actual incurred balance costs. When evaluating with traditional loss metrics, the gradient boosting LightGBM model demonstrated superior overall performance, consistently outperforming other approaches by a considerable margin. Furthermore, NBEATSx demonstrates significant improvement over the traditional time series model SARIMAX. Notably, ensemble methods, in this case the average over all model predictions, showed significant potential in mitigating individual model limitations, improving on every single model's performance. Critically, the study reveals that model performance is not uniform across different time scales, with significant variations in accuracy during working hours, weekends, and holiday periods. On the other hand, using incurred balance costs as the evaluation metric, which aligns more closely with practitioners' economic incentives, the tendencies of each model to over- or underestimate the expected value given a temporal dimension, such as hour of day, shuffle the leaderboard. In this case, SARIMAX, staying closer to the expected value, performs better than the complex NBEATSx model which consistently places the balance group in the unfavourable, costlier direction. Additionally, gradient boosting seems to learn holidays better than other model classes, putting it in the best performing position. The findings underscore the importance of moving beyond traditional symmetric loss metrics to train and evaluate forecasting models and instead emphasise the economic implications of prediction errors by accounting for the asymmetry of imbalance costs across temporal features like hour of day. The results also suggest that future work should focus on advanced ensembling techniques to combine models dynamically based on their complementary strengths and biases and devoting further attention to the critical holidays, which have been a pivotal factor in each model's final performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The electricity market is a complex, interconnected system reliant on recurrent forecasting and optimisation all the way up to the point of physical delivery of power. Without forecasts, no stakeholder in the electricity sector could function. Arguably, the two most important areas consist of the optimisation of production over a price forecast and the stable operation of the national grid, which depends on production and consumption forecasts. Addressing the latter, in the Swiss case, the transmission system operator (TSO), Swissgrid, is bound by the Electricity Supply Act (StromVG) to ensure balanced production and consumption at the highest grid level in order to achieve a target frequency of 50 Hertz. Small deviations aside, maintaining this target frequency ensures grid stability and prevents blackouts. Taking the perspective of the TSO, load forecasts also play a major role in the planned maintenance of grid elements.

Since 2009, the TSO (Swissgrid) procures additional production or consumption in case of frequency deviations on the ancillary services market in the form of frequency control energy. Control energy is reserved ahead of time and activated in case of grid imbalance in either direction. Owners of production or consumption facilities may enter into a bilateral contract with Swissgrid, enabling them to offer their assets in the bidding for control energy. Around 61 MW are allocated to the frequency containment reserve (FCR), 400 MW for the automatic frequency restoration reserve (aFRR) and 500 MW for the manual frequency restoration reserve (mFRR). These three components differ in the way they are activated and in the duration they are required to provide the reserved capacity. Figure 1.1 illustrates this concept: Following a frequency drop caused by underproduction, FCR ramps up within 30 seconds to contain the deviation, followed by aFRR restoring the frequency back to 50 Hertz after ramping up within 5 minutes. Lastly,

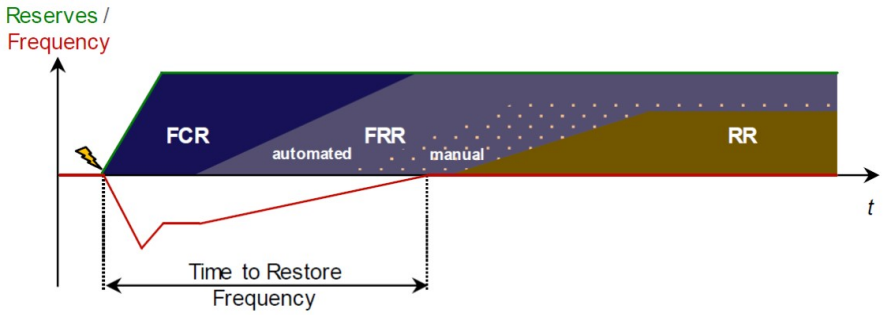


Figure 1.1: Order of operations of control energy (Swissgrid, 2020)

mFRR and the replacement reserves (RR) keep the frequency on target after 15 min in case of longer lasting grid disturbances (Swissgrid, 2020). In the case of up-regulation, i.e. positive control energy, the TSO pays the asset owner, whereas in the case of negative control energy, the TSO is paid by the asset owner. In both cases, the TSO is in a financially unfavourable situation because they either pay a high price for positive up-regulation or receive a very low price for down-regulation, because asset owners enter the tender process with an intention of earning a premium over the operation on the spot market.

1.1.1 The Balance Group Manager's Perspective

With the TSO's mechanism for managing grid imbalances outlined, the question arises: Who bears the costs of procuring control energy? In Switzerland, the concept of balance groups applies. A balance group is an area of production and consumption units managed by the balance group manager (BGM), enabling them to trade, produce, and deliver electricity. Each balance group is responsible for minimising the delta between its predicted production and consumption reported to the TSO. Should a balance group be long or short due to inevitable forecast errors, i.e., over- or under-producing electricity compared to the current consumption, the TSO will charge the BGM for balance energy, which constitutes the control energy needed to compensate for these differences (Art. 15 Para. 1 lit. b StromVV). Under the Swiss dual-price balance energy system, the BGM gets incentivised to align their production and consumption as closely as possible due to a penalisation in both directions. In the imbalanced case, the balance group is either short (deficit) or long (surplus), and the amount of balance energy that has to be paid or received depends on the spot price and control energy prices. When the balance group is **short** (deficit), the BGM must pay an amount in $\frac{\text{ct}}{\text{kWh}}$ calculated as:

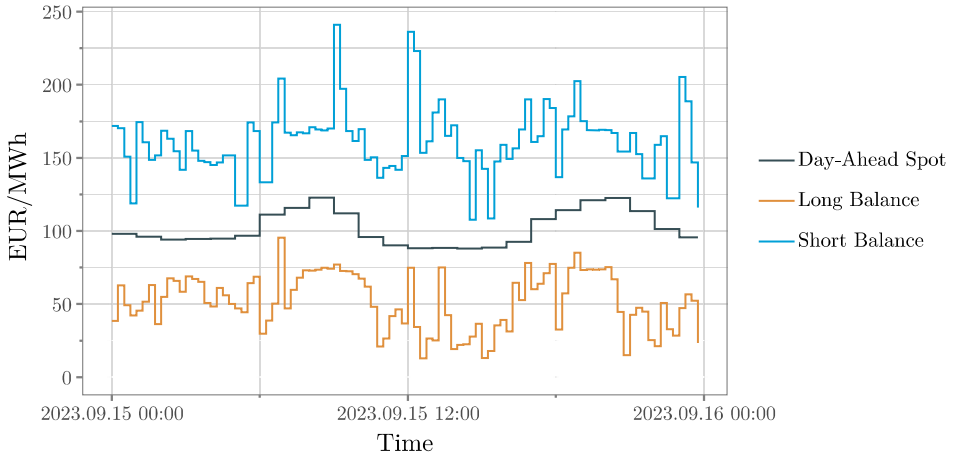


Figure 1.2: Comparison of balance energy prices (long and short) with day-ahead spot prices for September 15, 2023. The balance energy prices create a channel around the day-ahead spot price by design, making imbalances unattractive to BGMs.

$$(\max(P_{\text{spot}}, P_{\text{sek}+}, P_{\text{ter}+}) + P_1) \cdot \alpha_1$$

where α_1 is the penalty factor applied to deficits, with $\alpha_1 = 1.1$. When the balance group is **long** (surplus), the BGM receives an amount in $\frac{\text{ct}}{\text{kWh}}$ calculated as:

$$(\min(P_{\text{spot}}, P_{\text{sek}-}, P_{\text{ter}-}) - P_1) \cdot \alpha_2$$

where α_2 is the reward factor applied to surpluses, with $\alpha_2 = 0.9$ (Swissgrid, 2021). P_{spot}, P_{sek} and P_{ter} denote the prices of the day-ahead spot market, secondary (aFRR) and tertiary (mFRR) control energy prices. If there is no activation of control energy then P_{sek} and P_{ter} are not taken into account and the day-ahead spot price remains the only factor in the maximum/minimum statements. The base price used in both cases is:

$$P_1 = 0.5 \frac{\text{ct}}{\text{kWh}} = 5 \frac{\text{EUR}}{\text{MWh}}$$

As just shown and visualised in Figure 1.2, the balance energy prices form a collar around the day-ahead spot price and control energy prices by design. From the perspective of the TSO, this setup both creates an incentive for BGMs to correctly align their balance group and provides working capital for the provision of ancillary services. From the perspective of the BGM,

	long EUR/MWh	short EUR/MWh
mean	68.26	236.95
std	259.27	237.44
min	-16,505.50	-380.30
25%	32.30	114.40
50%	58.10	176.60
75%	129.70	297.60
max	731.60	14 864.60

Table 1.1: Summary Statistics of Balance Energy Prices since Dec 2020

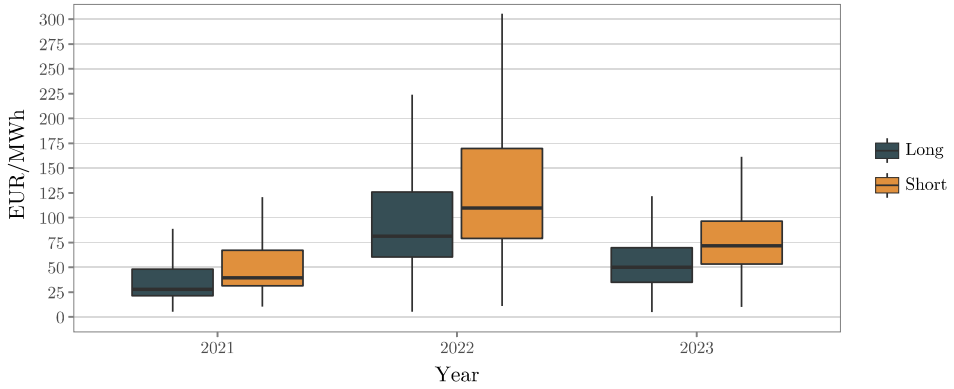


Figure 1.3: Distribution of balance energy costs (long and short) calculated as the loss in revenue compared to a transaction at the day-ahead spot price from 2021 to 2023. Outliers are omitted for legibility.

forecast errors now become very costly and an operative concern. As shown in Table 1.1, BGMs might even have to pay the TSO for long imbalance due to negative prices. Figure 1.3 shows the distribution of balance energy costs relative to the day-ahead spot price by year between 2021 and 2023. The impact of the energy crisis and its subsequent price explosion is clearly visible in 2022. However, comparing 2021 to 2023, it becomes apparent that balance costs remain elevated. Clearly, the accurate prediction of consumption has become a more pressing topic with increased financial incentives for BGMs.

1.1.2 Concept Drift due to Renewable Integration

Having outlined the financial interest of BGMs to have accurate load forecasts in order to minimise balance energy costs, I will introduce an external factor rendering forecasts more difficult: The rapid expansion of solar photovoltaic (PV) systems both on residential and commercial property. As more

households generate their own electricity, their reliance on the grid during daylight hours is significantly reduced, impacting their load profile virtually immediately following installation. Regarding the affected hours in the day, solar generation for self-supply diminishes the demand mostly around the peak window, which consists of the hours surrounding midday when solar radiation is at its highest. In theory, the installation of a PV system constitutes a considerable structural break in the load profile time series of any single customer. However, in larger balance groups, the adaption is not seen immediately, but rather introduces a seemingly continuous shift known as concept drift in time series forecasting, where historical patterns no longer represent future trends. Forecasting models trained on long horizons of past data may struggle to adapt to these changes, requiring constant recalibration or dedicated approaches. This added complexity highlights the need for techniques that can adjust to both short-term fluctuations and long-term structural shifts in load demand caused by the adoption of renewable energy. Figure 1.4 shows average end-user energy consumption by hour of the day and year for each quarter (1 to 4) for the years 2009 to 2024. Historically, daily load profiles shifted considerably, especially over the past four years, supporting the issue outlined above being a concern to BGMs in Switzerland. Figure 1.5 further confirms the fact that the hours most affected by this shift are in the peak window and that this development has only grown stronger in recent years, except for the effect of the pandemic.

1.2 Problem Statement and Objective

Recent advances in machine learning have led to a growing focus among practitioners on deep learning models for time series problems. Oftentimes, complex and computationally expensive deep learning models are assumed to outperform traditional statistical approaches. This leads to an important question: Does the increased complexity of advanced machine learning methods translate into proportionately better predictive performance?

The goal of this paper is to answer this question in the specific case of load forecasting by benchmarking the predictive performance of a classical statistical time series model, such as SARIMAX, against a state-of-the-art deep learning model, such as N-BEATSx, taking into account the performance of other model classes. The experimental setup for the evaluation of these models is designed to be as realistic as possible, simulating a live test environment that closely aligns with the needs of practitioners in the industry. Through this analysis, the strengths and weaknesses of both approaches will be evaluated.

The contribution of this paper is therefore twofold: Firstly, it provides a realistic and transparent investigation of the performance of statistical versus

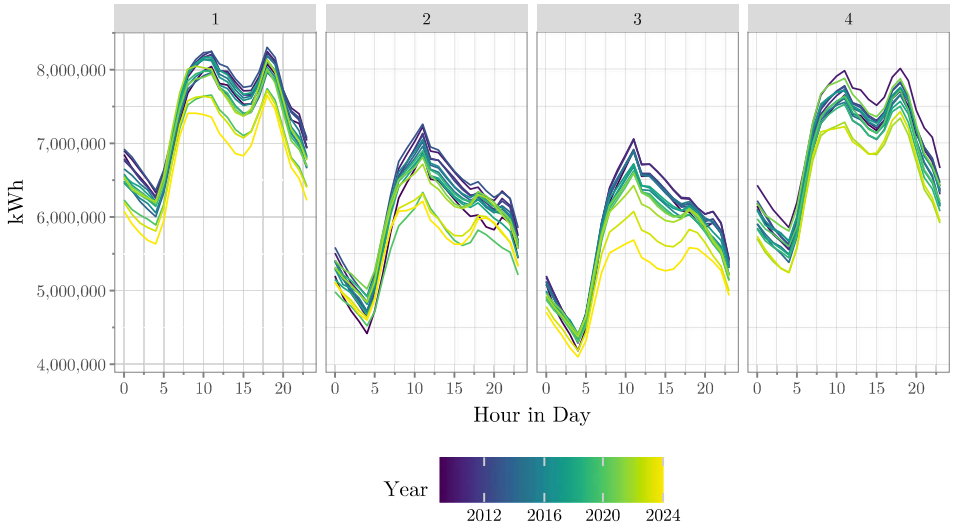


Figure 1.4: Average end-user energy consumption (kWh) by hour of the day, grouped by year and quarter since 2009. Each panel represents a quarter (1-4), showing hourly consumption patterns across years, with trends visualized by color-coded lines for each year. Note that the year 2024 was not complete at time of writing, with its last quarter missing.

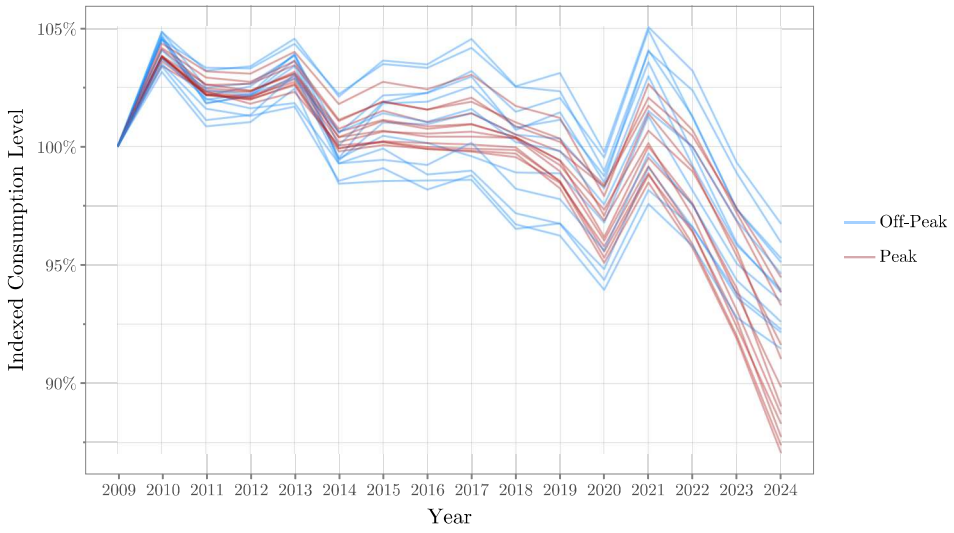


Figure 1.5: Indexed hourly energy consumption by year and time of day, with values relative to 2009 levels. Each line represents one of the 24 hours in a day, highlighting changes in consumption patterns over time for this hour. Peak hours (08:00–18:00 CET) are distinguished by color to emphasise variations during high-demand periods.

deep learning models on Swiss data, addressing a question relevant to both transmission system operators (TSOs) and utilities. Secondly, it places a specific focus on the Swiss dual-price balance energy system, with an emphasis on the perspective of BGMs facing pressure due to increased balance energy prices as well as concept drift in their existing models.

Chapter 2

Data and Methodology

2.1 Data

In applied forecasting problems, the data usually come before the models, which is why this chapter starts by focusing on the data sources used for model training and evaluation. The data can be broadly categorised into the target variable, namely, electrical load or consumption, and exogenous variables, namely weather. Data procurement was carried out with the goal of explaining as much variance of the target variable as possible, taking into account the financial and time constraints of a semester thesis. As consumption is driven by human activity, which is influenced by societal structures and environmental conditions, much of the variance can be explained by extracting information from timestamps and using weather data. Therefore, the input variables will be limited to calendar and weather data, which does not compromise the goal of providing a fair comparison of the methods with a reasonable amount of data.

2.1.1 Load

Quarter-hourly consumption data in kWh from Swissgrid (2024) is used as the target variable. The data, recorded in 15-minute intervals, aggregates the consumption across all network levels, as reported by the Swiss distribution system operators (DSOs). According to Swissgrid (2024), not included in this dataset are the grid losses, energy consumed for the operational requirements of power plants, and the energy used to power pumps in pumped-storage hydropower plants. This dataset serves as the basis for modelling load demand and forecasting end-user electricity consumption across Switzerland's regulated network. In order to render computations with SARIMAX models on longer seasonal horizons possible, the actual data used is aggregated to hourly resolution. Historical data is available from January 2009 to end of August 2024. For the purpose of exploratory

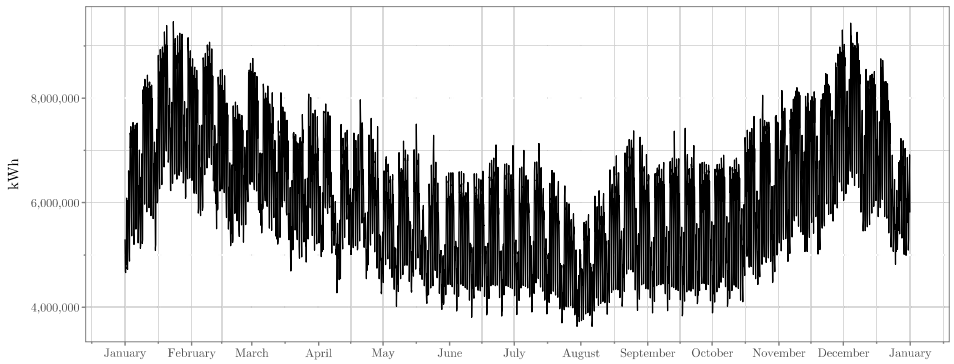


Figure 2.1: Hourly end-user energy consumption for the full year of 2023. The time series displays seasonal variations in energy usage throughout the year.

data analysis (EDA), the full history will be used, however for modelling the furthest timestamp will be in 2021 due to concept drift, which has been introduced in subsection 1.1.2. Figure 2.1 shows the aggregated hourly time series for the full year of 2023.

A key goal in time series modelling is to analyse and identify patterns in the target variable that may include:

- Trends: Long-term increases or decreases in the data.
- Seasonal Components: Periodic fluctuations at fixed intervals (e.g., daily, yearly or monthly).
- Outliers and Structural Breaks: Sudden spikes or shifts in the behavior of the series.

Trend Figure 2.2 shows the distribution of hourly consumption by year since 2009. Median consumption has decreased since its peak in 2010 by a little more than 0.5 GWh. The year 2020 is clearly visible as an anomaly due to the COVID-19 pandemic. However, since 2021, a strong downtrend is visible. The latter may likely be attributed mostly to the increase in PV installations behind metering points in the distribution network. However, there might exist other factors, such as shifts in the economic landscape, consumer changes or even government policies. For instance, there was widespread fear about blackouts in 2022 and 2023, leading to the Swiss government to mandate a winter reserve held back by hydrostorage power plants. Headlining regional newspapers for weeks, this new sentiment towards electricity might well constitute a motivating factor for consumers to shift their behaviour towards being more energy efficient.

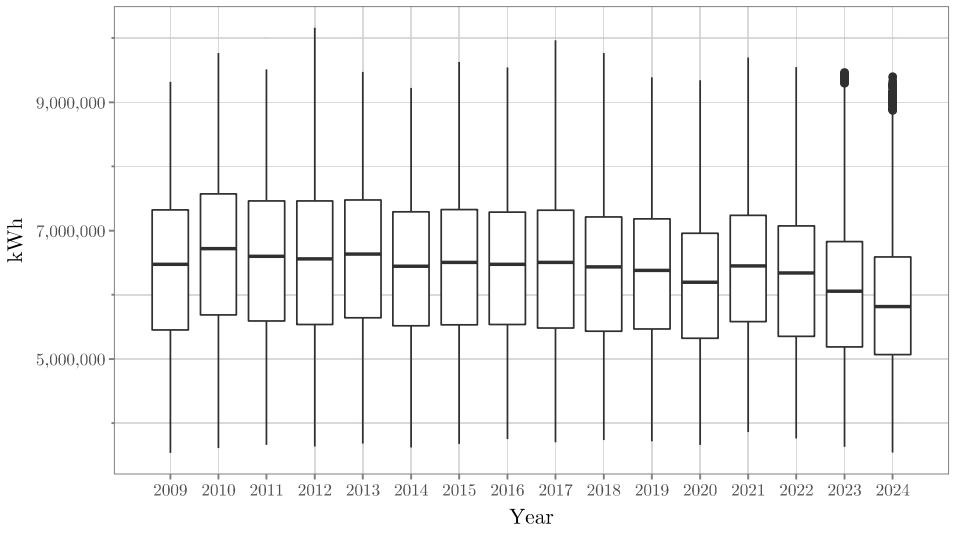


Figure 2.2: Distribution of hourly consumption by year. Note that the year 2024 contains only the first 9 months of data, likely left-skewing the distribution.

Seasonality Regarding seasonality, there is a lot to be said about electricity consumption data, as the underpinning mechanism is reflective of human behaviour, which is heavily dictated by societal structures and routine. Figure 1.4 in subsection 1.1.2 has already shown average daily load profiles with clear structures within a day and within a quarter. In winter months, evening consumption is a lot higher than in summer months due to increased demand for heating and cooking. Both business and residential activity drive up demand starting in the early morning hours and peak right around noon. Yet, these patterns are not set in stone as shown by Figure 2.3. Comparing the same data after a decade has gone by shows that both the absolute level and relative consumption patterns can change and usually do so gradually. Nightly consumption during summer months has remained around the same level for an entire decade, whereas the peak hours around noon have come down and changed the shape of the daily profile. Looking at average daily profiles omits the additional information provided by conventional work hours and holidays. Figure 2.4 and Figure 2.5 show how including information on the day type, such as workday, weekend or holiday can help explaining the variance in consumption tremendously. The same concept also holds on hourly level. In conclusion, there exist multiple seasonal structure in each day, week, month and year which make up the aggregate electricity use shown in Figure 2.6.

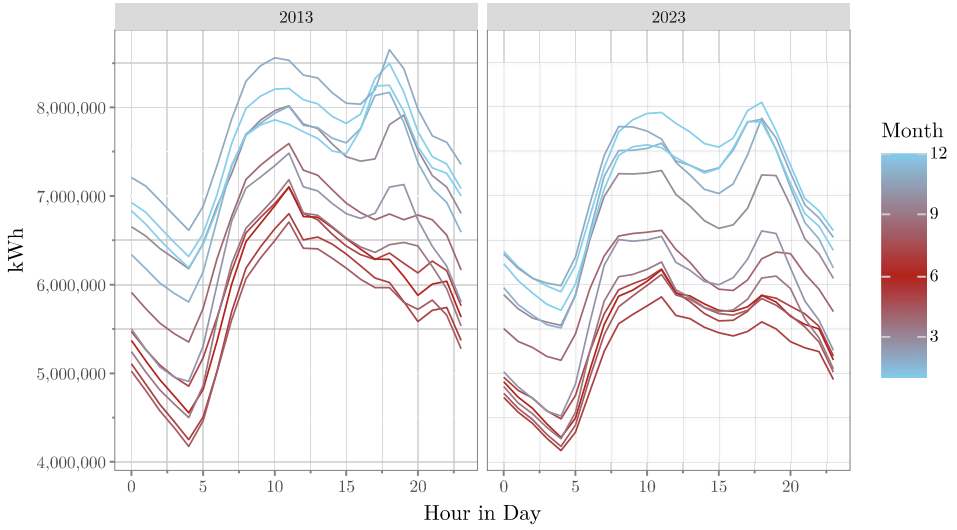


Figure 2.3: Comparison of average hourly energy consumption profiles for 2013 and 2023. Each line represents a monthly average for a given hour in day, with colors transitioning from winter months (blue) to summer months (red) and back. The panels illustrate changes in daily load patterns over the decade, given the additional dimension of month in year.

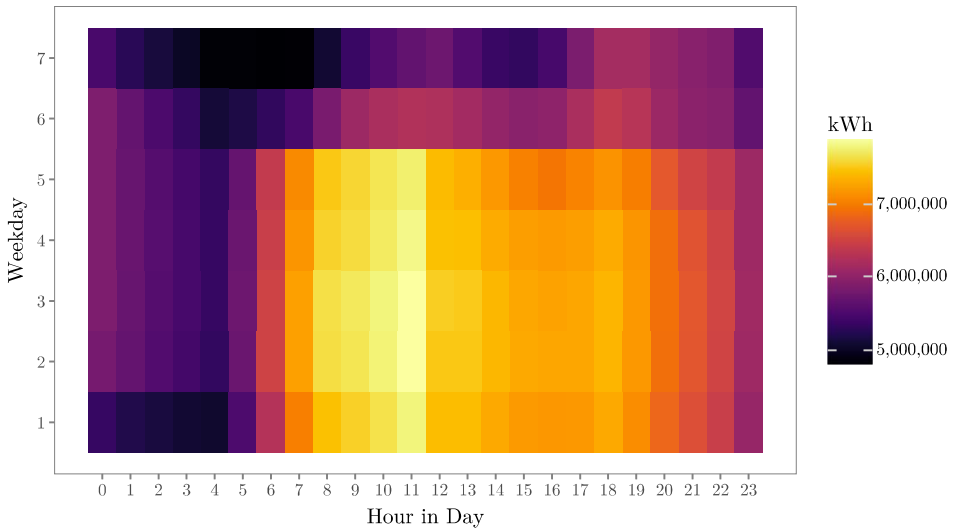


Figure 2.4: Heatmap of average hourly energy consumption by weekday and hour of the day. The color intensity represents the mean consumption (kWh), highlighting daily usage patterns and variations in energy demand throughout the week for the historical data of 2009-2024.

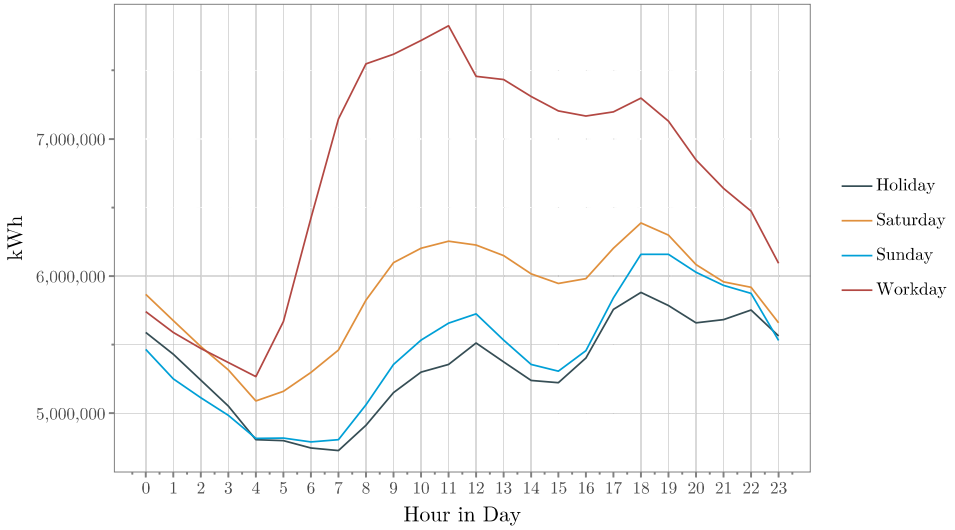


Figure 2.5: Average hourly consumption profiles categorised by day type: workdays, holidays, Saturdays, and Sundays. Each line represents the average hourly load for a specific category, highlighting distinct patterns in energy usage based on the nature of the day for the historical data of 2009-2024.

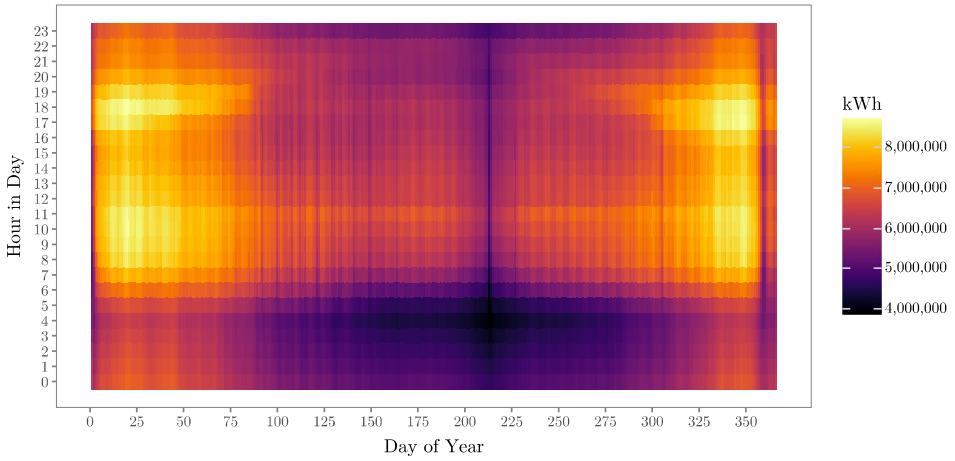


Figure 2.6: Heatmap of average hourly electricity consumption by day of the year and hour of the day. The color intensity represents the mean consumption (kWh), revealing seasonal trends and daily usage patterns across the year for the historical data of 2009-2024.

Outliers and Structural Breaks Figure 2.2 shows no apparent outliers and there are no missing values in the entire period. The data quality provided by Swissgrid (2024) is excellent. However, there seems to exist a break in the trend of the time series following the pandemic, where the consumption patterns are slowly shifting, as mentioned before.

2.1.2 Weather

Hourly weather data was obtained from OpenMeteo's API (Zippenfenig, 2023) for the cities: Zürich, Genf, Basel, Bern, Sankt Gallen, Lugano, Sion. These cities were chosen due to their size and geographic location, covering most of the populated surface area of Switzerland. Usually, one of the most important predictors of load is temperature, as heating and cooling systems respond directly to outdoor conditions. As temperature correlates very strongly with the day of the year, this effect can be predicted by day in year and can be seen Figure 2.6. During the colder months, an increase in heating demand leads to higher electricity consumption and keeps it elevated even at night. Even so, temperature is not the only predictor which can be used. Additional factors influencing demand are precipitation, fog, solar radiation and other weather variables, of which around 20 were available for this study.

During initial trials, the computational burden of the SARIMAX models posed a significant limitation to the simulations. Using more than a handful of predictors caused spikes in projected training times of up to several days on merely two weeks of historical data. Therefore, the most important external regressors were selected, namely solar radiation and temperature for Switzerland's largest city: Zurich. While it would have been possible to use the full set of around 160 weather variables in the deep learning, the same set of predictors was used for all models to keep the comparison fair.

Note that both temperature and radiation represent actual historical measurements. In practice, to make predictions, the BGM would have to rely on forecasts for these variables, introducing additional uncertainty via forecast errors into the predictions of the target variable. As this simulation study serves the purpose of model comparison, the actual values will be used for forecasting, keeping the playing ground level and the benchmark valid.

2.1.3 Calendar information

Based on the exploratory analysis of the target variable, the following calendar information was extracted from the timestamps and included in all the models:

- **Day of the Month** (d_{dom}): To account for potential daily seasonality or

trends within a month, where

$$d_{\text{dom}} \in \{1, 2, \ldots, k\}, \quad k \in \{28, 29, 30, 31\}.$$

- **Day of the Year** (d_{doy}): To account for changes in load based on yearly seasonalities, such as seasons, where

$$d_{\text{doy}} \in \{1, 2, \ldots, k\}, \quad k \in \{365, 366\}.$$

- **Day of the Week** (d_{dow}): Captures weekly seasonality patterns, such as differences between workdays and weekends, where

$$d_{\text{dow}} \in \{1, \ldots, 7\}$$

- **Month** (d_{month}): Represents the month of the year to capture seasonal effects, with $d_{\text{month}} \in \{1, 2, \ldots, 12\}$.
- **Hour** (d_{hour}): Includes the hour of the day to account for daily load patterns, where $d_{\text{hour}} \in \{0, 1, \ldots, 23\}$.
- **Year** (d_{year}): Useful for capturing long-term trends or changes over multiple years, where $d_{\text{year}} \in \mathbb{Z}$ (e.g., $d_{\text{year}} \in \{2020, 2021, \ldots\}$).

To summarise, the input categories for all models are

- **Weather Variables**: Temperature and Solar Radiation.
- **Calendar Information**: Features extracted from the timestamps to account for seasonalities unexplained by exogenous variables.

Any additional features introduced to each specific model are based on the above mentioned set of features and are mentioned in section 2.3.

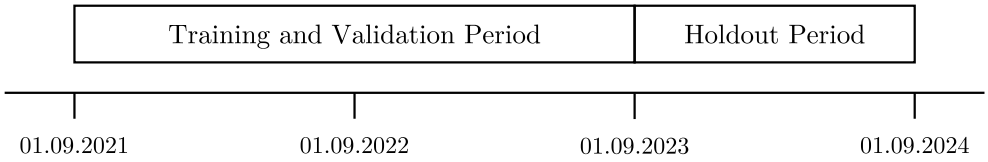


Figure 2.7: Temporal separation of training, validation and holdout periods for model selection and evaluation. Any training and tuning computations are exclusively done on the period prior to Sep 2023. Evaluation of all models takes place on the one-year holdout period directly following the training and validation split.

2.2 Experimental Setup

Before going into algorithms and model implementation, this section outlines the experimental setup of this simulation study. To lend the results credibility, it is of utmost importance to follow the best practice of time series modelling and machine learning applications, which is to have a strict separation of training and evaluation periods.

2.2.1 Model Selection

The structural break shown in subsection 2.1.1, as well as the computational complexity of some models led to the general time period of three years used for this simulation study seen in Figure 2.7. Of these three years, the first two are used for model selection, which includes training and validation in the context of hyperparameter tuning.

The last section starting in September 2023 represents the holdout period, which has been set aside at the start of the project. Having an entire year of evaluation data lends more credibility to the robustness of results, which was one of the main criticisms of previous studies by Olivares et al. (2023). Additionally, it enables the analysis of model performance over all major seasonalities: Yearly, weekly and daily.

The purpose of having a holdout set is to perform a final evaluation of all trained models. This purpose can only be fulfilled if models are not repeatedly evaluated on this period. Only if models are exclusively trained and tuned on the training and validation period and then finally evaluated only once on the holdout period can we say that this evaluation might be considered a realistic approximation of a live-test in production. Otherwise, repeated evaluation on the holdout period might lead to overfitting on the holdout set and hindsight bias, which would in turn defeat the entire purpose of having a holdout set in the first place. Therefore, the only data points the models receive from the holdout section are the exogenous features used for final model inference.

2.2.2 Model Evaluation using Loss Metrics

The following loss metrics will be used to evaluate the point forecasts over the full holdout period:

- **Root Mean Squared Error (RMSE)**:

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

  where y_i denotes the actual load at hour i, $\hat{y}_i$ the predicted value, and N the total number of observations in the holdout period, which is 8,784. RMSE penalises larger errors stronger than smaller errors.

- **Mean Absolute Error (MAE)**:

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

  which measures the average absolute magnitude of the errors in the forecasts.

- **Mean Absolute Percentage Error (MAPE)**:

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

  provided $y_i \neq 0$ for all i. MAPE expresses the error as a percentage of the actual value.

As RMSE and MAE are dependent on the absolute level of the target variable and predictions, a comparison of losses given another dimension, such as hour in day, is not possible. To enable the analysis of model performance over various seasonalities, the normalised equivalents are used:

- **Normalised Root Mean Squared Error (NRMSE)**:

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

  NRMSE adjusts RMSE relative to the mean load which makes it comparable across varying load levels.

- **Normalised Mean Absolute Error (NMAE)**:

$$\text{NMAE} = \frac{\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|}{\frac{1}{N}\sum_{i=1}^{N} y_i},$$

  which scales MAE relative to the mean load as above.

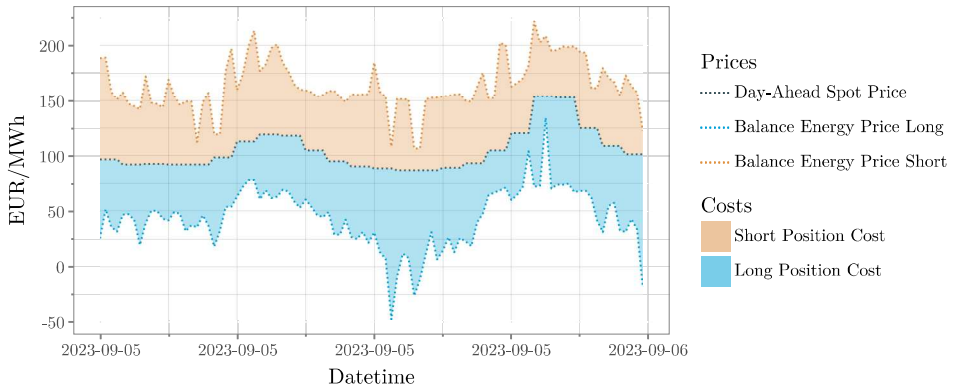


Figure 2.8: Balance cost calculation schema: The cost incurred by having an imbalance due to forecast errors is the difference to the spot price multiplied by the volume in MWh.

2.2.3 Model Evaluation using Balance Energy Costs

Other than loss metrics for point forecasts, actual historical balance energy prices are used to evaluate model performances across the board. As seen in section 1.1, imbalance is penalised in the form of energy and cash exchange with the TSO. In an ideal case, the BGM will be able to procure or sell all of their electricity at spot prices, e.g. in the day-ahead auctions. In practice, day-ahead imbalances can be corrected by flexibility and via the intraday markets. However, as a simple benchmark, the effective total balance energy cost will be calculated for each model as the sum of differences between the respective balance energy price and the day-ahead spot price multiplied by the imbalance. Figure 2.8 illustrates this concept by shading the areas between the long and short price, indicating the unit costs of imbalance per MWh for both directions in each quarter hour. As such, balance costs summarises the additional expenses or losses over the baseline of having no imbalance and relying solely on the spot market:

$$BC_{\text{total}} = \sum_{i=1}^{N} I_i \cdot \left[\mathbb{1}_{[I_i \geq 0]} \cdot (P_{\text{DA},i} - P_{\text{Long},i}) + \mathbb{1}_{[I_i < 0]} \cdot (P_{\text{Short},i} - P_{\text{DA},i}) \right]$$

where

- BC_{total} is the total balance cost in EUR incurred over a period of N observations, in this case $N = 8,784$ from September 2023 to September 2024.
- $I_i = \frac{\hat{y}_i - y_i}{4}$ is average quarter hourly imbalance in MWh caused by the forecast error of the respective model. When the BG is long, the imbal-

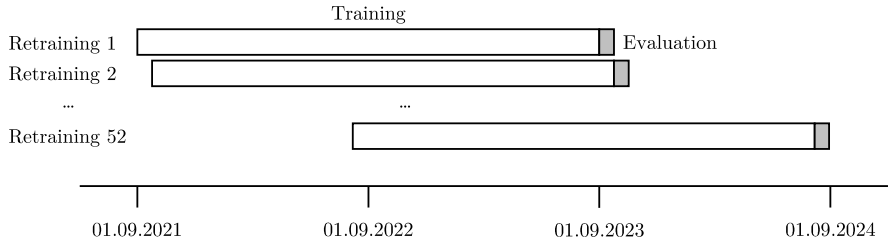


Figure 2.9: Experimental setup for the retraining scheme of models. Models which undergo weekly retraining are retrained on the training split using the best hyperparameter combination found during tuning on the training and validatio split. Evaluation takes place by making day-ahead predictions for 1 week and retraining on the new training split which includes the previous evaluation week and excludes the oldest week in the historical window.

ance is positive and vice-versa. The factor 0.25 is necessary as the load variable is being forecasted at hourly resolution while the imbalance penalisation and pricing is in quarter hourly resolution. The simplifying assumption is made that the imbalance is evenly distributed across all four quarter hours in each hour, even though this might not be the case in practice.

- $P_{\text{DA},i}$ is the day-ahead spot price at hour i in EUR/MWh.
- $P_{\text{Long},i}$ is the price to sell surpluses to the TSO in long imbalance positions at hour i in EUR/MWh.
- $P_{\text{Short},i}$ is the price to buy back deficits from the TSO in short imbalance positions at hour i in EUR/MWh.
- $\mathbb{1}_{[\cdot]}$ is the indicator function, which take a value of 1 when the condition holds and 0 otherwise.

Balance cost energy is harder to normalise as it depends on the imbalance direction, as well as the spot and the balance energy price. Therefore, all analyses except for the full holdout period have to be looked at with the potentially confounding effect of varying input factors in mind.

2.2.4 Retraining

Due to the strong effect of solar generation, as outlined in subsection 1.1.2 and visualised in Figure 2.3, model retraining might prevent deterioration of predictive performance in the holdout set. Therefore, a retraining scheme shown in Figure 2.9 was set up, where weekly retraining over the holdout period on a sliding window preceded inference over the following holdout week. It is important to note that this retraining scheme still follows strict

separation for training and evaluation sets, as the models are incrementally producing predictions on the week-ahead, one day at a time. Furthermore, no retuning takes place: The best hyperparameter combination from the training and validation splits in Figure 2.7 is fixed and carried forward throughout the entire retraining period. Due to the varying degrees of computational complexity of each model class, not all models were retrained. Specific details are outlined in section 2.3.

2.3 Models

Having shown the model inputs and experimental setup, this section outlines the selected model classes and introduced their parametrisations, tuning, retraining, as well as feature engineering approaches. Before comparing SARIMAX and NBEATSx, four other models are defined, which serve as benchmarks or challengers. In total, six model classes are compared:

- **Seasonal Naive**: This model relies on forward projecting the weekly seasonality and recurring patterns in energy consumption across workdays, Saturdays, and Sundays. The purpose of this naive benchmark is to set a lower limit for the holdout performance below which the use of another model would not make sense.
- **Ridge Regression**: This model employs ridge regression, a technique that applies $L2$ regularisation to the ordinary least squares objective used in linear regression. Due to its simplicity of implementation, it constitutes an upper limit of model complexity, should more complex models not be able to beat it, as parsimonious and interpretable models are generally more desirable in practice.
- **Multioutput Deep Neural Network**: Using a feed-forward and fully connected neural network for multioutput regression, this model constitutes a challenger to the NBEATSx architecture to answer the question if smart feature preprocessing eliminates the need for complicated architecture. As it is a simple and well known architecture, it represents an upper bound for the complexity of the network topology when using deep learning, should the more complicated NBEATSx not beat its holdout performance.
- **Recursive LightGBM**: This gradient boosting implementation producing predictions sequentially by recursively feeding previous predictions as lags relies on decision trees and is included as a challenger to the deep learning approaches to evaluate the performance of ensemble learning methods. Its ease of use and training speed gives it an edge over the more complicated neural networks and therefore represents an upper bound for development speed, which is also a considerable factor in a time-constrained corporation.
- **SARIMAX**: This classical statistical model for time series forecasting captures the linear relationships in the data, including seasonality, trends, and external regressors. It constitutes the first of two major model classes on which the spotlight of the thesis shines due to its interpretability, making it well-suited for understanding the contribution of different components. SARIMAX serves as comparison more flexible and modern methods to gauge whether the latter translate into significant performance gains to well established methods.

- **NBEATSx**: Building on the state-of-the-art N-BEATS architecture, this model enhances its performance by including exogenous covariates. Compared to traditional feed forward neural networks, it explicitly models trend and seasonality through specialised blocks. NBEATSx constitutes the second main contender, as it has been the winner several time series benchmarks in the last year and has been applied to several (Olivares et al., 2023). The inclusion of NBEATSx allows for testing whether the added complexity improves holdout performance over simpler methods, especially in the context of energy consumption forecasting.

2.3.1 Naive Benchmark

The naive benchmark model relies on weekly seasonality and recurring patterns in energy consumption across workdays, Saturdays, and Sundays to make forecasts. The model predicts the future electricity consumption to be the average historical consumption of the same weekday type over the past four weeks. Specifically, for each hourly forecast in the holdout period, the day of the week d_{dow} and hour d_{hour} of the target time point are queried. Then, the average consumption for that specific weekday-hour combination based on the previous four weeks is set as the predicted value.

2.3.2 Ridge Regression

Ridge regression serves as the second baseline model in this comparative study. The model was chosen because it constitutes a straightforward and computationally fast linear model, as it extends ordinary least squares by an L_2 penalty term on the squared magnitude of coefficients.

Model Specification

The estimator in the ridge regression is defined by Hastie (2009) as

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \alpha \sum_{j=1}^{p} \beta_j^2 \right)$$

where y_i is the target variable (electricity demand at time i), x_{ij} represents the predictor variable j at time i, β_j denotes the j-th coefficient in the linear regression equation, N is the number of observations, p is the number of predictors, and α is the tuneable regularisation hyperparameter (Hastie, 2009). Through α, the penalty term $\sum_{j=1}^{p} \beta_j^2$ can be scaled up, controlling the complexity of the model by introducing bias in the hope of reducing variance to improve holdout mean-squared error. It must be noted that ridge only asymptotically shrinks coefficients towards zero as opposed to its often

mentioned counterpart LASSO, which contains an $L1$ penalty term on the absolute value of coefficients β_j.

Feature Engineering and Preprocessing

Feature engineering for ridge regression involved generating lagged, rolling, polynomial, and interaction features, as well as encoding categorical variables and handling holidays. To summarise briefly, the following steps were taken before fitting the model:

- **Lagged Target Variable**: To capture autocorrelation in the target variable, two lagged features were added: A 24-hour lag and a 7-day rolling average.
- **Lagged Exogenous Variables**: A 7-day rolling average was introduced for temperature variable to smooth out short-term fluctuations and represent the inertia of average seasonal temperature effects.
- **Holiday Features**: Regional holidays for the Canton of Zurich were included, supplemented by manually specified dates for key seasonal periods (e.g., Easter and Ascension), as many Swiss residents go on holiday during these short-term holidays.
- **Polynomial Features**: For numeric columns, polynomial features up to degree 5 were added to allow for potential non-linear relationships between features and the target variable.
- **Interaction Terms**: Interaction terms were generated for all pairs of numeric columns to account for potential feature interactions. These terms capture the joint effect of two features, which may not be evident from individual features alone.
- **Scaling and Encoding**: Numeric and categorical variables were preprocessed separately. Numeric variables were imputed using forward filling first, then mean imputation, being mindful of data leakage by clearly separating training, validation and test periods. Additionally, they were standardised as is required for regularised methods to avoid varying penalisation based on the absolute levels of input variables. Standardisation was conducted with mean and variance computed from the training data only, in order to prevent data leakage. Categorical variables, such as holidays and ordinal calendar information, were one-hot encoded.

Hyperparameter Tuning and Holdout Evaluation

A time-series cross-validation strategy with 5 splits is used to optimise the ridge regression model's hyperparameter α, as shown in Figure A.2. Each split trains on a maximum of one year of data and validates on a 73-day

window, non-overlapping fifths of a year, to ensure that each validation set covers a diverse range of seasonal patterns. This sliding-window approach visualised in Figure A.2 ensures temporal consistency in the validation process by maintaining chronological order, as required in time-series tasks, and avoids data leakage between training and validation sets. For each split, the exogenous variables are preprocessed using a pipeline estimated on the training data separately each time. The ridge model is then fit on the training set, and predictions are made on the validation set. The root mean squared error (RMSE) is computed for each split to quantify the predictive performance.

The hyperparameter α is tuned over a grid search ranging from 5 to 20 by iterating through the grid and evaluating the average RMSE across all validation folds for each α. The results, depicted in Figure A.1, show the relationship between α and RMSE, enabling the selection of an optimal α that minimises the average validation RMSE. After identifying the best α on the validation set, the ridge model's performance is assessed on the holdout set covering the period from September 2023 to August 2024 both with and without weekly retraining, but in both cases without retuning α. It is important to note that in the dynamic retraining approach, the model is retrained weekly using historical data available up to the prediction point, not beyond that. In this case, because of the extremely fast fitting times of several seconds on two years of historical data, the training set is allowed to grow over time, including both the original training-validation data and the holdout data observed so far up until the current prediction point. Predictions are then made for the subsequent week after each retraining.

2.3.3 Multioutput Deep Neural Network

The multioutput deep neural network (DNN) was implemented to forecast day-ahead load by simultaneously predicting all 24 hourly values based on a window of historical features and lagged target values. This setup was chosen as a contrast to the recursive prediction strategy explained later on in subsection 2.3.4 in the hope that it will be able to learn relationships between predicted hours in the same way a model using its own predictions would. The way past and future predictors are fed into the model via prior transformation of the input data was inspired by the way the NBEATSx models uses past and future windows to make predictions.

Model Specification

The DNN is essentially a fully-connected multilayer perceptron (MLP) composed of an input layer, several hidden layers, and an output layer (Rosenblatt, 1961). Each layer consists of neurons connected to the neurons in the subsequent layer through learnable weights. Non-linear activation func-

tions, such as rectified linear unit (ReLU) applied to the outputs of each neuron enable MLPs to model complex relationships in data.

In a convential single-output regression setting, the output layer of the DNN would contain one node with a linear activation activation function to be able to map values to the real line. However, as we are predicting 24 hours into the future, without recursive feeding of past predictions, this setup would not be able to leverage lags and learn relationships between them and the next hour. Therefore, in this study, the output layer consists of 24 neurons, corresponding to the hourly predictions for the day-ahead horizon. The predictions can be expressed as:

$$\hat{\mathbf{y}}_{[t+1:t+H]} = f(\mathbf{y}_{[t-L:t]}, \mathbf{X}_{[t-L:t+H]}; \theta) \tag{2.1}$$

where

- f denotes the function learned by the DNN, which is parametrised by learned weights and biases θ.
- $L \in \{0, 1, ..., t\}$ corresponds to the lookback window, i.e. the number of observations in the past used to make predictions for the future (e.g. 168 hours for one week of history).
- $H \in \{1, 2, ..., N - t\}$ corresponds to the prediction horizon, i.e. the number of hours which are predicted simultaneously into the future given the current time point t. In this case, $H = 24$.
- $\mathbf{X}_{[\mathbf{t-L:t+H}]} \in \mathbb{R}^{L \times d}$ represents the exogenous variable matrix including observations from all d features in the lookback window and the prediction horizon window.
- and $\mathbf{y}_{[t-L:t]} \in \mathbb{R}^{L}$ is the vector of lagged historical values in the lookback window.
- and $\hat{\mathbf{y}}_{[t:t+H]} \in \mathbb{R}^{H}$ is the vector of predicted values for the prediction horizon, which constitutes the final output of the network after training.

With a similar structure of being able to access historical values and outputting predictions for a prediction horizon, the hope for the DNN is to come close to the performance of more complicated architectures like NBEATSx. As shown in the results section on the application of electricity prices, (Olivares et al., 2023) demonstrated a very close performance of NBEATSx and DNN, with the latter sometimes performing better than the more complex architecture of NBEATSx, especially in Belgian markets. This result was the core motivation for this additional challenger model with a specialised preprocessing approach, which will be outlined in the following subsection.

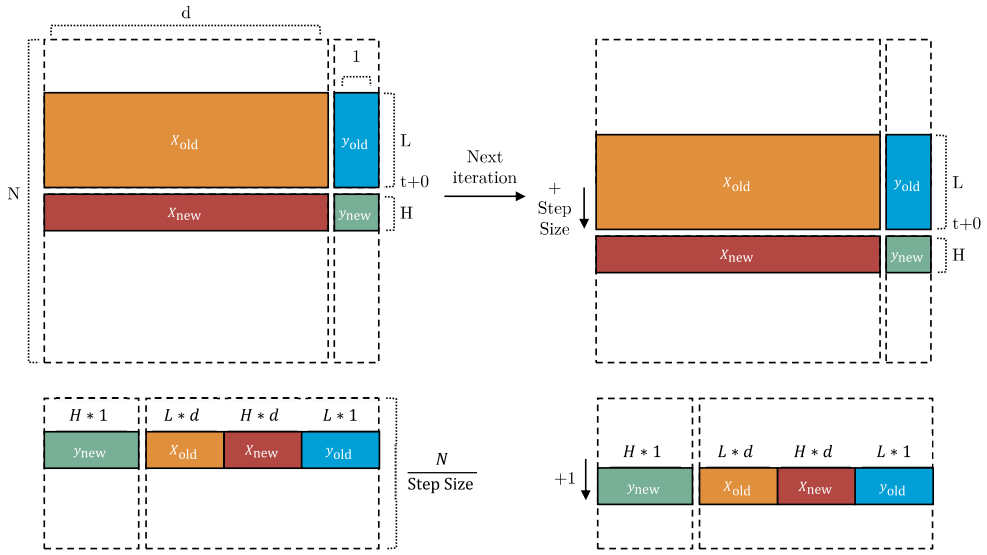


Figure 2.10: Reshaping loop logic for multioutput regression with DNN: Hourly data is reshaped from long to wide format by using a lookback window and a forecast horizon and flattening all components to wide format, outputting a dataframe with daily resolution and 24 target values for every row.

Feature Engineering and Preprocessing

Before the reshaping of the input data for the multioutput DNN, the exogenous variables were preprocessed the same way as for the ridge regression in subsection 2.3.2, including holiday dummies, scaling, cyclical and one-hot encoding based on estimates on the training data. However, no additional rolling features or lagged target variable values were included, as the setup of the lookback window was used instead.

Figure 2.10 shows the heart of the idea for the multioutput regression with the DNN. The top left table in the illustration shows the original dataset in hourly resolution at a given time point t with given L, H, N and d. Four distinct elements can be identified at each time point t, previously seen in Equation 2.1:

- $\mathbf{y}_{[t+1:t+H]} = \mathbf{y}_{\text{new}} \in \mathbb{R}^H$ (in green): The future target variable values used during training and unavailable at prediction time in the holdout set.
- $\mathbf{y}_{[t-L:t]} = \mathbf{y}_{\text{old}} \in \mathbb{R}^L$ (in blue): The past target variable values earlier than prediction time t, which are available both during prediction and training.
- $\mathbf{X}_{[t+1:t+H]} = \mathbf{X}_{\text{old}} \in \mathbb{R}^{H \times d}$ (in red): The future exogenous variables

Table 2.1: Hyperparameter search space for DNN

Hyperparameter	**Search Space**
Initial learning rate	$\{10^{-2}, 10^{-3}, \ldots, 10^{-6}\}$
Decay rate	$[0.5, 0.99]$
Decay patience (epochs)	$\{10, 25, 50, 100\}$
Number of layers	$\{1, 2, ..., 6\}$
Units per layer	$\{16, 32, 64, 128, 256, 512\}$
Activation function	{ReLU, LeakyReLU}
Batch size	$\{8, 16, 32, 64\}$
Epochs	Fixed at 2000
Patience for early stopping	Fixed at 200

used during training, which are also available at prediction time in this study. In practice, these would be weather predictions instead of actuals, but training could take place on actual values.

- $\mathbf{X}_{[t-L:t]} = \mathbf{X}_{\text{new}} \in \mathbb{R}^{L \times d}$ (in yellow): The past exogenous variables used during training, which are often available at prediction time as actual values too.

These four parts are then transformed by flattening every vector and matrix to $\mathbb{R}^{1 \times (r \cdot c)}$, where r and c represent their previous row and columns counts, respectively. Then, the resulting row vectors are stacked horizontally and separated from the new row vector of $\mathbf{y}_{\text{new}}$, as shown in the bottom left table in Figure 2.10. In the next iteration, the current point of view t is incremented by step size (24) and reshaping is reapplied, as visualised in the right half of the diagram. Setting step size $= 1$ was also tried in order to provide more rows of data with different viewpoints, however it did not improve validation performance. Interestingly, including historical exogenous variables (X_{old} made validation performance worse. This is likely due to the fact that flattening a large historical window introduces thousands of columns. Therefore, the network has a harder time to separate signal from noise. Ideally, a variable selection process would follow reshaping. However, in this study, the use of historical exogenous regressors was omitted entirely.

Hyperparameter Tuning and Holdout Evaluation

As all other models, the DNN model was tuned extensively on the training and validation split from 01.09.2021 to 31.08.2023. Compared to the ridge regression, neural networks take a lot longer to train, which is why cross validation is rarely used. Instead, a simple training-validation split was used with one year of data each, in chronological order to prevent data leakage.

The automated training loop relied on RayTune (Liaw et al., 2018), parallelising trials across CPU cores, which each used relied on Optuna, a state-of-the-art hyperparameter optimisation framework relying on the bayesian tree-structured parzen estimator (Akiba et al., 2019). The search space for key hyperparameters can be seen in Table 2.1. The hyperparameter search was conducted over a time budget of six hours, with each trial being evaluated on the validation RMSE. The DNN implementation contained a fixed patience for early stopping, after which the best weights based on validation loss were restored and the validation loss was reported to the RayTune instance. The model architecture in each trial was dynamically constructed based on the sampled hyperparameters. The Adam optimiser was used for gradient-based optimisation, with an initial learning rate determined during tuning. To mitigate overfitting, early stopping with a patience of 200 epochs was applied, alongside a learning rate reduction strategy triggered by a plateau in validation loss, with a decay patience parameter in epochs.

Following hyperparameter optimisation, the best-performing configuration was retrained on the combined training and validation set with a fixed number of epochs without early stopping. To avoid data leakage, this fixed number of epochs was determined by manually inspecting the validation loss curves from the best model in the tuning phase. The holdout evaluation was then performed once on the unseen test data. Furthermore, the same weekly retraining scheme as outlined in subsection 2.3.2 for ridge regression was used, although computations now took several hours instead of minutes, highlighting the added computational complexity.

2.3.4 Recursive LightGBM

The LightGBM model was chosen as a challenger for the deep learning approaches, as the training and tuning process is far easier and less fickle to handle while the model class of gradient boosting also provides a high degree of complexity and flexibility. As multioutput regression is not supported, a recursive prediction strategy was chosen, where earlier predictions of previous hours are fed back as lagged inputs to predict subsequent hours.

Model Specification

LightGBM is a gradient boosting framework which has as its competitive edge over other implementations computational speed, which made it well suited to this application with frequent retraining. However, one of the most popular implementations due to its success in machine learning competitions online has been XGBoost (Chen and Guestrin, 2016), which will be used to provide a short summary of the inner workings of gradient boosting.

Gradient boosting is an ensemble learning technique which combines the

predictions of weak learners, decision trees, to build a highly flexible model capable of handling non-linear exogenous variables and interactions. The approach builds decision trees sequentially, with each weak learner attempting to correct the residual errors of the previous iteration. This iterative process enables the model to progressively minimize a specified loss function, such as mean squared error for regression tasks.

Mathematically, given a dataset with m exogenous variables and n observations, the ensemble model uses K functions additively:

$$\hat{y}_i = \varphi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}$$

with $\mathcal{F}$ being the space of regression trees $\mathcal{F} = \{f(\mathbf{x}) = w_{q(x)}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$, where q is the structure of each tree f_k which maps an observation to one of T leaves which have a weight w each. The training process requires the differentiable convex objective function

$$\mathcal{L}(\varphi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$
$$\text{where} \quad \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

The regularisation term Ω penalises both the number of leaves T, as well as the leaf weights w, which helps against overfitting. Details on the additive training procedure are omitted but available in the original XGBoost paper (Chen and Guestrin, 2016). LightGBM (Ke et al., 2017) introduces several optimisations to improve the efficiency and scalability. It uses histogram-based learning to speed up tree construction, gradient-based one-side sampling (GOSS) to reduce the number of training samples, and exclusive feature bundling (EFB) to handle high-dimensional datasets effectively.

Feature Engineering and Preprocessing

In this study, a recursive forecasting strategy was used for gradient boosting, whereby the model predicts the next hour's load based on lagged predicted values and exogenous features. These predictions are iteratively used as inputs to forecast subsequent hours in the 24-hour prediction horizon. Simply put: When predicting for hour 10 on the following day, actual target values for hours 0 through 9 are not yet available, hence previous predictions are used in place. The recursive framework can be expressed as:

$$\hat{y}_{t+1} = f(\mathbf{X}_{t+1}, \hat{y}_t, \hat{y}_{t-1}, \ldots, \hat{y}_{t-24}, \hat{y}_{t-145}, \hat{y}_{t-146}, \ldots, \hat{y}_{t-168}; \mathcal{F})$$

Table 2.2: Hyperparameter search space for LightGBM

Hyperparameter	**Search Space**
Boosting rounds (Estimators)	$\{20, 21, \ldots, 800\}$
Learning rate	$[10^{-6}, \ldots, 0.5]$
Max leaves	$\{10, 11, \ldots, 100\}$
Subsampling ratio	$[0.5, 1.0]$
L1 regularization (alpha)	$[0, 100]$
L2 regularization (lambda)	$[0, 100]$

where

- f denotes the function learned by LightGBM, parametrised by the space of of regression trees $\mathcal{F}$.
- $\mathbf{X}_{t+1} \in \mathbb{R}^m$ represents the exogenous feature vector at time $t+1$ with m exogenous variables and one observation.
- $\hat{y}_{t+1}$ corresponds to the predicted value at time $t+h$, which is recursively used in subsequent predictions of the following 24 hour block. Note that only important lags at hours $\{1, ..., 24, 145, ...168\}$ are used, as there exists less predictive power in intermediate hours. Reducing the number of less predictive input features reduces the size of the sample space for creation of decision trees, increasing predictive performance.

During training time, the LGBM model receives actual target variable lags at each of the specified hours instead of using own predictions. All other preprocessing closely aligns with the one for ridge regression.

Hyperparameter Tuning and Holdout Evaluation

Hyperparameter optimisation was performed using RayTune (Liaw et al., 2018) and Optuna (Akiba et al., 2019), as before for DNNs. The tuning function received validation RMSE on the same training and validation split as before, with a time budget of thirty minutes allocated for the search. During this half hour, around the same number of trials ($\approx$ 200) completed as in 6 hours for the DNN implementation. The best-performing hyperparameters were identified on the validation set and subsequently used to retrain the model on the combined training and validation data before evaluation on the holdout set.

The trained LightGBM model was applied to the holdout set using the recursive prediction framework already used during validation. Like ridge regression, the model was integrated into a weekly retraining scheme to account for concept drift in the data, which only took several minutes to run.

2.3.5 SARIMAX

The Classical Approach to Time Series Modelling

The SARIMAX (Seasonal Autoregressive Integrated Moving Average with eXogenous factors) model comes from the class of classical time series modelling, which has been the foundation of producing forecasts for decades. Long before machine learning models came around, the general approach of classical time series forecasting was to identify and remove trend and seasonality components before modelling the so-called residuals (Brockwell and Davis, 2016). In doing so, the underlying assumption is that the estimation of time-dependent processes could be done additively by splitting up a single task into three:

$$X_t = m_t + s_t + Y_t$$

Defining X_t as the random variable producing observation x_t at each specific time t, we have the deterministic trend component m_t, a deterministic seasonal component s_t and the residual Y_t which is hoped to be stationary. When the residual consists of i.i.d. noise, the minimum mean squared error (MSE) predictor is simply a constant. In the latter case, there is no other model which can be applied to extract further predictive information. However, if the residual deviates from this i.i.d. assumption, then further modelling steps are possible. Specifically in the case where there exists an autocorrelation structure, using the correlations between the observed values and their lags might provide insight for making forecasts.

Stationarity of the residual Y_t, or even directly $\{X_t\}$, plays a crucial role in classical time series modelling as it results in desirable minimum mean squared error linear prediction properties. A time series $\{X_t\}$ is said to be stationary if it maintains a constant mean, variance and autocovariance (ACV) over time. Formally, $\{X_t\}$ is (weakly) stationary if the following conditions hold:

1. **Constant Mean**: $\forall t : \ \mu_X(t) = \mathbb{E}[X_t]$, where $\mu_X(t)$ is independent of t.
2. **Constant Autocovariance**: $\forall t, h : \ \gamma_X(h) := \gamma_X(t+h, t) = \mathrm{Cov}(X_t, X_{t+h})$, where $\gamma_X(t+h, t)$ is independent of t.

Brockwell and Davis (2016) demonstrate that, given stationarity, the best linear predictor in terms of minimising mean-squared error (MSE) for $X_n + h$ given X_n depends exclusively on the mean and the autocorrelation function (ACF) of the time series $\{X_t\}$. Hence, stationarity enables us to disregard the joint distributions of all random variables in the series and instead work with the first two moments uniquely. Coming back to the decomposition of trend, seasonality and residual, the goal of having a stationary residual Y_t is

now clearer: It enables us to use a linear predictor which is guaranteed to be MSE optimal.

There exists, however, a different approach to achieve stationarity and therewith an entirely different approach to the previous decomposition. Box and Jenkins (1976) proposed differencing the time series to remove trends and make the series stationary. This method involves taking the difference between consecutive observations and is referred to as first order differencing, depending on how many times it is applied. In cases where a time series exhibits more complex seasonal or cyclical patterns, higher-order differencing or seasonal differencing may be applied to achieve stationarity, i.e. removing trend and seasonality components. This method avoids assumptions about trend extrapolation or in-sample smoothing techniques, which often impose limits on forecasting out of the observed time horizon. The Box-Jenkins methodology plays a central role in the classical model applied in this thesis as it forms the I in SARIMAX. Other than the I, there also exist AR, MA, S and X, which will be explained in the following section.

Model Specification

Components of SARIMAX: AR, I, MA, S, and X The SARIMAX model is an extension of ARIMA that incorporates seasonality and external regressors, offering a robust framework for time series analysis when data exhibits both seasonal patterns and external influences. The model is composed of multiple parts, beginning with differencing to achieve stationarity, followed by autoregressive and moving average components, then additionally including seasonality and external factors. Large parts of this section are building on Brockwell and Davis (2016), however the external regressors are based on the work by Vagropoulos et al. (2016).

Differencing (I) The integrated component I addresses non-stationarity through differencing. A time series $\{Y_t\}$ often contains trends violating stationarity. To mitigate these, the series is differenced d times until the stationary criteria based on the sample moments are fulfilled:

$$W_t = \nabla^d X_t = (1 - B)^d X_t,$$

where B is the backshift operator, defined by $BX_t = X_{t-1}$. Here, W_t represents the differenced series, stationary under appropriate d. Brockwell and Davis (2016) list the two examples of both linear trend components and polynomial trends being removed after differencing once or k times depending on the degree k of the polynomial, respectively. This suggests that applying the differencing often enough will eventually lead to a stationary time series.

Autoregressive Component (AR) The autoregressive component $AR(p)$ captures the influence of past values in the series. For a time series $\{W_t\}$, the $AR(p)$ process is given by:

$$W_t = \varphi_1 W_{t-1} + \varphi_2 W_{t-2} + \cdots + \varphi_p W_{t-p} + \varepsilon_t,$$

where $\varphi_1, \ldots, \varphi_p$ are the autoregressive parameters, and ε_t is white noise with mean zero and constant variance σ^2. This process describes how current values are influenced by a weighted sum of past p values.

Moving Average Component (MA) The moving average component $MA(q)$ addresses serial correlation in the error terms by modelling the influence of past forecast errors. For a series W_t, the $MA(q)$ model is expressed as:

$$W_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q},$$

where $\theta_1, \ldots, \theta_q$ are the moving average parameters. This formulation corrects serial dependencies by using the error terms from previous time steps.

Combining AR, I and MA (ARIMA) Combining the previous AR, I and MA parts leads to the ARIMA(p, d, q) model, which can be represented by the linear difference equation

$$W_t - \varphi_1 W_{t-1} - \cdots - \varphi_p W_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

where $\{\varepsilon_t\}$ is white noise with constant zero mean and variance σ^2. Brockwell and Davis (2016) suggest the short form using the backwards operator and polynomials:

$$\varphi(B) W_t = \theta(B) \varepsilon_t$$

where $\varphi(\cdot)$ and $\theta(\cdot)$ are defined as the pth and qth degree polynomials:

$$\begin{aligned} \varphi(z) &= 1 - \varphi_1 z - \cdots - \varphi_p z^p \\ \theta(z) &= 1 + \theta_1 z + \cdots + \theta_q z^q \end{aligned}$$

with B representing the backward shift operator ($B^j X_t = X_{t-j}, j = 0, \pm 1, \ldots$) introduced earlier. Using $\{W_t\}$ suggests that the data has already been differenced. Instead using the original time series $\{X_t\}$, the difference equation for the ARIMA model is defined as

$$\varphi^*(B) X_t = \varphi(B)(1 - B)^d X_t = \theta(B) \varepsilon_t$$

Seasonal Component (S) The seasonal component extends the ARIMA model to account for seasonality, making it suitable for data with periodic patterns. Namely, introducing the parameters $\{P, D, Q, s\}$, the original time series $\{X_t\}$ is differenced twice using the original differencing parameter d and the new seasonal differencing parameter D which is applied at the seasonal frequency $s \in \mathbb{Z}$. Additionally, seasonal autocorrelation and moving average terms are included on top of the existing ones at the seasonal frequency using the seasonal backshift operator B^s. Therefore, using the differencing parameters $d, D \in \mathbb{Z}$, the time series $\{X_t\}$ can be modeled as a seasonal SARIMA$(p, d, q) \times (P, D, Q)_s$ process with period s:

$$\varphi(B)\Phi(B^s)(1-B)^d(1-B^s)^D X_t = \theta(B)\Theta(B^s)\varepsilon_t$$

where

$$\begin{aligned}
\varphi(z) &= 1 - \varphi_1 z - \cdots - \varphi_p z^p, \\
\Phi(z) &= 1 - \Phi_1 z - \cdots - \Phi_P z^P, \\
\theta(z) &= 1 + \theta_1 z + \cdots + \theta_q z^q, \\
\Theta(z) &= 1 + \Theta_1 z + \cdots + \Theta_Q z^Q.
\end{aligned}$$

represent the autoregressive and moving average polynomials for the regular and seasonal components. These polynomials represent the repeated application of the regular and seasonal backshift operators to the differenced time series.

External Regressors (X) To allow the inclusion of external regressors, denoted as X'_t, Vagropoulos et al. (2016) have proposed the addition of the additive component usually seen in ordinary least squares regression, which accounts for the effect of exogenous covariates at time t. In the implementation of SARIMAX used in this model, only the most recent information is taken into account, as the addition of historical information for exogenous covariates can easily be done in feature engineering by introducing lags or summary statistics as external covariates as well. In the context of load forecasting, the inclusion of external regressors is highly important, as important predictors such as temperature are now included in the model. A potential weakness the model might suffer from is that non-linear relationships are not intrinsically modelled and have to be manually feature engineered.

Complete SARIMAX Model The full SARIMA$(p, d, q) \times (P, D, Q)_s$ model can thus be written as:

$$\varphi(B)\Phi(B^s)(1-B)^d(1-B^s)^D X_t = \sum_{k=1}^{K} \beta_k X'_{k,t} + \theta(B)\Theta(B^s)\varepsilon_t$$

where the parameters $\varphi, \theta, \Phi, \Theta$ capture the non-seasonal and seasonal dynamics of the series, $\beta_k X'_{k,t}$ adjusts for the exogenous effects of regressors $k \in \{1, ..., K\}$ at time t, and ε_t constitutes the residual white noise which should no longer contain any information. In this formulation:

- p and q represent the orders of the autoregressive and moving average terms, respectively.
- d and D indicate the levels of non-seasonal and seasonal differencing applied to the time series.
- P and Q are the orders of the seasonal AR and MA terms.
- s denotes the length of the seasonal cycle.

Feature Engineering and Preprocessing

The feature engineering approach for SARIMAX is similar to the other models in order to ensure a fair comparison. For instance, missing values are forward filled and mean imputed based on estimates from the training set, calendar features are introduces and holiday dummies created the same way as for all other models. A rolling lagged feature for temperature is rolled like in ridge regression, though no additional preprocessing is added for the target variable, as SARIMAX works with lags already. Therefore, to keep a level playing field with the DNN and NBEATSx, no additional help is given via feature engineering here.

Order Selection and Holdout Evaluation

Before, selection of the order, stationarity of the time series is typically checked, either via the autocorrelation function (ACF) and the partial autocorrelation function (PACF), or via stationarity tests, such as the Augmented Dickey Fuller (ADF) for unit roots. For an i.i.d. process with finite variance, approximately $0.95n$ of the ACF values should fall within the $\pm 1.96/\sqrt{n}$ range, where n is the number of lags. Looking at Figure 2.11, we can reject the null hypothesis of stationarity due to the apparent seasonality and the lack of decay in the ACF. From EDA, we know that the time series exhibits daily, but more importantly weekly seasonality, so seasonal differencing is required in this special case. From Figure 2.11, the autocorrelation at lag 168 is stronger than at lag 24. Therefore, setting the parameter $D = 1$ at $s = 168$ is an initial step. In fact, this decision can be empirically supported by comparing two SARIMAX$(1,1,1) \times (1,1,1)_s$ models on a training period of one

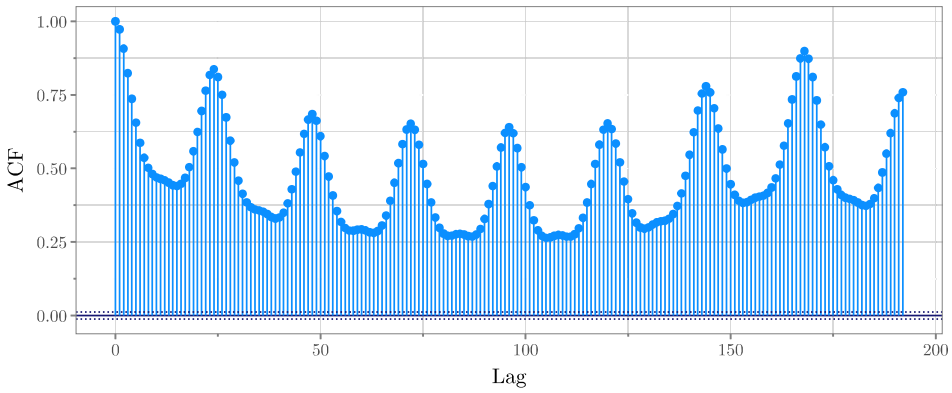


Figure 2.11: ACF plot of original time series without differencing

month (01.06.2023 - 30.06.2023) and evaluating their performance on a validation set of one month (01.07.2023 - 31.07.2023). Figure 2.12 nicely shows the issue of setting a seasonality: The underlying seasonality in the electricity data is mostly governed by the social construct of weekdays. When moving from weekdays to weekends and vice-versa, the 24 hour seasonality causes severe under- and overestimation as it takes the previous day as a reference point, instead of the same weekday type a week ago. Therefore, the seasonal parameter s is set to a full week, which is 168 hours for the hourly time series.

After seasonal differencing, an additional differencing degree of $d = 1$ produces Figure 2.13, which shows much more rapid decay of the ACF, though it does not remain in statistically insignificant territory. However, in validation performance, this differencing approach performed by far the best, suggesting that SARIMAX might be limited by its support for a singular seasonality only. The ADF test rejected the null-hypothesis of the time series containing a unit root at a p-value of approximately zero, although this has to be interpreted cautiously as ADF is not qualified to test for seasonal stationarity. One could then proceed and look at ACF and PACF to identify candidate values for p, q, P, and Q. However, in this application, the selection of candidates was mostly made taking into account two perspectives:

- **Computational complexity**: In the experimentation phase with the SARIMAX models it has become clear that their computation cost becomes at least an order of magnitude larger than neural networks on the same amount of data for lags larger than 20. This problem is significantly aggravated when using lags at long seasonalities. As 168 represents a long seasonality, the seasonal autoregressive order and seasonal moving average orders are limited to 1 at most. Anything larger becomes computationally infeasible within a reasonable amount

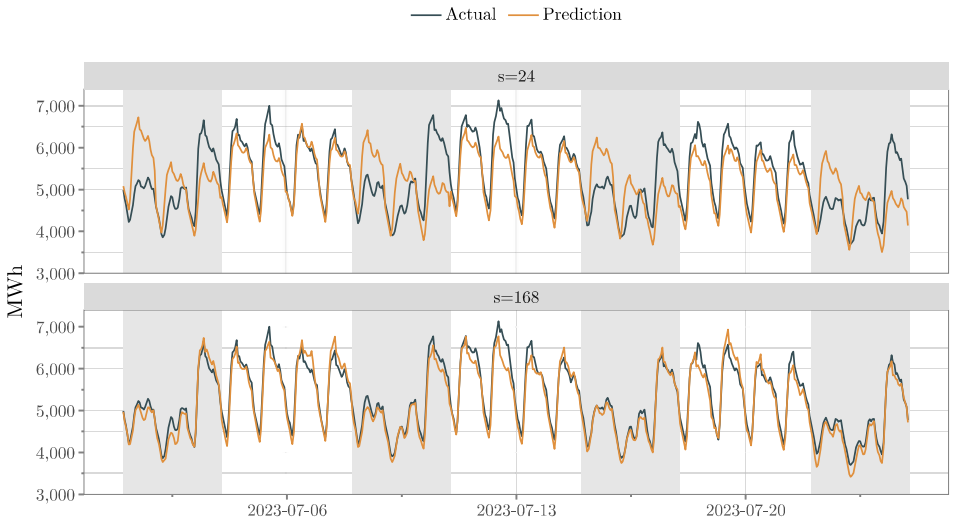


Figure 2.12: Comparing validation performance of two SARIMAX$(1,1,1) \times (1,1,1)_s$ models for $s \in \{24, 168\}$. When moving from weekdays to weekends and vice-versa, the 24 hour seasonality causes severe under- and overestimation when moving from workdays to weekends and vice-versa (shaded areas).

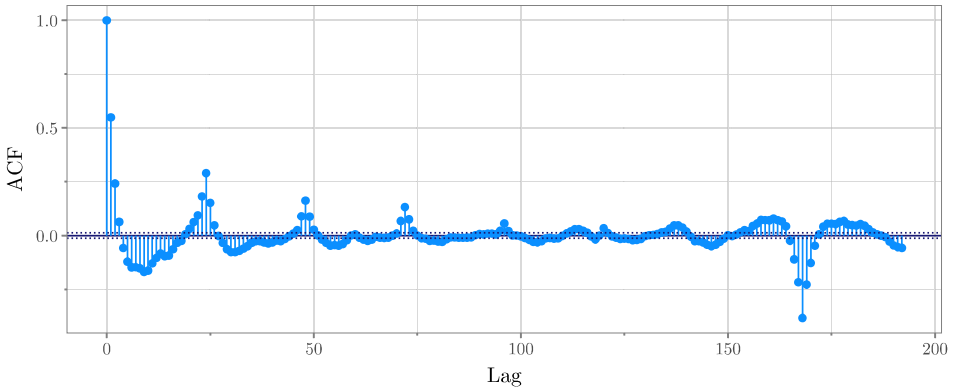


Figure 2.13: ACF plot of the seasonally ($D = 1, s = 168$) differenced and regularly differenced ($d = 1$) time series. Significantly large values remain, however this setup turned out as the optimal one for making predictions on the validation set.

Table 2.3: Hyperparameter search space for SARIMAX

Hyperparameter	**Search Space**
Autoregressive order (p)	$\{0, 1, \ldots, 24\}$
Moving average order (q)	$\{0, 1, \ldots, 24\}$
Seasonal autoregressive order (P)	$\{0, 1\}$
Seasonal moving average order (Q)	$\{0, 1\}$
Differencing order (d)	fixed at 1
Seasonal differencing order (D)	fixed at 1
Seasonal period	fixed at 168

of time.

- **Domain Knowledge**: From working with the data, it has become clear, that hours within the close proximity, i.e. the same day, are most relevant for predicting the absolute level of following hours. Therefore, the autoregressive and moving average order are limited to 24 respectively.

With this in mind, the search space for order selection is outlined in Table 2.3. Three different approaches were taken to estimate the best parameter combination:

- **AutoARIMA:** This method leverages an automated stepwise approach to order selection by optimising based on the Akaike Information Criterion (AIC).
- **Manual Validation:** In this approach, exhaustive search was performed over individual parameters (p, q, P, Q) by splitting the data into training (June 2023) and validation (July 2023) sets. Each parameter was varied independently while the others were held constant, and models were evaluated based on validation RMSE.
- **Ray Tune with Optuna:** Like for previous models, the approach of tuning on the training and validation split was also implemented using Optuna for optimisation and RayTune for parallelised trial execution. However, due to the computational complexity, only the reduced amount of data as in the manual validation approach was used. A time budget of six hours was allocated for the search, and models were evaluated based on RMSE on the validation set. This method enabled exploration of the full parameter space while balancing computational feasibility.

Each of these methods has advantages and drawbacks. AutoARIMA offers simplicity and automation, though a stepwise search is not guaranteed to provide a global optimum. Manual validation ensures detailed control and

interpretability, though synergies between parameters are disregarded, as all others are kept constant. RayTune with Optuna should in practice be the most useful approach, as it combines an intelligent search strategy over all possible combinations, although the interpretation of results has to be taken with a grain of salt as the validation period was much smaller than in previous models due to the increased computational cost. Keeping in mind that the goal is maximum predictive performance, the decision was made based on the performance of each approach on the validation set, where the RayTune model SARIMAX$(7,1,3) \times (0,1,1)_{168}$ ended up outperforming.

The best order was then selected to retrain the SARIMAX on the validation period, which includes one year of data. As opposed to the other models, two years of data were too much for the model to handle as fit times increased beyond reasonable waiting period. The trained SARIMAX model was then applied to the holdout set to make predictions only once without retraining for the same reason.

2.3.6 NBEATSx

The NBEATSx model is a state-of-the-art neural network architecture with a specific application of time series forecasting (Olivares et al., 2023). The architecture extends the original NBEATS framework by Oreshkin et al. (2019) incorporating exogenous variables, unlocking its ability to model external influences on the target variable. The latter represents a considerable improvement over the original framework. The model can be used in an interpretable manner by decomposing the time series into multiple forecast components, if the proper architecture is chosen.

Block and Stack Structure in the Original NBEATS Model

The core of the NBEATSx model consists of feed-forward fully connected neural networks which learn expansion coefficients for the fitted values of historical and future target values respectively. One or multiple blocks make up a stack, which in turn specialises on a basis functions. Given the vector of target values $\mathbf{y}$, the external regressors $\mathbf{X}$, a lookback window of length L and a forecast horizon of H, the first block b in the first stack s seen in Figure 2.14 learns forecast and backcast expansion coefficients $\boldsymbol{\theta}_{s,b}^{back} \in \mathbb{R}^{N_s}$ and $\boldsymbol{\theta}_{s,b}^{for} \in \mathbb{R}^{N_s}$, where N_s denotes the stack basis's dimension. With the learnt expansion coefficients and the block's basis vectors $\boldsymbol{V}_{s,b}^{back} \in \mathbb{R}^{L \times N_s}$ and $\boldsymbol{V}_{s,b}^{for} \in \mathbb{R}^{H \times N_s}$, the fitted values in the backcast window and the predictions for the forecast horizon window are produced as

$$\hat{\boldsymbol{y}}_{s,b}^{back} = \boldsymbol{V}_{s,b}^{back}\, \boldsymbol{\theta}_{s,b}^{back} \quad \text{and} \quad \hat{\boldsymbol{y}}_{s,b}^{for} = \boldsymbol{V}_{s,b}^{for}\, \boldsymbol{\theta}_{s,b}^{for}$$

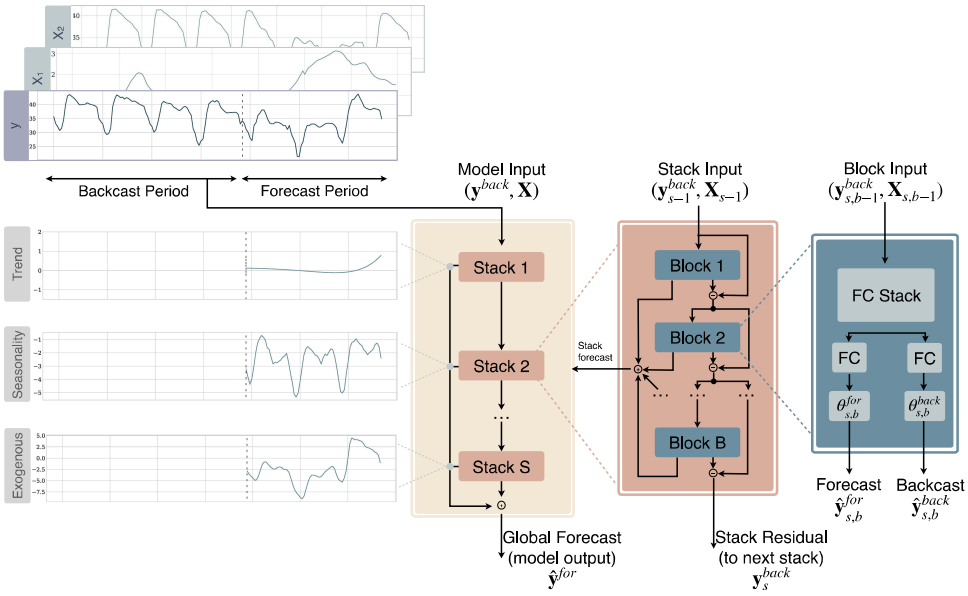


Figure 2.14: NBEATSx architecture (Olivares et al., 2023)

The backcast $\hat{\boldsymbol{y}}_{s,b}^{back}$ is then subtracted from the previous input which was passed into block b, while the resulting residual is passed onto the previous block:

$$\boldsymbol{y}_{s,b+1}^{back} = \boldsymbol{y}_{s,b}^{back} - \hat{\boldsymbol{y}}_{s,b}^{back}$$

This concept of sequentially removing parts from the original full signal can be thought of as a parallel to classical time series decomposition where a time series is broken down into components such as trend, seasonality, and residuals.

At prediction time, the final output of the NBEATS model is the aggregation of all forecast outputs from each stack by sum:

$$\hat{\boldsymbol{y}}^{for} = \sum_{s=1}^{S} \hat{\boldsymbol{y}}_{s}^{for}$$

The integration of exogenous variables $\mathbf{X}$ distinguishes NBEATSx from the original N-BEATS model. They are incorporated as additional inputs next to the backcast window at each block, allowing the model to explicitly account for external drivers, such as weather or calendar effects, which is highly relevant to load forecasting.

Table 2.4: Hyperparameter search space for NBEATSx

Hyperparameter	**Search Space**
Input Size	$\{24, 48, 72, 96, 120, 144, 168\}$
Harmonics	$\{1, 2, \ldots, 10\}$
Polynomials	$\{1, 2, \ldots, 10\}$
Stack types	$\mathcal{S}_3 = \text{Sym}(M)$ where $M = \{\text{trend, seasonality, identity}\}$
Blocks per stack	$\{1, 2, 3, 4, 5\}$ for each stack
Hidden layers per stack	$\{1, 2, \ldots, 5\}$
Units per hidden layer	$\{8, 16, 32, 64, 128, 512\}$
Dropout probability	$[0.0, 0.5]$
Activation function	{ReLU, Tanh, LeakyReLU, Sigmoid}
Learning rate	$[10^{-7}, 10^{-2}]$
Learning rate decays	$\{0, 1, \ldots, 10\}$
Early stopping patience	fixed at 100
Batch size	$\{16, 32, 64, 128, 256\}$

The NBEATSx model maintains interpretability by associating each block with a specific component of the time series, such as trend or seasonality. Depending on the basis function chosen for each block, users are enabled to gain insights into the contributions of different components to the forecast. For instance, for trend and seasonality, a polynomial and a harmonic basis expansion can be set respectively.

Feature Preprocessing, Hyperparameter Tuning and Holdout Evaluation

Feature preprocessing for the NBEATSx model takes the exact same form as for DNN barring the reshaping. The training, validation and holdout datasets preserve their hourly structure with calendar features and holiday dummies are added prior to normalising exogenous variables. Additional detail is omitted to prevent repetition.

Hyperparameter tuning of the model was done the same way as in the ridge, DNN and LightGBM models. Using a training and validation split with one year of data each in chronological order, various hyperparameter combinations were ran over an allocated time budget of 12 hours, as computations are more expensive compared to the multioutput DNN implementation. The search space for each tuned hyperparameter can be seen in Table 2.4. The input size denotes the lookback window in hours, which defines how many previous time steps are used as input features for the model. This hyperparameter was the main inspiration for the implementation of the lookback window in the DNN implementation. The number of harmonics and poly-

nomials varies the complexity of the seasonal and trend components in the model. Harmonics define the number of oscillations used in the seasonal basis expansion using trigonometric functions, while polynomials define the highest degree of the trend basis. The stack types hyperparameter specifies the permutation of components (seasonality, trend, and identity) to be used in the model. The model can stack multiple components in different configurations, with the number of blocks per stack and the number of hidden layers per stack affecting the model's capacity to learn complex relationships in the data. The units per hidden layer hyperparameter determines the number of neurons in each hidden layer which also governs flexibility through network size. Dropout is applied to the model to reduce overfitting, and its probability is tuned to balance model generalisation and capacity. The activation function is a crucial choice for introducing nonlinearity into the model, with various options like ReLU, Tanh, LeakyReLU, and Sigmoid, each having different properties suited for different data patterns. The learning rate and the number of learning rate decays control the optimisation via backpropagation. The learning rate determines the step size and therefore how fast the model learns, while the number of decays allows the learning rate to decrease over time to fine-tune the model as it converges. Early stopping is applied to avoid overfitting, with the model stopping training if the validation loss does not improve for a set number of steps (in this case, fixed at 100). Batch size determines how many samples are processed simultaneously during training. Larger batch sizes can speed up training but may reduce the generalization performance, while smaller batch sizes can result in more stable gradient updates but may require more training iterations.

After determining the best hyperparameter combination empirically on the validation set, it is used to retrain the model on the combined training and validation set. Then, the model is evaluated once on the entire holdout set with a step size of 24 to ensure the logic of making day-ahead predictions. The NBEATSx is fast enough to simulate weekly retraining, therefore it undergoes the same logic of sliding window retraining as outlined with ridge regression in subsection 2.3.2.

2.3.7 Model Aggregation using Bagging

Bagging (Bootstrap Aggregating) is an ensemble technique aimed at improving the predictive performance of models by reducing variance and mitigating the effects of overfitting (Breiman, 1996). In its applications in boosting, it achieves this by aggregating predictions from multiple models, e.g. weak learners, which are usually trained on different subsets of the data. Bagging effectively reduces the variance of predictions, a property arising from the averaging mechanism. In this context, variance reduction occurs be-

cause prediction errors specific to individual models, often referred to as idiosyncratic errors, tend to cancel each other out in the aggregation process if they have very low correlation. This phenomenon results in ensemble predictions that are less sensitive to fluctuations caused by model-specific biases or overfitting tendencies. For this study, five very distinct model classes are trained: Deep Neural Network (DNN), gradient boosting (LightGBM), N-BEATSx, SARIMAX, and Ridge Regression. All of those models have varying degrees of flexibility and entirely different approaches to the same problem. While models like NBEATSx and SARIMAX excel in capturing temporal dependencies, others like LightGBM and Ridge Regression are well-suited for feature-based regression tasks. Combining these diverse models might reveal synergies. Although there exist more sophisticated ensembling methods (Erickson et al., 2020), the approach taken in this study is a very easy one:

$$\hat{y}_i^{\text{BAGGED}} = \frac{1}{|M|} \sum_{m \in M} \hat{y}_i^{(m)},$$

where:

- $\hat{y}_i^{\text{BAGGED}}$: Ensemble prediction for hour i
- M : {DNN, LightGBM, NBEATSx, SARIMAX, Ridge Regression}
- $\hat{y}_i^{(m)}$: Prediction for hour i from model $m \in M$

Chapter 3

Results and Discussion

The results presented in this chapter represent the performance of models trained exclusively on the training and validation data from 01.09.2021 to 31.08.2023 as depicted in Figure 2.7, with hyperparameter confined to those periods. Furthermore, the holdout data was used for final model inference only once to avoid overfitting or hindsight bias. This ensures that the reported metrics reflect a realistic approximation of conditions as they would be in a live test and offer credible insights into model effectiveness. Additionally, the long evaluation horizon of making day-ahead predictions throughout a full year allows for the evaluation of robustness across all major seasonalities: yearly, weekly, and daily.

3.1 Evaluating Performance using Loss Metrics

Table 3.1: Holdout performances using loss metrics

Model	**Retraining**	**RMSE**	**MAE**	**MAPE (%)**
BAGGED	-	250,125.06	172,224.98	2.87
LGBM	Yes	252,853.24	177,004.18	2.92
LGBM	No	258,495.91	182,955.29	3.01
DNN	No	289,389.36	191,460.00	3.20
DNN	Yes	313,062.03	205,231.40	3.43
NBEATSx	No	319,813.00	220,405.47	3.72
NBEATSx	Yes	324,352.78	221,493.00	3.71
SARIMAX	No	369,562.82	236,811.10	3.89
RIDGE	Yes	335,067.88	257,353.52	4.40
RIDGE	No	338,158.51	261,399.10	4.48
NAIVE	-	590,135.64	421,695.81	6.89

3.1.1 General Performance of different Model Classes

The performance of the evaluated models on the holdout period is summarised in Table 3.2. Among the different model classes, the gradient boosting model with LightGBM (LGBM) managed to achieve significant outperformance over all other model classes with a strong edge over the deep learning based models. The performance of the latter, DNN and N-BEATSx, shows variability, but interestingly both lie in the same ballpark. Both DNNs outperforming both NBEATSx models goes to show the importance of feature engineering over network topology. The classical time series model (SARIMAX) showed limitations in predictive accuracy, quite strongly losing out on the deep learning method. Similarly, the RIDGE regression model had relatively higher errors. And to state the obvious: The baseline NAIVE model exhibited the highest error values across all metrics, highlighting the importance of more sophisticated time series modelling approaches to achieve better accuracy in load forecasting.

3.1.2 Impact of Retraining on different Model Classes

Interestingly, the retraining strategy had varying impacts across models. For LGBM, retraining improved loss metrics across the board. Conversely, for the deep learning models DNN and N-BEATSx, retraining slightly degraded the results, suggesting potential overfitting to the training and validation data or unstable training behaviour. Generally, neural networks are more complicated to tune, therefore this result suggests that additional care must be taken to achieve robustness of performance. In particular, this highlights the importance of assessing the trade-off between retraining benefits and the risk of overfitting, especially for more complex models. Future work could focus on ablation studies to make both frameworks more reliable.

3.1.3 Special Mention for Bagging

The ensemble model (BAGGED) achieved the lowest error metrics across all loss functions, including RMSE, MAE, and MAPE. Specifically, the ensemble reduced the RMSE by approximately 1.08% compared to the best-performing individual model (LGBM with retraining). This improvement was obtained by simply calculating an average over the already existing models, hence it is highly recommended to practitioners who often already have several models in place. Furthermore, practitioners are strongly encouraged to train a diverse set of models rather than relying solely on a single method. This approach not only enhances predictive accuracy but also reduces the risk of overfitting to specific patterns in the data. The results clearly highlight that diversity among models is a key factor in successful ensemble strategies, rather than hyperparameter tuning individual models.

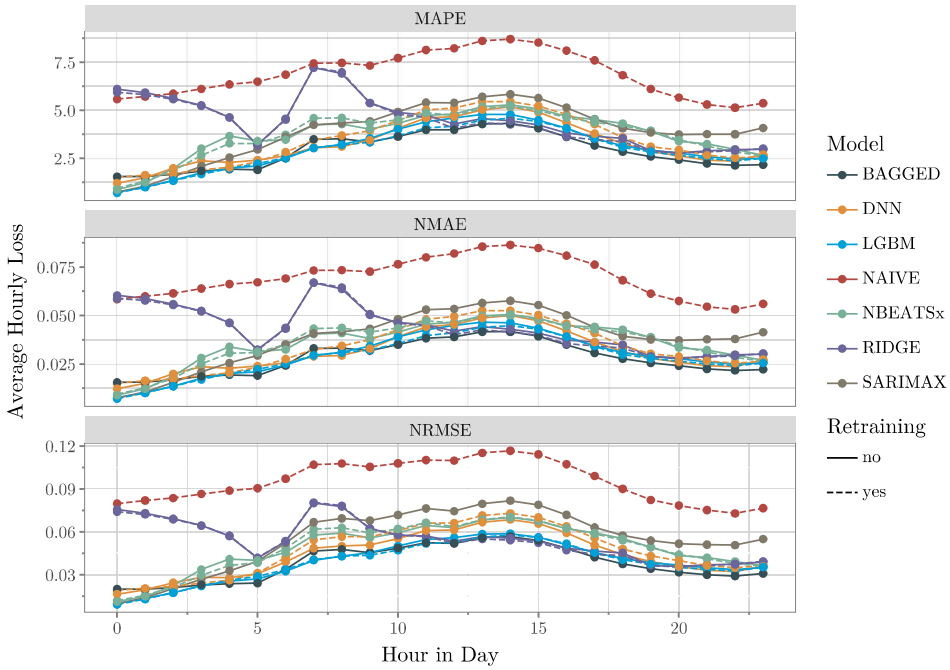


Figure 3.1: Average hourly loss metrics by hour in day for each model, stratified by retraining strategy. Each panel displays a specific metric: normalised root mean squared error (NRMSE), normalised mean absolute error (NMAE), or mean absolute percentage error (MAPE). Metrics are calculated on the holdout set.

Simple averaging was employed in this study, future work could explore more sophisticated ensembling techniques for this specific use case. For instance, weighted averaging based on model performance in a validation set, stacking methods that combine models using a meta-learner, or dynamic ensemble selection based on local data patterns could further improve performance, which is already hinting at section 3.3.

3.1.4 Seasonal Evaluation of different Model Classes

Instead of looking at the entire holdout period, this subsection focuses on yearly, weekly and daily tendencies of each model while still looking at loss metrics. Figure 3.1 shows the normalised equivalents (NRMSE, NMAE, MAPE) loss metrics over each hour in day. Several noteworthy observations can be made:

- Virtually all models outperform the naive benchmark across all hours of the day, which confirms the notion that more sophisticated models significantly improve over seasonal naive methods.

- Ridge Regression demonstrates strong performance after midday but struggles significantly during the early night and morning hours. Given the comparable performance with extremely flexible methods like gradient boosting, it might be worth to specifically target the reason for this, as it will come with the benefits of interpretability.
- The bagged model achieves the best results in the afternoon and evening hours.
- LightGBM consistently outperforms other models during the morning hours.
- N-BEATSx exhibits stable performance, lying firmly in the middle range across all hours without excelling in any specific time window.
- SARIMAX generally performs worse than all other models except during night hours, where Ridge Regression's performance drops notably.

These observations highlight the temporal nuances of model performance within daily windows. Ridge regression's weakness during night-time hours highlights a potential area for improvement. Future research could explore the benefits of combining interpretable models like Ridge with advanced feature engineering techniques to better capture temporal dynamics. Additionally, future studies should investigate more sophisticated ensembling approaches, such as stacking or dynamically selecting models based on their local performance. These methods could further enhance predictive accuracy by leveraging the strengths of individual models in a context-specific manner. For example, using LightGBM for the first and the bagged ensemble for the last half of the day would have outperformed even the bagged algorithm.

We can also look at intra-day effects by weekday type, intra-week and intra-year seasonalities. To keep this section short Figure A.3, Figure A.4 and Figure A.5 can be found in the appendix. Major takeaways from these visualisations are:

Intra-day by Hour of the Day: When evaluating the models by hour of the day and additionally extending by workdays, Saturdays, and Sundays, several distinct patterns emerged:

- SARIMAX struggled primarily during weekdays, particularly during the morning hours. However, on Sundays, it outperformed both the bagged model and LightGBM in the morning hours.
- During Sunday nights, the DNN consistently outperformed the other models, demonstrating its ability to capture patterns in these hours.
- In Sunday afternoons, the bagged model showed the best performance, proving its strength in the later parts of the weekend.

- On workdays, the performance was closely contested, with Ridge regression, the bagged model, and LightGBM competing for first place.

Intra-week by Weekday: Examining model performance across the weekdays $\{1, ..., 7\}$ revealed further insights into the models' behaviour:

- SARIMAX was generally outperformed by all other models on workdays. However, on weekends, SARIMAX performance improves, even outperforming N-BEATSx.
- N-BEATSx performed in the middle range across all weekdays, failing to outperform any model on specific days but demonstrating stable and balanced performance.
- The bagged model and LightGBM were consistently among the best performers across weekdays, often competing at the top.

Intra-year by Month:

- The seasonal naive model performed exceptionally well during June and September, even outperforming Ridge Regression. However, its performance dropped in November and December, where LightGBM worked best.
- N-BEATSx struggled with the many holidays in May. SARIMAX faced similar challenges in April and May, where the numerous holidays interfered with its weekly seasonality approach.
- LightGBM proved to be particularly effective during holiday-heavy months like April and May, handling these periods better than all other models.
- Ridge Regression did not show notable outperformance on an aggregated scale, but it would be interesting to explore its potential if the problem with night hours was fixed.
- LightGBM demonstrated robust performance throughout all months, though the deep learning methods (DNN, NBEATSx) outperformed it June and July.
- DNN showed strong performance, particularly in the holiday-free months, suggesting it is well-suited to capture patterns during this period and that choosing an intelligent weighting pattern based on seasonalities might improve performance by a lot.

3.2 Visualising Predictions

This subsection provides aggregated visualisations of predictions across calendar variables used in the previous section: hour in day, day of the week,

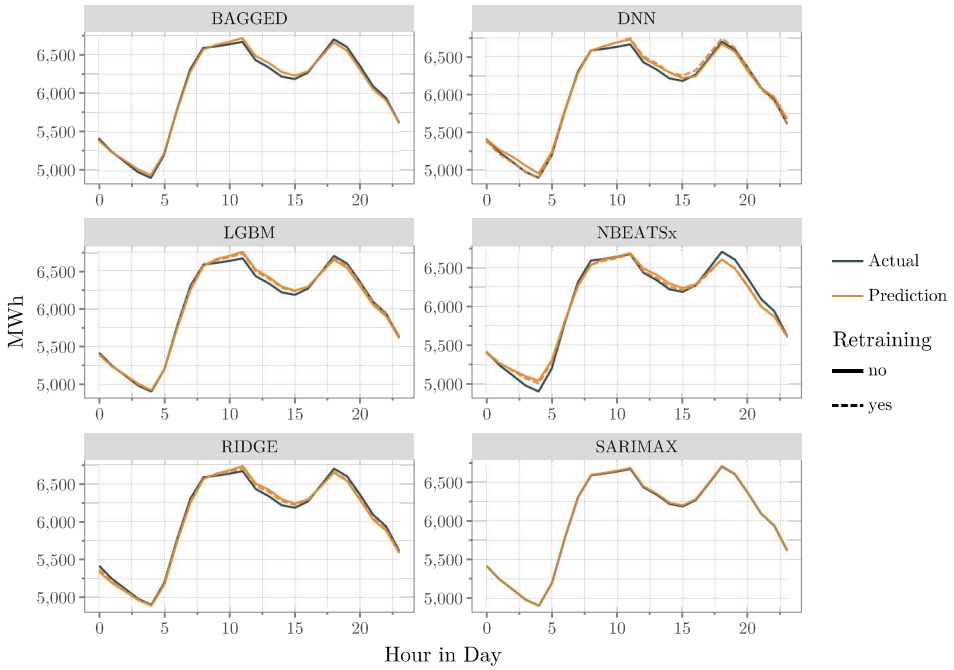


Figure 3.2: Comparison of average hourly predictions by hour in day and actual values for different predictive models, excluding the naive baseline. Predictions and actual values are averaged over all observations in the hold-out period for each x-axis unit.

and month of the year. These are used to identify tendencies of each model to systematically overestimate or underestimate, which forms the basis for evaluating their impact on actual balance costs in the subsequent sections.

By Hour in Day When aggregating predictions by hour of the day (Figure 3.2), SARIMAX demonstrates highly accurate average predictions, aligning closely with observed values. However, for SARIMAX to underperform overall, the variance of its predictions on actual days must be significantly higher than that of other models, given its accurate expected values. Conversely, N-BEATSx consistently underestimates during evening hours, highlighting a systematic bias in its performance. Both the DNN and LightGBM models, along with Ridge regression, exhibit a tendency to overestimate during working hours. This overestimation places the balance group in a long position, particularly during midday hours, which could result in higher imbalance costs.

By Day in Week Examining performance by day of the week (Figure A.8) reveals that the DNN slightly overestimates for most days but underesti-

mates on Mondays, suggesting an overreliance on daily seasonality. Similarly, N-BEATSx underestimates on Mondays and overestimates on weekends, which may also indicate an over-dependence on prior day seasonality. These patterns could stem from suboptimal model training or inherent weaknesses in these model architectures. In contrast, SARIMAX, LightGBM, and Ridge regression maintain a more balanced prediction profile across the week, demonstrating accurate performance without notable systematic biases.

By Month in Year Aggregating predictions by month (Figure A.9) reveals additional patterns. The DNN tends to overestimate energy consumption during the first two quarters (Q1 and Q2) but aligns well with actual values during the latter half of the year. N-BEATSx overestimates during the hot summer months, which is a major point of criticism given the effects of solar production outlined in subsection 1.1.2. Ridge regression, on the other hand, underestimates during winter months, which may reflect its limitations in handling seasonal variations. Both LightGBM and the bagged ensemble method achieve consistent accuracy across all months, with no significant bias observed.

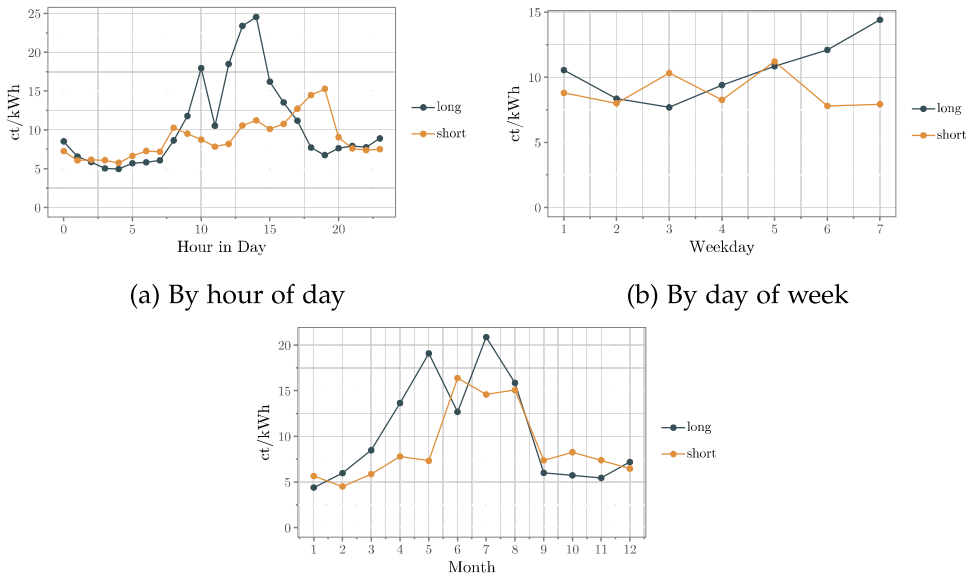
For the interested reader, Figure A.6 and Figure A.7 show visualisations of the actual predictions made by each model without aggregation to averages or loss metrics for both an easier and a more difficult week.

3.3 Evaluating Performance using Balance Costs

Model evaluation in most forecasting applications extends beyond conventional loss metrics by considering the economic implications of forecast inaccuracies. Balance costs serve as an important measure of performance from the perspective of the BGM, quantifying the financial impact of forecast errors due to imbalances in energy supply and demand. As outlined in section 1.1, these costs arise from the exchange of energy with TSO at prices that deviate from day-ahead spot prices. Using the methodology shown in subsection 2.2.3 of considering the price differential between the day-ahead market and the imbalance market, weighted by the magnitude of the imbalance (e.g. in kWh), this section will evaluate model performance by their financial losses on the holdout dataset spanning September 2023 to September 2024.

3.3.1 Empirical Evaluation of Balancing Prices

Hourly analysis in the first diagram in Figure 3.3.1 reveals that long positions are significantly more expensive, with prices reaching up to 25 ct/kWh between 08:00 and 17:00, while short positions become more costly during



(a) By hour of day

(b) By day of week

(c) By month of year

Figure 3.3: Comparison of average balance costs in the holdout period across different time resolutions: (a) hour of day, (b) day of week, and (c) month of year. Each plot shows mean long and short imbalance costs (ct/kWh), highlighting the average cost differences relative to the day-ahead spot price.

the evening hours, particularly right after the workday ends. In terms of weekdays, long positions are notably more expensive on weekends compared to weekdays. By month, long positions are more expensive during the first and second quarters (Q1 and Q2), while short positions are more expensive in the fourth quarter (Q4).

As outlined in Swissgrid (2022), ancillary services operate on a pay-as-bid basis, meaning that larger control energy activation volumes due to grid imbalances result in higher balance energy prices due to the activation of more expensive bids. This leads to skyrocketing balance energy prices, which will result in higher balance costs compared to the spot price, as illustrated in , ceteris paribus. As such, Figure 3.3.1 depicts the average status of the grid during the holdout period, which in itself is a product of the Balance Group Manager's (BGM) decisions, driven by models. Thus, these curves effectively reveal the shortcomings of the models currently used in production in Switzerland on average. Keeping this in mind, one could already make an informed guess about individual model performances with regard to balance costs, when thinking about the individual biases of models shown in section 3.2. For instance, it could be be deduced from the average balance

costs in Figure 3.3.1 that models are:

- Overestimating consumption around noon, which is likely a combined effect with concept drift due to solar production and self-sufficiency outlined in subsection 1.1.2,
- Overestimating weekends due focus on autocorrelation of the previous day instead of weekly seasonality,
- Struggling with the yearly seasonality effect shown by higher long imbalance prices when temperatures are rising again (and actual load is coming down) and vice-versa, higher short imbalance costs for winter months when temperature goes down and actual load comes up.

All of these issues have been shown to exist in virtually all models in section 3.2 when looking at average predictions compared to average actual values. Hence, focussing on these weaknesses in particular would constitute valuable future research. One suggestion already made is using smarter ensembling approaches which leverage strengths of different model classes.

3.3.2 General Performance of different Model Classes

Table 3.2: Holdout performances using balance costs

Model	Retraining	Average	Total	Change over Next (%)
LGBM	No	7471.23	2.63e+08	-1.55
LGBM	Yes	7588.54	2.67e+08	-1.61
BAGGED	-	7712.50	2.71e+08	-7.08
DNN	No	8300.07	2.92e+08	-4.46
SARIMAX	No	8687.70	3.05e+08	-0.80
RIDGE	Yes	8758.19	3.08e+08	-1.33
DNN	Yes	8876.65	3.12e+08	-2.92
RIDGE	No	9144.09	3.21e+08	-3.80
NBEATSx	Yes	9505.56	3.34e+08	-1.02
NBEATSx	No	9603.10	3.37e+08	-37.01
NAIVE	-	15244.47	5.36e+08	0.00

The holdout performances of various models with regard to balance costs are summarised in Table 3.2. The table shows the average incurred hourly balance costs (Average), the total costs over the full year (Total) and the improvement of each model over the next worse in the table, as they are sorted in descending order by average balance costs. Unlike traditional loss metrics, these costs not only depend on the predictive accuracy of each model, but also on on the direction and the timing in which they are wrong, as long

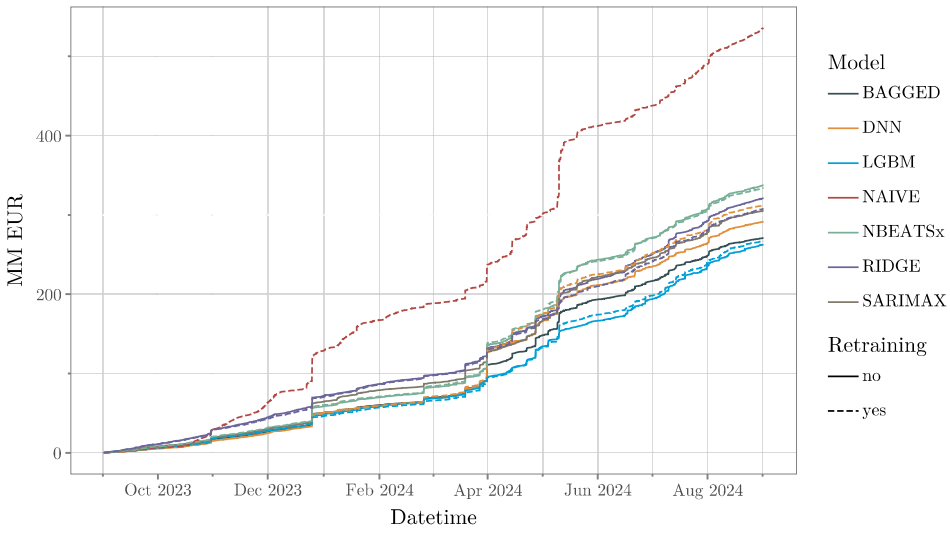
and short balance costs are generally not symmetric. The relationship between predictive biases and balance costs is illustrated in Figure 3.3.1. Long positions are most expensive during midday hours (08:00–17:00), and short positions are costliest in the evenings (after 17:00). These patterns align with the biases observed in model predictions. Models that effectively avoid overestimations during midday, when long prices are highest, or underestimations during evening hours, when short prices peak, generally perform better in terms of balance costs.

The LightGBM models, both with and without retraining, achieved the best balance cost performance, though retraining did not help with balance cost minimisation. It is striking that despite average overestimation during working hours, the balance costs are lower than for all other models, which is a hint at the following section on the impact of holidays. SARIMAX now exhibits much stronger performance, beating both NBEATSx implementations, ridge and the retrained DNN. This is likely due to its balanced prediction profile by hour in day (Figure 3.2), which puts it short more often than the other models, avoiding the costly long positions during working hours seen in other models. N-BEATSx consistently underestimates during evening hours and overestimates during midday. These unfortunate biases puts it in the wrong direction at both times, making it particularly vulnerable to high short costs in the evenings and high long costs around noon. Ridge regression exhibits moderate performance, with a tendency to overestimate consumption during midday hours. Its limitations in capturing nonlinear seasonality effects, such as those during summer and winter, result in poorer performance compared to more advanced models like LightGBM. The seasonal naive benchmark, as expected, performed the worst, incurring significantly higher balance costs. Interestingly, for LightGBM and DNN, retraining worsened balance costs. While retraining can help adapt to variable drift, it may exacerbate existing biases.

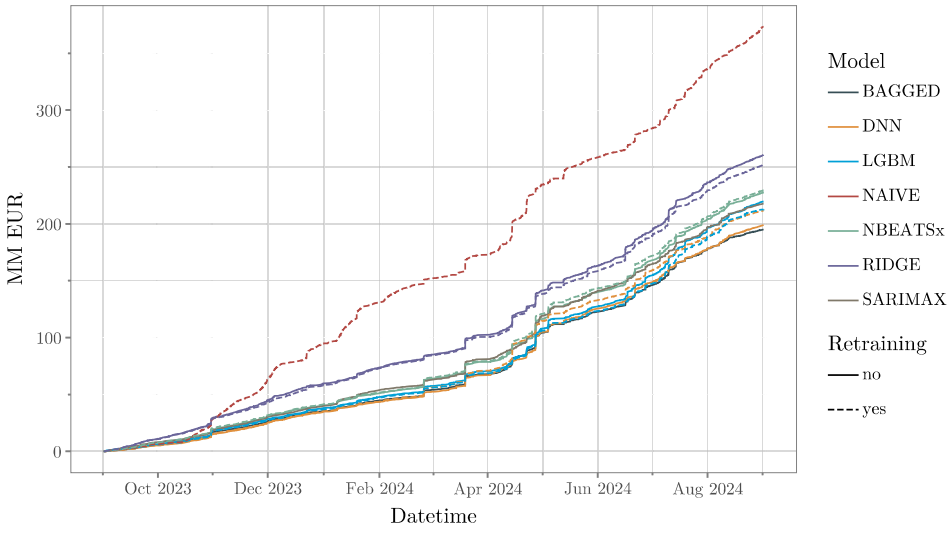
These insights suggest that improving model performance in critical periods (e.g., reducing midday overestimations and addressing evening underestimations) could significantly reduce balance costs. To repeat the previous conclusion: Advanced ensembling strategies that combine the strengths of different model classes offer a promising direction for future research.

3.3.3 Impact of Holidays on Balance Costs

The previous sections left one dimension out of the equation: Holidays. The methods section (2.3) mentioned that dummies for all major Swiss holidays are included as predictors for all models. The question then remains: Are all models equally able to learn these effectively? And the short answer is: Not to the same degree.



(a) Including holidays



(b) Excluding major holidays

Figure 3.4: Cumulative balance costs (in million EUR) for each model, with and without retraining, evaluated over the one-year holdout period. Subfigure (a) includes all days, including major holidays, while subfigure (b) excludes costs associated with major holidays, as identified for the region of Zurich. The comparison highlights the impact of holidays on cumulative cost trends.

Table 3.3: Holdout performances using balance costs without holidays

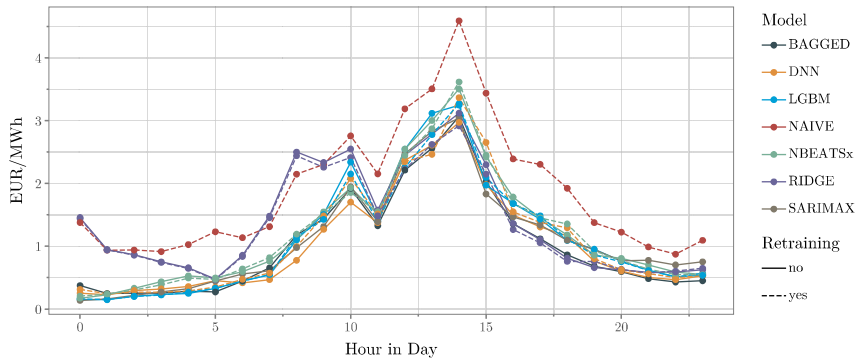
Model	Retraining	Average	Total	Change over Next (%)
BAGGED	-	5,840.28	1.95e+08	-1.92
DNN	No	5,954.69	1.99e+08	-5.94
DNN	Yes	6,330.84	2.12e+08	-0.57
LGBM	Yes	6,366.83	2.13e+08	-2.37
SARIMAX	No	6,521.39	2.18e+08	-0.81
LGBM	No	6,574.86	2.20e+08	-3.61
NBEATSx	No	6,821.43	2.28e+08	-0.67
NBEATSx	Yes	6,867.45	2.29e+08	-8.90
RIDGE	Yes	7,538.36	2.52e+08	-3.35
RIDGE	No	7,799.59	2.61e+08	-30.20
NAIVE	-	11,173.84	3.73e+08	0.00

Figure 3.4a depicts the cumulative balance costs incurred over the full holdout period. Clearly, single days between March and June 2024 have significant impact on the final balance cost in each model. Not coincidentally, the majority of these days are public holidays. Therefore, as a direct comparison to judge the ability of each model to learn the highly different load patterns during holidays, Figure 3.4b shows the the cumulative performance after removing these days entirely from the evaluation period. Additionally, Table 3.3 depicts the final position after summing up all hourly losses over the entire year.

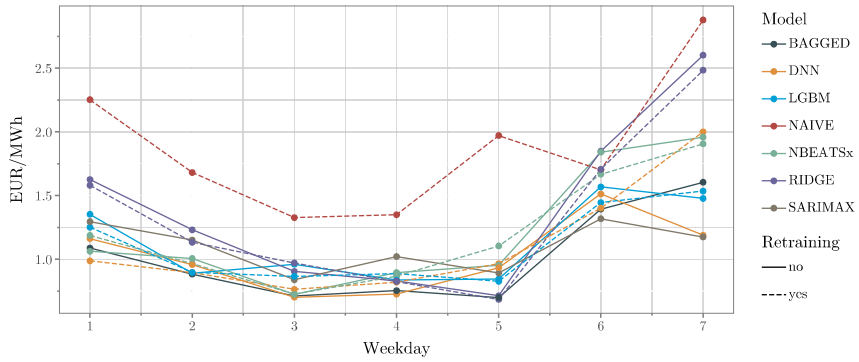
DNNs are highly affected by holidays, as despite providing the holiday dummies, the model did apparently not learn load patterns during these days very well. Due to the reshaping with daily step size, the data available for learning were hardly more than 700 observations. Likely, this does not constitute enough data for the highly flexible model class. Furthermore, the bagged model now improves over all other individual model classes, as seen before with loss metrics. LightGBM with retraining still takes the edge over SARIMAX, though the delta between them has shrunken to around 2.4%. Finally, the NBEATSx models both still place very low, likely due to their unfortunate tendency to stand in the wrong direction on average when it comes to daily profiles and balance cost structures.

3.3.4 Seasonal Evaluation of different Model Classes

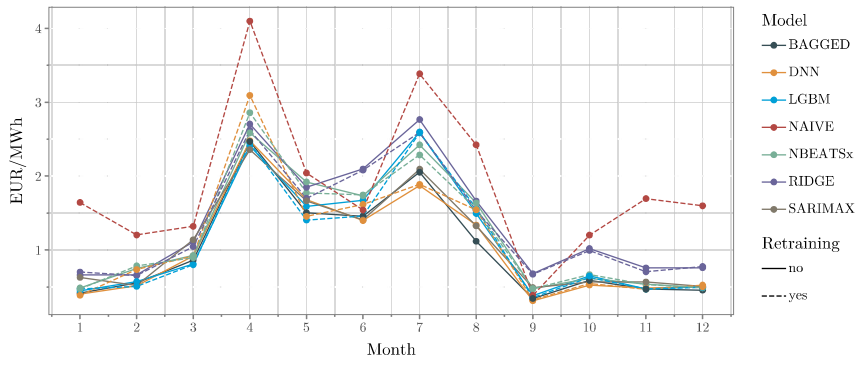
This section evaluates the performance of various model classes across different seasonal dimensions (hour of the day, day of the week, and month of the year) based on balance costs. Similar to the analysis conducted with loss metrics, this approach provides a more detailed understanding of the



(a) By hour in day



(b) By weekday



(c) By month

Figure 3.5: Average hourly balance costs for each model, with and without retraining, over the one-year holdout period. Major holidays have been excluded from the calculation in order to enable the analysis of general tendencies rather than outliers.

predictive tendencies of each model and their implications on balance costs. As major holidays are distorting the model performances by a considerable margin, they have been excluded from the evaluation. This exclusion allows for a clearer comparison of model performance under typical conditions.

By hour of the day, Ridge regression, which performs poorly in terms of loss metrics during nighttime, also incurs higher balance energy costs in these hours, whereas LightGBM emerges as the most effective model during the night. DNN performs best in the morning hours, while SARIMAX and Ridge show strong performance during the afternoons, as already explained by their less biased predictions around these hours. Analysing performance by weekday reveals that DNN and the bagged model excel during workdays, while SARIMAX performs well on weekends, as it avoids the overestimation tendencies of other models. This is due to the significantly higher costs of long positions on weekends compared to weekdays. Although NBEATSx is among the better models during weekdays, it loses out during Fridays and weekends. On a monthly scale, LightGBM demonstrates exceptional performance during the first two quarters, whereas the bagged model and DNN are most effective during the summer months.

These patterns suggest that incorporating custom loss metrics that account for the asymmetry of imbalance costs could enhance performance. Moreover, ensembling models based on their complementary strengths and their biases' negative correlation with long/short balance costs presents a promising strategy for practitioners seeking to minimise imbalance costs effectively.

Chapter 4

Conclusion

The original question of this simulation study was: How do traditional time series models like SARIMAX fare against state-of-the-art deep learning models like NBEATSx in load forecasting? In particular, this study focused on the perspective of Swiss balance group managers who want to avoid forecast errors in the day-ahead in order to save on expensive balancing costs positions with the transmission system operator. Next to the original model classes, several other model classes were introduced as challengers and benchmarks, from simple seasonal naive or ridge regression models, to highly flexible deep neural networks and gradient boosting.

Firstly, the findings indicate that the marginal gains from highly complex and computationally expensive models, such as NBEATSx, given identical input data, do not automatically translate into proportional performance. In this case, the state-of-the-art method did not generalise as well as gradient boosting, which is also easier and faster to operate. While this advanced model has been shown to excel in other energy applications (Olivares et al., 2023), its performance benefits were not extraordinary by any means in this context.

Every model class showed particular over- or underestimation patterns when aggregating by different seasonalities, which might expose balance group managers to vulnerabilities regarding balance cost patterns in the two-price (long/short) balancing system in Switzerland. For example, NBEATSx consistently exhibited misaligned predictions during working hours and evenings, resulting in significant balance cost penalties despite comparable statistical loss metrics as other models. This highlights the importance of selecting models that mitigate such tendencies or incorporating strategies like asymmetric loss functions based on the correct quantification of economic loss, which represent a promising extension of this thesis.

The issue of holidays posed challenges for some models, particularly deep

neural networks, which lacked data quantity after reshaping to the multioutput setting to fully utilise their flexibility. This finding suggests the potential for specialised models trained specifically on holiday data or augmented with synthetic data. Developing such approaches could enhance model performance in capturing atypical consumption patterns, an area that warrants further exploration in the context of this application.

Ensemble methods, here by simple mean aggregation, showed surprising performance improvements. Combining predictions from different models allowed to reduce overall variance without increasing bias disproportionately. However, the ensemble technique used in this study was very simplistic. Future work could focus on more sophisticated strategies, such as stacking models with dynamic weighting based on seasonal performance, as the insights from this thesis underline the importance of incorporating seasonal and contextual factors into load forecasting models. Developing weighted ensemble methods that account for the effective economical cost patterns given time-of-year, day-of-week, or other calendar variables could align outputs more closely with practical objectives, such as minimising costs.

In conclusion, this thesis highlights the value of model diversity, leveraging ensemble approaches to address inherent model biases, and accounting for temporal and contextual variations using appropriate loss metrics shaped by the objective. These findings offer practical guidelines for improving load forecasting accuracy to BGMs while minimising balance costs in the bidirectional imbalance penalty system in Switzerland.

Appendix A

Additional Charts

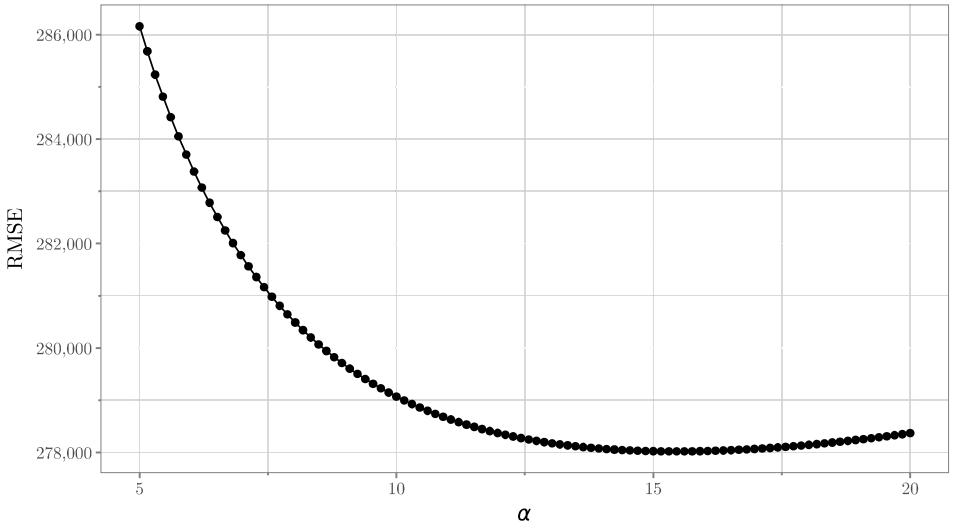


Figure A.1: Effect of the regularisation parameter α on validation RMSE for ridge regression

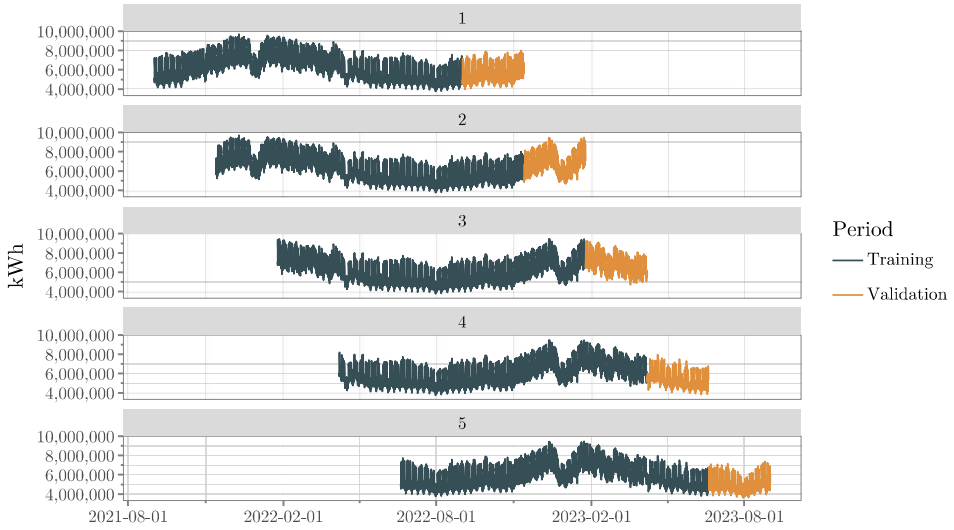


Figure A.2: Time series cross validation strategy for the hyperparameter tuning of ridge regression. Five sliding windows are used in the training and validation period shown in Figure 2.7.

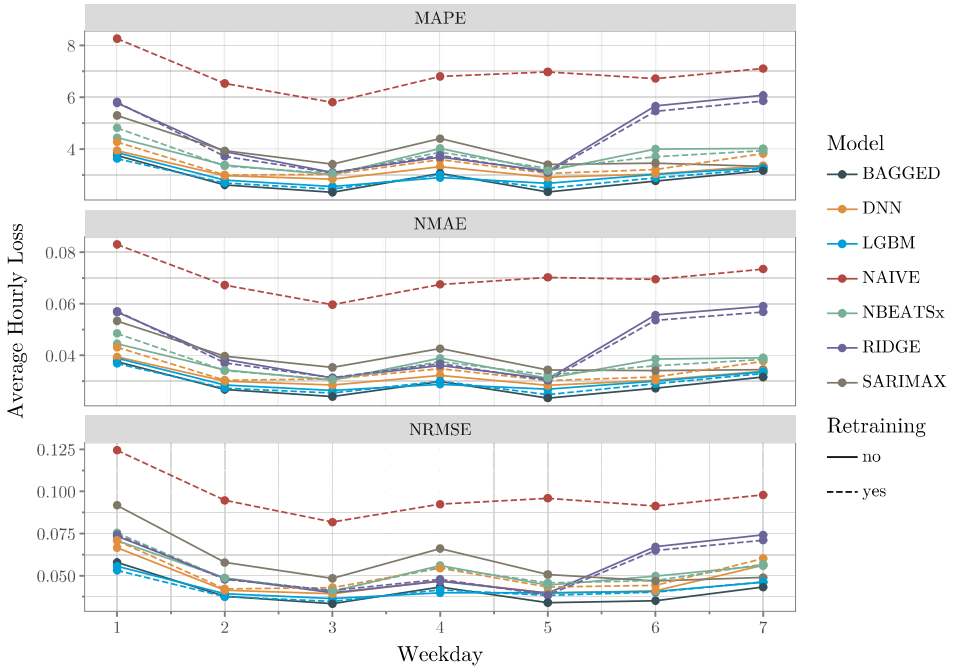


Figure A.3: Average hourly loss metrics by weekday for each model, stratified by retraining strategy. Each panel displays a specific metric: normalised root mean squared error (NRMSE), normalised mean absolute error (NMAE), or mean absolute percentage error (MAPE). Metrics are calculated on the holdout set.

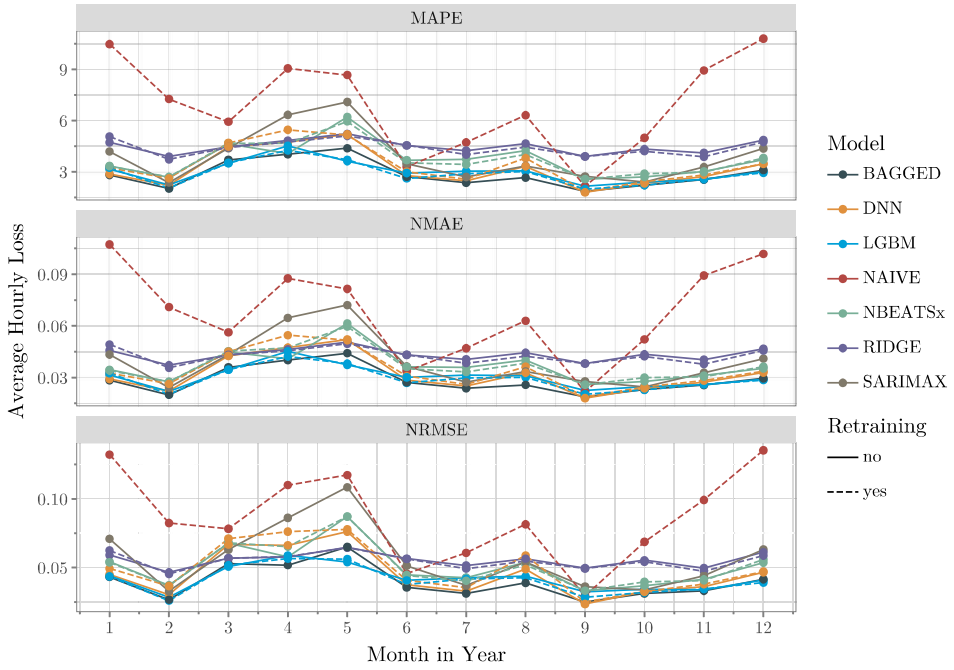


Figure A.4: Average hourly loss metrics by month for each model, stratified by retraining strategy. Each panel displays a specific metric: normalised root mean squared error (NRMSE), normalised mean absolute error (NMAE), or mean absolute percentage error (MAPE). Metrics are calculated on the holdout set.

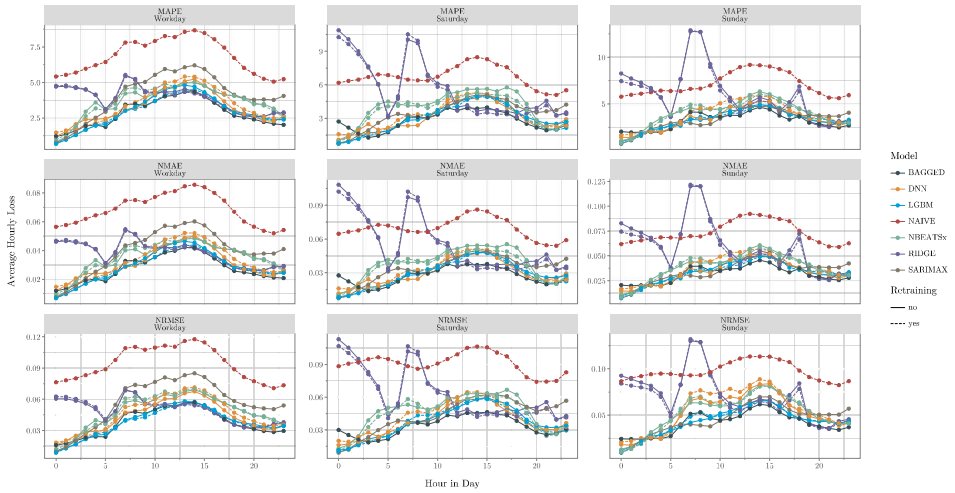


Figure A.5: Average hourly loss metrics by hour in day and weekday for each model, stratified by retraining strategy. Each panel displays a specific metric: normalised root mean squared error (NRMSE), normalised mean absolute error (NMAE), or mean absolute percentage error (MAPE). Metrics are calculated on the holdout set.

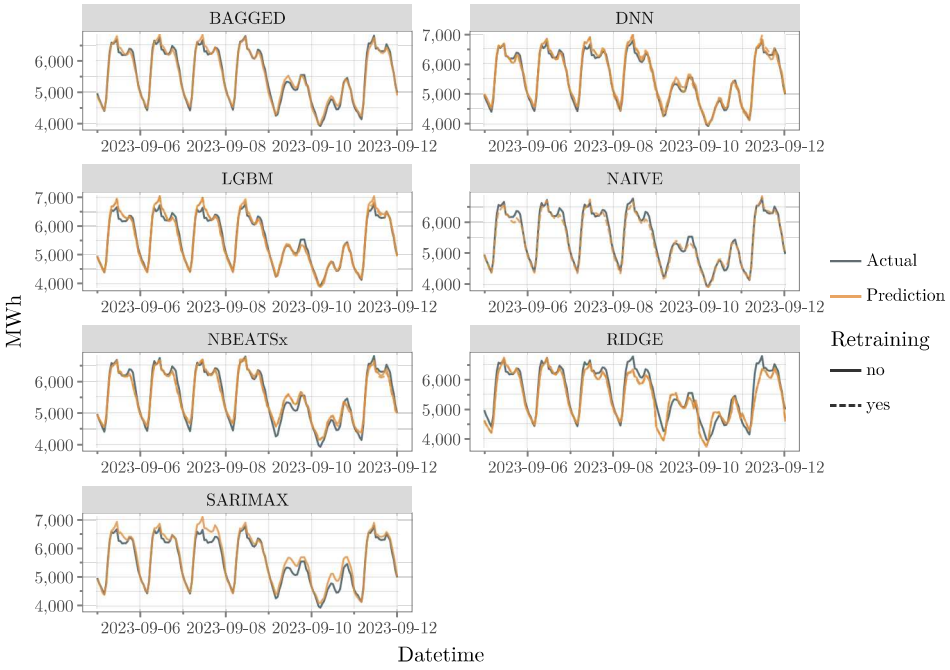


Figure A.6: Holdout predictions on an easy week without holidays. Each panel shows a model class's predictions for the respective time window as well as the ground truth in MWh.

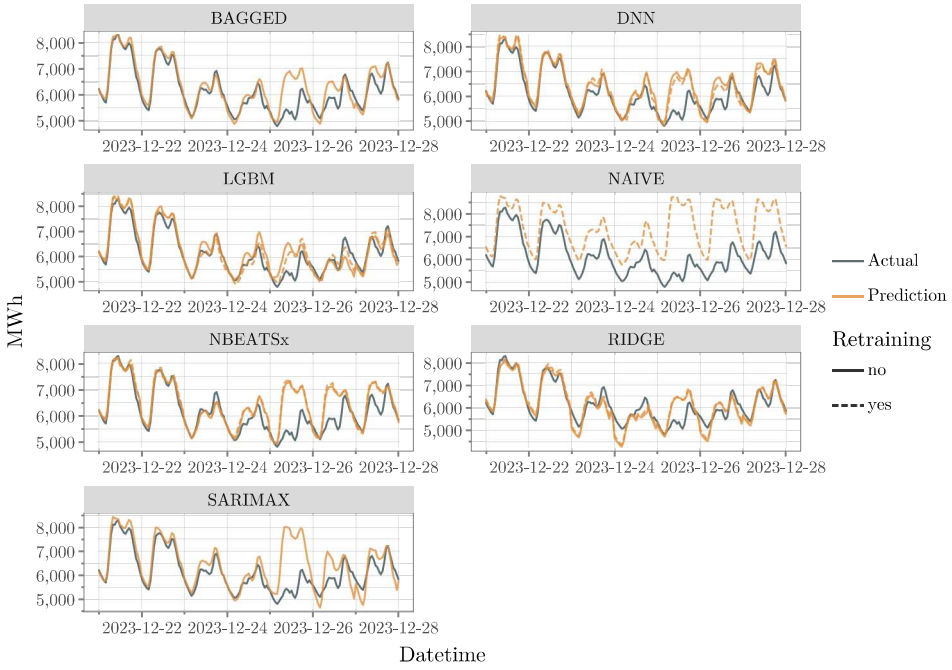


Figure A.7: Holdout predictions on a difficult week (Christmas Period). Each panel shows a model class's predictions for the respective time window as well as the ground truth in MWh.

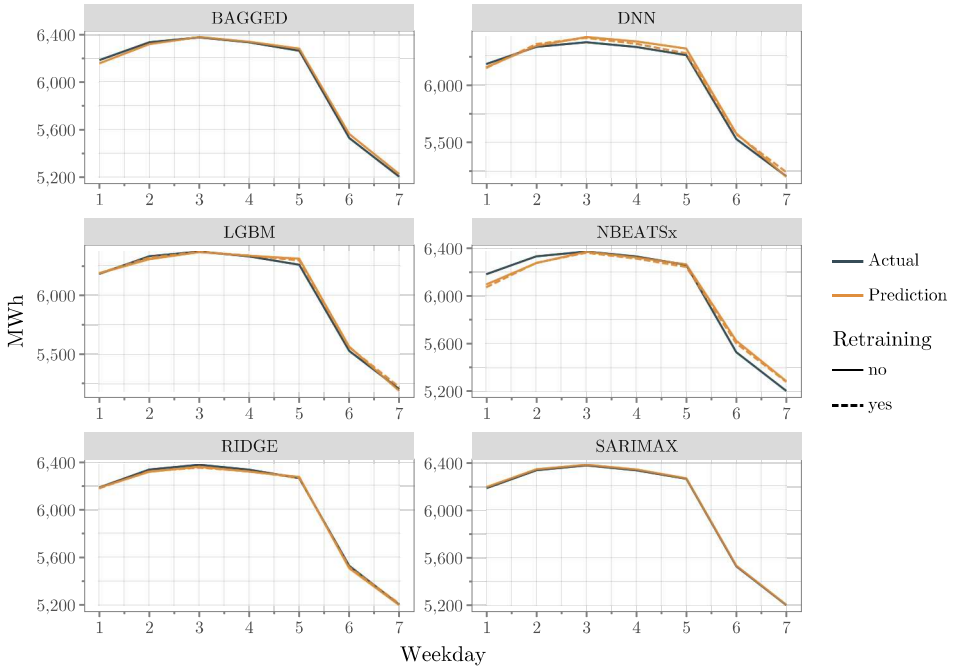


Figure A.8: Comparison of average hourly predictions by weekday and actual values for different predictive models, excluding the naive baseline. Predictions and actual values are averaged over all observations in the hold-out period for each x-axis unit.

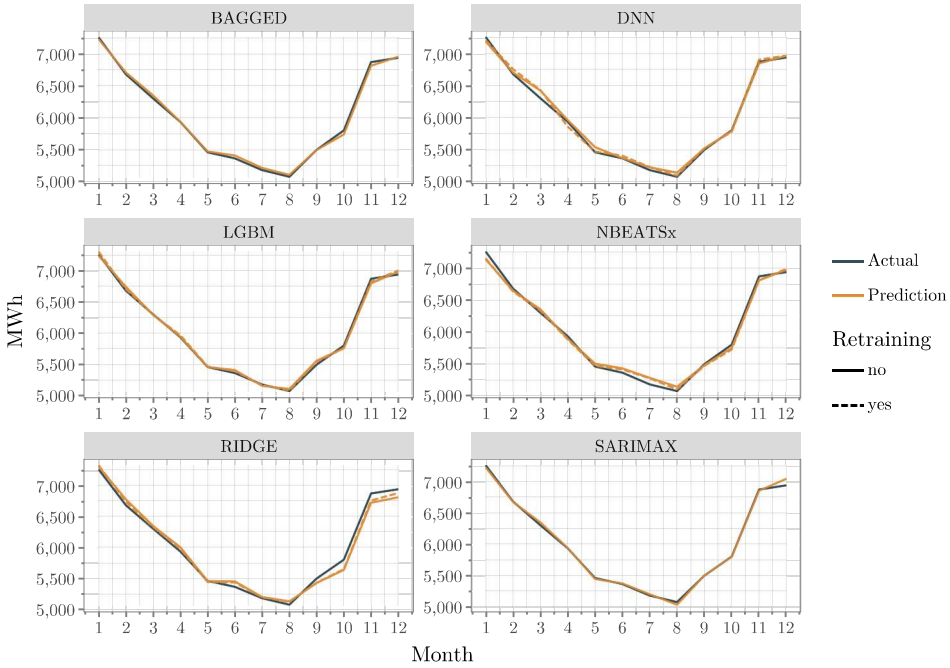


Figure A.9: Comparison of average hourly predictions by month and actual values for different predictive models, excluding the naive baseline. Predictions and actual values are averaged over all observations in the holdout period for each x-axis unit.

Bibliography

Akiba, T., S. Sano, T. Yanase, T. Ohta, and M. Koyama (2019). Optuna: A next-generation hyperparameter optimization framework. *CoRR abs/1907.10902*.

Box, G. E. P. and G. M. Jenkins (1976). *Time Series Analysis: Forecasting and Control* (Revised ed.). San Francisco: Holden-Day.

Breiman, L. (1996). Bagging predictors. *Machine learning 24*, 123–140.

Brockwell, P. J. and R. A. Davis (2016). *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. *CoRR abs/1603.02754*.

Erickson, N., J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola (2020). Autogluon-tabular: Robust and accurate automl for structured data.

Hastie, T. (2009). The elements of statistical learning: data mining, inference, and prediction.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems 30*.

Liaw, R., E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica (2018). Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Olivares, K. G., C. Challu, G. Marcjasz, R. Weron, and A. Dubrawski (2023). Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with nbeatsx. *International Journal of Forecasting 39*(2), 884–900.

Oreshkin, B. N., D. Carpov, N. Chapados, and Y. Bengio (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.

Rosenblatt, F. (1961). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms.

Swissgrid (2020). Tso report on balancing.

Swissgrid (2021). General balance group regulations.

Swissgrid (2022). *Principles of ancillary services products*. Swissgrid. Version 19 from 24.08.2022.

Swissgrid (2024). Aggregated energy data of the control block switzerland. `https://www.swissgrid.ch/en/home/customers/topics/energy-data-ch.html`. Accessed: 2024-10-01.

Vagropoulos, S. I., G. I. Chouliaras, E. G. Kardakos, C. K. Simoglou, and A. G. Bakirtzis (2016). Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting. In *2016 IEEE International Energy Conference (ENERGYCON)*, pp. 1–6.

Zippenfenig, P. (2023). Open-meteo.com weather api.