Beyond Point Forecasts: Quantile, Conformal, and Bootstrap Methods for Electricity Price Prediction

Emma Arussi (VU Student Number: 622537)

supervisor: Assistant prof. De Vlaming Ronald
co-reader: prof. dr. Lucas André

July 16, 2025

Abstract

Electricity price forecasting (EPF) is challenged by sharp volatility, structural breaks, seasonal effects, and strong heteroskedasticity.

In this thesis, I forecast next-day hourly electricity prices in the Dutch Day-Ahead market using XGBoost and two linear benchmark models, each optimized for hyperparameters and feature selection using Pseudo out-of-sample cross-validation (POOS), and model-specific statistical and ML techniques. I evaluate four uncertainty estimation methods: (1) XGBoostLSS for distributional forecasting; (2) Sequential Predictive Conformal Inference (SPCI), which applies sequential quantile inference to residuals; (3) EnbPI, a bootstrap-based conformal method; and (4) the Sieve Bootstrap. Methods 1-3 have shown to (partly) account for non-exchangeability of residuals in time series settings. The latter assumes linear dependence and weak stationarity with fast-decaying autocorrelations and i.i.d.-like innovations—conditions not met in EPF with XGBoost. To empirically relax these assumptions, I apply the Sieve Bootstrap to rolling forecast residuals within a sliding window, approximating local weak stationarity while accounting for residual autocorrelation.

For an average price of 90 EUR/MWh, the online Sieve Bootstrap achieves 80% coverage with 30 EUR/MWh interval width. SPCI produces narrower (16 EUR/MWh) intervals, but with reduced 60% coverage. Point forecasts with XGBoost and direct regression achieve 20–27% sMAPE, statistically outperforming AR benchmarks (19-34% sMAPE, horizon-dependent).

Keywords: electricity price forecasting, uncertainty quantification, XGBoost, conformal prediction, sieve bootstrap, SPCI, EnbPI, probabilistic forecasting

VU

Acknowledgements

This thesis was carried out with the support of many individuals and institutions to whom I am grateful. I would like to thank Alfred Rodenboog at Repowered for providing the opportunity to conduct this research and for allowing me the autonomy to shape it independently. I am also grateful to my thesis supervisor, Ronald de Vlaming, for his critical insights and consistent enthusiasm throughout the project. The Vrije Universiteit Amsterdam deserves recognition for offering a space where a background in molecular biology could be redirected toward econometrics and data science.

I am thankful to my employers and colleagues for their flexibility and understanding during this period. In particular, I appreciate Ricardo del Valle at Bax, and Winand Kok and Hans Luinge at Spiral, for enabling a balance between work and study commitments. I would also like to thank Kees Joosten at Bax NL for generously offering a comfortable and welcoming workspace, despite my work not being within his practical use.

Finally, I acknowledge the support of my partner, my mother, and several close friends, whose presence and encouragement helped sustain focus and joy up until the last day of this project.

Contents

List of Figures

List of Tables

1 Introduction

Electricity is a traded commodity, but unlike storable goods like gold or wheat, it is governed by unique physical and regulatory constraints that make its price highly volatile. It cannot be stored easily or in large quantities, and its supply must match demand in real time due to grid stability requirements. This makes electricity markets particularly sensitive to short-term disruptions. The 2022 war in Ukraine exemplified this vulnerability: the abrupt disruption of gas and power supplies from key regions triggered immediate and dramatic spikes in electricity prices (Hong et al., 2020).

Accurate short-term forecasts are essential for market participants, grid operators, and policymakers to make informed decisions on bidding strategies, load balancing, and risk management. Yet beyond point forecasts, the ability to construct reliable prediction intervals is critical for operational resilience and uncertainty-aware decision-making. Despite this, most electricity price forecasting research remains focused on improving point predictions through statistical or machine learning methods. Probabilistic forecasting is still underdeveloped, and methods that produce calibrated and adaptive prediction intervals are rarely applied or evaluated systematically, as noted by O'Connor et al. (2025). While quantile regression is gaining traction in energy forecasting and conformal prediction is increasingly used in machine learning for producing valid prediction intervals, their systematic application in electricity price forecasting remains limited. Bootstrapping techniques, though common in statistical analysis, are also rarely integrated with modern forecasting models in this domain.

This thesis aims to address this gap by systematically evaluating and extending recent probabilistic forecasting methods for short-term electricity price prediction. The central research question is straightforward: given historical Dutch day-ahead electricity prices and a set of relevant covariates, can I accurately predict prices one day ahead—and quantify the uncertainty around those predictions? To this end, the key contributions of this thesis are fivefold:

First, I implemented the XGBoostLSS model developed by März (2019), a distributional regression approach that estimates both the conditional mean and variance of the target variable. This enables direct probabilistic forecasting under a Gaussian assumption.

Second, I adapted the Sequential Predictive Conformal Inference (SPCI) framework (Xu and Xie, 2021b) to use XGBoost for both point forecasting and residual-based nonconformity scoring via quantile regression. Prior implementations relied on simpler models such as linear regression or Quantile Random Forests. This integration constitutes a novel application of SPCI within an entirely XGBoost-based pipeline.

Third, I applied the online variants of SPCI and Ensemble Neural Bootstrap Prediction Intervals (EnbPI), originally introduced by Xu and Xie (2021b) and Xu and Xie (2021a), to the multistep-ahead *direct forecasting* setting. This represents, to my knowledge, the first application of these online conformal methods to horizon-specific electricity price forecasting—extending their applicability to more realistic operational use cases.

Fourth, building on the adaptiveness of online SPCI and EnbPI, I developed a novel online Sieve Bootstrap method tailored for non-linear forecasting pipelines. Specifically,

I applied the Sieve Bootstrap of (Bühlmann, 1998) to residuals from XGBoost forecasts by fitting autoregressive (AR) models to capture temporal dependence. While the Sieve Bootstrap assumes linear dependence and weak stationarity, I empirically relax these conditions by applying it to rolling residuals within a sliding window, approximating local weak stationarity. This "online" approach allows real-time uncertainty quantification while explicitly incorporating autocorrelation in the residuals—an aspect often ignored by conformal and ensemble bootstrap methods. To my knowledge, this is the first implementation of the Sieve Bootstrap in a non-linear, non-stationary forecasting context with online adaptability.

Fifth, all methods were benchmarked under a unified experimental framework, using the same multivariate feature set and three point forecasters: XGBoost, autoregressive (AR) models, and linear regression with recursive feature elimination. Rolling-window cross-validation was applied across three forecast horizons (14, 24, and 38 hours). Performance was assessed using point forecast metrics (RMSE, aSMAPE, R^2) and probabilistic metrics (empirical coverage and interval width) at the 90% confidence level. The rest of this thesis is organized as follows. Section 2 reviews the literature on electricity price forecasting and uncertainty estimation. Section 3 describes the methodology. Section 4 presents the results and discussion.

2 Literature review

2.1 How day-ahead prices come about, and how to trade with them

The particularities of the Dutch energy market structure are worth explaining due to their direct impact on the forecastability of the energy prices. This is because the structure itself is set up to balance the prices through a semi-regulated yet liberalized framework. Elements of this framework might be recognizable in other countries and sectors, but as a whole, the Dutch system remains unique in its execution.
In the Netherlands, energy trading occurs across multiple interconnected markets, each serving a different time horizon and operational purpose. These include the long-term (forward) market, the day-ahead market (EPEX SPOT), the intra-day market, and the real-time balancing market operated by TenneT. Each of these markets has distinct rules, participants, and settlement mechanisms, necessitating tailored trading strategies and forecasting approaches. All of this basic information necessary to even consider participating in the energy trading market can be found on the websites of TenneT TSO B.V., EPEX SPOT SE, and Netherlands Authority for Consumers and Markets (ACM).

Prices in these markets are not determined by a central authority; instead, they are set through market-based mechanisms such as uniform-price auctions (day-ahead), continuous trading (intra-day), or balancing bids (real-time). Market participants indirectly determine prices through their bids and offers, while EPEX Spot and TenneT facilitate the process under the oversight of the ACM. Thus, although electricity prices are shaped by competition, the overall system is governed by a strict regulatory framework. Author-

ities define the market architecture, enforce operational standards, ensure transparency, and intervene when necessary to maintain grid stability and fair access. This interplay between free-market dynamics and regulatory oversight adds complexity to electricity price forecasting; one cannot fully rely on market stability through regulation, nor the heavy-tailed distributions found in for example, bitcoin trading, or the IT industry´s stocks.

Unlike other commodities or financial assets, electricity cannot be economically stored at scale, necessitating a continuous real-time balance between generation and consumption. This fundamental physical constraint causes electricity prices to be highly sensitive to short-term supply-demand mismatches. Moreover, the electricity market is bounded by the physical grid, which imposes transmission constraints and requires centralized interventions to maintain system stability (Lago et al., 2021).

The increasing share of renewable energy sources further complicates price forecasting. Power generation from wind and solar is inherently weather-dependent and exhibits significant day-to-day variability, making it challenging to predict accurately (Billé et al., 2023). In the context of the EPEX SPOT Day-Ahead market, this variability necessitates accurate forecasts of daily renewable energy output to support reliable price predictions.

While machine learning models may uncover relationships between past generation patterns and prices, these associations often fail to generalize to future conditions due to the high volatility and limited predictability of renewable output. In contrast, other covariates such as load forecasts or fossil fuel prices exhibit more stable and predictable behavior, making them more reliable inputs for short-term price forecasting models.

While REPOWERED has primarily operated within the real time balancing market, the company now aims to expand into the *EPEX SPOT; from now on referred to as Day-Ahead (DA) market*. The Market Clearing Price (MCP) in the DA market — often referred to as the 'electricity price' — is determined through an auction mechanism and is set per bidding zone, which in most of Europe corresponds to national borders. Electricity prices in the DA market are highly volatile and exhibit multiple seasonalities (weekly, dayly, monthly). The rise in renewable energy as described before, also effects the DA MCP significantly. In the Netherlands, the DA market is operated by EPEX Spot, where an automated market algorithm determines the MCP, which can range from -€500 to €3000 per MWh, reflecting both surplus and scarcity conditions. Market participants place bids directly on the EPEX platform, aiming to buy electricity when prices are low and sell when prices are high. In the DA market, bids are submitted one day in advance for the following 24 hours., and the MCP is set equally for everyone, regardless of the hight of the bid. Strategic selling, therefore, involves submitting offers low enough to ensure execution, and vice-versa for selling. Because most individuals and even many companies lack the tools or expertise to accurately forecast electricity prices day-to-day, a growing market has emerged for companies like REPOWERED, which offer automated bidding and portfolio optimization services. These services rely heavily on forecasting models rooted in econometrics and increasingly leverage ML methods.

2.2 Electricity Price Forecasting (EPF)

2.2.1 Models used for EPF

Weron (2014) described the most used modelling techniques for EPF in their review paper, indicating 6 categories; multiagent models, fundamental (structural) methods, reduced-form models, statistical models, and lastly artificial intelligence based models (which are typically non-parametric, and able to capture non-linearity). Although the paper is written in 2014, the same taxonomy still applies. A notable extension of these categories would be the introduction of hybrid solutions, in which two or more of the categories above are combined. The first category—multi-agent models—is primarily used to analyze market behavior and strategic interactions. While valuable for exploring qualitative aspects of market dynamics, these models are less suited for electricity price forecasting (EPF), which typically requires accurate quantitative predictions.The second class—fundamental models—relies on functional relationships between electricity prices and their underlying drivers, which are modeled independently. However, these models are typically designed to capture longer-term structural relationships, making them more suitable for medium-term forecasting than for short-term electricity price forecasting (EPF). The third class, described as reduced-form models, inspired from finance and price dynamic, is concerned with replicating the main characteristics of daily electricity prices, such as marginal distributions, price dynamics and correlations. These models are generally good at describing important factors of prices such as volatility, but perform less well forecasting short term prices accurately. The last 3 categories, described as statistical models, artificial intelligence models and hybrid models, are until this day the most researched, and (maybe partly because of this) the most successful at predicting energy prices. For example (Lago et al., 2021) report that statistical and artificial intellegence models can achieve an symmetric mean absolute percentage error (sMAPE) of 12,5 and 25%, respectively, on the European power exchange (EPEX)-Belgium, in the period from 01/01/2010 to 31/11/2016.

2.2.2 Statistical Models

Weron (2014) named many statistical models and noticed that many are still used for benchmarking the newer applications in EPF, which are more concerned with ML and hybrid techniques. Models named are for example the AR-type time series models, of which forecasting can be conducted via the Durbin–Levinson algorithm or through the Kalman Filter. Also they named the classical regression models, in which multiple regression assumes the relationship between the variables is linear. In the presence of spikes, however, statistical methods perform rather poorly. This is especially true for price-only models, but models with fundamental variables do not perform well either. Kapoor and Wichitaksorn (2023) present an empirical comparison between statistical and machine learning models, including XGBoost, for daily electricity price forecasting in the New Zealand electricity market. Their study demonstrates that while machine learning models such as XGBoost, LSTM (Long-Short Term Memory), and GRU (Gated Recurrent Unit) were included, traditional statistical models like GARCH and SV—with t-distribution variants

and feature selection techniques such as LASSO—often outperformed the machine learning approaches. However, their focus remains primarily on daily average price prediction, which is essentially a one-step-ahead forecast, whereas this thesis addresses short-term intraday forecasting over multiple horizons (14, 24 and 38 hourly step ahead forecasts). While Kapoor and Wichitaksorn (2023) did incorporate rolling window cross-validation and hyperparameter optimization, similar to the procedures applied in this study, the granularity of their tuning was not adapted to very short forecast horizons or prediction on an hourly basis.

It is important to mention that statistical models do not limit themselves to linear relationships. Billé et al. (2023) explore a total of 58 linear and 129 nonlinear specifications to evaluate forecasting performance in the Italian Day-Ahead market. They conclude that statistical models can perform very well, but only if the forecast values of exogenous variables are reliable, volatility is estimated within the model, the model is updated frequently using a rolling window approach, and feature selection techniques such as LASSO are applied prior to model fitting. Huang et al. (2024) point out that statistical methods often rely on the stability of time series data and may either fail to capture the nonlinear dynamics of electricity prices or require prior knowledge about the relationships between prices and input variables to appropriately specify the form of non-linearity. In this thesis, nonlinear models are optimized through feature selection and frequent re-estimation using a rolling window approach, enabling a fair comparison with machine learning methods.

2.2.3 Machine Learning (ML) models

A comprehensive overview of recent machine learning approaches to short-term electricity price forecasting is provided by Laitsos et al. (2024). The authors summarize several notable contributions in the field. For example, a three-stage model using ensemble empirical mode decomposition (EEMD) and extreme learning machine (ELM) was proposed for short-term electricity price forecasting. Another study introduced a probabilistic forecasting framework combining SHAP-based feature selection with a long- and short-term time series network (LSTNet) using quantile regression. They also report Artificial neural networks, with findings suggesting that architectures combining fully connected layers with recurrent or temporal convolutional layers perform best. Building on this, an STL-TCN-NBEATS model—integrating trend decomposition, temporal convolutional networks, and neural basis expansion analysis—was proposed to further improve forecast precision. Finally, a novel hybrid algorithm integrating linear regression with automatic relevance determination (ARD) and ensemble bagging using extra tree regression (ETR) was presented. Overall, a broader trend in the literature was concluded by the reviewing authors: Applying machine learning methods for efficiency and accuracy, whilst leveraging the structural and explainability qualities of statistical frameworks.

Abbasimehr et al. (2023) propose a two-stage forecasting framework where statistical features are first extracted from time series data and selected using an XGBoost regression, before being fed into deep learning models based on Temporal Convolutional Networks (TCN) and Multi-head Attention (MhA) mechanisms. Their work makes use

of the XGBoost feature importance functionality, but leaves the forecasting effort to a more time-series attuned ML model. Furthermore, the study is performed on monthly-interval energy consumption, which is a lot more predictable and stable time series than hourly energy prices. Moreover, while hyperparameter optimization is performed, the validation framework does not explicitly address the challenges posed by short-horizon, hourly predictions across multiple forecast horizons, as is the case in this thesis. Also, no cross-validation was performed. Zhang et al. (2021) apply XGBoost to the problem of short-term sales volume forecasting for retail stores, combining historical transaction data with external variables such as weather and temperature. Whereas their target variable is considerably different than output prices, the use of historical and external variables, as well as short (15-min) forecast intervals does resemble this thesis. However, unfortunately, the study does not disclose the specific forecast horizon, and does not make use of cross-validation either. Maleki et al. (2024) investigate citywide power consumption forecasting using aggregated data from commercial buildings, employing several models including Random Forest, XGBoost, SARIMAX, Facebook Prophet, and a Convolutional Neural Network (CNN). Their approach incorporates external features such as electricity price and temperature alongside calendar-based variables, and targets a 24-hour ahead hourly forecasting horizon. The study demonstrates the relative performance of different modeling techniques, with CNNs achieving the highest accuracy overall. While XGBoost is included as a predictive model, also no cross-validation was performed here, and the forecasting problem focuses on aggregate power demand rather than electricity price dynamics. This thesis differs by addressing multiple forecast horizons (14, 24 and 38 hours ahead) within a rolling validation framework and specifically tailoring the modeling to the challenges of electricity price forecasting. The paper of Tightiz et al. (2024) actually first used load to make load predictions, to feed into the XGBoost price forecast. They also included holidays and other features. They state that short-term variables that affect the price of electric energy include temperature, weather, holidays, and consumption; long-term factors that affect the price of electric energy include social and economic issues as well as population growth. Forecasting the price of electricity and load is crucial for planning and effective power system operation because, in general, the price of electricity fluctuates over the course of hours, days, weeks, months, and even years.

2.2.4 The model of choice: XGBoost

XGBoost is a powerful gradient boosting algorithm widely used in applied machine learning, with nearly 70% of Kaggle competition winners leveraging it. Known for its efficiency and strong predictive performance, XGBoost has proven its reliability in prominent challenges like the Netflix Prize, Kaggle competitions, and the KDD Cup. It is delivered as a structured, efficient package and has become the de facto choice for ensemble methods in applied machine learning (XGBoost Developers, 2022; Chen and Guestrin, 2016). The exact mechanics of the algorithm—including its loss function, regularization strategies, and optimization procedure—are detailed in the methodology section, but in short, XGBoost fundamentally extends the concept of gradient tree boosting as originally introduced by Friedman (2001), and can be applied to both regression and classification

problems. For EPF, limited peer-reviewed/published sources are available which indicate the use of XGBoost. However, the use of XGboost for forecasts is popularized in open-sourced platforms as Hugging Face and Github, and private sector companies such as REPOWERED.

2.2.5 Hyperparameter tuning

Hyperparameter tuning plays a crucial role in improving model performance across electricity price forecasting tasks. For complex, nonlinear models such as XGBoost, Bayesian optimization is often used to efficiently search high-dimensional, non-convex parameter spaces. Packages like Optuna and Hyperopt offer automated solutions that integrate well with cross-validation procedures to optimize performance metrics such as the root mean squared error (RMSE) (Chen et al., 2024). However, one downside of this approach is the significant computational overhead: hyperparameter tuning typically requires repeatedly training the same model across multiple parameter combinations, and when combined with K-fold cross-validation, the training process is repeated K times for each set of parameters. This can substantially slow down model development and prediction workflows.

XGBoost exposes a wide range of hyperparameters that govern model complexity and regularization, including maximum tree depth, number of estimators, subsample fractions, and minimum child weight, among others (Chen et al., 2024). The choice of objective function also depends on the forecasting goal. In point forecasting tasks using linear models such as AR or OLS, model selection is often guided by information criteria like the Bayesian Information Criterion (BIC), which penalize overfitting while favoring parsimony. In contrast, machine learning models typically minimize prediction error metrics such as RMSE. For probabilistic forecasting, where the goal is to estimate entire predictive distributions, loss functions like the negative log-likelihood (e.g., for scale and shape parameters) or the continuous ranked probability score (CRPS) are more appropriate, as used in the development of the probabilistic XGBoost varient of März (2019).

2.2.6 Time series cross-validation

Time series cross-validation has become an integral component of forecasting with machine learning models, as it enables more realistic performance evaluation, informed hyperparameter tuning, and ongoing model adaptation to new data. This retraining process helps models generalize more effectively to unseen future observations and maintain robust predictive performance over time. However, because the chosen cross-validation method can significantly influence performance estimates, it is essential to select an approach that aligns with the forecasting objective and the structure of the data.

Goulet Coulombe et al. (2019) discuss two primary methods for cross-validation in time series: pseudo out-of-sample (POOS-CV) with rolling-origin recalibration, and the standard K-fold cross-validation. The POOS-CV method respects the temporal order of observations by ensuring that future data is never used to predict the past. This is achieved by training the model on a window of data that shifts forward through time.

The window can be fixed or expanding, depending on whether the training size remains constant or increases over time. This method is especially suited to non-stationary time series, where model performance may vary across different periods.

In contrast, K-fold cross-validation randomly splits the data into K equally sized folds, using $K-1$ folds for training and one fold for testing, repeated K times. While this approach allows for the full dataset to be used for evaluation, it does not account for the temporal dependence inherent in time series data. Nonetheless, Bergmeir et al. (2018) show that standard K-fold cross-validation can still be valid for autoregressive models under the condition that the model residuals are uncorrelated.

In the context of XGBoost, which is not inherently autoregressive and does not assume independent and identically distributed (IID) residuals, the use of POOS-CV is generally preferable. This method reduces the risk of overfitting and prevents data leakage by respecting the chronological structure of the time series during model evaluation and tuning.

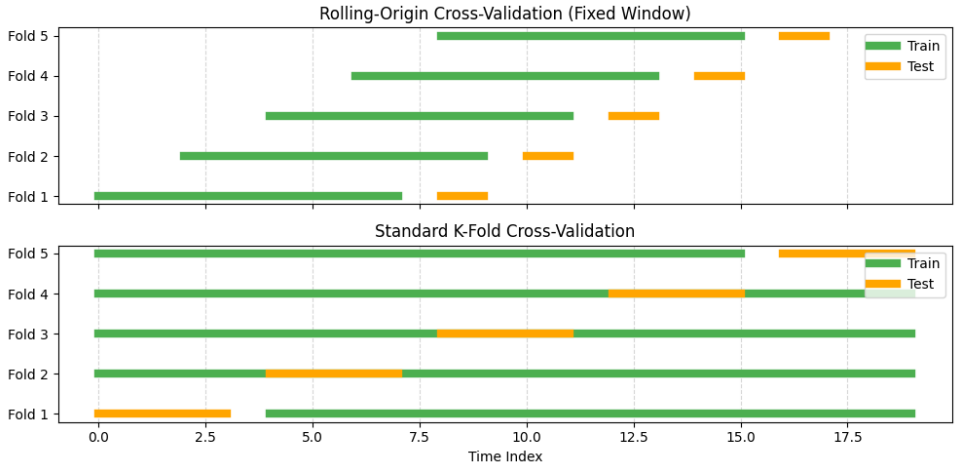

Figure 1: Comparison of Pseudo Out-of-Sample (POOS) and K-fold cross-validation.
Note: Insights are from Bergmeir et al. (2018), but this figure is designed for this thesis.

2.2.7 Uncertainty estimation in time series forecasting

Forecasting models in energy markets, especially electricity price forecasting, often focus on predicting point estimates—typically the conditional mean $\mathbb{E}(Y|X=x)$. However, many real-world applications require knowledge of the full predictive distribution, particularly under conditions of volatility, asymmetry, or structural breaks. In such cases, accurate and well-calibrated prediction intervals can offer more actionable insights than point forecasts alone. This is especially important in domains like energy forecasting, where decision-making under uncertainty (e.g., bidding strategies, risk hedging) relies on reliable interval estimates.

Machine learning models, including XGBoost, are powerful point predictors but often lack principled mechanisms for uncertainty quantification. As noted by Hothorn et al. (2012), many regression models assume that higher moments of the distribution are independent of the input features—a strong assumption that fails in heteroskedastic or

skewed environments. Indeed, when such assumptions are violated, standard approaches to interval construction—like $\hat{y} \pm z_{1-\alpha/2} \cdot \hat{\sigma}$—become invalid. XGBoost, for instance, implicitly assumes homoskedasticity but does not estimate the predictive variance. Out-of-sample, these models often overestimate coverage, especially for high-volatility targets like electricity prices.

Methods for uncertainty estimation can be broadly catagorized into parametric and non-parameteric approaches. One parametric approach is for example Bayesian models, which quantify uncertainty by placing prior distributions over model parameters and updating these priors with data to obtain posterior distributions. This allows for coherent propagation of both parameter (epistemic) and data (aleatoric) uncertainty. Bampoulas et al. (2023) apply a multitask Bayesian neural network to forecast building electricity loads and quantify both types of uncertainty. Their approach explicitly models uncertainty through the posterior predictive distribution, enabling probabilistic estimates of demand flexibility. While effective in capturing structured uncertainty in load forecasting, Bayesian neural networks are computationally intensive and difficult to scale for short-term electricity price forecasting with high temporal resolution.

Quantile regression and extreme value theory. Standard quantile regression estimates conditional quantiles using the pinball loss, but tends to underperform in the tails, especially when extrapolating beyond observed data. To address this, Velthoen et al. (2023) propose GBEX, a parametric method that models the upper tail using a Generalized Pareto Distribution (GPD) with covariate-dependent parameters. These parameters are estimated via gradient boosting, combining the flexibility of tree ensembles with the extrapolative power of extreme value theory. Unlike standard quantile regression, GBEX is designed specifically for accurate estimation of extreme conditional quantiles. GBEX is not used in this study, as the forecasting framework centers on direct interval construction methods that do not assume a specific tail distribution.

Distributional forecasting through XGBoostLSS. Another parametric approach to probabilistic forecasting is to model the full predictive distribution directly. März (2019) extend the XGBoost framework using Generalized Additive Models for Location, Scale, and Shape (GAMLSS) (Rigby and Stasinopoulos, 2005), resulting in the XGBoostLSS model. Unlike standard XGBoost, which only estimates the conditional mean, XGBoostLSS predicts additional distributional parameters—such as scale (variance) and shape (skewness or kurtosis)—as functions of the input features. Prediction intervals are then derived by minimizing the negative log-likelihood (NLL) of the assumed distribution.

XGBoostLSS offers several advantages: it produces variable-width intervals that adapt to the feature space and avoids the need for a separate calibration set. However, these benefits come at the cost of increased complexity. The model must be trained in a two-step iterative process, is sensitive to outliers, and relies on parametric assumptions about the target distribution. Moreover, jointly estimating multiple distributional parameters increases the model's degrees of freedom, heightening the risk of overfitting, particularly for scale and shape. Careful preprocessing (e.g., winsorization or robust scaling) and regularization techniques (e.g., early stopping) can help to ensure stability and generalization

(Kuhn and Johnson, 2013; James et al., 2013).

Conformal Prediction (CP). In contrast, conformal prediction offers a non-parametric, distribution-free framework for constructing prediction intervals. Instead of estimating distributional parameters, it uses residuals on a held-out calibration set to construct intervals with guaranteed coverage under the assumption of exchangeable data (Romano et al., 2019). While this assumption is problematic in time series settings (due to temporal dependence), conformal prediction still provides a practical and widely used uncertainty estimation tool. It wraps around any point forecasting model—including XGBoost—and is particularly attractive for its flexibility and minimal assumptions.

However, conformal methods also come with limitations. Standard split conformal prediction produces fixed-width intervals that do not adapt to local uncertainty, and it requires sacrificing a portion of the data for calibration—potentially degrading model performance in time series tasks with limited training data. Furthermore, coverage guarantees may fail when temporal dependence or regime shifts are present, as is often the case in electricity markets.

Conformalized Quantile Regression (CQR). To address some of these limitations, Romano et al. (2019) proposed Conformalized Quantile Regression (CQR), which combines conformal prediction with quantile regression. CQR constructs feature-dependent, variable-width intervals by estimating conditional quantiles (e.g., 10th and 90th percentiles) using quantile regression and correcting them post hoc using conformal calibration. In this approach, essentially a forecast model is made for each quantile of interest. This hybrid approach retains the data-driven coverage guarantees of conformal prediction while capturing feature-dependent heteroskedasticity. Still, like other conformal methods, it requires a held-out calibration set and assumes exchangeability, which is often violated in sequential data.

Ensemble Batch Prediction Intervals (EnbPI). Another promising direction is Ensemble Batch Prediction Intervals (EnbPI), which was introduced by Xu and Xie (2021a) and further explored in the energy domain by O'Connor et al. (2025). EnbPI constructs prediction intervals by applying conformal calibration methods across an ensemble of models, using either block bootstrap to preserve local dependencies and address seasonality or trend components, leave one out (LOO) bootstrap estimation to make maximum use of the length of the train data. However, EnbPI primarily captures distributional uncertainty and does not model temporal autocorrelation in residuals. The original paper by Xu and Xie (2021a), however, states that EnbPI relies on the assumptions of a linear relationship between y_t and $f(x_t)$, as well as exchangeability of residuals. These assumptions can be violated in the context of electricity price forecasting, particularly under temporal dependence or distributional shifts. As a result, EnbPI may produce overly wide prediction intervals if the underlying point forecaster fails to adequately capture such dynamics.

Sequential Predictive Conformal Inference (SPCI). SPCI, developed by Xu and Xie (2021b), extends conformal prediction by modeling the conformity scores—specifically, the residuals—using quantile regression. Unlike Romano et al. (2019), who estimate the 5th and 95th conditional quantiles directly via pinball loss and then apply conformal calibration to adjust interval coverage, SPCI first extracts past residuals from a point predictor and then fits a quantile regression model to forecast their conditional quantiles. The main novelty is the time-adaptiveness reestimation of quantiles, leveraging some of the temporal dependency among residuals. It also maintains the main quality of conformal prediction, which is that under exchangeability, no assumptions about the data-generating process have to be made, as well as no assumptions on the quality of estimation by the point estimator. According to O'Connor et al. (2025), SPCI often results in narrower confidence bounds compared to ENbPI, without sacrificing coverage.

Sieve Bootstrap Prediction Intervals. To specifically address the autocorrelation structure in residuals arising from electricity price forecasting, I apply the Sieve Bootstrap. This is particularly relevant since nonparametric prediction interval methods rely entirely on residual structure, while XGBoost does not explicitly model temporal dependence. Originally introduced by Bühlmann (1998), this method fits a high-order autoregressive (AR) model to residuals and generates synthetic series through resampling, preserving temporal dependence. Prediction intervals are then constructed from the empirical quantiles of these bootstrapped series. Because the Sieves Bootstrap does not require a calibration set, it is particularly useful when training data is limited.

The Sieve Bootstrap has previously been applied in various modeling contexts. For example, it was used to construct the individual trees within a Random Forest framework (Fokam et al., 2024). In other settings, the Autoregressive Sieve (Wild) Bootstrap has been employed to construct confidence intervals for parametric trend estimation, while the Sieve Bootstrap has been applied to time-varying coefficient regression models using nonparametric local linear estimators Friedrich and Huber (2024); Friedrich et al. (2020). To my knowledge, no attempt has been made to apply the Sieve Bootstrap for confidence interval estimation in a machine learning forecasting context.

The classical Sieve Bootstrap is asymptotically valid for weakly stationary linear processes with short memory and homoskedastic, i.i.d. innovations. However, in the context of electricity price forecasting, these assumptions are violated. To relax these assumptions, I introduce a novel method: **Sequential Sieve Bootstrap Predictive Inference (SSBPI)**. This method combines the autoregressive resampling approach of the Sieve Bootstrap (Bühlmann, 1998) with the rolling-window, online nature of conformal methods like SPCI (Xu and Xie, 2021b). Unlike standard conformal or ensemble bootstrap methods, which assume exchangeable errors, SSBPI captures serial dependence by fitting AR models sequentially to a rolling buffer of XGBoost residuals. This is particularly important as XGBoost, while flexible, does not model temporal structure explicitly, leading to residual autocorrelation—especially at longer horizons.

In summary, while machine learning models like XGBoost offer strong point forecast performance, their predictive reliability weakens in the presence of volatility, nonstationarity,

and autocorrelation. Distributional models (e.g., XGBoostLSS) rely on strong parametric assumptions, while conformal methods provide flexible intervals but often overlook residual structure. The SSBPI algorithm complements conformal methods by preserving residual dependencies through autoregressive resampling, enabling distribution-free uncertainty quantification without calibration data, and adapting to local temporal dynamics.

3 Methodology

For the set-up of this study, I roughly followed the structure for machine learning based forecasting methods described by Bojer (2022). This methodology section is set up in two phases. The first phase describes data selection, preprosessing, feature engineering, model training, hyperparameter selection and (cross) validation. The second phase describes uncertainty estimation. In the data selection and pre-processing sections, I describe how the data was retrieved, which variables were chosen and why, and missing data handling and outlier detection/handling, as well as scaling. In the second section, I describe the features that were engineered and later on how a select set of features was determined for all the models in question. In the third section, I describe how the Autoregressive (AR), Linear regression, and XGBoost models were set up. In the uncertainty estimation section, I describe the algorithms used or developed in this study to construct confidence intervals around the point predictor. I conclude with performance metrics used in this study.

3.1 Data selection and preprocessing

This section provides a detailed description of the data used in this study. I have aimed to structure the research in a logical and reproducible way. To support this, all data is openly accessible and sourced from a single provider: The ENTSO-E (2025) Transparency Platform. Access requires only an API key, which can be easily requested on the platform's website.

3.1.1 Construction of the dataset

First of all, the length of the dataset and interval between the values was selected based on the forecast objective (hourly forecasts) and an initial analysis of the available data availability and structure, as described in Appendix A.1. The timeframe chosen to train and calibrate is shown in Figure 2. In order to catch full seasonality, the test set contains a full year, spanning January 2023 until January 2024. The calibration set spans January 2024 until March 2024. The final test set that was put aside spans from March 2024 until May 2025.

For the period spanning 2023 to 2025, hourly values were retrieved for the Netherlands on electricity prices (in euros per megawatt-hour), electricity generation from wind, solar, and coal (in megawatt-hours), and electricity consumption, or load (also in megawatt-hours). In addition to historical values, the corresponding day-ahead forecasts for each of

Figure 2: Dutch electricity prices; the train-test split of this study.

Note: The data ranges from January 2023 until May 2024. The validation period of January 2023 until March 2024 is used to optimize model performance iteratively. The period from March 2024 until May 2024 is used to test coverage of constructed confidence intervals through various uncertainty estimation methods.

these exogenous variables— except for coal, since this variable had little variability-was also collected. These forecasts were not very accurate, as can be observed in figure 4 , but since bids are largely determined by prognoses of generation, they were included for analysis also. Other variables were briefly evaluated, but left out due to insufficient correlation with the hourly prices, or due to insufficient data quality (in the case of electricity generation from gas, for example.) The selected variables and their correlation to prices are shown in Figure 3.

Figure 3: Correlation matrix of Dutch energy prices with covariates selected in this study

Outliers were initially removed using a combination of domain-specific value ranges and IQR-based statistical filtering. Variables such as electricity price, renewable generation, and consumption were checked against predefined plausible bounds and interquartile thresholds. This process flagged and removed a small percentage of extreme or anomalous values, which were then imputed using forward and backward filling.

However, a later comparison of model performance on both the cleaned and raw datasets revealed that retaining the outliers actually improved forecasting accuracy, particularly for methods sensitive to rare but informative price spikes. As such, the full dataset—including outliers—was ultimately used for model training and evaluation.

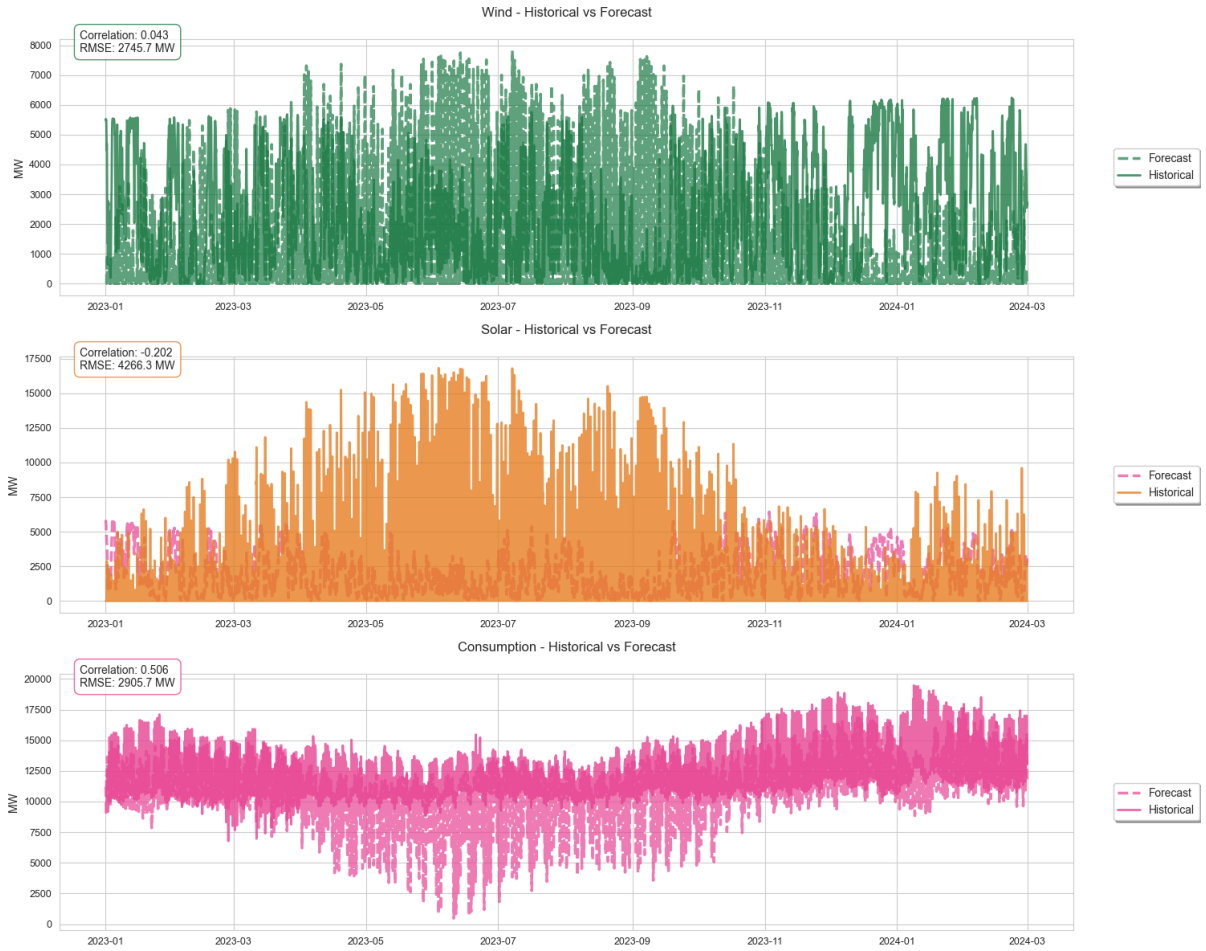

Figure 4: Historical energy generation vs. ENTSO-E forecasted energy generation, from January 2023 until March 2024.

Before model fitting, all input features were standardized using z-score normalization (zero mean, unit variance). This scaling step is essential for regularized linear models and feature selection procedures (e.g., RFECV), which are sensitive to the magnitude of input features.

Due to the regulatory framework built around the Dutch Electricity prices, I expected some non-linear relationships between the covariates. These nonlinearities are evaluated using Generalized Additive Models (GAM), and the results can be found in Figure 5. Based on the plots correlation and GAM plots, one can derive that all the variables considered most probably have some predictive power over the prices, but judging from the slight waves or ¨U¨ shapes in the plots, and the heavy line around a Price of 0 MW, which can be interpreted of a regulatory pull away from negative prices, those relationship are most probably not entirely linear.

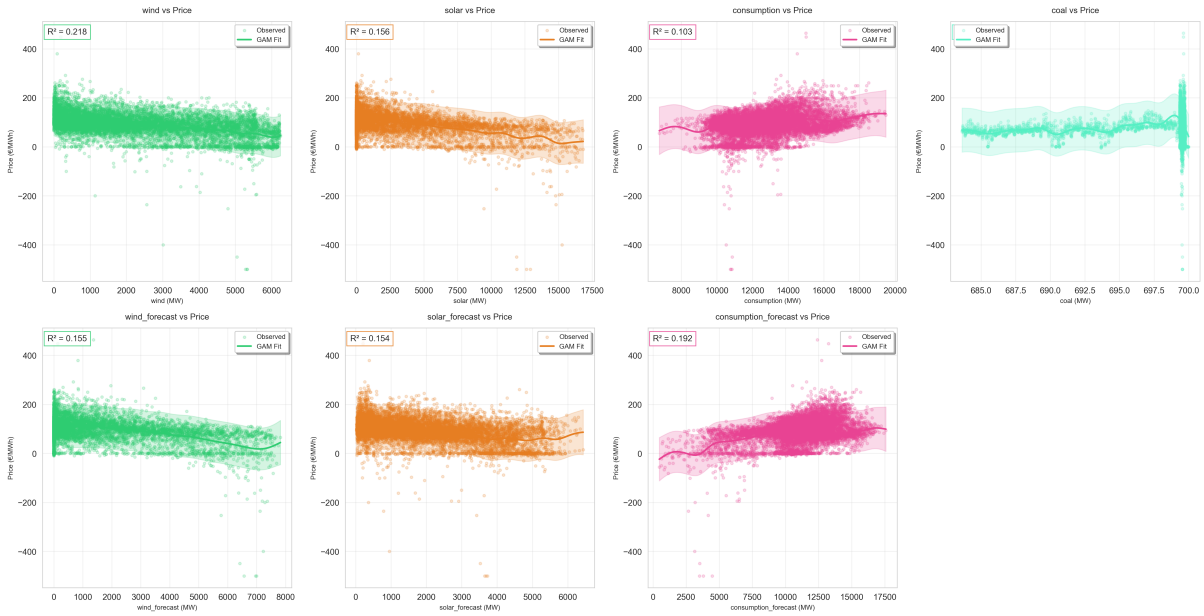

Figure 5: GAM plots of the covariates vs. DA electricity prices

The focus on short-term electricity price forecasting—specifically the 14 to 38-hour horizon—is directly motivated by the structure of the Dutch DA electricity market. In this market, participants are required to submit their bids for each hour of the following day by noon on the day before delivery. As a result, I chose to create forecasting targets for the hours t+14, t+24 and t+38 hours ahead. This way, the models can be run at 10:00 AM everyday, to ensure a 2-hour interval between prediction and submission of bids for the earliest, middle,and final hour of the next day.

Below in Table 1 the basic statistics of the train, validation and test set are described. Interestingly, the average prices and min and max value vary a lot, whereas the standard deviation is roughly similar. The additional test set on top of the validation set is to ensure no data leakage or overfitting.

Additional stationarity diagnostics are reported in Table 2. This is important for

Statistic	Training Set	Validation Set	(uncertainty) Test Set
Timeframe	Jan2023-Jan2024	Jan2024-Mar2024	Mar2024-May2024
Observations	8017	1248	1257
Mean	88.27	133.70	59.06
Standard Deviation	45.70	42.79	36.28
Minimum	-500.00	-0.33	-125.30
Maximum	463.77	270.20	176.28

Table 1: Summary statistics of electricity prices (EUR/MWh)

autoregressive modeling, as ARIMA-type models assume that the time series is stationary—i.e., its mean and autocovariance structure do not change over time. The Augmented Dickey-Fuller (ADF) test rejects the null hypothesis of a unit root, indicating that the DA prices in the selected period are stationary in levels.

Test	**Test Statistic**	**p-value**
Original Series		
ADF	-10.014	1.74×10^{-17}
KPSS	4.216	0.010
First-Differenced Series		
ADF	-24.175	0.000
KPSS	0.063	0.100

Table 2: Stationarity analysis of the electricity price series

Note: The dataset is complete with no missing values or temporal gaps. First differencing results in a clearly stationary series, as confirmed by both ADF and KPSS tests.

Lastly, the KPSS test, which evaluates the null hypothesis of stationarity around a deterministic trend, rejects this assumption for the electricity price series. This suggests the presence of structural changes—such as shifting means and time-varying volatility—consistent with well-documented properties of electricity markets. Specifically, prices exhibit mean reversion around changing levels, strong intra-day and intra-week seasonality (see Figure 6), pronounced heteroskedasticity, and clear level shifts across seasons (see Figure 7).

These findings have two key implications. First, they justify the use of **flexible, non-linear models** such as XGBoost and XGBoostLSS, which can learn from high-dimensional and seasonal features without requiring stationarity. Second, they underscore the limitations of point forecasts, especially when underlying assumptions (e.g., constant variance, fixed trend structure) are violated. Accordingly, I evaluate a range of uncertainty estimation techniques that can help capture uncertainty in imperfect point forecasts.

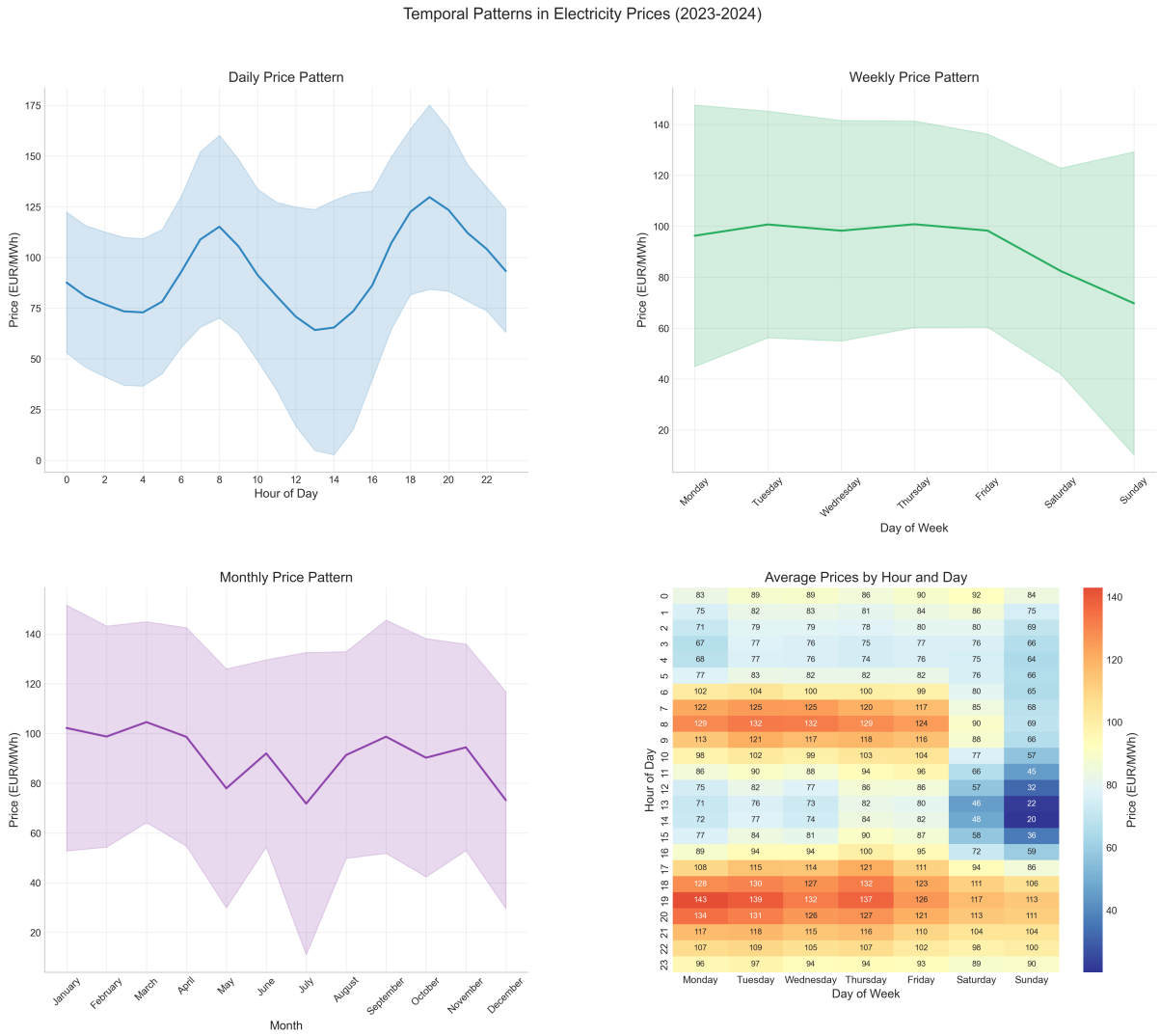

Figure 6: Temporal patterns of DA prices of the entire train+validation period. *Note:* From top left to bottom right; average daily pattern, average weekly pattern, average monthly pattern, and day of the week combined with hour of day.

Figure 7: Seasonal patterns of DA prices of the entire train+validation period. *Note:* From top left to bottom right; average daily price pattern by season, price distribution per season, price volatility by season - plotted over the hours of the day, and average prices by hour and season.

3.1.2 Feature engineering

In this study, I applied extensive feature engineering to transform raw variables into informative predictors that enhance forecasting performance. The primary challenge was to create meaningful transformations without introducing irrelevant features that could lead to overfitting. The feature set includes standard time-series features such as lagged values, rolling statistics, calendar indicators, and exogenous covariates, consistent with best practices in time-series forecasting (Bojer and Meldgaard, 2020). A detailed description of the features engineered can be found in Appendix A.2.

Features were derived from both the target variable (hourly electricity prices) and external variables such as wind and solar generation, coal-based production, and electricity consumption. These transformations were applied to both historical and forecasted values, resulting in an empirically grounded set of 305 features. An overview is presented in Table 3.

To mitigate overfitting, I performed a second round of feature selection, tailored to each model class. For the Autoregressive (AR) model, I began with 72 lags and used the BIC to select an optimal subset. For the Linear Regression model using direct multi-horizon forecasting, I employed RFECV, which iteratively removes features with low predictive contribution. For XGBoost, I selected the top 100 most predictive features based on SHAP values, using a model already optimized for its hyperparameters (see Section 3.4 for details). As a benchmark, I also used fixed lag orders: 1, 2, 3 (near-term dynamics), and 24, 48, 168 hours (capturing daily and weekly patterns), chosen relative to the forecast horizon (e.g., for a 14-hour ahead forecast, lag 24 corresponds to time $t - 10$).

3.2 Model formulations

Although XGBoost is not inherently designed for time series forecasting, by explicitly introducing lagged versions of the target variable as input features, the model can learn temporal dependencies in a way that mimics traditional autoregressive models, such as the AR(p).

The model can be applied in either a direct or recursive forecasting setup. In the recursive approach, I would fit the model on all train data, predict the price one step ahead, shift the train window to include the predicted price as if it was ¨real¨, and then predict the price for the next step ahead, until I reach the desired forecast horizon. This would, however, lead to accumulation of serial autocorrelation in the residuals, and by the time h14-h38 is reached, not much predictive power would be left if the model did not capture this autocorrelation fully in the first place. To show the problem with recursive forecasting, I use a simple AR(p) model fitted on the full training set and applied recursively to the test set. As seen in Appendix A.3, the model quickly produces a flat line and fails to follow actual price movements. This shows how prediction quality can drop sharply over longer horizons if the model doesn't capture time dependencies well. As a fairer benchmark Autoregressive model, I allow for daily retraining and reparameterisation for the AR model, which is described below in Section 3.2.1. However, this

Category	Count	Description
Time Features	18	Hour, day, month, quarter, year, and week of year; cyclical encodings (sine/cosine); part-of-day indicators (morning, afternoon, evening, night); weekend indicator.
Holiday Features	3	Dutch public holiday indicator, day before holiday, day after holiday.
Lagged Features	224	Historical lags for electricity price (1–168h), and for generation and forecast variables (wind, solar, coal, consumption) at 0h, 1h, 2h, 3h, 6h, 12h, 24h, 48h, and 168h.
Rolling Statistics	14	Rolling mean and standard deviation of electricity prices over 3h, 6h, 12h, 24h, 48h, 72h, and 168h windows.
Price Differences	7	First-order price differences over lags of 1h, 2h, 3h, 24h, 48h, 72h, and 168h.
Target Features	38	Forecast horizons from t+1 to t+38 hours, used as separate target variables for multi-horizon prediction.
Other	1	Current price at time t included for reference.

Table 3: Feature set for energy price forecasting

model, not surprisingly, just mimics the last train window exactly for each fold because it is purely autoregressive and univariate, relying only on past prices.

Alternatively, direct forecasting involves training separate models to predict directly for each desired prediction horizon (e.g., h=14 untill 38 hours ahead). This would mean that the model would directly try to find patterns between features before time t and the target variable at t+h. This approach resembles more a direct linear regression model than an autoregressive model, with the XGBoost benefit of finding nonlinear relationships quite efficiently.

3.2.1 Autoregressive AR(p) model with recursive forecasting

The rolling AR model refits the model at the start of each forecast window t_{forecast}, using all data up to that point. At each such time t_{forecast}, a new AR($p^{(t)}$) model is fitted:

$$y_\tau = \beta_0^{(t)} + \sum_{i=1}^{p^{(t)}} \beta_i^{(t)} y_{\tau-i} + \epsilon_\tau \quad \text{for all } \tau \leq t_{\text{forecast}} \tag{1}$$

where:

- $p^{(t)}$ is the optimal lag order selected dynamically at time t_{forecast} (e.g., via BIC minimization),
- $\beta^{(t)}0, \beta^{(t)}1, \ldots, \beta^{(t)}p^{(t)}$ are the re-estimated coefficients at each $t$forecast.

Forecasts are then made recursively:

$$\hat{y}_{t_{\text{forecast}}+h} = \beta_0^{(t)} + \sum_{i=1}^{p^{(t)}} \beta_i^{(t)} \tilde{y}_{t_{\text{forecast}}+h-i} \tag{2}$$

For refitting, both the lag structure $p^{(t)}$ and the coefficients $\beta^{(t)}$ are allowed to change over time.

3.2.2 Linear regression with direct multi-horizon forecasting

The second benchmark model in this study is a direct forecasting linear regression, where a separate model is trained for each forecast horizon. Since this model relies entirely on its input features, special attention was paid to feature engineering and selection.

First, I created a baseline feature set containing only time-based and calendar features (hour, day of week, holidays, etc.). Then, I extended this with lagged price values to capture recent trends (lags 1–3 hours), daily patterns (lags 24 and 48 hours), and weekly cycles (lag 168 hours). These lag features were carefully aligned with the forecast target time $(t + h)$.

To evaluate the contribution of covariates like generation and consumption, I tested additional feature sets: one excluding all covariate-related features, and one including them. Finally, RFECV was applied to select the most informative features from the full set for each horizon. The linear regression model with lags can be formally defined as:

$$y_{t+h} = \beta_0 + \sum_{i \in \mathcal{L}} \beta_i \, y_{t+h-i} + \sum_{j \in \mathcal{F}} \gamma_j \, X_{t+h,j} + \epsilon_{t+h} \tag{3}$$

Where:

- $\mathcal{L}$ is the set of included lag orders (e.g., $1, 2, 3, 24, 48, 168$ if price lags are used),
- $\mathcal{F}$ is the set of included feature indices (time-based variables, calendar indicators, generation, consumption, etc.),
- $X_{t+h,j}$ represents the j-th covariate at forecast time $t + h$, which where predetermined covariates (see Appendix A.2).
- ϵ_{t+h} is the forecast error term.

Depending on the variant of the linear model under consideration, the lag set $\mathcal{L}$ may be empty (when no lag features are included), and the feature set $\mathcal{F}$ may vary by including or excluding external covariates. The final model may also represent a reduced subset of features selected through RFECV.

3.2.3 XGBoost

The third and final model selected for this regression task is XGBoost. While this choice may not seem immediately self-evident, I will use this section to explain its underlying rationale. The use of XGBoost was suggested by analysts at energy provider REPOWERED. REPOWERED has experience forecasting energy prices and has previously achieved promising results using XGBoost in the Passive Imbalance market. Although this reasoning is not purely scientific, it is practically grounded: One of the goals of this thesis is to develop a model that is not only accurate, but also usable and interpretable by the team at REPOWERED. In that sense, building on existing internal knowledge and proven results makes XGBoost a logical and relevant starting point.

For the description of XGBoost I refer back to the paper of Chen and Guestrin (2016), which was also the main reference of the system in the literature review. As stated in the paper, the scalability of XGBoost is due to several important systems and algorithmic optimizations, including a novel tree learning algorithm used for handling sparse data; a theoretically justified weighted quantile sketch procedure that enables handling instance weights in approximate tree learning. Parallel and distributed computing making learning faster which enables quicker model exploration. And most importantly, XGBoost exploits out-of-core computation, which further speeds the process and allows for really fast data handling. To understand how it operates, I will first explain gradient tree boosting, which is the core of the system.

In XGBoost, the estimation at each iteration t is based on minimizing the following regularized objective function:

$$L^{(t)} = \sum_{i=1}^{n} \ell \left[y_i, \hat{y}_i^{(t)} \right] + \Omega(f_t) \tag{4}$$

Where ℓ is a differentiable convex loss function that measures the discrepancy between the prediction of the i-th instance at the t-th iteration $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$ and the true value y_i, while $\Omega(\cdot)$ is a regularization term that penalizes the complexity of the model to avoid overfitting.

Expanding this for the second-order approximation of $\ell[\cdot]$ and dropping constant terms, we can re-write the objective function as:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{5}$$

Where:

$$g_i = \frac{\partial \ell[y_i, \hat{y}_i^{(t-1)}]}{\partial \hat{y}_i^{(t-1)}} \; h_i \qquad\qquad = \frac{\partial^2 \ell[y_i, \hat{y}_i^{(t-1)}]}{\partial \hat{y}_i^{(t-1)2}}$$

Further expanding the regularization term $\Omega(\cdot)$, the objective function becomes:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2 \tag{6}$$

Where w_j are the leaf weights, γ is a parameter that controls the penalization for the number of terminal nodes T of the trees, and λ is an L_2 regularization term on the leaf weights.

XGBoost employs a tree-pruning method that is governed by the following gain equation:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] \tag{7}$$

Where:

- G_L and H_L are the sum of gradients and Hessians in the left child node.
- G_R and H_R are the sum of gradients and Hessians in the right child node.
- λ is the regularization term that controls the model complexity.

The pruning occurs if the calculated gain is negative, leading to the removal of the split. XGBoost adopts a greedy algorithm to determine the best splits in each tree, selecting the node with the highest gain. However, the standard greedy algorithm can be computationally inefficient, especially when there are numerous features and observations in the dataset. To mitigate this, XGBoost introduces an approximate algorithm: The data is first divided into quantiles, effectively reducing the number of potential split points. A weighted quantile sketch is then used to determine candidate split points. Unlike a standard percentile that evenly distributes candidates, this method ensures that each quantile has an equal sum of weight. Additionally, XGBoost incorporates several regularization techniques to prevent overfitting:

- **Learning Rate (η)**: This parameter, ranging between 0 and 1, controls the contribution of each tree in the final model. A lower learning rate reduces the impact of each individual tree, providing more stable training.
- **Subsampling**: XGBoost can randomly select a fraction of features (column subsampling) and training instances (row subsampling) for each tree, similar to the random forest method. These subsampling ratios help prevent overfitting by introducing randomness into the model training process.

In the direct forecasting approach, which is used in this study, a separate model $f^{(h)}$ is trained for each forecast horizon h to directly map the features available at time t to the target value y_{t+h}:

$$\hat{y}_{t+h} = f^{(h)}(x_t) \tag{8}$$

Where x_t represents the feature vector at time t, and $f^{(h)}$ is the XGBoost model specifically trained to predict the future target at horizon h. This allows the model to learn horizon-specific patterns without relying on intermediate recursive predictions.

3.3 XGBoostLSS

To address the need for probabilistic forecasting in the presence of heteroskedasticity and skewed target distributions, März (2019) propose XGBoostLSS, an extension of gradient boosting to distributional regression. Unlike standard regression models that only predict the conditional mean of the target variable, XGBoostLSS allows for simultaneous prediction of multiple distributional parameters (e.g., mean, standard deviation, skewness, kurtosis) of a pre-specified likelihood distribution.

This is achieved by fitting a separate XGBoost model to each distributional parameter in a two-step procedure. In the first step, each parameter is estimated independently using a boosting model optimized with respect to the negative log-likelihood, assuming the other parameters are fixed. This structure allows each model to use its own set of hyperparameters, tuned independently, which is particularly important since the learning dynamics for location (mean) and scale (standard deviation) can differ significantly. In the second step, the parameters are iteratively updated in a joint fashion to improve global fit, while typically retaining the independently chosen hyperparameters. The full procedure is described below in Algorithm 1.

Algorithm 1: Rolling forecasting with XGBoostLSS distributional regression

Data: Multivariate feature set $\{x_t\}$, targets $\{y_{t+h}\}$, forecast horizons $h \in \mathcal{H}$, rolling window size $T = 365$ days, step size $s = 7$ days, coverage level $1 - \alpha$

Result: Point forecasts $\hat{y}_{t+h}$ and Prediction intervals $\hat{C}^{\text{LSS}}_{t+h}$ for all t in the test period

for *each forecast horizon* $h \in \mathcal{H}$ **do**

  for *each rolling window fold* k **do**

    Step 1: Data Preparation

      Select training window $[t_k - T,\ t_k)$ and test window $[t_k,\ t_k + s)$;

      Extract feature matrix X and target vector $y = y_{t+h}$ from training data;

      Drop any future target columns from X to avoid leakage;

    Step 2: Hyperparameter Optimization (Inner CV)

      Initialize XGBoostLSS model with Gaussian distribution;

      Perform 5-fold cross-validation within training data to minimize negative log-likelihood (NLL);

      Select optimal hyperparameters and number of boosting rounds n_{rounds};

    Step 3: Model Training and Distributional Prediction

      Train the XGBoostLSS model on full training data using optimized parameters;

      Predict distributional parameters $\hat{\mu}_{t+h}$ and $\hat{\sigma}_{t+h}$ on test data;

    Step 4: Prediction Interval Construction

      Compute prediction intervals at level $1 - \alpha$:

$$\hat{C}^{\text{LSS}}_{t+h} = [\hat{\mu}_{t+h} - z_\alpha \hat{\sigma}_{t+h},\ \hat{\mu}_{t+h} + z_\alpha \hat{\sigma}_{t+h}]$$

      where z_α is the standard Gaussian quantile;

    Step 5: Evaluation and Drift Tracking

      Compute RMSE, SMAPE, R^2, prediction interval coverage, and average interval width;

Step 7: Aggregate Results Across All Folds

Point forecast fold results are aggregated and reported for Jan 2024 – March 2024, Interval width and coverage fold results are aggregated and reported for March 2024 – May 2024;

return $\tilde{C}_{t+h}$ *for all* $t \in \mathcal{T}_{test}$

3.4 Cross validation and hyperparameter tuning

To evaluate model performance in a manner that respects the temporal structure of electricity price data, a rolling window cross-validation strategy is implemented. This approach uses a 365-day training window followed by a 7-day test window, with a step size of 7 days to ensure non-overlapping folds. For the AR(p), the rolling implementation

was switched to a daily interval at a later stage to increase performance. At each iteration, the model is trained on the most recent year of data and validated on the subsequent week. This simulates a realistic operational scenario where the forecasting model is retrained weekly with the latest available data. For every fold, predictions are made for each target horizon—specifically 14, 24, and 38 hours ahead—and forecast are collected. After completing all folds, performance metrics as described in Section 3.6 are computed and aggregated for each horizon to report the metrics across folds. The exact setup is described in Figure 8 below.

Hyperparameter tuning and model evaluation were conducted in a sequential and structured manner to ensure consistency and avoid data leakage. The period from January 2023 to January 2024 was reserved for hyperparameter tuning. In the case of the XGBoost model, this was done using automated optimization with Hyperopt, where the data was split into the first 80 percent for training and the last 20 percent for validation, using the full feature (305 columns) set. In case of XGBoostLSS, the package standard of hyperparameter optimization through cross-validation used as described in Algorithm 1.

For hyperparameter optimization of the regular XGBoost point forecaster, which was later used as the base model for all uncertainty estimation except the distributional uncertainty estimation (which was XGBoostLSS), cross-validation was attempted but due to computational limits, finally kept at a single split. The evaluation of the optimized hyperparameter set was tested on multiple week-sized test windows to ensure it was consistent over multiple timeframes. Finally, since different forecast horizons led to the selection of distinct optimal hyperparameter sets during tuning, I chose to respect these outcomes and hard-code three separate configurations: one for short-term forecasts (t+14h), one for mid-term (t+24h), and one for long-term (t+38h). For the benchmark AR(p) model, no explicit hyperparameter tuning was required, apart from the automatic selection of lag length based on the BIC. For the linear regression, also only features had to be selected.

Following hyperparameter tuning, model evaluation was carried out on the period from January 2024 to March 2024 using the rolling window approach described earlier. This evaluation phase included comparisons between different feature subsets: one set incorporating external variables such as electricity generation and consumption, and another set based solely on lagged price features. In all cases, the models used the fixed hyperparameters determined prior to final feature selection, ensuring a fair and consistent comparison.

Feature selection for XGBoost was guided by SHAP values, which provide a model-agnostic measure of feature importance. For the Autoregressive and Direct Linear forecasting methods, BIC and RFECV were used, respectively. Both selection procedures were embedded within the same rolling window framework applied during evaluation to maintain temporal consistency. Once the optimal set of features was determined for each model, the final models were retrained on the full dataset from January 2023 to March 2024 using the previously selected hyperparameters and features. These finalized models were then used for subsequent uncertainty estimation methods, including probabilistic and conformal approaches.

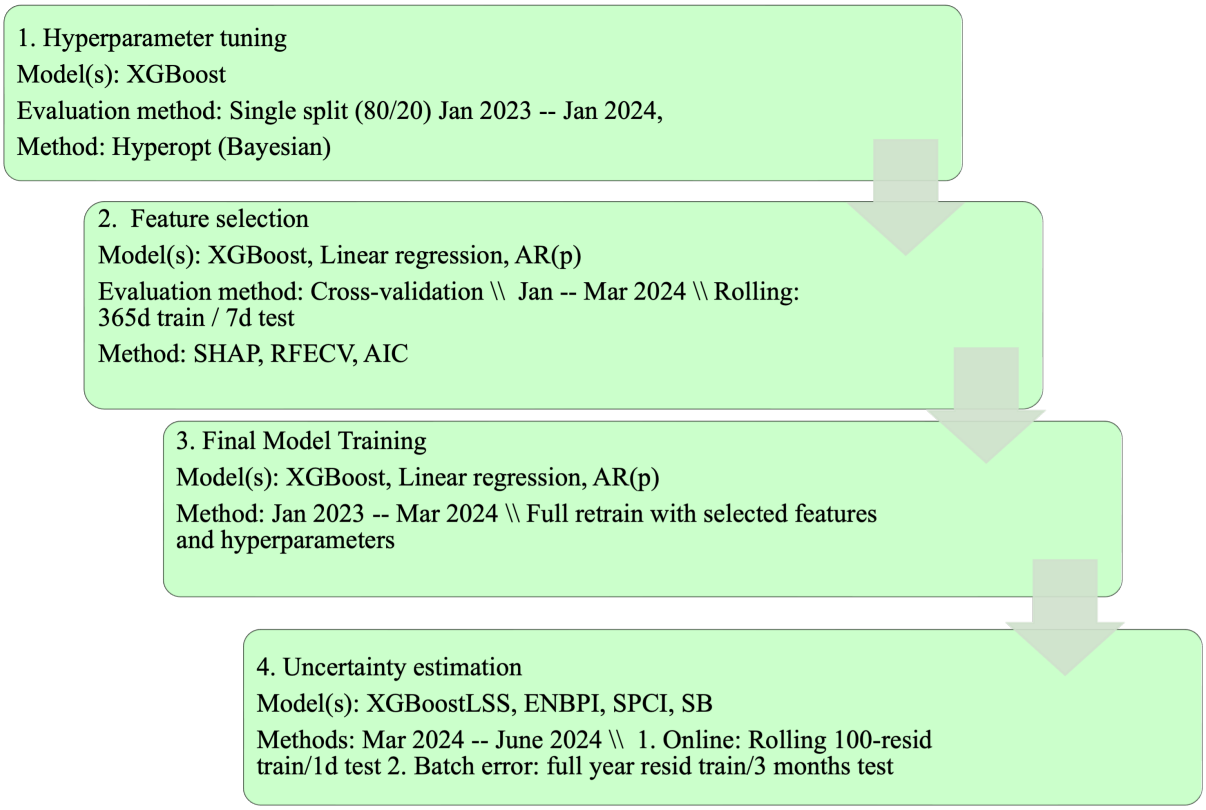

Figure 8: Chronological modeling and evaluation workflow

Note: In step 2 The feature selection methods SHAP, RFECV and BIC are linked to XGBoost, Linear regression and AR(p) respectively. In step 3-4 XGBoostLSS is a significantly different procedure, where the rolling window method of step 2 is followed throughout the entire period of Jan 2023-May 2024, in which point forecasts are evaluated and reported for Jan-2024 – March 2024, and lower and upper confidence bounds (parametrically estimated through same algorithm as the point forecaster) are evaluated only for March 2024 – May 2024 to match the timeframe of the other uncertainty estimation methods. Also for XGBoostLSS, hyperparameters are reevaluated for each cross validation split, thus for this method Step 1 can be disregarded.

3.5 Uncertainty estimation

The second phase of this study focuses on uncertainty estimation. Rather than generating only point forecasts, the goal is to estimate either the full distribution of outcomes for each forecast horizon or, using non-parametric methods, construct prediction intervals with 90% nominal coverage. These intervals are evaluated on a separate test set that was not used during the optimization of the point forecasting models, enabling a true out-of-sample assessment of coverage.

Dataset construction. First, the optimal models were retrained on the full training dataset (January 2023 to March 2024). Then, predictive intervals for the test set are constructed using one of three approaches: parametric (XGBoostLSS), non-parametric (ENBPI, SPCI), or semi-parametric (Sieve Bootstrap). These next subsections describe these techniques. It must be noted that due to the separate construction of the train and test set and subsequent construction of the features, a data gap was created of 1

week between the train and test data. This was because the test feature set, which had lags of variables going back 1 week in time, was trimmed from the beginning to start in March 2024. This resulted in the entire dataset being shifted one week forward to accommodate these lagged features. This greatly impacted performance of the point predictor, especially visible in the benchmark models.

3.5.1 XGBoostLSS confidence intervals

The first method for uncertainty estimation is the XGBoost probabilistic forecasting framework proposed by März (2019).

In this approach, the target variable is modeled using a Gaussian likelihood, allowing the model to learn both the conditional mean μ and standard deviation σ as functions of the input features. XGBoostLSS leverages Newton boosting, where each boosting iteration solves a regularized weighted least squares problem based on a second-order Taylor approximation of the negative log-likelihood:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \tag{9}$$

where g_i and h_i are the first and second derivatives of the loss with respect to the predicted parameter, and $\Omega(f_t)$ is a regularization term that penalizes model complexity.

As described in Algorithm 1, each distributional parameter is modeled by a separate XGBoost model, with its own set of hyperparameters optimized via cross-validation. For consistency, the same feature set selected during earlier XGBoost experiments was used.

To construct prediction intervals, the parametric standard deviation $\hat{\sigma}_{t+h}$ estimated by the model is used directly. Assuming normality, the $(1-\alpha)$ prediction interval is given by:

$$\hat{C}_{t+h}^{\text{LSS}} = \left[\hat{\mu}_{t+h} - z_\alpha \hat{\sigma}_{t+h},\ \hat{\mu}_{t+h} + z_\alpha \hat{\sigma}_{t+h} \right], \tag{10}$$

where z_α is the corresponding quantile of the standard normal distribution. This yields symmetric, analytically tractable confidence intervals centered around the predicted mean.

3.5.2 Ensemble Batch Predictive Intervals (EnbPI)

To construct distribution-free prediction intervals using model ensembles, I applied the Ensemble Batch Predictive Intervals (EnbPI) method. EnbPI extends conformal prediction to ensemble settings by calibrating prediction intervals based on out-of-bag (OOB) residuals from bootstrapped models (Xu and Xie, 2021a).

This approach is particularly appealing because it avoids distributional assumptions and does not require a separate calibration set. Instead, it uses residuals from training samples that were excluded from each bootstrap sample—thus leveraging the inherent structure of the ensemble for calibration.

The method unfolds in the following steps:

1. **Bootstrap sampling.** I generated B bootstrap datasets from the training set using sampling with replacement. For each dataset, I tracked which samples were out-of-bag.

2. **Model training.** The optimized XGBoost model was trained on each bootstrap dataset, yielding an ensemble of B forecasters.

3. **Residual calibration.** For each training sample, I computed the OOB residual as the absolute difference between the true value and the average prediction from all models for which the sample was not in the training set.

4. **Ensemble prediction.** The ensemble prediction for each test point was taken as the median of predictions across the B models.

5. **Interval construction.** I computed the empirical $(1 - \alpha)$ quantile of the OOB residuals and constructed symmetric prediction intervals centered on the ensemble median forecast.

This method provides an intuitive and model-agnostic way to estimate uncertainty. I used $B = 20$ due to computational constraints, which yielded moderate coverage at the expense of interval sharpness.

Online EnbPI variant. To adapt the EnbPI method for sequential forecasting, I implemented an online extension that retains the core bootstrap ensemble structure while updating the uncertainty estimates dynamically. An ensemble of B XGBoost models is trained once on the full training set using bootstrap resampling with replacement. Each model f_b learns to predict the target y_{t+h} based on input features observed at time t, and their outputs are retained throughout testing for consistent inference.

At each prediction time t, the point forecast $\hat{y}_t$ is computed as the mean of ensemble predictions:

$$\hat{y}_{t+h} = \frac{1}{B} \sum_{b=1}^{B} f_b(x_t) \tag{11}$$

To construct prediction intervals, I maintain a rolling buffer of recent residuals, defined as $\hat{\varepsilon}_t = |y_{t+h} - \hat{y}_{t+h}|$. This buffer is updated sequentially and truncated to a fixed length T (e.g., 100) to ensure adaptability to recent dynamics and avoid long-term drift.

The simplest interval construction uses the empirical $(1 - \alpha)$ quantile of the residual buffer:

$$\tilde{C}_{t+h} = [\hat{y}_{t+h} - q_{1-\alpha},\ \hat{y}_{t+h} + q_{1-\alpha}] \tag{12}$$

where $q_{1-\alpha}$ is the empirical upper quantile of the most recent T residuals.

To enhance robustness against noise and local variability, I incorporate a strided quantile refinement step. Using the `strided_app` method, the residual buffer is transformed into a matrix of overlapping windows $\mathcal{E}^{(t)}$ of fixed length L and stride $s = 1$. The final interval width is determined by averaging the $(1 - \alpha)$ quantiles computed within each window:

$$q^{(t)} = \frac{1}{|\mathcal{E}^{(t)}|} \sum_{w \in \mathcal{E}^{(t)}} \text{Quantile}_{1-\alpha}(w) \tag{13}$$

This procedure mitigates the effect of outliers by ensuring each residual only influences a subset of windows, and it implicitly captures local residual autocorrelation. The resulting interval is:

$$\tilde{C}_t = [\hat{y}_t - q^{(t)},\ \hat{y}_t + q^{(t)}]$$

The complete procedure is summarized in Algorithm 2.

Algorithm 2: Ensemble Bootstrap Prediction Intervals (EnbPI) with online updating

Input: Full dataset $\mathcal{D}$ (Jan 2023 – Jun 2024), forecast horizon h, target coverage $1-\alpha$, residual window size w

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t in test set

Step 1-3: Final model setup

Fix tuned hyperparameters and selected features;

Train point predictor f on all training pairs $\{(x_t, y_{t+h})\}$ from Jan 2023 – March 2024;

Step 4: Ensemble bootstrap training

Let B be the number of bootstrap models;

for $b = 1$ **to** B **do**

  Sample a bootstrap dataset $\mathcal{D}^{(b)}$ with replacement from $\mathcal{D}_{\text{train}}$;

  Train XGBoost model f_b on $\mathcal{D}^{(b)}$;

  Predict $f_b(x_t)$ on all x_t in train + test set and store outputs;

Compute ensemble mean prediction:

$$\hat{f}(x_t) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_t)$$

Step 5: Calibration via out-of-bag residuals

Initialize residual buffer ε as empty list;

for *each training sample* x_i **do**

  Use only models f_b where $x_i \notin \mathcal{D}^{(b)}$;

  Compute:

$$\hat{f}_{-i}(x_i) = \text{mean}(\{f_b(x_i) \mid x_i \text{ is OOB}\})$$

  Compute residual $\hat{\varepsilon}_i = |y_i - \hat{f}_{-i}(x_i)|$;

  Append to buffer ε;

Keep only the last w residuals: $\varepsilon \leftarrow \varepsilon_{n-w+1:n}$;

Step 6: Online forecasting and interval update

for *each test sample* x_t *with observed* y_t **do**

  Compute ensemble prediction $\hat{f}(x_t)$ from all f_b;

  Estimate quantile width q_α from current buffer ε:

$$q_\alpha = \text{Quantile}_{1-\alpha}(\varepsilon)$$

  Construct prediction interval:

$$\tilde{C}_{t+h} = [\hat{f}(x_t) - q_\alpha,\ \hat{f}(x_t) + q_\alpha]$$

  Observe y_t and compute new residual $\hat{\varepsilon}_t = |y_t - \hat{f}(x_t)|$;

  Update buffer $\varepsilon \leftarrow (\varepsilon \cup \{\hat{\varepsilon}_t\})[-w:]$;

Step 7: Evaluation

Compute empirical coverage and average interval width over test set;

return $\tilde{C}_{t+h}$ *for all* $t \in \mathcal{T}_{test}$

3.5.3 (Online) Sequential Predictive Conformal Inference (SPCI)

To construct adaptive, time-aware prediction intervals around my XGBoost point forecasts, I implemented the Sequential Predictive Conformal Inference (SPCI) method. This method extends traditional conformal prediction by modeling the conditional distribution of residuals based on their recent history, thus allowing for intervals that reflect temporal variation in forecast error.

Standard conformal prediction methods estimate a single quantile of the residual distribution from a held-out calibration set, assuming residuals are independent and identically distributed (i.i.d.). However, this assumption does not hold in time series contexts, especially in high-volatility domains such as electricity price forecasting. Residuals often exhibit autocorrelation and non-stationary behavior, which makes static, global quantiles suboptimal for coverage calibration.

SPCI addresses this limitation by estimating the residual quantiles conditionally on recent residual sequences. Specifically, rather than using a global quantile, it learns a mapping:

$$\mathrm{PI}(x_t) = \hat{f}(x_t) \pm \mathrm{QR}(\epsilon_{t-w}, \ldots, \epsilon_{t-1}), \tag{14}$$

where $\hat{f}(x_t)$ is the point forecast, ϵ_t denotes residuals, w is the window size (e.g., 100), and $\mathrm{QR}(\cdot)$ is a quantile regressor that maps the residual history to the predicted lower and upper residual quantiles. The implementation proceeds in three stages:

1. **Point forecasting.** I trained the Optimized XGBoost model $\hat{f}$ on historical data from January 2023 to March 2024 to predict electricity prices at horizons of 14, 24, and 38 hours. Residuals were computed as $\epsilon_t = y_t - \hat{f}(x_t)$ on the training set.

2. **Quantile regression on residual windows.** Using a sliding window approach with $w = 100$, I formed feature–target pairs from the training residuals: each feature vector contained the past 100 residuals, and the target was the subsequent residual. These were used to fit three XGBoost-based quantile regression models with custom loss functions to predict the 5%, 50%, and 95% quantiles of the residuals per forecast horizon. For this standard pinball loss is applied as in Xu and Xie (2021a)

3. **Prediction interval construction.** At test time (March 2024 —May 2024), the most recent residual window was fed into the trained quantile models to estimate the future residual quantiles. These were added to the point forecasts to construct prediction intervals:

$$\mathrm{PI}(x_t) = \hat{f}(x_t) + (\hat{q}_{0.05} - \hat{q}_{0.50}, \hat{q}_{0.95} - \hat{q}_{0.50}), \tag{15}$$

   where $\hat{q}_\tau$ are the predicted conditional residual quantiles.

The full procedure is described in Algorithm 3.

Algorithm 3: Online SPCI for forecast horizon h using XGBoost and Residual Quantile Regression

Input: Full dataset $\mathcal{D}$ (Jan 2023 – Jun 2024), forecast horizon h, target coverage $1-\alpha$, residual window size w

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t in test set

Step 1: Fit point forecaster

Fit optimized (XGBoost) model f on training set to predict y_{t+h} from x_t;

Compute residuals on training set: $\hat{\varepsilon}_t = y_{t+h} - f(x_t)$;

Initialize residual buffer $\mathcal{R} \leftarrow \{\hat{\varepsilon}_{T-w}, \ldots, \hat{\varepsilon}_{T-1}\}$

Step 2: Initialize quantile models

Set lower quantile level $\beta = \alpha/2$ and upper level $1-\beta$;

Initialize quantile models Q_β, $Q_{0.5}$, $Q_{1-\beta} \leftarrow$ None

Step 3: Online prediction with periodic quantile updates

for $t \in \mathcal{T}_{test}$ **do**

  Predict point forecast: $\hat{y}_{t+h} = f(x_t)$;

  if $t \bmod k = 0$ *and* $|\mathcal{R}| \geq w$ **then**

    Construct training set $\{(\tilde{x}_t, \tilde{y}_t)\}$ from rolling buffer $\mathcal{R}$:

    for $j = 1$ *to* $|\mathcal{R}| - w$ **do**

      $\tilde{x}_j \leftarrow [\mathcal{R}_j, \ldots, \mathcal{R}_{j+w-1}], \quad \tilde{y}_j \leftarrow \mathcal{R}_{j+w}$;

    Train or update XGBoost quantile regressors Q_β, $Q_{0.5}$, $Q_{1-\beta}$ on this dataset;

  if *quantile models are available* **then**

    Form test residual input: $\tilde{x}_t \leftarrow [\mathcal{R}_{-w}, \ldots, \mathcal{R}_{-1}]$;

    Predict residual quantiles: $\hat{r}_t^{\text{low}} = Q_\beta(\tilde{x}_t)$,;

    $\hat{r}_t^{\text{mid}} = Q_{0.5}(\tilde{x}_t)$,;

    $\hat{r}_t^{\text{up}} = Q_{1-\beta}(\tilde{x}_t)$;

    Compute prediction interval:

$$\tilde{C}_{t+h} = \left[\hat{y}_{t+h} + (\hat{r}_t^{\text{low}} - \hat{r}_t^{\text{mid}}),\ \hat{y}_{t+h} + (\hat{r}_t^{\text{up}} - \hat{r}_t^{\text{mid}})\right]$$

  else

    Use default interval: $\tilde{C}_{t+h} = [\hat{y}_{t+h} \pm c]$ (e.g., $c = 15$);

  Observe y_{t+h} and compute $\hat{\varepsilon}_t = y_{t+h} - \hat{y}_{t+h}$;

  Append $\hat{\varepsilon}_t$ to buffer $\mathcal{R}$, discard oldest if $|\mathcal{R}| > w$

return *Prediction intervals* $\tilde{C}_{t+h}$ *and evaluation metrics*

3.5.4 Sieves Bootstrap, autoregressive modelling of errors for uncertainty estimation

To construct prediction intervals that account for autocorrelation in the forecast errors, I implemented the Sieve Bootstrap method. This approach extends classical bootstrapping by fitting an autoregressive (AR) model to the residuals of a point forecaster, capturing temporal dependencies that would be missed by i.i.d. resampling.

Unlike conformal prediction methods, which assume exchangeability, the sieve bootstrap explicitly models residual dynamics via an AR(p) process. The fitted AR model is then used to simulate new residual sequences, which are added to the point forecasts to produce synthetic trajectories. Quantiles of these trajectories form the final prediction intervals.

Specifically, the procedure consists of five key steps:

1. **Point forecasting.** I trained an XGBoost regressor $\hat{f}$ on data from January 2023 to March 2024 for each forecast horizon. Residuals were computed as $\epsilon_t = y_t - \hat{f}(x_t)$.
2. **Residual modeling.** I selected an appropriate lag order p via BIC and fit an AR(p) model to the residual sequence to capture its temporal structure.
3. **Bootstrap simulation.** Residuals of the AR model were resampled with replacement to generate bootstrap innovations. Using the estimated AR coefficients and sampled innovations, I simulated B new residual series.
4. **Forecast path generation.** Each bootstrapped residual sequence was added to the corresponding point forecast $\hat{f}(x_t)$ to obtain a bootstrapped forecast path.
5. **Interval construction.** Final prediction intervals were derived from the empirical quantiles of the B bootstrapped trajectories for each forecast time point.

This method is fully data-driven and model-agnostic beyond the AR assumption for residuals. However, it assumes stationarity and homoskedasticity in the residual process, which may not hold in volatile environments like electricity markets. As a result, the method tends to produce narrow intervals and suffers from undercoverage, as confirmed in the empirical results.

The full procedure is described in Algorithm 4.

Algorithm 4: Prediction intervals with XGBoost and Sieve Bootstrap

Input: Trained XGBoost model f, residuals $\{\hat{\varepsilon}_t\}_{t\in I_1}$, forecast inputs x_t for $t \in \mathcal{T}_{\text{test}}$, number of bootstrap samples B, coverage level $1-\alpha$

Output: Prediction intervals $\tilde{C}_{t+h}$ for $t \in \mathcal{T}_{\text{test}}$

Step 1: Fit AR model to residuals

Fit an autoregressive model of order p (e.g., AR(p)) to $\{\hat{\varepsilon}_t\}_{t\in I_1}$ to obtain estimated parameters $\phi = (\phi_1, \ldots, \phi_p)$ and residuals $\{\hat{z}_t\}$;

Step 2: Generate B bootstrap residual series

for $b = 1$ *to* B **do**

Resample $\{\hat{z}_t\}$ with replacement to get bootstrap innovations $\{\tilde{z}_t^{(b)}\}$;

Generate synthetic residual series $\{\tilde{\varepsilon}_t^{(b)}\}$ using:

$$\tilde{\varepsilon}_t^{(b)} = \sum_{i=1}^{p} \phi_i \tilde{\varepsilon}_{t-i}^{(b)} + \tilde{z}_t^{(b)}$$

Step 3: Forecast with added residuals

Predict mean forecasts $\hat{y}_{t+h} = f(x_t)$ for all $t \in \mathcal{T}_{\text{test}}$;

for $b = 1$ *to* B **do**

Sample residuals $\tilde{\varepsilon}_{t+h}^{(b)}$ from the synthetic series;

Generate bootstrap forecast:

$$\tilde{y}_{t+h}^{(b)} = \hat{y}_{t+h} + \tilde{\varepsilon}_{t+h}^{(b)}$$

Step 4: Construct prediction intervals

for $t \in \mathcal{T}_{test}$ **do**

Define prediction interval as empirical quantiles:

$$\tilde{C}_{t+h} = \left[\text{Quantile}_{\alpha/2}\left(\{\tilde{y}_{t+h}^{(b)}\}_{b=1}^{B}\right),\ \text{Quantile}_{1-\alpha/2}\left(\{\tilde{y}_{t+h}^{(b)}\}_{b=1}^{B}\right)\right]$$

return $\tilde{C}_{t+h}$ *and evaluation metrics*

Sequential Sieve Bootstrap Predictive Inference (SSBPI) The final uncertainty estimation model is the Sieve Bootstrap with online updating of the most recent residual window, analogous to the approaches used in SPCI and online ENBPI. The continuous updating of the residual window allows the prediction intervals to better adapt to structural shifts in the data and also partially mitigates the issue of residual non-exchangeability, since the bootstrap samples are drawn from a temporally localized and evolving error distribution rather than assuming stationarity across the entire dataset.

Algorithm 5: Sequential Sieve Bootstrap Predictive Inference (SSBPI) with XGBoost point forecasts

Input: Full dataset $\mathcal{D}$ (Jan 2023 – Jun 2024), forecast horizons $\mathcal{H} = \{14, 24, 38\}$, training window size W, residual buffer size R (e.g., 100), number of bootstrap samples B, significance level α

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t and $h \in \mathcal{H}$

Step 1: Rolling point forecasts with XGBoost

for *each forecast time t in test set (hourly)* **do**

  Extract training window $\mathcal{D}_{\text{train}} = \{y_{t-W}, \ldots, y_{t-1}\}$;

  Train XGBoost model on $\mathcal{D}_{\text{train}}$;

  for *each horizon $h \in \mathcal{H}$* **do**

    Predict $\hat{y}_{t+h}$ using model;

    Store $(t, t+h, \hat{y}_{t+h}, y_{t+h})$ in forecast log $\mathcal{F}_h$;

Step 2: Initialize residual buffer

If pre-test forecasts are available, compute residuals and initialize buffer $\mathcal{E}$ with the most recent R residuals;

Step 3: Online interval construction with Sieve Bootstrap

for *each horizon $h \in \mathcal{H}$* **do**

  for *each forecast time t in $\mathcal{F}_h$* **do**

    Retrieve residual buffer $\mathcal{B}_t = \mathcal{E}[t - R : t]$;

    if $|\mathcal{B}_t| < 20$ **then**

      skip to next t

    // Fit AR model to residuals

    Fit initial AR model on $\mathcal{B}_t$ with max lags;

    Select significant lags using BIC penalty;

    Fit final AR(ϕ) model and extract residuals η_t;

    // Sieve Bootstrap

    Simulate B bootstrap residual paths $\{\tilde{e}^{(b)}_{1:h}\}_{b=1}^{B}$ from AR(ϕ) with sampled η_t;

    Extract h-step residuals: $\tilde{e}^{(b)}_{h} = \tilde{e}^{(b)}_{1:h}[h]$;

    // Construct prediction interval

$$\text{lower}_{t+h} = \hat{y}_{t+h} + \text{Quantile}_{\alpha/2}\left(\{\tilde{e}^{(b)}_{h}\}_{b=1}^{B}\right)$$

$$\text{upper}_{t+h} = \hat{y}_{t+h} + \text{Quantile}_{1-\alpha/2}\left(\{\tilde{e}^{(b)}_{h}\}_{b=1}^{B}\right)$$

    Store interval $(\text{lower}_{t+h}, \text{upper}_{t+h})$;

    Append observed residual $y_{t+h} - \hat{y}_{t+h}$ to buffer $\mathcal{E}$;

Step 4: Evaluation

for *each horizon $h \in \mathcal{H}$* **do**

  Compute empirical coverage: $\frac{1}{N}\sum_t \mathbb{1}\{y_{t+h} \in [\text{lower}_{t+h}, \text{upper}_{t+h}]\}$;

  Compute mean interval width: $\frac{1}{N}\sum_t (\text{upper}_{t+h} - \text{lower}_{t+h})$;

return $\tilde{C}_{t+h}$ *and evaluation metrics*

3.6 Evaluation metrics

To evaluate the quality of both the point forecasts and the uncertainty estimates, I use a combination of standard and custom-designed metrics tailored to the electricity price forecasting setting.

Root Mean Squared Error (RMSE). RMSE is one of the most commonly used metrics for evaluating the accuracy of point forecasts in regression problems. It measures the square root of the average of the squared differences between predicted values $\hat{y}_t$ and actual observations y_t:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}. \tag{16}$$

Coefficient of determination (R^2). This metric evaluates how well the model explains variance in the target variable:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2}. \tag{17}$$

Adapted Symmetric Mean Absolute Percentage Error (SMAPE). RMSE IS more sensitive to large errors, while the Mean Absolute Percentage Error (MAPE) is more sensitive to small errors and outliers, and is also not defined for observations with zero actual values. To alleviate some of these issues, I consider the symmetric mean absolute percentage error (sMAPE) I developed an adapted version of SMAPE that addresses two common challenges in electricity price forecasting: the occurrence of zero and negative prices.

$$\text{SMAPE} = \frac{100}{n}\sum_{t=1}^{n}\frac{2|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t| + \varepsilon}, \tag{18}$$

where ε is a small constant added to prevent division by zero.

Diebold-Mariano test. The Diebold-Mariano (DM) test is used to statistically compare the predictive accuracy of two competing forecasting models. It evaluates whether the difference in forecast errors—typically based on squared or absolute loss—is significantly different from zero. For a given loss function $L(\cdot)$, the test computes a loss differential series and assesses its mean relative to its sampling distribution:

$$\text{DM Statistic} = \frac{\bar{d}}{\sqrt{\widehat{\text{Var}}(\bar{d})}}, \tag{19}$$

where $\bar{d}$ is the mean difference in losses and $\widehat{\text{Var}}(\bar{d})$ is an estimate of its variance. A significant DM statistic (with a small p-value) indicates that one model outperforms the other in a statistically meaningful way. The test is particularly useful for confirming whether improvements in forecasting metrics like RMSE are robust or due to chance.

Coverage. Coverage indicates the proportion of actual values that fall within the predicted prediction intervals. For a nominal confidence level of $1 - \alpha$, I evaluate:

$$\text{Coverage} = \frac{1}{n} \sum_{t=1}^{n} \mathbb{1}\left\{ y_t \in [\hat{y}_t^{\text{lower}}, \hat{y}_t^{\text{upper}}] \right\} \cdot 100. \tag{20}$$

Mean Interval Width. The average width of the prediction intervals reflects their informativeness. Narrow intervals are preferred as long as they maintain proper coverage:

$$\text{Mean Width} = \frac{1}{n} \sum_{t=1}^{n} \left(\hat{y}_t^{\text{upper}} - \hat{y}_t^{\text{lower}} \right). \tag{21}$$

4 Results

4.1 Point forecast accuracy evaluation

This section presents and analyzes the results of both point forecasting and probabilistic forecasting models. Evaluation metrics for point forecasts include RMSE, adapted SMAPE (aSMAPE), and R^2. The evaluation metrics are evaluated per fold and then aggregated to form one final outcome for each forecast horizon. Forecasts are evaluated across three horizons: 14, 24, and 38 hours ahead. Results are displayed in Table 4.

Model	**RMSE**			**aSMAPE (%)**			R^2			**Time (s)**
	t+14	t+24	t+38	t+14	t+24	t+38	t+14	t+24	t+38	
AR(p) (BIC)	17.72	19.49	18.40	**19.39**	33.87	**19.34**	0.32	0.22	0.10	4.96
AR(p) (preselected lags)	17.42	20.27	18.81	20.13	34.92	20.35	0.34	0.16	0.06	370.54
Linear (RFECV)	35.67	32.59	39.94	35.35	31.17	37.63	-1.15	-0.81	-1.80	64161.21
Linear (no exogenous)	18.97	20.01	21.71	24.83	25.59	27.11	0.42	0.35	0.21	43679.70
XGBoost (SHAP)	17.46	**17.59**	19.54	24.98	23.61	26.59	**0.38**	**0.36**	0.20	353.73
XGBoost (SHAP, No exogenous)	**17.23**	17.99	**16.08**	28.35	29.97	29.97	0.29	0.20	**0.36**	342,22
XGBoostLSS	17.34	18.71	18.83	20.28	**22.72**	23.50	0.37	0.28	0.22	6520,21

Table 4: Mean point forecast performance (Jan 2024 – March 2024) Across models, folds, and horizons

AR. First, the results do not immediately give away a clear difference in performance, especially considering different that for the 3 metrics (RMSE, aSMAPE, and R^2) and for different horizons, the performance seems scattered over the different models. The simplest model, containing only future values of the predictor of interest (the electricity prices at a certain horizon), performs competitively for RMSE values below 20, which is quite acceptable for average prices of 133.70, and a st.dev of 42.79 Euro per MWh, as shown in the validation set column in Table 1. Also, the use of BIC is slightly preferred over preselected lags (1-24, 24,48 and 168, which were chosen for their ability to catch the nearest values due to autocorrelation, and specific time points like one or two days or 1 week ago due to pressumed patterns). Furthermore, from ACF plots and accompanying Ljungbox tests that is presented in Table 6, it seems that the AR model is well specified,

since the residuals have a white noise (IID) structure. Only in the h38 forecast horizon was the IID assumption of the residuals not achieved. However, AR(p) is allowed daily retraining due to computational efficiency, something that is not evaluated in this study for XGBoost or the linear regression models. This is a clear advantage because at each day, the most recent values available are used to construct both the lag order and the AR(p) coefficient values. And considering that the electricity prices show both daily and weekly temporal dependence as shown in Table 6, for a time series that does not show the greatest outliers and/or volatility over a timespan of 2 months, AR(p) is a good option. The Diebold-Mariano test acknowledges this, as can be seen in Table 5, AR does not underperform to the linear regression. However, the AR does seem to underperform in comparison to XGBoost, when aggregating predictions across folds and horizons.

Linear Regression with RFECV. The second benchmark catagory, the linear regression with RFECV using the full feature set (including exogenous values) seems to be the least successful in predicting energy prices across horizon according to the RMSE, aSMAPE, and R^2. his could be due to the fact that there were non linear patterns found between the exogenous variables and the target variables as indicated by the GAM plots in Figure 5. One would think that the RFECV cross-validation method would adress this. When removing the exogenous variables prior to applying RFECV however, performance increases by more than 10 euro/mwh in the RMSE catagory, 10% in the sMAPE and the R^2 also aproximates the values seen in XGBoost and AR(p). The computational efficiency of this method, however, is inferior to the other methods. The time to run the model with weekly retraining took up to 17.8 hours. Due to this, regardless of residual drift concerns, for uncerainty estimation the weekly retraining was excluded. Looking at Table 6, the best-performing linear model—specifically the version excluding exogenous variables—did not fully eliminate autocorrelation in the residuals. Furthermore, the Diebold-Mariano test in Table 5 indicates no statistically significant difference in predictive accuracy between the best-performing XGBoost model and the best-performing linear regression model. The Linear Regression model shows better performance than the AR model, but this difference is only statistically significant at the 10% level, not at the conventional 5% threshold.

XGBoost. The XGBoost with SHAP-selected features achieves the lowest RMSE at the medium horizon (t+24h) and performs competitively across all horizons, particularly with the best R^2 at t+14h and t+24h. The XGBoost (SHAP, no exogenous) variant yields the lowest RMSE at t+14h and t+38h, and the highest R^2 at t+38h, despite a slight increase in aSMAPE. This is perhaps due to the fact that sMAPE is more sensitive to low-valued observations, which are lest caught by the XGBoost model, as is visible in 11.

The XGBoostLSS model performs best in terms of aSMAPE for t+24h, and remains competitive in RMSE and R^2. However, a key distinction is that XGBoostLSS includes hyperparameter optimization per test fold, which was intentionally omitted for the XGBoost SHAP and no-exogenous variants to ensure computational feasibility in the uncertainty estimation experiments (ENbPI, SPCI, Sieve Bootstrap). Since hyperparameter

tuning is central to the original XGBoostLSS formulation by März (2019)—affecting both the location and scale predictions—it was retained here. Still, this introduces an unfair comparison to the fixed-hyperparameter XGBoost variants. A Diebold-Mariano test was performed between the optimized XGBoost model that was retrained weekly, and the one that is not retrained weekly, showing weekly retraining significantly increases performance (Diebold-mariano statistics of no retrain vs retrain= -3.1693, p-value = 0.0015). A general decrease in model performance for longer horizons is observed. Which is expectable due to the temporal dependency of the data.

Model Comparison	DM Statistic	p-value
AR vs Linear	-1.9147	0.0555
AR vs XGBoost	-2.4397	0.0147
Linear vs XGBoost	-0.4767	0.6336

Table 5: Diebold-Mariano Test Results (Squared Error Loss)

Note: Results are aggregated across all horizons. Actual and predicted values for t+14, t+24, and t+38 are grouped by model and compared at matching timestamps across models.

Forecast Horizon	Ljung-Box p-value	Interpretation
AR (BIC)		
t+14h	0.4394	white noise
t+24h	0.8624	white noise
t+38h	0.0009	autocorrelation
Linear (RFECV-No exogenous)		
t+14h	0.0000	autocorrelation
t+24h	0.0000	autocorrelation
t+38h	0.0000	autocorrelation
XGBoost (optimized feature set)		
t+14h	0.0000	autocorrelation
t+24h	0.0000	autocorrelation
t+38h	0.0000	autocorrelation

Table 6: Ljung-Box test results for residual autocorrelation at lag 10 across AR, Linear, and XGBoost models.

4.2 Uncertainty estimation evaluation

XGBoostLSS prediction intervals. For distributional forecasting, I employed the XGBoostLSS model using a Gaussian likelihood. Separate models were trained for each forecast horizon (14h, 24h, and 38h) and evaluated over March–May 2024. The resulting 90% prediction intervals underperformed significantly in terms of coverage—achieving only 62.4%, 66.0%, and 53.0%—despite moderately wide average widths between 41 and 54 EUR/MWh. This suggests a misestimation of heteroskedasticity, possibly due to

Table 7: Prediction Interval Performance (March–May 2024): Coverage, Width, and Calibration Time at 90% Confidence Level

Method	**Coverage (%)**			**Avg. Interval Width**			**Time (sec)**
	t+14h	t+24h	t+38h	t+14h	t+24h	t+38h	
XGBoostLSS	62.4	66.0	53.0	47.99	54.05	41.11	9042.30
XGBoost EnbPI (Batch error)	69.2	66.3	64.8	46.43	45.66	45.29	1096.97
XGBoost EnbPI (Online error)	79.5	79.73	78.4	63.56	78.76	77.25	1247.96
XGBoost SPCI (Batch error)	43.7	43.0	52.3	22.51	22.81	35.32	216.67
XGBoost SPCI (Online error)	59.8	57.8	61.3	**16.27**	**16.18**	**23.91**	5011.03
XGBoost Sieve Bootstrap (Batch error)	24.9	22.2	17.3	10.48	10.38	10.77	83.54
XGBoost Sieve Bootstrap (Online error)	**80.3**	**79.2**	**78.5**	30.49	31.30	30.92	78.84
AR(p) EnbPI (Simplified)	86.5	82.4	88.2	70.52	116.05	76.72	114.13
AR(p) SPCI (Simplified)	42.6	43.1	42.0	47.00	48.60	48.09	672.32
AR(p) Sieve Bootstrap (Simplified)	65.3	67.5	70.9	93.42	95.58	107.22	73.74
Linear RFECV EnbPI (Simplified)	78.4	78.8	78.1	65.83	83.98	67.51	263.53
Linear RFECV SPCI (Simplified)	39.6	48.0	39.7	37.36	53.92	39.72	593.95
Linear RFECV Sieve Bootstrap (Simplified)	54.4	72.0	62.7	77.65	96.88	82.27	272.08

Note: Batch error refers to using the full residual sequence from the training set to estimate prediction intervals for the entire test set in one pass.This is standard for Sieve Bootstrap and EnbPI. For SPCI, which is inherently sequential, batch estimation is shown for comparative purposes only. Note two: Here, for period March-May, no second round of weekly cross-validation and retraining is performed for XGBoost. For AR and RFECV, retraining was performed also for the test period of March 2024 – May 2024. This complicated the online adaptation of the uncertainty estimation methods, resulting in simplification of the algorithms (first estimating the full testset point forecast, and then feeding a residual buffer with these predetermined point forecasts). Lastly, a week gap between the train and test set for these prediction should be reported.

overfitting in the scale component or incorrect distributional assumptions. Moreover, the use of K-fold cross-validation for hyperparameter tuning—without preserving temporal order—may have exacerbated overfitting and degraded generalization to out-of-sample residual structure.

SPCI prediction intervals. Sequential Predictive Conformal Inference (SPCI) was implemented using quantile regressors trained on residual windows of size w=100. In the batch (non-updating) version, SPCI failed to reach the nominal 90% coverage, achieving only 43.7%, 43.0%, and 52.3% coverage for t+14h, t+24h, and t+38h, respectively. However, the online version, which updates residuals sequentially, showed large improvements: coverage rose to 59.8%, 57.8%, and 61.3%, while keeping intervals narrow (16–24EUR).

ENBPI prediction intervals. The Ensemble Batch Prediction Interval (ENBPI) method, applied to XGBoost, achieved moderate coverage in its batch version: 69.2%, 66.3%, and 64.8%. The online variant significantly improved coverage to 79.5%, 79.7%, and 78.4%, though at the cost of wider intervals (64–78EUR). These results show that for moderate point forecast performance, ENBPI's comes at a hightened computational cost and wider bounds.

Sequential Sieve Bootstrap Predictive Inference (SSBPI) The SB method using batch residuals from XGBoost performed poorly, with coverage far below nominal—only 24.9%, 22.2%, and 17.3%. In contrast, the online SBPI version achieved 80.3%, 79.2%, and 78.5% coverage with moderate interval widths (31EUR), making it one of the most accurate and efficient methods in this study. Figure 9 shows that the uncertainty bands for the SSBPI are more centred around the actual values than the predicted values, which is a clear difference from, for example, ENbPI, in which the residuals are naturally centered around the predicted values.

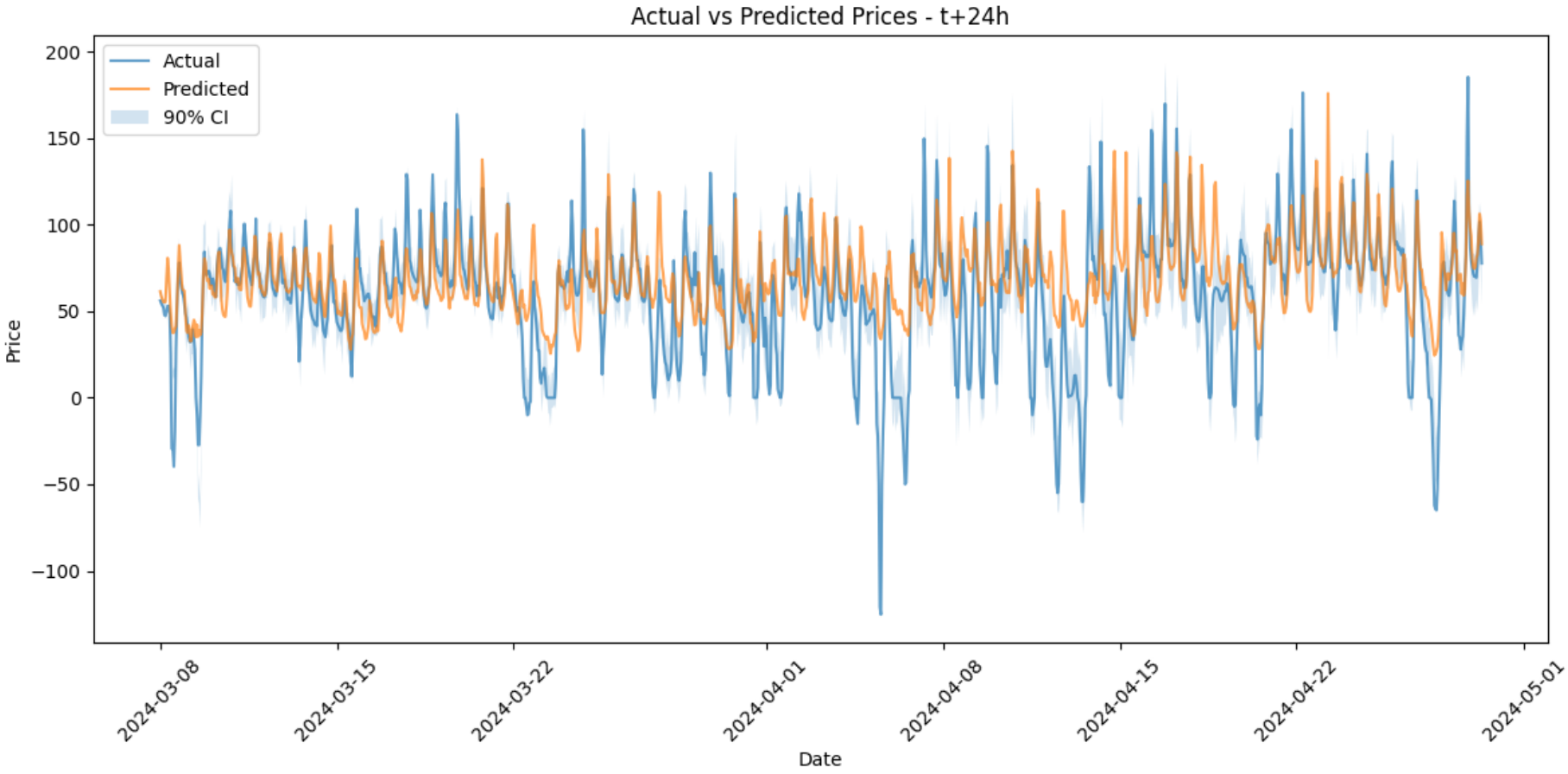

Figure 9: Sequential Sieve Bootstrap Predictive Inference (SSBPI) around XGBoost point forecaster

Simplified uncertainty estimation of the Benchmark models None of the uncertainty estimation applied to the benchmark models showed promising results. This can be attributed to the fact that the 1-week data gap greatly reduced forecasting efficiency of these relatively simple, linear models. Furthermore, due to time constraints, the uncertainty estimation methods had to be adapted to accommodate daily/weekly retraining in favor of point forecast accuracy. The simplified algorithms for the benchmarkmodels can be found in Appendix A.6.

5 Discussion

Based on the literature reviewed in Section 2, several persistent challenges in electricity price forecasting (EPF) emerge. These challenges motivate the methodological choices made in this thesis and frame the interpretation of empirical results. This section aims to explain how these choices, as well as additional concerns and considerations for future direction.

1. **Non-stationarity around a deterministic trend; Regime Shifts affecting accuracy of point forecasts:** Electricity prices often exhibit nonstationary behavior, including abrupt regime shifts, seasonal effects, and changing trends. Many statistical point forecasting methods such as AR(P) models assume (trend-)stationarity or exchangeability, assumptions that are frequently violated in real-world electricity markets. While linear regression and XGBoost do not explicitly require stationarity or exchangeability, they can still suffer from model misspecification and degraded performance if nonstationarity is not properly addressed through feature engineering or retraining.

2. **Variate selection** As the forecasted values from the ENTSOE platform were not really accurate, it could have been beneficial to include other covariates such as humidity, rainfall, GDP, or other macroeconomic or weather variates. As was done in for example in (Kapoor and Wichitaksorn, 2023). This could have increased the accuracy of the XGBoost an linear regression point forecasters, and thus have given a better base for further uncertainty estimation.

3. **Effect of extreme volatility and heteroskedasticity in uncertainty estimation:** A critical distinction among uncertainty estimation methods lies in their treatment of heteroskedasticity—time-varying variance in the target variable. When variances change over time, or when volatility clusters are found in data, naïve or heuristic Interval Estimation can fall short. In contrast, methods like quantile regression, XGBoostLSS, conformalized quantile regression (CQR), and SPCI explicitly accommodate heteroskedasticity. Quantile regression captures conditional variance by modeling different parts of the distribution, while XGBoostLSS estimates scale and shape parameters directly. Sieve bootstrap and residual-based SPCI indirectly address heteroskedasticity by resampling or modeling residuals conditioned on recent lags, offering greater flexibility in nonstationary settings. Static prediction intervals, such as those derived from split conformal prediction, do not

adapt to local data conditions. The online sequential methods applied or developed in this thesis provide more adaptive and temporally responsive interval estimates.

4. **Residual Autocorrelation due to unsuited point forecaster.** Time series, in most cases, exhibit autocorrelation. For this reason, statistical models have been developed to account for this. Even though there are also Machine Learning models that specifically address autocorrelation in time-dependent data, such as LSTM, XGBoost is not one of them. In this thesis, a great attempt was done to account for this autocorrelation. For example, lags of prices were included, and seasonalities were attempted to be caught Fourier transformations of certain time characteristics. Nonetheless, the Ljung box test showed significant autocorrelation in the residuals of both the XGBoost model as the linear regression direct forecasting model. This resulted in another complication in EPF, (next to the already evident challenges mentioned in the two points above; and inaccurate generation forecast data, non-complete open source databases, and unpredictable weather patterns.) Namely, what if your model has trouble in characterizing time dependency in the first place? In hindsight, perhaps for the challenge of electricity price forecasting, I would have chosen a different base model. But for this study, the true challenge was thus to balance out the lack of time-dependency estimation in point forecast, whilst also accounting for other reasons of non-exchangeability of errors, such as heteroskedasticity, regime shifts, structural breaks and volatility spikes.

5. **Retraining of the point forecast vs Online adaptation of uncertainty estimation** In a later stage of this study, I discovered that retraining made a significant difference in results of the point forecaster. Ofcourse, this has a direct influence on the performance. The original studies of (Xu and Xie, 2021b) and (Xu and Xie, 2021a) treat the point forecaster as a fixed user-specified black box, trained on a static training set. Thus, to go beyond their initial algorithm set up, I subjected all models to a retraining regime to optimize the point forecast and tested on the period of Jan 2024 – March 2024. Then when applying the uncertainty estimation methods to the same models in the uncertainty testing period March 2024 – May 2024 without retraining (as was done in (Xu and Xie, 2021b) for example, yielded very bad results due to drifting point forecasters, especially for the AR(p) and Linear Regression Models. To fix this as a next step I reimplemented the retraining for each train-test fold in all the models. However, I discovered that this complicated the daily estimation of uncertainty intervals, which is supposed to happen consecutively with the daily estimation of the point forecaster. Retraining on a weekly (or even daily) timeframe and applying the uncertainty estimation methods to this on a daily interval is beyond the scope of this study due to time and computational constraints. I showcase the differences of the uncertainty estimation under the pretrained point XGBoost forecaster as in (Xu and Xie, 2021b). For the Benchmark models, I showcase the differences between uncertainty estimation for slightly simplified online calibration described in Appendix A.6, in favour of more accurate point forecasts due to weekly retraining. Thus, in-between model comparisons are hard to make from this study, this is a shortcoming.

6. **Short Forecast Horizons (Multi-step Direct):** Much of the literature focuses on one-step-ahead or daily forecasting. Whereas this study in theory also performs a one step ahead forecast, this step size is adapted to the horizon of interest, thus evaluating seperate models for seperate horizons to prevent accumulation of serial autocorrelation across horizons. (Xu and Xie, 2021b) introduce Sequential Predictive Conformal Inference (SPCI) as a method for constructing valid prediction intervals in time series forecasting. When applying SPCI to multi-step ahead predictions, they highlight a key challenge: jointly estimating the residual distribution $\{\hat{\epsilon}_{t+1}, \ldots, \hat{\epsilon}_{t+S}\}$ across future time steps. This joint estimation is necessary in recursive or autoregressive models, where each future prediction $\hat{y}_{t+s}$ depends on previous predicted values $\{\hat{y}_{t+1}, \ldots, \hat{y}_{t+s-1}\}$, thereby inducing dependencies among residuals. Chen et al. refer to this as "highly challenging," motivating a simplified divide-and-conquer approach where one fits S separate models per horizon using leave-one-out (LOO) bootstrap sampling. However, in this thesis, this joint residual modeling is not necessary due to the choice of a **direct forecasting strategy**. Specifically, for each forecast horizon $h \in \{14, 24, 38\}$, I train a separate model to directly predict y_{t+h} using only information available up to time t. This setup is used both for the XGBoost quantile regression model and the RFECV-based linear regression baseline. In this setting, each model learns a mapping $f_h : X_t \mapsto y_{t+h}$, and does not rely on previously predicted values for other horizons. As such, the residuals $\hat{\epsilon}_{t+h} = y_{t+h} - \hat{y}_{t+h}$ for different values of h are *conditionally independent given the features* at time t. This justifies treating each forecast horizon separately when applying conformal prediction, such as in our implementation of SPCI. However, for a fixed horizon h, the residuals $\{\hat{\epsilon}_{t+h}\}_t$ may still exhibit temporal dependence, especially in volatile time series like electricity prices. These autocorrelations arise from underlying temporal patterns not fully captured by the features and can affect the sharpness of the prediction intervals.

7. **Computational Efficiency and Model Complexity:** Fully probabilistic methods, such as Bayesian or distributional models like XGBoostLSS, offer accurate uncertainty estimates but are computationally intensive and sensitive to irrelevant features. Efficient feature selection and scalable implementations are essential to ensure generalization and real-time applicability in high-dimensional settings. Furthermore, probabilistic forecasting methods such as XGBoostLSS or Bayesian models rely on parametric assumptions about the data distribution. If these assumptions are violated, resulting prediction intervals may be invalid or misleading. One of the core strength of XGBoost as a regression technique is its fast and efficient data handling characteristics, which was why it was chosen in this study. However, despite this, applying other ML techniques such as hyperparameter optimization, cross validation and multiple loss functions in the case of quantile regression, did eventually lead to computational constraints.

8. **Cross-Validation for Time Series forecasting and uncertainty estimation:** Standard K-fold cross-validation violates the temporal ordering of data, making it unsuitable for time series. Rolling window or pseudo out-of-sample (POOS) cross-

validation is more appropriate, especially for hyperparameter tuning and performance evaluation. A relevant question posed by my thesis supervisor was whether uncertainty estimation—specifically the trade-off between prediction interval width and coverage—can be optimized using cross-validation. While this may appear appealing in theory, several studies argue against using standard cross-validation to directly optimize for uncertainty quality. For instance, Angelopoulos and Bates (2021) emphasize that "coverage is a discrete quantity that cannot be optimized with gradient descent," and that conformal prediction instead adjusts predictions post hoc to guarantee valid coverage. Similarly, Gneiting and Raftery (2007) caution that coverage and sharpness represent distinct objectives, and that strictly proper scoring rules like the negative log-likelihood do not directly control coverage. Attempts to merge these goals during model training can distort estimation. Empirical findings from Tagasovska and Lopez-Paz (2019) support this view, showing that models trained to optimize proxy uncertainty losses often require additional post-hoc calibration. Likewise, Pearce et al. (2021) highlight that optimizing coverage as a training objective tends to yield unstable behavior, further reinforcing that calibration-based approaches are more robust for uncertainty quantification.

9. **Centering of CI around point forecast:** A key distinction among the interval estimation methods lies in how the prediction intervals are constructed relative to the point forecast. In ENBPI, the intervals are explicitly centered around the point predictor. This is because ENBPI constructs its prediction intervals by estimating empirical quantiles of past residuals and symmetrically adding and subtracting these from the current point forecast. The validity of this approach relies on the assumption of residual exchangeability and a well-calibrated, stable point forecaster. By contrast, both SPCI and the Sieve Bootstrap do not assume symmetry around the point prediction. In SPCI, the residuals are modeled explicitly using a separate quantile regression model (e.g., XGBoost) that predicts future residual quantiles based on recent residual history. This allows the resulting intervals to shift asymmetrically, adapting to non-Gaussian or skewed error distributions. Similarly, in the Sieve Bootstrap, future forecast paths are simulated using bootstrapped residuals from an autoregressive process. The intervals are derived from the empirical distribution of these paths, which inherently captures autocorrelation, skewness, and evolving error dynamics. Thus, the intervals from SPCI and Sieve Bootstrap may be off-center and more reflective of the underlying temporal structure in the residuals.

10. **Theoretical Guarantees for Prediction Intervals.** Lastly, below follows a description of theoretical Guarantees of the proposed uncertainty methods, and in which way they are met, or violated in this study.

Ensemble Batch Prediction Intervals (EnbPI) EnbPI, introduced by Xu and Xie (2021a), combines conformal prediction with ensemble learning. Theoretical coverage guarantees are based on the assumption that residuals from the ensemble

model are exchangeable and that the underlying relation between Y_t and X_t is linear, i.e., $Y_t = f(X_t) + \epsilon_t$ with ϵ_t exchangeable. Under these assumptions, EnbPI provides *marginal coverage* guarantees in finite samples, i.e.,

$$\mathbb{P}(Y_t \in \hat{C}t(X_t)) \geq 1 - \alpha, \tag{22}$$

where $\hat{C}t(X_t)$ is the prediction interval constructed using conformal calibration.

However, if residuals exhibit autocorrelation or structural changes over time, this exchangeability assumption breaks down, potentially leading to overly wide or miscalibrated intervals. While block bootstrapping (not adressed in this thesis) or online variants of EnbPI help partially address this, they do not explicitly model temporal dependence.

Sequential Predictive Conformal Inference (SPCI) In contrast, more recent conformal methods (e.g., Xu and Xie, 2021b) avoid making assumptions about the form of the data-generating process and instead provide distribution-free guarantees that do not rely on the linearity or exchangeability of errors. SPCI, as proposed by Xu and Xie (2021b), extends conformal prediction by estimating the conditional quantiles of residuals using machine learning methods (e.g., quantile regression forests or XGBoost). The approach leverages recent residual windows and builds prediction intervals by estimating the 5th and 95th quantiles of the conformity scores.

SPCI enjoys the following theoretical properties:

- Under exchangeability of the full data, SPCI retains *finite-sample marginal coverage*:
$$\mathbb{P}(Y_t \in \hat{C}t(X_t)) \geq 1 - \alpha. \tag{23}$$
- Under *weak dependence* (e.g., stationarity and decaying autocorrelation), and assuming a consistent quantile estimator (e.g., QRF or XGBoost quantile regression), SPCI achieves *asymptotic conditional coverage*:
$$\left|\mathbb{P}(Y_t \in \hat{C}t(X_t)|X_t) - (1 - \alpha)\right| \xrightarrow{T\to\infty} 0. \tag{24}$$

These guarantees depend on assumptions about the convergence of the quantile estimator (see Lemma 1 and Proposition 2 in Xu and Xie (2021b)) and the residual dependence structure.

Sieve Bootstrap Prediction Intervals The Sieve Bootstrap, originally proposed by Bühlmann (1998), constructs prediction intervals by resampling from a fitted autoregressive (AR) model on the residuals of a point forecast. It preserves serial dependence by assuming that residuals can be well-approximated by a finite-order AR process:

$$\hat{\epsilon}t = \sum j = 1^p a_j \hat{\epsilon}_{t-j} + \eta_t, \quad \eta_t \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}^2). \tag{25}$$

Synthetic residual series are generated and added to the point forecast to produce empirical prediction intervals.

Unlike conformal methods, the Sieve Bootstrap does not provide formal coverage guarantees unless additional assumptions hold. For instance:

- The residual-generating process must be stationary and weakly dependent (mixing conditions).
- The AR model used for residuals must be well-specified, and the number of lags p must increase slowly with the sample size.
- The bootstrapped sample must converge to the true distribution asymptotically.

Under these assumptions, Bühlmann (1998) showed that the Sieve Bootstrap is asymptotically valid for constructing prediction intervals. However in this study, as in most empirical cases, these assumptions are (at least partly) unmet.

6 Conclusion

This thesis set out to illustrate the often-overlooked challenges in forecasting highly volatile and unpredictable data—challenges where standard assumptions such as homoskedasticity, stationarity, or residual exchangeability do not hold. Accurately forecasting electricity prices, whilst being the actual goal of this study, ultimately functioned as a suitable emperical challenge for a scala of new forecasting and uncertainty estimation techniques. While the final results may not have achieved perfect coverage or outstanding forecast accuracy, I am certain that with extended research on feature engineering, cross validation, hyperparameter tuning, the uncertainty methods developed here could near the 90% target in the near future. This could effect not only electricity price forecasting, but also any other empirical regression challenge in which the data generation process is not pitch perfect.

The most promising contribution of this study is the introduction of the Sequential Sieve Bootstrap Predictive Inference (SSBPI) algorithm, which addresses both residual autocorrelation and the approximate exchangeability of forecast errors by adapting confidence intervals online in response to evolving error dynamics. In the future, the performance of this Algorithm can be further established by combining it with more optimal point forecasters.

References

Abbasimehr, H., Paki, R., and Bahrini, A. (2023). A novel xgboost-based featurization approach to forecast renewable energy consumption with deep learning models. *Sustainable Computing: Informatics and Systems*, 38.

Angelopoulos, A. N. and Bates, S. (2021). A gentle introduction to conformal prediction. *arXiv preprint arXiv:2107.07511*.

Bampoulas, A., Pallonetto, F., Mangina, E., and Finn, D. P. (2023). A bayesian deep-learning framework for assessing the energy flexibility of residential buildings with multicomponent energy systems. *Applied Energy*, 348.

Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics and Data Analysis*, 120:70–83.

Billé, A. G., Gianfreda, A., Grosso, F. D., and Ravazzolo, F. (2023). Forecasting electricity prices with expert, linear, and nonlinear models. *International Journal of Forecasting*, 39:570–586.

Bojer, C. S. (2022). Understanding machine learning-based forecasting methods: A decomposition framework and research opportunities. *International Journal of Forecasting*, 38:1555–1561.

Bojer, C. S. and Meldgaard, J. P. (2020). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603.

Bühlmann, P. (1998). Sieve bootstrap for time series. *Bernoulli*, 4(4):373–397.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 785–794. Association for Computing Machinery.

Chen, T., Guestrin, C., and contributors (2024). Xgboost documentation: Parameter tuning. https://xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html. Accessed: 2025-05-23.

ENTSO-E (2025). Transparency platform. https://www.entsoe.eu. Accessed: April 2025.

EPEX SPOT SE (2024). European power exchange - market information. Accessed: 2025-04-22.

Fokam, C. T., Jentsch, C., Lang, M., and Pauly, M. (2024). Ar-sieve bootstrap for the random forest and a simulation-based comparison with rangerts time series prediction.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Friedrich, M., Smeekes, S., and Urbain, J. P. (2020). Autoregressive wild bootstrap inference for nonparametric trends. *Journal of Econometrics*, 214:81–109.

Friedrich, T. and Huber, F. (2024). Bootstrap confidence bands for time-varying parameter models. *Econometrics and Statistics*. Forthcoming.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Goulet Coulombe, P., Leroux, M., Stevanovic, D., and Surprenant, S. (2019). How is machine learning useful for macroeconomic forecasting?. *University of Pennsylvania, Université du Québec à Montréal*.

Hong, T., Pinson, P., Wang, Y., Weron, R., Yang, D., and Zareipour, H. (2020). Energy forecasting: A review and outlook.

Hothorn, T., Kneib, T., and Bühlmann, P. (2012). Conditional transformation models.

Huang, S., Shi, J., Wang, B., An, N., Li, L., Hou, X., Wang, C., Zhang, X., Wang, K., Li, H., Zhang, S., and Zhong, M. (2024). A hybrid framework for day-ahead electricity spot-price forecasting: A case study in china. *Applied Energy*, 373.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

Kapoor, G. and Wichitaksorn, N. (2023). Electricity price forecasting in new zealand: A comparative analysis of statistical and machine learning models with feature selection. *Applied Energy*, 347.

Krupa, J. and Jones, C. (2013). Black swan theory: Applications to energy market histories and technologies. *Energy Strategy Reviews*, 1:286–290.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Lago, J., Marcjasz, G., Schutter, B. D., and Weron, R. (2021). Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark.

Laitsos, V., Vontzos, G., Bargiotas, D., Daskalopulu, A., and Tsoukalas, L. H. (2024). Data-driven techniques for short-term electricity price forecasting through novel deep learning approaches with attention mechanisms. *Energies*, 17.

Maleki, N., Lundström, O., Musaddiq, A., Olsson, T., Ahlgren, F., and Jeansson, J. (2024). Future energy insights: Time-series and deep learning models for city load forecasting. *Applied Energy*, 374.

März, A. (2019). Xgboostlss – an extension of xgboost to probabilistic forecasting. *arXiv preprint*.

Netherlands Authority for Consumers and Markets (ACM) (2024). Energy markets regulation in the netherlands. Accessed: 2025-04-22.

O'Connor, C., Bahloul, M., Rossi, R., Prestwich, S., and Visentin, A. (2025). Conformal prediction for electricity price forecasting in the day-ahead and real-time balancing market. *arXiv preprint*.

Pearce, T., Leibfried, F., Zisserman, A., Kohli, P., and Tosi, A. (2021). Uncertainty in neural networks: Approximately bayesian ensembling. In *International Conference on Machine Learning (ICML)*.

Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 54(3):507–554.

Romano, Y., Patterson, E., and Candès, E. J. (2019). Conformalized quantile regression.

Tagasovska, N. and Lopez-Paz, D. (2019). Single-model uncertainties for deep learning. *arXiv preprint arXiv:1905.08160*.

Taleb, N. N. (2007). *The Black Swan: The Impact of the Highly Improbable*. Random House, New York.

TenneT TSO B.V. (2024). Tennet electricity market information. Accessed: 2025-04-22.

Tightiz, L., Yoo, J., and Al-Shibli, W. K. (2024). Strategic enhancements in electricity price forecasting: The role of xgboost and error correction features.

Velthoen, J., Dombry, C., Cai, J. J., and Engelke, S. (2023). Gradient boosting for extreme quantile regression. *Extremes*, 26:639–667.

Wen, Q. and Liu, Y. (2025). Feature engineering and selection for prosumer electricity consumption and production forecasting: A comprehensive framework. *Applied Energy*, 381:125176.

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future.

XGBoost Developers (2022). *XGBoost Python Package — xgboost 3.0.0 documentation*. ReadTheDocs.io. Accessed April 23, 2025.

Xu, C. and Xie, Y. (2021a). Conformal prediction interval for dynamic time-series. Technical report.

Xu, C. and Xie, Y. (2021b). Sequential predictive conformal inference for time series. Technical report, Georgia Institute of Technology.

Zhang, L., Bian, W., Qu, W., Tuo, L., and Wang, Y. (2021). Time series forecast of sales volume based on xgboost. In *Journal of Physics: Conference Series*, volume 1873. IOP Publishing Ltd.

A Appendix

A.1 Train-test selection rationale

In this section I briefly explain why I chose the train and test set at the length of 1 year and 3 months, from Januari 2023 until April 2024. For this the entire price period of 2021-2025 is plotted below in Figure 10.

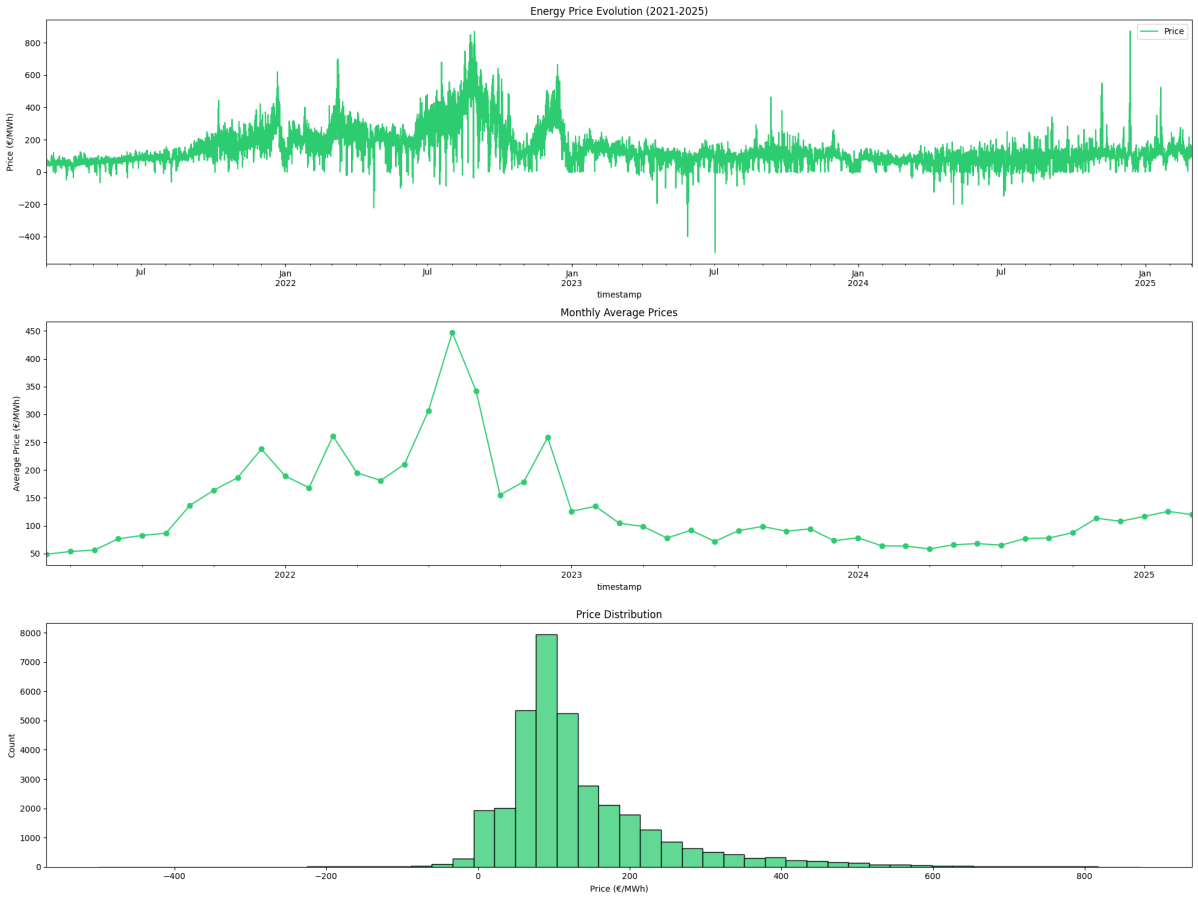

Figure 10: Price analysis 4 years (2021-2025)

The first notable observation is the period of extreme volatility and the overall surge in energy prices in 2022. This coincided with the outbreak of the war in Ukraine, which followed the COVID-19 pandemic and had a significant impact on electricity prices. Such dramatic effects caused by external shocks are often referred to as structural shifts, breaks, or, more recently, black swan events— referring to highly improbable yet impactful occurrences, formalized in the Black Swan Theory (BST) book Taleb (2007). BST highlights the low probability and unpredictability of highly impactful events. Black swan events are not new to electricity markets. For instance, Krupa and Jones (2013) conducted a detailed review of academic literature and government reports on black swans, focusing on historical disruptions in energy security and technological innovation. They advocated the need for a more conservative approach to energy forecasting. Policy recommendations include maintaining skepticism toward long-term projections, avoiding the selection of specific technology "winners," and improving valuation systems for environmental externalities. Notably, this study was set in 2013, before the surge of renewable energy, increasing seasonal dependence in the exogenous variables, and not in the Netherlands per se. Yet, learning from this, I consider it wiser to not use the data from a black swan event, to forecast short-term (14-38 hours) price forecasts, which is the objective of this study. Unlike long-term forecasts that can be heavily influenced by structural shifts (such as geopolitical crises or major policy changes), short-term price movements are more influenced by real-time supply and demand fluctuations, weather conditions,

grid stability, and market sentiment. These factors are **probably** sufficiently captured by using only the data after the extreme period of 2022. Therefore in this study, the training set is anchored at January 2023-April as shown in Figure 2.

A.2 Feature description

This appendix provides an overview of the feature engineering process I applied to the electricity price forecasting dataset.

Time-based features were extracted from the datetime index to capture periodic and seasonal effects. These include hour, day of the week, day of the month, month, quarter, year, and ISO week of the year. To better model cyclical behavior, I applied sinusoidal encodings to hour, month, and weekday values using sine and cosine transformations. In addition, binary flags were created to indicate parts of the day (morning, afternoon, evening, night) and whether the timestamp fell on a weekend. This approach was also aplied in for example (Laitsos et al., 2024)

Holiday features were constructed using the `holidays.NL()` package. Each timestamp was labeled as a holiday, the day before a holiday, or the day after a holiday. These indicators help the model adjust for known calendar effects on electricity demand and market behavior.

To capture temporal dependencies in prices, I included **lagged price features** with hourly lags ranging from 1 to 168 hours (1 week). These lagged variables allow the model to learn from recent historical prices, which often influence current market behavior. A similar methodology was adopted by Lago et al. (2021), who incorporated temporal lags and applied feature selection strategies tailored to the specific needs of their forecasting models. In this study, I follow a comparable approach by experimenting with multiple feature selection techniques, including SHAP value analysis, recursive feature elimination with cross-validation (RFECV), and statistical criteria such as BIC. This optimization step is used to adapt the feature set to each model type and improve forecasting performance.

I also calculated **rolling statistics** of price over several time windows (3h, 6h, 12h, 24h, 48h, 72h, and 168h). For each window, I computed the rolling mean and rolling standard deviation, which help describe local trends and short-term volatility.

In addition, I computed **price differences** over various time steps to capture momentum and recent changes in the market. These include differences over 1, 2, 3, 24, 48, 72, and 168-hour windows.

All these features where developed following the review on feature engineering for electricity consumption and production forecasting of Wen and Liu (2025), assuming they would apply to prices to0.

Lagged features for generation and forecast variables were included for wind, solar, coal, consumption, consumption forecasts, and renewable energy forecasts. For each of these variables, I included lagged values at 1h, 2h, 3h, 6h, 12h, 24h, 48h, and 168h intervals. These features provide context on supply and demand dynamics, helping the model anticipate how renewable variability and demand forecasts affect future prices.

All features were concatenated into a single multivariate dataset. I removed duplicate columns, replaced infinite values with NaN, and dropped any rows with missing values to ensure a clean and consistent feature matrix.

A.3 Autoregressive (AR(p)) recursive, fixed parameters forecasting

The fixed AR model is trained once on historical data up to a specific point in time (January 1, 2024), and then used to make all future predictions without further re-estimation. Formally, I estimate:

$$y_t = \beta_0 + \sum_{i=1}^{p} \beta_i y_{t-i} + \epsilon_t \quad \text{for all } t \leq t_{\text{train end}} \tag{26}$$

where:

- y_t is the electricity price at time t,

- p is the selected lag order (chosen once during training),
- $\beta_0, \beta_1, \ldots, \beta_p$ are the fixed coefficients,
- ϵ_t are the residual errors.

Forecasts at time t_{forecast} for a horizon h hours ahead are computed recursively:

$$\hat{y}_{t_{\text{forecast}}+h} = \beta_0 + \sum_{i=1}^{p} \beta_i \tilde{y}_{t_{\text{forecast}}+h-i} \tag{27}$$

where $\tilde{y}_t$ are observed values if available, otherwise recursively predicted values.
This model produces a flat line, as it is severely overfitting and not adapting to regime changes.

A.4 Conformalized quantile regression

Conformal Prediction (CP) is a distribution-free approach for constructing prediction intervals that are valid under minimal assumptions. Given a training set $(x_i, y_i)_{i=1}^{n}$, a predictive model $\hat{y} = f(x)$ is trained. The following steps are used to construct conformal prediction intervals:

1. Split the data into a training set and a calibration set.
2. Train the model f on the training set.
3. Generate predictions $\hat{y}_i$ on the calibration set.
4. Calculate residuals (non-conformity scores) as:

$$r_i = |y_i - \hat{y}_i| \tag{28}$$

Lastly, one sorts the residuals and determine the quantile $Q_{1-\alpha}$ for the desired confidence level $1-\alpha$. The conformal prediction interval for a new observation is then:

$$[\hat{y}_{n+1} - Q_{1-\alpha},\ \hat{y}_{n+1} + Q_{1-\alpha}] \tag{29}$$

This distribution free approach that relies on non conformity scores through residuals is the backbone of the more sophisticated and time series adapted uncertainty estimation methods of this thesis.
Quantile Regression (QR) is a regression technique that directly models specified quantiles of the response variable, such as the lower τ_l and upper τ_h quantiles. For a model f, the quantile predictions are denoted as $\hat{y}^{low}$ and $\hat{y}^{high}$. this can typically be done using a quantile loss or in other words , pinball loss function.
The pinball loss for a single prediction $\hat{y}_\tau$ and true value y is defined as:

$$L_\tau(y, \hat{y}_\tau) = \begin{cases} \tau \cdot (y - \hat{y}_\tau), & \text{if } y \geq \hat{y}_\tau \\ (1-\tau) \cdot (\hat{y}_\tau - y), & \text{if } y < \hat{y}_\tau \end{cases} \tag{30}$$

Alternatively, it can be written more compactly using the indicator function $\mathbb{I}\{\cdot\}$:

$$L_\tau(y, \hat{y}_\tau) = (\tau - \mathbb{I}\{y < \hat{y}_\tau\}) \cdot (y - \hat{y}_\tau) \tag{31}$$

The loss is asymmetric: it penalizes under-predictions (when $y > \hat{y}_\tau$) and over-predictions (when $y < \hat{y}_\tau$) differently, with the weights determined by the quantile level τ. For example, with $\tau = 0.95$, the model is more tolerant of overestimates than underestimates, which encourages predictions to cover the upper tail of the distribution.

Minimizing the expected pinball loss ensures that the model learns to predict the conditional quantile corresponding to τ. This makes the pinball loss especially useful for constructing prediction intervals when estimating multiple quantiles (e.g., 5th and 95th percentiles).

Combining these two approaches, Romano et al. (2019) proposed to apply conformal prediction following up on quantile regression to recallibrate the upper and lower quantiles from the regression with the non-comformity scores. The process is as follows:

Train a quantile regression model to predict $\hat{y}^{low}$ and $\hat{y}^{high}$.
Calculate non-conformity scores on the calibration set:

$$r_i^{low} = \hat{y}_i^{low} - y_i \tag{32}$$

$$r_i^{high} = y_i - \hat{y}_i^{high} \tag{33}$$

Determine the quantiles Q_{low} and Q_{high} of these scores at the chosen confidence level:

$$Q_{low} = Q_\alpha(r_i^{low}) \tag{34}$$

$$Q_{high} = Q_\alpha(r_i^{high}) \tag{35}$$

Adjust the initial quantile regression predictions to obtain the final conformal prediction interval:

$$[\hat{y}^{low} - Q_{low}, \hat{y}^{high} + Q_{high}] \tag{36}$$

Although this approach was tested in an initial state of the research, the results did not show much accuracy. Xu and Xie (2021b) also mentioned that the downside of quantile regression (with or without conformal prediction require special hyper-parameter tuning and can be computationally expensive, as the pinball loss depends on α. In contrast, SPCI, ENbPI and online Sieves bootstrap is compatible with any user-specified point prediction model, remain distribution- free, and provides coverage guarantees under assumptions named in Section 5. Furthermore, the additional split conformal prediction step nesesscitates a holdout callibration set for constructing the conformity scores.

Algorithm 6: Forecasting and Conformal Prediction Pipeline

Input: Full dataset $\mathcal{D}$ from Jan 2023 to June 2024, forecast horizon h, coverage level $1-\alpha$
Output: Prediction intervals $\tilde{C}_{t+h}$ for t in test set

Step 1: Hyperparameter tuning
Split $\mathcal{D}_{\text{train}}$:= Jan 2023 to Jan 2024 into 80/20 train-validation split;
Use `hyperopt` to optimize XGBoost hyperparameters on validation fold using RMSE;
Step 2: Feature selection
Use rolling window evaluation on Jan 2024– March 2024 to evaluate performance across folds;
Select top 100 features based on average feature importance across folds;
Step 3: Final model training
Fix hyperparameters and features from Step 1 and Step 2;
Retrain final XGBoost model on entire period Jan 2023–March 2024;
Step 4: Conformal calibration
Define calibration set $\mathcal{I}_2$:= March–April 2024 for Conformalized Quantile Regression (not used in any tuning or selection);
for *each* $t \in \mathcal{I}_2$ **do**
    Predict lower and upper quantiles:

$$\hat{q}^{(h)}_{\alpha/2}(x_t), \quad \hat{q}^{(h)}_{1-\alpha/2}(x_t)$$

    Compute conformity score:

$$E_t^{(h)} = \max\left\{\hat{q}^{(h)}_{\alpha/2}(x_t) - y_{t+h},\ y_{t+h} - \hat{q}^{(h)}_{1-\alpha/2}(x_t)\right\}$$

Compute empirical quantile:

$$Q_{1-\alpha} = \text{Quantile}_{(1-\alpha)(1+\frac{1}{|\mathcal{I}_2|})}\left(\{E_t^{(h)} : t \in \mathcal{I}_2\}\right)$$

Step 5: Conformalized prediction
for *each test time* $t >$ *April 2024* **do**
    Predict $\hat{q}^{(h)}_{\alpha/2}(x_t)$ and $\hat{q}^{(h)}_{1-\alpha/2}(x_t)$;
    Construct conformalized interval:

$$\tilde{C}_{t+h} = \left[\hat{q}^{(h)}_{\alpha/2}(x_t) - Q_{1-\alpha},\ \hat{q}^{(h)}_{1-\alpha/2}(x_t) + Q_{1-\alpha}\right]$$

return $\tilde{C}_{t+h}$ *for all* t *in test set*

A.5 Prediction interval construction: SPCI vs. ENBPI

In this section, I detail the theoretical formulation behind the prediction intervals generated using Sequential Predictive Conformal Inference (SPCI) and Ensemble Neural Bootstrap Prediction Intervals (ENBPI). While both approaches aim to provide valid uncertainty estimates in time series forecasting, they differ fundamentally in how residuals are used and how prediction intervals are centered.

SPCI, developed by Xu and Xie (2021b) leverages the temporal dependence among forecast residuals to improve interval calibration in time series settings. The key idea is to use a quantile regression model trained on past residual sequences to predict the conditional distribution of future residuals. For a given forecast horizon h, we define the residual at time t as:

$$\hat{\varepsilon}_t = y_{t+h} - \hat{f}(x_t),$$

where $\hat{f}$ is the point forecasting model (e.g., XGBoost) trained to minimize RMSE.

To model the uncertainty, a quantile regression model Q_q is fit on a dataset of the form:

$$\tilde{x}_t := [\hat{\varepsilon}_{t-w}, \ldots, \hat{\varepsilon}_{t-1}], \quad \tilde{y}_t := \hat{\varepsilon}_t,$$

using a sliding window of width w over the training residuals. The predicted residual quantiles are used to construct the prediction interval as:

$$\tilde{C}_{t+h} = \left[\hat{f}(x_t) + \hat{r}_t^{\text{low}},\ \hat{f}(x_t) + \hat{r}_t^{\text{up}}\right],$$

where $\hat{r}_t^{\text{low}} = Q_{\hat{\beta}}(\tilde{x}_t)$ and $\hat{r}_t^{\text{up}} = Q_{1-\alpha+\hat{\beta}}(\tilde{x}_t)$ for some optimized $\hat{\beta} \in [0, \alpha]$. The centering relative to the median ($Q_{0.5}$) ensures sharper and more adaptive intervals.

ENBPI, also by the talented Xu and Xie (2021a) takes a different approach by creating an ensemble of point predictors using bootstrap resampling. Each model in the ensemble is trained on a different resampled subset of the training data. For each time step, the prediction is the median of the ensemble predictions:

$$\hat{y}_{t+h}^{\text{median}} = \text{median}\left\{\hat{f}_b(x_t)\right\}_{b=1}^{B}.$$

To quantify uncertainty, ENBPI uses the empirical distribution of *out-of-bag* (OOB) residuals:

$$r_i^{\text{oob}} = \left|y_i - \bar{f}^{\text{oob}}(x_i)\right|,$$

where $\bar{f}^{\text{oob}}(x_i)$ is the average prediction from ensemble members that did not include x_i in their bootstrap sample.

The prediction interval is then computed as a symmetric expansion around the median prediction using the $(1-\alpha)$ empirical quantile of the OOB residuals:

$$\tilde{C}_{t+h} = \left[\hat{y}_{t+h}^{\text{median}} - q,\ \hat{y}_{t+h}^{\text{median}} + q\right],$$

where $q = \text{Quantile}_{1-\alpha}(\{r_i^{\text{oob}}\})$.

While both methods aim to deliver calibrated prediction intervals:

- **SPCI** is adaptive and model-free, and can track changing residual distributions by explicitly learning a mapping from past residuals to future ones.
- **ENBPI** is simpler to implement and relies on bootstrapped model variation and empirical quantiles rather than residual modeling.

Notably, the "low-mid" centering used in SPCI is unnecessary for ENBPI, as the latter constructs symmetric intervals without explicit quantile regression. Thus, the choice between SPCI and ENBPI depends on the application: SPCI offers finer control and adaptivity, whereas ENBPI is robust and less sensitive to hyperparameter tuning.

A.6 Simplified online algorithms for point forecasts

In contrast to the XGBoost model, the RFECV Linear model and AR models are retrained for each rolling forecast window. This is necessary because the models´ strength lies in dynamic feature/lag selection, given the strong seasonal and structural variations in electricity prices. Thus, instead of updating on-the-fly, residuals are recalculated for each test fold, and intervals are constructed in batch mode. This is thus oficially not ¨online¨ , since in the online version that is applied in XGBoost, each prediction is made consequtively and subsequently, the uncertainty is evaluated by one of the methods at that specific time point. Here, for each uncertainty method, the exact procedures are described.

Simplified ENBPI Implementation for the AR and Linear regression models. In traditional ENBPI, multiple models are trained on bootstrapped versions of the training set to generate a prediction ensemble. But for AR, the entire input is just a single time series, so bootstrapping would require perturbing or resampling from this temporal structure, which introduces complexity and instability. Furthermore, for the RFECV Linear regression, which is based on cross validation for feature selection, the additional bootstrapping would induce significant computational overhead. Instead, for both the AR model and the linear regression, I simulate the ENBPI idea by using the POOS rolling-window approach to produce point forecasts and then maintain an online buffer of residuals (i.e., past absolute forecast errors). To construct prediction intervals, I apply a moving window over this residual buffer and estimate upper quantiles to adjust the interval width. The procedure is explained in Algorithm 7.

Algorithm 7: Simplified Online ENBPI for Benchmark models

Input: Full dataset $\mathcal{D}$ (Jan 2023 – Jun 2024), forecast horizon h, target coverage $1-\alpha$, residual window size w

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t in test set

Step 1: Rolling Point Forecasts

for *each test fold with forecast start time* t **do**

- Define training window $\mathcal{D}^{(t)}_{\text{train}} = \{y_{t-W}, \ldots, y_{t-1}\}$;
- Define test window $\mathcal{D}^{(t)}_{\text{test}} = \{y_t, \ldots, y_{t+S-1}\}$;
- **for** *each horizon* $h \in \mathcal{H}$ **do**
  - Construct lag and time features for horizon h;
  - Fit rolling point predictor $f_t^{(h)}$ on training data:
    - Either a Linear Regression with RFECV-selected features,
    - Or an AR model with BIC-selected lags;
  - Predict $\hat{y}_{t+h}$ on test data and store $(\hat{y}_{t+h}, y_{t+h})$ in log $\mathcal{F}_h$;

Step 2: Initialize Residual Buffers for *each* $h \in \mathcal{H}$ **do**

- Prefill residual buffer $\mathcal{E}_h = [r_t]_{t<t_0}$ with $r_t = y_t - \hat{y}_t$;

Step 3: Online Forecasting with Interval Updates;

for *each new test sample* $(x_t, \hat{y}_{t+h})$ **do**

- Estimate prediction interval width:

$$q_\alpha = \text{Quantile}_{1-\alpha}(\varepsilon)$$

- Construct prediction interval:

$$\tilde{C}_t = [\hat{y}_t - q_\alpha,\ \hat{y}_t + q_\alpha]$$

- Observe actual y_t and compute new residual:

$$\hat{\varepsilon}_t = |y_{t+h} - \hat{y}_{t+h}|$$

- Update buffer: append $\hat{\varepsilon}_t$ and retain only the most recent w values;

Step 4: Evaluation;

Compute empirical coverage and average interval width over all t in test set;

return $\tilde{C}_{t+h}$ *and evaluation metrics*

Simplified SPCI Implementation for the AR and Linear regression models. While the SPCI method follows the same core principle across all models—using a residual buffer to fit quantile models that adaptively construct prediction intervals—the application differs due to the nature of each point forecaster. For the XGBoost model, the point predictor is trained once on the full training set, and only the residual quantile models are updated online using a fixed-length residual buffer, making the entire pipeline online after initial training. In contrast, the Benchmark models are refitted at each

fold using a rolling historical window. The residuals from these rolling forecasts are then filled in a residual buffer used for SPCI-based interval calibration online.

Algorithm 8: Simplified Online Sequential Predictive Conformal Inference (SPCI) for Benchmark Models

Input: Full dataset $\mathcal{D}$ (Jan 2023 – Jun 2024), forecast horizons $\mathcal{H}$, target coverage $1-\alpha$, residual window size w

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t in test set

Step 1: Rolling Point Forecasts

for *each test fold with forecast start time* t **do**

- Define training window $\mathcal{D}^{(t)}_{\text{train}} = \{y_{t-W}, \ldots, y_{t-1}\}$;
- Define test window $\mathcal{D}^{(t)}_{\text{test}} = \{y_t, \ldots, y_{t+S-1}\}$;
- **for** *each horizon* $h \in \mathcal{H}$ **do**
  - Construct lag and time features for horizon h;
  - Fit rolling point predictor $f_t^{(h)}$ on training data:
    - Either a Linear Regression with RFECV-selected features,
    - Or an AR model with BIC-selected lags;
  - Predict $\hat{y}_{t+h}$ on test data and store $(\hat{y}_{t+h}, y_{t+h})$ in log $\mathcal{F}_h$;

Step 2: Initialize Residual Buffers

for *each* $h \in \mathcal{H}$ **do**

- Prefill residual buffer $\mathcal{E}_h = [r_t]_{t<t_0}$ with $r_t = y_t - \hat{y}_t$;

Step 3: Online SPCI Calibration (in time order)

for *each horizon* $h \in \mathcal{H}$ **do**

- **for** *each forecast* $(t+h, \hat{y}_{t+h}, y_{t+h})$ *in* $\mathcal{F}_h$*, in chronological order* **do**
  - **if** $|\mathcal{E}_h| \geq w$ ***and*** $(t \bmod f = 0)$ **then**
    - Construct residual sequences $\{\mathcal{E}_h^{(i)}\}_{i=1}^{N}$ of length w;
    - Fit quantile regressors to estimate:

$$\hat{r}_t^q = Q_q(\mathcal{E}_h | \text{recent sequence}) \quad \text{for } q = \alpha/2,\ 0.5,\ 1-\alpha/2$$

  - **if** *quantile models available* **then**
    - Predict residual quantiles $\hat{r}_t^{\alpha/2}$, $\hat{r}_t^{0.5}$, $\hat{r}_t^{1-\alpha/2}$;
    - Construct prediction interval:

$$\tilde{C}_{t+h} = \left[\hat{y}_{t+h} + (\hat{r}_t^{\alpha/2} - \hat{r}_t^{0.5}),\ \hat{y}_{t+h} + (\hat{r}_t^{1-\alpha/2} - \hat{r}_t^{0.5})\right]$$

  - **else**
    - Use fallback interval: $[\hat{y}_{t+h} - 15,\ \hat{y}_{t+h} + 15]$
  - Observe true y_{t+h} and compute residual $r_{t+h} = y_{t+h} - \hat{y}_{t+h}$;
  - Append r_{t+h} to buffer $\mathcal{E}_h$ (truncate to most recent R if needed);

Step 4: Evaluation (per horizon)

for *each* $h \in \mathcal{H}$ **do**

- Compute empirical coverage: fraction of $y_{t+h} \in \tilde{C}_{t+h}$;
- Compute average interval width: $\frac{1}{n}\sum(\text{upper}_{t+h} - \text{lower}_{t+h})$

return *Prediction intervals* $\tilde{C}_{t+h}$ *and evaluation metrics*

Online Sieves bootstrap Implementation for the AR and Linear regression models. Same as for ENbPI and SPCI, the residuals are precollected from the entire test set through batch prediction in a first step. Then using a residual buffer adapted to this specific algorithm (200 residuals to accommodate enough lags for the AR model). The complete procedure is described in

Algorithm 9.

Algorithm 9: Rolling Forecasts and Sieve Bootstrap to benchmark models

Input: Training dataset $\mathcal{D}_{\text{train}}$, test dataset $\mathcal{D}_{\text{test}}$, forecast horizons $\mathcal{H} = \{14, 24, 38\}$, rolling window size W, test step size S, buffer size R, bootstrap samples B, significance level α

Output: Prediction intervals $\tilde{C}_{t+h}$ for each t and $h \in \mathcal{H}$

Step 1: Rolling Point Forecasts

for *each test fold with forecast start time* t **do**

- Define training window $\mathcal{D}_{\text{train}}^{(t)} = \{y_{t-W}, \ldots, y_{t-1}\}$;
- Define test window $\mathcal{D}_{\text{test}}^{(t)} = \{y_t, \ldots, y_{t+S-1}\}$;
- **for** *each horizon* $h \in \mathcal{H}$ **do**
  - Construct lag and time features for horizon h;
  - Fit rolling point predictor $f_t^{(h)}$ on training data:
    - Either a Linear Regression with RFECV-selected features,
    - Or an AR model with BIC-selected lags;
  - Predict $\hat{y}_{t+h}$ on test data and store $(\hat{y}_{t+h}, y_{t+h})$ in log $\mathcal{F}_h$;

Step 2: Online Interval Construction with Sieve Bootstrap

for *each horizon* $h \in \mathcal{H}$ **do**

- Let $\mathcal{F}_h = \{(t_i, \hat{y}_{t_i+h}, y_{t_i+h})\}_{i=1}^{N}$;
- Initialize residual buffer $\mathcal{E}$ using $\hat{y} - y$ differences in early test folds;
- **for** *each time* t_i *in* $\mathcal{F}_h$ **do**
  - Compute residual buffer $\mathcal{B}_{t_i} = \mathcal{E}[t_i - R : t_i]$;
  - **if** $|\mathcal{B}_{t_i}| < 20$ **then**
    - Skip to next t_i
  - **// Fit AR model on residuals**
  - Fit initial AR model on $\mathcal{B}_{t_i}$ with max lags;
  - Select lags using BIC-based criterion;
  - Fit final AR(ϕ) model and extract innovations η_{t_i};
  - **// Sieve Bootstrap**
  - Simulate B bootstrap residual paths $\{\tilde{e}_{1:h}^{(b)}\}_{b=1}^{B}$ from AR(ϕ) with sampled η_{t_i};
  - Extract h-step residuals $\tilde{e}_h^{(b)}$;
  - **// Construct prediction interval**
  - $$\text{lower}_{t_i+h} = \hat{y}_{t_i+h} + \text{Quantile}_{\alpha/2}(\{\tilde{e}_h^{(b)}\})$$
    $$\text{upper}_{t_i+h} = \hat{y}_{t_i+h} + \text{Quantile}_{1-\alpha/2}(\{\tilde{e}_h^{(b)}\})$$
  - Store interval $(\text{lower}_{t_i+h}, \text{upper}_{t_i+h})$;
  - Append residual $y_{t_i+h} - \hat{y}_{t_i+h}$ to buffer $\mathcal{E}$;

Step 3: Evaluation

for *each* $h \in \mathcal{H}$ **do**

- Compute empirical coverage over test set;
- Compute average interval width;

return $\tilde{C}_{t+h}$ *and evaluation metrics*

A.7 Point prediction graphs; predicted and actual values over time

This sections gives a visual presentation of all the predictions made by the model. The first 3 images are of the point forecasts made by the 3 models over time. The difference between Figure 11 and Figure 12 the (XGBoost and Linear regression) and Figure 13 (AR) is that the first two are trained each week on

a years data and make a direct forecast for the week after, which is visible in each of the subplots. The dates are hard to read, but the images are kept at this size for showcasing purposes of the ¨shape¨of the predictions. However, each ticket on the xaxis represents a day in the (weekly) fold. The AR model, due to it being a recursive model, had to be retrained every day on the years test data and predicts the DA price value of the 14, 24 and 38 horizon ahead. These predictions are visualized in consequtive order in Figure 13. No folds are shown here since each fold would represent a single day.

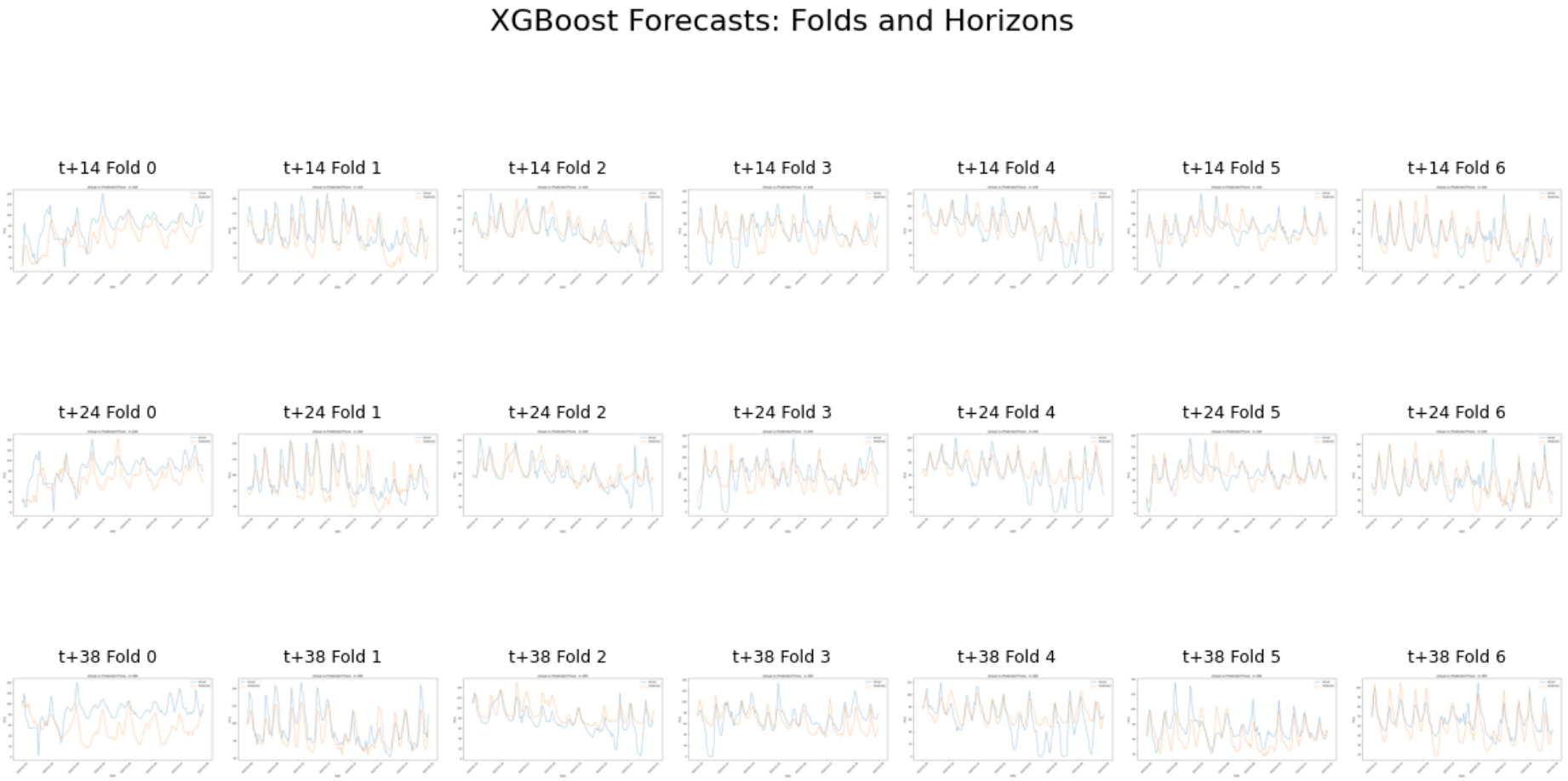

Figure 11: XGBoost forecast per fold and horizon in test set of Jan2024 -March 2024

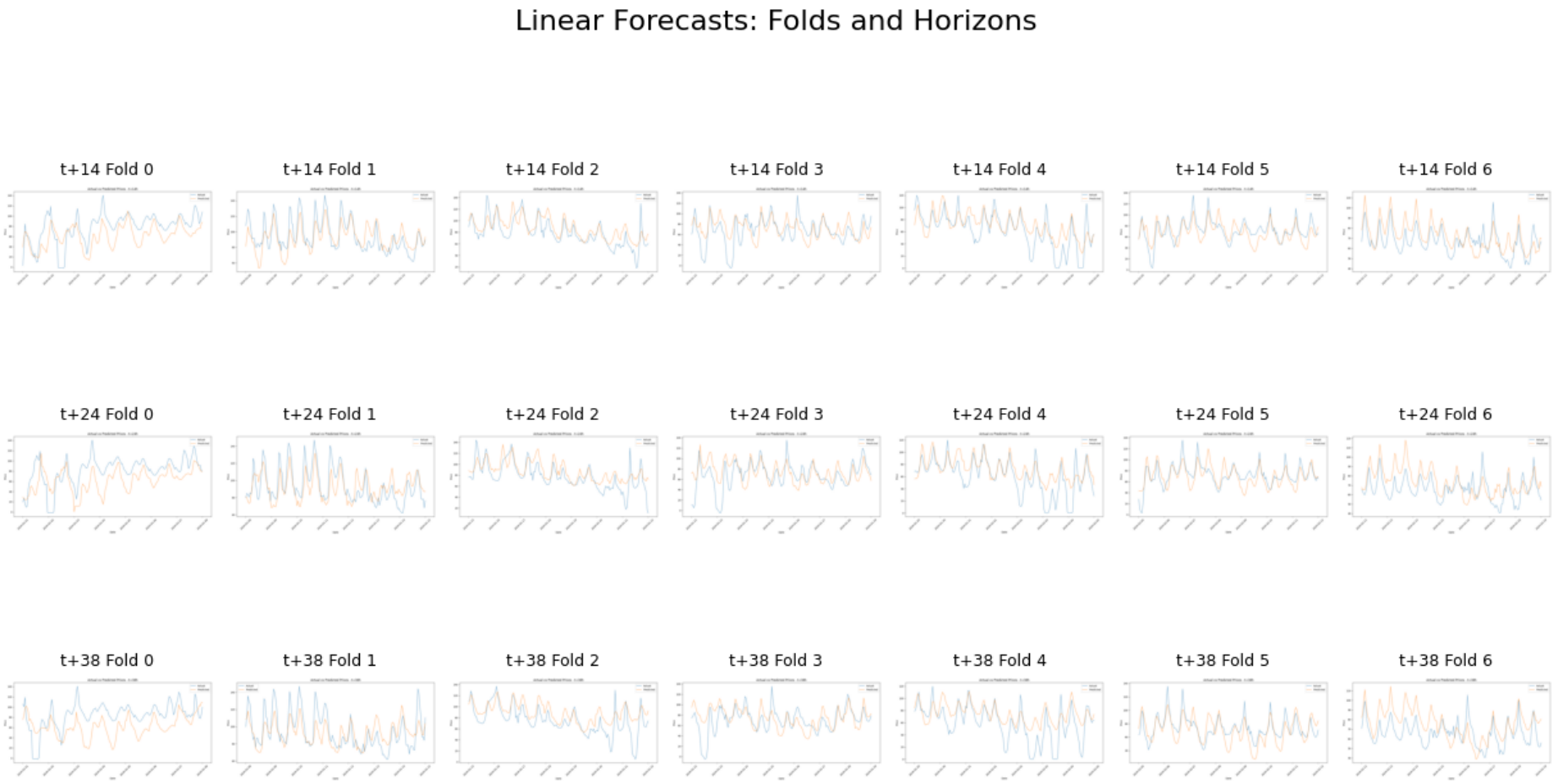

Figure 12: Direct forecast Linear regression, per fold and horizon in test set of Jan2024 -March 2024

Figure 13: Recursive AR predictions, per horizon in test set of Jan2024 -March 2024