

Article

Energy Markets Forecasting. From Inferential Statistics to Machine Learning: The German Case

Emma Viviani ^{1,*}, Luca Di Persio ^{1,†}  and Matthias Ehrhardt ^{2,†} 

¹ Department of Computer Science, College of Mathematics, University of Verona, 37134 Verona, Italy; luca.dipersio@univr.it

² Department of Applied Mathematics and Numerical Analysis, University of Wuppertal, 42119 Wuppertal, Germany; ehrhardt@uni-wuppertal.de

* Correspondence: emma.viviani@studenti.univr.it

† These authors contributed equally to this work.

Abstract: In this work, we investigate a probabilistic method for electricity price forecasting, which overcomes traditional ones. We start considering statistical methods for point forecast, comparing their performance in terms of efficiency, accuracy, and reliability, and we then exploit Neural Networks approaches to derive a hybrid model for probabilistic type forecasting. We show that our solution reaches the highest standard both in terms of efficiency and precision by testing its output on German electricity prices data.

Keywords: electricity price; statistical method; autoregressive; probabilistic forecast; neural network



Citation: Viviani, E.; Di Persio, L.; Ehrhardt, M. Energy Markets Forecasting. From Inferential Statistics to Machine Learning: The German Case. *Energies* **2021**, *14*, 364. <https://doi.org/10.3390/en14020364>

Received: 21 December 2020

Accepted: 5 January 2021

Published: 11 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The electricity market has always aroused the interest of many people due to the importance of its product. One can find the beginnings of electricity price prediction in the early 90s. It should be remembered that electricity is a specific commodity. It cannot be stored economically and, therefore, necessitates a continuous balancing of production and consumption; it depends on weather conditions, the season and the nature of human activity. These characteristics are unique and they lead to a price dynamic that is not found on any other market. It is often possible to recognize a day, weekly, or yearly seasonality and possible sudden, short-term, and generally unforeseen price peaks. Moreover, the presence of possible cycles characterizing this product can be characterized by irregularities, which is then difficult to properly model.

Electricity Price Forecasting (EPF) is a field of energy prediction that is focused on forecasting spot and forward prices on the wholesale power markets. It has become a significant element for power companies in their decision-making and investment process. Therefore, as usually happens within financially related areas, a number of EPF-related questions have to consider specific time horizon constraints. Within standard literature, it is customary to identify three types of forecasting horizons, namely: short, medium, and long-term, which are then better specified accordingly to the particular financial problem. As a general rule, one can consider: Short-term: a period of time ranging from few minutes to few days in advance; medium-term: a period of time that ranges from a few days to a few months; and, long-term: a period of time that ranges from some months to years. In most cases, the forecast horizons are of the first two types, with particular emphasis on day ahead previsions. Nevertheless, it is worth underlying that each of the above-mentioned horizons has its own importance, indeed a company is typically obliged to program its activities at different time scales. According to the particular time scale of interest, different modeling approaches can be exploited. In particular, most of the used techniques belong to the following five groups: statistical models, intelligent computing models, reduced form, fundamental, and multi-agent models. In this paper, we will start focusing on the first two

families, and then we combine them as to end up with a hybrid proposal representing the major novelty of the present paper.

As to proceed further, let us first classify the possible predictions type. For point forecasts, the goal is to predict the energy price at a given hour or day, usually within the short time horizon framework. To this end, a high accuracy level is mandatory. During recent years, a previous task lost in relevance in favour of so-called probabilistic predictions based on intervals forecasting w.r.t. the determination of 38 related quantiles or even aiming at predicting the characterizing entire probability distribution, see, e.g., [1] and the references therein. Next, we will look, in more detail, at all of the above aspects of EPF by applying specific models that were fed by average daily prices of the German electricity market, from 2016 to 2018. For more recent references regarding statistical methods for electricity forecasting, we refer the interested reader to e.g., [2–7].

This paper is structured, as follows: Section 2 briefly describes the possible types of forecasts and the main characteristics of the German electricity market. Section 3 explains the special case of a probabilistic forecast. Section 4 presents, in detail, the models that we will use for data analysis, especially regarding the SARIMA model (Seasonality AutoRegressive Integrate Moving Average) and the methods of artificial neural networks (ANN). The last section deals with the implementation of the aforementioned models on the basis of the German-data set already mentioned. We conclude with a brief recap that also provides critical remarks.

2. General Overview

Price forecasting constitutes a milestone in decisions making as to optimize market industries' strategies. In electricity price forecasting EPF related literature, we can distinguish three main types of predictions, according to requested previsions' time lengths. First, we can speak of short-term horizons, ranging from minutes to days, namely those for which reliable forecasts for meteorological variables, e.g., wind speed, temperature, cloud cover, etc., are available with sufficient accuracy grade. Short-term forecasts are mainly appropriate for market operations, intraday trading, and system stability, see [8].

Medium-term forecasts cover horizons beyond reliable weather predictions, without considering the large impact of technological and political uncertainty, with lead times being measured in weeks, months, or longer periods. Such a type of forecasts is mainly exploited for maintenance planning, reallocation of resources, bilateral contracts, valuation of derivatives, risk management, and budgeting. In the end, we have long-term horizons previsions, referring to everything from a few years to several decades. This type of forecasting is needed, for example, in order to address problems relating to long term future investment, global planning for large companies or cities, particularly when the behaviour of socio-economical conditions have to be taken into account, see, e.g., [9].

Price forecasting is essential for the energy market, with the latter typically being a day-ahead market that does not permit continuous trading. Agents have to place their offers and bids for electricity deliveries during each hour of the following day before a specific market close time. Subsequently, prices are determined simultaneously during a unit price auction for all load periods of the next days [8]. In electricity frameworks with zonal price formation, as in Europe, there are day-ahead markets and intraday markets. They begin to operate after the announcement of the results of the day-ahead auction and run until a few minutes before delivery, trying to compensate for differences that result from unexpected variations in demand or supply and positions in day-ahead contracts.

2.1. Type of Forecasts

Time series forecasting is the prediction of system changes in the future and, in general, it is based on information regarding the past and current state of the phenomena we observe. There are different types of forecasting approaches, and the right choice depends on the goal that is to be achieved.

2.1.1. Point Forecast

The series for the day-ahead price is usually obtained from an auction held once a day, where each hourly price for the next day is announced at once. Load periods can last less than one hour in the intraday markets, e.g., at the European Power Exchange (EPEX), and half-hourly or quarter-hourly products are also traded. The day can be partitioned in both cases into a finite number of load periods $h = 1, \dots, H$. Therefore, it is common to indicate, with $P_{d,h}$, the price referring to day d , with load period h . A point forecast of $P_{d,h}$, expressed by $\hat{P}_{d,h}$, is usually understood in the EPF literature as the expected value of random variable indicating the price, i.e., $\mathbb{E}(P_{d,h})$. One could extend this term to quantile forecasts that are a reasonable starting point for probabilistic predictions.

2.1.2. Probabilistic Forecast

There exist two fundamental approaches for probabilistic predictions. The most popular one is based on point prediction and the associated error distribution. A different choice is to consider the distribution of the spot price as, e.g., in Quantile Regression Average (QRA), see Nowotarski and Weron [10]. The focus can be on selected quantiles, prediction intervals, or the overall predictive distribution. Let us consider the average price at a future point in time, i.e., $\hat{P}_{d,h} = \mathbb{E}(P_{d,h})$, as a “point prediction”. We can write $P_{d,h} = \hat{P}_{d,h} + \varepsilon_{d,h}$, which means:

$$F_P(x) = F_\varepsilon(x - \hat{P}_{d,h}), \quad (1)$$

where, with F_ε , we refer to the distribution of errors that are associated with $\hat{P}_{d,h}$. Hence, the distribution of errors has the same form as the distribution of prices. Indeed, it is only shifted to the left by $\hat{P}_{d,h}$ on the horizontal axis. The corresponding quantile function $\hat{q}_{\alpha,\varepsilon}$ is likewise shifted concerning $\hat{q}_{\alpha,P}$. Equivalently, in the sense of the inverse empirical cumulative distribution function (CDF), we have: $\hat{F}_P^{-1}(\alpha) = \hat{P}_{d,h} + \hat{F}_\varepsilon^{-1}(\alpha)$, therefore obtaining the basic framework for the generation of probabilistic predictions from distributions of prediction errors. Let us underline that probabilistic predictions can be useful, e.g., when one needs information regarding huge future investments.

2.1.3. Ensemble Forecast

The probabilistic prediction concept is very general. Hence, it can happen that it is not enough to solve many energy prediction problems and it might be difficult to verify its validity. Indeed, $\hat{P}_{d,h}$ is considered alone, independent of the predictions for the adjacent hours. Nevertheless, rather than considering the \mathcal{H} univariate price distributions, we can focus on the \mathcal{H} -dimensional distribution F_P of the \mathcal{H} -dimensional price vector $P_d = (P_{d,1}, \dots, P_{d,\mathcal{H}})$. Therefore, we need a prediction for the multivariate distribution, but many models cannot supply such a direct distribution prediction. The latter problem can be overcome while computing an ensemble forecast. We can define an ensemble as a set of M paths $\mathcal{E}_M(\hat{P}_{d,h}) = (\hat{P}_d^1, \dots, \hat{P}_d^M)$ simulated from a forecast model, which is typically obtained by Monte Carlo methods.

2.2. Modelling Approaches

Usually the techniques for electricity price forecasting (EPF) are divided into five groups of models: statistically, computationally intelligent, reduced form, fundamental and multi-agent. We will focus on statistical and computational intelligent (CI) models.

2.2.1. Statistical Approaches

Statistical approaches are used in order to predict the actual price through a weighted combination of past prices and any current values of exogenous variables that could be associated with electricity, such as weather forecasts or demand, usually via a linear

regression. Autoregressive terms take into account the dependencies between prices of today and those of the days before, e.g., a possible structure could be:

$$\begin{aligned}
 P_{d,h} = & \beta_{h,0} + \beta_{h,1}P_{d-1,h} + \beta_{h,2}P_{d-2,h} + \beta_{h,3}P_{d-7,h} \\
 & + \beta_{h,4}P_{d-1,min} + \beta_{h,5}L_{d,h} \\
 & + \beta_{h,6}D_{sat} + \beta_{h,7}D_{sun} + \beta_{h,8}D_{mon} + \epsilon_{d,h},
 \end{aligned} \tag{2}$$

where $P_{d,h}$ denotes the price for day d and hour h , $P_{d-1,min}$ represents the minimum of the 24-h prices of the previous day (example of a non-linear component), $L_{d,h}$ is the load prediction for day d and hour h (known on day $d-1$), and $(D_{sat}, D_{sun}, D_{mon})$ are three dummies for modelling the weekly seasonality. Some authors call expert models previous, parsimonious structures, because they exploit a certain amount of experts' prior knowledge.

The standard approach for estimating the model (2) is Ordinary Least Squares (OLS), implemented by exploiting the electricity prices of the past \mathcal{D} days in order to predict prices for the next day(s). The optimal value \mathcal{D} is not given a priori, instead it should be chosen as to have a long enough estimation sample from which to extract samples, but not too long as to avoid to give an exaggerate of weight to events far in the past. Overall, there are no standard parameters. Many studies assume one or two years with hourly prices, but some use just 10–13 days as short time windows, while others are up to four years long, see, e.g., Marcjasz et al. [11].

Autoregression models represent the largest subset of statistical models, including

- similar-day methods, such as the naive method, which sets $\hat{P}_{d,h} = P_{d-7,h}$ for Monday, Saturday, or Sunday, and $\hat{P}_{d,h} = P_{d-1,h}$, otherwise,
- the generalized autoregressive conditional heteroskedasticity (GARCH) models, typically in connection with volatility forecasting, and
- shrinkage techniques, such as the least absolute shrinkage and selection operator (LASSO), but also Ridge regression and elastic nets.

Statistical models are attractive, because one can add physical interpretations to the regressors, being then useful to identify significant variables. Moreover, they allow for operators to better understand models' behaviors as to finally obtain models with a meaningful structure. A well-known disadvantage of statistical models is the representation of nonlinear factors, even if they can be approximated while using linear terms under certain circumstances. However, non-linear dependencies can be explicitly included by non-linear variables, like $P_{d-1,min}$ in (2). Otherwise, spot electricity prices, as well as exogenous variables, can be transformed by nonlinear functions before fitting a statistical model.

2.2.2. Computational Intelligence Methods (CI)

CI methods have been developed to solve problems that cannot be treated efficiently with statistical methods, such as, e.g., the presence of non-linear terms, or when a distribution-free approach is required. They combine different elements to create approaches that are able to analyze complex dynamic systems. In particular, they do not need exact knowledge, but they can also be confronted with incompleteness. CI models are flexible, and they are used for short-term predictions, being able to deal with different types of non-linearities, but at the cost of high computational complexity. It is worth mentioning that Statistics is more concerned with the analysis of finite samples, misspecification of models, and computational considerations, while probabilistic modeling is inherent in computational intelligence. Regarding Artificial Neural Networks (ANN) alternatives, different options are effectively used. In general, one can mainly focus on three aspects to identify a specific ANN algorithm for a prediction problem: data characteristics, solution complexity, and desired prediction accuracy, see [12].

In order to summarize the conceptual difference between these two approaches, we can observe Figure 1. Computational intelligence works more like a "black box", in which we can only decide a few parameters; each time that we train a network, we can get similar

but slightly different results; the statistical approach models a certain relationship between the data.

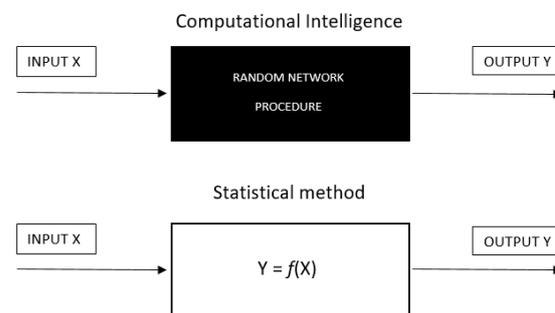


Figure 1. The computational intelligent (CI) method and statistical approaches.

2.3. The German Electricity Market

We will use price data from the German electricity market. The EPEX is the most relevant trading platform for electricity prices in Europe, also being the most important in terms of volume. It is also relevant from a policy point of view, since, for example, several regulatory calculations are based on the day-ahead EPEX price, such as the feed-in tariffs for renewable energies. It offers trading, clearing, and settlement in both the day-ahead and intraday market. Day-ahead hourly prices in Germany are traded on EPEX and they are referred to as “Phelix”, Physical electricity index. This index is the daily mean value of the system price for electricity traded on the spot market, which is calculated as the arithmetic mean of the 24-h market clearing price.

The day-ahead market is the primary market for electricity trading. Here, buyers and sellers set up hourly contracts for the delivery of electricity the following day. This is done through a daily auction, usually at 12:00, where the market clearing price, which is commonly known as the spot price, is determined by matching supply and demand. Most market designs choose a uniform-price auction market (see [13]): buyers with bids that are above the clearing price pay that price, and suppliers with bids below that price receive the same price.

The demand for electricity is a function of temperature, seasonality, and consumption patterns, from which the periodic character of electricity prices is derived. Demand is very price inelastic in the short term because consumers have few options available to them in response to price changes. Positive price peaks are often caused by high (unexpected) demand, cf. Figure 2. On the other hand, the merit-order curve plays a decisive role in the electricity price formation process. It is derived by arranging the suppliers’ offers according to rising marginal costs. The point of intersection of the demand curve with the merit order curve, also called supply curve, determines the market clearing price, i.e., the electricity spot market price.

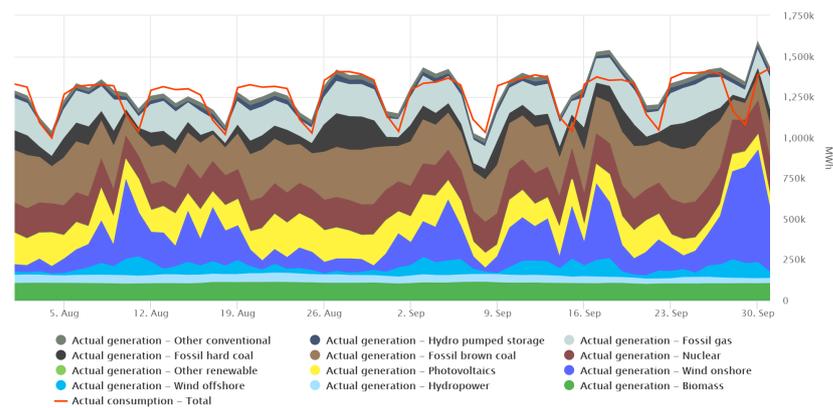


Figure 2. Electricity generation and consumption in August and September 2019. We can see the total electricity generation and consumption every day of the period, see [14] (CC BY 4.0 license).

In times of low demand, base load power plants, such as nuclear power plants and coal-fired power plants, generally serve as price-determining technologies. These plants are inflexible, due to their high costs. In contrast, in times of high demand, the prices are set by expensive peak load power plants, such as gas and oil-fired power plants. These plants have high flexibility, high marginal costs, and they lead to a convex shape of the merit order curve. With the lowest marginal costs, renewable energy sources are at the lower end of the merit order curve. Wind and solar energy have attracted the most attention in Germany, and regulatory changes are also the main driver for the growth of renewable energies, as several subsidies and political measures have been recently introduced. There is a large share of intermittent renewable energy sources, which makes the difficult task of forecasting and describing prices of changing energy markets. The hours of increased renewable energy supply are causing difficulties for inflexible plants that should be running continuously. This is because inflexible base-load plants have shutdown and start-up costs that force them to accept negative marginal returns in order to continuously generate electricity. This has a lowering effect on electricity prices. Negative prices are mainly caused by high wind production in times of low demand. Therefore negative price peaks mainly occur at night. For further details, we refer the reader to [15].

3. The Probabilistic Approach

The probabilistic approach is based on a prediction that takes the form of a probability distribution of future quantities or events. It allows for treating different types of problems, also extending the spectrum of the point estimation approach, e.g., long-term forecast becomes feasible.

3.1. The Problem

We can use a point forecast of the electricity spot price for the definition of the probabilistic forecasting problem, i.e., the “best estimate” or expected value of the spot price, see [16]. Notice that the actual price at t , P_t can be expressed as:

$$P_t = \hat{P}_t + \epsilon_t, \quad (3)$$

where \hat{P}_t refers to the point forecast of the spot price at time t , made at a previous time, and ϵ_t is the corresponding error. In most of the EPF papers, the analysis stops here, because the authors are only primarily interested in point forecasts, see [8].

The construction of prediction intervals (PI) is a very natural extension from point to probabilistic forecasts. Several methods can be used for this purpose and a popular one considered both the point prediction and the corresponding error: at the $(1 - \alpha)$ confidence level, the center of the PI is set equal to \hat{P}_t and its bounds are determined by the $\frac{\alpha}{2}$ th and

$1 - \frac{\alpha}{2}$ th quantile of the CDF of ϵ_t . As to provide an example, for the 90% PIs, the 5% and 95% quantiles of the error term are requested.

A forecaster can further expand such an approach by considering multiple PIs. As a finale result, we obtain a series of quantiles at many levels.

An appropriate discretization of the price distribution is also a set of 99 quantiles ($q = 1\%, 2\%, \dots, 99\%$). Generally, a density prediction that corresponds to (3) can be referred to as a set of PIs for all $\alpha \in (0, 1)$. The calculation of a probabilistic prediction requires an estimate of \hat{P}_t and the distribution of ϵ_t . Analogously, we reformulate it inverting the CDF of P_t and ϵ_t , namely:

$$F_{P_t}^{-1}(q) = \hat{P}_t + F_{\epsilon_t}^{-1}(q). \quad (4)$$

The probability forecast can be then defined as a probability distribution w.r.t. future quantities. Such an identification can be then carried on while using random variables. For our purposes, the latter implies the distribution of \hat{F}_{P_t} itself, which is the way that is implied by the QRA method. It is worth mentioning that we are not considering the Probability Density Function (PDF) of ϵ_t ; moreover, since 24-h prediction distributions have to be created in a single step, their interconnected probability properties have to be taken into account all at once. Following [16], we can either try to model the correlation between boundary distributions or we can simulate the 24-h paths characterizing the day-ahead dynamics; hence, using a multivariate model.

3.2. Construction of Probabilistic Forecasts

There are several ways to construct a probabilistic interval. We report four of them, mainly following [16].

3.2.1. Historical Simulation

We can use a historical simulation approach to compute the PIs. The latter is a path-dependent approach that consists of evaluating sample quantiles out of the empirical distribution for the ϵ_t errors' quantities.

3.2.2. Distribution-Based Probabilistic Predictions

When the models used for time series are driven by noises of the Gaussian type, which is the case, e.g., for the AR, ARMA, ARIMA, etc., methods, we directly use the Normal distribution to model the density forecasts, while computing the PIs accordingly via analytical tools. The latter differs from historical simulation because of the standard deviation of the error that we are considering in modelling the density, let us indicate it by $\hat{\sigma}$, which is first calculated, then one considers the lower, resp. upper, bound of the PIs to be set according to a $\mathcal{N}(0, \hat{\sigma}^2)$.

3.2.3. Bootstrapped PIs

An alternative method is the bootstrap one, which is often used within the NNs scenario. When considered to produce step-ahead forecast, the bootstrap approach works, as follows:

1. Provide an estimate for the parameters $\hat{\theta}$ characterizing the model and obtaining a fit with corresponding set of residuals ϵ_t .
2. Provide a set of (simulated) data, which are then not real world based, with distribution that is steered by the parameters' set $\hat{\theta}$ and normalized residuals ϵ_t^* . The latter means that, for a general autoregressive model of order r , w.r.t. variables (that constitute the external sources of noise) $P_2^* = P_1, \dots, P_r^* = P_r$, we recursively define:

$$P_t^* = \hat{\beta}_1 P_{t-1}^* + \dots + \hat{\beta}_r P_{t-r}^* + \hat{f}(X_t) + \epsilon_t^*, \quad \text{for all } t \in \{r+1, \dots, T\}. \quad (5)$$

3. Provide a new estimation for the model to then compute the bootstrap one step-ahead forecast for the new time step $t = T + 1$.

4. Starting from the the bottom, repeat both step 2 and 3, N times to obtain the (bootstrapped) price predictions: $\{\hat{P}_{T+1}^i\}_{i=1}^N$.
 5. Calculate the previously (step 4) quantities in order to provide the requested PIs
- This type of construction is more accurate, because it takes both historical forecast errors and parameter uncertainty into account. However, it is computationally heavier.

3.2.4. Quantile Regression Averaging

The method that was proposed by Nowotarski and Weron in [10] is called QRA and it involves quantile regression for a group of point forecasts from individual forecasting models. It is not necessary to split the probabilistic forecast into point forecast and error term distribution, because it works directly with the electricity spot price distribution. The quantile regression problem can be expressed as

$$Q_{P_t}(q|X_t) = X_t \beta_q, \quad (6)$$

where $Q_{P_t}(q|X_t)$ represents the conditional q -th quantile of the electricity price distribution, X_t are the explanatory variables, also called regressors, and β_q is a vector of parameters for the q quantile. Subsequently, minimizing the loss function for a given q -th quantile, the parameters can be estimated. There is no limitation of the components of X_t . Until it contains predictions from individual models, it is considered to be QRA.

3.3. Validity

In the case of a probabilistic forecast, the most important problem is that we do not know the actual distribution of the underlying process. It is not possible to compare the predicted distribution and the true distribution of the spot price of electricity with just the values of prices that were observed in the past. Probabilistic forecasting can be evaluated in several ways and the chosen method also depends on final goal that we aim to reach. Of course, we can rely on tests and parameters to check the validity of the model and to have criteria for choosing the optimal model. An evaluation is usually based on reliability, sharpness, and resolution.

Statistical consistency between distributional forecasts and observations is called reliability (or also calibration or unbiasedness). For example, if a 90% PI covers 90% of observed prices, then this PI is considered to be reliable, well-calibrated, or unbiased. The sharpness means how closely the predicted distribution covers the true distribution, i.e., the concentration of predicted distributions. This is a property only of forecasts, in contrast to reliability, which is a joint property of predictions and observations. Finally, resolution represents how strongly the predicted density changes over time, in other words, the ability to provide probabilistic predictions (e.g., wind power) depending on the forecast conditions, e.g., wind direction.

To formally check whether there is an “unconditional coverage” (UC), i.e., whether $\mathbb{P}(I_{d,h} = 1) = (1 - \alpha)$ where $I_{d,h} = 1$ if $P_{d,h}$ is in the interval, we can use the approach of Kupiec (1995), which allows for knowing whether $I_{d,h}$, also known as an indicator for “hits and misses”, is determined as a i.i.d. Bernoulli series, with an average of $(1 - \alpha)$, i.e., violations are assumed to be independent. Because the Kupiec test is not based on the order of the PI violations, but only on the total number of violations, Christoffersen (1998) introduced the independence test and the conditional coverage test (CC).

It is generally more difficult to test the goodness-of-fit of a predictive distribution than it is to assess the reliability of a PI. A very common method is to use the probability integral transform (PIT)

$$PIT_{d,h} = \hat{F}_P(P_{d,h}). \quad (7)$$

If the distribution forecast matches the actual distribution of the spot price process, then $PIT_{d,h}$ is uniformly distributed and independent, which can be verified by a statistical test, see [17].

In contrast to reliability, which is a joint property of observations and predictions, sharpness is only a property of predictions. Sharpness is strongly related to the notion of correct scoring rules. Indeed, scoring rules simultaneously evaluate the reliability and sharpness [17]. The pinball loss (PL) and the continuous ranked probability score (CRPS) are the two most popular correct valuation rules for energy forecasting: the former for quantile predictions and the latter for distribution predictions. The pinball loss (PL) is a particular case of an asymmetric piece-wise linear loss function:

$$PL(\hat{Q}_{P_{d,h}}(\alpha), P_{d,h}, \alpha) = \begin{cases} (1 - \alpha) (\hat{Q}_{P_{d,h}}(\alpha) - P_{d,h}) & \text{for } P_{d,h} < \hat{Q}_{P_{d,h}}(\alpha) \\ \alpha (\hat{Q}_{P_{d,h}}(\alpha) - P_{d,h}) & \text{for } P_{d,h} > \hat{Q}_{P_{d,h}}(\alpha) \end{cases} \quad (8)$$

so PL depends on the quantile function and the actually observed price. The PL is a strictly correct score for the α -th quantile. In order to obtain an aggregated score, the PL can be averaged over different quantiles.

It is also necessary to have statistically significant conclusions regarding the out-performance of the forecasts of one model by those of another model. For this purpose, we use the Diebold Mariano (DM) test, which is an asymptotic z-test of the hypothesis that the mean value of the loss differential series:

$$\delta_{d,h} = S_1(\hat{F}_{P_t}, P_t) - S_2(\hat{F}_{P_t}, P_t) \quad (9)$$

is zero, where $S_i(*, *)$ is the score of the forecasts of the model ($i = 1, 2$). In the context of probabilistic or ensemble forecasts, any strictly correct scoring rule can be used, for example, the pinball loss. Given the loss difference series, we calculate the statistics:

$$DM = \sqrt{T} \frac{\hat{\mu}(\delta_{d,h})}{\hat{\sigma}(\delta_{d,h})}, \quad (10)$$

where $\hat{\mu}(\delta_{d,h})$ and $\hat{\sigma}(\delta_{d,h})$ are the sample mean or standard deviation of $\delta_{d,h}$ and T represents the length of the test period outside the sample. The fundamental hypothesis of equal predictive accuracy, i.e., equal expected loss, corresponds to $\mathbb{E}(\delta_{d,h}) = 0$, in which case, while assuming a steady-state covariance of $\delta_{d,h}$, the DM statistic is asymptotically standard normal and one- or two-sided asymptotic tail probabilities are easily computed. The DM test makes a comparison between the forecasts of two models and not the models themselves. In day-ahead power markets, the forecasts for all hours of the next day are simultaneously made with the same set of information, implying that forecast errors for a given day usually have a high serial correlation. Therefore, it is recommended to run the DM tests separately for every load period considered, for example, at each hour of the day. [18]. It is also worth mentioning that probabilistic forecast performance can be improved, exploiting approaches that are related to regime-switching models, see, e.g., [19,20].

4. Models

We now present the theory behind some possible models for forecasting the energy price. We will see two statistical models and then one from the group of computational intelligence.

4.1. (S)ARIMA Models

Auto Regressive Moving Average (ARMA) models represent an important class of statistical models for analyzing and forecasting time series data. The autoregressive component uses the dependencies between observations and a given number of delayed observations; the moving average component uses the dependency between an observation and the corresponding residual error from a moving average model that is applied to delayed observations. This type of model relates the signal to its own past and does

not explicitly use information from other possible related time series. An ARMA(p, q) is defined as

$$\phi(B) X_t = \theta(B) \epsilon_t, \quad (11)$$

where B denotes the backward shift operator, i.e., $B^h = x_{t-h}$, $\phi(B)$ is the notation for the polynomial of the autoregressive component

$$\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p,$$

$\theta(B)$ for the polynomial of the moving average component

$$\theta(B) = 1 + \theta_1 B + \dots + \theta_p B^p,$$

and ϵ_t denotes a white noise $WN(0, \sigma^2)$. Thus, p and q are the orders of the two polynomials. The observed time series is considered as a realization of a stochastic process, whose parameters are the coefficients of the polynomials of the operator B , which determine the properties of persistence (and also the variance of the white noise). This kind of model assumes that the time series is stationary, i.e., mean and covariance of the time series shall be time independent. Usually, the series has to be transformed into the stationary form by differentiation. A model that explicitly includes differentiation is a generalization of the ARMA model for non-stationary time series: the "Auto Regressive Integrated Moving Average" (ARIMA) models. The equation of a ARIMA(p, d, q) is given by

$$\phi(B) \nabla^d X_t = \theta(B) \epsilon_t, \quad (12)$$

where $\nabla x_t \equiv (1 - B)x_t$ is the lag-1 differencing operator, a particular case of the general lag- h differencing operator that is given by

$$\nabla^h x_t \equiv (1 - B)^h x_t. \quad (13)$$

If $d = 0$, ARIMA($p, 0, q$) \equiv ARMA(p, q), i.e., ARIMA processes reduce to ARMA processes when differenced finitely many times.

Seasonality fluctuations are indeed caused by changing climatic conditions that influence demand and supply. For this reason, we introduce a model that can handle this behavior: the Seasonal Autoregressive Integrated Moving Average (SARIMA) process. The notation for a SARIMA model with both seasonal and non-seasonal parameters is SARIMA(p, d, q) \times (P, D, Q) $_s$. Here, (p, d, q) indicates the order of the non-seasonal part, while (P, D, Q) $_s$ is the seasonal part. The parameter s represents the number of observations in the seasonal pattern, e.g., for monthly data, we would have $s = 12$, for quarterly observations $s = 4$, etc. The SARIMA model is defined by the following formula:

$$\phi(B) \Phi(B^s) \nabla^d \nabla_s^D X_t = \theta(B) \Theta(B^s) \epsilon_t. \quad (14)$$

Any SARIMA model can be converted to an ordinary ARMA model in the variable $\tilde{B} = \nabla^d \nabla_s^D B$. Consequently, the estimation of the parameters of ARIMA and SARIMA models is analogous to that for the ARMA processes.

Time Series Analysis in the Simple Case of an ARMA Model

In what follows, we recall basic steps of time series analysis via the ARMA model, we refer to [21] for more details. This approach is also known as Box and Jenkins method, by the name of its authors, namely George Box and Gwilym Jenkins. The main methodological assumption here is to consider data as drivers of the model and not vice versa, so that the time series can "speak for itself". The ARMA scheme can be summarized, as follows:

1. Preliminary analysis: it is necessary to know the main features of the series such, e.g., its stationarity. To this aim, so-called "unit roots tests" are often used as well as the Dickey–Fuller test, see [22].

2. Order selection or identification: It is important to select appropriate values for the orders p and q . One can start by analyzing the total Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF), and then make some initial guesses based on their characteristics, see [21]. From these, one can obtain more reliable results while using various “Information Criteria” (IC), such as Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC). ICs are indexes trying to find a balance between the goodness of fit and the number of parameters. A “penalty factor” is included in the ICs to discourage the fitting of models with too many parameters. Hence, the preferred model is the one that minimizes the specific IC considered.
3. Estimation of the coefficients: once p and q are known, then the coefficients of the polynomials can be estimated, e.g., by a least squares regression or with a maximum likelihood method. In most cases, this problem can be solved with numerical methods.
4. Diagnostic check: to check whether the residuals, follow a random process. If the fitted model is suitable, the rescaled residuals should have similar properties to a “white noise” $WN(0, 1)$ sequence. One can observe the sample “autocorrelation function” (ACF) of the residuals and perform tests for randomness, e.g., the Ljung–Box test, see, e.g., ([21](Chapter 1)).

If the results are not satisfactory, the above described procedure can be restarted with a different order selection. Once the model has successfully passed the verification phase, then it can be used for forecasting.

4.2. Seasonality and Its Decomposition

There are different ways to deal with seasonality. An option consists in trying to find out what kind of seasonality we have and see whether it is possible to include it in the model, e.g., by using the SARIMA model. Otherwise, one can remove the seasonality at all.

In order to deseasonalize our series, we will use the MATLAB function `deseasonalize` written by Rafał Weron, see [13], and whose mathematical basis are described in what follows. In particular, such a function returns the deseasonalized data, the Short-Term Seasonal Component (STSC) and the Long-Term Seasonal Component (LTSC) obtained from the original data series. It also creates the periodograms of the original and deseasonalized data. With more detail, spectral analysis involves identifying possible cyclical patterns of data, so as to break down a seasonal time series into a few underlying periodic (sine and cosine) functions of certain wavelengths. It is known that the variability of many physical phenomena can depend on frequency. Therefore, the information regarding frequency dependence could lead to a better knowledge of the true mechanisms. Spectral analysis and its basic tools, such as the periodogram, can be useful in this direction. Let us recall that, for an observation vector x_1, \dots, x_n , the associated periodogram (or the pattern analog of spectral density) is defined as:

$$I_n(\omega_k) = \frac{1}{n} \left| \sum_{t=1}^n x_t e^{-i(t-1)\omega_k} \right|^2, \quad (15)$$

where $\omega_k = 2\pi(k/n)$ represents the Fourier frequencies that are expressed in radians per unit time, $k = 1, \dots, [n/2]$, while $[x]$ means the largest integer that is less than or equal to x , see, e.g., [23] for further details.

This function implements a simple but quite efficient method for eliminating the short-term seasonal component. The concept is to divide data into a matrix with rows of length T and then to take the average or median of the data in every column. In order to provide an example, we can think about seven element rows for a week period, calculated in average daily data. The obtained row vector, again of length T , constitutes the estimate of the seasonal component and it is the part to subtract from the original series.

The Wavelet Decomposition for the Long-Term Seasonal Component

After removing the weekly or daily seasonality from the data, one is often faced with the annual cycle. Even if a sine function is often a natural and good first approximation of the annual cycle, there are markets where it is revealed to not be the right choice, as in the case of the German market. Indeed, the latter has no recognizable annual seasonality and one can notice that spot prices behave similarly across the year, with peaks occurring in both winter and summer, as can be seen in Figure 3. One possible option is to use a wavelet decomposition, as suggested by Weron in [13].

A wavelet family consists of pairs of “father” (φ) and “mother” (ψ) wavelet. The former is the “low-frequency” smooth components: those that require wavelets with the greatest support, while the latter intercepts the “higher-frequency” detail components. In other words, father wavelets are used for trends or cycle components, and parent wavelets are used for any deviation from the trend, see [13,24]. More specifically, a signal wavelet decomposition uses one father wavelet and a sequence of mother wavelets:

$$f(t) = S_J + D_J + \dots + D_1, \quad (16)$$

where $S_J = \sum_k s_{J,k} \varphi_{J,k}(t)$ and $D_j = \sum_k d_{j,k} \psi_{j,k}(t)$. The coefficients $s_{J,k}$, $d_{j,k}$, and $d_{j-1,k}, \dots, d_{1,k}$ represent the wavelet transform coefficients that quantify the contribution of the corresponding wavelet function to the approximating sum, while $\varphi_{j,k}$ and $\psi_{j,k}(t)$ are the approximating father and mother wavelet functions, respectively. As soon as the signal is decomposed with (16), the procedure can be inverted to obtain an approximation to the original signal. If you want to use an unrefined scale, $f(t)$ can be estimated by S^J . The signal can be estimated by $S_{J-1} = S_J + D_J$ for a higher degree of refinement. At each step, we obtain a better estimate of the original signal by adding a mother wavelet D_j to a lower scale $j = J - 1, J - 2, \dots$. The reconstruction process can always be interrupted, especially when we reach the desired accuracy. The resulting signal can be seen as a de-noised (or filtered or smoothed) time series. Figure 3 shows both the deseasonalization results and the periodogram.

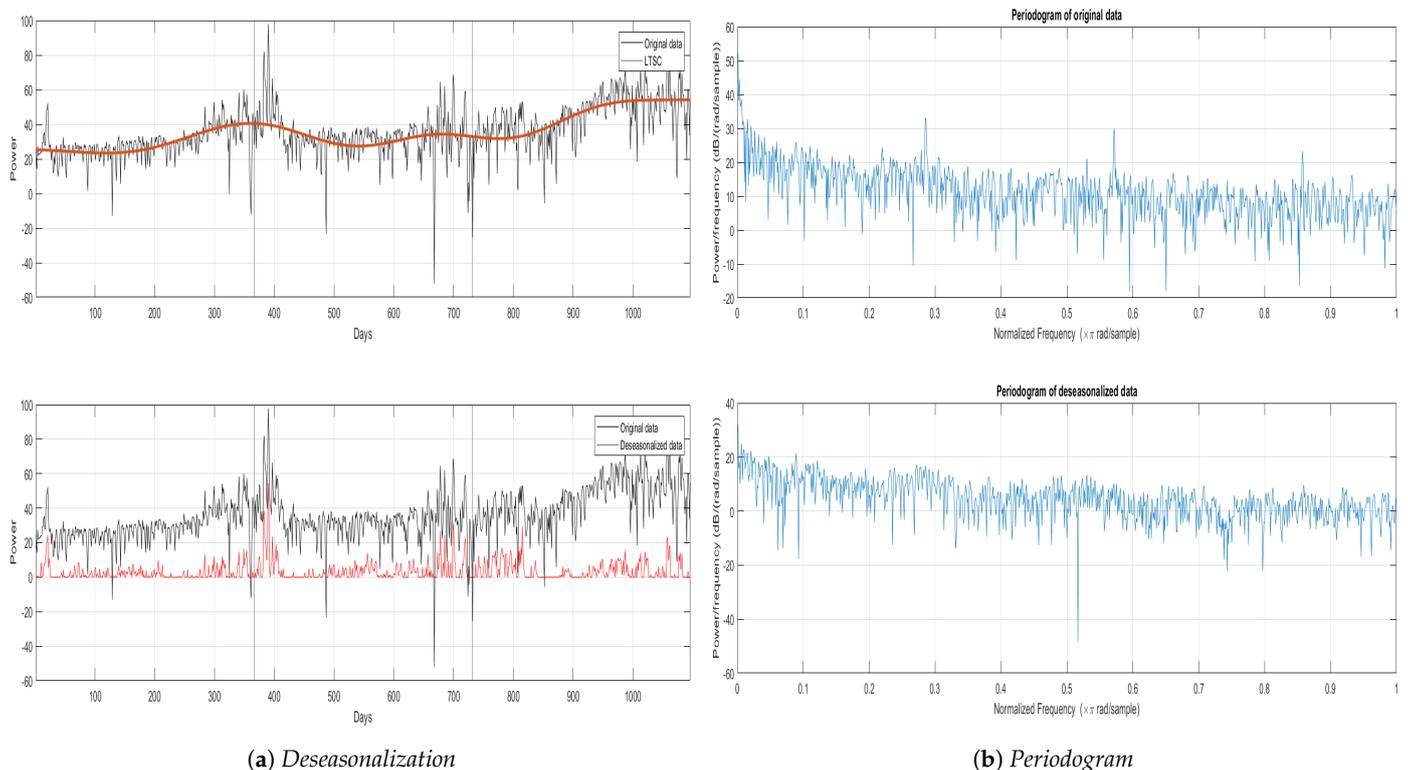


Figure 3. Example of results of “deseasonalize” on the three years.

4.3. Expert Models

The basic structure of an expert model is an autoregressive model, but other exogenous or input variables can be, and, in fact, are, also introduced. The latter is due to the fact that electricity prices are not only connected to their own past, but they are also influenced by the current and past values of different exogenous variables, such as energy loads and weather changes.

4.3.1. The ARX MODEL

Within this expert model, the centred log-price on day d and hour h is determined by:

$$P_{d,h} = \beta_{h,1}P_{d-1,h} + \beta_{h,2}P_{d-2,h} + \beta_{h,3}P_{d-7,h} + \beta_{h,4}P_{d-1,\min} + \beta_{h,5}L_{d,h} + \beta_{h,6}D_{\text{sat}} + \beta_{h,7}D_{\text{sun}} + \beta_{h,8}D_{\text{mon}} + \epsilon_{d,h} \quad (17)$$

where we assume the $\epsilon_{d,h}$ to be independent and identically distributed normal random variables. We define this autoregressive benchmark with ARX in order to underline that, in (17), the (zonal) load forecast is used as an exogenous variable. In contrast to naive models, which did not require parameter estimation, we have to estimate the model characterizing parameters.

4.3.2. The mARX Model

It can be advantageous to use not only different parameter sets, but also other model structures for each day of the week, see [18]. For example, the multi-day ARX model or mARX approach, defined, as follows

$$P_{d,h} = \left(\sum_{i \in I} \beta_{h,1,i} D_i \right) P_{d-1,h} + \beta_{h,2}P_{d-2,h} + \beta_{h,3}P_{d-7,h} + \beta_{h,4}P_{d-1,\min} + \beta_{h,5}L_{d,h} + \beta_{h,6}D_{\text{sat}} + \beta_{h,7}D_{\text{sun}} + \beta_{h,8}D_{\text{mon}} + \beta_{h,11}D_{\text{mon}}P_{d-3,h} + \epsilon_{d,h} \quad (18)$$

where $I \equiv \{0, \text{Sat}, \text{Sun}, \text{Mon}\}$, $D_0 \equiv 1$ and the term $D_{\text{mon}}P_{d-3,h}$ explain the autoregressive effect of Friday's prices on Monday's prices for the same hour. Notice that this structure is similar to periodic autoregressive models to a certain extent (i.e., PAR, PARMA). We could estimate both autoregressive models, hence ARX and mARX, exploiting least squares (LS) methods.

4.4. Artificial Neural Networks (ANNs)

The basic idea behind ANNs methods is to build algorithms that gradually learn from input data. They are mainly inspired by the structure and functioning of the human brain, which justifies the name: Artificial Neural Networks, or simply NNs. A remarkable property of NNs, which is also the main point distinguishing them from statistical methods, relies on the fact that they are trained from data through a "learning process".

Indeed, a NN is based on a structure of interconnected nodes, called artificial neurons and organized in layers. As it happens in a human brain via synapses, such components can transmit signals to other neurons. To be more precise, there is an input layer of source nodes that projects onto an output layer of neurons, possible intermediate steps being possible according with the presence of one or more hidden layers whose function is to mediate between the external input and the network output Figure 4. The more hidden layers we consider, the more higher order statistics are accessible, see, e.g., [25] for details.

Many different types of NNs have been considered over the years, each of which with own properties and possible applications. A key distinction is made between NNs whose compounds form "cycles" and those whose compounds are "acyclic". NNs with cycles are called recursive, feedback, or recurrent neural networks, while NNs without cycles are called feedforward neural network (FNN). However, both of them have the three basic components in common: a set of synapses, each of which is characterized by a weight representing the relevance of the connection it realizes between neurons, an adder allowing

to sum input signals and an activation function in order to both limit the amplitude of neurons' spikes and introduce possible nonlinearities into the neurons' output.

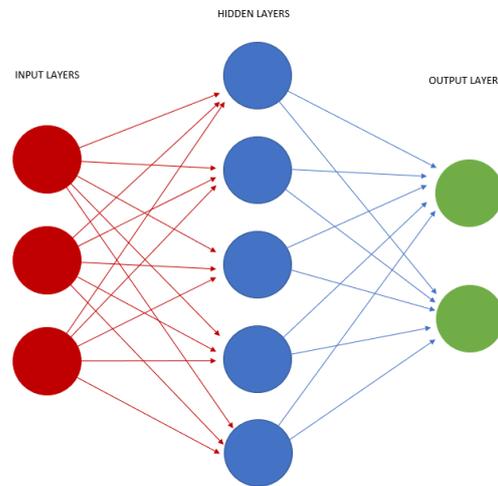


Figure 4. Example of a structure of a simple Neural Network (NN).

In mathematical terms, we can describe a neuron k by writing the following:

$$v_k = \sum_{j=1}^m w_{kj} x_j \quad \text{and} \quad y_k = \varphi(v_k), \quad (19)$$

where x_j are the signals, w_j are the synaptic weights of a neuron k , φ is the activation function, which maps a node's inputs to its corresponding output, and y_k the output signal of the neuron.

There exist different activation functions, such as, e.g., the logistic sigmoid function, or the tanh function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \quad \text{or} \quad \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \quad (20)$$

Previous functions are linked by the following linear transformation:

$$\tanh(x) = 2\sigma(2x) - 1, \quad (21)$$

implying that any function that is computed by a neural network with a hidden layer of tanh units can also be computed by another network with logistic sigmoid units and vice-versa.

Weights are calibrated while using an optimization algorithm minimizing a given loss function. The standard choice in this context is to consider the Stochastic Gradient Descent (SGD) optimizer, along with the Root Mean Squared Error (RMSE) as a loss function. The basic idea of gradient descent consists in finding the derivative of the loss function w.r.t. each of the network weights, then adjusting them along the negative slope direction. The "back-propagation" technique provides an efficient method for computing such a gradient, see, e.g., [26,27].

Once all of the data points have passed through the network, we say that an epoch is complete, i.e., an epoch refers to a single pass of the entire data set through the network during training. This procedure is repeated several times: this is what "training the network" means. At the end of each epoch, the loss of a single output is evaluated and it is possible to calculate the gradient of this loss in relation to the selected weight. The path to this minimized loss occurs in several stages, with its magnitude depending on the learning rate, which is typically between 0.01 and 0.0001. Therefore, the new weight is the old weight minus the learning rate times the gradient.

In summary, the workflow for the general NN design process comprises six primary steps: creating the network, configuring the network, initializing the structure, training the network, validating the network (post-training analysis), and using the network, see [25]. For our data analysis, we will use a special category of Recurrent Neural Network (RNN), called Long Short Term Memory (LSTM).

Recurrent and Long-Short Term Memory Networks

RNNs allow for cyclic connections, i.e., when time series are involved and we do not want to lose information regarding past relationships between data. In fact, an RNN with enough hidden units can approximate any measurable sequence-to-sequence mapping with any degree of accuracy, see, e.g., [28] for details. This allows for the network to remember earlier inputs. Indeed, the output at time t is not only connected to the input at time t , but additionally to recursive signals before that time. A problem arises when the gradient becomes too small, hence preventing the weight from changing its value, potentially inhibiting the neural network from further training, see [29]. For such a reason, RNNs are not suitable to provide time series predictions when, e.g., learning of dependencies over long distances or long-term storage of contexts are required. This problem can be solved by a variant of the RNN: the long-term short-term memory (LSTM) architecture [29].

We will now briefly review how LSTM-based architectures work, see, e.g., [29,30], Figure 5 for more details. In what follows, σ and \tanh from (20) are used as activation functions, with other choices being possible according to specific purposes.

- We define an input at time t as (X_t) and the hidden state from the previous time step as (S_{t-1}) , which is inserted into the LSTM block. Subsequently, we have to calculate the hidden state (S_t) .
- It is important to decide which information from the cell state should be discarded. This is decided by the following “forget gate”:

$$f_t = \sigma(X_t U^f + S_{t-1} W^f + b_f). \quad (22)$$

- Then you must determine which new information should be saved in the cell state. This part consists of two steps: firstly, the “Input-Gate” layer (i_t) determines which values are updated. Secondly, a \tanh layer creates a vector of new candidate values \tilde{C}_t . These two can be written as

$$i_t = \sigma(X_t U^i + S_{t-1} W^i + b_i), \quad \tilde{C}_t = \tanh(X_t U^c + S_{t-1} W^c + b_c). \quad (23)$$

- We can obtain the new cell state C_t updating the old cell state, C_{t-1} , as:

$$C_t = C_{t-1} \otimes f_t \oplus i_t \otimes \tilde{C}_t. \quad (24)$$

- Finally, the chosen output will be based on the cell state, but it will be a sort of filtered version. At this point, the output gate (o_t) determines which parts of the cell state should be computed as output. Subsequently, the cell state passes through the \tanh layer and it is multiplied by the output gate

$$o_t = \sigma(X_t U^o + S_{t-1} W^o + b_o), \quad S_t = o_t \otimes \tanh(C_t). \quad (25)$$

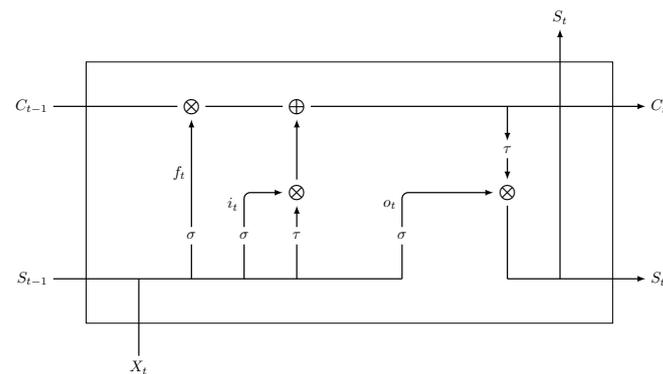


Figure 5. Example of a LSTM's memory block, where f_t , i_t , o_t mean forget, input, and output gates, respectively; τ and σ represent the tanh and sigma activation functions of [30].

5. Data Analysis

Our analysis is based on three years of hourly energy prices data for the German electricity market, i.e.: from 01.01.2016 to 31.12.2018. We will consider the daily average for each day, for a total of 1096 observations. We use historical observations, our endogenous variable, as input, aiming at forecasting the related future values as output. Subsequently, we are considering a regression type problem. Indeed, we want to create a multilevel forecast of a numerical quantity. Our time series is coherent, because the observations are collected and structured uniformly over time, e.g., we have a seasonal pattern.

First of all, it is important to have a general idea of the behavior of the data. As an example, we record the data for the year 2016 and, additionally, their monthly average as well as two weeks in May. We can see the pattern of weekly seasonality, see Figure 6b: we can clearly see the weak seasonality and the difference between Mon–Fri and the weekend. This example proves the dependence of the price on external factors, like business activities, etc. There are some outliers and, on average, the same days show a negative price. For more details on how peaks are handled in the energy market, see, e.g., [31].

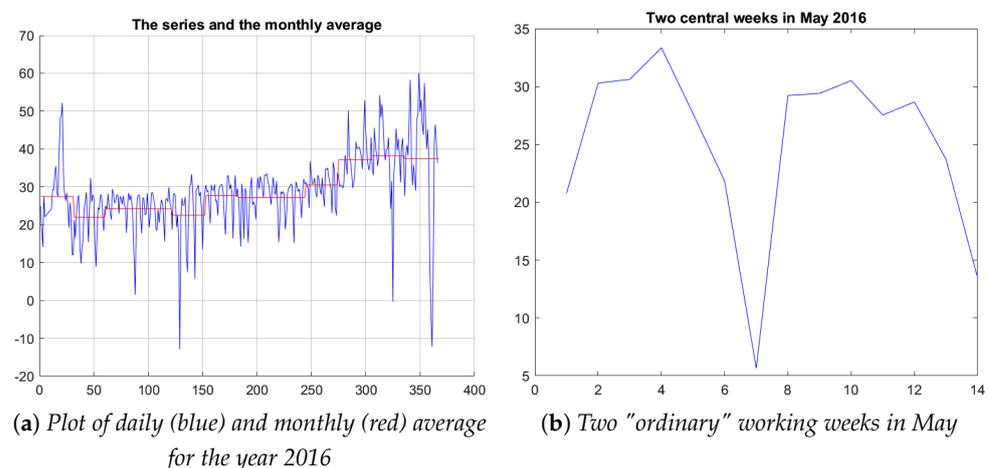


Figure 6. Data visualization.

5.1. From the Probability Density Function to a Possible Realization of the Stochastic Process

There are some interesting questions that we would like to answer, for example: whether it is possible to estimate a stochastic process that obeys a certain PDF; whether we can create a kind of forecast from it; and, whether it could be a suitable starting point for more structured forecasts. This approach could help a user to have a better knowledge of his future probable data for a mid-term period. We start by describing the general idea from a mathematical point of view, and we see how we can try to apply it. We introduce

the rejection method and one of its improvements, for more details, see, e.g., [32]. They generate random variables with the non-uniform distribution.

The Rejection Method

The rejection method creates a random variable with a given PDF f by using two independent generators $U(0,1)$. The idea of John von Neumann is to create a box $[a,b] \times [0,c]$ containing the graph of f . Subsequently, a random point (U,Y) in the box and U is accepted if the point is below the graph. This is a simple rejection algorithm that can be generalized for a d -dimensional PDF:

1. Generate $U \sim U(a,b)$ from $U = a + (b-a)U_1$ with $U_1 \sim U(0,1)$.
2. Generate $Y \sim U(0,c)$ from $Y = cU_2$ with $U_2 \sim U(0,1)$.
3. If $Y \leq f(U)$, then accept $X = U$, or else reject U and return to step 1.

Subsequently, we know from the Fundamental Theorem of Simulation that the random variable X that is generated by the general rejection algorithm has the PDF f .

5.1.1. Marsaglia's Ziggurat Method

The Marsaglia's Ziggurat method Figure 7 is a highly efficient rejection method [33], which can be implemented, e.g., in MATLAB while using the function `randn`. It is based on an underlying source of uniformly distributed random numbers, but it can also be applied to symmetric unimodal distributions, such as the normal one. An additional random sign \pm is generated for the value that is determined by the Ziggurat method applied to the distribution in $[0, \infty)$. Instead of covering the graph of a given PDF f with a box, the idea now is to use a "ziggurat of rectangles", a cap and a tail, which all have the same area. These sections are selected, so that it is easy to choose uniform points and determine whether to accept or reject them.

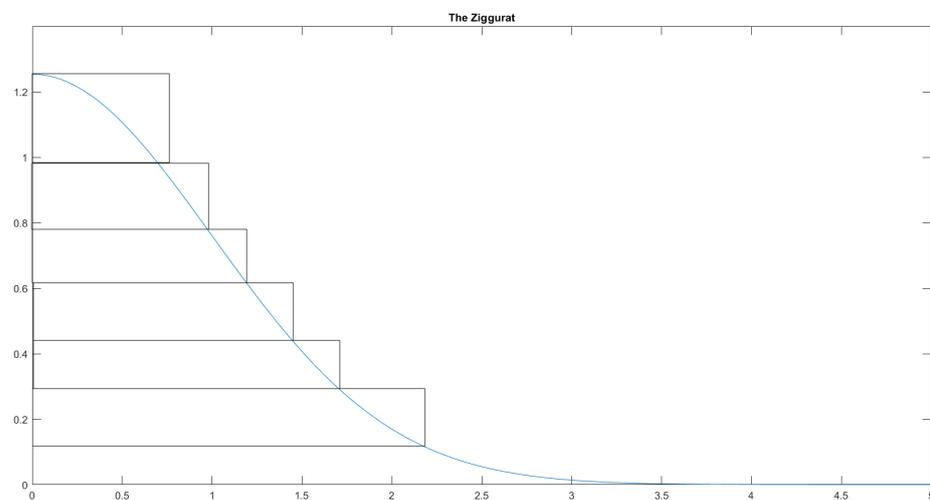


Figure 7. The Ziggurat basic idea.

For the sake of simplicity, we assume that f has support in $[0, \infty)$ and decreases monotonously. For a given number M , we ask for the nodes $0 = x_0 < \dots < x_M < \infty$, so that the ziggurat rectangles Z_i and the tail Z_0 have the same area. Subsequently, the Ziggurat method reads:

1. Choose $i \in 0, \dots, M$ at random, uniformly distributed. Generate $U \sim U(0,1)$.
2. If $i \geq 1$, then
 - Let $X = Ux_i$.
 - If $X < x_{i-1}$ then return X ,
else generate $Y \sim U(0,1)$ independent of U .
If $f(x_i) + Y(f(x_{i-1}) - f(x_i)) < f(X)$, then accept X , otherwise reject.

3. If $i = 0$ then
 - Set $X = (vU) / f(x_M)$
 - If $X < x_M$, accept X ,
or else generate a random value $X \in [x_M, \infty)$ from the tail.

It is important to realize that the resulting distribution is exact, even if the zigurat step function is only an approximation of the probability density function.

5.1.2. The Function Randn and Its Possible Application

Let us consider the summer season (01/06–31/08) of the average daily price for the two years 2016 and 2017 in order to obtain similar data. A natural question arises: whether it is possible to create a probable realization for the summer of 2018 (or only for one period), while taking the probability density functions of the previous data into account. Before continuing, we first standardize the data so that they have a normal distribution in the mean. In the following figure, we can observe the behavior of our data for this particular period. In Figure 8a, we can observe that 2016 and 2017 are very similar, while the 2018 data are much higher on average.

Afterwards, we use the MATLAB function `randn`, implemented with the Ziggurat's algorithm with the options: $X = \text{randn}(_ 'like', p)$ returns an array of random numbers like p . X will be our possible realization for summer 2018. For example, if we generate 1000 paths and evaluate the *RMSE*, we get, in Figure 9, the following histogram, where we can check that the values are mainly in the range [1.35,1.46]. As an example, we could get with $RMSE = 1.33$:

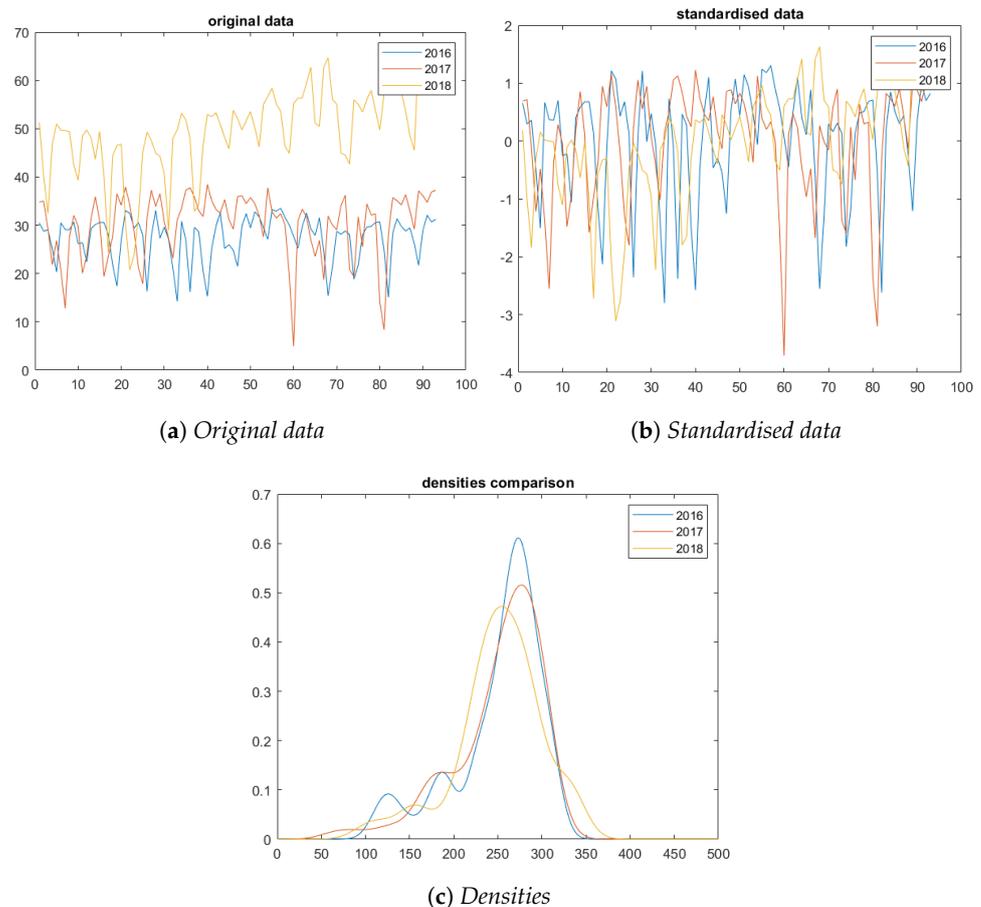


Figure 8. Behaviour of data and their densities.

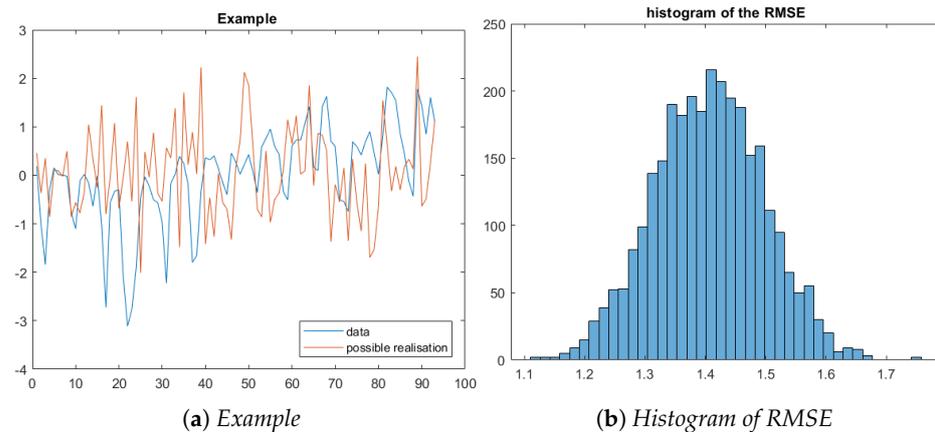


Figure 9. Example of a possible realisation with standardised data.

It is important to see what could happen if we perform the inverse standardization procedure; Figures 9 and 10 show the results. We conclude that such a big difference could be due to the fact that summer 2018 was quite different from the last two summer times. Additionally, the data densities are not completely unimodal and symmetric, so the algorithm may not work properly. Therefore, one can say that this first approach gives better results if the situation is regular over time and always follows the same “trend”, or if one has access to additional information and variables that can improve performance. Indeed, applying the same procedure, but only considering 2016 to provide a forecast for 2017, we obtain Figure 11. We can also note that any possible realization never reaches an “outlier” or a particularly high spike.

Finally, our opinion is that this approach cannot be very useful from a practical point of view. In fact, even if it can be used to get a general idea of the medium and long term, it does not provide an adequate level of accuracy for the required forecasts.

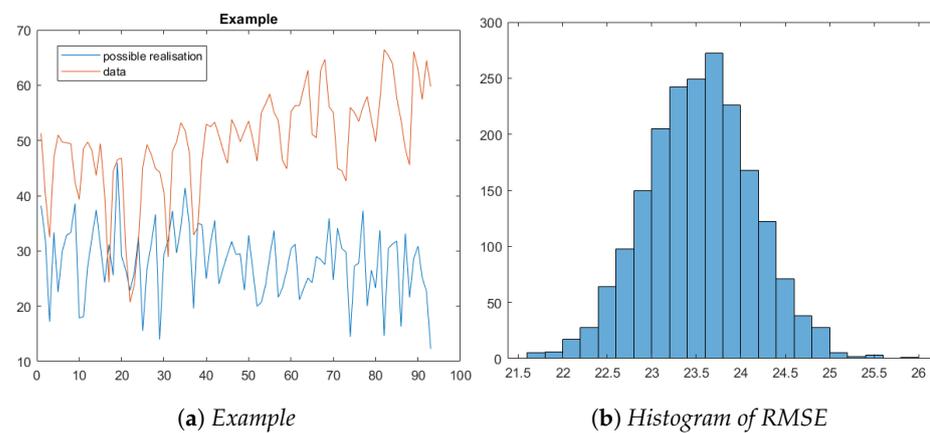


Figure 10. Use of summer of 2016 and 2017 for possible data of summer in 2018.

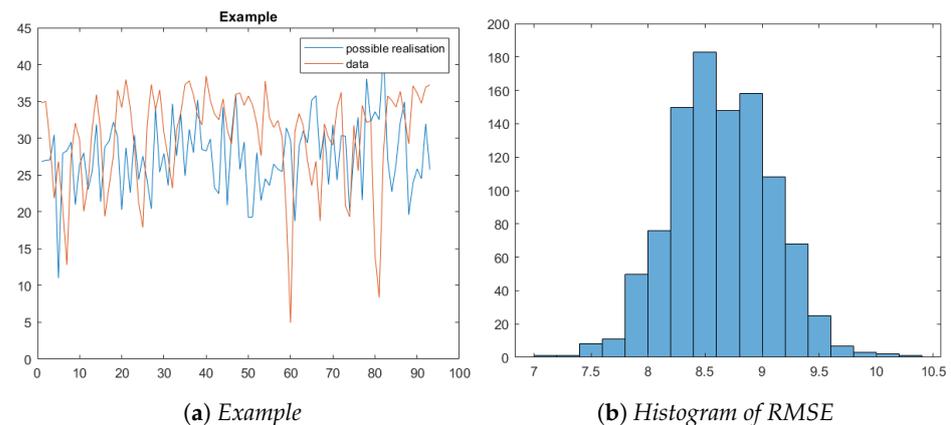


Figure 11. Use of only the summer of 2016 for possible data of summer in 2017.

5.2. Example with SARIMA Model

In this section, we present the results of a forecast that was obtained by implementing a SARIMA model. We use the Matlab model `sarima` with the parameters while using the estimator and the function `forecast` to predict the average prices for the next few days in order to calculate the forecast. We try to predict two weeks of the year 2018, and we will use three different training periods: one year (2017), the spring/summer season from April to September 2017, and the autumn/winter period from October 2016 to March 2017. Table 1 shows the parameters that are used for each analyzed period. With $s = 7$, we refer to the fact that we implement a weekly seasonality, as our kind of data suggests.

Table 1. SARIMA model's parameters.

Training Period	SARIMA(p, d, q) \times (P, D, Q) _s
One year	(2, 0, 2) \times (1, 0, 1) ₇
Spring/Summer	(2, 1, 2) \times (1, 1, 1) ₇
Autumn/Winter	(3, 1, 3) \times (1, 1, 1) ₇

In addition, a test can also be performed to check the fit of the model, e.g., the well-known “Ljung-Box Q-Test”, for more details, see [34]. We test it for 20 lags with a significance level of 0.05. For all of the tested models, we do not have to discard the null hypothesis, so that we can consider them to be good and reasonable models for our data. The correlations in the population from which the sample is drawn are 0, so that all of the correlations observed in the data result from the randomness of the sampling procedure.

Figure 12 shows the results of the forecasts and their confidence intervals at 95%. In Table 2, we see the values of the RMSE: a shorter and more specific time period for the train has a better outcome.

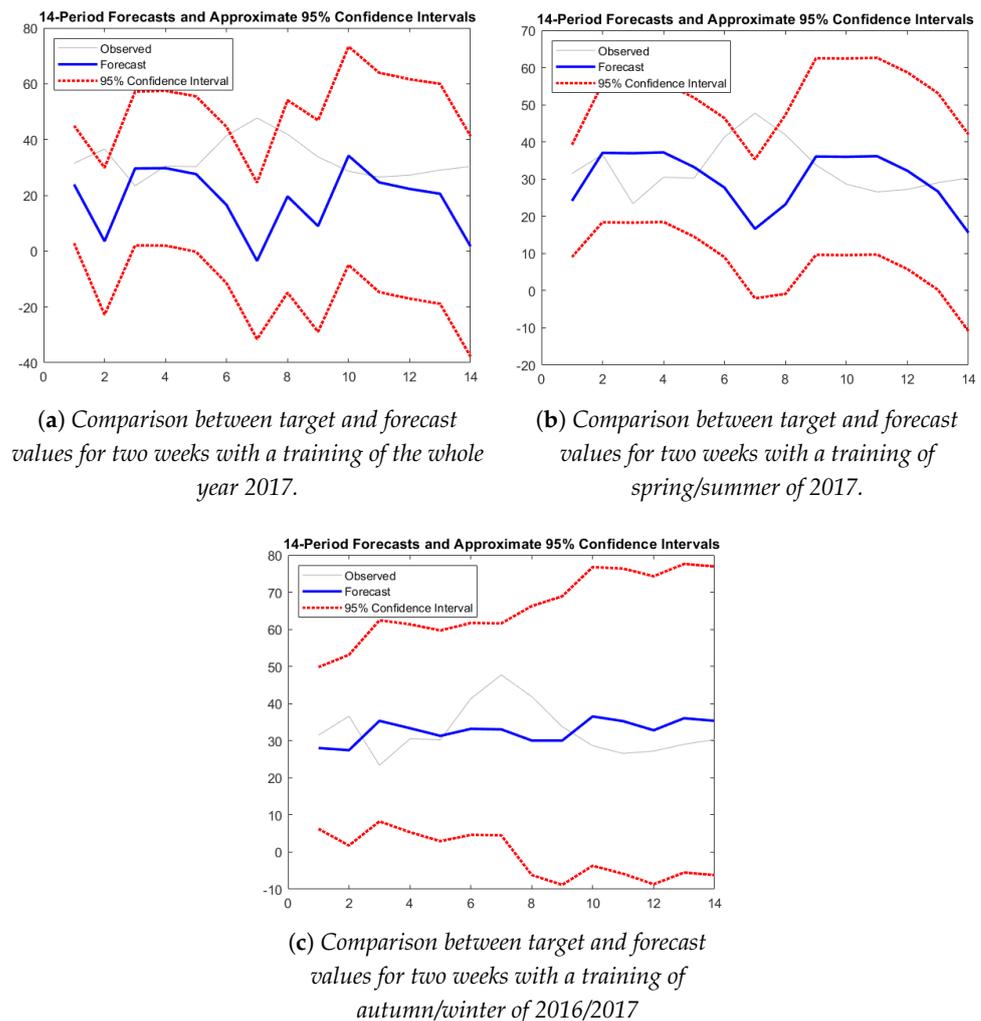


Figure 12. Comparison between target and forecast values for two weeks with the given training data.

Table 2. Summary of the Root Mean Squared Error (RMSE).

Train Period	One Year	Spring/Summer	Autumn/Winter
RMSE	21.5418	12.5163	8.1396

The results may not be very accurate, but let us note that we are considering a daily average of the price. Therefore, it is more important that the general trend shows similar behavior. In addition, we can observe that fluctuations occur for certain periods of time, but these latter are not as regular as for other types of data, which implies a higher level of difficulties in analyzing these data.

5.3. ARIMA Model

Because our series are quite long, it may not be sufficient to model weekly seasonality, but it may be necessary to include other components. As in the SARIMA example, we will start by modeling 2017 so as to provide a forecast for the first two weeks of 2018. Subsequently, we will see the results referring at only one period of the year (spring/summer; fall/winter).

- We decide to deseasonalize the series while using the Matlab function `deseasonalize`, see Section 4.2. It considers both a short-term and a long-term seasonal component.
- We have used an Augmented Dickey–Fuller test to verify that our deseasonalized series is stationary. It tests the null hypothesis that a unit root is in a time series sample.

In Matlab, we can use the function `adftest`; it returns a logical value with the rejection decision for a unit root in a univariate time series. For example, the result $adf = 0$ indicates that this test does not reject the null hypothesis of a unit root against the trend-stationary alternative.

In our cases, we have that the series is not stationary, so we can differentiate it, then obtaining the series to become stationary, which is equivalent to having $d = 1$.

- Our goal now is to find a suitable ARIMA(p, d, q) model for estimating the series. To guess a plausible order of the parameters, we consider the autocorrelation and partial autocorrelation functions, as proposed in the procedure of Box and Jenkins.
- We try to estimate different types of models, then choosing the one minimizing the information criterion AIC and BIC. In particular, we end up with choosing ARIMA(1,1,2):

$$(1 - \phi_1 B)(1 - B) x_t = (1 + \theta_1 B + \theta_2 B^2) \epsilon_t. \quad (26)$$

- Subsequently, we calibrate our parameters in order to optimize the error in the L^2 norm of the difference between the Probability Density Function (PDF) of the data and the forecast we calculated based on the estimated model. The PDF is estimated by the Matlab function `ksdensity`. It uses a non-parametric method, called kernel density estimation. We are interested in estimating the form of the density function f from a sample of data (x_1, \dots, x_n) ; its kernel density estimator is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^N K\left(\frac{x - x_i}{h}\right), \quad (27)$$

where K is the kernel, $h > 0$ denotes a smoothing parameter, called bandwidth, and N denotes the number of observations. By default, K is set as the normal density function.

- The optimal prediction results from the solution of this problem:

$$\min_{\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q} \|PDF(\text{forecast})_{\{\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q\}} - PDF(\text{data})\|_2, \quad (28)$$

where $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q$ are the parameters of the chosen model, and we choose a compact interval of \mathbb{R} , where they can vary before solving the problem. With $PDF(\text{forecast})$, we denote the estimated density function from the obtained forecasts. In order to solve this minimization problem we use the Matlab function `fminsearch`, which determines the minimum of a multi-variable function using a derivative-free method. In this case, we use the Matlab function `arma_forecast` for the point forecasts, see [35]. In particular, `fminsearch` uses the Nelder–Mead simplex algorithm, which is a direct search method that does not use numerical or analytical gradients. Indeed, it depends on the given initial values. In particular, we use the parameters that come from the first estimate of the ARIMA model.

Let us note that such an approach differs from the one that was used by Ziel and Steinert, see [9]. In fact, the optimal parameters of the problem are given by the solution of minimizing the BIC criteria while using the lasso technique. The general idea is that, for a linear model $Y = \beta'X + \epsilon$, the LASSO estimator is given by:

$$\hat{\beta} = \min_{\beta} \|Y - \beta'X\|_2^2 + \lambda \|\beta\|_1 \quad (29)$$

where $\lambda \geq 0$ is the penalty term.

- We plot the density functions and see the results in Figure 13.

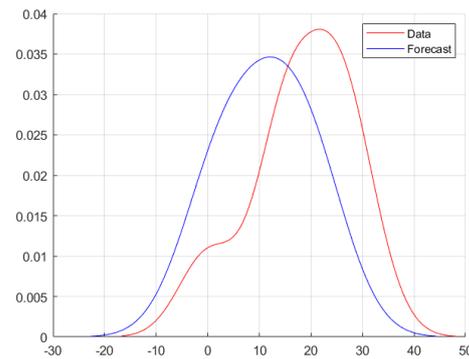


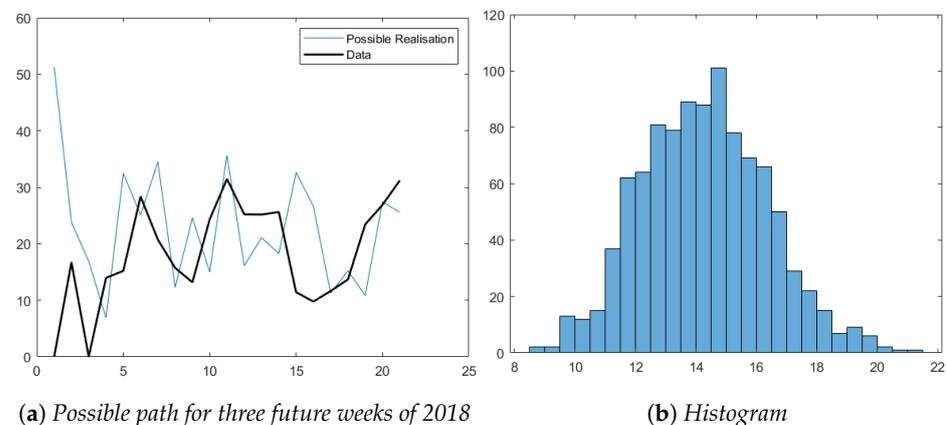
Figure 13. Estimated density functions of forecasted and original data.

- In Table 3, one can compare between the error of the difference before and after optimization:

Table 3. Errors before and after the optimization

Forecasted Days	Error before	Error after
14	$1.83 \cdot 10^3$	0.1098

- Finally, we try to implement the idea that is described in Section 5.1. In view of the Probability Density Function (PDF) of the point estimates, we can guess a possible realization of the future period, for example, for the first three weeks of January 2018. A plausible result can be observed in Figure 14.



(a) Possible path for three future weeks of 2018

(b) Histogram

Figure 14. From PDF to the stochastic process.

Now, we follow the same steps—test for stationary, identification of the model, calibration of the parameters—to see whether we can achieve an improvement of the RMSE by reducing the number of observations and choosing a certain time period for training and testing (see Table 4). Therefore, we first use the autumn/winter period and then the spring/summer period (See Figures 15 and 16).

Table 4. Summary for the two restricted period.

Period	ARIMA	Forecasted Days	Error before	Error after
Autumn/Winter	(2,0,0)	14 days in January	1.7390	0.1223
Spring/Summer	(1,1,4)	14 days in June	11.1979	0.1375

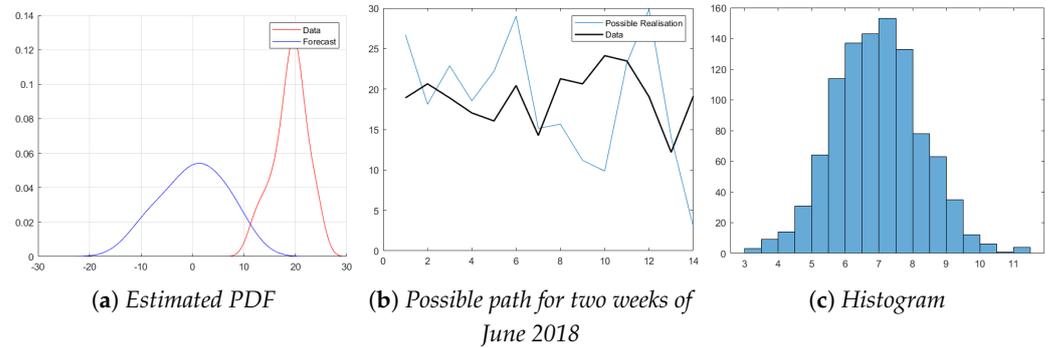


Figure 15. Summary of results for spring/summer period.

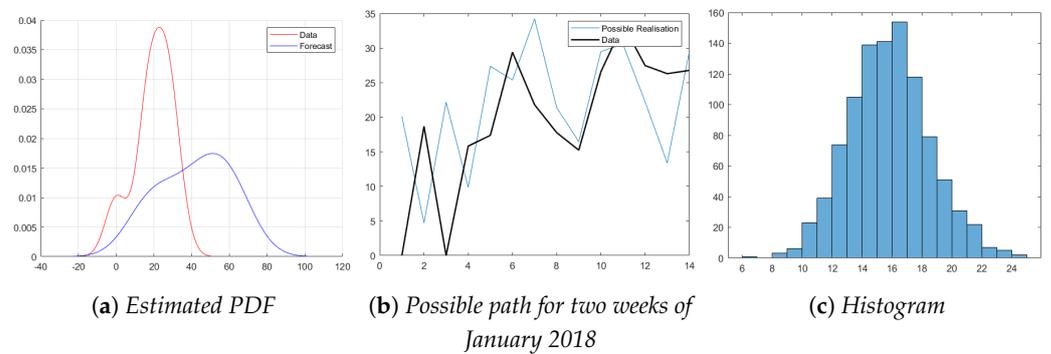


Figure 16. Summary of results for autumn/winter period.

Let us note that the values of *RMSE* are now generally not as high as in the case that is considered in Section 5.1. This is due to the fact that we use a more structured procedure in the preparation of our forecast. We are aware that this type of approach may not be appropriate if we want to achieve a high degree of accuracy; there are already a lot of other methods for this purpose. Here, we try to imagine a plausible trend for future data within a medium-term period. Such an approach in combination with ARIMA leads to better results when the behavior of the series is quite regular, as already mentioned and as it is the case when dealing with statistical models.

5.4. Neural Networks

For the same data, we try to use an ANN to calculate the forecast. In particular, we implement in MATLAB the LSTM architecture from Section 4.4. The selection of a suitable representation and preprocessing of the input data plays a central role in any machine learning task. However, neural networks have the tendency to be relatively robust to the choice of input representation, see [26]. The only requirements for input representations of NNs are that they are complete, i.e., that they contain all of the information needed to successfully predict the output. Although irrelevant inputs are not a major problem for NNs, a large input space could lead to too many input weights and poor generalization.

A common procedure is the standardization of the components of the input vectors. This means that, first, the mean value and standard deviation are calculated with the classical estimators:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i, \quad \text{and} \quad \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2}, \tag{30}$$

then compute the standardised input vector \hat{x} with $\hat{x} = \frac{x-\mu}{\sigma}$.

This method does not change the training set information, but it does improve the performance by moving the input values to a range that is more appropriate for the standard activation functions. Standardizing the input values can have a great impact on the performances [26].

Once we have standardized, we train the network exploiting specific Matlab functions. In particular, we use the function `trainNetwork`, specifying the train set, the `lstmLayers`, and appropriate options for the training. With the function `trainingOptions`, which is part of the Matlab Deep Learning Toolbox, we can set all of the parameters that we need, see Table 5. It is worth mention that there is not a specific procedure for selecting these values. Contrary to what we saw for the (S)ARIMA example, we do not have any criteria or index that we can exploit to justify a choice. Here, the decision is made after several attempts. There are two main problems to pay attention to. The first one is the so-called over-fitting. This implies that it is fundamental to prescribe an appropriate number of epochs for the training phase. For example, we can stop once we see that the loss function is stable, or we can try to optimize the parameters: a very high number of hidden layers could be complex to handle from a computational point of view.

In Table 5, we also see the duration of the training. The training time is another difference to the statistical methods, which are more immediate. Here, the elapsed time is about 1 min., but, for larger data sets, this value can become hours.

Table 5. Summary of Neural Network architecture in Matlab.

Number of Epochs	250
Hidden Layers	200
Initial Learn Rate	0.005
Elapsed Time	1 min 8 s

We know that the purpose of the SGD during training is to minimize the loss between the actual performance and performance expected by our training masters. The loss minimization is progressive. The training process begins with randomly set weights, and then these weights are updated incrementally as we get closer to the minimum. The steps size depends on the learning speed. During training, once the loss has been computed for our inputs, then the gradient of that loss is computed in relation to every weight in the model. We can observe this type of process in Figure 17a: We see that the loss function, the RMSE, is minimized epoch by epoch.

Having obtained gradients values, we multiply them for the learning rate, which results in small numbers, usually between 0.01 and 0.0001. Nevertheless, the actual value can change, and any value that we obtain for the gradient will be smaller after we multiply it by the learning rate. If we set the learning rate on the higher part of this range, then there is a risk of overshoot. This happens when we take a “too big” step toward the minimized loss function and exceed this minimum and miss it. We can avoid this by setting the learning rate to a value on the lower side of the range. Our steps will be very small with this option, so it will require more time to reach the point of loss minimization. Overall, the act of choosing between a higher and lower learning rate leaves us with this kind of compromise idea. All of these starting parameters are something to test, and then you can choose the more suitable one for the problem, it is impossible to determine them in advance. When considering the example with statistical methods, where we have information criteria that can lead us to optimal parameters, in this case we can only observe the final RMSE and the course of the training process. For example, if we see high peaks or irregularities until the end of the training, it might be necessary to add some epochs; conversely, if the loss function suddenly becomes very low and remains constant, then we can decide to reduce the number of epochs or decrease the learning rate to avoid over-fitting phenomena.

In the following figures, we can see the results: the training procedure, a comparison between forecasts and original data and the comparison between densities.

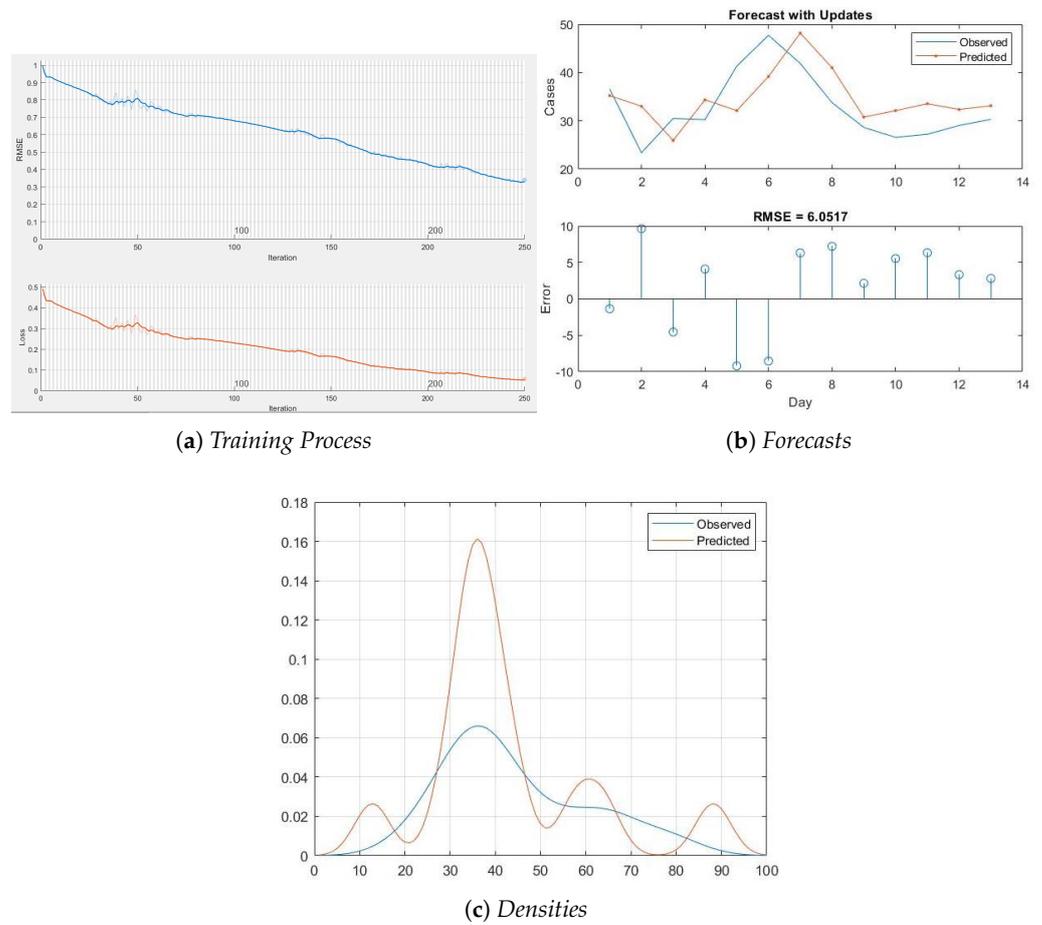


Figure 17. Summary of the results of long-term short-term memory (LSTM) network on one year data for two weeks forecast.

As done in the previous sections, we can repeat the same procedure while using the spring/summer period, Figure 18; and then using the autumn/winter period, Figure 19 and Table 6.

Table 6. Summary of Neural Network architecture in Matlab for the summer and winter period.

	Spring/Summer	Autumn/Winter
Epochs	280	250
Hidden Layers	190	190
Initial Learn Rate	0.009	0.007
Elapsed Time	56 s	43 s

We can observe that we obtain good results and that it is possible to simplify calculations shortening the period as well as to obtain more precision for a given season, as done for previous analysis.

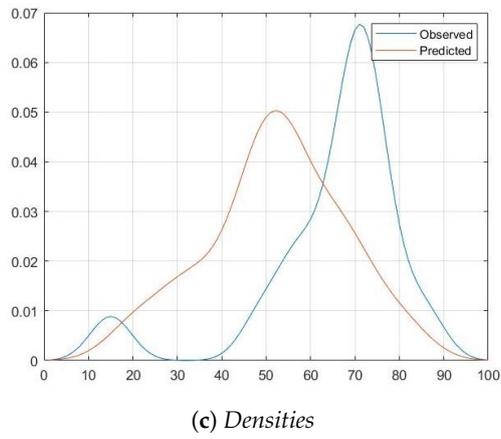
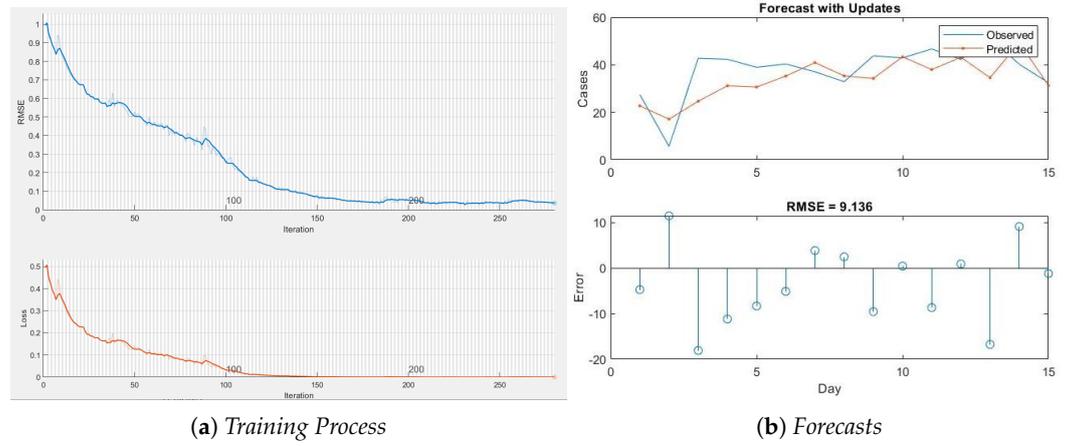


Figure 18. Summary of results of LSTM network on spring/summer period for two weeks forecast.

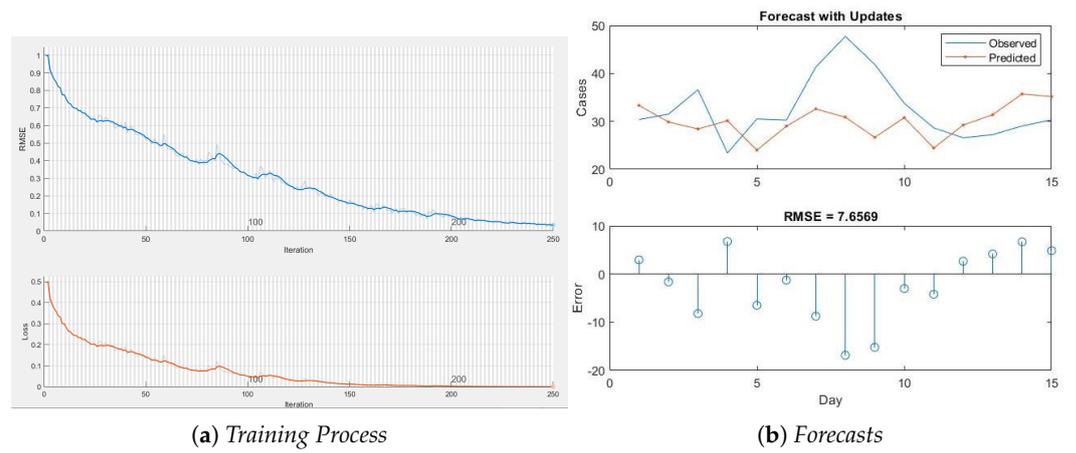
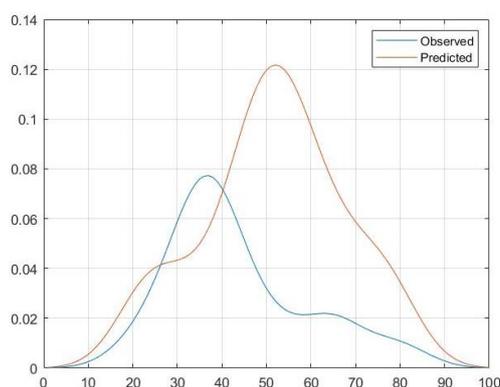


Figure 19. Cont.



(b) Densities

Figure 19. Summary of results of the LSTM network on autumn/winter period for two weeks forecast.

5.5. Hybrid Approach

In the literature, there are other important approaches used to forecast the price of electricity, namely: hybrid methods realized by combining other models to capture different patterns characterizing time series of interest. A common example of these hybrid models is a combination between a statistical model, such as the ARIMA and an LSTM-based model. For more details, we refer the interested reader to [36]. As to proceed further, the simplest approach is to average between them, finding suitable weights w_i :

$$forecast = w_1 SARIMA(results) + w_2 LSTM(results). \quad (31)$$

This procedure is quite common, allowing to easily decide which component needs to be highlighted. In Table 7, we specify the weights for each component, while, in Table 8, we can see the final comparison between RMSE, noticing that the hybrid approach can be a valid tool for obtaining a lower RMSE. In Figure 20, we summary the results of hybrid approach for each time period of training.

Table 7. Percentage of component for each training period.

	One Year	Autumn/Winter	Spring/Summer
SARIMA	10%	30%	50%
NN	90%	70%	50%

Table 8. A comparison between RMSE.

	One Year	Autumn/Winter	Spring/Summer
SARIMA	21.54	8.13	12.53
NN	6.05	7.65	9.13
Hybrid	4.15	7.08	9.38

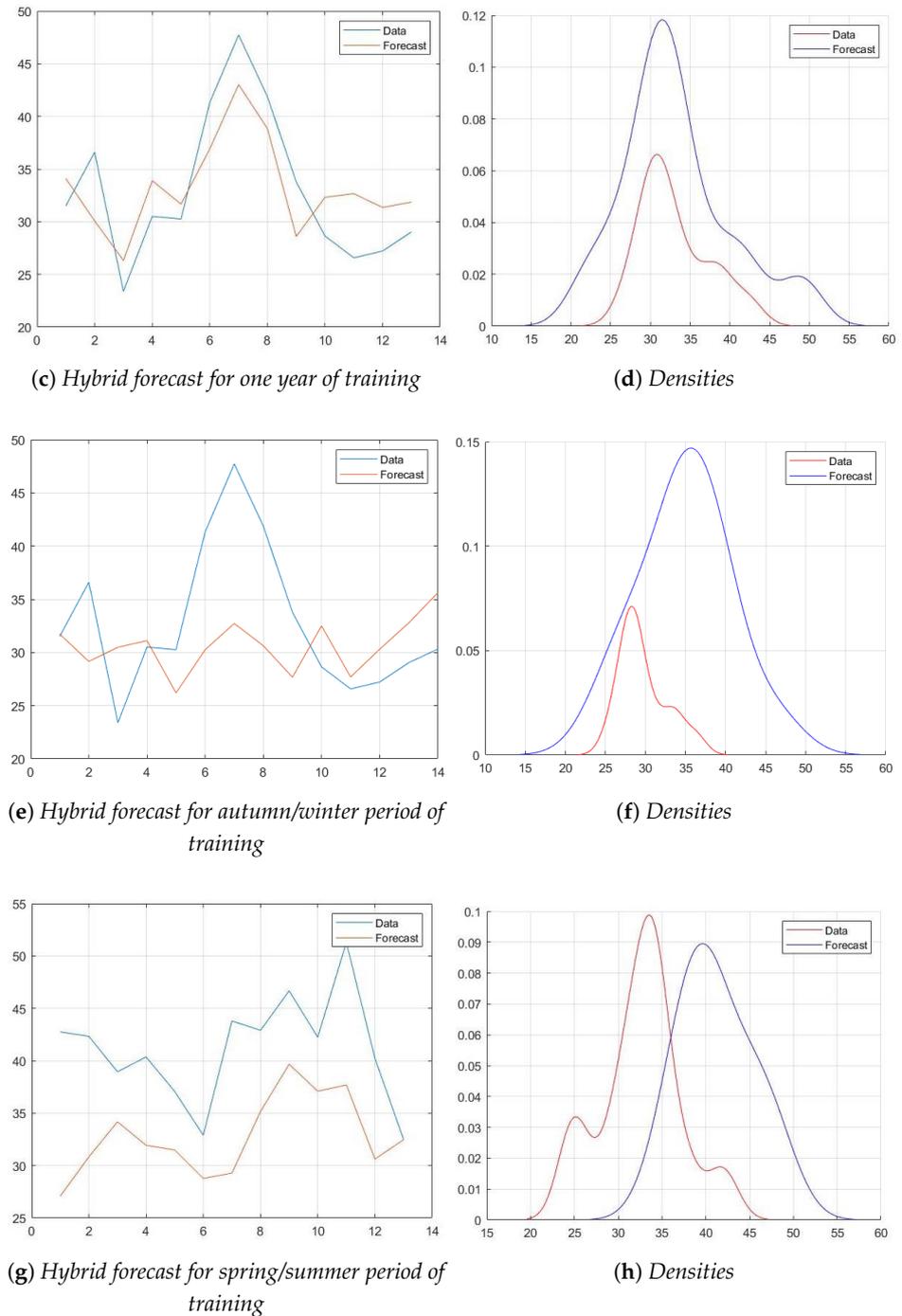


Figure 20. Summary of results of hybrid approach for each time period of training.

In the following, we test our idea to capture the dynamic of the underlying possible stochastic process, and Figure 21 shows the results. We try to obtain good approximations to the investigated processes on the basis of the examined data, while assuming that their development can be well shaped by a certain (previously specified) type of stochastic process. The estimated densities are not precise, so our approach does not yield better results, as we can see from the previous figure. From this point of view, a hybrid approach does not yield any improvement.

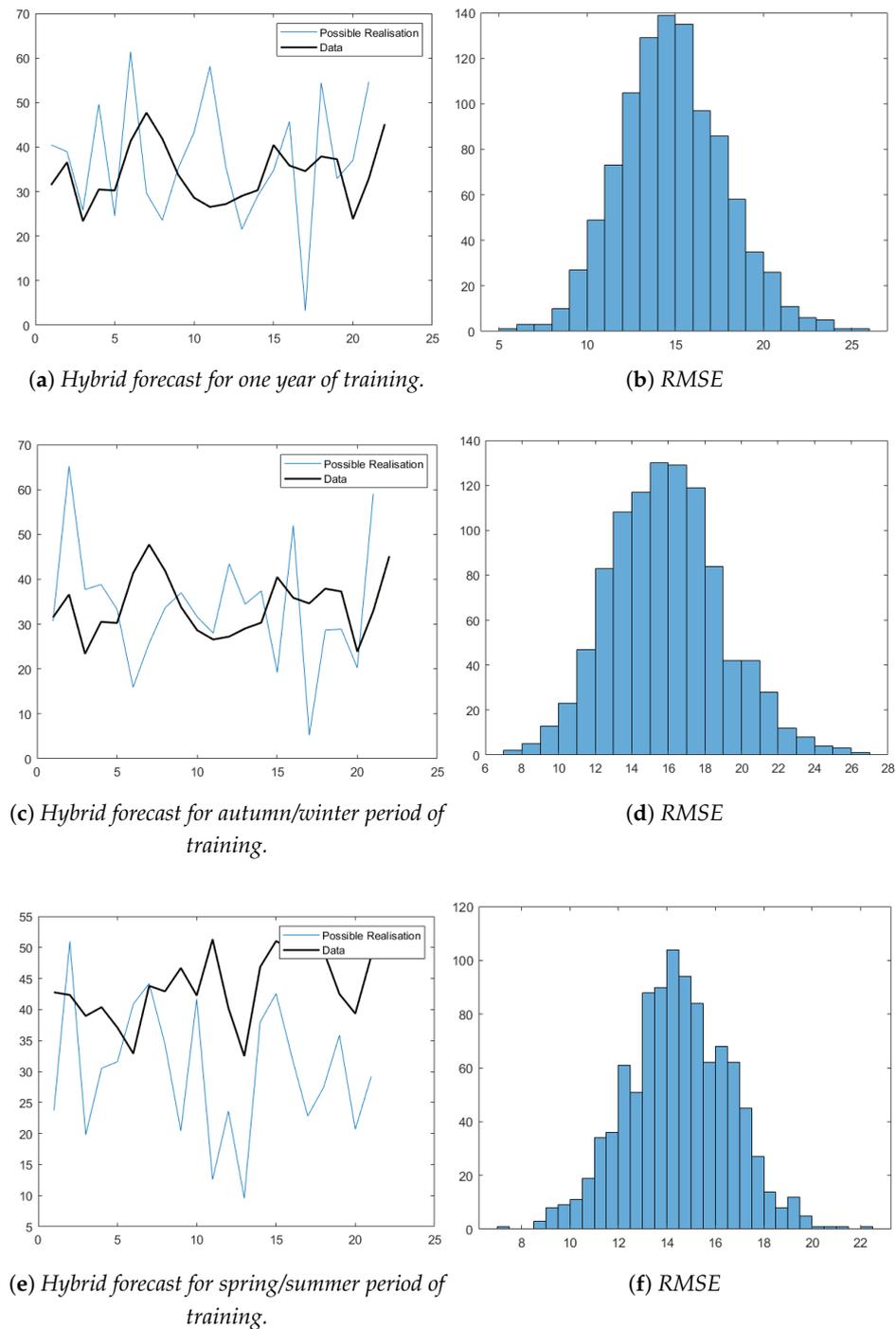


Figure 21. Summary of results of probabilistic hybrid approach for each time period of training for a time horizon of three weeks.

6. Conclusions

In this paper, we gave a general overview of energy price forecasting, highlighting its highest relevant quantitative aspects, smartly using two of the most frequently exploited approaches in order to analyze the corresponding behavior over time. We also included applications that are based on real-time series. In particular, we first focused our attention on (S)ARIMA models, which are widely implemented as typical starting point for most time series analyses. Even if the obtained results have been shown to be quite adequate, better ones can be obtained by using Neural Networks (NNs) based alternatives. Specifically,

we considered LSTM-architectures. The latter being between the most popular within the NNs-oriented community because of the highly accurate results provided, see, e.g., [37]. These improvements are characterized by both a more complex computational structure and a higher data processing time.

In general, one could say that NNs-based results are often better than those that are obtained with an ARIMA model, cf. Table 8. This is because these networks allow for us to obtain and store more information, even for more complex seasonal influences. It is also worth remembering that ARIMA works better in very regular situations. Both are valuable models, but, in this case with this kind of data and context, LSTM can be a suitable tool, not only for our probabilistic approach, but also for point estimation for short and medium term period forecasts, for more details, see With the hybrid approach, as in Section 5.5, we have shown that we can balance our results, also achieving significant improvements w.r.t. the accuracy of the two weeks—point forecast.

Let us finally emphasize that our results regarding the prediction of the (random) electricity price behavior (expressed in the form of a stochastic process) are based on an innovative probabilistic proposal, which gives us remarkable accuracy results. However, one must not forget that these models are developed to answer specific questions. Therefore, they have to be carefully redesigned according to the constraints of the particular problem that a company intends to solve. This fact reflects the need to include more information, such as temperature or another type of exogenous variable, in order to better describe some spikes or other phenomena. We could also consider working with hourly data in order to achieve a higher degree of prediction accuracy.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Acknowledgments: Emma Viviani would like to underline that the material developed within the present paper has been the result of her Erasmus+ experience she spent exploiting the Erasmus+ agreement between the University of Wuppertal and the Dept. of Computer Science of the University of Verona, collaborating with Matthias Ehrhardt, under the supervision of Luca Di Persio.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACF	Autocorrelation Function
AIC	Akaike Information Criteria
BIC	Bayesian Information Criteria
CDF	cumulative distribution function
CRPS	continuous ranked probability score
DM	Diebold Mariano
EPEX	European Power Exchange
EPF	electricity price forecasting
FNN	feedforward neural network
GARCH	generalized autoregressive conditional heteroskedasticity
i.i.d.	independently identically distributed
LASSO	least absolute shrinkage and selection operator
LSTM	Long Short Term Memory
LTSC	Long-Term Seasonal Component
OLS	Ordinary Least Squares

PACF	Partial Autocorrelation Function
PDF	Probability Density Function
PI	prediction intervals
PIT	probability integral transform
PL	pinball loss
QRA	Quantile Regression Average
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal Autoregressive Integrated Moving Average
SDG	Stochastic Gradient Descent
STSC	Short-Term Seasonal Component
VAR	Value-at-Risk

References

- Ding, Y. *Data Science for Wind Energy*; CRC Press: Boca Raton, FL, USA, 2019.
- Singh, S.; Yassine, A. Big Data Mining of Energy Time Series for Behavioral Analytics and Energy Consumption Forecasting. *Energies* **2018**, *11*, 452. [[CrossRef](#)]
- Ziel, F.; Weron, R. Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks. *Energy Econ.* **2018**, *70*, 396–420. [[CrossRef](#)]
- Khairalla, M.A.; Ning, X.; Al-Jallad, N.T.; El-Faroug, M.O. Short-Term Forecasting for Energy Consumption through Stacking Heterogeneous Ensemble Learning Model. *Energies* **2018**, *11*, 1605. [[CrossRef](#)]
- Arora, S.; Taylor, J.W. Rule-based autoregressive moving average models for forecasting load on special days: A case study for France. *Eur. J. Oper. Res.* **2018**, *266*, 259–268. [[CrossRef](#)]
- Sapsis, T.P. New perspectives for the prediction and statistical quantification of extreme events in high-dimensional dynamical systems. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2018**, *376*, 20170133, [[CrossRef](#)] [[PubMed](#)]
- Virginia, E.; Ginting, J.; Elfaki, F.A. Application of GARCH model to forecast data and volatility of share price of energy (Study on Adaro Energy Tbk, LQ45). *Int. J. Energy Econ. Policy* **2018**, *8*, 20170133.
- Weron, R. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *Int. J. Forecast.* **2014**, *30*, 1030–1081. [[CrossRef](#)]
- Ziel, F.; Steinert, R. Probabilistic mid-and long-term electricity price forecasting. *Renew. Sustain. Energy Rev.* **2018**, *94*, 251–266. [[CrossRef](#)]
- Nowotarski, J.; Weron, R. Computing electricity spot price prediction intervals using quantile regression and forecast averaging. *Comput. Stat.* **2015**, *30*, 791–803. [[CrossRef](#)]
- Marcjasz, G.; Serafin, T.; Weron, R. Selection of calibration windows for day-ahead electricity price forecasting. *Energies* **2018**, *11*, 2364. [[CrossRef](#)]
- Zhang, G.; Patuwo, B.E.; Hu, M.Y. Forecasting with artificial neural networks: The state of the art. *Int. J. Forecast.* **1998**, *14*, 35–62. [[CrossRef](#)]
- Weron, R. *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*; Wiley: Hoboken, NJ, USA, 2007; Volume 403.
- Bundesnetzagentur SMARD Strommarktdaten. Electricity Generation in August and September 2019. 2020. Available online: <https://www.smard.de/en/topic-article/5870/14626> (accessed on 13 November 2020).
- Westgaard, S.; Paraschiv, F.; Ekern, L.L.; Naustdal, I.; Roland, M. Forecasting price distributions in the German electricity market. *Int. Financ. Mark.* **2019**, *1*, 11.
- Nowotarski, J.; Weron, R. Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renew. Sustain. Energy Rev.* **2018**, *81*, 1548–1568. [[CrossRef](#)]
- Gneiting, T.; Katzfuss, M. Probabilistic forecasting. *Annu. Rev. Stat. Its Appl.* **2014**, *1*, 125–151. [[CrossRef](#)]
- Nowotarski, J.; Weron, R. On the importance of the long-term seasonal component in day-ahead electricity price forecasting. *Energy Econ.* **2016**, *57*, 228–235. [[CrossRef](#)]
- Aziz Ezzat, A.; Jun, M.; Ding, Y. Spatio-temporal short-term wind forecast: A calibrated regime-switching method. *Ann. Appl. Stat.* **2019**, *13*, 1484–1510. [[CrossRef](#)]
- Di Persio, L.; Frigo, M. Gibbs sampling approach to regime switching analysis of financial time series. *J. Comput. Appl. Math.* **2016**, *300*, 43–55. [[CrossRef](#)]
- Brockwell, P.J.; Davis, R.A. *Introduction to Time Series and Forecasting*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2016.
- Dickey, D.A.; Fuller, W.A. Distribution of the estimators for autoregressive time series with a unit root. *J. Am. Stat. Assoc.* **1979**, *74*, 427–431.
- Bloomfield, P. *Fourier Analysis of Time Series: An Introduction*; Wiley: Hoboken, NJ, USA, 2004.
- Percival, D.B.; Walden, A.T. *Wavelet Methods for Time Series Analysis*; Cambridge University Press: Cambridge, UK, 2000; Volume 4.
- Haykin, S. *Neuronal Networks, A Comprehensive Foundation*; Mc Master University: Hamilton, ON, Canada, 1999.
- Graves, A. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 5–13.

27. Werbos, P.J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Netw.* **1988**, *1*, 339–356. [[CrossRef](#)]
28. Hammer, B. On the approximation capability of recurrent neural networks. *Neurocomputing* **2000**, *31*, 107–123. [[CrossRef](#)]
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
30. Sagheer, A.; Kotb, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing* **2019**, *323*, 203–213. [[CrossRef](#)]
31. Hagfors, L.I.; Kamperud, H.H.; Paraschiv, F.; Prokopczuk, M.; Sator, A.; Westgaard, S. Prediction of extreme price occurrences in the German day-ahead electricity market. *Quant. Financ.* **2016**, *16*, 1929–1948. [[CrossRef](#)]
32. Gentle, J.E. *Random Number Generation and Monte Carlo Methods*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
33. Marsaglia, G.; Tsang, W.W. The Ziggurat method for generating random variables. *J. Stat. Softw.* **2000**, *5*, 1–7. [[CrossRef](#)]
34. Ljung, G.M.; Box, G.E. On a measure of lack of fit in time series models. *Biometrika* **1978**, *65*, 297–303. [[CrossRef](#)]
35. Sheppard, K. *MFE MATLAB Function Reference Financial Econometrics*; Oxford University: Oxford, UK, 2009. Available online: http://www.kevinshppard.com/images/9/95/MFE_Toolbox_Documentation.pdf (accessed on 15 November 2020).
36. Maciejowska, K.; Nowotarski, J. A hybrid model for GEFCom2014 probabilistic electricity price forecasting. *Int. J. Forecast.* **2016**, *32*, 1051–1056. [[CrossRef](#)]
37. Brownlee, J. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*; Machine Learning Mastery: Victoria, Australia, 2018.