Université de Montréal

European Day-Ahead Electricity Price Forecasting

par

Alexandre Beaulne

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

23 mai 2020

Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

European Day-Ahead Electricity Price Forecasting

présenté par

Alexandre Beaulne

a été évalué par un jury composé des personnes suivantes :

Dr Bernard Gendron

(président-rapporteur)

Dr Fabian Bastin

(directeur de recherche)

Dr Manuel Morales

(codirecteur)

Dr Pierre L'Écuyer

(membre du jury)

Résumé

Dans le contexte de l'augmentation de la part de la production énergétique provenant de sources renouvelables imprévisibles, les prix de l'électricité sont plus volatiles que jamais. Cette volatilité rend la prévision des prix plus difficile mais en même temps de plus grande valeur. Dans cette recherche, une analyse comparative de 8 modèles de prévision est effectuée sur la tâche de prédire les prix de gros de l'électricité du lendemain en France, en Allemagne, en Belgique et aux Pays-Bas. La méthodologie utilisée pour produire les prévisions est expliquée en détail. Les différences de précision des prévisions entre les modèles sont testées pour leur signification statistique. La méthode de *gradient boosting* a produit les prévisions les plus précises, suivi de près par une méthode d'ensemble.

Mots clés: *Prévision des prix de l'électricité*

Abstract

In the context of the increase in the fraction of power generation coming from unpredictable renewable sources, electricity prices are as volatile as ever. This volatility makes forecasting future prices more difficult yet more valuable. In this research, a benchmark of 8 forecasting models is conducted on the task of predicting day-ahead wholesale electricity prices in France, Germany, Belgium and the Netherlands. The methodology used to produce the forecasts is explained in detail. The differences in forecast accuracy between the models are tested for statistical significance. Gradient boosting produced the most accurate forecasts, closely followed by an ensemble method.

Keywords: *Electricity Price Forecasting*

Contents

List of tables

List of figures

List of acronyms and abbreviations

ADAM	Adaptive Moment Estimation
ARX	Autoregressive with Extra Input
ATC	Available Transfer Capacity
CDF	Cumulative Distribution Function
CI	Computational intelligence
DM	Diebold-Mariano test statistic
EEX	European Energy Exchange
EMH	Efficient-market hypothesis
ENTSO-E	European Network of Transmission System Operators for Electricity
EPEX	European Power Exchange
EPF	Electricity price forecasting
FFANN	Feedforward artificial neural network
GBM	Gradient boosting
GPU	Graphical processing unit
GWh	Gigawatt hour, 1,000 MWh
IID	Independent and identically distributed
k**NN**	k Nearest Neighbors
LR	Linear regression
MCP	Market-clearing price
MWh	Megawatt hour, a unit of energy
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
NaN	Not a Number
OTC	Over-the-counter
ReLU	rectified linear unit, $y = max(0, x)$
RF	Random forest
RTE	Réseau de Transport de l'Électricité
SGD	Stochastic Gradient Descent
sMAPE	Symmetric Mean Absolute Percentage Error
SNR	Signal-to-noise ratio
SVD	Singular Value Decomposition
SVM	Support Vector Machine

Acknowledgments

I would like to express my gratitude to my co-supervisors Dr Fabian Bastin and Dr Manuel Morales, who always answered my pleas for help in a patient and timely manner. Special thanks also to Céline Bégin, the administrative technician of the graduate program, for her guidance and counsel which made my return to school after years in the industry a much more pleasant experience.

This research was realized via a partnership with BCM Energy, who generously provided the funding, the dataset used in this work, and the domain expertise needed to bring this project to fruition.

Last but not least many thanks to MITACS whose funding greatly facilitated this collaboration between academia and the industry.

Chapter 1

Introduction

In the past few decades, the electricity power sector has undergone a liberalization and deregulation process leading to the creation of public markets in which supply and demand from private participants set prices. Participants in these markets consist of producers, distributors, brokers, speculators, and industrial consumers. In Europe, single country wholesale power markets have emerged in the early 1990s followed by the creation of the European single market in electricity now in effect [**77**].

Electricity has special features absent from most other commodities. First, electricity is uneconomical to store and costly to transport long distances, which makes its distribution across space and time complex relative to durable and fungible commodities. This feature prevents the application of otherwise widespread arbitraging strategies such as cash-and-carry. Second, electricity demand, and supply to a lesser extent, are inelastic. Prices are consequently volatile, with spikes more extreme and more frequent than in other markets. Figure 1.1 illustrates these price outliers. Third, aggregate electricity production requires time to ramp up or down, occasionally causing oversupply and negative prices. Fourth, both electricity supply and demand have strong daily, weekly, and yearly seasonality, as shown in Figure 1.2. These stylized facts result in fragmented markets, both across geographies and time horizons.

Each electricity market has idiosyncrasies of varying levels of significance regarding its auction type(s), participant obligations, regulations, and overall market structure. In Europe, electricity trading falls in one of four market types:

(1) tailor-made transactions occur in the Over-the-counter (OTC) market, where settlement can be physical or cash-settled,
(2) longer-term exposure (in the order of months) can be acquired using cash-settled futures contracts listed on financial exchanges,
(3) auctions for day-ahead electricity delivery are conducted in the spot market[1],

[1]While 'spot' usually refers to the continuously traded underlying asset in financial markets, in electricity trading, especially in Europe, 'spot' usually refers to the day-ahead auction.

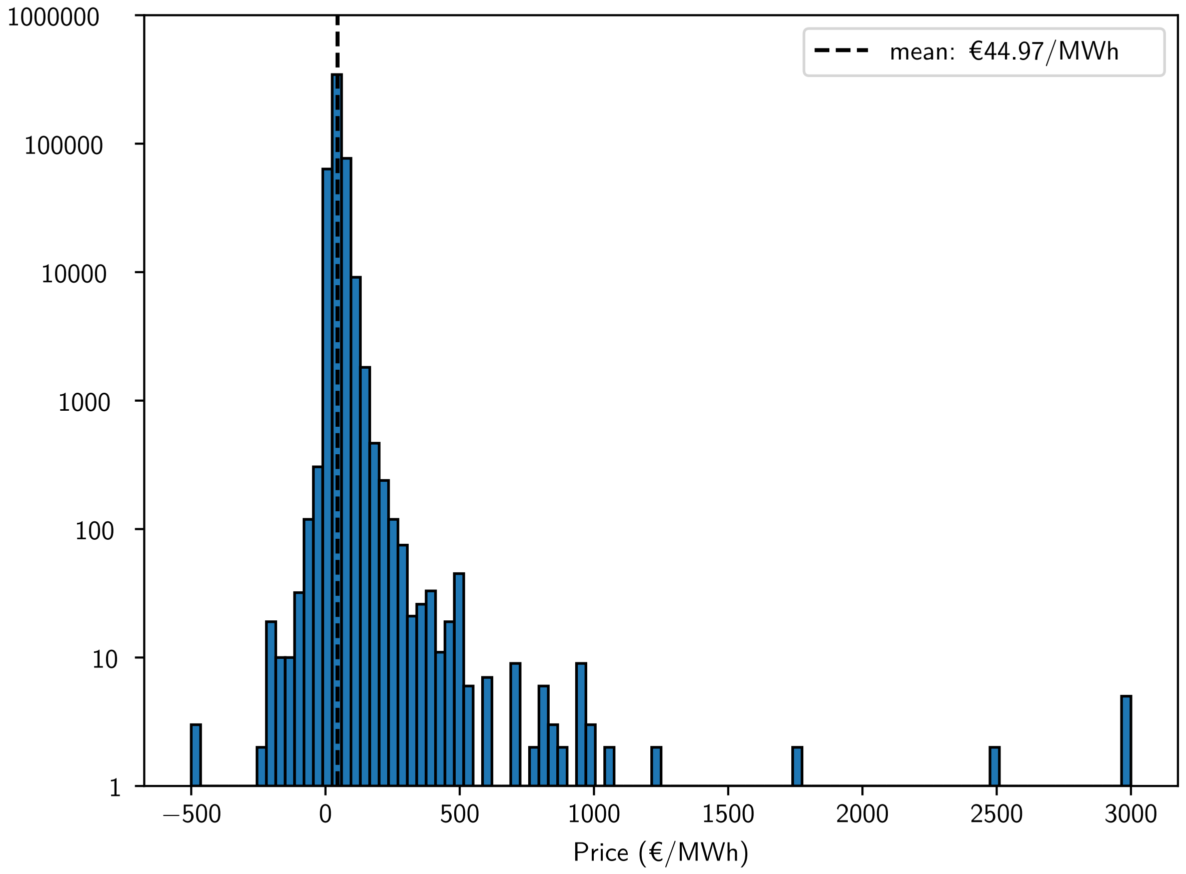


Fig. 1.1. Illustration of extreme outlier electricity prices. The sample consists of hourly day-ahead auction prices for France, Germany, Belgium and the Netherlands between 2005-02-08 and 2020-01-08.

(4) electricity is continuously traded in the intraday market, where last-minute imbalances between supply and demand are corrected.

The day-ahead auction market, referred by (3), is the focus of this work. It is particularly important because of the large volume traded and its role as a reference price for the OTC and futures markets. Unlike most other organized markets, the day-ahead market is not a continuous auction in which transactions occur at any time. Instead, participants all submit bids and offers for delivery of electricity for each hour (or shorter timeframe) of the next day. After the set closing time of the auction, the system operators compute a market-clearing price (MCP) that satisfies transmission constraints between delivery nodes. The auction is *blind*, because participants do not see the orders of other participants, and *uniform-price*, because all transactions occur at the MCP. The MCP is the price at which the supply curve, composed from the aggregation of the supply offers, and the demand curve, composed from the aggregation of the demand bids, intersect. See Figure 1.3 for an actual example. Notice that orders with negative prices are permitted, which results in occasional but rare negative

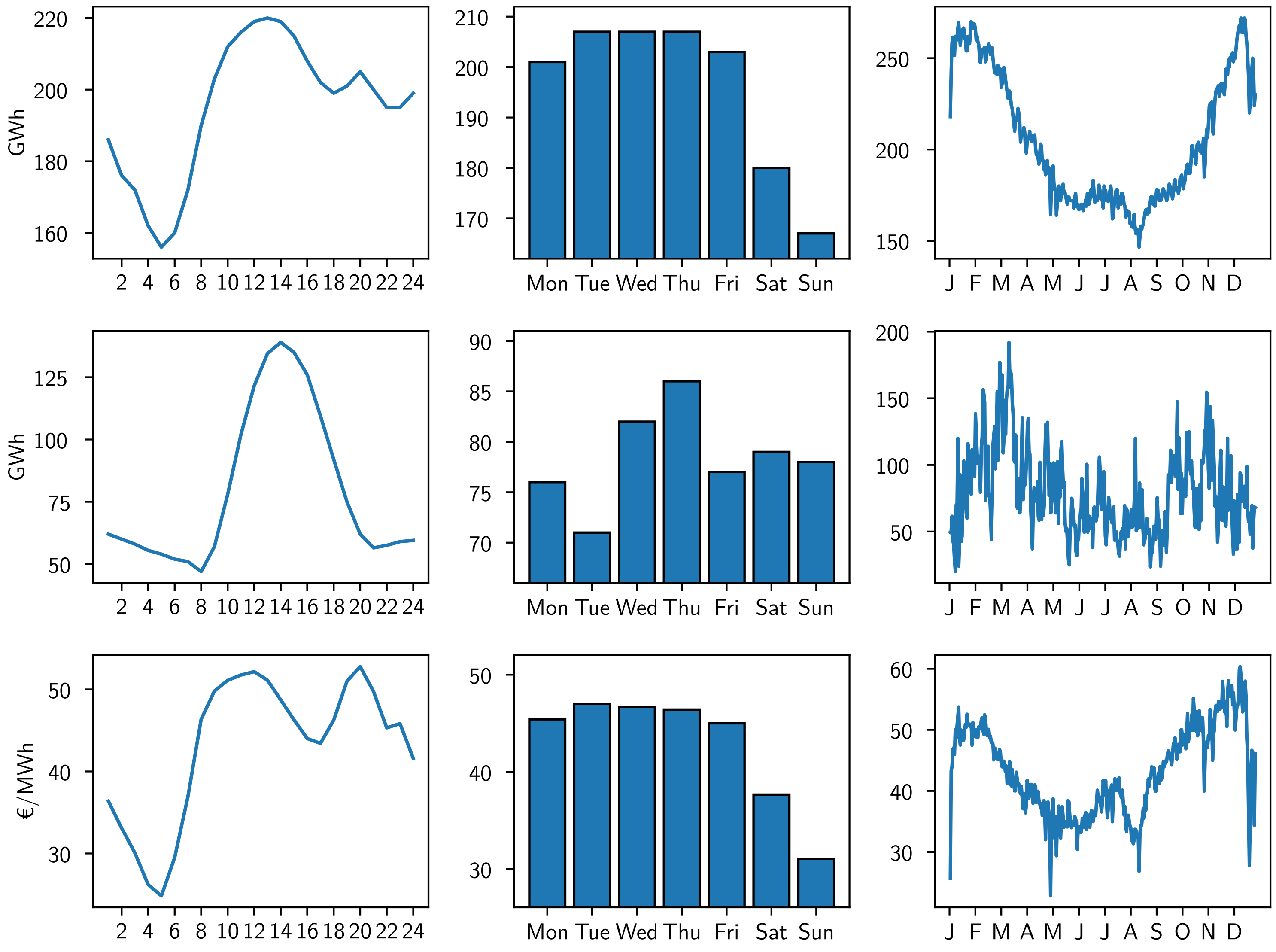


Fig. 1.2. Seasonality in the French electricity sector. The top row displays the median consumption forecast. The middle row displays the median production forecast for wind and solar energy. The bottom row displays the median wholesale electricity price. For each row, the seasonality is shown across the day (left column), across the week (middle column), and across the year (right column).

MCPs. Those occur at the confluence of some factors: low demand, power plants that are expensive to ramp down (e.g., nuclear), or surges in production from intermittent renewable sources.

Accurate electricity price forecasts are valuable for every type of participant in the electricity markets. Producers able to forecast volatile prices with some accuracy can ramp up or down their production schedules by contrasting their costs to projected prices. Distributors and bulk consumers of electricity, such as some energy-intensive factories, often have

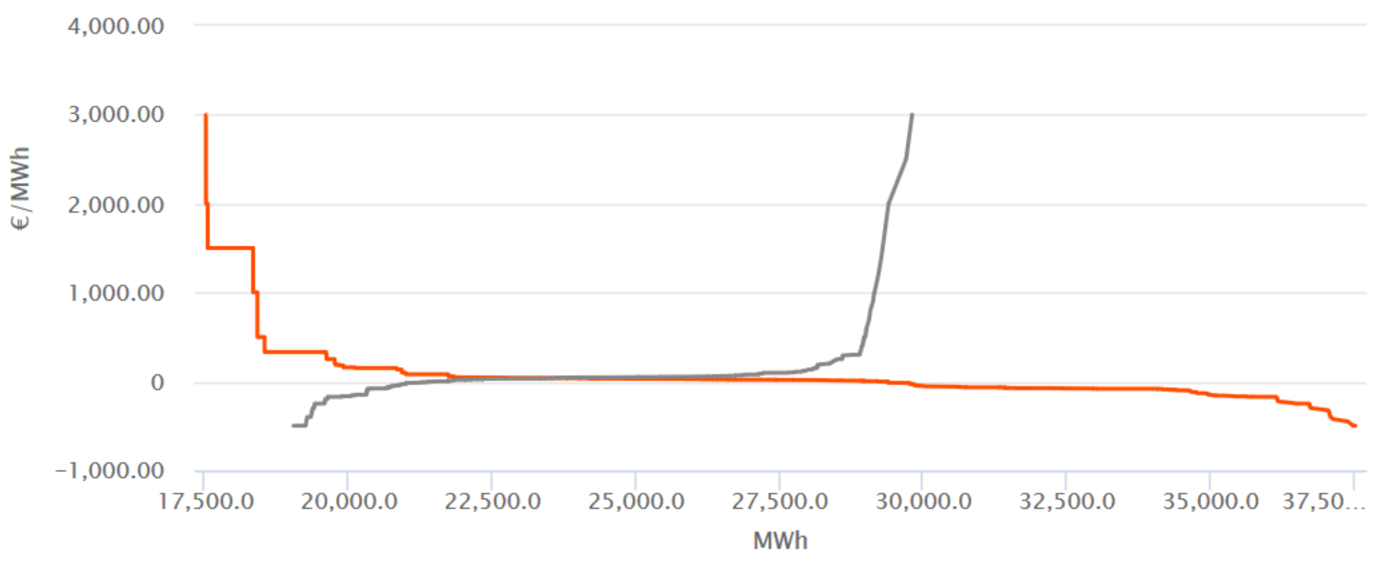


Fig. 1.3. Aggregated supply and demand curves for electricity for delivery in the Germany-Luxembourg node during hour 11-12 of 2019-12-22. The market-clearing price was 37.60 €/MWh and the market-clearing volume was 23,570.2 MWh. Source `https://www.epex.com`.

fixed revenues downstream while paying floating electricity prices upstream. Accurate forecasts allow them to hedge their risk or lock-in electricity at attractive prices. Brokers and speculators can profit from potential mispricings.

Electricity price forecasting (EPF) requires familiarity with many adjacent disciplines. Weather is a critical factor, impacting energy production by the increasingly significant renewable energies via solar irradiance for photovoltaics and wind speed for wind turbines. For this reason, firms participating in the power markets often employ meteorologists. Electricity production, transport, and distribution are purely engineering problems, while price formation through supply and demand lies in the realm of economics. Electricity production is also a matter of public policy since it causes pollution, notably from burning fossil fuels and nuclear fission.

Chapter 2

Background and literature review

The European Power Exchange (EPEX) operates the European day-ahead electricity market [**6**]. A blind auction takes place once a day, every day of the year. Anytime before 12:00 on day D, participants can submit orders for electricity deliverable on each of the twenty-four hours of day $D + 1$. The results of the auction are published as soon as possible from 12:50.

The objective of this research is to build a model to make accurate point forecasts for each of the day-ahead hourly prices, given all the information available before the closing of the auction at 12:00. Note that the forecasts for each day-ahead hour are made at the same time with the same set of available information.

The approach taken by this work is to attempt to *learn* the relationship between arbitrary features and the day-ahead electricity prices from a dataset of historical values of these features and prices. Note that other approaches not using historical data, at least not directly, exist. For example, multi-agent simulations [**57**] and fundamental models [**23**], in which basic physical and economic relationships drive the price of electricity, have been applied to EPF. Such approaches are most often hybrids and used for longer-term forecasts than the day-ahead.

Mitchell [**74**] provides a succinct definition of learning as used here: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." In our case, the experience E is the historical realizations of features and prices. The task T is a regression; the output is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The performance measure P is a measure of distance between the forecasted price and the actual price (see Section 2.5).

This chapter gives an overview of the techniques used in this work. Section 2.1 surveys the existing literature on EPF. Section 2.2 introduces data preprocessing steps often necessary prior to modeling. Section 2.3 gives a high-level description of the models used in this work. Section 2.4 discusses hyperparameters optimization of the models. Section 2.5 describes metrics used to measure and contrast the accuracy of the forecasts made by the different models. Section 2.6 proposes a test to determine the statistical significance of the metrics described in Section 2.5. Finally, Section 2.7 introduces competitive forecasting, an adjacent field rich in practical lessons for the objective of this work.

2.1. EPF research

Research on EPF quickly followed the creation of electricity markets in the 1990s. Since the electricity markets involve diverse activities (engineering, economics, finance, public policy, meteorology, etc.), there has been a wide variety of methods proposed for EPF, some more successful than others [**86**]. Weron [**92**] proposes a five-groups classification to organize EPF techniques: econometrics, computational intelligence, reduced-form, fundamental, and multi-agent. Of those, the first two are relevant to our objective to forecast *short term* electricity prices. Naturally, most techniques do not fall cleanly into one of the five groups but rather are hybrids of two or more approaches.

2.1.1. Econometrics techniques

Econometrics techniques forecast future prices via a weighted combination of past prices and exogenous variables such as the weather and/or electricity production and consumption. The optimal weights for these past variables are typically found via linear regression. Such models are *autoregressive* because the inclusion of past values in the forecast is meant to capture temporal dependence in the prices. An archetypical econometrics technique is the Autoregressive with Extra Input (ARX) model proposed by Misiorek et al. [**73**]:

$$\begin{aligned}\hat{P}_{d,h} &= \beta_{h,0} + \underbrace{\beta_{h,1}P_{d-1,h} + \beta_{h,2}P_{d-2,h} + \beta_{h,3}P_{d-7,h}}_{\text{autoregressive effect}} + \underbrace{\beta_{h,4}P_{d-1,\min}}_{\text{non-linear effect}} \\ &+ \underbrace{\beta_{h,5}L_{d,h}}_{\text{load forecast}} + \underbrace{\beta_{h,6}D_{\text{Sat}} + \beta_{h,7}D_{\text{Sun}} + \beta_{h,8}D_{\text{Mon}}}_{\text{weekdays dummies}} + \epsilon_{d,h}\end{aligned}$$

where $P_{d,h}$ is the electricity price for day d and hour h, $P_{d,\min}$ is the minimum price on day d, $\beta_{h,i}$ is the i^{th} coefficient for hour h, $L_{d,h}$ is the load forecast for day d and hour h, D_x is a dummy variable for weekday x and $\epsilon_{d,h}$ is white noise also known as *innovation.* This model was later used in multiple EPF studies [**35, 56, 63, 70, 75, 89, 91**]. The strength of econometrics techniques is their interpretability. Their weakness is they have difficulty representing nonlinear phenomena.

2.1.2. Computational intelligence techniques

Computational intelligence (CI) techniques are essentially applied statistics with more emphasis on leveraging modern computers to estimate complex nonlinear multivariate functions and less emphasis on producing confidence intervals for those estimates [**42**]. The most well-known models belonging to this category are neural networks (Section 2.3.8), support vector machines (SVMs), random forests, and gradient boosting. This work belongs to this lineage of forecasting techniques, and more in-depth descriptions of the techniques used are available in Section 2.3.

In the first survey of CI techniques applied to EPF, Aggarwal, Saini and Kumar [**9**] found that neural networks tend to outperform econometrics techniques, however, there was no systematic evidence of out-performance of any model over the others on a consistent basis. Likewise, Areekul et al. [**10**] found that neural networks provided accurate forecasts in simulations, and Chen et al. [**25**] found that a combination of neural networks and bagging (Section 2.3.4.1) yielded accurate price forecasts on the Australian National Electricity Market. In a recent study, Lago, De Ridder, and De Schutter [**58**] did a rigorous and systematic (albeit limited to the Belgium market) comparison of forecasting techniques. They found that CI techniques significantly outperform econometrics ones, and that neural networks produce the most accurate forecasts among the CI techniques.

CI techniques have an unparalleled ability to model complex, nonlinear relationships. However, they require vast amounts of data to do so, their predictions are hard to interpret (black box models), their accuracy is sensitive to correct calibration, and they are computationally expensive.

2.2. Data preprocessing

Most learning algorithms for regression have requirements for their input data not met by raw, real-world datasets. This section describes preprocessing steps applied to the raw data in order to use regression algorithms.

2.2.1. Encoding

Learning algorithms most often expect real-valued input data. However, some useful features might be binary (e.g., a day a public holiday or not) or categorical (e.g., France, Germany, or Belgium). Encodings turn categorical features into real-valued ones.

2.2.1.1. *One-hot encoding*

One-hot encoding is the most common method used to transform categorical variables into real-valued ones. It is analogous to dummy variables in statistics, which can be traced in the literature as far back as 1854 when George Boole proposed their use to represent a class [**17**]. A single categorical variable X of cardinality $|X| = N$ is transformed into N binary variables $S_1, \ldots, S_N$, where

$$S_i = \begin{cases} 1.0 & \text{if } X = i \\ 0.0 & \text{otherwise} \end{cases}$$

Table 2.1 shows an example of one-hot encoding. Strengths of one-hot encoding are that it is intuitive to understand and implemented in most statistical software. Weaknesses are that it increases the dimensionality of the data (proportional to the cardinality of the categorical variable) and degrades the performance of tree-based learning algorithms (see Sections 2.3.3, 2.3.5, 2.3.6 and 2.3.7) by diluting the importance of categorical variables compared to continuous ones [**30**].

Table 2.1. Example one-hot encoding of a categorical variable.

country	price		Belgium	France	Germany	price
Germany	40.0		0.0	0.0	1.0	40.0
France	38.0		0.0	1.0	0.0	38.0
Germany	35.0	→	0.0	0.0	1.0	35.0
Belgium	40.0		1.0	0.0	0.0	40.0
France	42.0		0.0	1.0	0.0	42.0
Belgium	39.0		1.0	0.0	0.0	39.0

2.2.1.2. *Ordinal encoding*

Ordinal encoding consists of categorical labels with arbitrary numbers. Table 2.2 shows an example. Its strength is its simplicity. Its weakness is that some learning algorithms may assume an ordering of the encoded variable even if there is none.

Table 2.2. Example ordinal encoding of a categorical variable.

country	price		country	price
Germany	40.0		1.0	40.0
France	38.0		2.0	38.0
Germany	35.0	→	1.0	35.0
Belgium	40.0		3.0	40.0
France	42.0		2.0	42.0
Belgium	39.0		3.0	39.0

2.2.1.3. *Target encoding*

Target encoding has for motivation to map individual values of an independent categorical variable into the expected value of the dependant variable [**72**]. The categorical variable X is replaced by S_i, a Bayesian estimate of the expected value of the target variable Y given that $X = X_i$:

$$\begin{aligned} X_i \to S_i &\overset{\text{def}}{=} \alpha_i \times \mathbb{E}\left[Y | X = X_i\right] + (1-\alpha_i) \times \mathbb{E}\left[Y\right] \\ &\approx \alpha_i \times \frac{\sum_{j=1}^{N} \mathbb{1}\left(X_j = X_i\right) Y_j}{\sum_{j=1}^{N} \mathbb{1}\left(X_j = X_i\right)} + (1-\alpha_i) \times \frac{\sum_{j=1}^{N} Y_j}{N} \end{aligned} \qquad (2.2.1)$$

where N is the number of samples in the dataset, $\mathbb{1}$ is the indicator function, and α_i is the frequency of X_i in the dataset:

$$\alpha_i \overset{\text{def}}{=} \frac{\sum_{j=1}^{N} \mathbb{1}\left(X_j = X_i\right)}{N}$$

If X_i is relatively rare in the dataset, α_i is small, and the estimate is weighted toward the (training) population mean of Y. If X_i is relatively frequent in the dataset, α_i is large, and the estimate is weighted more toward $\mathbb{E}\left[Y|X = X_i\right]$. Note from Equation 2.2.1 that target encoding can handle new data points with values of the categorical variables not present in the training dataset. Table 2.3 displays the application of target encoding to our previous example.

Table 2.3. Example target encoding of a categorical variable.

country	price		country	price
Germany	40.0		38.50	40.0
France	38.0		39.33	38.0
Germany	35.0	$\rightarrow$	38.50	35.0
Belgium	40.0		39.17	40.0
France	42.0		39.33	42.0
Belgium	39.0		39.17	39.0

2.2.1.4. *Leave-one-out encoding*

Leave-one-out encoding is identical to target encoding except that the target value of a given data point is left out when computing the encoding [**4**]:

$$X_i \rightarrow S_i = \alpha_i \times \frac{\sum_{j=1}^{N} \mathbb{1}\left(j \neq i\right) \mathbb{1}\left(X_j = X_i\right) Y_j}{\sum_{j=1}^{N} \mathbb{1}\left(j \neq i\right) \mathbb{1}\left(X_j = X_i\right)} + \left(1 - \alpha_i\right) \times \frac{\sum_{j=1}^{N} Y_j}{N}$$

This reduces the effect of outliers. Table 2.4 illustrates leave-one-out encoding.

Table 2.4. Example leave-one-out encoding of a categorical variable.

country	price		country	price
Germany	40.0		37.67	40.0
France	38.0		40.00	38.0
Germany	35.0	$\rightarrow$	39.33	35.0
Belgium	40.0		39.00	40.0
France	42.0		38.67	42.0
Belgium	39.0		39.33	39.0

2.2.2. Missing data imputation

For various reasons, notably erroneous data collection, real-world datasets often contain missing values. Meanwhile, most learning algorithms require numerical and meaningful values as input. Data points with blanks or NaNs in one or more dimensions could be discarded prior to modeling, however valuable data could be lost doing so. This section describes two schemes to substitute missing values with numerical ones the learning algorithms accept as input.

2.2.2.1. *Mean imputing*

Mean imputing [**60**] is a univariate imputing scheme. It consists merely of substituting missing values in the i^{th} feature by the mean of the non-missing values.

2.2.2.2. *Iterative imputing*

Iterative imputing [**22**] is a more sophisticated, multivariate approach. It works in a round-robin fashion. Every round, a feature X_i with missing values is selected in a sequential manner. This feature is then treated as a dependent variable and is regressed against the other features $X_{j \neq i}$ acting as the independent variables for known X_i's. The missing features of X_i are then filled with the prediction from this regression. The process is repeated for an arbitrary number of iterations or until some stopping criterion is met.

A good survey of other missing data imputation methods is found in Bertsimas, Pawlowski and Zhuo [**14**].

2.2.3. Input standardization

Many learning algorithms require input features to share the same scale, while for others it may not be a requirement but help with learning nevertheless [**78**]. This section describes the two most common scaling methods to standardize input data.

2.2.3.1. *Z-score normalization*

Z-score normalization standardizes features by removing the mean and dividing by unit variance. The standard score Z of a sample X is calculated as:

$$Z = \frac{X - \mu}{\sigma}$$

where μ and σ are respectively the mean and the standard deviation of the feature in the training set.

2.2.3.2. *Min-Max scaling*

Min-Max scaling works by scaling data to a fixed range, typically between 0 and 1. The transformation is given by:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where X_{min} and X_{max} are respectively the minimum and the maximum of the feature in the training set.

Note that statistics required for the transformation (mean and standard deviation for Z-score normalization and minimum and maximum for Min-Max scaling) are computed from the training data, not from the validation or test data. The validation and test data must be standardized using the statistics computed from the training data [**83**].

2.2.4. Target transformation

In order to reduce the influence of outliers, the target variable can be transformed into a uniform or normal distribution prior to learning [**12**]. The transformed target sample y'_i are given by:

$$y'_i = G^{-1}\left(\hat{F}\left(y_i\right)\right)$$

where y_i is the raw target sample, $\hat{F}$ is the empirical cumulative distribution function (CDF) of the training dataset, G is the CDF of the uniform or normal distribution and G^{-1} is the inverse CDF. See Figure 3.4 for graphs illustrating the concept.

A prediction y'_j from the resulting learning algorithm is then transformed back to the empirical distribution as follow:

$$y_j = \hat{F}^{-1}\left(G\left(y'_j\right)\right)$$

2.3. Models

Each subsection in this section describes either a learning algorithm used in this work or a building block used in the construction of one or more of such learning algorithms.

2.3.1. Linear Regression

Linear Regression (LR) is a straightforward learning algorithm described here to provide context and used in this work to provide a baseline model to compare the accuracy of more sophisticated models. As explained above, the objective of a regression task is to build a model that can take an input vector $\mathbf{x} \in \mathbb{R}^n$ and predict the value of a scalar $y \in \mathbb{R}^1$ as its output. The linear regression model is:

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} + b$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of weights and $b \in \mathbb{R}^1$ is the bias term.

In the context of a linear regression, learning signifies finding the parameters (the weights $\mathbf{w}$ and the bias b) that minimizes an error function between the predictions $\hat{y}$ and the actual values y. If the target error function is mean squared error (MSE, Section 2.5.1.1), then there exists a closed-form solution for the optimal parameters. If the input vectors are extended with a constant: $\mathbf{x} = [x_1, \ldots, x_n] \to [1, x_1, \ldots, x_n]$ and the bias term concatenated to the weights: $\mathbf{w} = [w_1, \ldots, w_n] \to [b, w_1, \ldots, w_n]$, then the optimal weights are:

$$\mathbf{w}^*_{\text{MSE}} = \left(X^T X\right)^{-1} X^T Y$$

where $X \in \mathbb{R}^{T \times n}$ is the design matrix made by concatenating the T input vectors together and $Y \in \mathbb{R}^{T \times 1}$ is the vector of actual values.

In practice, computing the inverse of $X^T X$ is both computationally expensive and numerically unstable, so other linear systems resolution techniques are employed. One such technique leverages the Singular Value Decomposition (SVD) [**41**] $X = U \Sigma V^T$. The solution becomes:

$$\mathbf{w}^*_{\text{MSE}} = V \Sigma^{-1} U^T Y$$

Since the rank of Σ is smaller or equal to n, the inverse is usually easy to compute. If the target error function is not MSE, other optimization techniques can be used to find the optimal parameters, notably stochastic gradient descent (SGD, see Section 2.3.8.2).

2.3.2. k-Nearest Neighbors

k-Nearest Neighbors (kNN) [**33**] is a non-parametric method applicable to classification and regression tasks. Unlike every other learning algorithms discussed in this work, no training is needed. Instead, all computation takes place when trying to predict new, unseen inputs. The predicted target for a new input is given by aggregating the k closest inputs from the training set, for some predefined aggregation function and distance measure. For classification tasks, the aggregation function is most often the majority vote. For regression tasks, the aggregation function is most often the weighted average, where the weights are either uniform or inversely proportional to the distance between the input and the neighbors [**31**]. The most popular distance measure is the Euclidian distance. Strengths of kNN are their easy interpretability and the fact that they do not impose any distribution on the data. Weaknesses are that predictions are computationally intensive (and therefore slow) compared to other learning algorithms, and their accuracy suffers greatly in the presence of noisy or irrelevant features or when features lie on different scales.

2.3.3. Decision Tree

A decision tree [**21**] is a binary tree where each node represents a test on one of the input features, each branch is the outcome of the test and each leaf is the target label (for classification tasks) or target value (for regression tasks). There exist many ways to construct decision trees from a training dataset [**81**], two of which are described below.

2.3.3.1. *Greedy construction of a decision tree*

Greedy construction of a decision tree is a recursive top-down algorithm. At each step, all possible splits of every input variables are evaluated, and the split yielding the lowest average Gini impurity [**39, 21**] (for classification tasks) or variance (for regression tasks) across its two child nodes is selected. This binary splitting procedure ends once any of three stopping criteria is met:

(1) if all targets are constant within a leaf.
(2) if the number of targets within a leaf is under a minimum specified count.
(3) if the depth of the tree is over a maximum specified depth.

2.3.3.2. *Random construction of a decision tree*

Random construction of a decision tree is also a recursive top-down algorithm. At each step, a random split is uniformly drawn for each input feature, using the range of values from the training dataset. The random split that minimizes the average variance across its child nodes is selected. The same stopping criteria as the greedy construction (Section 2.3.3.1) are used.

2.3.3.3. *Feature importances*

A useful by-product of the construction of decision trees is estimates of the relative importance of input features [**61**]. Since splits on features are selected in decreasing order of contribution to the accuracy of the decision tree, features split near the top of the tree plays a more important role in the prediction produced by the tree than features split near the leaves of the tree. This observation can help inform which input features are useful for the task and which ones are irrelevant.

Moreover, if multiple decision trees are trained on the same task (see Section 2.3.4), the variance of the feature importance estimate can be reduced by averaging the feature importance of each tree.

2.3.4. Ensemble learning

Ensemble learning is the integration of multiple learning algorithms together to produce a single meta-learning algorithm whose predictive performance is better than any of the constituent learning algorithms used independently. It is now well known that ensemble methods are among the most accurate learning algorithms available [**26, 76, 80**]. This section describes three ensemble methods used in this work.

2.3.4.1. *Bootstrapped aggregation*

Bootstrapped aggregation [**19**] is most commonly referred to as *bagging*. It is a meta-algorithm that combines many unstable predictors to produce a more stable one. Given a standard training dataset of size N, M datasets of size N are sampled with replacement. M independent predictors are then trained on each of those M bootstrapped datasets. In order to predict a new value, the predictions from the M predictors are combined (again, via majority votes for classification tasks and via the arithmetic mean for regression tasks). The predictors are trained independently of each other, potentially in parallel. Bagging can often result in a substantial decrease in variance compared to any of the underlying predictor, sometimes at the price of a small increase in bias.

2.3.4.2. *Boosting*

In 1988, Kearns asked the question [**53**]: "this problem asks whether an efficient learning algorithm [...] that outputs a hypothesis whose performance is only slightly better than random guessing implies the existence of an efficient algorithm that outputs a hypothesis of arbitrary accuracy." Shapire [**84**] introduced boosting as an affirmative answer to this question. A boosted model $F_T(\mathbf{x})$ is an additive model of its constituent predictors $f_t(\mathbf{x})$:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \gamma_t f_t(\mathbf{x}) \tag{2.3.1}$$

where T is the total number of ensembled weak learners and γ_t is the learning rate (also known as shrinkage factor). Each constituent predictor $f_t(\mathbf{x})$ is trained on the results of the boosted model up until its inclusion $F_{t-1}(\mathbf{x})$. As such, constituent of a boosted model are trained *sequentially* and training cannot be parallelized. This becomes clear in the alternative, recursive expression of Equation 2.3.1:

$$F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + \gamma_t f_t(\mathbf{x}) \tag{2.3.2}$$

There exists multiple learning algorithms satisfying this definition, see [**71**] for a unifying framework. Section 2.3.7 describes the boosting learning algorithm used in this work, gradient boosting machines.

2.3.4.3. *Ensemble averaging*

Ensemble averaging is the simplest useful combining method for learning algorithms. It consists of taking the arithmetic mean of multiple regressors in the case of regression tasks or the majority vote of multiple classifiers in the case of classification tasks. Unlike bagging (Section 2.3.4.1) or boosting (Section 2.3.4.2), the constituents predictors usually come from different family of models.

2.3.5. Random Forest

Random forests (RF) [**48, 20**] are bagging (Section 2.3.4.1) of greedily constructed decision trees (Section 2.3.3.1), with a twist: when sampling bootstrapped datasets, only a fraction p out of N features are randomly included. Recommended values [**46**] are $p = \sqrt{N}$ for classification tasks and $p = N/3$ for regression tasks. The purpose of training each tree on a subset of features is to reduce correlation between the trees, which further reduces variance in the aggregated learner.

2.3.6. Extremely Randomized Trees

Extremely randomized trees [**38**], commonly referred to as *extra trees*, are identical to random forests except for two differences:

(1) Extra trees use randomly constructed decision trees (Section 2.3.3.2) instead of greedily constructed ones.

(2) Extra trees do not use replacement when sampling datasets. In other words, each decision tree is constructed using the original training dataset.

2.3.7. Gradient Boosting

Gradient Boosting (GBM) [**34**] is an ensemble method implementing the stage-wise boosting model described in Section 2.3.4.2. At each stage, the additional weak learner f_t (most commonly a greedily trained decision tree (Section 2.3.3.1)) is trained on the *pseudo-residuals* of the previous stage boosted model: $\epsilon_t = L\left(y, F_{t-1}(\mathbf{x})\right)$ where L is a chosen loss function. More explicitely:

$$f_t(\mathbf{x}) = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_{t-1}(\mathbf{x}_i)\right) \tag{2.3.3}$$

Combining Equations 2.3.2 and 2.3.3, one can see why gradient boosting is said to be optimize a loss function by doing gradient descent in *function space*:

$$F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + \gamma_t \underset{f}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_{t-1}(\mathbf{x}_i)\right)$$

F_0 depends on the loss function L. For example, it is the mean of the target value if the loss function is the mean squared error (MSE) or the median of the target value if the loss function is the mean absolute error (MAE).

2.3.8. Feedforward Artificial Neural Network

Feedforward Artificial Neural Networks (FFANNs) are the quintessential deep learning models [**42**]. The *network* part of the name comes from their composition of many connected units. The *neural* part of the name comes from their loose inspiration on neuroscience. The

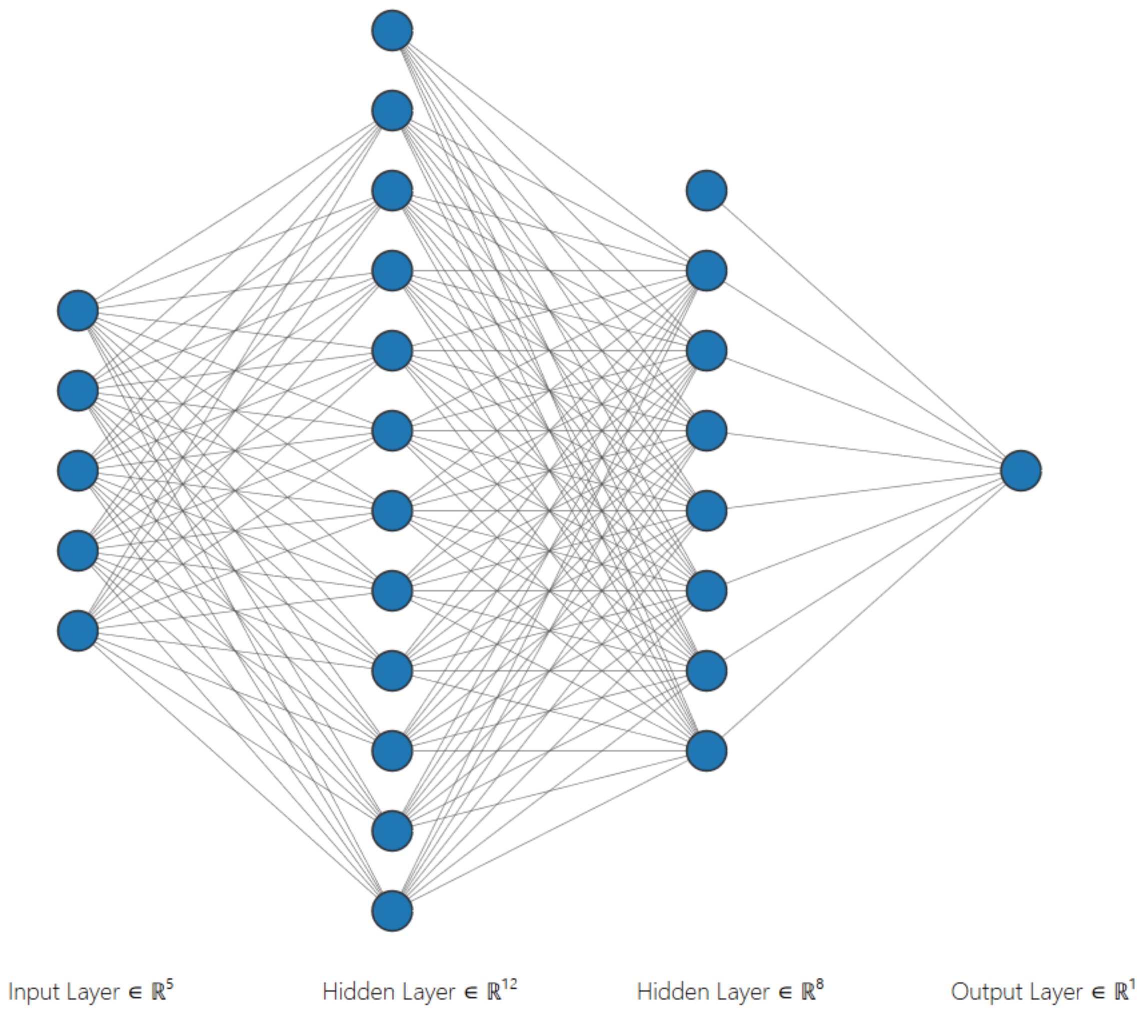


Fig. 2.1. Schematic representation of a FFANN. Each vertice represents an activation unit and each edge represents a weight. The unit with no input at the top of each hidden layer is the *bias unit*; its role is to provide every node with a trainable constant value, analogous to the role of the constant in a linear regression.

feedforward part of the name is because, unlike *recurrent* neural networks, FFANNs have no feedback connections where outputs of the model are fed back into itself.

The building blocks of neural networks are activation units. In the case of FFANNs, these activation units are arranged in layers: an input layer, some hidden layers, and an output layer. See Figure 2.1 for an example architecture. Each activation unit performs a weighted sum of the outputs of the units in the previous layer, applies a nonlinear function (the *activation function*, see Section 2.3.8.1) to the total, and output the result for units in the next layer to use. More formally:

$$h_{i,j} = \begin{cases} g\left(\sum_{k=1}^{n_{i-1}} w_{i-1,j,k}\ x_k + b_{i-1}\right) & \text{if } i = 0 \\ g\left(\sum_{k=1}^{n_{i-1}} w_{i-1,j,k}\ h_{i-1,k} + b_{i-1}\right) & \text{otherwise} \end{cases} \tag{2.3.4}$$

where $h_{i,j}$ is the output value of the j^{th} unit of the i^{th} layer, $g()$ is the activation function, n_i is the number of units in the i^{th} layer, $w_{i,k,j}$ is the weight between the k^{th} unit of the i^{th} layer and the j^{th} unit of the $(i+1)^{\text{th}}$ layer, x_k is the value of the k^{th} input feature and b_i is the bias term for the $(i+1)^{\text{th}}$ layer. Equation 2.3.4 can be expressed more succinctly using vector notation:

$$h_{i,j} = \begin{cases} g\left(\mathbf{w}_{i-1,j}^\top \, \mathbf{x} + b_{i-1}\right) & \text{if } i = 0 \\ g\left(\mathbf{w}_{i-1,j}^\top \, \mathbf{h}_{i-1} + b_{i-1}\right) & \text{otherwise} \end{cases}$$

Finally, the prediction $\hat{y}$ of a network of depth D for input features $\mathbf{x}$, in the case of a regression task, is

$$\hat{y} = \mathbf{w}_{D-1,1}^\top \, \mathbf{h}_{D-1} + b_{D-1}$$

The purpose of training the neural network (see Section 2.3.8.2) is to find the weights $w_{i,j,k}$ than minimize a loss function between the predictions $\hat{y}$ and the actual values y.

2.3.8.1. *Activation functions*

Activation functions introduce the nonlinearity necessary for neural networks to learn arbitrary, nonlinear functions. The universal approximation theorem [**27, 62, 44**] states that, under some mild assumptions on the activation function, a feed-forward network with a single hidden layer with n activation units can approximate any continuous function of n-dimensional input variables.

Given the biological inspiration of neural networks, the first activation functions mimicked action potential firing in the cell. The most widespread first-generation activation function was the sigmoid: $g_{\text{sigmoid}}(x) = {}^{1}\!/_{1+e^{-x}}$.

Subsequently, the rectified linear unit (ReLU) was introduced: $g_{\text{ReLU}}(x) = max(0, x)$. ReLU has been shown to have better biological motivations, mathematical properties, and easier trainability [**43, 40**]. It has become the most popular activation function in neural networks.

Figure 2.2 displays a plot of the sigmoid and ReLU activation functions.

2.3.8.2. *Training*

Training a FFANN with D hidden layers means finding the weights $\theta = \mathbf{w}_1, \ldots, \mathbf{w}_{D+1}$ via empirical risk minimization [**90**]. Define the *cost function* J as the average loss over the training set:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{y}_i | \theta\right)$$

Training is then equivalent to perform the optimization:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta)$$

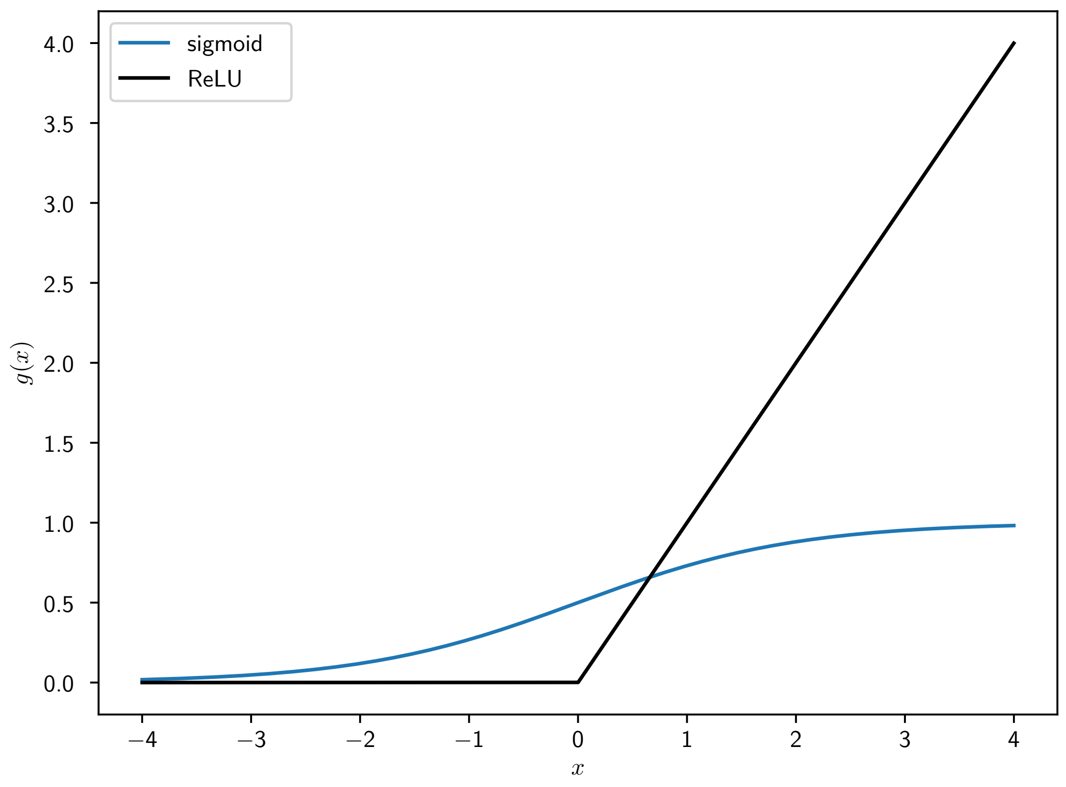


Fig. 2.2. Sigmoid and ReLU activation functions.

This optimization can be done by gradient descent, by iteratively updating an estimate of θ until a minimum is reached:

$$\theta \leftarrow \theta + \gamma \nabla_\theta J(\theta)$$

where γ is the learning rate. Computing $\nabla_\theta J(\theta)$ can be analytically and computationally difficult, however two methods remedy this: *backpropagation* [**82**] and *stochastic gradient descent* (SGD) [**79, 54**] respectively.

Backpropagation is short for *backward propagation of errors.* While the computation of a prediction from an input sample is done by propagating computed activation values *forward* through the neural network, backpropagation works by propagating the gradient of the error $\nabla_\theta J(\theta)$ *backward*, from the last layer to the first one. Gradients of the $(i+1)^{\text{th}}$ layer are used to compute the gradients of the i^{th} layer, in a process reminiscing of backward induction in dynamic programming. Consult [**42**] for an in-depth walkthrough of backpropagation in a FFANN.

Unlike standard gradient descent, which computes the gradient $\nabla_\theta J(\theta)$ on the entire dataset, SGD only computes the gradient on a so-called “minibatch,” a small subset of the dataset. The gradient of the minibatch is an unbiased approximation of the gradient on the full dataset. The minibatch can be as small as a single training example, but generally, using more examples is more cost-effective [**88**]. There have been numerous improvements made to SGD over the years [**18**], but the general concept stays the same. The variant currently most used in practice is Adaptive Moment Estimation [**55**]. Moreover, a criterion for when to stop the training needs to be chosen. The most popular approach is "early stopping" [**85, 16**]: train the network for an arbitrarily large number of epochs[1], storing the parameters and validation error at each epoch and return the set of parameters with the lowest validation error. Early stopping is a form of regularization, helping prevent overfitting.

2.3.8.3. *Batch normalization*

Batch normalization [**51**] is a technique to improve the training speed and stability of artificial neural networks, therefore improving their accuracy. In essence, it consists of normalizing the input features, as discussed in Section 2.2.3.1, across each minibatches instead of across the entire dataset. Additionally, the activation values of the i^{th} layer can also be normalized before being fed to the $(i+1)^{\text{th}}$ layer.

If a dataset is big and the distribution of the input features fat-tailed, normalizing the inputs across the whole dataset squeeze the vast majority of inputs near zero. Batch normalization fixes this since normalization occurs only within small minibatches of data.

2.3.8.4. *Dropout*

Dropout [**87**] is a regularization technique, that is, a technique to prevent neural networks from overfitting the training dataset. During training, for each training sample, the output of activation units in the input or hidden layers of a FFANN are zeroed out with probability p. p is a hyperparameter of the model, fixed before the start of training. Once training is over, all weights in the network are multiplied by $1-p$ [**47**] and no activation units are canceled anymore when it comes the time to make predictions from new input samples.

Dropout works for the same reasons as bootstrapped aggregation (bagging, Section 2.3.4.1). Effectively, it is equivalent to training all potential sub-architectures of a FFANN instead of the FFANN itself, and the predictions of the resulting model are the average of the predictions of all sub-architectures, as with bagging.

As the architecture of the FFANN grows larger (more, wider hidden layers), the number of sub-architectures grows exponentially. Training so many architectures independently would be computationally impossible, but made easy and computationally cheap thanks to dropout.

Note that dropout is not exactly equivalent to bagging, because in the latter the bagged models are independent, while in the former models (sub-architectures) share some parameters (weights).

[1]An epoch is defined as running SGD over the entire training set once.

2.4. Hyperparameters search

The goal when creating a forecasting model using historical data is not to model past data precisely, but rather to forecast the future accurately. The ability of a model to perform well on previously unseen data is called *generalization*. *Underfitting* occurs when a model fails to capture the signal from a training dataset. *Overfitting* occurs when a model confuses noise for signal. Underfitting and overfitting are the two central challenges in modeling. *Capacity* refers to the complexity of functions a model can learn. Underfitting models have too little capacity. Overfitting models have too much capacity.

Training a model means finding settings that maximize generalization. Settings can be separated into two subsets: parameters and hyperparameters. Hyperparameters are settings that control the capacity of a model. Parameters are adapted by the learning algorithm itself. For example, in a polynomial regression $\hat{y}_i = \sum_{j=0}^{N} \beta_j x_i^j$, the degree of the polynomial N is a hyperparameter, and the coefficients β are parameters. Table 4.2 shows the hyperparameters for each learning algorithm introduced in Section 2.3.

Best practices when training a model are to separate a dataset in three subsets as early as possible in the data preparation process: the *training* set, the *validation* set and the *test* set. The test set is kept aside until the training process is fully completed. It is used to compute the out-of-sample error, which gives the expected performance of the model on new inputs. The parameters of the model are trained with different combinations of the hyperparameters on the training set, and the model with the combination of hyperparameters that yield the best performance on the validation set is deemed the best. Since hyperparameters control the capacity of the model, they cannot be optimized on the training set, because the hyperparameters yielding the maximum capacity would always come out ahead, resulting in overfitting.

In modeling, the samples in the dataset are often assumed to be *IID*: independent and identically distributed. This assumption considerably facilitates the assignment of samples to the training, validation, and test sets: randomly assign data points uniformly in the desired proportions. For time series, especially the ones considered in this work, these assumptions are incorrect. Data points close to each other in time have correlation (therefore are not independent), and the data generation distribution changes over time (therefore the samples are not identically distributed). This breach of IID assumptions is particularly marked for the dataset described in Section 3.2, notably because of the marked increase in renewable electricity production over the period spanning the dataset. It is therefore critical to prevent information from the future to contaminate the validation and test sets, otherwise, models would use information not available at the time of the forecast and their performance would be overestimated.

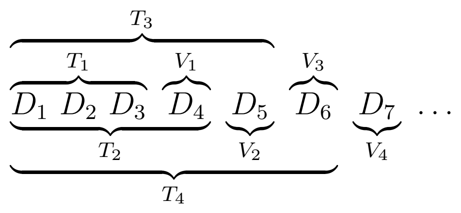


Fig. 2.3. Stylized walk-forward validation example. D_i is the ith day, T_j is the jth training set, and V_j is the jth validation set. In this made-up example, the training sets start with a length of three days and expand, while the validation sets all have a length of one day.

Two schemes are commonly used to validate hyperparameters for time series datasets: contiguous data splits and walk-forward validation. With contiguous data splits, the dataset is split into three contiguous subsets: training, validation, and test, ordered chronologically. See Figure 3.2b for an illustration. This validation scheme is the one used in [**58, 59**]. In walk-forward validation [**28**], the model is trained multiple times using an expending-length training set and a fixed-length validation set. See Figure 2.3 for a stylized illustration.

2.5. Performance metrics

Performance metrics are used to compare the relative accuracy of forecasting methods.

2.5.1. Notable forecasting performance metrics

This section describes four performance metrics, along with their strengths and weaknesses.

2.5.1.1. *Mean Squared Error*

The MSE is the most widespread error metric thanks to its tractable mathematical properties. It is defined as

$$\mathrm{MSE} \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

where N is the sample size, y_i is the ith actual value and $\hat{y}_i$ is the ith predicted value. MSE has two problems when used to measure a forecasting method's performance [**24**]:

(1) Outliers dominate the value of the metric, which may or may not be desired.
(2) It is scale-dependent, so forecasts of different time series cannot be compared.

2.5.1.2. *Mean Absolute Error*

MAE is defined as

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

MAE does not suffer from the excessive influence of outliers like the MSE, however it is also scale-dependent.

2.5.1.3. *Mean Absolute Percentage Error*

Mean Absolute Percentage Error (MAPE) was, for a long time, one of the most used performance metrics in forecasting settings. It is defined as

$$\text{MAPE} \overset{\text{def}}{=} \frac{100}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

A strength of MAPE is that it is unit free; the MAPE of forecasts of different time series can be compared directly. However, MAPE has other problems [**64**]:

(1) It is undefined for any actual target value of zero ($y_i = 0$ causes a division by zero).
(2) It is asymmetric; interchanging y_i and $\hat{y}_i$ yields a different performance metric, even if the absolute error is identical before and after the substitution. This asymmetry causes negative errors ($\hat{y}_i < y_i$) to be more heavily penalized than positive ones ($\hat{y}_i > y_i$).

2.5.1.4. *Symmetric Mean Absolute Percentage Error*

Symmetric Mean Absolute Percentage Error (sMAPE) addresses the asymmetry of MAPE. It is defined as [**64**]:

$$\text{sMAPE} \overset{\text{def}}{=} \frac{200}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}$$

sMAPE is the performance metric used in this work.

2.5.2. Distinction between performance metrics and loss functions

It can be challenging to grasp the subtle difference between performance metrics and loss functions. Both measure the distance between sample observations and labels predicted by a learning algorithm. In many cases, the same function can act as the performance metric and the loss function. However, performance metrics and loss functions play a different role in the exercise of producing a learning algorithm.

The goal of a performance metric is to measure the prediction performance of a learning algorithm in the context of the task at hand. It is directly related to the economic objectives of the agent using the learning algorithm. The choice of performance metric is a decision theory concern; the chosen performance metric should maximize the utility of the agent using the learning algorithm. If the performance metric is appropriately chosen, an agent should always prefer the learning algorithm with the best performance metric out of the available ones. However, it is sometimes difficult to optimize a learning algorithm for a given performance metric if the latter is discontinuous or not strictly convex. In such a case, surrogate loss functions are used.

Loss functions are an optimization parameter, similar to the learning rate in gradient descent or population size in genetic algorithms. The goal of the optimization is to find the learning algorithm parameters that optimize the performance metric. However, sometimes the optimization algorithm cannot work with the performance metric directly. In those cases, a surrogate loss function is substituted in the optimization algorithm to act as a proxy for the performance metric. Different surrogate loss functions can act as proxies for the same performance metric and can be treated as hyperparameters of the learning algorithm.

To illustrate this point, consider the classification task and the gradient descent optimization algorithm. The objective is to classify datapoints accurately. Therefore accuracy is the natural performance metric. However, accuracy is a step function; either a datapoint is correctly labeled or not. Consequently, the gradient of the accuracy function is zero almost everywhere, and gradient descent is ineffective. In order to use gradient descent effectively, the cross-entropy loss function is substituted for accuracy. Since cross-entropy gradients have the same sign as the accuracy function gradients, but the cross-entropy function is smooth and strictly convex, gradient descent can effectively train a learning algorithm to optimize accuracy.

A similar but subtler substitution is also possible for regression tasks. For example, one might use the linear regression learning algorithm minimize the sMAPE (Section 2.5.1.4) but use the MSE surrogate loss function since it yields a closed-form solution to the optimization problem (Section 2.3.1).

2.6. Statistical significance of forecasts performance

While the performance metrics discussed in Section 2.5 allow us to compare the accuracy of different forecasts, they do not guarantee that differences in accuracy are statistically significant. The most popular approach do to so is the *Diebold-Mariano* test [**29**].

2.6.1. Diebold-Mariano test

Suppose there is a pair of forecasts with different accuracy according to a performance metric $g(\mathbf{y}, \hat{\mathbf{y}})$ such as the ones in Section 2.5. Let $y_1, \ldots, y_N$ be the actual values, $\hat{y}_1^+, \ldots, \hat{y}_N^+$ the predictions of the more accurate forecast, and $\hat{y}_1^-, \ldots, \hat{y}_N^-$ the predictions of the less accurate forecast. The objective is to know whether the difference in accuracy between the two forecasts is statistically significant. Define the *loss differentials* to be $d_i \overset{\text{def}}{=} g(y_i, \hat{y}_i^+) - g(y_i, \hat{y}_i^-)$; $i = 0, \ldots, N$. The null hypothesis of the one-sided Diebold-Mariano test is that the predictive accuracy of the supposedly more accurate forecast is equal or worst than the supposedly less accurate forecast:

$$\text{Diebold-Mariano} \begin{cases} H_0 : \mathbb{E}\left[\mathbf{d}\right] \geq 0 \\ H_1 : \mathbb{E}\left[\mathbf{d}\right] < 0 \end{cases}$$

It is natural to base the test on the mean of $\mathbf{d}$ and its variance. The mean is given by $\bar{d} = \sum_{i=1}^{N} d_i/N$. The variance is more difficult to compute because the d_i are likely to be autocorrelated. It can be shown that the variance of $\bar{d}$ is, asymptotically,

$$V(\bar{d}) \approx \frac{\left[\gamma_0 + 2\sum_{k=1}^{h-1}\gamma_k\right]}{N} \tag{2.6.1}$$

where h is the number of steps after which there are no more autocorrelations and γ_k is the k^{th} autocovariance of $\mathbf{d}$, estimated by

$$\hat{\gamma}_k = \frac{\sum_{i=k+1}^{N}(d_i - \bar{d})(d_{i-k} - \bar{d})}{N} \tag{2.6.2}$$

Note that if $k = 0$ then the autocovariance is equal to the population variance. The Diebold-Mariano z-test statistic is then

$$\text{DM} = \frac{\bar{d}}{\sqrt{V(\bar{d})}} \tag{2.6.3}$$

where the variance $V(\bar{d})$ is obtained by substituting the estimated autocovariances from Equation 2.6.2 in Equation 2.6.1. Under the null hypothesis, DM has an asymptotic standard normal distribution.

2.6.2. Harvey's adjustment

While the Diebold-Mariano test performs satisfactorily for moderately large samples, it is quite oversized for small and medium sample sizes. Harvey, Laybourne and Newbold [**45**] proposed two adjustments to mitigate this. First, the test statistic is substituted by

$$\text{DM}^* = \sqrt{\frac{N + 1 - 2h + h(h-1)/N}{N}} \times \text{DM}$$

where DM is the original test statistic from Equation 2.6.3. Second, the test statistic is compared to critical values from the Student's t distribution with $N - 1$ degrees of freedom, rather than from the standard normal distribution.

2.7. Competitive Forecasting

Some problems have plagued the quality and comparability of EPF research: (i) the use of different datasets (ii) different software implementations of forecasting models (iii) different error metrics used (iv) uneven statistical rigor across studies and (v) the reticence of researchers to publish negative findings. Consequently, many published results contradict each other.

A powerful way to mitigate those problems is *forecasting competitions.* By explicitly defining the success measure ex-ante and by requiring researchers to commit their forecasts before the realized target values are available, forecasting competitions solve the selection bias endemic in forecasting research.

The most prominent times series forecasting competitions are the Makridakis series of competitions [**66, 67, 68, 69, 65**], in which hundreds of participants submit forecasts for thousands of time series to earn prizes and recognition. The two principal takeaways for these M-competitions are:

- Ensemble of forecasts (Section 2.3.4.3) always outperform individual "pure" models.
- The initial superiority of econometrics approaches (Section 2.1.1) over computational intelligence ones (Section 2.1.2) has vanished in the latest iteration of the competitions.

While there is currently an explosion in forecasting competitions, few address the topic of EPF. A notable exception was the price forecasting track of the 2014 Global Energy Forecasting Competition [**49, 50**], in which the top entry (out of 117) used an ensemble of computational intelligence techniques, and the second entry used an ensemble of econometrics techniques.

Chapter 3

Methodology

3.1. Context

The research presented in this work is the result of a collaboration between the author and BCM Energy [**7**], a private company active in the European energy markets headquartered in Lyon, France. A sponsorship from Mitacs [**8**] partially funded the collaboration. The objective of the collaboration was to apply the forecasting techniques covered in Chapter 2 to BCM Energy's proprietary dataset. The following three chapters expose the bulk of the findings. Some implementation details, immaterial to the conclusions of this work but deemed intellectual property by the business partner, are omitted.

3.2. Dataset

The dataset underpinning this work consists of time series data gathered between May 21st 2015 until December 1st 2019, totalling $N = 159,036$ samples. 15 input features were selected based on the availability of the data and judgment from human experts regarding the factors material to day-ahead electricity prices. Table 3.1 lists properties of each selected feature. Figure 3.1 shows the frequency of missing data in each feature using a density matrix [**15**].

Feature 1 is the country of delivery of the transacted electricity. Four countries were selected based on the business interests of the industry partner and to overlap countries studied in other recent EPF research [**58, 59**]. Note that the selected countries are conterminous. Since electricity can be imported and exported to neighboring countries, proximity is a factor in electricity prices [**59**]. Features 2, 4, and 5 are the day of the week, week of the year and hour of the day, respectively. They are included to capture the strong seasonalities of electricity prices (Figure 1.2). Feature 4, week of the year, was chosen to capture yearly seasonality over the month of the year or season because it is finer-grained. Feature 3, year, is included to potentially capture a secular trend or regime change during the period of the dataset. For example, the rise in electricity production from renewable sources over the last

Table 3.1. Properties of the input features

#	**name**	**units**	**domain**	**fraction of missing values**
1	country	N.A.	France, Germany, Belgium, Netherlands	0.00
2	weekday	weeks	1 to 7	0.00
3	year	years	2015 to 2019	0.00
4	week	weeks	1 to 52	0.00
5	hour	hours	1 to 24	0.00
6	holiday	N.A.	True or False	0.00
7	wind forecast	MWh	0.0 to 44,360.4	0.41
8	solar forecast	MWh	0.0 to 32,133.0	0.41
9	consumption forecast	MWh	6,019.7 to 236,052.0	0.27
10	inbound capacity	MWh	76 to 10,279	0.00
11	outbound capacity	MWh	210 to 9,660	0.00
12	last trade price	Euros (€)	-21.00 to 220.00	0.23
13	last trade size	MWh	3 to 1,000	0.23
14	last trade age	seconds	0 to 31,590	0.23
15	previous day price	Euros (€)	-500.00 to 874.01	0.00

few years plausibly translates into a greater influence of climatic conditions on electricity prices. Everything else equal, electricity consumption is usually lower during the holidays. Whether a given day is a public holiday or not is therefore included as feature 6. Features 7 and 8 are wind and solar electricity production forecasts, respectively. Wind speed and solar irradiance forecasts were excluded because it is hoped that wind and solar electricity production forecasts encapsulate all the information provided by them. Feature 9 is an electricity consumption forecast. The wind production forecast, the solar production forecast, and the consumption forecasts are produced by Réseau de Transport d'Électricité (RTE, [**5**]) in the case of France and by IHS Markit [**3**] in the case of Germany, Belgium, and the Netherlands. Features 10 and 11 are cumulative inbound and outbound transmission capacity, respectively. The European Network of Transmission System Operators for Electricity (ENTSO-E, [**2**]) publishes bidirectional transmission capacity between each conterminous EU member countries. Supply and demand imbalances between countries cannot be netted out if there is insufficient transmission capacity, affecting equilibrium prices. In order to prevent an explosion in the number of features, the transmission capacities are summarized by summing the inbound and outbound transmission capacities for each country. The European Energy

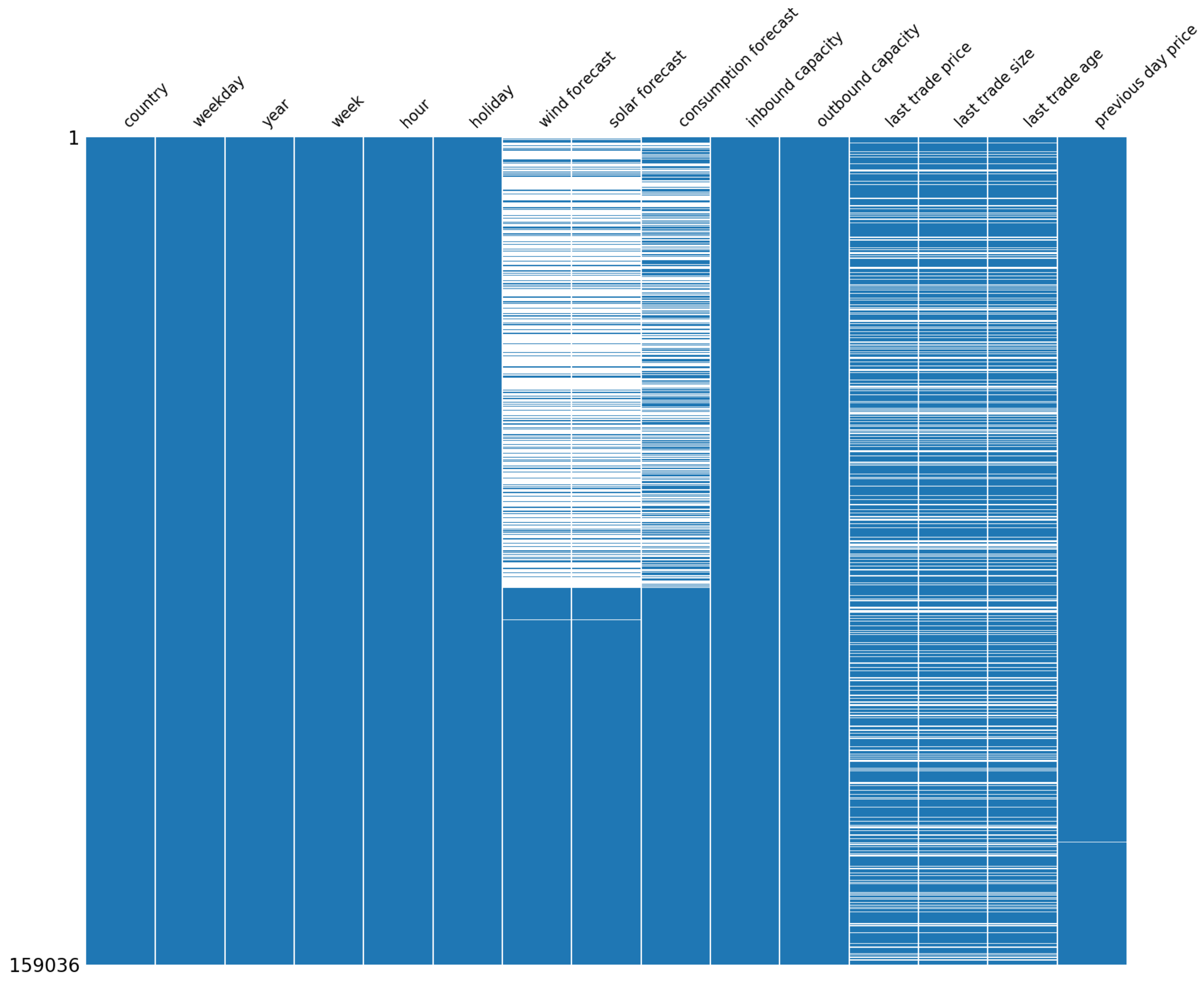


Fig. 3.1. Density matrix of the features in the dataset. Blank lines signify missing values.

Exchange AG (EEX, [**1**]) conducts a parallel, continuous auction for cash-settled futures whose underlying is the average of EPEX day-ahead hourly auction prices. The activity on these futures is a useful proxy for the forecasts of other market participants. Feature 12, 13, and 14 are the price, size, and age of the last market trade made before the closing of the EPEX day-ahead auction. Finally, feature 15 is simply the previous day's price for a given hour for a given country. It is a naive but often accurate prediction for the next day's price. The target variable is the EPEX SPOT day-ahead hourly auction price for each of the four selected countries.

3.3. Experimental setup

This section describes how the research was conducted and expands on the implementation of the techniques used.

3.3.1. Dataset split

The very first step is to isolate an out-of-sample test set out of the entire dataset. It is critical to do so before any other data processing work to avoid contamination or information leakage. The last year of available data, from December 2nd, 2018 until December 1st, 2019, was used for the out-of-sample test set. While the test set proportion of the complete dataset is larger (22%) than typically used, it is preferable for it to cover a full year of data given the strong yearly seasonality of electricity prices. A test set shorter than a year might promote a model that overperforms over the period but underperforms the rest of the year.

The remainder of the dataset is used for training and validation. Unfortunately, the same validation scheme could not be used across all models. For all models except FFANN, walk-forward cross-validation (Section 2.4), with expanding training sets starting with a length of 730 days and validation sets of fixed 30 days length (Figure 3.2a), was used. The long training time of the FFANN model, despite the use of graphical processing units (GPUs), prohibited the use of the walk-forward cross-validation scheme. Instead, a contiguous validation scheme (Section 2.4), with a single validation test set of 30 days (Figure 3.2b), was used.

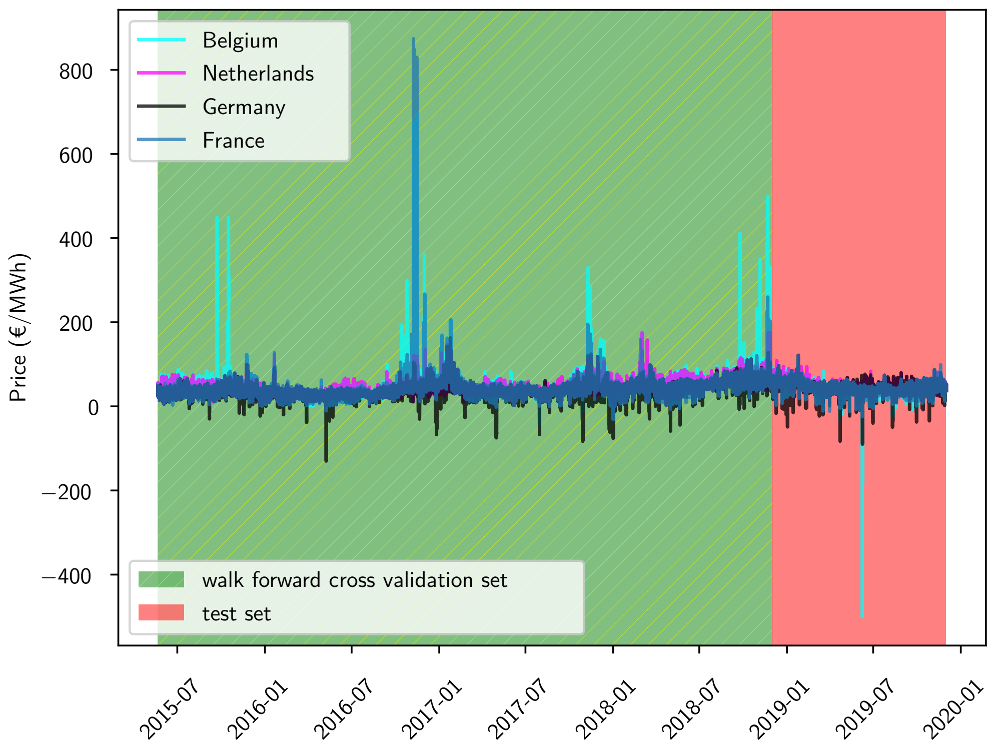
3.3.2. Hyperparameters search

The next step after splitting the dataset is to perform hyperparameter optimization for each of the models described in Section 2.3. In addition to each model's intrinsic hyperparameters (Table 4.2, second column), the choice of technique used for each data preprocessing step described in Section 2.2 is also treated as a hyperparameter of the model.

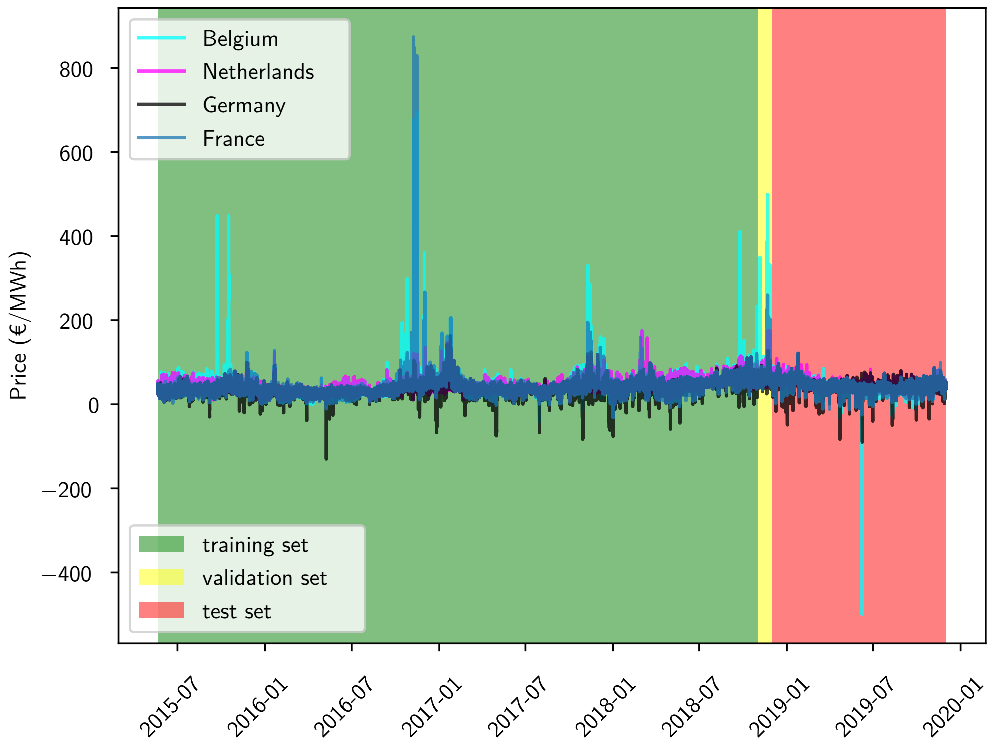
A figurative multidimensional grid is constructed, with each dimension being a hyperparameter. For hyperparameters taking on discrete values, their dimension includes every possible value. For hyperparameters taking on ordinal or continuous values, some evenly spaced (on a linear or logarithm scale) candidate values are selected within a reasonable range. The size of the grid is the cartesian product of every hyperparameter's number of possible values. Therefore, the grid can get very large.

Each node in the hyperparameter grid represents one possible combination of hyperparameters values. For each node (Figure 3.3):

(1) data is preprocessed
(2) a model is trained on the training set
(3) the value of the sMAPE performance metric is computed for the trained model on the validation set



(a) Walk-forward data split used to train all models except FFANN.



(b) Contiguous data split used to train the FFANN model.

Fig. 3.2. Illustration of the two validation schemes used.

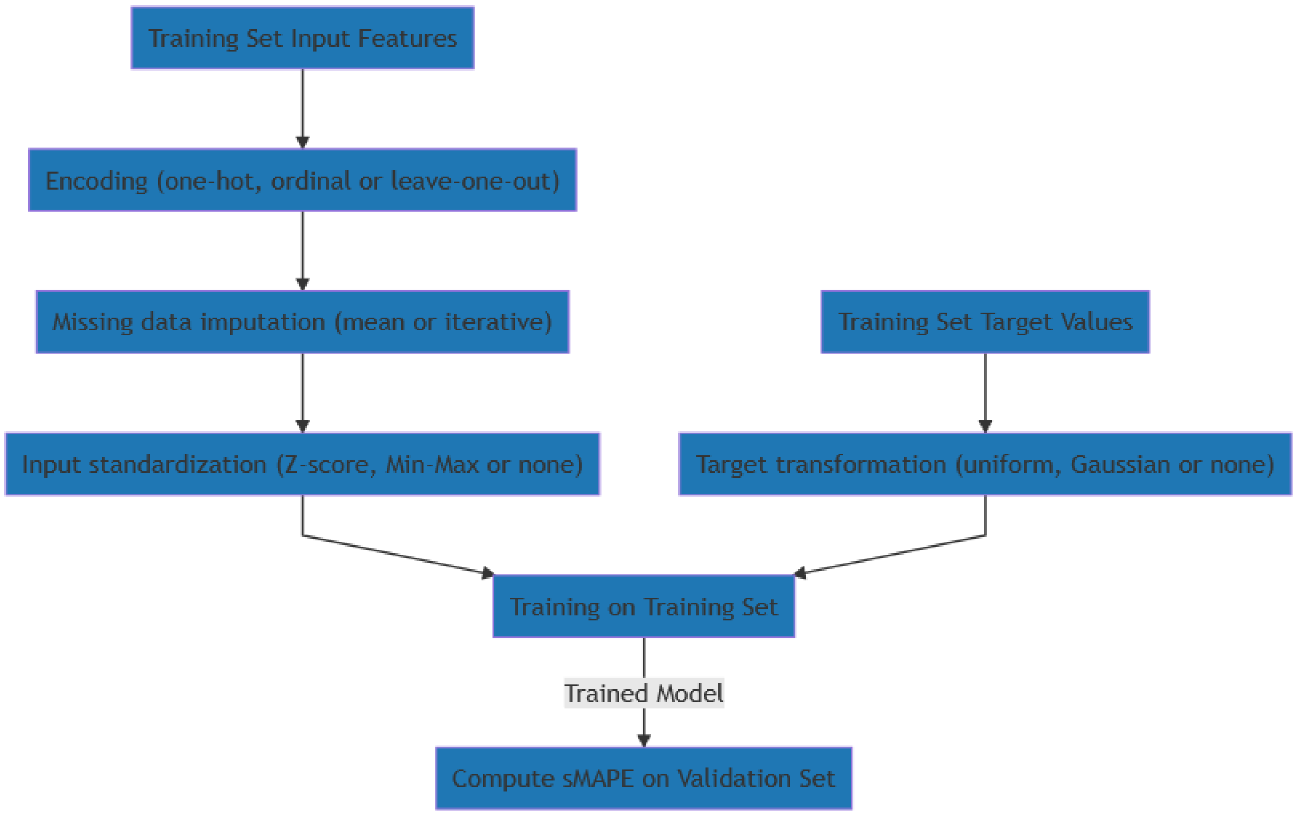

Training Set Input Features
Encoding (one-hot, ordinal or leave-one-out)
Missing data imputation (mean or iterative)
Training Set Target Values
Input standardization (Z-score, Min-Max or none)
Target transformation (uniform, Gaussian or none)
Training on Training Set
Trained Model
Compute sMAPE on Validation Set

Fig. 3.3. Data preprocessing pipeline.

A grid search is conducted, and the node with the lowest sMAPE is chosen as the optimal hyperparameters for a given model.

Figure 3.4 illustrates the application of the target transformation described in Section 2.2.4. The top panel shows the target values before any transformation. They are tightly clustered around small positive values. Applying a rank-based transformation to the target values spreads the data more evenly, which may help learning algorithms discover subtle relationships. Target values are spread before training the models, and interim predictions of the models are projected back to the empirical distribution before comparison to the actual values.

Figure 3.5 illustrates the optimization learning curves of the FFANN model on the training and validation sets. One epoch signifies performing stochastic gradient optimization over the complete training dataset. After each epoch, the loss function is computed over both the training set and the validation set. Learning curves are particularly useful to diagnose problems with training neural networks, such as underfitting or overfitting, or whether the dataset is suitably representative to solve the task at hand.

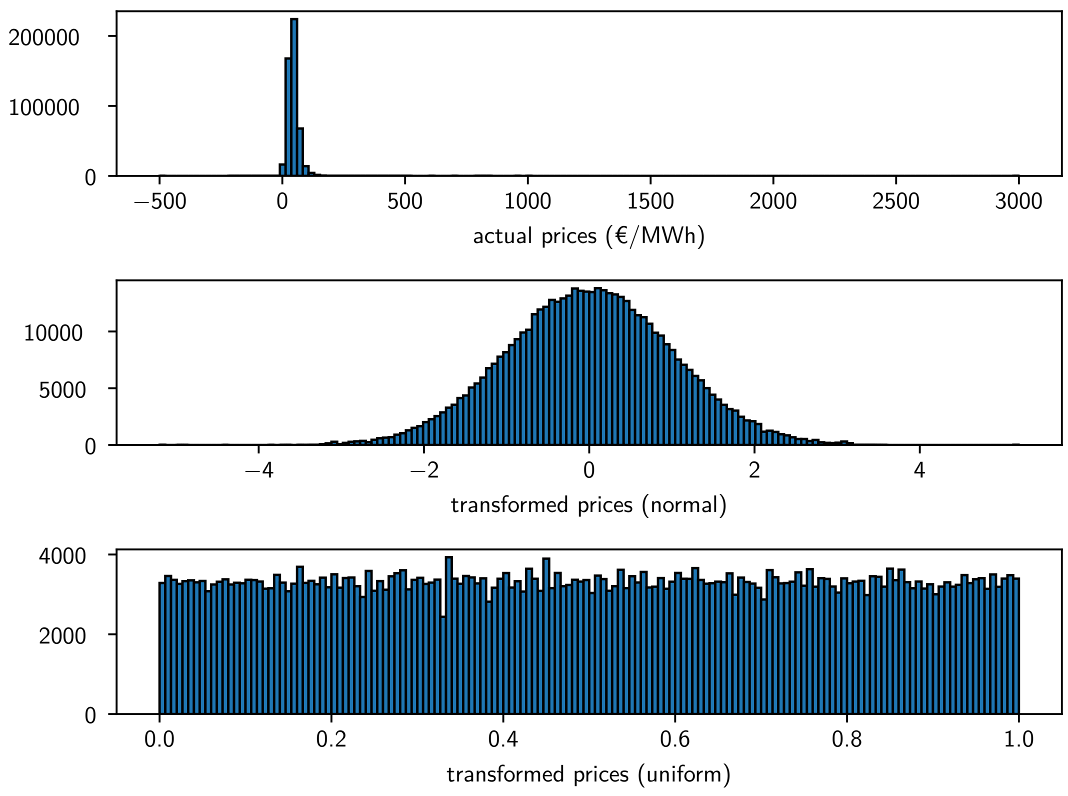


Fig. 3.4. Quantile transformation of the target values.

3.3.3. Out-of-sample testing

After the optimal hyperparameters for each model were determined, the models were then used to predict day-ahead prices in the hold-out test set. This backtesting was done in a walk-forward fashion. Every day D, the models forecasted $D + 1$ prices using only information available before noon on day D. Each week, the parameters of each model (*but not the hyperparameters*) are recalibrated, using all data available from the start of the dataset until and including the week that just became available (expanding window). The sMAPEs (Section 2.5.1.4) between the predicted price of each model and the actual price were computed, and one-sided Diebold-Mariano tests (Section 2.6.3) between the relevant forecasts were conducted.

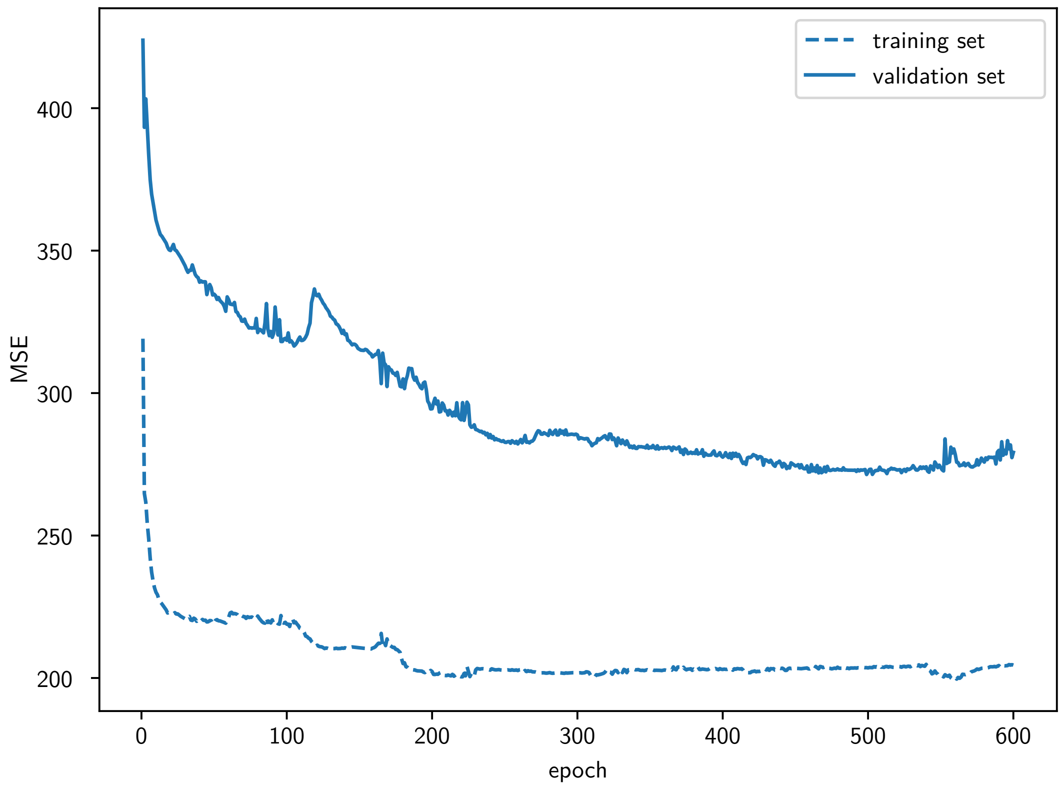


Fig. 3.5. Optimization learning curves for the FFANN model.

Chapter 4

Results

4.1. Data preprocessing

Table 4.1 displays the optimal data preprocessing techniques for each model, defined as the combination of techniques that performed best on the validation set.

Table 4.1. Optimal data preprocessing techniques for each model.

model	encoding	missing data imputation	input standardization	target transformation
Linear Regression	one-hot	iterative	none	normal
Extra trees	ordinal	simple	none	uniform
FFANN	one-hot	simple	none	none
Random Forest	one-hot	iterative	none	uniform
Gradient Boosting	one-hot	iterative	none	normal
kNN	ordinal	iterative	Min-Max	uniform

4.2. Hyperparameters search

Table 4.2 lists the hyperparameters for each model (second column) as well as the hyperparameters values that produced the predictions with the lowest sMAPE on the validation set (third column). Figure 4.1 shows the effect of influential hyperparameters on the validation sMAPE for the kNN model, while Figure 4.2 shows the effect of influential hyperparameters on the validation sMAPE for the tree-based models (extremely randomized trees, random forest, and gradient boosting).

4.3. Out-of-sample model performance

Table 4.3 displays the out-of-sample sMAPE metric for each model, in increasing order of performance (lower values are better). The *Ensemble* model predictions is the arithmetic mean (Section 2.3.4.3) of the kNN, Gradient Boosting and Random Forest predictions. The

Table 4.2. Notable hyperparameters of learning algorithms with their optimal values. See Section 2.5.2 for a clarification regarding treating surrogate loss function choice as an hyperparameter.

model	hyperparameters	optimal values
linear regression	N.A.	
extra trees	• number of trees B • surrogate loss function L • stopping criterion for training individual trees • fraction of features to include in each bootstrapped training sets	250 MSE maximum depth of trees = 10 1.0
FFANN	• number of hidden layers • number of units in each hidden layer • activation function of each unit (sigmoid, ReLU, etc) • surrogate loss function (MSE, MAE, Huber, etc) • minibatch size for SGD • learning rate γ for SGD • whether to use batch normalization or not • dropout probability p • stopping criterion to end training	2 $n_1 = 100$, $n_2 = 50$ ReLU MSE 64 0.001 No 0.0 stop if best validation error has not improved for 5 consecutive epochs
random forests	• number of trees B • surrogate loss function L • fraction of features to include in each bootstrapped training sets • stopping criterion for training individual trees	250 MSE 0.5 nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples
gradient boosting	• number of weak learners T • learning rate γ_t • surrogate loss function L • stopping criterion for training weak learners • fraction of features to include in each bootstrapped training sets	100 0.1 MSE maximum depth of trees = 5 0.25
kNN	• number of neighbors k • distance function of the neighbors d • weighting function of the neighbors w	10 Euclidean distance proportional to distance

Naive model serves as benchmark; its predicted price for hour H on day D for country C is the realized price for hour H for country C on day $D - 1$.

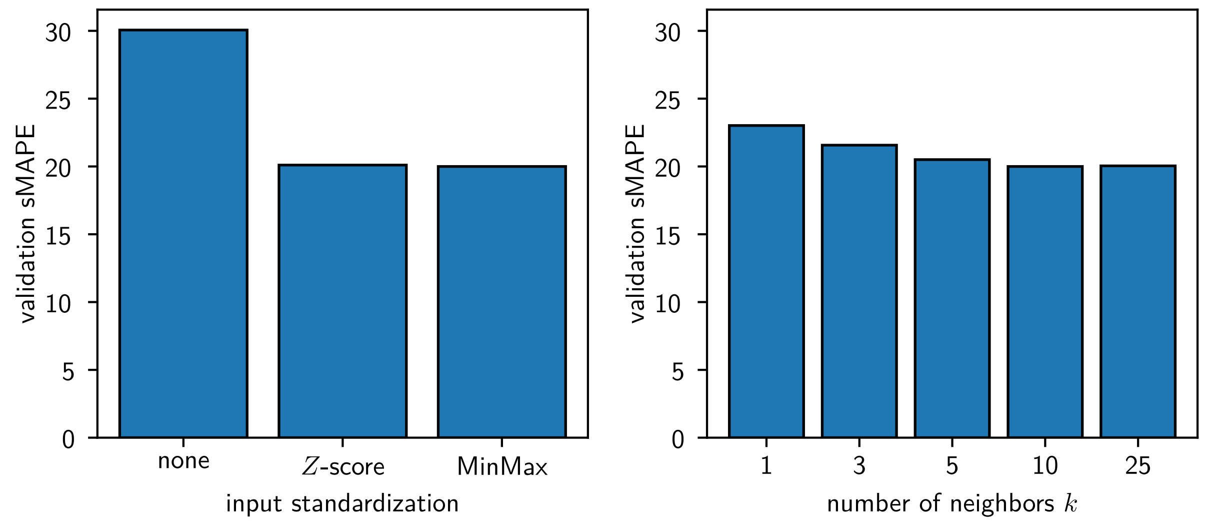


Fig. 4.1. Effect of influential hyperparameters for kNN. All other hyperparameters held equal at their optimal value.

Table 4.3. Out-of-sample sMAPE performance metrics. Lower values are better.

model	sMAPE
Extra trees	23.37
FFANN	22.86
Naive	22.42
Linear Regression	18.47
kNN	18.06
Random Forest	14.46
Ensemble	13.68
Gradient Boosting	13.44

Table 4.4 illustrates the statistical significance of the results in Table 4.3 using one-sided Diebold-Mariano tests. Note that 28 tests are performed; therefore the null hypothesis is expected to be rejected with confidence $p = 0.05$ once by random chance alone.

4.4. Feature importances

Figure 4.3 displays the normalized importance of each input feature as inferred from the gradient boosting model, as described in Section 2.3.3.3. Each importance is normalized so that they sum to 1.

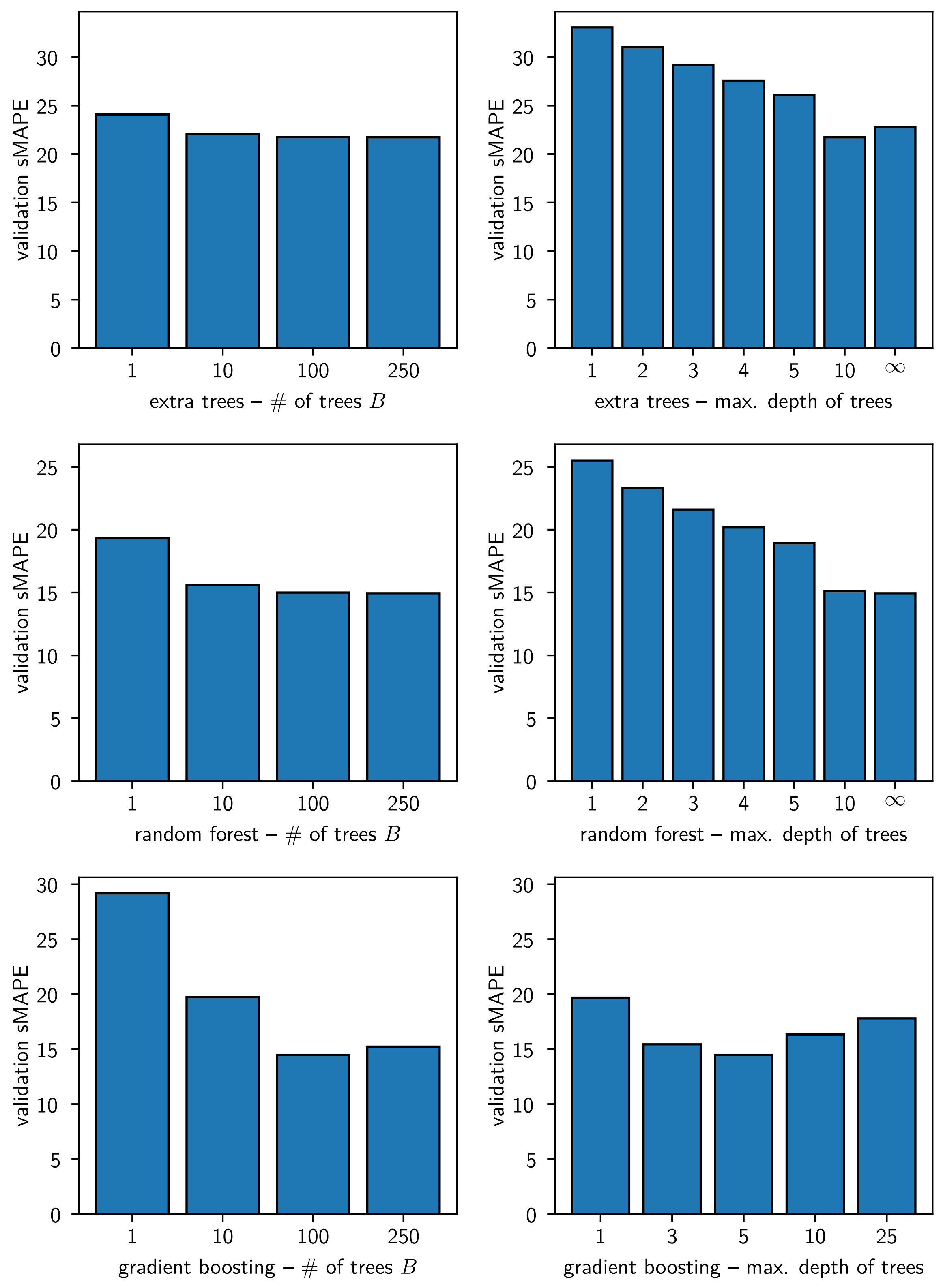


Fig. 4.2. Effect of influential hyperparameters for tree-based models. All other hyperparameters held equal at their optimal value.

Table 4.4. Statistical significance of performance metrics. The Diebold-Mariano one-sided test null hypothesis (H_0) is that forecast A's accuracy is equal or worst to forecast B's. Red crosses (✗) signifies H_0 is not rejected and green ticks (✓) signifies H_0 is rejected with confidence $p = 0.01$.

↓ A \ B →	Extra trees	FFANN	Naive	LR	kNN	RF	Ensemble	GBM
Extra trees	-							
FFANN	✗	-						
Naive	✗	✗	-					
LR	✓	✓	✓	-				
kNN	✓	✓	✓	✗	-			
RF	✓	✓	✓	✓	✓	-		
Ensemble	✓	✓	✓	✓	✓	✓	-	
GBM	✓	✓	✓	✓	✓	✓	✗	-

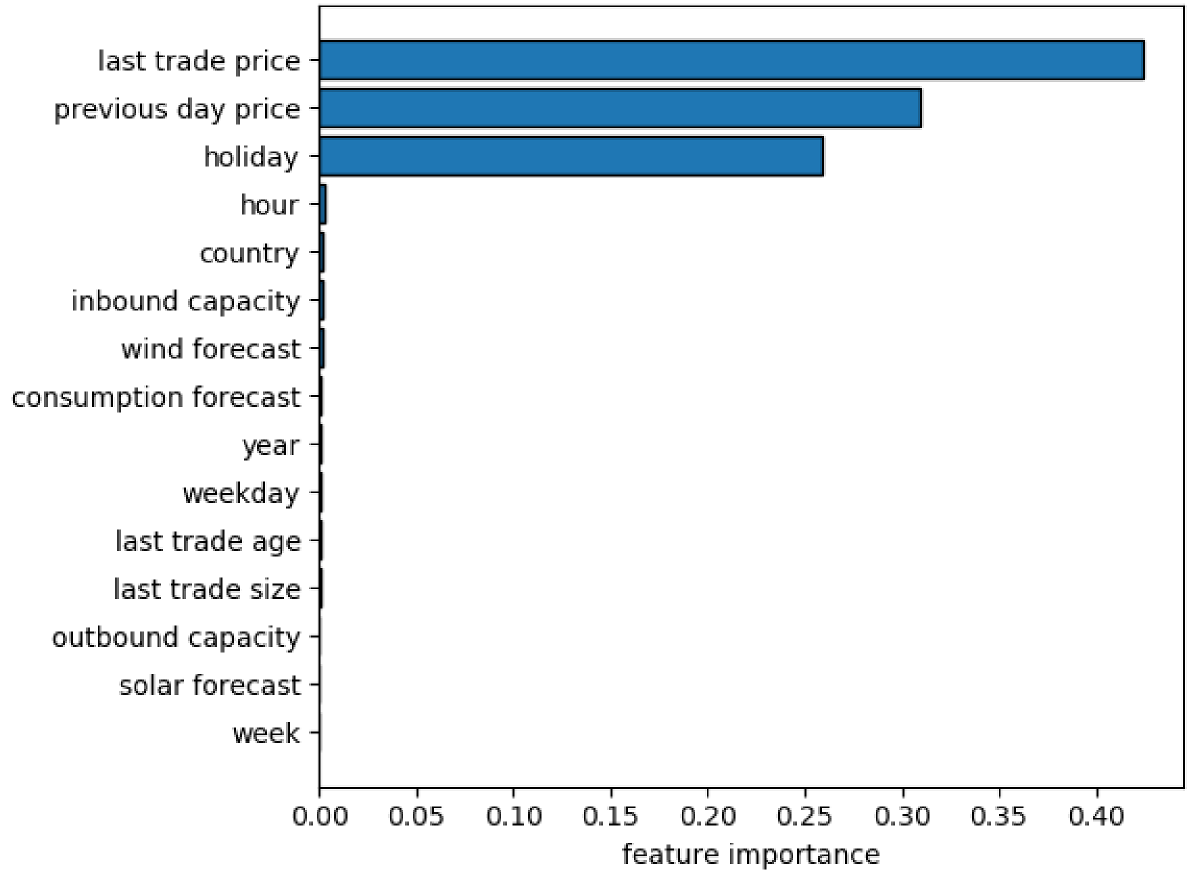


Fig. 4.3. Feature importances inferred from the gradient boosting model.

Chapter 5

Discussion

The general takeaway of the obtained results is: forecasting day-ahead electricity prices from the selected input features is a remarkably low signal-to-noise ratio (SNR) problem. While the selected features do influence the day-ahead prices, a significant amount of the variance in prices arises from factors not captured by the features used. This informational insufficiency can be observed in Figure 4.3; most of the features are barely used to produce the price forecast. Other manifestations of this low signal-to-noise ratio are highlighted below.

5.1. Data preprocessing

The choice of categorical variable encoding and missing data imputation techniques did not impact the accuracy of the forecasts materially. The models achieved similar performance irrespective of the encoding or imputation technique used. By Occam's razor, it is therefore wise to use the simplest encoding and imputation technique out of the trialed ones, namely one-hot encoding and mean imputation.

All models except kNN did not benefit from input standardization. In the case of kNN (Figure 4.1, left), the criticality of standardizing input features is intuitive; since the nearest neighbors are determined using Euclidian distance, features with different scale would skew the results significantly. For other models, the absence of input standardization did not negatively impact performance.

Target transformation benefitted all models except FFANN. The benefits of target transformation are most likely due to the concentration of electricity prices in a small range with some large outliers; spreading the distribution of prices allows the models to learn separating hyperplanes more easily.

5.2. Hyperparameters search

The most salient outcome of the hyperparameters search is that the combination of hyperparameters that yielded the best forecast accuracy for the FFANN model is the simplest one (fewer hidden nodes per layer, no use of batch normalization nor dropout). This outcome is most likely due to the training dataset being too small to train more complex models. Indeed, the dataset used (159,036 data points) is orders of magnitude smaller than the 10,000,000 data points needed to train deep models rule of thumb advocated in the literature [**42**].

Another interesting observation is that increasing the number of underlying estimators does not harm the accuracy of bagging models (random forests and extremely randomized trees) but does reduce the accuracy of the gradient boosting model (Figure 4.2, left). This observation is in accordance with the theory. Since bagging weights training samples equally, adding more estimators does not cause overfitting. However, since boosting weights erroneous training samples more and more heavily as the number of estimators grows larger, adding more estimators causes overfitting.

5.3. Out-of-sample model performance

Gradient boosting significantly outperformed every other model besides the ensemble method. Gradient boosting machines are a recurring theme in forecasting competitions' leaderboards, so this outcome did not come as a surprise.

One salient finding is the relative outperformance of the ensemble method. The superior accuracy of ensembles of models is analogous to the wisdom of the crowds[1]. The superiority of ensembles also relates to Jensen's inequality [**52**]. Since common performance metrics m are convex:

$$m\left(\sum_{i=1}^{n} w_i e_i\right) \leq \sum_{i=1}^{n} w_i m(e_i) \tag{5.3.1}$$

where e_i is the error of the i^{th} forecast and w_i is the weight assigned to the i^{th} forecast ($0 \leq w_i \leq 1$). The error of the average of n predictions (left hand side of Equation 5.3.1) will always be lower than the average of these errors (right hand side of Equation 5.3.1). Furthermore, the error of the ensemble cannot be larger than the largest error of the n forecasts.

Another notable outcome is how accurate the forecasts produced by the kNN model were despite kNN relative simplicity and interpretability. It is not clear why this is the case. One theory is that the data generating process changed significantly throughout the dataset (in order words, the distribution producing the samples evolved, yielding non-IID samples) and that the entirely non-parametric nature of kNN is more robust to such regime changes than some of the other models. Given the evolution of the European electricity markets over the period of the dataset, with a larger contribution from irregular renewable sources, this hypothesis is plausible.

[1]At the 1906 West of England Fat Stock and Poultry Exhibition in Plymouth, 787 participants entered a contest to predict the weight of an ox for a prize. The entries ranged over 219 lbs, with an average of 1197 lbs. The actual weight of the ox was 1198 pounds; therefore the average of the predictions had an error below 0.1%! [**36**]

The performance of neural networks was unexpectedly poor despite their growing representation in the forecasting literature. This underperformance is plausibly due to their voracity for data; the dataset in this work being not large enough for the FFANN model to learn the relevant relationships between the variables well. The optimization learning curves in Figure 3.5 corroborate this hypothesis. The large gap between the training set loss and the validation set is a symptom of an unrepresentative training set, pointing to the low signal-to-noise ratio in the data.

Chapter 6

Conclusion

Electricity Price Forecasting is a notably difficult task for two reasons. Firstly, it is a low signal-to-noise ratio task. A small number of input features cannot fully capture the large number of factors driving electricity prices. Secondly, the migration toward cleaner energy sources results in a continually evolving electricity production landscape. Therefore, relationships from the past may not hold in the future, disabling forecasting models.

Nonetheless, this research demonstrates that standard statistical learning tools can produce satisfactory results. Furthermore, ensemble methods combining the prediction of multiple models helps in producing more accurate forecasts.

There are three promising avenues for further research. Firstly, the incorporation of additional, pertinent input features would improve the signal-to-noise ratio problem. A methodology to systematically test features for predictive ability while protecting against spurious correlation would be especially beneficial. Forays into this area of research with contributions from the field of information theory are underway [**37, 13**]. Secondly, there exists research on more sophisticated methods to combine forecasts than the simple arithmetic mean, starting with Bates and Granger seminal paper on the topic [**11**]. The application of these methods would likely yield improvements in forecast accuracy. Thirdly, a reformulation of the objective could make the problem both more tractable and relevant. Instead of predicting the 24 hourly prices of the day ahead, predict if the average day-ahead hourly price will be under or over the price currently traded on the cash-settled futures market (EEX). The task changes from one of regression to one of binary classification, which is possibly easier to model. Moreover, this formulation of the task allows for the testing of the efficient-market hypothesis (EMH) [**32**].

References

[1] European energy exchange. `https://www.eex.com/`, Dec 2019.

[2] European network of transmission system operators for electricity. `https://www.entsoe.eu/`, Dec 2019.

[3] Ihs markit. `https://ihsmarkit.com/index.html`, Dec 2019.

[4] Leave one out. `https://contrib.scikit-learn.org/categorical-encoding/leaveoneout.html`, Dec 2019.

[5] Réseau de transport de l'Électricité. `https://www.services-rte.com/en/home.html`, Dec 2019.

[6] Trading on epex spot 2019-2020. `https://www.epexspot.com/sites/default/files/2019-02/2019-01-17_TradingBrochure_V2.pdf`, 2019.

[7] BCM Energy. `https://www.bcmenergy.fr`, Jan 2020.

[8] Mitacs accelerate international. `https://www.mitacs.ca/en/programs/accelerate/mitacs-accelerate-international`, Jan 2020.

[9] Sanjeev Kumar Aggarwal, L.M. Saini, and Ashwani Kumar. Short term price forecasting in deregulated electricity markets: A review of statistical models and key issues. *International Journal of Energy Sector Management*, 3(4):333–358, 2009.

[10] Phatchakorn Areekul, Tomonobu Senjyu, Hirofumi Toyama, and Atsushi Yona. Next day price forecasting for electricity market. In *2011 International Conference on Advanced Power System Automation and Protection*, volume 2, pages 1390–1395, Oct 2011.

[11] J. M. Bates and Clive W. J. Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.

[12] Timothy Mark Beasley, Stephen Erickson, and David B. Allison. Rank-based inverse normal transformations are increasingly used, but are they merited? *Behavior Genetics*, 39:580–595, 2009.

[13] Mohamed Bennasar, Yulia Hicks, and Rossitza Setchi. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532, 2015.

[14] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. From predictive methods to missing data imputation: An optimization approach. *Journal of Machine Learning Research*, 18(196):1–39, 2018.

[15] Aleksey Bilogur. Missingno: a missing data visualization suite. *The Journal of Open Source Software*, 3:547, 02 2018.

[16] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.*, 7(1):108–116, Jan 1995.

[17] George Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities.* London: Walton and Maberly, 1854.

[18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[19] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[20] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[21] Leo Breiman, Jerome H. Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[22] S. F. Buck. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society. Series B (Methodological)*, 22(2):302–306, 1960.

[23] Rene Carmona, Michael Coulon, and Daniel Schwarz. Electricity price modeling and asset valuation: A multi-fuel structural approach. *Mathematics and Financial Economics*, 7:167–202, May 2012.

[24] Chris Chatfield. Apples, oranges and mean square error. *International Journal of Forecasting*, 4(4):515–518, 1988.

[25] Xia Chen, Z.Y. Dong, Ke Meng, Yan Xu, Kit Wong, and H.W. Ngan. Electricity price forecasting with extreme learning machine and bootstrapping. *IEEE Transactions on Power Systems*, 27(4):2055–2062, Nov 2012.

[26] Robert T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989.

[27] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, Dec 1989.

[28] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition, 2018.

[29] Francis X. Diebold and Roberto S. Mariano. Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, Jul 1995.

[30] Nick Dingwall and Chris Potts. Are categorical variables getting lost in your random forests? `https://roamanalytics.com/2016/10/28/are-categorical-variables-getting-lost-in-your-random-forests/`, Oct 2016.

[31] Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, Apr 1976.

[32] Eugene Fama. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25:383–417, 1970.

[33] Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[34] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, Oct 2001.

[35] Pierre Gaillard, Yannig Goude, and Raphaël Nedellec. Additive models and robust aggregation for gefcom2014 probabilistic electric load and electricity price forecasting. *International Journal of Forecasting*, 32(3):1038–1050, 2016.

[36] Francis Galton. Vox populi – the wisdom of crowds. *Nature*, 75(1949):450–451, 1907.

[37] Yun Gao, Ioannis Kontoyiannis, and Elie Bienenstock. Estimating the entropy of binary time series: Methodology, some theory and a simulation study. *Entropy*, 10:71–99, 2008.

[38] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63:3–42, 2006.

[39] Corrado Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.* Tipogr. di P. Cuppini, Bologna, 1912.

[40] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, Apr 2011. PMLR.

[41] Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14(5):403–420, April 1970.

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[43] Richard H. R. Hahnloser and Sebastian H. Seung. Permitted and forbidden sets in symmetric threshold-linear networks. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 199–205, Cambridge, MA, USA, 2000. MIT Press.

[44] Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. https://arxiv.org/abs/1710.11278, 2017.

[45] David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291, Jun 1997.

[46] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[47] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[48] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–282, Washington, DC, USA, 1995. IEEE Computer Society.

[49] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32, 03 2016.

[50] Tao Hong, Jingrui Xie, and Jonathan Black. Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1389–1399, 2019.

[51] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

[52] Johan L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

[53] Michael Kearns. Thoughts on hypothesis boosting. https://www.cis.upenn.edu/ mkearns/papers/boostnote.pdf, Dec 1988.

[54] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, Sep 1952.

[55] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[56] Tarjei Kristiansen. Forecasting nord pool day-ahead prices with an autoregressive model. *Energy Policy*, 49(C):328–332, 2012.

[57] Ahmed A. Ladjici, Ahmed Tiguercha, and Mohamed Boudour. Nash equilibrium in a two-settlement electricity market using competitive coevolutionary algorithms. *International Journal of Electrical Power & Energy Systems*, 57:148–155, 2014.

[58] Jesus Lago, Fjo De Ridder, and Bart De Schutter. Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, 221:386–405, Jul 2018.

[59] Jesus Lago, Fjo De Ridder, Peter Vrancx, and Bart De Schutter. Forecasting day-ahead electricity prices in europe: The importance of considering market integration. *Applied Energy*, 211:890–903, 2018.

[60] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[61] Gilles Louppe. *Understanding Random Forests: from Theory to Practice*. PhD thesis, University of Liège, 2014.

[62] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.

[63] Katarzyna Maciejowska, Jakub Nowotarski, and Rafał Weron. Probabilistic forecasting of electricity spot prices using Factor Quantile Regression Averaging. *International Journal of Forecasting*, 32(3):957–965, 2016.

[64] Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993.

[65] Spyros Makridakis. M5 announcement. `https://twitter.com/spyrosmakrid/status/1209153609432211457`, Dec 2019.

[66] Spyros Makridakis, Allan Andersen, Robert F. Carbone, Robert Fildes, Michèle Hibon, Rudolf Lewandowski, Jonathan Newton, Emanuel Parzen, and Robert L. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982.

[67] Spyros Makridakis, Chris Chatfield, Michèle Hibon, Michael Lawrence, Terence Mills, Keith Ord, and LeRoy F. Simmons. The m2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 9(1):5–22, 1993.

[68] Spyros Makridakis and Michèle Hibon. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. The M3- Competition.

[69] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[70] Grzegorz Marcjasz, Bartosz Uniejewski, and Rafał Weron. On the importance of the long-term seasonal component in day-ahead electricity price forecasting with narx neural networks. *International Journal of Forecasting*, 35(4):1520–1532, 2019.

[71] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 512–518, Cambridge, MA, USA, 1999. MIT Press.

[72] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor. Newsl.*, 3(1):27–32, Jul 2001.

[73] Adam Misiorek, Stefan Trueck, and Rafal Weron. Point and Interval Forecasting of Spot Electricity Prices: Linear vs. Non-Linear Time Series Models. *Studies in Nonlinear Dynamics & Econometrics*, 10(3):1–36, Sep 2006.

[74] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[75] Jakub Nowotarski, Eran Raviv, Stefan Trück, and Rafał Weron. An empirical comparison of alternative schemes for combining electricity spot price forecasts. *Energy Economics*, 46(C):395–412, 2014.

[76] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *J. Artif. Int. Res.*, 11(1):169–198, Jul 1999.

[77] Michael G. Pollitt. The european single market in electricity: An economic assessment. *Review of Industrial Organization*, 55(1):63–87, Aug 2019.

[78] Sebastian Raschka. About feature scaling and normalization – and the effect of standardization for machine learning algorithms. `https://sebastianraschka.com/Articles/2014_about_feature_scaling.html`, Jul 2014.

[79] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[80] Lior Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1–2):1–39, Feb 2010.

[81] Lior Rokach and Oded Z. Maimon. Top-down induction of decision trees classifiers - a survey. *Trans. Sys. Man Cyber Part C*, 35(4):476–487, Nov 2005.

[82] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[83] Warren S. Sarle. comp.ai.neural-nets faq, part 2 of 7: Learning. `ftp://ftp.sas.com/pub/neural/FAQ2.html`, Oct 2002.

[84] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, Jun 1990.

[85] Jonas Sjöberg and Lennart Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995.

[86] Uğur Soytaş and Ramazan Sarı. *Routledge Handbook of Energy Economics*. London: Routledge, 2019.

[87] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan 2014.

[88] Ilya Sutskever. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, 2013.

[89] Bartosz Uniejewski, Jakub Nowotarski, and Rafał Weron. Automated variable selection and shrinkage for day-ahead electricity price forecasting. *Energies*, 9:621, 08 2016.

[90] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, USSR, 1974.

[91] Rafał Weron. *Modeling and Forecasting Electricity Prices*, chapter 4, pages 101–155. John Wiley & Sons, Inc., 2013.

[92] Rafał Weron. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, 30(4):1030–1081, 2014.