POLITECNICO DI TORINO

Master degree in Energy and Nuclear Engineering

Master Degree Thesis

Machine learning strategies assessment for energy commodities forecasting

Politecnico di Torino

Supervisor
Prof. Maurizio Repetto

Co-supervisor
Ivan Mariuzzo

Candidate
Francesco Superchi

Academic Year 2020/2021

Abstract

In recent years, the liberalization of the electricity market and the increasing penetration of distributed energy generation based on renewable sources have contributed to expand the uncertainty in the energy commodities trend. Effective methods able to make accurate predictions about electricity price and demand, as well as forecasting of the amount of power produced by photovoltaic systems and wind generators, would have a beneficial impact on the system management and would maximize profits for the actors involved in the energy exchange market. This work aims to test and compare different Machine Learning methods (Linear Regression, Support Vector Regression and Neural Networks) to obtain predictions of two energy commodities: zonal electricity price and PV power production. The algorithm that performed better in this sense (SVR) was then applied to a real life like case study to understand what kind of results can be obtained.

Data that have been fed to algorithms are referred to Northern Italy. The total load curve comes from `Terna`, the Italian transmission system operator, while the zonal price (PZ) history has been provided by `GME`, the exchange for electricity and natural gas spot trading in Italy. The RES power production forecast was related to a PV field located in *Fossano, Italy*, whose production history was provided by `EGEA SPA`, owner of the system.

Contents

List of Figures

Code Listings

Introduction

Since the down of time, humans have been fascinated by the ability to predict the future. The rise of Artificial Intelligence (AI) have opened new doors in the the forecasting field. Machine Learning algorithms are able to produce effective models that can be applied to the prediction of many kinds of time series. The machine must only analyse a sequence of data to provide the user a tool that can be used to predict the the future evolution of it. In the field of energy, there are several quantities that sector operators have been trying to predict for many years. One of those is power produced by inconstant sources as photovoltaic generation and wind. The increasing penetration of distributed energy sources has led to the need of forecasting tools able to predict the amount of electrical power that such systems will inject into the grid. Another field that has been characterized by the necessity of forecasting tools is the electricity market. The liberalization of this market have seen the introduction of buyers and sellers joined by free exchange market dynamics. Both of those players would benefit from accurate forecasting tools that allow them to predict the evolution of the electricity price. Several other fields would benefit from accurate forecasting models. Machine Learning seems a viable candidate for this kind of operation. From this premise, this master thesis aims to test this kind of algorithms on the above mentioned applications: PV power forecasting and electricity price forecasting. It was considered the current situation in Italy, which features a liberalized electricity market, and a PV field located in the northern part of the peninsula was takes as example for the renewable energies related forecasting.

A first section is dedicated to present to the readers how the electricity market works now in Italy, which dynamic rules it, why forecasting would be beneficial and for whom. It is also explained what would be the benefits in forecasting of renewable generation and which methods can be used. A second section aims to describe how forecasting can be done and validated and machine learning is introduced. In a third section the three methods employed are mathematically described and is explained how their parameters can be optimized for this kind of operation. Then two sections are dedicated to the description and a review of the results coming from the application of the prediction methods to the PV power generation and the electricity price. At the end of both sections, the model that performed best is highlighted. In a last section the models that have performed best have been applied to case studies as the prediction of the quantities of interest along a month. Years 2019, 2020 and 2021 have been selected to show how the performances of the predictor can vary due to economic fluctuations that affects the electricity market.

Part I

Background

Chapter 1

Electricity market

Nowadays, the electric power market has become a competitive market due to the liberalization carried out in the last years: Buyers, producers, investors and traders can actively participate to it, and the price of electricity is determined on the basis of this buying and selling scheme [1]. In this kind of system, what rules are the dynamics of the free market. In this work is considered a the current market situation in Italy, for this reason here is described how the electricity market in this country is structured. In Italy, every day energy providers attend the 1-day-ahead market session, the so called *Mercato del Giorno Pima* [5]. The submission from market participants takes place between the ninth day before the day of physical delivery (that opens at 8 a.m) and the day before the day of delivery (closes at 12 p.m) [6]. The trade of electricity among producers and consumers is possible thanks to a *pool* and a framework. A *pool* is an e-commerce marketplace that makes the trade possible, a framework is what enables bilateral contracts. In the pool, power producers submit generation bids and corresponding bidding prices, consumers do the same with consumption bids. The market operator manages the overall system through auctions that determine the resulting market price and accepts production and consumption bids [3]. Energy is sold starting from the lower price offers until the overall demand is met without exceeding the physical limitation of the national electricity grid. Through this mechanism, the price of electricity is determined. Italy is divided in six geographical market zones interconnected by power lines of limited capacity [5], in each of those a zonal price (PZ) is determined. The average of those prices makes the national electricity price, the so called *Prezzo Unico Nazionale*, abbreviated in PUN. In this work it was considered the zonal price of the northern part of the country.

1.1 Electricity price and load forecasting

Electricity markets benefit from load and price forecast.

Load forecasting benefits

Demand forecasting plays an important role for electricity power suppliers because both the excess and shortage of energy production may lead to reduction of profit. In fact, important operating decisions are based on load forecast: power generation scheduling, demand supply management and maintenance planning. In addition, load forecast can be used to identify load peaks during the day and allows the operation of peak shaving. Reducing peaks in important to reduce the number of power plants that are switched on to operate only for a few minutes during the day, since those are the one with the highest cost and emissions [34].

Price forecasting benefits

However, what is crucial for energy market participants for bidding is price forecast, since an effective bidding strategy helps market participants in maximizing gains [1]. Both produces and costumers may take advantages from those tools: A producer uses the day-ahead forecast to establish a pool bidding technique to achieve its maximum benefit and costumer can use forecasts to develop a plan to maximize its utility usage of the electricity purchased. A *prosumer*, that is a consumer that has a self-production capability, can use the forecast to avoid high prices. Moreover, in electricity markets, when a power producer does not follow the scheduled bid, it will be penalized: during the time period in which there is a difference between the electric energy amount that was presented in the bid and the actual amount that is produced, it will receive lower retributions than what was established [2]. Those penalties are applied because the incorrect scheduling of the day-ahead market session causes imbalances to the network. Market zone can be negatively or positively imbalanced and according to the situation the energy network manager must perform an up-regulation or down-regulation of the system. In case of up-regulation, the missing energy is acquired from renewable producers which have positive imbalance (excess of energy production with respect to what was announced), and the remaining quantity is exchanged during a balancing market, called *Mercato di Bilanciamento*, where conventional producers offer to scale up their production. In case of down-regulation, some power plants are turned off. Producers are paid to scale up the production or have to pay for the quantity of energy that they failed to provide. The price depends on the balancing market outcome price, which can be different from the one previously established after the 1-day-ahead market [5]. Generally, prices of the balancing market are higher [6].

Price forecasting logic

There are several different parameters that may influence the price in addition to the demand. Some of those are fuel price, availability of renewable generation sources, weather and more [1]. The basic idea to predict the future price of electricity is to use the past values of the price itself and other quantities such as load and temperature, together with forecast of those, to perform the prediction. In other words, use history and other estimated factors in the future to *fit* and *extrapolate* future prices [6]. Not only the accuracy but also stability plays a vital role in electricity price forecasting models in order to guarantee the reliability and economic performance of power grid [17].

1.1.1 Characteristics of the electricity price time series

In the perspective to predict the evolution of the electricity price, it is important to analyse and understand the time series that describes its evolution in time. In competitive electricity market, price time series are characterized by many complex features. Electricity price is [17]:

- Non-stationary;
- Non-linear;
- Highly volatile;
- Seasonal (daily and weekly periodicity);
- Affected by calendar (weekends and holidays);
- Characterised by many unusual prices in periods of high demands.

Those characteristics makes the forecasting of electricity price and load very difficult and complex [3]. Non-stationary and non-linearity are the most important characteristics of the price signal in the electricity market, which reduces the accuracy and stability of many forecasting methods [17].

1.1.2 Inputs for price forecasting

Various factors have an impact on electricity price and load such as, weather, population growth, fuel price, RES production and different economic attributes. In literature there are many studies in which it is investigated the effectiveness of including several energy related time series in the model. In the context of hourly market price prediction, the most influential external variable is considered to be the load [14]. Load fluctuations impact the price and price fluctuations impact the load. For this reason, historical load records are good candidates as inputs [13]. In [6] and [13] it was performed a forecast using a method that takes as input the daily fossil fuel price, in addition to the electricity price and load historical trend. Since the fuel costs are main part of total generation cost, the change in fuel prices may affect the market prices. In [13] it was also considered wind power production and solar production.

Lagged price time series

In the majority of forecasting problems, historical values of the parameter under study are always input candidates [14]. A strong correlation exists between the price at a given time t and past records of it [6] [13].

The hourly electricity market price has cyclic characteristics [6]:

- Daily cycle: the basic cycle is 24h;
- Weekly cycle: in a week span, each day's patter is different. The difference is substantial between a weekday and weekend;
- Yearly cycle: electricity price reflects the load variation, that is present in a year for different seasonal climate. A model that takes as input a lagged price time series is able to capture this cyclic trend and perform a more precise forecast.

In [14] the *Pearson correlation coefficient* was computed to measure the degree of dependence between current values and values up to 1 week before. Results have shown that current hour price shows high correlation with those of hour $h - 24$ and $h - 168$, a fact that indicates daily and weekly periodicity. The exact meaning of this metric is described in following sections. Lagged data frames are introduced in order to exploit the correlation between the price that must be predicted and its past records.

Chapter 2

Renewable energy forecast

The electricity production by renewable energy sources (RES) is increasing year after year and all around the globe thanks to government policies and technical progress [10]. This increasing penetration, that is going to continue in the following decades, introduces several challenges in the power system operation. This is due to the intrinsic intermittent and uncertain power generation of sources like photovoltaic and wind power generation, because they are strongly influenced by constantly changing meteorological conditions. The ability to accurately forecast the energy production from renewable sources must be developed to obtain short and midterm forecasts. PV and wind power forecasting, as an estimation of the expected power production, is crucial to help the grid operators to better manage the electric balance between power demand and supply, and to improve the penetration of such technologies.

Accurate forecasts of RES production may provide several benefits [7]:

- A first benefit comes in terms of **dispatchability and reliability**: power system is managed mainly on day-ahead dispatches [33] and meaningful day-ahead plans can be performed only if accurate predictions of power generation and load consumption forecasts are available [34].

- A second benefit comes in terms of **efficiency**: in many markets there are penalties for power generators that fail to accurately predict their power generation because this cause frequency and voltage fluctuations in the transmission system [35]. For this reason, several producers underestimate their day-ahead power generation forecast to avoid penalties, leading to an inevitable inefficient conservative behaviour. If the prediction is accurate, generators are able to produce the optimal amount of power, without such restrictions.

- A third benefit regards **monitoring and safety**: the difference between the predicted and generated power allows to evaluate the degradation of the efficiency of the plant due to ageing of components [36]. This is useful to early detection of incipient faults.

In this work, among the various RES technologies it is addressed the photovoltaic power generation forecasting. As it is with the price of electricity, such generation can be predicted using related external variables. Power generation from photovoltaic panels mostly depends on meteorological variables, mainly from solar radiance. Temperature, humidity and cloud amount influence the power production too. For this reason, weather forecast can be used as input to forecasting methodologies of PV generation [7].

In the energy field is possible to perform very short-term, short-term, medium-term and long-term forecasting. The forecast up to 24-h ahead is used to develop the power dispatching plans, the optimization operations of grid-connected RES plants and control of energy storage devices [10]. In this work it is considered this kind of prediction.

2.1 Models for RES forecasting

Energy production can be predicted using two kinds of models: physical and statistical.

2.1.1 Physical models

Physical models are based on a set of mathematical equations that try to describe how the photovoltaic system is able to convert the incoming solar radiation into electrical power [37][38]. Complexity of those models can vary, since the PV power production depends on several parameters: solar radiation, cell temperature, shadows, load resistance and more [39]. This method presents some disadvantages as it must be designed specifically for the particular plant and location and often the information provided by the manufacturer are not sufficient of only related to the nominal conditions [10].

2.1.2 Statistical models

Statistical models are based on time series. Machine learning methods are able to recognize patterns in data using training data sets [10]. Historical data about weather conditions and power production are required to train an algorithm that can be later used to perform the prediction. Those are the kind of models have been employed to perform the PV power prediction presented in this work.

Combinations of two or more of the previously described methods can be considered a hybrid model.

Part II

Forecasting methodologies

Chapter 3

Forecasting methods

According to [9], the forecasting process can be divided into a set of steps depicted in figure 3.1

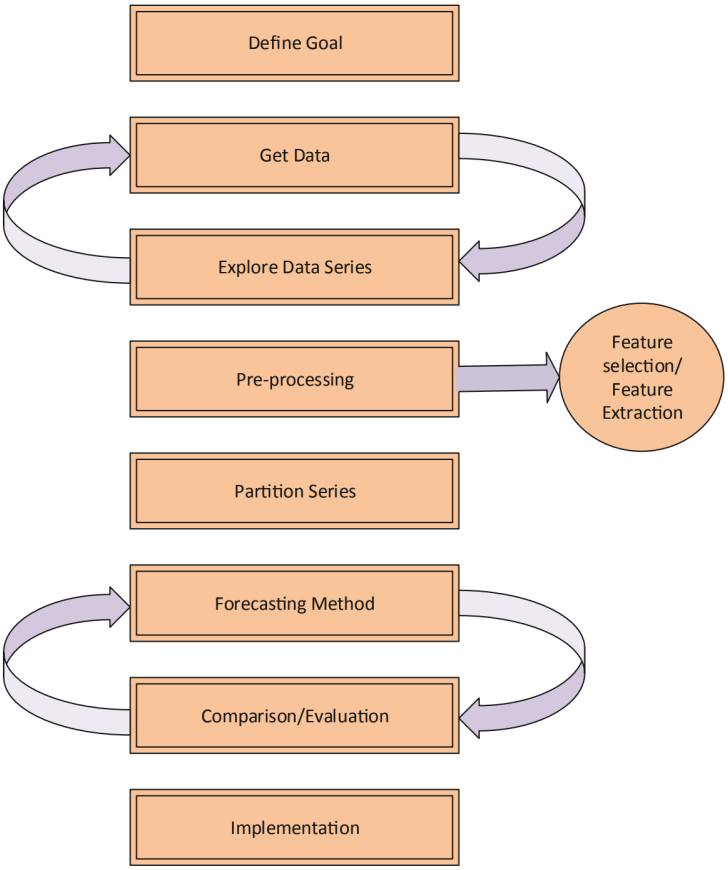


Figure 3.1: Forecasting process steps [9].

3.1 Forecasting methodology applied in this work

As previously assessed in the preceding sections, the goal of the predictions performed in this work is to forecast PV power production, as an example of renewable energy physical quantity, and electricity price, a central energy market quantity. The ultimate goal is to perform a day by day forecast of both those quantities using optimized algorithms. Required data have been harvested from proper databases ad providers. Starting from data sets containing both historical trends of the quantities that must be predicted and related variables, different algorithms have been applied in order to perform the forecasting itself. A data-set containing a historical trend is a time series and must be analysed managed in a proper manner. For this reason, a pre-processing was applied each time in order to feed the forecasting algorithm with data in appropriate shapes. The most frequent operations to be applied to original data sets are the conversion in the desired unit of measure, removal of null data, and conversion into a standardized time scale. If data pre-processing is not performed properly the prediction will probably be inefficient and incorrect [40]. After the data acquiring and pre-processing phases, the algorithms that have been employed here are:

- Linear Regression;
- Support Vector Regression;
- Neural Networks.

Results coming from different approaches has been compared using suitable parameters as the *Normalised mean absolute error* (NMAE) and *Mean absolute percentage error* (MAPE), coming from both a *standard validation* process and a *cross-validation* process. At the end of the process, the algorithm that performed best has been saved and applied to a case study for a simulation of the actual forecasting process. In the following paragraphs, a summary of the theoretical background behind those concepts has been reported.

3.2 Time series

A time series is composed by a sequence of values observed during a certain time span and chronologically ordered. Each observation x_i is recorded at a specific time t. Even if time is a continuous variable, samples are reordered at certain intervals in practice. A time series is defined as *"discrete-time time series"* when the set of times at which observations are made is a discrete set, while it is be described as *"continuous-time time series"* if observations are recorded continuously over a defined time interval [25]. A time series is better manageable and understandable if the intervals between data are constant. A graphical representation of a time series on a time plot is reported in figure 3.2, the x-axis identifies the time t and the y-axis identifies the punctual historical values y_t [1]. In this work several time series have been considered, both for predictions target quantities (PV power and electricity price) and related quantities (e.g; temperature, radiation, load).

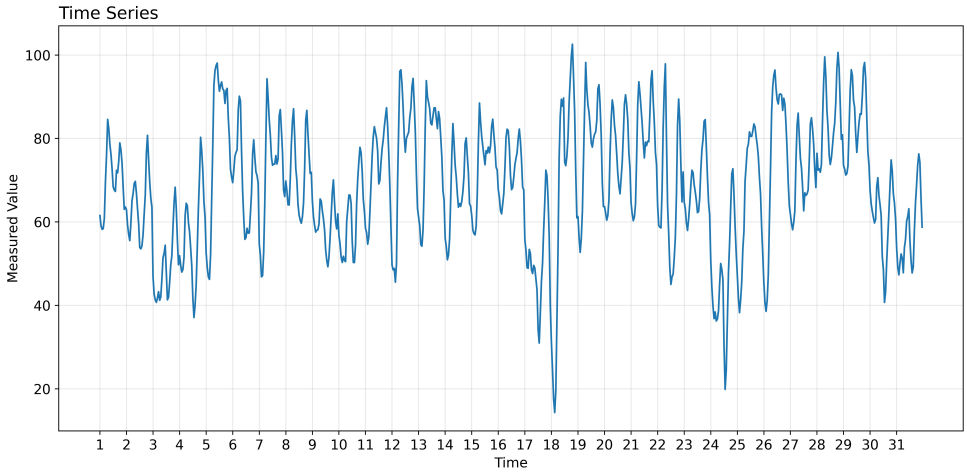


Figure 3.2: Graphical representation of a time series.

A time series can be decomposed in three main components [25] [26]:

- **Trend**: general behaviour of the variable during the observation time period. This can be seen as the long-term movement of the data set without considering irregular components, like abnormal peaks, or seasonality. The time series can follow peculiar trends such as linear, parabolic, exponential and other.
- **Seasonality**: periodical fluctuations of the variable. Its effect is stable in time, magnitude and direction. Several sources can produce seasonality on the time series like weather and calendar effect.
- **Residuals**: what remains after the detection and removal of trend and seasonality. When residuals are enough to hide the trend and seasonality of the time series, they are called *outliers*. Many sources can produce irregularities in the time series and they presence makes the prediction a very challenging task.

A visual representation of a time series decomposition is visible in figure 3.3

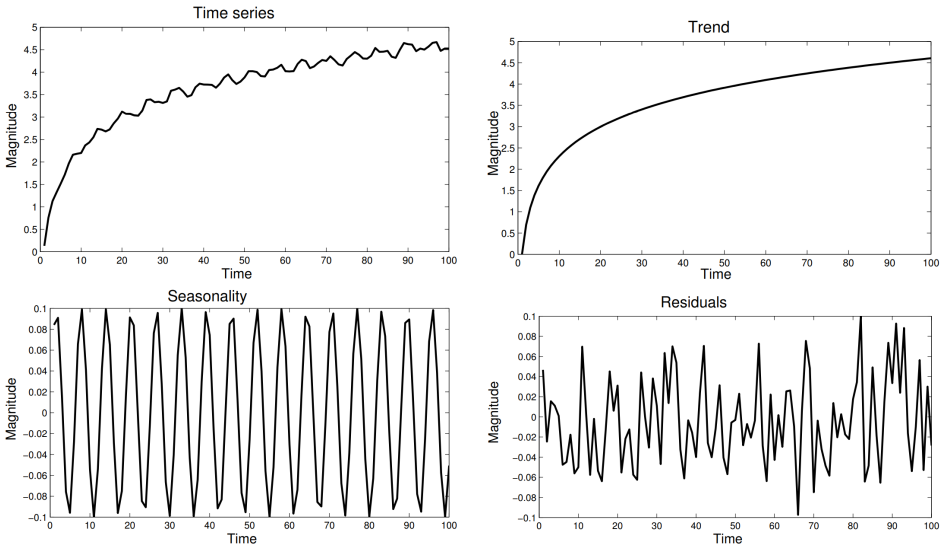


Figure 3.3: Graphical representation of a time series decomposition [1].

3.2.1 Time series forecasting

Given historical data of a given time series y_t to a certain time instant t, forecasting process consists in estimating the value y^*_{t+1} that the time series will assume after a time *t+1*. This is the purpose of the Machine Learning algorithms that have been tested in this work. The goal of the forecasting process is to minimize the error between this prediction (y^*_{t+1}) and the actual value that will occur (y_{t+1}). The best prediction is reached when a function (called "loss" function) of $sum(y_{t+1} - y^*_{t+1})$ is minimized: smaller it is the error, higher it is the accuracy.

Energy price presents peculiarities such as non-constant mean and variance, high volatility and presence of *outliers*. These characteristics makes the forecasting process a difficult task to fulfil [1].

3.3 Correlation

The key point of the prediction methods based on Machine Learning is to find a correlation between the time series that must be predicted and other available related quantities. Because of this, the concept of correlation between data sets is an important notion in this field. In statistics, there are different techniques that can be utilized to determine how one data set is associated with others. Each of those methods produce a measure of the correlation that lies between -1 and 1. A correlation coefficient of 1 means perfect correlation, while a value of -1 shows that data sets are negatively correlated. Negative correlation means *"anti-correlation"*: when a variable is increasing, the other one is decreasing but in a perfectly correlated manner. Despite there are several techniques available, the most commonly used one to determine correlation is the linear correlation [41].

3.4 Pearson correlation coefficient

The *Pearson correlation coefficient* (ρ or r) is a measure of linear correlation between two sets of data [41]. It is calculated as the co-variance of two variables divided by the product of their standard deviations. In this sense, it can be seen as a normalised measurement of the covariance. Given a pair of random variables X and Y, the formula for the Pearson correlation coefficient $\rho_{X,Y}$ is [20]:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Where:

- $cov(X,Y)$ is the covariance;
- σ_X is the standard deviation of X;
- σ_Y is the standard deviation of Y.

Given paired data $\{(x_1, y_1), \ldots, (n_n, y_n)\}$, by substituting the estimates of covariances and variances and rearranging, the correlation coefficient r_{xy} can be calculated as [41]:

$$r_{x,y} = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[N \sum x_i^2 - (\sum x_i)^2][N \sum y_i^2 - (\sum y_i)^2]}}$$

The correlation coefficient can take any value from -1 to 1 and its sign indicates whether the linear relationship is positive or negative and its absolute value indicates the "strength" of the linear relationship.

- $\rho = 1$ means that two variables are perfectly correlated (a positive change in one variable perfectly predicts a positive change in the other);
- $\rho = 0$ means that two variables are not linearly correlated;

- ρ = -1 means that two variables are negatively correlated (a positive change in one variable perfectly predicts a negative change in the other).

In figure 3.4 are reported some examples of differently correlated sets of data. In figure (a) is reported an example of perfect correlation (r = 1) and anti-correlation (r= -1). In figures (b), (c) and (d) is possible to see that, when r becomes closer to zero, the linear correlation is lost.

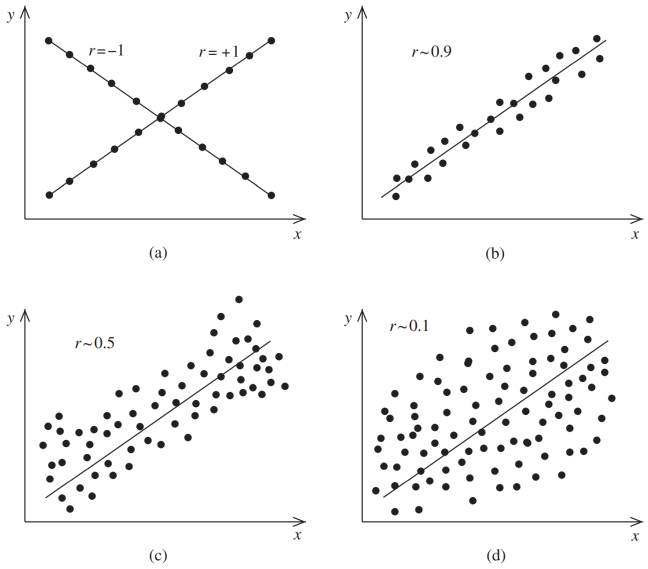


Figure 3.4: Pearson correlation coefficient [41].

In figure 3.5 are reported some other graphical representation of data sets characterized by different values of ρ. It can be observed that the correlation coefficient well reflects the strength and direction of the linear relationship but is not able to detect other kind of relation (third line of the image).

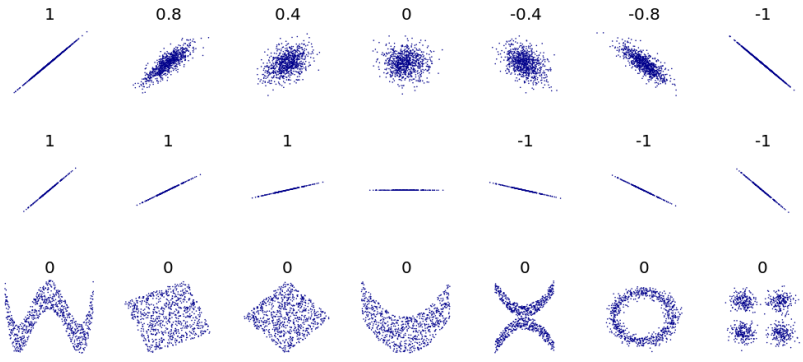


Figure 3.5: Pearson correlation coefficient.

3.5 Machine learning

Technological improvement in computer science has brought the ability of pattern recognition, which was unique to humans, to computers. Computers can now be used to make complex actions as predict the future evolution of a quantity and more. Machine learning (ML) is a standardized field founded upon the recurring patterns present in solving problems in which an algorithm is required to ”learn” from data, similarly to what happens in human brain. Nowadays ML is the most common way to forecast the future values of a time series [28].

Traditional programming is based on a deterministic output for each input. Differently, machine learning can be used to solve problems for which the relationship between inputs and outputs are not completely understood, by using a software that learns from previous experiences. Performances improve when a right amount of examples are available and a ML approach to solve problems can be thought as tuning the parameters of the model until satisfactory results are achieved. In this logic, the programmer can leave some values undecided and let the ML system to figure out the optimal values for them. Those undecided values are called ”parameters”. Providing historical data to the algorithm, it will be able to observe existing examples to figure out how to best tune parameters to achieve the best model.

A machine learning algorithm can be examined in two stages [24]:

- **Learning**: Description of the data, in form of *feature vector*, and summarization in a *model*. This approach usually follows a structured path, starting from the transformation of the data set into a representation (often a list of features) that is used by the learning algorithm to chose a model and search for the model's optimal parameters.

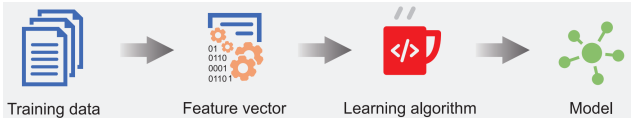


Figure 3.6: Learning step [24].

- **Interference**: Utilization of the model to process unseen data. This step uses a model that has been learned or given to produce the intended output starting from test data converted into a usable representation. This step is much faster than the learning one, which wakes it able to work with real time data.

Figure 3.7: Interference step [24].

3.5.1 Learning approaches

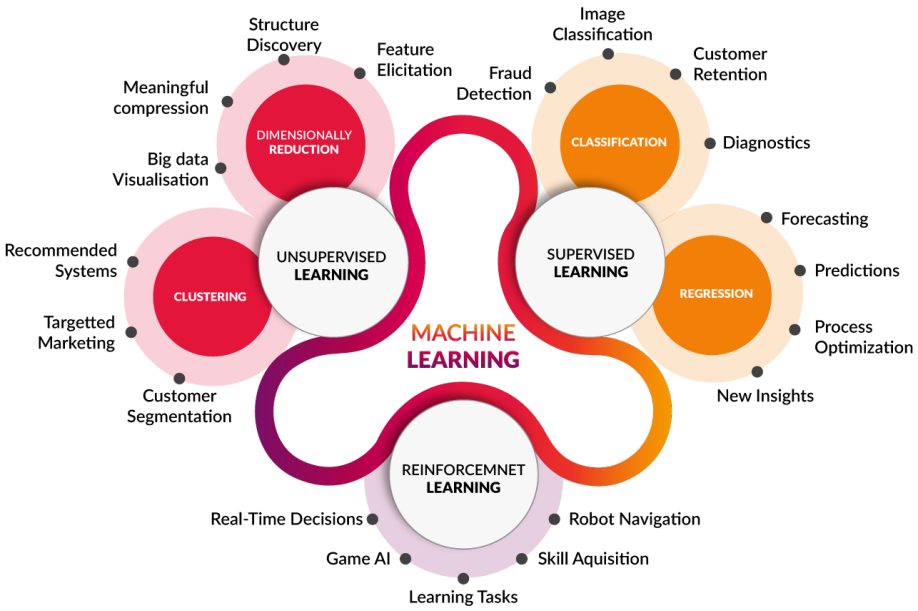


Figure 3.8: Learning categorization.[1]

There are several learning approaches that are employed in different applications of Machine learning. The most considered three types are:

- Supervised learning;
- Unsupervised learning;
- Reinforcement learning.

A more detailed scheme is depicted in figure 3.8. Predictions belongs mainly to the supervised learning logic, which is the one used by the algorithms tested in this work.

[1] https://towardsdatascience.com/the-data-fabric-for-machine-learning-part-1-2c558b7035d7

Supervised learning

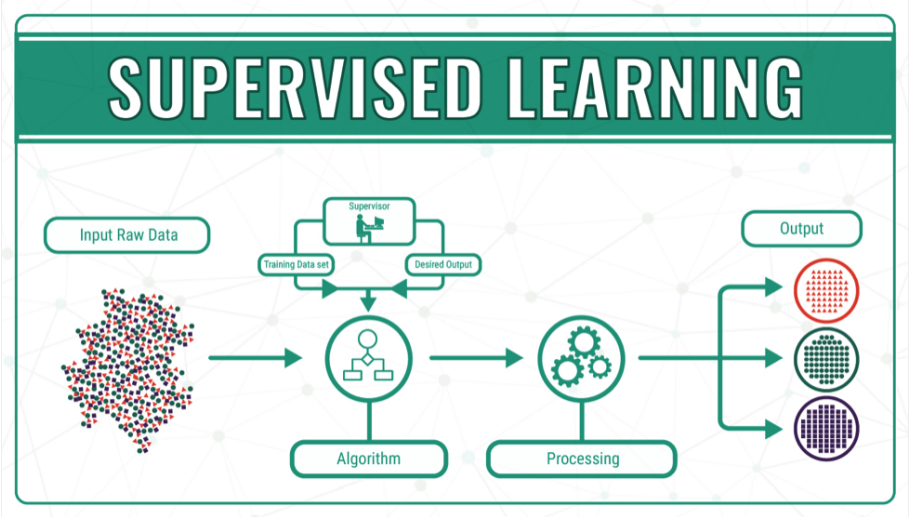


Figure 3.9: Supervised learning scheme [42].

The supervised method consists in learning from examples given by a *supervisor*. This type of learning needs labeled data, that is a collection of previous examples called *training data set*, to develop the model. In this learning phase, the output is fed to the algorithm: the machine already knows the output of the algorithm before it starts working on it. The systems needs to extrapolate the steps or the process needed to reach the output variable y starting from the input variable x [42]. At the end, a supervised machine learning algorithm must compute the parameters of the model that result in the best success of the model. A model that produces a prediction will have its success measured by the distance between this prediction and the actual values that will happen in reality. This distance is called *cost*, so the model will look for parameters that minimize the cost along all data points. Available data for learning are usually divided into two subsets, *training* and *test*, since it is better to evaluate the model on a different data set with respect to the one used for training to avoid bias errors. Since it is the learning type used in this work, a more detailed explanation of the process is present in the next chapter.

Unsupervised learning

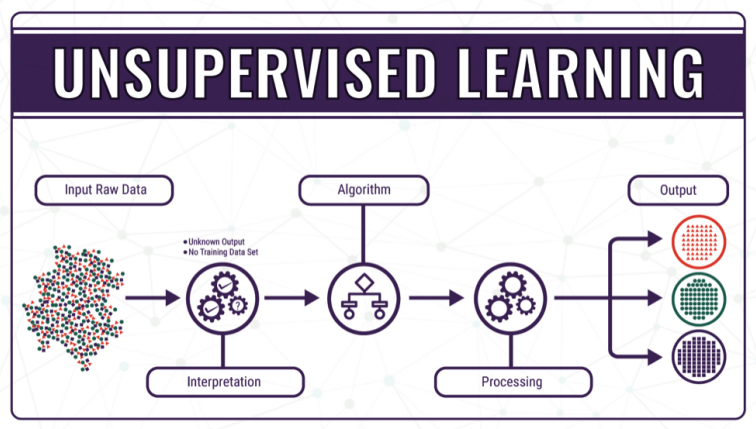


Figure 3.10: Unsupervised learning scheme [42].

Differently from supervised learning, in unsupervised learning consists in modelling data that are not given with corresponding labels. It is used in processes like "Clustering", which consists of splitting data that are not provided of corresponding labels into "buckets" of similar items.

Reinforcement learning

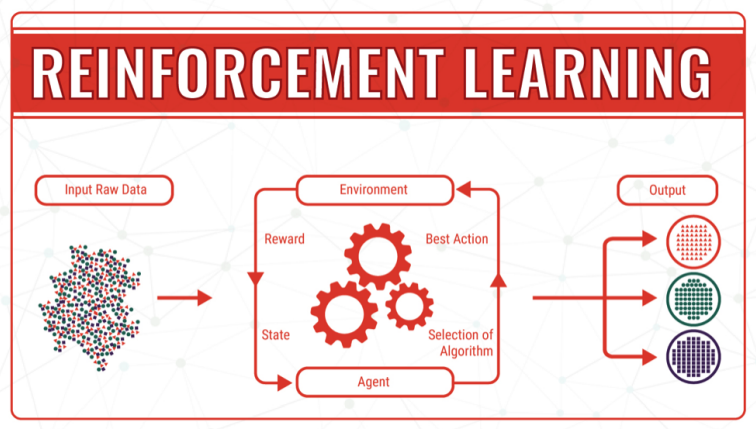


Figure 3.11: Reinforcement learning scheme [42].

Reinforcement learning approach uses the environment as supervisor: trains on the information gathered by the reaction of the environment to actions. Interacts with the environment to learn which combination of actions brings to the best results.

Chapter 4

Validation strategies

In machine learning data are used to train the machine. The trained model that the algorithm produces must then be tested on different task in order to verify if the training phase has bees successful or not. A prediction model cannot be tested on the same data set on which it was trained: the model can present a perfect score on data that it has already seen but at the same time fail to predict anything on unseen data. To avoid this situation, data is split into two parts:

- **training set**: portion of data used for training;
- **test set**: portion of data used to test the predictive performances of the trained model.

One important factor to be considered when this splitting is done is the validation size: ratio between the training and test set [49]. Different ways of splitting the data set are available and have been tested in literature. The two methodologies used in this work are the *Hold-out validation* and the *Cross-validation*.

4.1 Error assessment

In order to define the accuracy of the prediction in the test set, some error metrics are introduced to evaluate the performance of the forecasting process. The error between the predicted value and the actual one is measured in order to assess the effectiveness of one technique or another [1].

$$\textit{Prediction Error} = \textit{Actual Value} - \textit{Predicted Value}$$

In this work, the aim is to predict the future trend of time series. Error in this kind of prediction is shown in time plots, an example of this is reported in figure 4.1. The distance between the predicted trend (blue line) and the actual trend (black line) is calculated and reported in red in correspondence of the time axis. In this way is possible to visualize the evolution of the error in time.

For practical use, some parameters to evaluate the overall error committed by the prediction are calculated. Those are particularly useful to compare performances of different algorithms and different predictions.

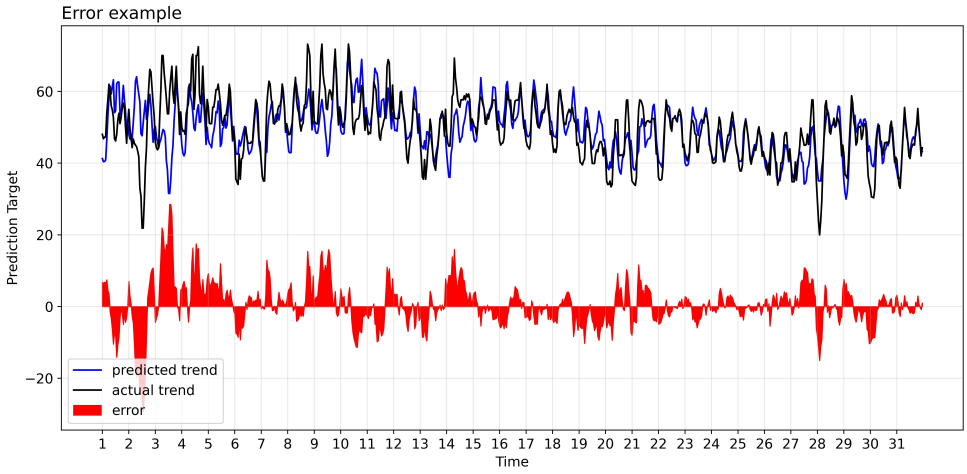


Figure 4.1: Error evolution in forecasting.

4.1.1 Mean absolute error

Mean absolute error is a measure of errors between paired observations expressing the same phenomenon [1]. Absolute errors are calculated as the absolute value of the difference between the prediction and the actual value, then they are averaged by arithmetical mean to compute the mean absolute error.

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

Where:

- $\hat{y}_i$ is the predicted value;
- y_i is the actual value;
- n is the number of observations.

Mean absolute error is scale dependent: its magnitude depends on the scale of the data. It is useful to compare different methods on the same set of data but should not be used when comparing across data sets that have different scales.

4.1.2 Normalized mean absolute error

Mean absolute error can be normalized with respect to the maximum value (y_{max}) or the mean value (y_{mean}) of the quantity that is predicted, and expressed as a percentage [7]. This parameter allows to understand the order of magnitude of the error, quantifying in percentage the deviation from the actual value. NMAE is frequently used to evaluate forecasting errors and is also used to compare the results obtained for plants of different size [27]. In this work, the normalised mean absolute error (NMAE) is used as main performance metric to compare the different algorithms.

In case of normalization with respect to the maximum value:

$$nMAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_{max}} * 100$$

In case of normalization with respect to the mean value:

$$nMAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_{mean}} * 100$$

4.1.3 Mean absolute percentage error

The mean absolute percentage error is defined as:

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

Each punctual error is normalized with respect to the actual value itself. Being a percentage error, the MAPE is scale independent, thanks to this it can be used to compare performance across different data sets [1]. One possible disadvantage is that this parameter is infinite or undefined if the actual value is null ($y_t = 0$). The *scikit* metric function used in this work solves this problem by introducing a arbitrary small yet strictly positive number (ϵ) to avoid these undefined results. In this case, the expression becomes:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{max(|y_i|, \epsilon)}$$

In this work, MAPE will be utilized to compare performances of the same algorithm applied to different data sets.

4.2 Hold-out validation

The simplest way to validate the accuracy of a prediction model is the hold-out validation. It consists in splitting data in two non-overlapping parts: one is the training part, used to fit the model, and hold-out part is the validation part used to evaluate the performances [49]. If the amount of data is large, hold-out strategy performs well, since it is expected a similar trend of the final sequence of data with respect to the whole set. Hold-out validation can have different different validation sizes (percentages of data that are held-out for testing). A common split is using 80% of data for training and the remain 20% for testing. Is not recommended to use a very small percentage of the data set for the validation part (as 10%) because data may not be properly distributed and differ vastly from the test portion of the data set. This method is utilized in this work to have a first estimation of the accuracy of the tested algorithms. In using the hold-out validation, the time taken for testing the model is shorter than the time required by a testing method like cross-validation. However, to have a better understanding of how well an algorithms performs, the cross-validation must be employed.

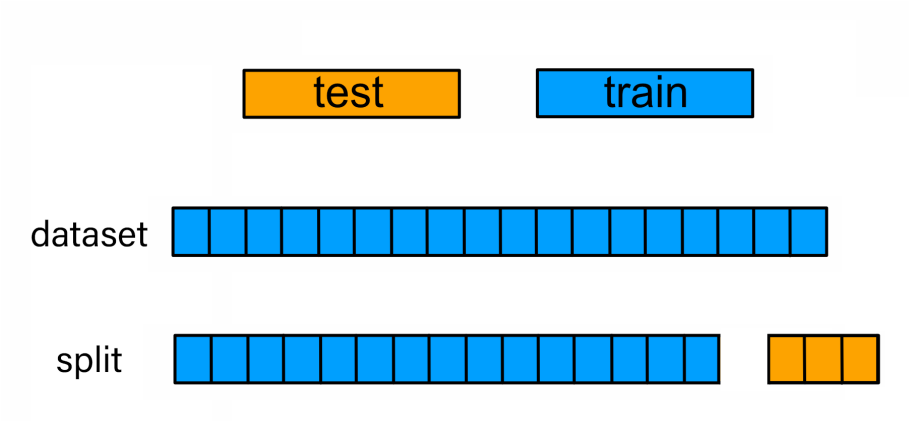


Figure 4.2: Hold-out validation visual representation.

4.3 Cross-validation

Cross-validation (CV) is a procedure for estimating the generalization performance of algorithms in the machine learning field [51][52]. It is the validation methodology used in this work to select the best algorithm for the final prediction model. CV consists in testing the algorithm on different portions of the data set, not only in one. The accuracy obtained in each of this trial is averaged to obtain the global one. According to [50], cross-validation can be used for :

- Check the generalizability of an algorithm. It can be used to predict the performances that a prediction made using the learned model will have.
- Find the most suitable algorithm for the prediction by testing multiple algorithms.

4.3.1 Leave-one-out cross-validation

The leave-one-out cross-validation iterates the hold-out strategy in order to test the algorithm on the overall data set. Given a data set of N samples, one sample is used for validation and the rest of the data set is used for training. This process is iterated N times, one for each sample of the data set and the result is a prediction for each sample. With a cross-validation is possible to compute the average of single errors to obtain the global error. A visual representation of the Leave One out strategy is reported in figure 4.3.

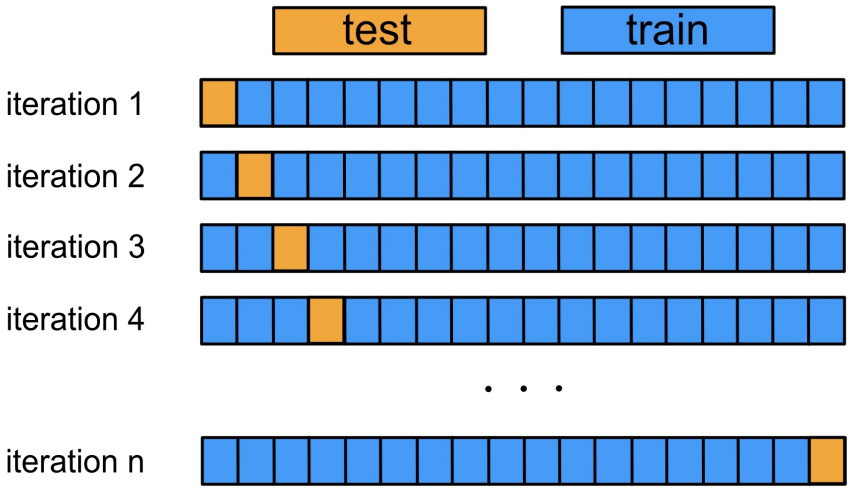


Figure 4.3: Hold-one-out cross-validation visual representation.

4.3.2 k-fold cross-validation

In a k-fold cross-validation the data set is equally portioned in K equal groups of samples of the same size, those samples are called "folds". If the number of folds corresponds to the number of samples of the data set, k-fold strategy corresponds to the Leave-one-out strategy [21]. As in the previous method, the K-fold validation trains the model on k-1 folds and uses the remaining one for test, then iterates this K times. The result is still a set of predictions, one for each fold. The accuracy obtained in each iteration is then averaged to get the model accuracy. This strategy allows to compute the mean and variance of the error, useful parameters in order to understand and compare the performance of methods. A visual representation of this strategy is reported in figure 4.4. Generally, higher is the number of k, higher is the accuracy in cross-validation. However, this may lead to problems with large data sets. For this reason, [49] suggests to use leave-one-out cross-validation only for small sets of data, with a number of instances lower than 100. For bigger data sets like the ones managed in this work, a k-fold cross-validation is more indicated. A 2-fold cross-validation must not be confused with a 50% hold-out validation: in the second one data is trained in one half of data and tested in the second half, while in the first one even if data are still split into two equal parts, the model is trained on each part in 2 different iterations and tested on both parts of the data set.

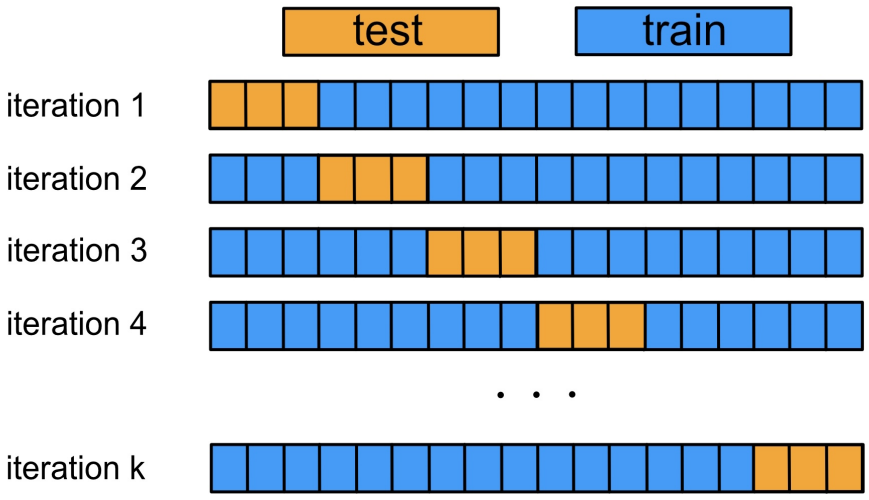


Figure 4.4: k-fold cross-validation visual representation.

4.3.3 Representation of cross-validation results

Results from cross-validations performed in this work have been reported in the form of "box-and-whisker plots". An example is visible in figure 4.5. This kinds of plots are used to graphically visualize a distribution of data (as in the case of different accuracy values coming from the validation made on different portions of the data set). The box extends from the lower to upper quartile values of the data, with a line at the median. The whiskers (vertical line) extend from the box to the most extreme, non-outlier data points (abnormal values that differs significantly from other observations [56]), to show the range of the data [1].

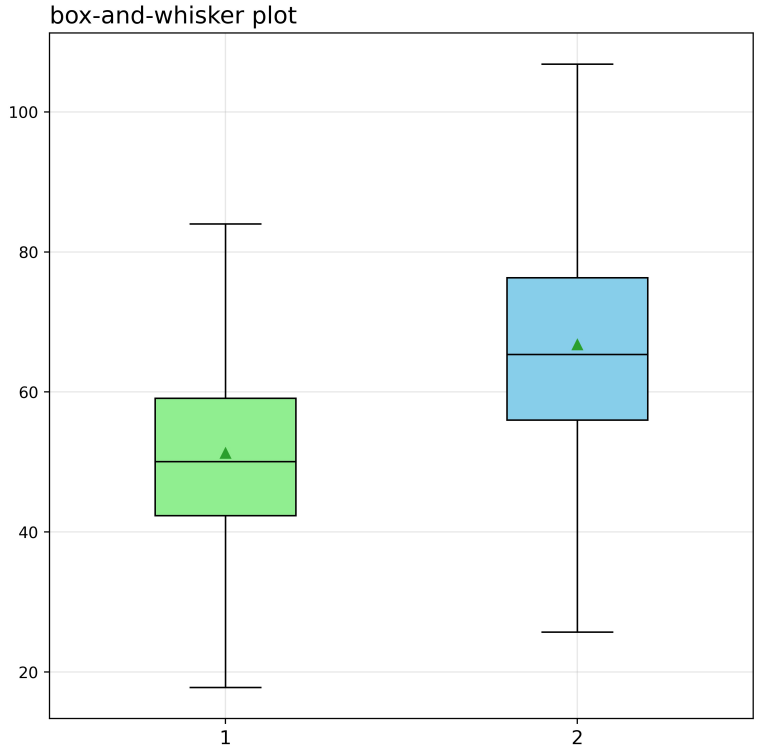


Figure 4.5: Box-and-whisker plot.

[1] https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.boxplot.html

Part III

Mathematical description of algorithms

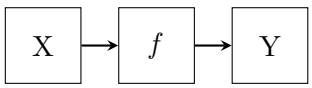
Chapter 5

Regression

In statistical modeling, processes that aim to estimate the relationship between a dependent variable and one or mode independent variables belong to *Regression Analysis*. The basic idea behind regression it to fit a function that closely represent the trend of the data set [41]. After this process, this function can be used to make predictions and estimate the future trend of the variable involved. Functions that can be used for this purpose are many, the most common ones are polynomial, exponential and logarithmic. However, it is not always possible to fit a standard function to the data through regression. A hint about the success of this process is given by the previously described Pearson correlation coefficient. The easiest form of regression analysis is represented by the linear regression.

5.1 Linear regression

The first method tested in this work is the linear regression (LR). A linear model provides the equation of a line that describes the relationship between a predictor variable X and an outcome variable Y. It can be visualised as a function f that receives the feature input and returns a predicted output.



input function output

5.1.1 Simple linear regression

The simple linear regression fits a straight line to the data. In mathematical terms, the linear function is expressed as:

$$\hat{y} = m + wx$$

The model coefficients m and w are the *intercept* of the function and its *slope*. Those parameters determine the orientation and position of the line in the xy diagram and the

aim of the machine is to find their optimal values. The best model corresponds to the model characterized by the set of coefficients that minimize the distance between the estimated values of the model and the actual values in data, this difference corresponds to the residuals (R).

$$R = \hat{y}_i - y_i = m + wx_i$$

This can be visualized in figure 5.1.

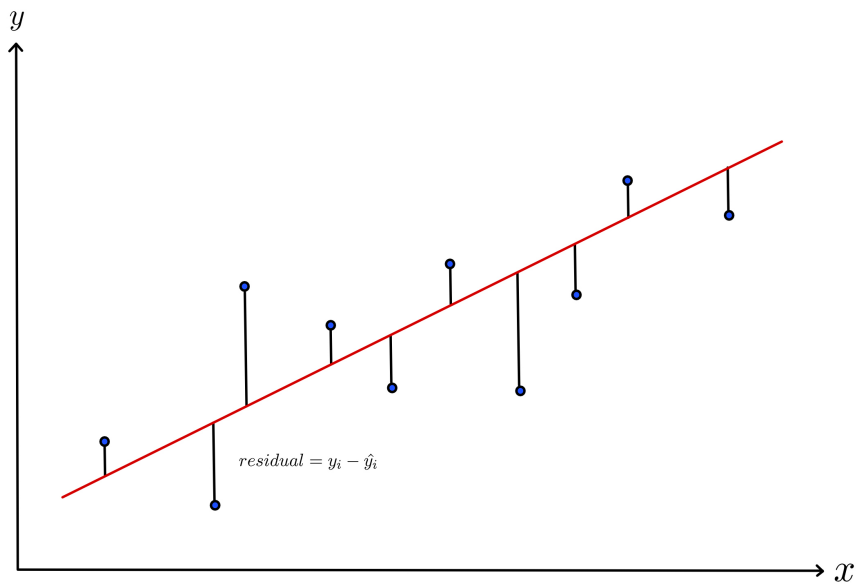


Figure 5.1: Linear regression residuals.

The minimization of residuals means that their sum must be a minimum. However, a simple minimization of the residual summation would produce a null result since positive residuals would cancel out the negative ones. To overcome this problem, it is employed the minimization of the mean squared error, that is a measure of the average of the squares of the residuals. In this case, the line of best fit is the line whose coefficients m (y-intercept) and w (slope) minimize the mean squared error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

5.1.2 Multiple linear regression

Multiple linear regression refers to the case in which there are more than two variables involved. This happens when the target of the model is to predict the output y exploiting its correlation with multiple input variables $[x_1, \ldots, x_n]$. In this case the dependent variable is dependent upon several independent variables.

A generalized regression model involving multiple variables is represented by an equation of an hyperplane:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

As before, this approach fits a linear model with coefficients $w = (w_0, \ldots, w_n)$ to minimize the residual sum of squares between the observed values in the data set, and the values predicted by the linear approximation [11].

This is the model utilized in this work, since the performed predictions of target variables always take in more than one input.

Chapter 6

Support vector regression

The second method tested in this work is the support vector regression (SVR), which has been already adopted in the field of electricity market prediction [30][31]. SVR applies the concept of supported vector machines (SVM), which is an effective and significant method for classification problems [23], in the process of regression. SVM can be categorized as a supervised learning method for the application of regression and classification [29].

In linear regression models, the aim is to minimize the error of the test set by means of proper optimization of model parameters: in particular, the interested error is evaluated between the predicted value ($\hat{y}_i$) and the observed one (y_i). Predicted value ($\hat{y}_i$) in regression is computed starting from the input feature (x_i) thought a slope coefficient (w_i). The equation for Linear Regression can be reformulated as:

$$MIN \sum_{i=1}^{n} (y_i - w_i x_i)^2$$

In SVR is possible to define a degree of acceptance for the error and thanks to this is possible to obtain better performances in some cases [15]. The objective function of this method is to minimize the coefficients, not the error.

According to this, the error must remain bounded between pre-defined constrains: the absolute error is set to be less or equal to a specified margin given by the maximum error (ϵ). This parameter can be tuned to gain the desired accuracy of the model. Errors that are smaller than epsilon are accepted, but any deviation larger than this is rejected. The possibility to have errors larger than epsilon is handled by slack variables: for every error that falls outside the specified range, its deviation from the margin is denoted as ξ. The training error is minimized by minimizing ξ_i, and $||w||^2$ is minimized to raise the flatness off $f(x)$.

Function to minimize:

$$MIN \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} |\xi_i|$$

Constraints:

$$|y_i - w_i x_i| \leq \epsilon + |\xi_i|$$

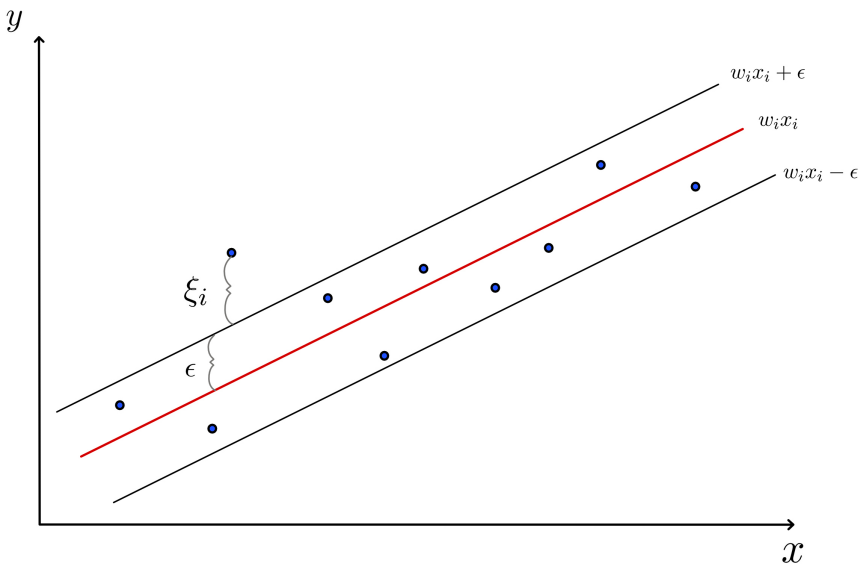


Figure 6.1: Support vector regression representation.

C is the hyper-parameter that can be tuned, it determines the trade-off between the flatness of the function and the maximum tolerated error [23]. A higher value of C makes the tolerance for points outside the maximum error epsilon increase. When C is null, there is no tolerance for values outside the range. C can be optimized to better fit the model with data.

6.1 Linear SVR

The library used in this work applies a linear support vector regression: a SVR that employs a linear *kernel* [1], a function that maps data from the *input space* to a higher dimensional *feature space* where the regression itself is performed [55]. Throughout this function is possible to find the hyper-plane in the hyper dimensional space starting from input data. If it is considered the problem in a 2 dimensional space, a linear function used as kernel means that the shape in which data will be mapped is going to be a straight line. This can be visualized as the red line shown in figure (6.1). Linear SVR is recommended for large data sets and provides the fastest implementation among the possible kernels that can be used with this method [2]. As linear regression, this prediction method is again based on the linear correlation between variables.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html

[2] https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0

Chapter 7

Neural networks

One of the most used machine learning methods to perform forecast are neural networks (NN) [32]. A prediction algorithm based on a NN consists in an iterative procedure by which a collection of units (neurons) transmit and process signals in order to create an associative mapping between the input x and an output y. A scheme of a 2-layer neural network is reported in figure 7.1.

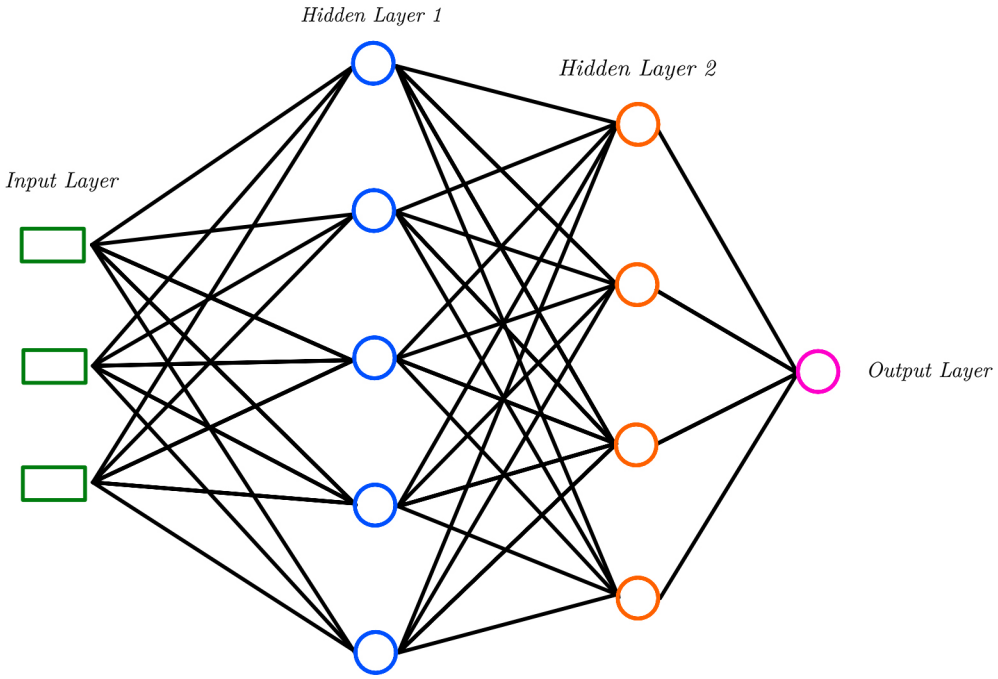


Figure 7.1: Graphical representation of a neural network topology.

In feed-forward networks, units are arranged in layers [8]:

- An input layer;
- One or more hidden layers;
- An output layer.

All neurons are interconnected: each neuron of the first hidden layer receives as input each data of the input layer. Neurons of successive layers receives as input the output of each neuron of the previous hidden layer [11]. In this sense, a neural network is considered fully connected. The interconnection of neurons across all the hidden layers of the network gives to the model the ability to fit the non-linear parts of the data set. A neural network follows an iterative procedure based on a stochastic process, since the first value of the weights of neurons are randomly set at the beginning. Those trials values are optimized during iterations to minimize the error. For this reason, the resulting forecasts depend on the specific trial. Therefore, different trials can provide slightly different results [10]. The number of units in the hidden layer is chosen by trial and error [19].

7.1 How neurons work

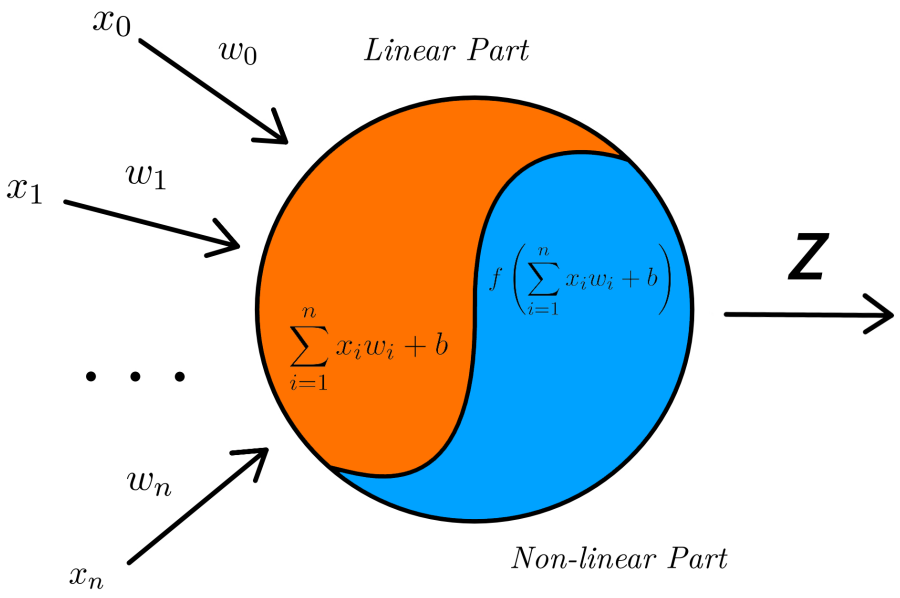


Figure 7.2: Graphical representation of weighted connection to neurons.

Each neuron of the network is a non-linear activation function that describes the movement of the signal from a layer j to a layer i. Generally, each neuron is characterised by a connection characterized by a weight (w_i) that modifies the input, before that it arrives to the neuron. Each neuron also includes a bias term (b_i) and an activation function, as it is represented in figure 7.2. Each of them performs a weighed sum of its inputs, the bias (b) is added.

The same input arrives to different neurons after being modified by different weights.

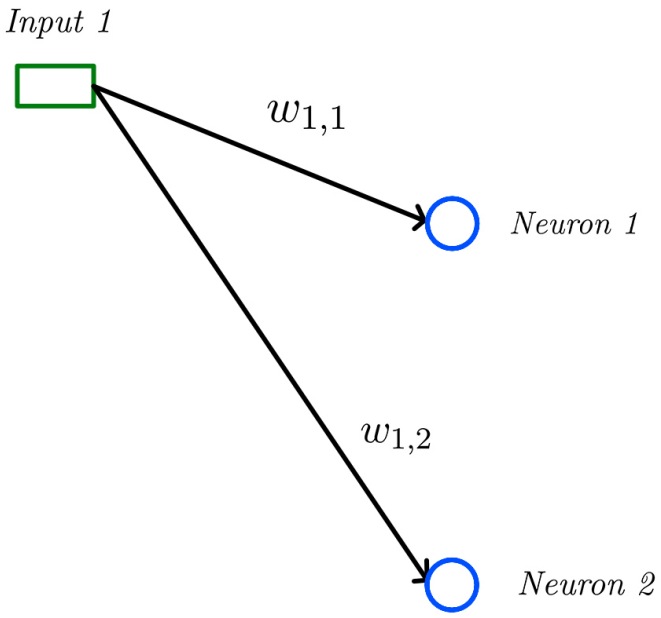


Figure 7.3: Graphical representation of weighted connection to two neurons.

Once the vector **Z** of outputs is computed, the activation function can be applied to each element of it to obtain the output vector of the layer. This output will be the input for successive layer of neurons that works in the same way. The procedure is repeated until the final output of the model is computed.

7.2 Activation function

The activation function is what defines the output of neurons in artificial networks. Non-linear activation functions allow networks to compute complex problems using a small number of nodes. The function employed in the algorithms tested in this work is a rectifier linear unit (ReLU) [22].

$$f(x) = x^+ = max(0, x)$$

ReLU is defined as the positive part of its argument and it is also known as "ramp function". Since is nonzero only in the positive part, it allows sparse activation of neurons. A visual representation of its trend is shown in figure 7.4.

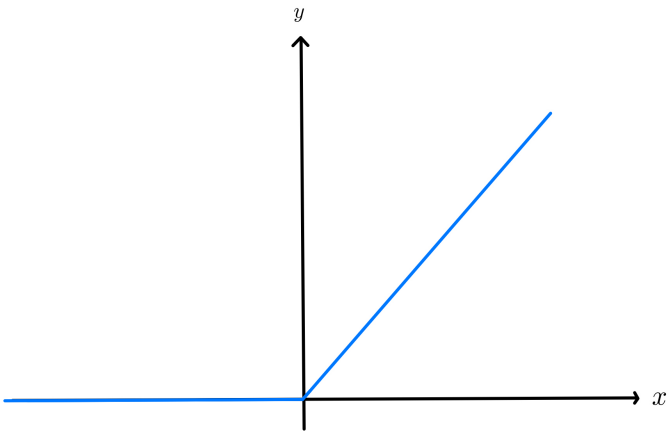


Figure 7.4: Graphical representation of weighted connection to two neurons.

7.3 Learning process

Training of a feed-forward neural network is performed in supervised manner: a training set is available, containing both the inputs for the algorithm and the expected outcome. In the training process, a neural network constructs an input–output mapping, adjusting the weights and biases at each iteration based on the minimization of some error measure between the output produced and the desired output. The process is iterated until a convergence criterion is reached [19]. The training process of a neural network is based on the definition of a cost function that is minimized by a gradient descend optimization. The cost function is again a measure of the error of the prediction with respect to the actual value. The aim is to find the set of weights and biases that minimize this cost function. After the definition of the cost function, a gradient descend optimization is applied to find its minimum. The gradient of a function is vector whose elements represent partial derivatives of the function with respect to each of its parameters. Each partial derivative can be used to estimate how much the function changes when a particular parameter changes. Based on this definition, a gradient descend optimization is subdivided in the following steps:

- Gradient computation in a point;
- Modification of each parameter by an amount proportional and opposite to its partial derivative;
- Recompute the gradient using modified parameters and repeat until the minimum is reached.

7.4 Back-propagation

The most common learning algorithm is back-propagation: error is propagated from output to inputs of the neural network, adjusting the weights and biases in each layer [19]. The aim is to calculate the error attributable to each neuron starting from the last layer before the output to the first one after the input. The error committed by each neuron is proportional to the impact that that neuron's output has on the cost function: if the error attributed to a certain neuron is considerably higher than the one attributed to the other neurons, changes on weight and bias of that neuron will have a greater impact on the overall output of the model. Back propagation of the error starts from the last neuron, that is the ultimate source of error of the model, since it is the one that produces the final output. This error is moved backwards through the model using the same weight and connections that are used for the forward propagation of the signal (figure 7.5).

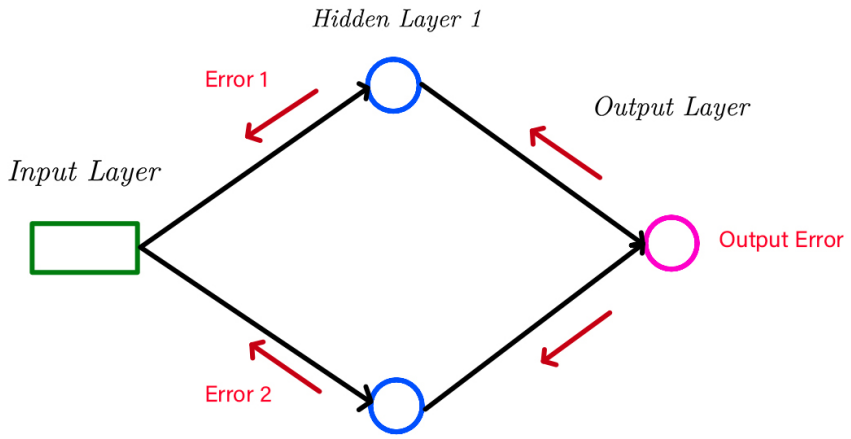


Figure 7.5: Hold-one-out visual representation.

When the error attributable to each neuron is obtained, it is possible to obtain the gradient vector of the cost function. Each error allows to obtain the partial derivative of the cost function with respect to the input of the single neuron, and those are the components of the gradient vector. The most active neurons that are activated mode frequently are going to have their weight and bias modified more. When the gradient vector is computed, the gradient descend optimization can be applied. In order to accelerate the learning process, one parameter of the back-propagation that is adjusted is the learning rate: proportion of error gradient by which the weights should be adjusted. Larger values can give a faster convergence to the minimum but also may produce oscillation around the minimum [19].

Chapter 8

Hyperparameter optimization

In ML, several learning algorithms are characterized by a set of hyperparameters: parameters whose values can be used to control the learning process and maximize the usefulness of the learning approach. Those are used to configure various aspects of the learning algorithm and have various effects on the resulting model and its performance. Hyperparameters are different from other parameters whose values are obtained by the algorithm during the learning process, as it happens with the weights w_i in previously describer algorithms. The *hyperparameter optimization* is the process that aims to obtain the best set of hyperparameters for the learning algorithm: the best performances are reached when the algorithm is tuned on those optimized values. This optimization is carried out by minimizing a predefined loss function on given data, as the training set of the learning algorithm [43].

8.1 Grid search

The *grid search optimization* is the simplest way to perform the hyperparameter optimization. In consists in an exhaustive search through a specified set of values. The fist step is to specify which parameters of the algorithm must be optimized and input the set of values that each of those can assume. This can be done by specifying a range and then dividing it into a discrete set of subsequent numbers. The grid search optimized will test the algorithm using every possible combination of those values. The combination that performed better according to a previously specified performance metric is chosen as the best one.

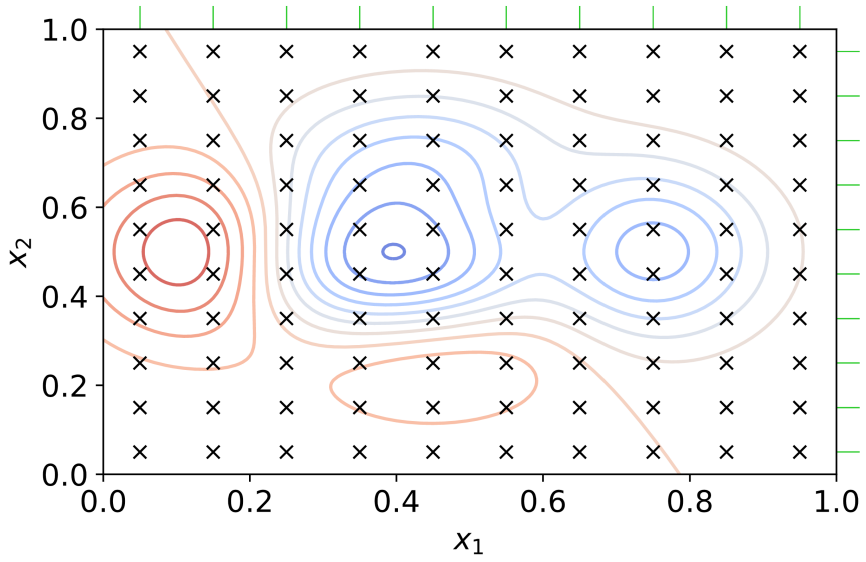


Figure 8.1: Grid search graphical representation.[1]

This is the optimization method that has been adopted in this work and the performance metric utilized was the mean absolute error. At the end of this process, what is obtained is the set of parameters that make the algorithm produce the prediction with the lowest MAE between the tested ones. The algorithms that have been optimized are the SVR and NN. For SVR, the optimized parameters were:

- C: hyperparameter that tunes the tolerance of errors greater than the maximum one ϵ. Its first trial value is 1.0 and it is optimized in a range from 1.0 to 10.0;

- ϵ: maximum error accepted by the model. This parameter specifies the range in which no penalty is associated in the training loss function to predicted points that falls withing a distance equal or smaller than ϵ from the actual value. Its initial trial value is 0.0 and is optimized between 0.0 and 1.0.

[1] Alexander Elvers, CC BY-SA 4.0 `https://creativecommons.org/licenses/by-sa/4.0`

In [44] is recommended the usage of a grid search for the optimization of parameters in the Supported Vector Regression among most advanced methods since it is "safer" and the computational time required is not much more since only two parameters must be optimized.

For NN, the optimized hyperparameters were:

- Batch Size: number of samples that are propagated through the network in one forward/backward pass [45]. The size of the batch have a strong effect on accuracy of the network and on time required to reach convergence. Using a small batch the algorithm can converge faster than with larger one but may not find minima that would reach with a larger batch size. In addition, a small batch size can have a regularization effect because of its high variance [46] but a smaller learning rate is required in this case to prevent the loss of minima [47];

- Learning Rate: this parameter quantifies how much the weights of neurons are modified with respect to the loss gradient at each iteration. Lower it is this value, slower the algorithm goes towards the minimum. The advantage is that local minima are less likely to be missed [48]. The first trial value is 0.001.

- Topology: structure of the network as the number of hidden layers and number of neurons inside them. Several configurations can tested to try to achieve a performance boost.

Part IV

Application and selection of the models

Chapter 9

PV power forecasting

The first task that was addressed in this work was the intermittent renewable energy production forecasting: prediction of the amount of power that will be produced by a PV field. Starting from the historical trend of the produced power and related quantities as modules temperature and irradiation, models have been set and trained to perform the desired task. The aim of this section is to find the best algorithm to perform this prediction and optimize the correspondent parameters to develop a final model that can be used in future. In a following section, this model has been utilized for a case study in which a real situation is simulated.

As it was already mentioned, the three methods that have been tested and compared are:

- Linear regression (LR);
- Supported vector regression (SVR);
- Neural networks (NN).

Each of them was trained and tested on the same data set and the best one is selected according to a common proper validation strategy. The validation strategies employed to establish which method was the best one were:

- A hold-out technique: calculation of the normalized mean absolute error of the model, after being trained on the 70% of data and tested on the remaining 30%. This serves as a preliminary check for the performances of the algorithm: since it is computationally inexpensive, it is used as first test to evaluate if initial setup of the model was correct.
- A cross-validation: k-fold cross-validation has been adopted for the estimation of the general performances of the algorithms only after having checked that the algorithm works by means of the previous validation method. This validation is computationally more expensive but is crucial to test properly the algorithm on the overall data set and generalize its performances.

Parameters of both support vector regression and neural networks have been optimized by means of a Grid Search Optimization. At the end, the cross-validation is repeated on optimized algorithms in order to evaluate the improvement coming from this last step and only after this operation the best and optimized model is chosen.

9.1 PV data set

The analysis was based on a data set of hourly measures related to a PV field located in *Fossano (CN)*, provided by the owner of this system `EGEA SPA`. The selected data set contains measures related to a time period that goes from August 1st 2018 to May 31st 2021 and includes measures of:

- Produced power [MW]: measured power in correspondence of three different inverters along the system was reported on the original data set. In this work, it is considered the total electric power produced by the system obtained as the summation of those three measures;
- Solar radiation [W/m^2]: power per unit area received from the Sun by panels;
- Panels temperature [°C]: measured temperature on PV panels surface.

9.2 Data preprocessing

The original data have been provided in a *.csv* file in which samples are are arranged in rows and features in columns. Each row contain the measure of power by the three inverters, solar radiation and panels temperature related to a certain hour in a certain day. Those five information are spaced by a semi-colon as delimiter. Data have been imported using `Pandas`, a data analysis and manipulating tool. Thanks to this library it was possible to manipulate the original file and obtain a data set suitable for the algorithm [53]. Only the required columns are imported in python, the rest is skipped. The first step is to import the library into the script, then is possible to use the specific csv file reader command to import the file, specifying the file name, the delimiter and the index of rows to skip.

Code Listing 9.1: PV data set import

```
import pandas as pd

data = 'pv_egea_acaja_04062021135526.csv'
PV_data = pd.read_csv(data, delimiter = ";", skiprows = rows_to_skip)
```

At this point, a name is assigned to each column, the total power is computed as the summation of the three measures by the three inverters and time information are converted in a suitable date format to be used as index for the final set.

Code Listing 9.2: PV data set creation

```
PV_data.columns = ['day','time','power A','power B','power C','radiation','
    temp']
PV_data[['power A','power B','power C','radiation','temp']].apply(pd.
    to_numeric)

power = pd.Series((PV_data['power A'] + PV_data['power B']
                   + PV_data['power C'])*1e-6 , name = 'power')

date = pd.Series(PV_data["day"] + [" "] + PV_data["time"], name = 'date')

PV_prod = pd.concat([date, power, PV_data["radiation"],
                     PV_data["temp"]], axis = 1).set_index('date')
```

The result of the pre-elaboration is a data set containing three columns (power, radiation and temperature), indexed by date and with a resolution of 1 hour.

In figure 9.1 is reported the graphical representation of the pre-elaborated data, corresponding to the trend of the three time series utilized in this section, contained in the data set that will be elaborated. Solar radiation (represented in yellow) almost perfectly follows the power trend (represented in orange). Panels temperature is represented in brown and it also shows a compatible trend with the other two quantities.

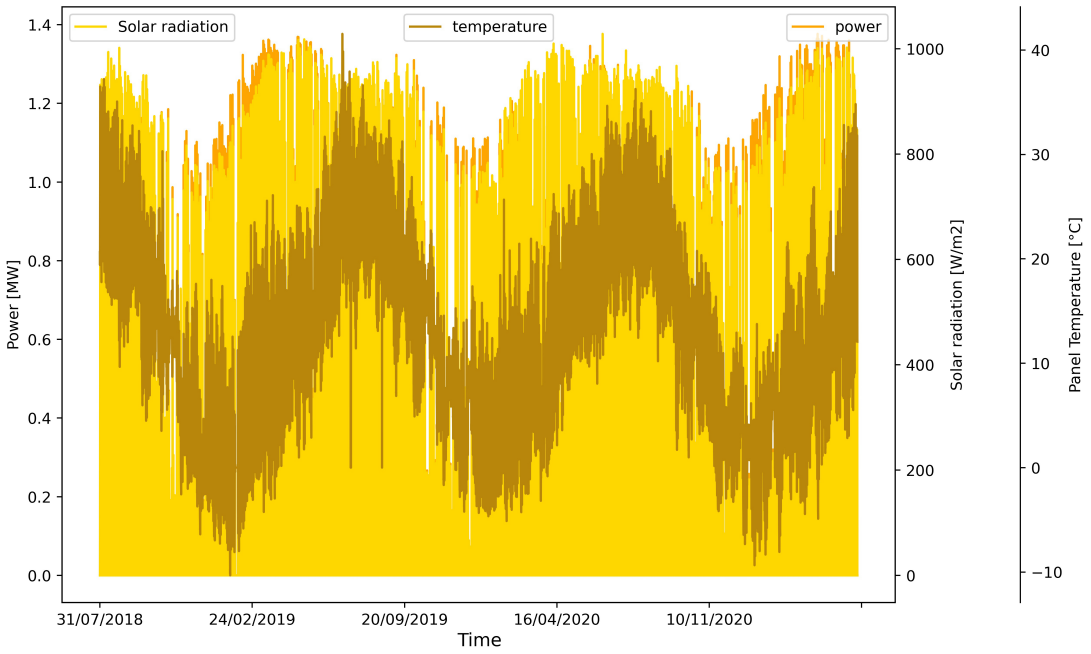


Figure 9.1: Power, radiation and temperature evolution.

Before the accuracy estimation of the proposed methods, the Pearson correlation coefficient related to the correlation between time series fed to the algorithms has been computed. This parameter gives an important information about the linear correlation

that exist between the power provided by PV panels, the incoming direct normal irradiation and the panels temperature. This parameter is computed again using a Pandas feature that creates a diagonal matrix containing the correlation value between the different columns contained in the data set.

Code Listing 9.3: PV Pearson correlation coefficient calculation

```
PV_prod.corr()
```

The obtained matrix is reported in figure 9.2 in form of a heat map.

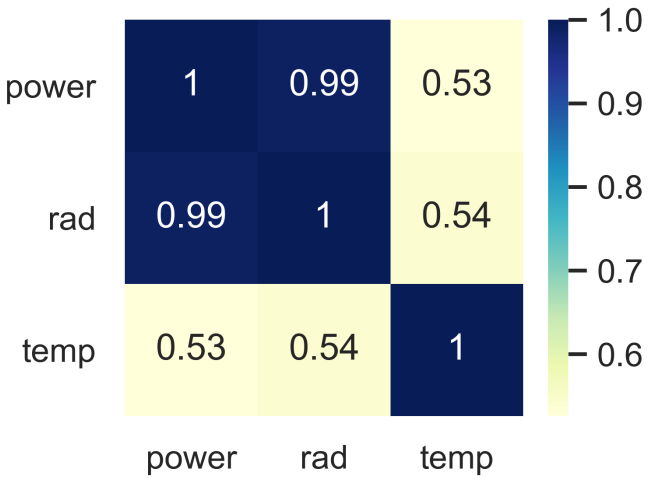


Figure 9.2: Pearson correlation coefficient between PV forecasting data sets.

The correlation between power and direct normal irradiation is roughly 0.99, meaning that a very strong linear correlation exists between those two parameters. On the other hand, the correlation between power and temperature of panels is around 0.5, a correlation is present between those two, but not as linear as it is with radiation. This value gives a hint that for this application a simple linear estimator, such as a classic linear regression, could also achieve good results. A linear model trained on the correlation between power production and solar radiation would be suitable to predict the power production when the radiation is provided.

9.3 Hold-out validation

As a preliminary verification of the algorithms, a simple hold-out validation strategy was performed: the model was trained on 70% of data and tested on the remaining 30%.

Train data set includes all the three given time series that are given to the model, while in the test data set the power production is not included as it is what the algorithms must predict. The former is represented by a series of features plus target combinations so that the algorithm can map the proper fitting function that in the future will be used to evalue the output giving the same inputs. This is the classical supervised learning

approach. Algorithms take in scaled data in order to perform the optimisation of fitting parameters on a scaled range. To do so, another function of the *sklearn* library was employed: *MinMaxScaler*. This method saturates all the data between 0 and 1 before giving them to the model helps model convergence towards the global minimum. A function named *linear_model_scaling* takes in the training size, the variable that must be predicted y (PV power) and the predictor variables X (solar radiation and panels temperature). This function scales those quantities, creates the training and test sets and outputs a model-set containing both.

Code Listing 9.4: Linear model scaling

```
X = PV_prod.drop(columns = ["power"]).values
y = PV_prod["power"].values
m = X.shape[0]
nx = X.shape[1]
train_size = 0.7

model_set = linear_model_scaling(train_size, X, y)

def linear_model_scaling(train_size, X, y):

    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler()

    X_scaled = scaler.fit_transform(X)

    m_train = int(train_size*m)
    x_train, y_train = X_scaled[:m_train, :], y[:m_train]
    x_test, y_test = X_scaled[m_train:, :], y[m_train:]

    model_set = {}
    model_set["x_train"] = x_train
    model_set["x_test"] = x_test
    model_set["y_train"] = y_train
    model_set["y_test"] = y_test

    return model_set
```

First configuration of algorithms

For the first trial, algorithms have been configured in this way:

- Linear regression was employed as reference.
- Support vector regression was configured with default parameters: hyper-parameter $C = 1.0$ and the maximum admissible error $\epsilon = 0.0$.
- The neural network was tested using a 10, 10, 6 topology: 10 neurons in the fist hidden layer, anther 10 neurons in the second hidden layer and 6 neurons in the last one. The learning rate was set as 10^{-3} and the solver used was *Adam*.

Models are imported from the library *sklearn* and fitted to the model-set created by the previous function. At this point algorithms are used to predict the trend of power production and predictions are stored in a vector *y_pred*.

Code Listing 9.5: Linear regression hold-out validation

```
from sklearn.linear_model import LinearRegression

model = LinearRegression().fit(model_set["x_train"], model_set["y_train"])
y_pred = model.predict(model_set["x_test"])
```

Code Listing 9.6: Support vector regression hold-out validation

```
from sklearn.svm import LinearSVR

model = LinearSVR().fit(model_set["x_train"], model_set["y_train"])
y_pred = model.predict(model_set["x_test"])
```

Code Listing 9.7: Neural networks hold-out validation

```
from sklearn.neural_network import MLPRegressor

model = MLPRegressor(hidden_layer_sizes = (5, 5, 3),
                     solver = "adam",
                     learning_rate_init = 1e-3,
                     shuffle = False).fit(model_set["x_train"],
                     model_set["y_train"])
y_pred = model.predict(model_set["x_test"])
```

A graphical comparison is reported in figures 9.3, 9.4 and 9.5 between the prediction and actual values along a time span corresponding to a week inside the testing period. It is possible to observe that a satisfactory accuracy is reached by all the three algorithms as the prediction (blue curve) is able to follow the real trend (orange curve).

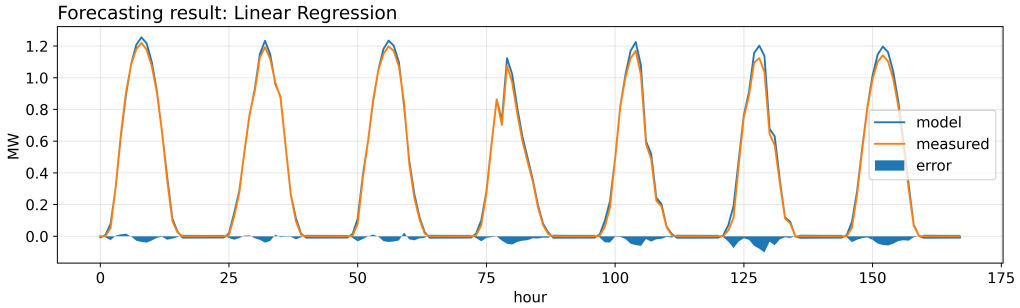


Figure 9.3: Model vs measured plot for PV forecasting using LR.

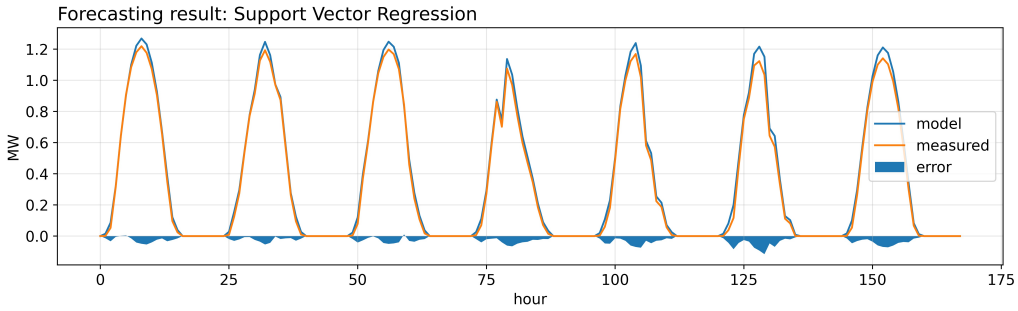


Figure 9.4: Model vs measured plot for PV forecasting using SVR.

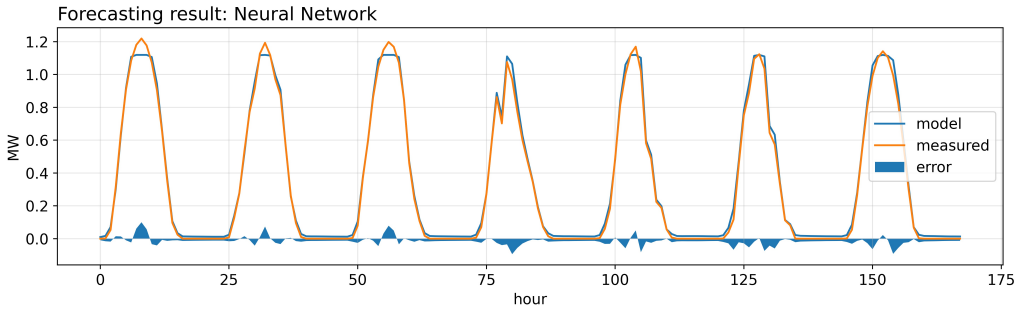


Figure 9.5: Model vs measured plot for PV forecasting using NN.

Normalized mean absolute error

A first numerical assessment of the accuracy of three different algorithms is performed by evaluating the normalized mean absolute error (NMAE) between the computed prediction and actual data. The MAE metric is imported from *sklearn* library.

Code Listing 9.8: Mean absolute error

```
from sklearn.metrics import mean_absolute_error

MAE = metrics.mean_absolute_error(model_set["y_test"], y_pred)
```

The computed MAE is then normalized with respect to the maximum power that is produced by the PV field: $max\_power = np.max(y)$. Results are shown in a bar plot reported in figure 9.6.

Each of the three tested algorithms show satisfactory results as the NMAE is around 2%. The best results comes from the supported vector regression, with an error of around 2.1%, while the other two algorithms are characterized by a slightly higher error.

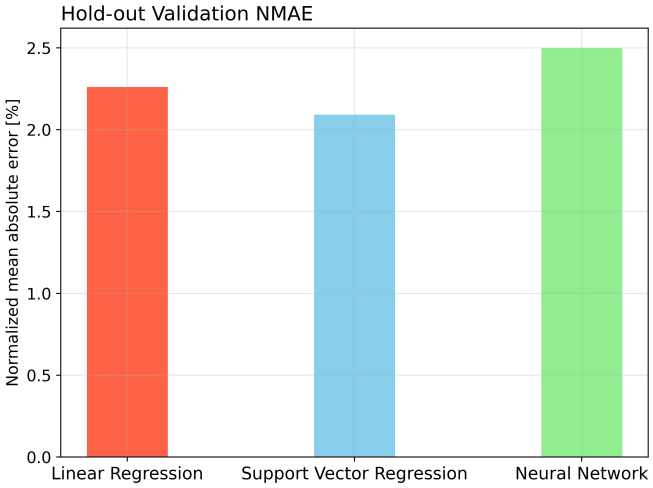


Figure 9.6: Normalized mean absolute error.

9.4 k-fold cross-validation

A cross-validation procedure was performed to better estimate the accuracy of the model using multiple sub-sets of data as test sets, not only the last portion of it. Such validation was performed using a *sklearn* library *cross_val_score*. The number of folds used in this analysis matched the number of months of the data set employed, so that the model is tested on each month present in the time frame. Folds are obtained using a tool named *KFold*, again from *sklearn*. Since the time period covered by the input data set goes from August 1st 2018 to May 31st 2021, 34 folds have been used. Since the test is repeated k times, the result will be a distribution of the error among all months that can be described using a mean value and a deviation. In this way it is possible to assess if an algorithm is consistent: a low value of the error deviation among all the different test periods shows that it is able to correctly predict the variable in different conditions. For this validation the data set is again divided between the variable that must be predicted y (PV power) and the predictor variables X (solar radiation and panels temperature). Cross-validation is then applied to the three algorithms, using scaled predictor variables to predict the power. The aim is again to evaluate the normalize mean absolute error. For this reason, among the available scoring metrics, the negative mean absolute error (*neg_mean_absolute_error*) was selected. To store a positive value of this metrics, a negative sign is imposed before the application of the function. In this first cross-validation, algorithms are tested using the standard configuration described for the simple hold-out validation.

Code Listing 9.9: k-fold validation

```
from sklearn.model_selection import cross_val_score, KFold

kfold = KFold(n_splits = 34, shuffle = False)
from sklearn.model_selection import cross_val_score, KFold

kfold = KFold(n_splits = 34, shuffle = False)
X = MinMaxScaler().fit_transform(PV_prod.drop(columns = ["power"]).values)
y = PV_prod["power"].values

## Cross-validation for Linear Regression
lr_acc = - cross_val_score(LinearRegression(), X, y,
                           scoring = "neg_mean_absolute_error",
                           cv = kfold, n_jobs = -1)

## Cross-validation for Support Vector Regression
SVR_acc = - cross_val_score(LinearSVR(), X, y,
                              scoring = "neg_mean_absolute_error",
                              cv = kfold, n_jobs = -1)

## Cross-validation for Neural Networks
NN_acc = - cross_val_score(MLPRegressor(hidden_layer_sizes = (5, 5, 3),
                          solver = "adam",
                          learning_rate_init = 1e-3,
                          shuffle = False).fit(model_set["x_train"],
                          model_set["y_train"]),
                          X, y, scoring = "neg_mean_absolute_error",
                          cv = kfold, n_jobs = -1)
```

Three vectors LR_acc, SVR_acc and NN_acc are obtained. Each of them stores the MAE of each iteration of the cross-validation. Those errors are normalized on the maximum power produced by the system and such distributions are represented in figure 9.7 by means of a box-and-whisker plot.

The algorithm that produced the best results in terms of mean value of the normalized absolute mean error was the Support Vector Regression. Both SVR and LR are quite consistent: they both show a small error deviation. The neural network algorithm, in this first tested configuration, produced the worse results both in terms of mean value of the absolute error and consistency.

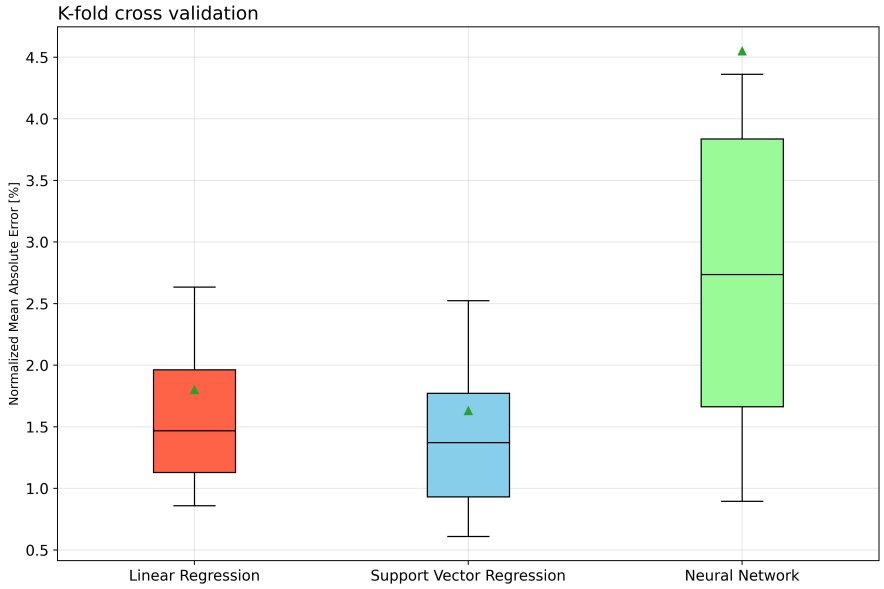


Figure 9.7: k-fold validation results for PV forecasting.

9.5 Optimization of parameters

Parameters of support vector regression and neural networks were optimized by means of a "grid search optimization". In this kind of analysis, it is possible to assess which are the parameters that allows the algorithm to perform best with respect to a certain scoring metric that can be selected. In this case, the scoring metric selected is the mean absolute error. The algorithm is tested using different values of the parameters to be involved in the optimization, and at the end the value that produces the best result is highlighted. Another *sklearn* library was employed, namely the *GridSearchCV*.

Code Listing 9.10: Grid search library input

```
from sklearn.model_selection import GridSearchCV
```

Ranges of values that each parameter can assume were defined and algorithms were tested using each combination of those. The optimization outputs the parameter set that produced the lowest value of the MAE.

9.5.1 SVR optimization

Parameters to be optimized in support vector regression are: the hyperparameter C and the maximum acceptable error ϵ. In the standard configuration, C in set by default at 1.0 and in the optimization it was tested in the range between 1.0 and 10, discretized in 30 intervals of the same length, while Epsilon is 0.0 by default and tested in the range between 0.0 and 10, managed as before.

Code Listing 9.11: Grid search optimization for support vector regression

```
estimator = LinearSVR()
N = 30
parameters = { 'C':np.linspace(1.0,10,N,dtype=np.float32).tolist(),
               'epsilon': np.linspace(0,10,N,dtype=np.float32).tolist()}

GS_SVR = GridSearchCV(estimator,parameters,
                      scoring = "neg_mean_absolute_error", n_jobs = -1)
GS_SVR = GS_SVR.fit(X, y)
SVR_best_param = GS_SVR.best_params_
print(SVR_best_param)
```

The optimization output was:

- *C = 1.62*
- *epsilon = 0.0*

The best algorithm configuration according to Grid Search is very close to the default one. The acceptable error Epsilon that still produces the best accuracy is still null and the hyperparameter C that best performs is very close to 1. The strongly linear correlation between the solar radiation and the produced power means that a linear model is perfectly suited for this application, and also a simple linear regression could perform this task. This is why there is no need to tolerate relatively higher errors (ϵ is null) or manage values outside the toleration band (small value of C) Those parameters are saved inside a variable and used for the final application of the algorithm.

9.5.2 NN optimization

The neural network was optimized in the same way as the SVR. Parameters to be optimized in NN are the learning rate and the batch size. In addition, results from two different topologies of the network were compared: the original topology composed by three hidden layers, where first two are made of five neurons and the last one of three (5, 5, 3) , and a more articulated configuration composed by two layers of ten neurons and the last one composed by six (10, 10, 6), keeping same number of hidden layers fixed. To perform the optimization, learning rate test values were 10^{-4}, 10^{-3} and 10^{-2} while test batch sizes were 16, 32 and 64.

Code Listing 9.12: Grid search optimization for neural network

```
estimator = MLPRegressor(solver = "adam",
                         n_iter_no_change = 100,
                         max_iter = 250,
                         shuffle = False)

parameters = { 'hidden_layer_sizes':[(5,5,3), (10,10,6)],
               'learning_rate_init':[1e-4, 1e-3, 1e-2],
              'batch_size':[16, 32, 64]}

GS_NN = GridSearchCV(estimator,parameters,
                     scoring = "neg_mean_absolute_error", n_jobs = -1)
GS_NN = GS_NN.fit(X, y)
NN_best_param = GS_NN.best_params_
print(NN_best_param)
```

The optimization output was:

- *learning rate = 0.01*
- *batch size = 16*
- *topology = (10, 10, 6)*

It resulted that a smallest batch size performed better even if coupled with the highest learning rate. In this way the algorithm converges really fast towards the minimum of the cost function. The topology that performed best was the most articulated one, showing that an higher number of neurons per layer produces the best result in this case.

9.6 Selection of the best model

The same validation that has been done using non-optimized algorithms was repeated on algorithms configured with optimized parameters evaluated by the grid search: a K-fold cross-validation that splits the data set in 34 folds and repeats the testing process for each of them. Results from this validation, as mean and distribution of the normalized mean absolute error, are again presented in the form of a box-and-whisker plot in figure 9.8 and compared with results of the non optimized algorithms.

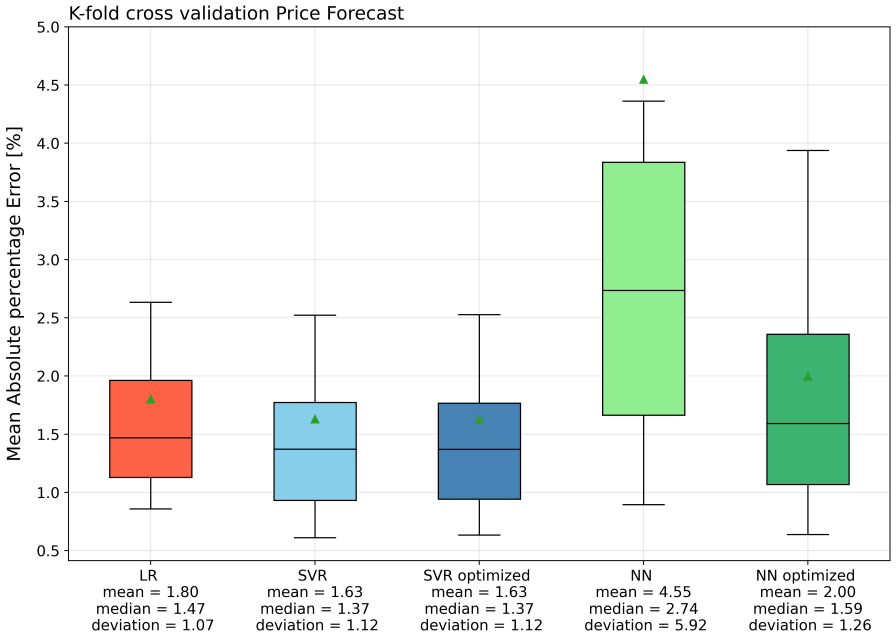


Figure 9.8: k-fold validation results of optimized algorithms for PV forecasting.

As it is shown in the plot, performances of the NN significantly improved, especially if the mean value of the absolute error committed by the algorithm is taken into account. Performances of the SVR remained the same, showing that even the default parameters were well suited for the prediction. Overall, the best results come from the SVR: even if the deviation is marginally higher than the one obtained with Linear Regression model, the mean error is smaller. Because of this, SVR was selected as the best method to perform predictions regarding PV power.

The best model among the three is stored in order to be used for the actual forecasting process. This operation was done using the *pickle* library and the best model with optimized parameters was saved in a *.sav* file.

Chapter 10

Electricity price forecasting

It this chapter, Machine Learning methods have been tested on the electricity price forecasting problem. The quantity that must be predicted now follows a less foreseeable trend with respect to the PV power production and the relationship that it has with its related quantities is different and, as it will be seen, not strongly linear as the previous case. The same three methods that has been used for the PV power production prediction have been tested and compared, again to assess which is the best one in terms of accuracy:

- Linear regression;
- Supported vector regression;
- Neural networks.

Similarly to photovoltaic case study, the two validation strategies employed for PV are here proposed again in this section:

- Hold-out validation: preliminary evaluation of the prediction performances using a model trained on the 80% of available data and tested on the remaining 20%.
- K-fold cross-validation: more expensive but detailed method for a better estimation of the general performance of the algorithms.

Algorithms have been tested with their default configuration and parameters, then developed thought a grid search optimization. The model selected at the end is used for predictions different proposed case studies.

10.1 Data sets for price forecasting

Algorithms were tested on a data set containing price and load evolution of electric power in the northern area of Italy, in a time span that goes from January 1st 2017 to December 31st 2019.

Price data set

Price time series was provided by the GME, acronym for *"Gestore dei mercati energetici"*, which is the society that manages the electricity market in Italy. Takinig into account that the aim of this section is to predict the variation of the zonal price (PZ) in euros per MWh, which is related to the electric energy sales offers, the hourly time series of the price in the northern geographical zone was selected among all the data provided by GME. Price data set for the algorithm was created by the concatenation of yearly records of the three considered years. For this purpose, the Pandas command *pd.concat* was utilized and missing data in the final data sets were replaced with the previous information using the command $fillna$ and the method of *"backfill"*, where a Not-A-Number (NaN) value is replaced with the previous one. This approach has been used due to the almost negligible quantity of missing terms, otherwise other proper data filling strategies should be adopted instead.

Code Listing 10.1: Price data set import

```
ds_price_2017 = pd.read_excel('2017 prices.xlsx',
                              sheet_name = "Prezzi-Prices",
                              usecols = "A:B,O")
ds_price_2018 = pd.read_excel('2018 prices.xlsx',
                              sheet_name = "Prezzi-Prices",
                              usecols = "A:B,O")
ds_price_2019 = pd.read_excel('2019 prices.xlsx',
                              sheet_name = "Prezzi-Prices",
                              usecols = "A:B,N")

ds_price = pd.concat([ds_price_2017,ds_price_2018,ds_price_2019],
                     ignore_index=True)

ds_price.columns = ['DATA','ORA','price']

ds_price.isna().sum()
ds_price.fillna(method = "backfill", inplace = True)
```

Load data set

Load time series was also referred to the north of Italy and provided by `Terna`, the society that manages the Italian transmission grid. The total electric load time series provides the load evolution in GW on a 15-minute basis, so it was averaged on an hourly basis and reported in GWh In order to match the Zonal Price time resolution. The load data set was created in the same way as for the price one. For the re-sample of time it was used the `Pandas` command *pd.resample*, specifying a time frame of 1 hour. By using the command *.mean*(), the hourly value is computed on the average value of the particular hour.

Code Listing 10.2: Load data set import

```
ds_load_2017=pd.read_excel('2017 total load.xlsx',usecols = 'A:B',
                           skiprows = [0,1])
ds_load_2018=pd.read_excel('2018 total load.xlsx',usecols = 'A:B',
                           skiprows = [0,1])
ds_load_2019=pd.read_excel('2019 total load.xlsx',usecols = 'A:B',
                           skiprows = [0,1])

ds_load = pd.concat([ds_load_2017,ds_load_2018,ds_load_2019])
ds_load.columns = ['date','load']
ds_load = ds_load.set_index('date')
load = pd.Series(ds_load['load'].resample('1H').mean().values,
       name = 'load')

load.isna().sum()
load.fillna(method = "backfill", inplace = True)
```

Price and load data set

After the input of price and load data sets and re-sampling on the same time resolution of 1 hour, the final dataset to be used by the predictor model is here created. For indexing purposes, a date time series is created using the `Pandas` command *pd.Series*, setting January 1st 2017 as initial time instant and December 31st 2019 as final time instant.

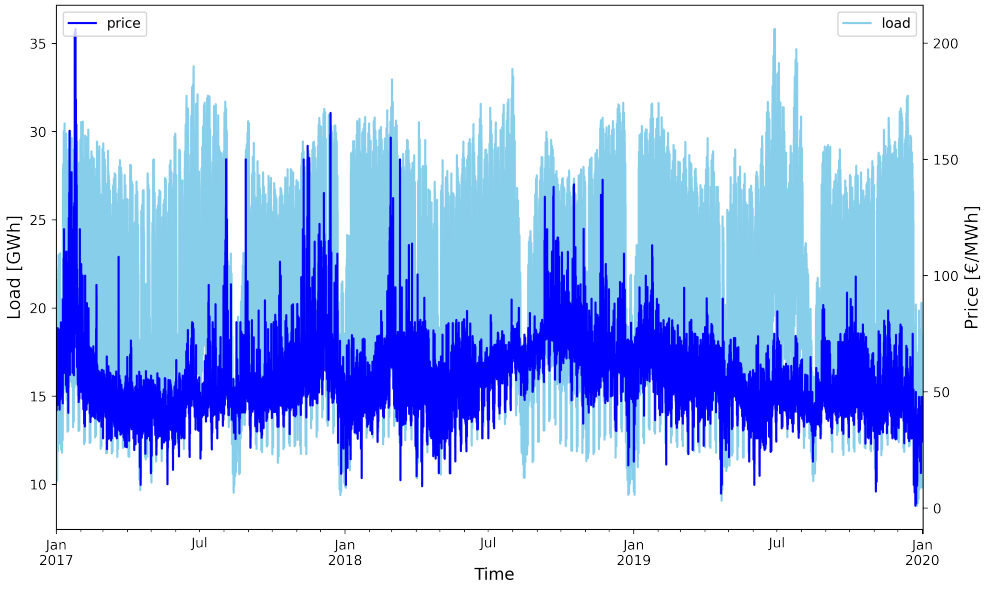


Figure 10.1: Time series trend for price forecasting.

Code Listing 10.3: Price and load data set creation

```
d0 = "01-01-2017 00:00:00"
de = "01-01-2020 23:59:59"
date = pd.Series(pd.date_range(d0, de, freq = '1H'), name = 'date')
df = pd.concat([date, load, ds_price], axis = 1)
                .drop(columns = ["DATA", "ORA"])
                .set_index('date').asfreq("1H")
```

In figure 10.1 it can be seen a graphical representation of the final data set. Price trend is represented in blue and load's one in light blue. It can be noticed that the two time series do not follow the same evolution: load appears to be more cyclical and regular while price is subjected to frequent peaks and irregular variations.

The Pearson correlation coefficient relative to the correlation between price and load has been computed in order to assess the strength of the linear relationship between the two quantities. Results are reported in form of a heat map in figure 10.2, where lighter colours are representative of stronger correlations.

Figure 10.2: Pearson correlation coefficient between price forecasting data sets.

The correlation coefficient obtained between price and load is 0.56. This positive value means that when load increases, price sees an increment too, but the correlation is not in a strictly linear manner since is considerably lower than 1. Since the correlation with the price at a time t and the load at the same time instant is not that much dominant, it was necessary to find another quantity that could provide a better base upon which the algorithm could perform the forecast. As it was anticipated in the introductory section of this work, for this purpose it was exploited the correlation with past values of the price itself, in order to perform its prediction by considering its past values and trends. In this perspective, the time series should be decomposed appropriately so that past values can be extracted and treated as new features to be added in problem description (i.e. add columns to the price dataset). This process allows to obtain as output a "lagged" data series dataset.

Lagged time series

Lagged data sets must be obtained to exploit the correlation between the electricity price that must be predicted. Giving that current regulations foreseen that Italian Energy Market prices (both PUN and Zonal Prices) are established one day in advance with respect to the hour it is then sold or purchased, it can be assumed that no information for the price are available until the 24 previous hours. Following this assumption, new feature are extracted starting by price at the same hour of the day, week and month before. This was achieved by shifting the price data set of 24, 168 and 672 hours. This was performed using a function that takes the data set and the number of hours by which it must be lagged and returns a data set containing both the original time series and the lagged ones.

Code Listing 10.4: Lagging data set funcion

```
lags_pre = [0,24,168,672]
price = lagged_dataframe(lags_pre, lags_post, df)

def lagged_dataframe(lags_pre, lags_post, df):
    features = list(df.columns)
    n_features = len(features)
    vals, names = list(), list()

    # previous lags : t-1,...,t-lags_pre
    for i in range(len(lags_pre)):
        if i == 0:
            names += (['%s(t)' % (features[k]) for k in range(n_features)])
            vals.append(df.shift(lags_pre[i]))
        else:
            names += (['%s(t-%i)' % (features[k], lags_pre[i])
                       for k in range(n_features)])
            vals.append(df.shift(lags_pre[i]))

    df_lags = pd.concat(vals, axis = 1)
    df_lags.columns = names
    df_lags.dropna(inplace = True)
    return df_lags
```

The Pearson correlation coefficient between price and past records of it is computed in order to estimate the potentiality of this method. In the heat-map reported in figure 10.3 it is possible to see that a strong relation exists between the price at time t and its past record at the day before $t - 24$. The correlation is still strong with records of the week before $t - 168$ and a slightly lower value is observed for the month before $t - 672$.

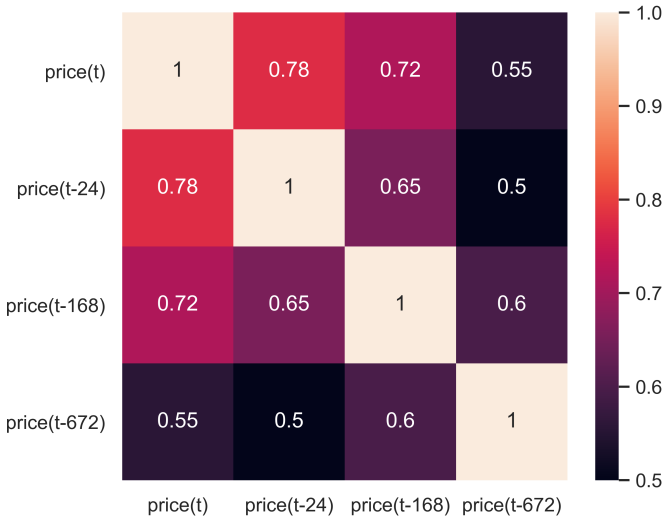


Figure 10.3: Pearson correlation coefficient between price forecasting data sets.

10.2 Hold-out validation

A first estimation of the accuracy of the algorithm can be given by a hold-out testing method using 80% of samples as training set and the remaining 20% as test set. The same procedure described in the PV power forecasting section was applied.

In figures 10.4 and 10.5 the plots for a graphical comparison between the prediction and actual values are reported, obtained during a week belonging to the testing period using Linear Regression and Support Vector Regression. As is was expected, since there is not a related quantity that shows a very high correlation coefficient, now the forecast (blue line) from linear models (LR and SVR) cannot follow with the same degree of accuracy the real trend (orange line) as it did it the case of prediction of PV power. Unexpected peaks and wide variations happens more frequently among the price trend, and in those sections the accuracy of the forecasting inevitably decreases. Even if a non-linear approximator as a Neural Network is used, whose prediction plot is shown in figure 10.6, it is not possible to produce a forecast matching correctly the actual price evolution when unexpected variations happen.

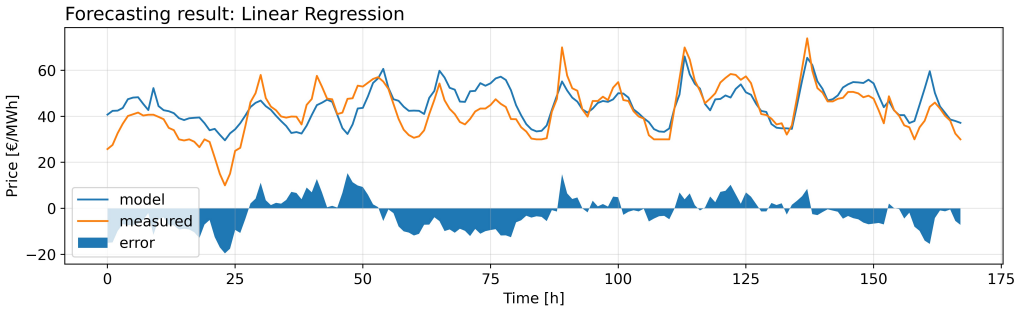


Figure 10.4: Model vs measured plot for price forecasting using LR.

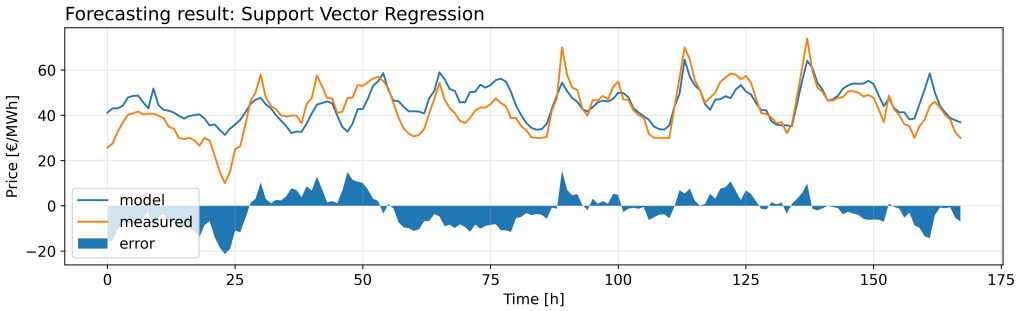


Figure 10.5: Model vs measured plot for price forecasting using SVR.

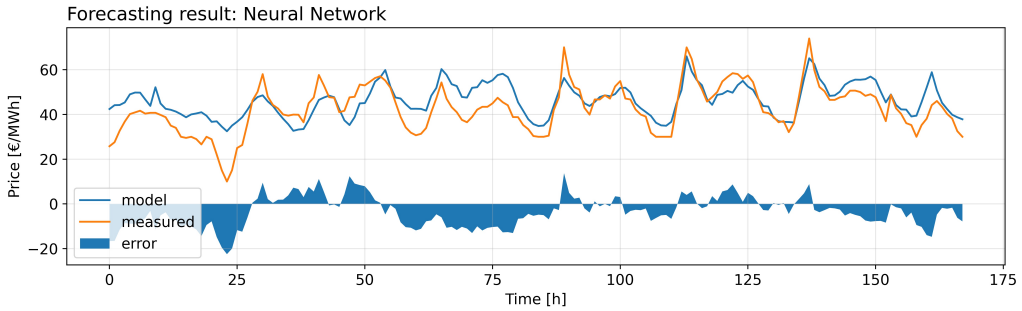
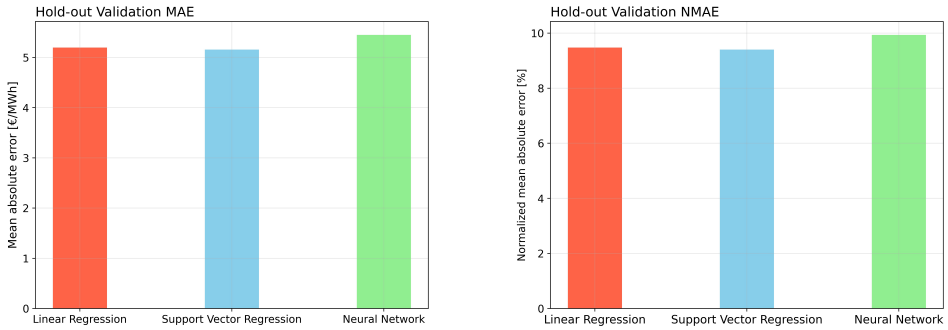


Figure 10.6: Model vs measured plot for price forecasting using NN.

10.2.1 Normalized mean absolute error

As first numerical evaluation of the performances of the three algorithms is assessed through the evaluation of the mean absolute error on the test and reported in figure 10.7 (a). The best results come from linear regression and support vector regression, which show a very similar accuracy. To obtain a more generic metric, the mean absolute error is normalized with respect to the average energy price that has been registered in the time frame: $mean\_price = np.mean(y)$. Results are shown in a bar plot reported in figure 10.7 (b).



(a) MAE

(b) NMAE

Figure 10.7: Price forecasting mean absolute error and normalized mean absolute error.

In this application the NMAE has higher values, in the order of magnitude of 10%, due to the more complex target of the prediction and the lower linear correlation with predictor variables.

10.3 Optimization of parameters

Before the application of the final cross-validation, parameters of support vector regression and neural network were again optimized by means of a grid search optimization.

10.3.1 SVR optimization

Parameters to be optimized in support vector regression are C and ϵ. C in set by default at 1.0 and in the optimization it was tested in the range between 1.0 and 10, divided in 30 intervals of the same length. The maximum error ϵ is 0.0 by default and tested in the range between 0.0 and 10. The optimization output was:

- *C = 9.526*
- *epsilon = 0.526*

Results are much different with respect to what was obtained for the PV power forecast (C = 1.62 and epsilon = 0.0). Involved time series are much different with respect to the previous case, and the model fits its coefficients using different values for the hyperparameters. The correlation between price and predictor quantities is not as linear as it was between PV power and solar radiation. This means that the model must have a higher tolerance to points that fall outside the fitted model.

10.3.2 NN optimization

Parameters to be optimized in neural networks are the learning rate and the batch size of the algorithm. In addition, two different topologies of the network were compared: the original topology is composed by three hidden layers with sixteen, eight and four neurons respectively (final architecture 16-8-4), and the same topology with twice the number of neurons: (32, 16, 8). Tested architectures are more complex with respect to what was tested in the previous case of PV power production since this time there is a more complex trend to predict and a strong linear correlation between features and target can't be observed. Learning rate test values were chosen as 10^{-4}, 10^{-3} and 10^{-2}. The optimization output was:

- *learning rate = 0.0001*
- *batch size = 16*
- *topology = (32, 16, 8)*

The best topology is still the most complex one. This time the small batch size is coupled with the slowest learning rate.

10.4 k-fold cross-validation

The k-fold cross-validation gives a better overview of the performance of the algorithms since it tests them on the entire data set by sequentially changing both training and test set. Again, a number of folds that matching the number of months of the data set employed has been selected, so that the model is tested on each month present in the time frame and the metric selected was again the MAE.

In Figure 10.8 are reported the results of this validation, applied both to the algorithms with the default configuration and with optimized parameters.

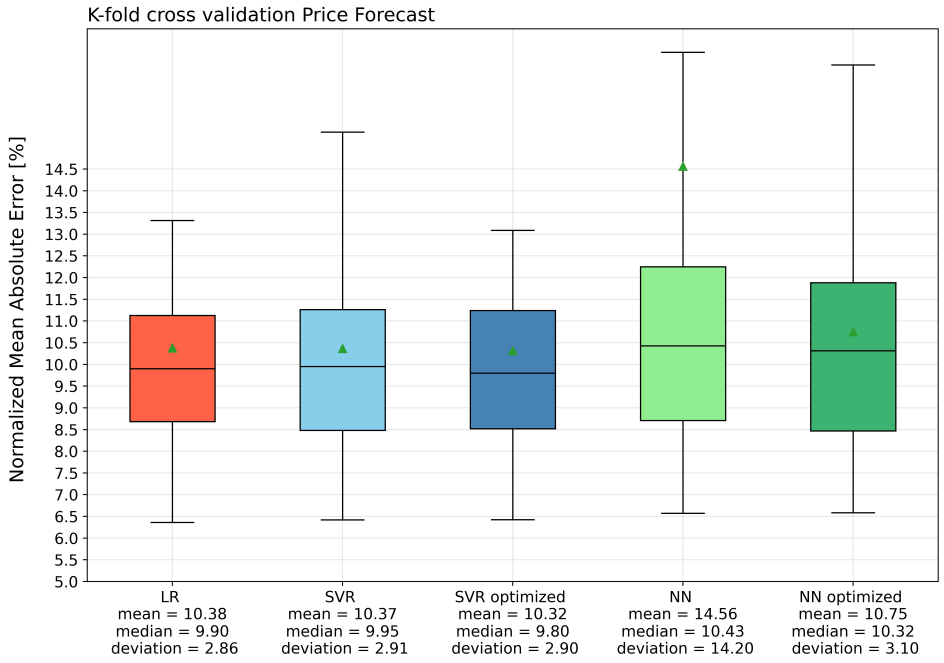


Figure 10.8: k-fold validation results for price forecasting.

Another time, the best result in terms of NMAE comes from Support Vector Regression, showing the lowest mean value of the error and amongst the smallest deviation. The grid search optimization improved the mean value of the NMAE from 10.37% to 10.32% and the deviation from 2.91% to 2.90%. Neural networks also improved after the optimization of parameters but still shows an higher mean error and a larger deviation. For those reasons, the model that was employed at the end in the case study is again the SVR with optimized parameters.

10.5 Influence of external temperature on price forecasting

In this section, the external temperature effect of a location in the considered price area was tested, adding another external predictor variable to the forecasting algorithm. It was employed a temperature time variation by `Arpa` related to the location of *Fossano (CN), Italy.*

Temperature trend (figure 10.9) can influence on the fluctuation of price in the electricity market since it bring to electric load variations: hot days which are typically observed in summer season will see an increase in air conditioning systems usage and an increase of cooling loads, and consequently an increase in the electricity consumption. The Pearson correlation coefficient (figure 10.10) between load and temperature is positive but low, meaning that an increase in temperature may lead to a week and non linear increase in load. However, the correlation between temperature and price is characterized by a low and negative coefficient. As it is expected, the inclusion of temperature time series does not produce a significant increase on the forecasting performances, as it is shown in figure 10.11. But considering the introduction of the temperature as describing feature leads to a small but positive model performance improvement, it can be used as a feature to describe electricity price evolution.

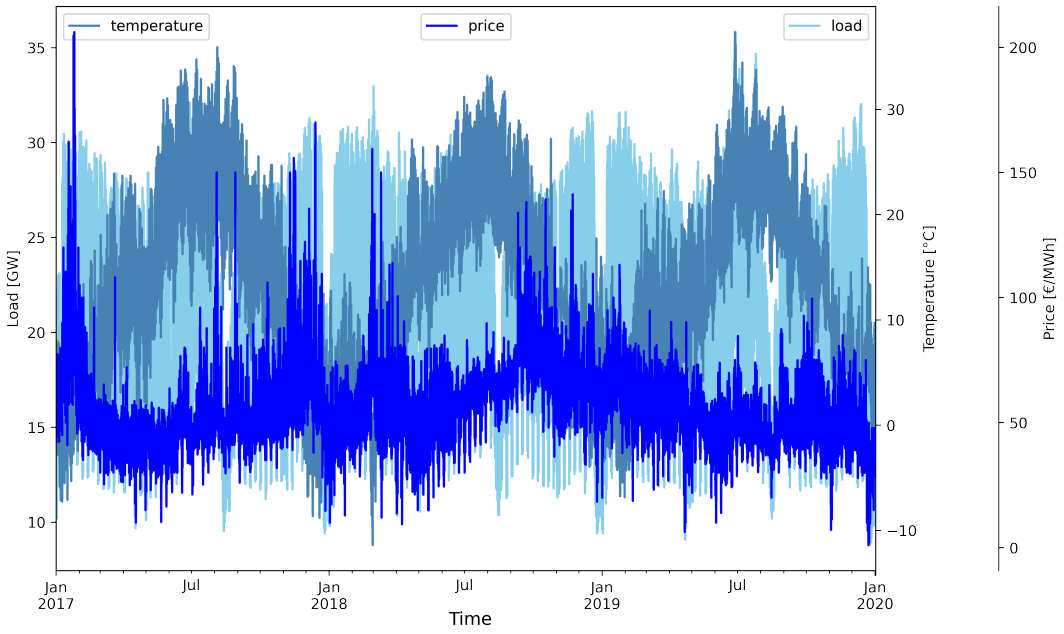


Figure 10.9: Price, temperature and load time series.

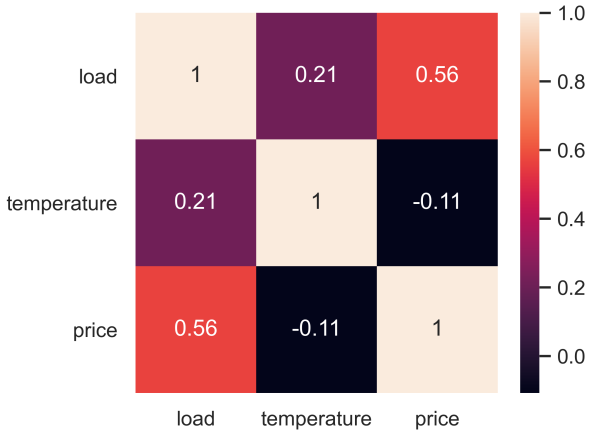


Figure 10.10: Price, temperature and load Pearson correlation coefficient.

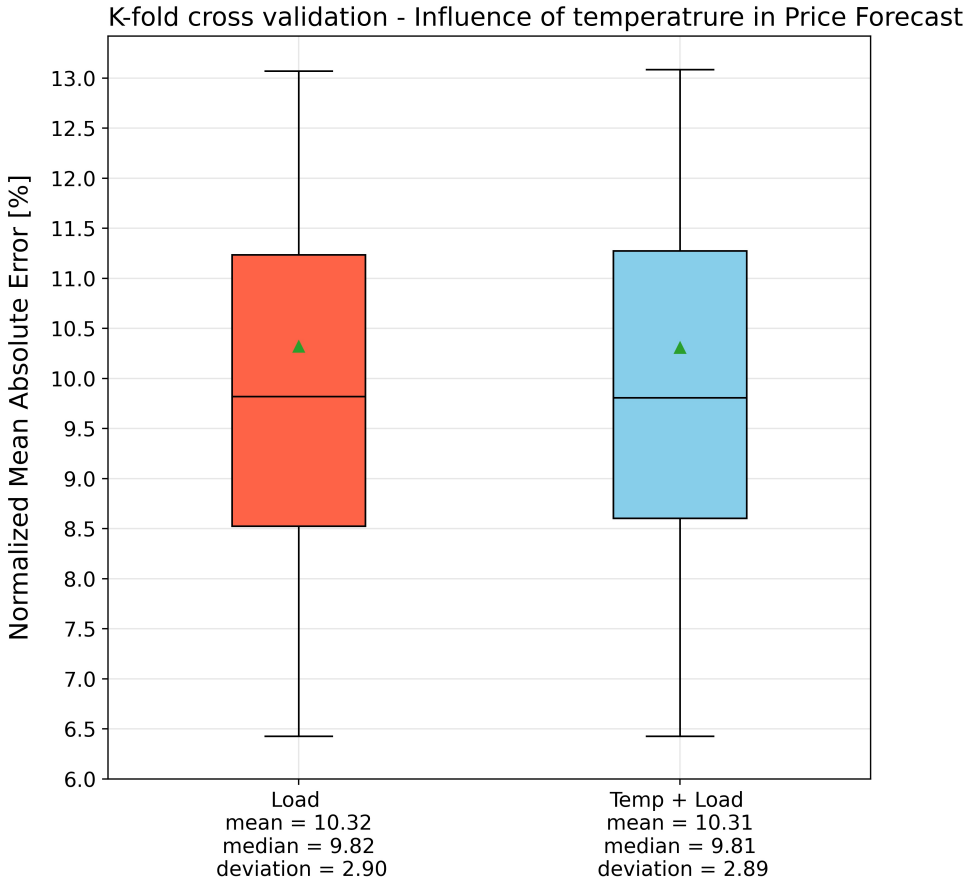


Figure 10.11: k-fold validation of price forecasting including temperature time series.

Part V

Case studies

Chapter 11

Forecasting power and price in a typical month

Support vector regression (SVR) was the prediction model that produced the best results in terms of accuracy for both PV power forecasting and electricity price forecasting. After being optimized for those kind of predictions, it has also been possible to obtain the its best configuration in terms of parameters. By using the *pickle* python library, it was possible to save the model with optimized parameters for further applications. In many applications, this step could save time because there is no further need to train the algorithm, but its parameters are already well suited for the application. In this chapter the results of a typical application of this algorithm are reported: predictions of the PV power and electricity price during a typical month. May 2021 was considered as a suitable and recent time period to be used as an example.

A fist prediction approach consisted in the overall month prediction: this time the algorithm was trained on a data set starting from January 1st 2020 up to April 31st 2021 and applied in the period from May 1st to May 31st 2021. A second approach followed the strategy of day by day prediction, performing the forecast 24 hour at a time and moving the training window for each iteration.

Data

Regarding the PV power prediction, the same data set provided by `EGEA`, that was used for the optimization of the algorithms, it was employed. The difference is that only the portion referred to the time period described before was taken in consideration this time. Their trend is shown in figure 11.2. The prediction of power is again based on the correlation with the solar radiation and modules temperature. Regarding the price forecasting, data of 2020 and up to May 2021 has again been provided by `GME`. In the same way, load data referred to this time span has been used taking advantage of `Terna` availability of open-source data. Also the influence of temperature has been taken into account for price prediction task, using again the temperature data sets provided by `Arpa`. The improvement that the inclusion of this last predictor variable produced was marginal but nevertheless positive. Time series employed for the price prediction are reported in figure 11.2.

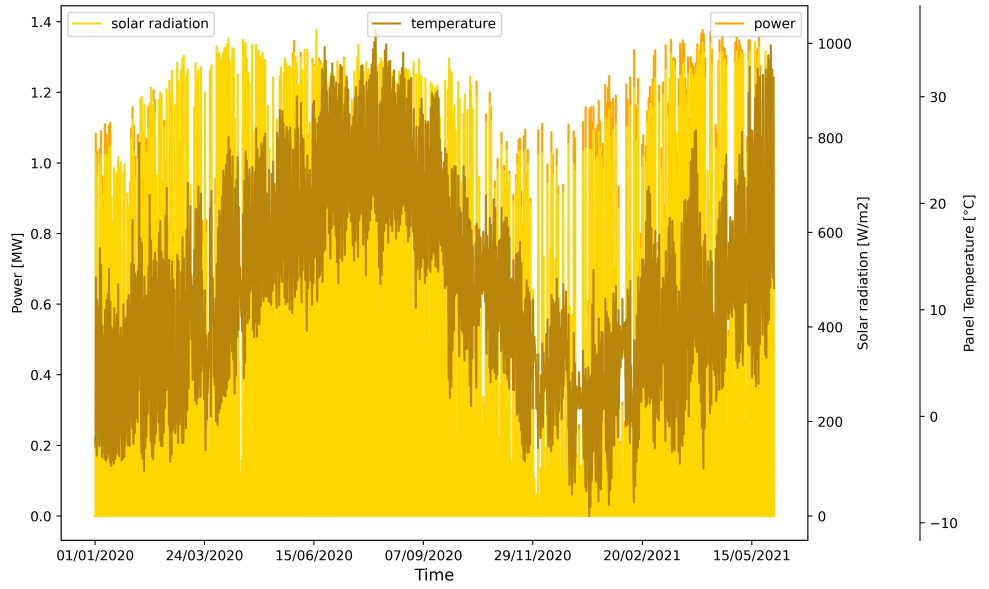


Figure 11.1: PV time series for May 2021 prediction.

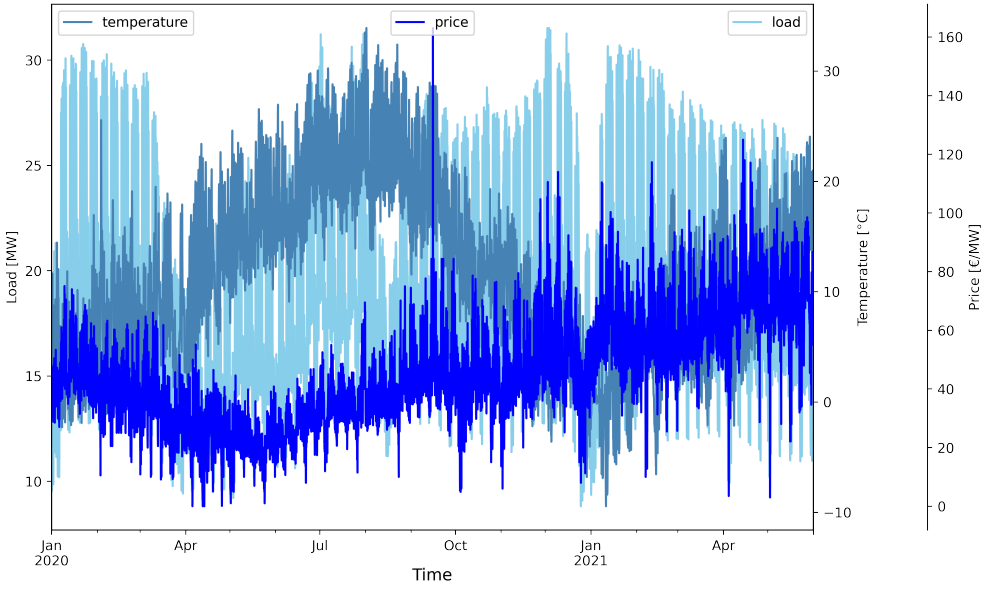


Figure 11.2: Price time series for May 2021 prediction.

It is interesting to notice that, during the month of May 2021, the price that must be predicted was characterized by a mean value and a standard deviation of:

- Mean value: 69.61 €/MWh;
- Deviation: 14.11 €/MWh;

Results

Results of the power prediction are shown in figure 11.3. In this application, the prediction (blue line) follows relatively good the real trend of the power production (orange line). In the plot it is visible that the error between the predicted trend and the actual one is low, especially during the last days of the month. In addition, it seems that most of the observed errors are located in the central hours of the day where both solar radiation and photovoltaic power peaks are expected.

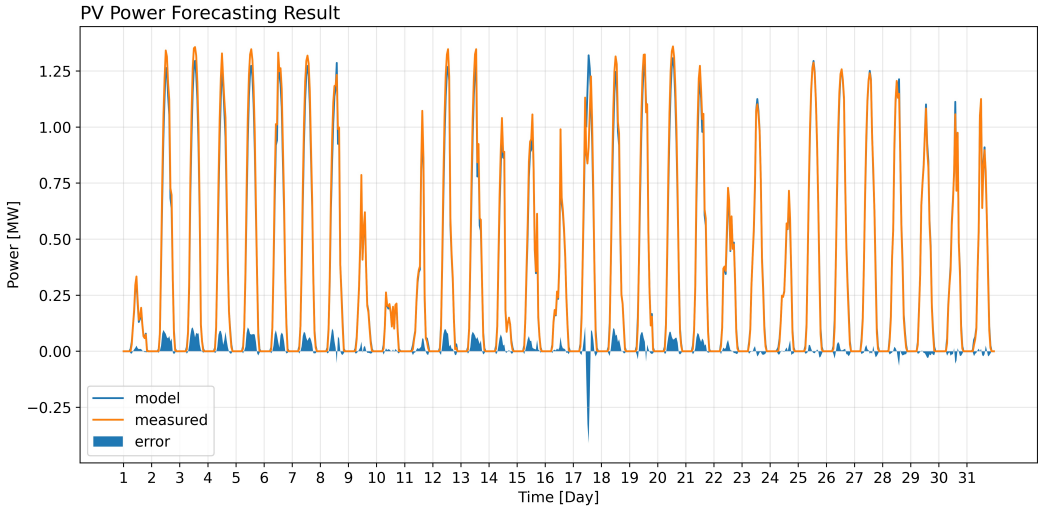


Figure 11.3: Prediction of power for May 2021.

Actual and predicted price values are shown in figure 11.4. Here the prediction succeed in capturing the overall weekly trend of the price variation but the difference between the two lines is higher compared with the previous PV power prediction case study. The forecasting algorithm was able to follow small variations but, in presence of high peaks and abrupt variations, the accuracy of the prediction was considerably lower and the error in those points is consistently higher. SVR has been used as a regressive model that benefits from linear relationships between data. It succeeds very well in PV power forecasting since there is a strong linear relationship between the power production and the solar radiation (characterized by a Pearson correlation coefficient almost equal to 1). But in the case of price prediction, there is not such a linear correlation with predictor variables with the same emphasis as the photovoltaic case. SVR produces the best result but fails while catching other kind of relations between data, and because of this the prediction error is inevitably higher.

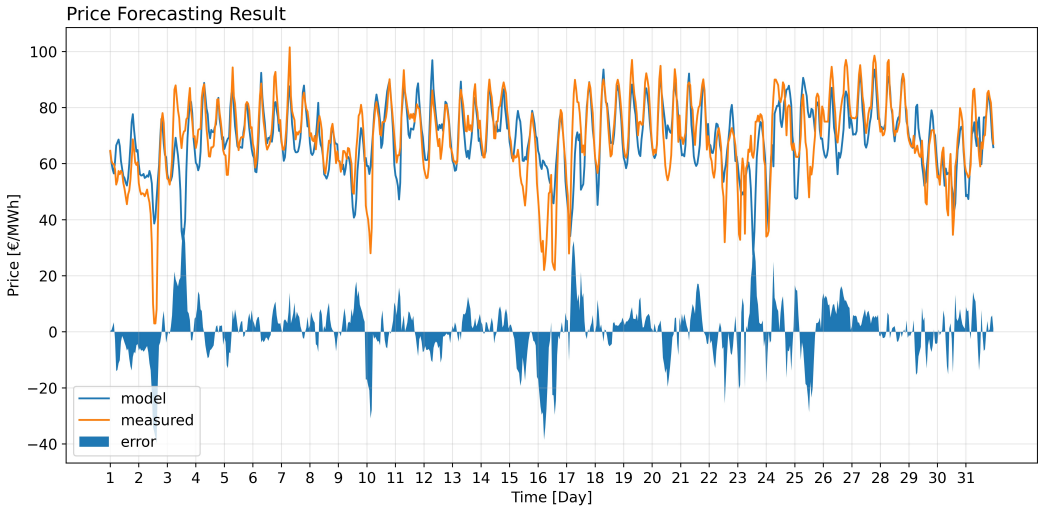


Figure 11.4: Prediction of price for May 2021.

The error committed for the power prediction was normalized with respect to the maximum power produced by the panel registered in the data set, while the error committed for the price prediction was normalized with respect to the mean price that characterized the month of May 2021.

Normalized mean absolute errors:

- PV power NMAE = 1.31% (normalized on maximum power);
- Price NMAE = 10.71% (normalized on mean price);

The comparison between those two metrics is not straightforward since the errors are normalized with respect to two different quantities. PV power has a cyclic variation between zero production at night and peak of production in the middle of the day. If the error would be normalized on the mean value of power, the metric would assume an unreasonably higher value since it would consider also the night periods of no production. This would not reflect the actual quality of the prediction. On the other hand, price has an irregular trend. If the error on price would be normalized on the maximum value registered in the selected time period, the metric would assume a much lower value that cannot reflect correctly the accuracy of the prediction and the result would be misleading. One way to compare the two may be the observation of the predicted and actual trends on figures 11.3 and 11.4. It can be said that the PV prediction has an higher accuracy since it better follows the real evolution of the quantity of interest.

11.1 Moving training window

Following last section results, PV power forecasting is very effective if a correct solar radiation prediction is fed to the algorithm. Since it exists such a strong linear correlation with this external variable, a linear regressive model applied as shown before produce predictions characterized by a consistently good accuracy.

Regarding the electricity price forecasting, another approach was tested and compared with the one that has been shown before. Since the price prediction is based on the correlation between the price at a time t and its past values, it is expected a tangible variation of the algorithm accuracy if the time span of the data set used for the training is modified. Price has a very irregular trend and is affected by various external factors, whose influence have a complex effects. It may be not effective to train a prediction algorithm on a very wide time history, since the future trend of price is affected by new or more recent events and situations with respect with the past. In fact, it has been seen that energy price are in general subjected to a daily and weekly pattern, which put in light the needs to "follow" short-term trend variations while predicting this quantity. In this specific case, data sets considered in this application are referred to years 2020 and 2021. Due to the COVID-19 pandemic that characterized those years, the electricity price has seen a significant variation along the time period in question. In the previous section, it was described a prediction model that trained the algorithm using a data set starting from the beginning of the year 2020, up to the end of April 2021. Then the algorithm was used to perform the prediction of the price trend along the whole month of May 2021. Following the approach presented in this section, it has been explored the possibility to perform the forecast 24 hours at a time, using an algorithm trained on a data set referred only to certain time span before that specific day.

Code Listing 11.1: For cycle day-by-day price prediction - 1 month training window

```
prediction = []
actual = []

for i in range(0,32):
    start = len(price) - (2*31*24 - 24*i)
    end = len(price) - (31*24 - 24*i)
    price_red = price.iloc[start:end]
    X_price = price_red.drop(columns = ['price(t)']).values
    y_price = price_red["price(t)"].values
    m_price = X_price.shape[0]
    nx_price = X_price.shape[1]
    model_set_price = linear_model_scaling(X_price, y_price, m_price)
    pred_i, act_i, _ = loaded_model(show_plots, model_set_price,
                                    loaded_model_price)
    prediction.extend(pred_i)
    actual.extend(act_i)
```

In this case, the training window changes each time that is performed the prediction for a different day of the month, so it is possible to say that this method uses a moving training window, while the previous one used a "static" training window. In fact, each time a prediction is performed, the training set is updated with the most recent 24 values

but keeping the same window size. Than means that the day at the beginning of the train set is removed each time so that newest daily values can be added at the end.

Several attempts were carried out in order to asses which can be the best time period to be used in order to have the best accuracy for the electricity price prediction, applying the day-by-day forecasting technique. The best number of months to be used for the prediction was obtained after testing the algorithm using a training window equal to 1 to 12 previous months.

This method was again employed to perform the price prediction for the month of May 2021 in order to compare results with the previous approach and the method used to assess the accuracy of the prediction is again the normalized mean absolute error.

Code Listing 11.2: NMAE calculation for day-by-day price forecasting

```
moving_MAE = metrics.mean_absolute_error(actual, prediction)
```

NMAE of the prediction obtained using different training window is reported in a plot in figure 11.5

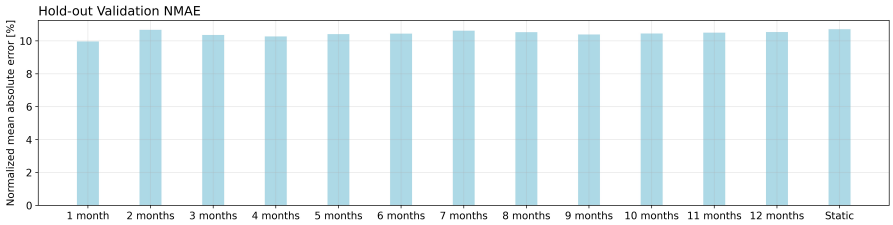


Figure 11.5: Normalized mean absolute error from different training windows.

As it can be seen in the plot, there is a substantial improvement in accuracy from the first static method (last bar). The best result comes from the 24 hour prediction based on 1 month of previous data, this predicted trend is reported in figure 11.6. Comparing this error with the result obtained with the previous method:

- *NMAE from 1 month moving training window = 9.96%*
- *NMAE from static training window = 10.71%*

Training this kind of algorithm on a restricted moving window improved the accuracy of 0.75%. This is beneficial not only for the accuracy of the model, but also shows that machine learning methods do not require huge data set to be trained on. Using small training windows makes the harvesting of historical data easier and lightens the computational requirements of the software.

By observing the plot that reports the actual and predicted trend of the price during the whole month in figure 11.6, it can be noticed that the main sources of error are the abrupt variation peaks that happen among this time period. The forecasting algorithm succeeds in capturing the evolution during more regular periods (e.g. May 6th to May 9th), but it is not able to predict a sudden drop of price (e.g. May 3rd). This peaks are extremely adverse to good performance since it is visible that the algorithm requires some time to "re-adjust" after one of those. As it was already pointed out in the previous section, SVR has a regressive nature that is not able to capture all the non linear and complex links between electricity price and its predictor variables.

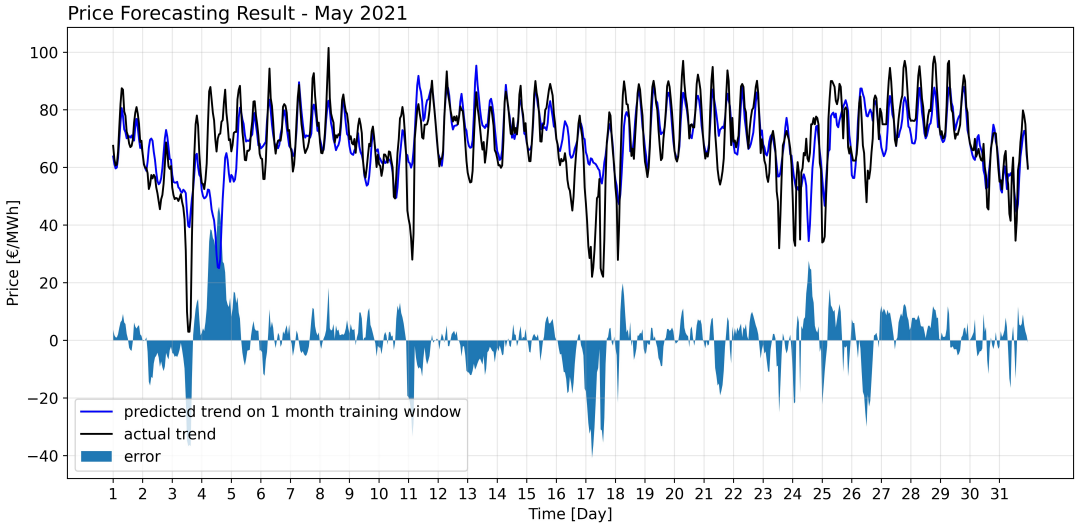


Figure 11.6: Predicted trend of price May 2021 with 1 month training window.

Chapter 12

Comparison with previous years

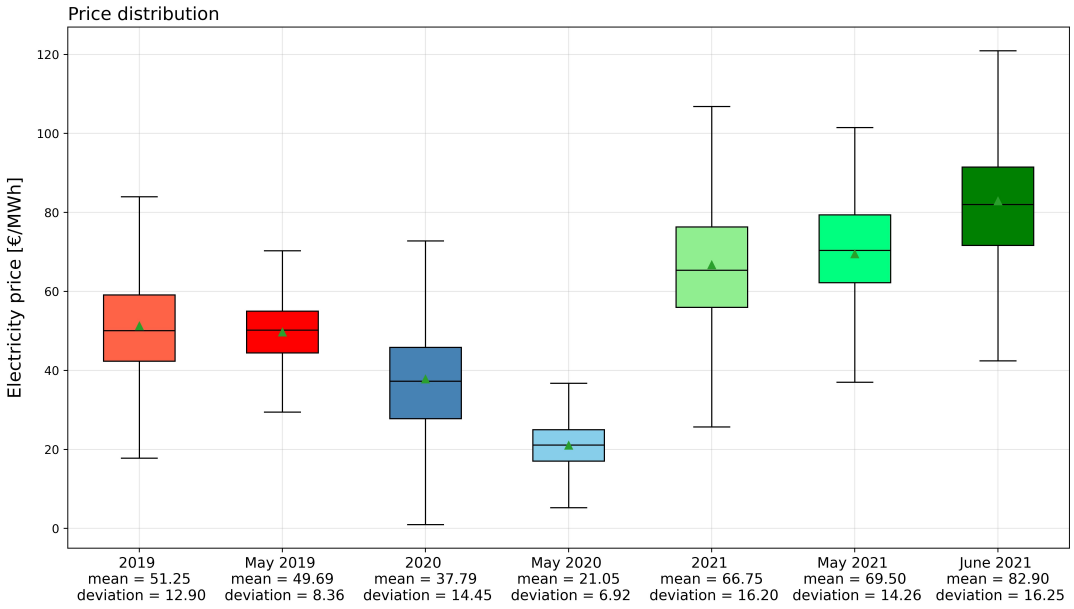


Figure 12.1: Price distribution across 2019, 2020 and 2021.

2020 was a peculiar year from an economic point of view due to the COVID-19 pandemic. For this reason, in this section the same sliding window procedure described before was applied to May 2019 and May 2020 to compare results coming from the same application of prediction algorithms in different market situations. Also in this case, predictions performed with both static and moving training window are compared. Tested time spans used for the second approach ranged again between 1 to 12 months before the 24 hours predicted one by one.

Expected outcomes:

- 2019 was a more regular year and represents an unperturbed situation before the pandemic. It is expected the best result from a prediction algorithm since the price trend was not subjected to an additional degree of complexity added by the virus;
- 2020 was the first year affected by Coronavirus pandemic. Electric demand dropped of almost 7% with respect to the previous year and prices dropped down dramatically during spring 2020 [54]. The month of May 2020, presents the price evolution immediately after the first lockdown that affected Italy. It is expected the worst result in terms of accuracy by the prediction algorithm since that month was characterized by a very peculiar situation;
- 2021 is the year after the pandemic. It was characterized by a second period of restrictions that had dramatic health effects. However, the electric market and electricity consumption were considerably less affected. Consumes increased again reaching back 2019 levels and the electricity price have seen a 40% increase in six months, reaching an higher level than 2019 [54]. Nevertheless, the situation still is characterized by an high level of uncertainty. Is expected that the accuracy obtained before is in between what is obtained for 2019 and 2020.

In figure 12.1, price distributions across 2019, 2020 and 2021 are represented by mean of box plots. Years 2019, 2020 and 2021 were characterized by price trends with considerably different mean values and deviations. The metric that was used until now to evaluate the performances of different methods, the normalized mean absolute error, would produce trivial results since it would be normalized with respect to very different values in different periods. For the sake of the comparison, another metric was employed to evaluate the performance of the algorithms applied to different years: the mean absolute percentage error (MAPE).

Code Listing 12.1: Mean absolute percentage error (MAPE)

```
from sklearn.metrics import mean_absolute_percentage_error

MAPE = metrics.mean_absolute_percentage_error(actual, prediction)
```

12.1 May 2019

The price trend during 2019 was considerably different from the one observed in 2021. In figure 12.2 is reported the price trend during this year, in red is highlighted the price trend during the month of May, which is the target of this next forecast.

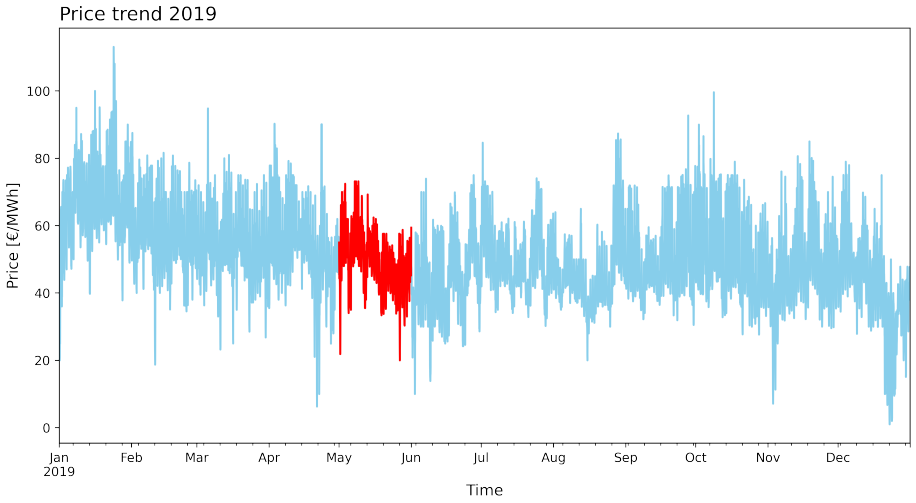


Figure 12.2: Price trend during 2019.

The overall price trend during this year was characterized by:

- Mean value: 51.25 €/MWh;
- Deviation: 12.91 €/MWh;

While, during the month of May:

- Mean value: 49.69 €/MWh;
- Deviation: 8.36 €/MWh;

Regarding the prediction of the price trend for May 2019, the static training window prediction was trained on a time span from January 1st 2018 to April 31st 2019. Mean absolute percentage errors coming from different approaches and trials are reported on a plot in figure 12.3

The best result comes from the moving training window prediction using data coming from 5 months before the 24 hour prediction. The predicted trend obtained by using this approach, compared with the actual one, is reported in figure 12.4

As it was for May 2021, the moving training window approach produced better results with respect to the static one:

- *MAPE from 5 months moving training window = 8.91%*
- *MAPE from static training window = 9.88%*

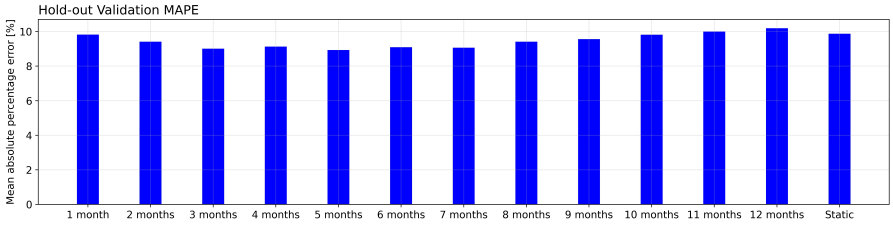


Figure 12.3: Mean absolute percentage error from different training windows May 2019.

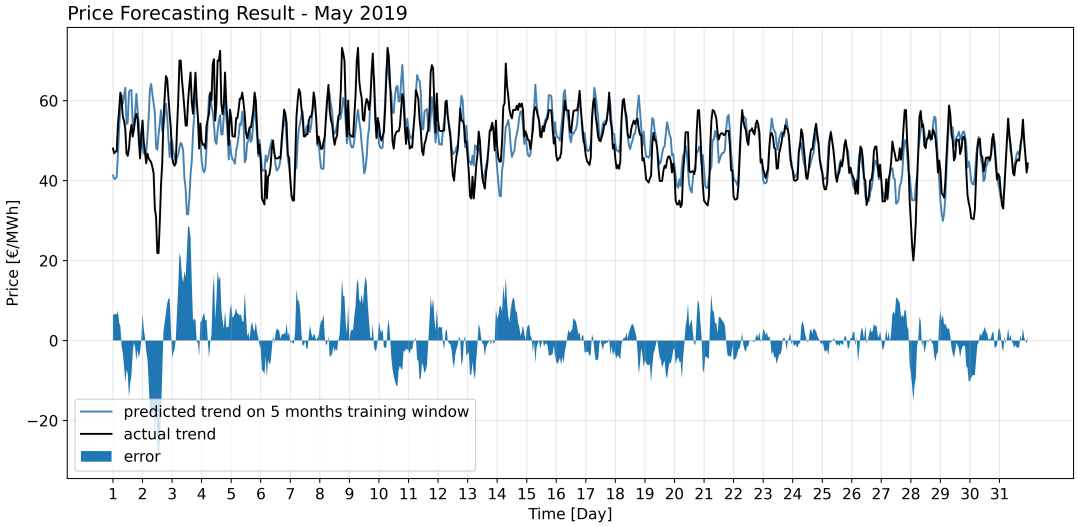


Figure 12.4: Predicted trend of price May 2019 with 5 months training window.

2019 can be considered as a standard year, since it was not affected by the pandemic situation that happened after it. The mean value of the electricity price that characterized the whole year was very close to the one of the month of May, in which the price trend was predicted. This results in a satisfactory forecast, identified by a MAPE lower than 9%. This value is comparable with values found in literature, referred to results of algorithms that have been applied to ”normal” price trends that characterized the market before the pandemic. Comparison with literature results is developed in a following section.

12.2 May 2020

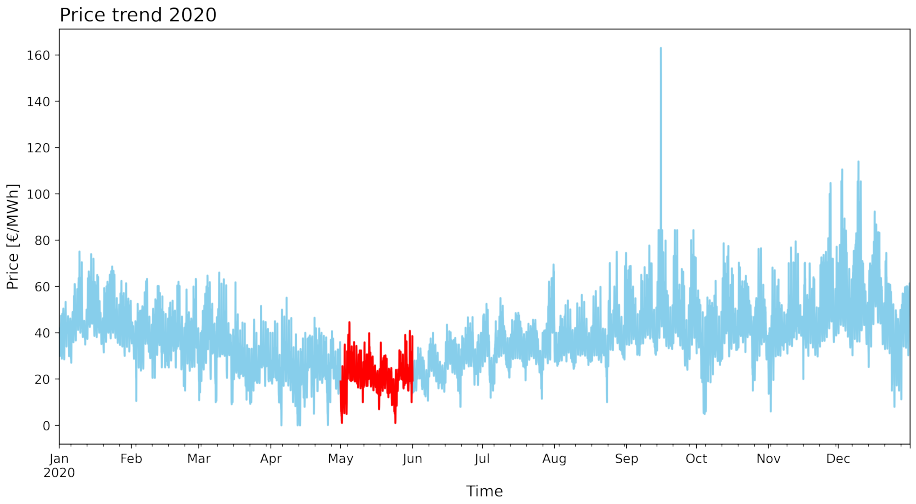


Figure 12.5: Price trend during 2020.

Also for the prediction of the price trend of May 2020, where the adopted approach was the same as before. Price trend of 2020 is reported in figure 12.5 and characteristics of this series are:

- Mean value: 37.79 €/MWh;
- Deviation: 14.45 €/MWh;

For the month of May (highlighted in red in figure 12.5):

- Mean value: 21.05 €/MWh;
- Deviation: 6.93 €/MWh;

MAPEs coming from different trials are reported in figure 12.6. Best results from different methods:

- *MAPE from 3 months moving training window = 31.52%*
- *MAPE from static training window = 33.92%*

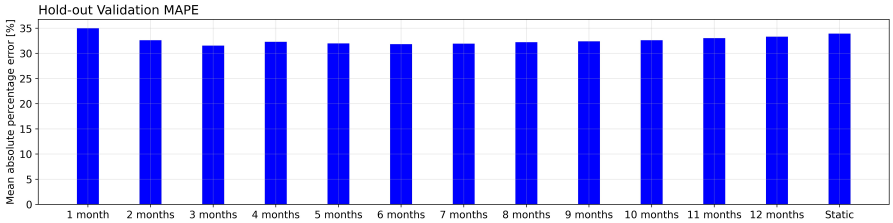


Figure 12.6: Mean absolute percentage error from different training windows May 2020.

A comparison between the predicted and the actual trend is reported in figure 12.7.

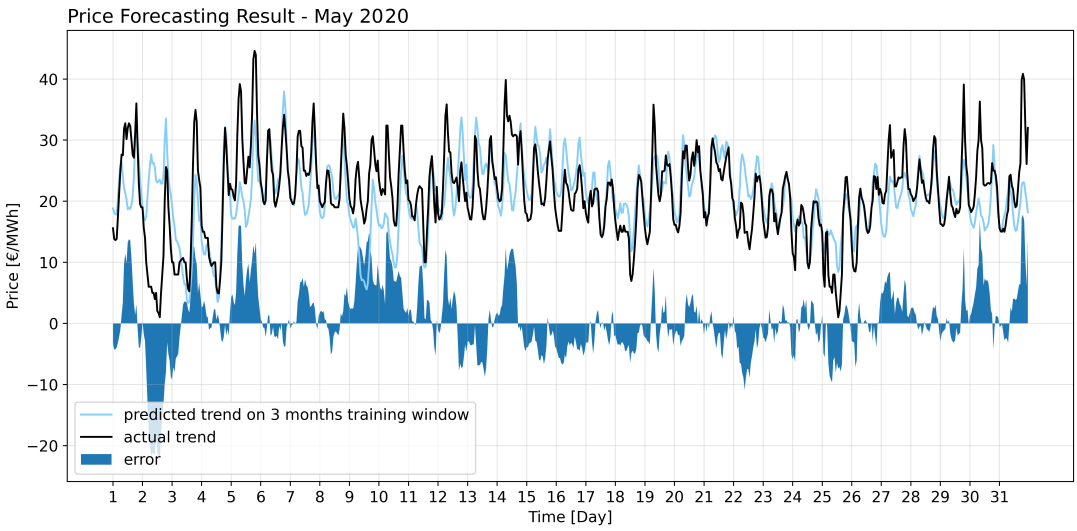


Figure 12.7: Predicted trend of price May 2020 with 3 months training window.

2020 was the year that have seen the most unpredictable price variation. Mean price during the month of May was considerably lower than the average across the whole year (21.05 €/MWh vs 37.79 €/MWh). Also the deviation across the whole year was higher than the one characterizing the price trend in 2019. This resulted in a very high MAPE, in the best case still higher than 30%. The magnitude of mean absolute error itself in €/MWh was comparable with the one obtained for the other two years, but the lower price means that this error is committed on a smaller value. This inevitably means that the error in percentage is considerably higher.

12.3 May 2021

Due to the limited availability of data, the price trend of 2021 up to the end of June is reported in figure 12.8 with the trend during the month of May highlighted in red.

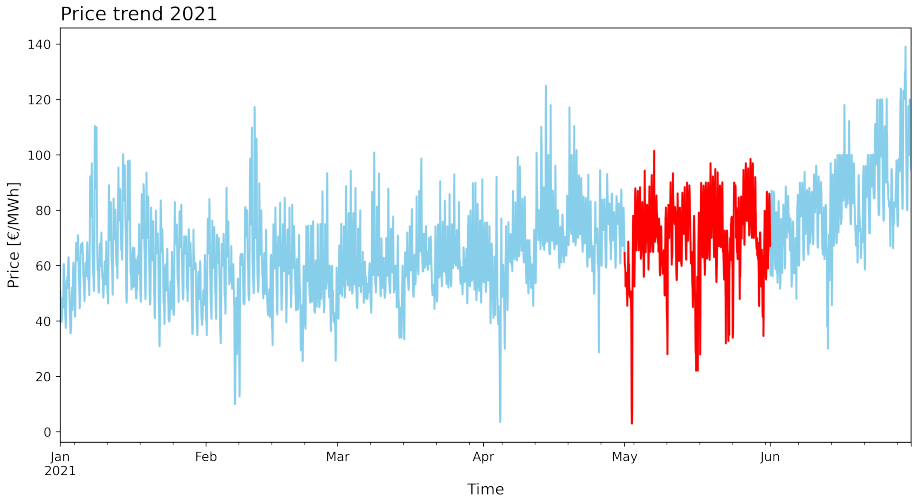


Figure 12.8: Price trend during 2021.

For what concerns this portion of the year, mean price was higher than both 2020 and 2019. 2021 is an interesting year since it presents the effects of a post-pandemic situation, with a still high uncertainty but still characterized by an economic upturn. This time series is characterized by:

- Mean value: 63.62 €/MWh;
- Deviation: 14.26 €/MWh;

Mean price during the month of May 2021 is close to the one characterizing the previous months of the year but slightly higher. The deviation of the available data during the month in which the prediction is performed is significantly higher with respect to the one that characterized the month of May 2019.

In the case of the month of May:

- Mean value: 69.5 €/MWh;
- Deviation: 14.27 €/MWh;

The MAPE for the prediction of May 2021 was evaluated too and reported in figure 12.9.

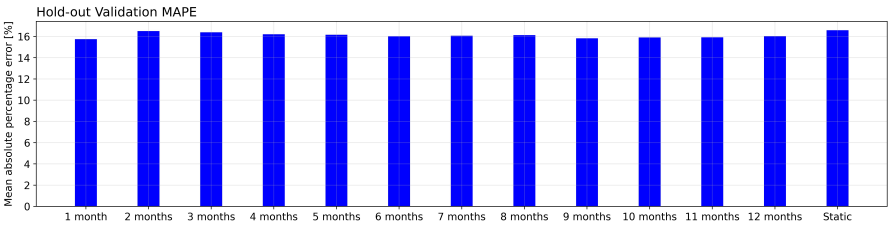


Figure 12.9: Mean absolute percentage error May 2021.

Considering this parameter, the best result still comes from the day by day prediction using 1 month of data:

- *MAPE from 1 month moving training window = 15.73%*

The MAPE obtained for the prediction of the price is almost half of the one obtained in 2020 (15.73% vs 31.52%). What surely contributes to this improvement is the substantially higher mean price during the month of May 2021 (21.05 €/MWh) with respect to the one during May 2020 (69.5 €/MWh), which means that the obtained absolute errors are normalized with respect to inevitably higher numbers.

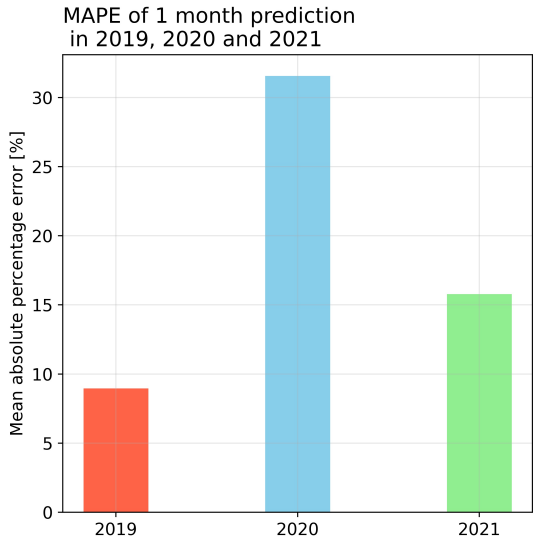


Figure 12.10: Mean absolute percentage error for price prediction in 2019, 2020 and 2021.

12.4 June 2021

The fact that abrupt peaks of variation affects substantially the quality of the prediction is corroborated by another application of the algorithm, this time performed on the price trend during June 2021. In figure 12.1 it is also visible that the mean electricity price during that month is even higher than the one during May 2021 and that the deviation of the price is still consistently high. However, by looking at the actual trend reported in figure 12.11, it can be noticed that during that month the price trend was not subjected at abrupt large variation as it was during the previous month (figure 11.6). The high deviation on the overall month is given by an almost constant rise in the mean electricity price. This more stable trend makes possible to produce a better prediction: the mean absolute percentage error committed on this trial has a value of 8,1%, a better result even compared to the one obtained with the prediction of the price trend during May 2021.

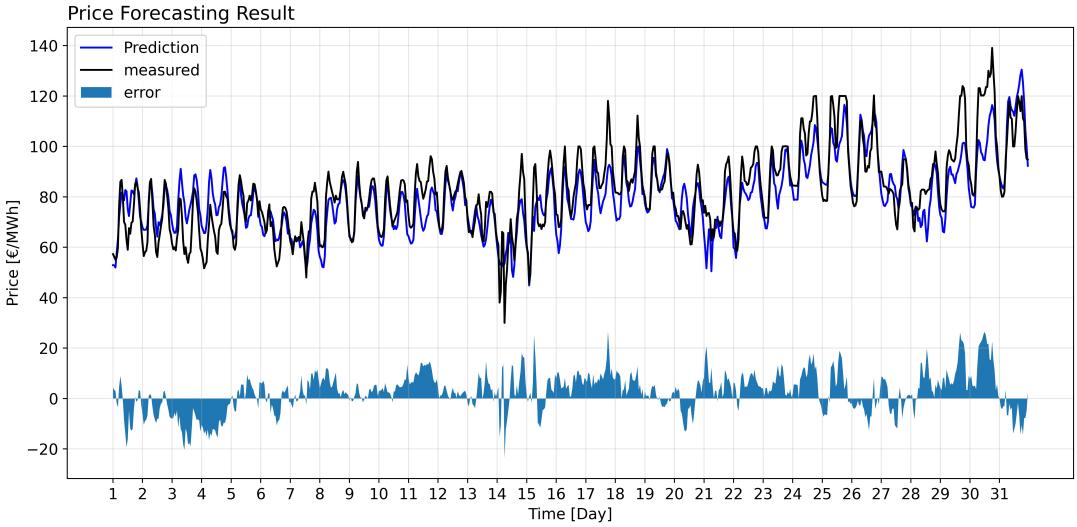


Figure 12.11: Predicted price trend June 2021.

12.5 Comparison with literature

Many attempts have been made to find the best algorithm to perform forecast of energy commodities. Tests related to the prediction of the PV power production and the electricity price have been reported in literature several times. Results obtained on the previously describes tests can be compared with results of other studies.

PV power forecasting error in literature

Regarding PV power prediction, a study made in 2018 [7] tested several methods for this kind of application. The study tested the prediction algorithm on each month of the year reports a NMAE of about 3%, claiming that, by increasing the accuracy of the weather prediction it can be improved to 2%. In this work the power production used as predictor variable the solar radiance measured on panels and it was characterized by and average value of the NMAE of 1.63%. This confirms that, with a very accurate weather prediction, is possible to forecast power production with an error lower than 2%.

[10] performed power predictions using neural networks based on weather forecasts in different weather conditions and got a NMAE of around 7.5%. This confirms again the importance of a good accuracy in the prediction of atmospheric conditions.

Electricity price forecasting error in literature

Regarding the price prediction, it must be considered that trials upon different markets and different periods make the comparison unreliable. Markets of other countries may have different dynamics with respect to the Italian one, which was considered in this work. In addition, results coming from pre-pandemic time periods must be compared only with tests done in May 2019, since it is clear that the virus introduced a variability on the market that makes forecasting a more different task to fulfill in this period.

In [13], a study published in 2011, a neural network was employed to perform the prediction. They also pointed out that the error committed increases dramatically in presence of spikes, as it happens with the prediction model used here. Their results were:

- MAE = 0.682 in days characterized by negligible spikes;
- MAE = 2.655 in normal trend (medium spikes);
- MAE = 9.282 in presence of large spikes.

This value can be compared to what is obtained in this work if is converted in a metric that can be used for comparison across different data sets. Normal trend analysed in [13] was characterized by a mean price of 33.41 €/MWh, meaning that the resulting normalized mean absolute error is roughly 7.95%. This result is referred to a trend similar to what was analysed in this work relatively to May 2019. The prediction made here using SVR produced a NMAE of 8.64%, comparable to what achieved in the 2011 study.

[14] also uses different configurations of neural networks and clustering. This study was made in 2016 and the best value of MAPE obtained was around 10%. Again this result is in line with what is achieved in this study.

[19] is a study made in Spain in 2007. MAPE obtained using again neural networks have an average of 8.91%, the same of what is obtained here in the prediction of May 2019.

In [5] was performed a prediction of the national electricity price (PUN) in Italy considering years form 2015 to 2018. Among several tested methods, the best result in terms of MAE came from a linear SVR, with an error of 4.49 €/MWh. Mean price during those years was 52.59 €/MWh and if the error is normalised on this number, it results in a NMAE of 8.54%. This value is very similar to that have been obtained here in the pre-covid market situation. This is the most meaningful comparison with the result obtained in this work, since is referred to the same market (Italy), even if referred to a slightly different price marker: national price (PUN) instead of zonal price (PZ).

Overall, it can be said that the error that characterize the price forecasts performed here is comparable with results reported in literature.

Part VI

Conclusion

Conclusion

In the work presented in this thesis, three different machine learning algorithms have been tested for both the photovoltaic power prediction and the zonal electricity price prediction. Each of them have been applied to large data sets containing records from several years for the training part, optimization of hyperparameters and selection of the most accurate one. Several metrics have been employed to establish which one performed better.

Photovoltaic power production resulted a relatively easy quantity to predict if an accurate weather prediction is available. Solar radiation on panels is linked to the power production by a correlation coefficient very close to 1. A linear model is able to predict effectively the future power production trend when solar radiation can be accurately predicted. The best model for this application resulted to be an optimized support vector regression that produced a forecast with a normalized mean absolute error of 1.63%, in line with what literature considers to be achievable with an high accuracy weather prediction.

Electricity price is a much more complex quantity to predict. It is influenced by many factors and there is not a related quantity that shows a such high Pearson correlation coefficient as it is between PV power and solar radiation. The external variable that has the highest correlation with price is the electrical load, due to the free market dynamics. However, it was pointed out that load alone is not sufficient to get an acceptable accuracy in price forecasting. Lagged time series proved to be the best candidate for this application: the highest correlation that it can be achieved with price trend comes from past values of it. The inclusion of other external variables, as the atmospheric temperature, showed an almost negligible improvement in terms of accuracy. The algorithm that performed best in this application was again a SVR with optimized hyperparameters. This method produced a NMAE of 10.32% in the cross-validation made upon the first tested years, a result in line with methodologies available in literature.

The application of the best models on some case studies confirmed results about the PV power forecasting and served as comparison between different market situations in the case of electricity price forecasting. It has been shown how prediction algorithms performed better before the pandemic situation, in 2019, when the market was not under the influence of such an unpredictable event. As it was expected, forecasting accuracy was drastically worse in 2020. In 2021, the economic rebound produced an increase in electricity prices and predictions showed an accuracy in between 2019 and 2020. The methodology of the moving training window proved that it is better to train the algorithm on a meaningful but not oversized amount of data. In highly fluctuating quantities as

prices, is better to focus on a recent history instead of whole years, as the present dynamics can be drastically different from the past ones.

Overall, obtained results are satisfactory as they are in line with what is reported in literature. In both cases, the best result comes from a regression algorithm. This algorithm performed best on average, producing the best result in terms of the selected metric. However, a linear model sees its accuracy decrease consistently when applied to a complex variable as the electricity price. In tests made to predict the trend of typical months, it was clear that the regression suffers the presence of peaks. For this application it would be interesting to test hybrid methods that employs a regression as SVR to predict the general trend of the price and nonlinear methods as neural networks to adjust the prediction in presence of abnormal events.

Acknowledgements

First of all, I would like to thank my supervisor *Professor Maurizio Repetto* and my co-supervisor *Ivan Mariuzzo* for having guided me through this work with constancy and patience. I have learned a lot from this final project and it is all thanks to them.

I would also like to thank `EGEA SPA` for providing all the required data for the photovoltaic production prediction and to `Arpa` for the atmospheric data.

Last but not least, a huge thanks goes to my family, for their support thorough this journey.

Bibliography

[1] Francisco Martínez-Álvarez, Alicia Troncoso, Gualberto Asencio-Cortés 1 and José C. Riquelme (2015). *A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting* Energies 2015, 8, 13162–13193.

[2] Sana Mujeeb1, Nadeem Javaid, Mariam Akbar, Rabiya Khalid1, Orooj Nazeer, and Mahnoor Khan (2019). *Big Data Analytics for Price and Load Forecasting in Smart Grids* Springer Nature Switzerland AG 2019L. Barolli et al. (Eds.): BWCCA 2018, LNDECT 25, pp. 77–87, 2019.

[3] Francisco J. Nogales, Javier Contreras, Member, IEEE, Antonio J. Conejo, Senior Member, IEEE, and Rosario Espínola (2002). *Forecasting Next-Day Electricity Prices by Time Series Models* IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 17, NO. 2, MAY 2002.

[4] Marco Cerchio 1, Francesco Gullí, Maurizio Repetto and Antonino Sanfilippo (2020). *Hybrid Energy Network Management: Simulation and Optimisation of Large Scale PV Coupled with Hydrogen Generation* Electronics 2020, 9, 1734.

[5] Mahmood Hosseini Imani, Ettore Bompard, Pietro Colella, Tao Huang (2020). *Predictive methods of electricity price: an application to the Italian electricity market* 978-1-7281-7455-6/20 2020 IEEE.

[6] MAngelica Gianfreda, Francesco Ravazzolo, Luca Rossini (2020). *Comparing the forecasting performances of linear models for electricity prices with high RES penetration* International Journal of Forecasting 36 (2020) 974–986.

[7] Lorenzo Gigoni, Alessandro Betti, Emanuele Crisostomi, Senior Member, IEEE, Alessandro Franco, Mauro Tucci, Fabrizio Bizzarri, and Debora Mucci (2017). *Day-Ahead Hourly Forecasting of Power Generation From Photovoltaic Plants* IEEE TRANSACTIONS ON SUSTAINABLE ENERGY, VOL. 9, NO. 2, APRIL 2018.

[8] Arooj Arif, Nadeem Javaid, Mubbashra Anwar, Afrah Naeem, Hira Gul, and Sahiba Fareed (2020). *Electricity Load and Price Forecasting Using Machine Learning Algorithms in Smart Grid: A Survey* Springer Nature Switzerland AG 2020 L. Barolli et al. (Eds.): WAINA 2020, AISC 1150, pp. 471–483, 2020.

[9] Mubbashra Anwar, Afrah Naeem, Hira Gul, Arooj Arif, Sahiba Fareed, and Nadeem Javaid. S (2020). *Electricity Price and Load Forecasting Using Data Analytics in Smart Grid: A Survey* Springer Nature Switzerland AG 2020L. Barolli et al. (Eds.): EIDWT 2020, LNDECT 47, pp. 427–439, 2020.

[10] S. Leva, A. Dolara, F. Grimaccia, M. Mussetta, E. Ogliari (2015). *Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power*

Mathematics and Computers in Simulation 131 (2017) 88–100.

[11] David Scott, Tom Simpson, Nikolaos Dervilis, Timothy Rogers, Keith Worden (2018). *Machine Learning for Energy Load Forecastin* IOP Conf. Series: Journal of Physics: Conf. Series 1106 (2018).

[12] Guido Cocchi, Leonardo Galli, Giulio Galvan, Marco Sciandrone, Matteo Cantù, Giuseppe Tomaselli (2017). *Machine learning methods for short-term bid forecasting in the renewable energy market: A case study in Italy* Wind Energy. 2018;21:357–371.

[13] Deepak Singhal, K.S. Swarup (2011). *Electricity price forecasting using artificial neural networks*0 Electrical Power and Energy Systems 33 (2011) 550–555.

[14] Ioannis P. Panapakidis, Athanasios S. Dagoumas (2016). *Day-ahead electricity price forecasting via the application of artificial neural network based models* Applied Energy 172 (2016) 132–151.

[15] Ciwei Gaoa, Ettore Bompard, Roberto Napoli, Haozhong Cheng (2007). *Price forecast in the competitive electricity market by support vector machine* Physica A 382 (2007) 98–113.

[16] Katarzyna Maciejowska, Weronika Nitka, Tomasz Weron (2021). *Enhancing load, wind and solar generation for day-ahead forecasting of electricity prices* Energy Economics 99 (2021) 105273.

[17] Azim Heydari, Meysam Majidi Nezhad, Elmira Pirshayan, Davide Astiaso Garcia, Farshid Keyniac, Livio De Santoli (2020). *Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm* Applied Energy 277 (2020) 115503.

[18] Jinxing Che, Jianzhou Wang (2010). *Short-term electricity prices forecasting based on support vector regression and Auto-regressive integrated moving average modeling* Energy Conversion and Management 51 (2010) 1911–1917.

[19] J.P.S. Catalao, S.J.P.S. Mariano a, V.M.F. Mendes b, L.A.F.M. Ferreira (2007). *Short-term electricity prices forecasting in a competitive market:A neural network approach* Electric Power Systems Research 77 (2007) 1297–1304.

[20] Howell, D. C. (2010) . *Statistical methods for psychology* 7th Ed. Wadsworth. Cengage Learning.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay (2011) *Scikit-learn: Machine Learning in Python* Journal of Machine Learning Research 12.

[22] Brownlee, Jason (2019). *A Gentle Introduction to the Rectified Linear Unit (ReLU)* Machine Learning Mastery.

[23] Nyoman Setiawana, Robert Kurniawana, Budi Yuniartoa, Rezzy Eko Carakab„ Bens Pardameanc (2021). *Parameter Optimization of Support Vector Regression Using Harris Hawks Optimization* Procedia Computer Science 179.

[24] Nishant Shukla, with Kenneth Fricklas, (2018). *Machine Learning with Tensorflow.* Manning publication.

[25] Brockwell, P.J.; Davis, R.A. (2002) *Introduction to Time Series and Forecasting.* Springer: Heidelberg.

[26] Shumway, R.H.; Stoffer, D.S. (2011) *Time Series Analysis and Its Applications (with R Examples)* Springer: Heidelberg.

[27] M. G. De Giorgi, P. M. Congedo, and M. Malvoni (2014) *Photovoltaic power forecasting using statistical methods: Impact of weather data* IET Sci., Meas. Technol., vol. 8, no. 3, pp. 90–97.

[28] S.H. Chen, A.J. Jakeman, J.P. Norton (2008) *Artificial intelligence techniques: An introduction to their use for modelling environmental systems* Math. Comput. Simulation 78 (2–3) 379–400.

[29] Gao, Ciwei, et al. (2008) *Bidding strategy with forecast technology based on support vector machine in the electricity market.* Physica A: Statistical Mechanics and its Applications 387.15.

[30] Moguerza, Javier M., and Alberto Muñoz. (2006) *Support vector machines with applications.* Statistical Science 21.3.

[31] Sansom, Damien C., Tom Downs, and Tapan K. Saha. (2003) *Evaluation of support vector machine based forecasting tool in electricity price forecasting for Australian national electricity market participants.* Journal of Electrical & Electronics Engineering.

[32] Keynia, Farshid. (2012) *A new feature selection algorithm and composite neural network for electricity price forecasting.* Engineering Applications of Artificial Intelligence 25.8.

[33] M. Marinelli, F. Sossan, G. T. Costanzo, and H. W. Bindner (2014) *Testing of a predictive control strategy for balancing renewable sources in a microgrid* IEEE Trans. Sustain. Energy, vol. 5, no. 4, pp. 1426–1433

[34] M. Tucci, E. Crisostomi, G. Giunta, and M. Raugi (2016) *A multi-objective method for short-term load forecasting in European countries* IEEE Trans. Power Syst., vol. 31, no. 5, pp. 3537–3547

[35] A. Yona, T. Senjyu, T. Funabashi, and C.-H. Kim (2013) *Determination method of insolation prediction with fuzzy and applying neural network for longterm ahead PV power output correction* IEEE Trans. Sustain. Energy vol. 4, no. 2, pp. 527–533

[36] F. Bizzarri, M. Bongiorno, A. Brambilla, G. Gruosso, and G. S. Gajani (2013) *Model of photovoltaic power plants for performance analysis and production forecast* IEEE Trans. Sustain. Energy, vol. 4, no. 2, pp. 278–285.

[37] A. Mellit, A.M. Pavan (2010) *A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid connected pv plant at trieste, italy* Sol. Energy 84 807–821.

[38] C. Monteiro, L.A. Fernandez-Jimenez, A. Ramirez-Rosado, I.J. ans Munoz-Jimenez, P.M. Lara-Santillan (2013) *Short-term forecasting models for photovoltaic plants: Analytical versus soft-computing techniques* Math. Probl. Eng. 2013 9.

[39] A. Dolara, G.C. Lazaroiu, S. Leva, G. Manzolini (2013) *Experimental investigation of partial shading scenarios on PV (photovoltaic) modules* Energy 55 466–475

[40] Wang, J., Liu, F., Song, Y., Zhao, J. (2016) *A novel model: dynamic choice artificial neural network (DCANN) for an electricity price forecasting system* Appl. Soft Comput. 48, 281–297

[41] Syed Naeem Ahmed (2015) *Physics and Engineering of Radiation Detection (Second Edition)*

[42] Ronald van Loon, (2018). *Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning.* Big Data Made Simple. `https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/`

[43] Claesen, Marc; Bart De Moor (2015). *Hyperparameter Search in Machine Learning* The XI Metaheuristics International Conference.

[44] Chin-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin (2010) *A practical guide to support vector classification* Technical Report, National Taiwan University.

[45] Ibrahem Kandel, Mauro Castelli (2020) *(*The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset) ICT Express, Volume 6, Issue 4.

[46] D.R. Wilson, T.R. Martinez (2003) *The general inefficiency of batch training for gradient descent learning* Neural Netw. 16 1429–1451.

[47] I. Goodfellow, Y. Bengio, A. Courville (2016) *Deep Learning* The MIT Press.

[48] Ian Goodfellow & Yoshua Bengio & Aaron Courville [Goodfellow, Ian & Bengio, Yoshua & Courville, Aaron (2016) *Deep Learning (Adaptive Computation and Machine Learning series)* MIT Press

[49] Yadav, S., & Shukla, S. (2016, February) *Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification* In 2016 IEEE 6th International conference on advanced computing (IACC) (pp. 78-83). IEEE.

[50] P. Refaeilzadeh, L. Tang, and H. Liu. (2009) "Cross validation, encyclopedia of database systems."

[51] Allen, David M (1974). *The Relationship between Variable Selection and Data Agumentation and a Method for Prediction* Technometrics. 16 (1): 125–127.

[52] Stone, M (1974). *Cross-Validatory Choice and Assessment of Statistical Predictions* Journal of the Royal Statistical Society, Series B (Methodological).

[53] Wes McKinney, (2010). *Data Structures for Statistical Computing in Python.* Proceedings of the 9th Python in Science Conference, Pages 56-61

[54] Energy Advisors (2021) *Gli indici IPEX del IV trimestre 2020 e del I trimestre 2021* Nuova Energia Pages 105-108

[55] G. Santamaría-Bonfil, A. Reyes-Ballesteros, C. Gershenson (2016) *Wind speed forecasting for wind farms: A method based on support vector regression* Renewable Energy 85

[56] Maddala, G. S. (1992). *Outliers* Introduction to Econometrics (2nd ed.) pp. 89