energies

MDPI

Article

Comparison of Data-Driven Techniques for Nowcasting Applied to an Industrial-Scale Photovoltaic Plant

Simone Sala [1,*], Alfonso Amendola [1], Sonia Leva [2], Marco Mussetta [2], Alessandro Niccolai [2] and Emanuele Ogliari [2]

1 Eni S.p.A., via Felice Maritano 26, San Donato Milanese, 20097 Milano, Italy; alfonso.amendola@eni.com
2 Dipartimento di Energia, Politecnico di Milano, 20156 Milano, Italy; sonia.leva@polimi.it (S.L.); marco.mussetta@polimi.it (M.M.); alessandro.niccolai@polimi.it (A.N.); emauelegiovanni.ogliari@polimi.it (E.O.)
* Correspondence: simone.sala@eni.com

Received: 29 October 2019; Accepted: 26 November 2019; Published: 27 November 2019

check for updates

Abstract: The inherently non-dispatchable nature of renewable sources, such as solar photovoltaic, is regarded as one of the main challenges hindering their massive integration in existing electric grids. Accurate forecasting of the power output of the solar plant might therefore play a key role towards this goal. In this paper, we compare several machine learning and deep learning algorithms for intra-hour forecasting of the output power of a 1 MW photovoltaic plant, using meteorological data acquired in the field. With the best performing algorithms, our data-driven workflow provided prediction performance that compares well with the present state of the art and could be applied in an industrial setting.

Keywords: photovoltaic power; forecasting; PV; machine learning; deep learning; nowcasting

1. Introduction

In recent years, among renewable energy sources, solar photovoltaic (PV) has experienced rapid growth. According to the most recent survey by the IEA [1], by the end of 2017, the total PV power capacity installed worldwide amounted to 398 GW, leading to the generation of over 460 TWh of energy (around 2% of global power output). Utility-scale projects account for about 60% of total PV installed capacity, with the rest in distributed applications (residential, commercial, and off-grid) [1]. It is well established that one of the main challenges posed by the massive integration of renewable energy sources, such as PV or wind power in the national grids is their inherently non-programmable nature. Therefore, the capability to forecast the energy produced by a PV plant in a given time frame with good accuracy is a key ingredient in overcoming this challenge [2], and is also often connected with economic benefits [3]. The recent evolution of microgrid and smart grid concepts is also a significant driving force for the development of accurate forecasting techniques [4].

In this paper, we will compare several data-driven techniques, such as machine learning and deep learning, for intra-hour forecasting of the output power of an industrial-scale PV plant (1 MW nominal capacity) at 5 min temporal resolution. Meteorological data acquired from sensors in the field will be used as input data to perform the predictions. This framing of the problem belongs to the class of approaches referred to as *nowcasting* [2,5].

From an operational point of view, short forecast horizons are relevant for AC power quality and grid stability issues. At these time scales the main source of variability in the power output is cloud movement and generation on a local scale [6]. Although these phenomena could be modeled from a physical point of view, their complex and turbulent behavior is a challenge that can be tackled effectively with data-driven techniques. Some approaches make use of imaging techniques, either from ground-based [6] or satellite [7] cameras, while others only make use of endogenous data [8] (i.e., the power output is used as the only input feature to the forecasting). The advantages of a data-driven approach for solar power forecasting lies in its ability to learn, from historical data, the complex patterns that are difficult to model (for instance shading phenomena). No knowledge of the specifications of the plant such as the orientation and electrical characteristics of the panels are required.

The predictive performance of machine learning-based methods has been shown to be superior to physical modeling in several cases [2,9,10]. In particular, our comparison includes several well-known regression algorithms that were tested on an extensive range of parameters.

The paper is structured as follows: Section 2 describes the available dataset and the performed preprocessing, Section 3 contains the description of the applied methods and the definition of the applied error metrics. The results are collected in Section 4, and the concluding remarks are drawn in Section 5.

2. Data Sources and Preprocessing

The input data for the present study were gathered from the continuous monitoring of a 1 MW nominal power photovoltaic plant operated by Eni, an Italian multinational energy company. The available time series covers approximately one year's worth of data, at 5 min time resolution. A list of the available variables is provided in Table 1.

Table 1. List of the available data.

Variable	Description	Units
P	Output power of the PV plant	[kW]
T_p	Panel temperature	[°C]
I_{GHI}	Global horizontal irradiance	[kW/m^2]
T	Ambient temperature	[°C]
H	Relative humidity	[%]
p	Atmospheric pressure	[mbar]
W_d	Wind direction	[deg]
W_s	Wind speed	[m/s]

The panel temperature T_p is measured by a sensor installed on the back side of one reference module in the field. Meteorological data are monitored by a weather station installed next to the PV arrays. The PV modules are equipped with a 1-axis solar tracking device.

2.1. Data Preprocessing

An example of the P and T time series is shown in Figure 1, data were collected between 3 August 2018 and 8 July 2019, and a total of 103,913 time steps are available.

2.2. Feature Selection

To find the best compromise between predictive performance and the computational load of the training phase, we have used several considerations to select the most relevant variables among those listed in Table 1.

The physics of the photovoltaic system represents the first source of intuition to guide the selection process. Sophisticated models for the power generated as a function of the environmental conditions have been developed [11]; in essence, the power output of a PV panel is proportional to the solar irradiance.

The proportionality coefficient contains information on the power conversion efficiency of the panel, and it is generally a decreasing function of the photovoltaic cell temperature [11]. Therefore, it is expected that I_{GHI}, followed by the panel temperature, should play a major role, as T_p is a good proxy for cell temperature.

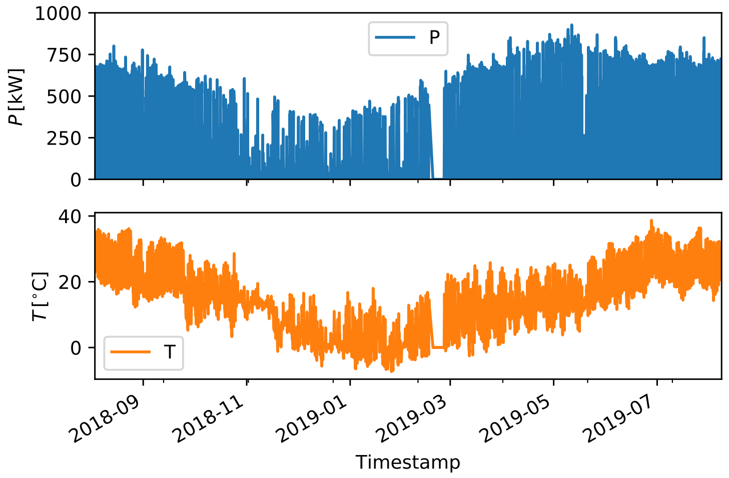

Figure 1. Plot of the P and T variables in the time frame used in the analysis.

Air temperature and humidity are also generally expected to be useful predictors, as they are related to solar irradiance by well-known physical laws. Relative humidity, for instance, is intuitively higher during rainy days, which most likely feature heavy cloud cover resulting in low ground-level solar irradiance hence low power output. Nevertheless, the impact of these two variables on the forecasting prediction is highly site-dependent. For this specific dataset, we have found that including both features did not increase predictive performance significantly; some algorithms actually performed worse when humidity was used as the input feature.

Weather parameters, such as wind direction and wind speed, for a given time instant t, they might influence the power generated at successive instants in time $P(t + \Delta t)$, for example by indicating the direction and speed of cloud movement. Wind speed could also influence panel temperature, albeit with a delayed response due to thermal capacitance effects [12]. For the specific dataset implemented, by testing the selected algorithms in an exploratory phase, results that no significant performance gains were found when including W_s, W_d, and p in our case study. This conclusion was also supported by examining statistical indicators, such as the feature importance provided by the Random Forest algorithm [13].

The expectations derived from the physics are also corroborated by an analysis of the correlation of each variable in the dataset with P, reported in Table 2. The most highly correlated variables, as anticipated, are I_{GHI} and T_p.

Table 2. Pearson's correlation coefficient between the available features and the output power P.

	I_{GHI}	T_p	H	T	W_s	p	W_d
Correlation with P	0.97	0.80	−0.69	0.58	0.28	0.028	0.016

The behavior of I_{GHI}, as displayed in Figure 2 clearly has an envelope due to the day-night cycle [10]. This envelope also embeds the seasonal variations of peak irradiance during the year. Local cloud cover introduces a stochastic component in the measured irradiance, which constitutes the challenging aspect of PV power forecasting.

Here, a synthetic feature has been used to separate these two contributions. For identifying the envelope, the so-called clear sky irradiance [14] I_{CS} was computed (To simplify the notation we will drop the subscript GHI in the derived variables; we consider the global horizontal component unless otherwise stated). This is the solar irradiance that would be expected in the absence of cloud cover but accounting for atmospheric turbidity (the effect of aerosols and water content on light scattering and absorption).

The clear sky irradiance takes into account the most common trends, like the day-night cycle and the seasonal variation. In this paper, no additional considerations have been introduced regarding trends, like the ones reported in [15], because they can be learned by the algorithms thanks to the fact that the dataset used covers an entire year.

The clear-sky irradiance model used in the implementation of the Ineichen model [16] was provided by the pvlib library [17].

The decomposition used can be formalized as:

$$I_{GHI}(t) = I_{CS}(t) + I_{\text{stoc}}(t), \tag{1}$$

is shown in Figure 2. Dropping the time dependence for simplicity, the following hypothesis can be made for the output power:

$$P = \varepsilon I_{GHI} = \varepsilon(t)(I_{CS} + I_{\text{stoc}}) = P_{CS} + P^{\text{s}}, \tag{2}$$

where we have introduced $\varepsilon(t)$ and have defined $P_{CS} = \varepsilon I_{CS}$ and $P^{\text{s}} = \varepsilon I_{\text{stoc}}$. The coefficient $\varepsilon(t)$ was estimated as the ratio P/I_{GHI} where $I_{GHI} > 10\,\text{W}/\text{m}^2$, and set to zero otherwise.

To reduce the impact of inconsistent data, a rolling average with a window size of 4 samples was then applied to ε. This was found to improve performance and window size has been tuned manually to achieve this goal.

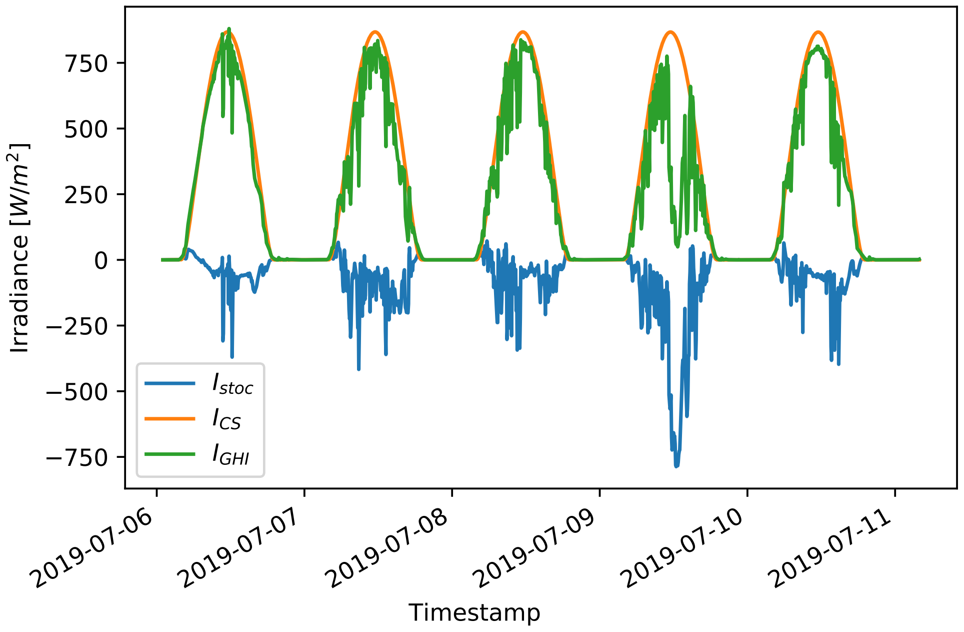

Figure 2. Separation in the stochastic (I_{stoc}) and deterministic (I_{CS}) contribution of the total irradiance (I_{GHI}), as shown in Equation (1).

The forecasting models were then trained to predict only the stochastic component of the power. The total power was then computed by inverting the transformations defined in Equations (1) and (2).

Finally, all variables were rescaled to the interval $[-1, 1]$. Scaling of all the features to the same interval is a standard practice to improve the performance of some regression algorithms, in particular, those based on neural networks [18].

To conclude this section, we remark that the main aim of the paper is not to demonstrate the best possible predictive performance, but rather to compare different models on equal grounds. The preprocessing steps outlined above were driven by a combination of physical reasoning, analysis of statistical indicators, and by the necessity of limiting the computational load when working at 5 min time resolution. Appropriate synthetic features were defined and the final list of physical features selected as the input and the output of the regression are summarized in Table 3. Finally, we stress that these choices were optimized for the case at hand and might not be as effective for different applications, as it is natural to expect in any data-driven approach.

Table 3. Selected physical variables (features) for power forecasting.

Selected Physical Variables				
I_{stoc}	T_p	T	P_{CS}	P^s

3. Methods

To apply machine learning techniques, forecasting was framed as a regression problem as follows:

$$\left[\hat{P}^s_{i+1}, \hat{P}^s_{i+2}, \ldots, \hat{P}^s_{i+12}\right] = f\left[\underline{x}_i, \ldots, \underline{x}_{i-B+1}\right] \tag{3}$$

where $\underline{x}_i \in \mathbb{R}^F$ is a vector containing the set of selected input features.

In the present case, $F = 5$ and $\underline{x}_i = \left[P^s, I_{\text{stoc}}, T_p, T, P_{CS}\right](t_i)$. Discrete time steps are defined so that $\Delta t = t_i - t_{i-1} = 5$ min, and a similar notation is adopted for the output of the model $\hat{P}^s_j = \hat{P}^s(t_j)$.

We have introduced the parameter B, which controls how far back in time the regressor will look to perform the prediction. This *lookback time*, denoted as ΔT_B, was varied between 2, 4, and 8 h, corresponding to $B = 24, 48, 96$ at 5 min time resolution (see Figure 3). Generally, the goal of a regression problem is to find the function $f : \mathbb{R}^{B\times F} \to \mathbb{R}^{12}$ which minimizes a suitable definition of the error between the predicted output $\hat{P}^s$ and the actual value P^s.

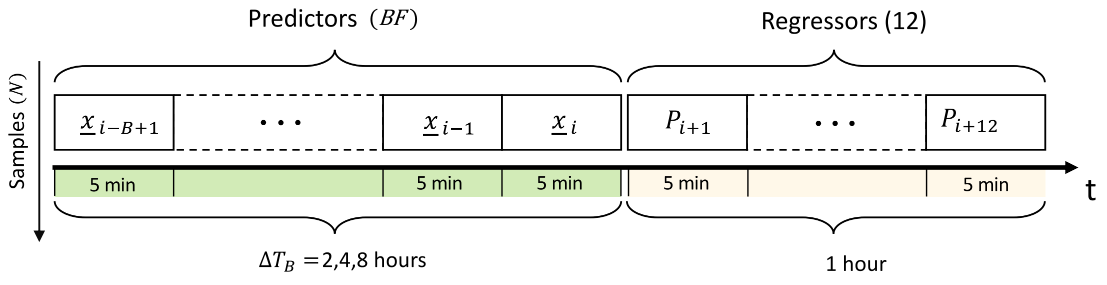

Figure 3. Structure of the input and output of the regression problem. B time steps in the past for each of the F physical features, for a total of BF predictors are fed as an input to the regression algorithm. The output is selected to simultaneously predict the 12 time steps corresponding to a forecast interval of 1 h at 5 min time resolution. The picture shows one sample (observation) indexed with $i = 1, \cdots, N$ where N is the number of samples considered (for instance $N = N_{\text{tr}}$ in the training phase).

To this end, a set of pairs of input-output variables on which to *train* the parameters (if any) of f was constructed and arranged as shown in Figure 3.

The dataset adopted is affected by some gaps (missing data) and some invalid data in the time series. The missing data have managed to split the dataset into subsets containing only consecutive observations: data within each subset were then cast in the desired form by constructing the time-lagged time series.

On the other hand, a first statistical analysis on the time series shows the inconsistent data are a negligible percentage of the entire dataset, and the machine learning methods analyzed in this study are able to automatically detect and ignore these invalid data. However, IEC 61724 recommendations for PV data monitoring and recording have been carefully followed even if the performance ratio of the considered PV plant has been shown here for non-disclosure reasons.

The final resulting dataset comprised N_{tot} = 103,740 samples: the first 80% samples were used as a training set, with a dimension of N_{tr} = 82,992, while the last 20% constituted a test set of $N_{\text{te}} = 20{,}748$ samples.

Many regression algorithms to implement the function f exist and have been applied to solar irradiance or photovoltaic power forecasting: artificial neural networks [19], tree-based methods [20], and more recently deep learning approaches [21]. The algorithms tested in this study are summarized in Table 4 and Figure 4 shows the flow-chart, which has been followed in the current analysis of the above-listed algorithms. A detailed description of the working principles of each algorithm is out of the scope of this work and we refer the interested reader to the references provided. For this reason, we now provide only a minimum of details that are useful to motivate the choice of algorithms listed in Table 4. Linear regression was included as a representative example of a robust and computationally inexpensive algorithm. The Lasso regressor [22] implements variable selection [23] by penalizing the importance of features which have little impact on the prediction error. Therefore, it should represent an improvement in the performance of simple linear regression, with a comparable computational cost. The Random Forest [24] algorithm is based on an ensemble of decision trees aggregated together following the principle of *bagging* [18,23]. The multilayer perceptron (MLP) [23] is a type of artificial neural network with multiple *hidden* neural layers. The number of layers and the number of neurons in each layer are the main parameters that control the flexibility of the network, that is its capability to learn complex patterns. Training of the MLP is based on the feed-forward and back-propagation paradigm.

In k-nearest-neighbor (KNN) regression [25], the output y for an input x is estimated as the average value of the k training outputs whose corresponding inputs are the first k *nearest* to input x. An appropriate notion of distance in the space of inputs must be defined, the choice of which impacts regression performance.

Long short-term memory [26] (LSTM) neural networks belong to the category of recurrent neural networks. The LSTM network structure can be described in terms of units communicating through input and output gates and endowed with a *forget gate* that regulates the flow of information between successive units. This enables the network to remember such information for long time intervals. This architecture is well suited to process data that have a sequential nature, as is the case of time series data. This is reflected in how the input data has to be structured (see Figure 5), either for training or to perform a prediction with an LSTM.

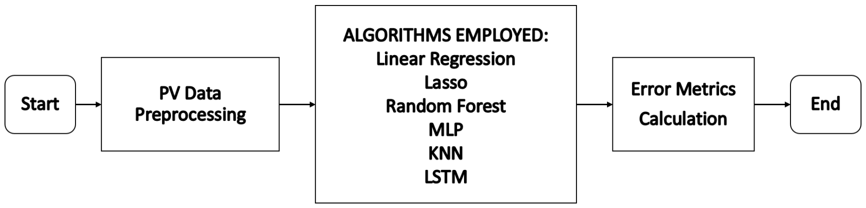

Figure 4. Flow-chart of the employed algorithms in this analysis.

Table 4. Selected regression algorithms.

Algorithm	Tuning	Implementation
Linear regression	CV and training parameter search	scikit-learn [13]
Lasso	CV and training parameter search	scikit-learn
Random Forest	CV and training parameter search	scikit-learn
MLP	CV and training parameter search	scikit-learn
KNN	CV and training parameter search	scikit-learn
LSTM	Manual tuning	keras [27]/Tensorflow [28]

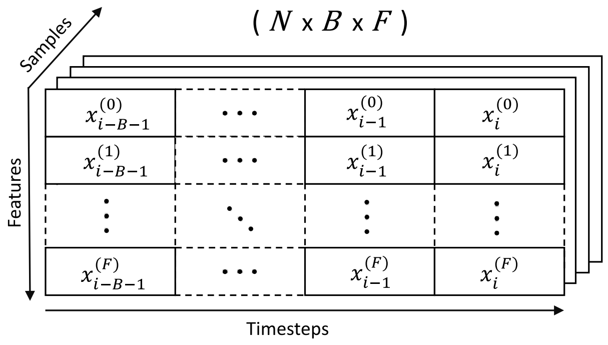

Figure 5. Sketch of the regression input specific for the long short-term memory (LSTM) network. Each sample is in this case composed of F sequences of B consecutive and chronologically ordered time steps. The variables $x_m^{(k)}$ are denoted so that the $k = 1, \cdots, F$ spans the input features while m is the time index.

Given a set of N samples, F physical features and B lookback time steps, each sample is a two-dimensional array of dimension $B \times F$, therefore a portion of the time series comprising B sequential steps is used to perform the prediction. On the contrary, each input sample to other algorithms is a flat array of $B \cdot F$ predictors (see Figure 1), which are all considered equally and independently. For instance, the order in which they are provided is completely irrelevant. The effectiveness of LSTM for time series forecasting has been demonstrated in several studies targeting different types of data [21,29].

All the algorithms listed in Table 4 (except for linear regression) have several parameters that can be tuned. An example is the number and size of the hidden layers in the MLP network. We refer to such variables as *training parameters* (especially in the context of deep learning, these are also often referred to as *hyperparameters*).

These can have a significant impact on the performance of the algorithm. A robust strategy to find the optimal configuration is to couple a search in training parameter space with k-fold cross-validation (CV). The latter consists of splitting the training set in k subsets (*folds*) of the same size, then over the course of k iterations, select one of the folds as the test set, while the remaining $k - 1$ constitute the training set. At the end, k performance scores are obtained, for instance, the determination coefficient (R^2) on the fold acting as the test set. An overall cross-validation score is then built as the average of the k scores. This procedure can, in turn, be iterated on a grid of possible values of the training parameters to find the combination yielding the highest cross-validation score. If m combinations of training parameters are tested, a total of mk fits of the model is performed. In this study, the general machine learning workflow used for all algorithms except for the LSTM was the following, for each value of the lookback time ΔT_B:

- Three-fold cross-validation and hyperparameter search on the training set (N_{tr}) samples: training parameters yielding the best cross-validation score were selected. This should select the algorithm with the best generalization performance, which is the one which most likely performs best on new, previously unseen, input data.
- Evaluation of the performance of the algorithm on the test set, using common error metrics, after reconstruction of the actual power from the predicted stochastic component.

All algorithms were tested on the same CPU-based hardware. The training time of the LSTM network was considerably longer than the other algorithms. The full cross-validation and hyperparameter search for the MLP, totaling 2400 fits required about 200 min for $\Delta T_B = 8$ h, whereas one single fit of the LSTM required 400 min. Both computations were parallelized to make use of all the threads available on the machine. For this reason, it was not feasible to apply the same hyperparameter tuning approach to the LSTM. Hence, we restricted the search to a family of networks known as encoder-decoder models [30]. In particular, two architectures were tested, one with a fixed number of units, the other with network size proportional to the lookback times steps B. We will label the two models LSTM and LSTM2, respectively.

Training of the LSTM networks consists of finding the optimal weights of the neural connection that minimize a suitable error metric on the training set, for instance, the mean square error (MSE). This was performed using the Adam [31] algorithm, a standard approach for deep neural networks. Training consists of several successive steps (*epochs*). The train MSE, being the target of the optimization, is by design expected to decrease through the epochs. This does not guarantee that the error on unseen data, for instance on the test set, will decrease as well. This is the well-known issue of overfitting [23], an example of which is shown in Figure 6. To prevent this occurrence, an *early stopping* strategy was implemented, keeping track of the MSE on the test set during training and halting the process when the test MSE stops decreasing. Early stopping was also enabled in the scikit-learn library for the MLP regressor. More technical details on the choice of parameters for all the algorithms are provided in Appendix A.

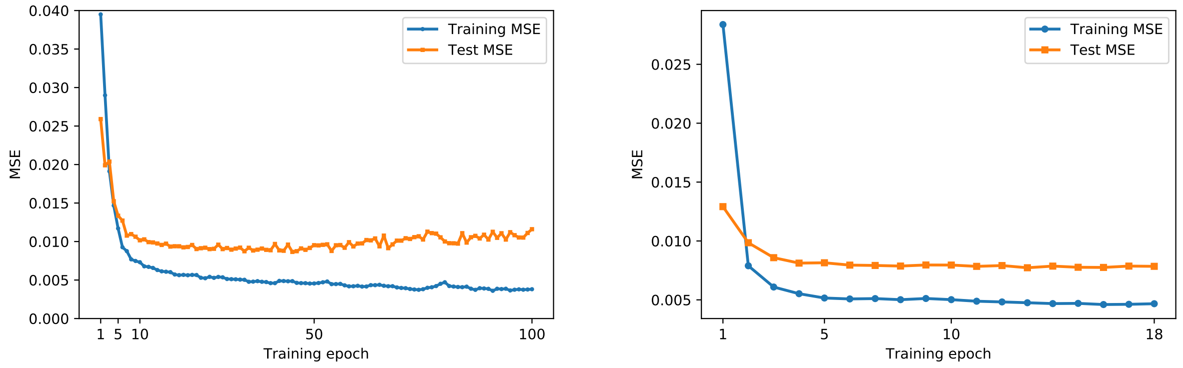

Figure 6. Comparison of the mean square error on the training and on the test set, as the LSTM is trained. Overfitting can occur if training is extended for too long (**left**). A suitable early stopping strategy helps to prevent this occurrence (**right**).

Error Metrics

We introduce the main error metrics that will be used in Section 4. Since the output of our intra-hour forecast is the 12-dimensional vector of the predicted power for each 5-min time step, we will define an error metric for each forecast horizon. That is, for $j = 1, \dots, 12$:

$$\mathrm{MAE}_j = \frac{1}{N}\sum_{i=1}^{N}\left|P_j^{(i)} - \hat{P}_j^{(i)}\right|, \tag{4}$$

$$\mathrm{RMSE}_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(P_j^{(i)} - \hat{P}_j^{(i)}\right)^2}, \tag{5}$$

$$\mathrm{MBE}_j = \frac{1}{N}\sum_{i=1}^{N}\left(P_j^{(i)} - \hat{P}_j^{(i)}\right), \tag{6}$$

$$R_j^2 = 1 - \frac{\mathrm{Var}(\hat{\mathrm{P}}_\mathrm{j} - \mathrm{P})}{\mathrm{Var}(\mathrm{P_j})}, \tag{7}$$

where the index i spans over the samples.

The performance metrics defined in Equations (4) to (7) are commonly referred to as the mean absolute error (MAE), the root mean square error (RMSE), mean bias error (MBE), and the coefficient of determination (R^2), respectively.

It is also interesting to evaluate the mean performance of the algorithm in the forecasted hour with a single representative indicator. Several indicators could be built to characterize different qualitative aspects of the prediction. A possible choice is to define the average power in the hour as $P_{1h} = \frac{1}{12}\sum_{j=1}^{12} P_j$. With a similar definition for the predicted power ($\hat{P}_{1h}$), standard definitions of the performance metrics (MAE, RMSE, R^2) can be applied to the hourly average power. We will label these metrics as $\mathrm{MAE_{1h}}$, $\mathrm{RMSE_{1h}}$, and $\mathrm{R^2_{1h}}$.

A persistence forecast was defined as

$$\left[\hat{P}_i^s, \hat{P}_{i+1}^s, \dots, \hat{P}_{i+11}^s\right] = \left[P_{i-12}^s, P_{i-11}^s, \dots, P_{i-1}^s\right] \tag{8}$$

where the 12 predicted time steps of $\hat{P}^s$ are equal to the measured values P^s observed in the previous 12 time intervals.

The last error metric introduced in the skill scores (SS) [32], that compares the prediction of the analyzed model with the error made by a persistence method on the same forecast horizon:

$$SS_j = 1 - \frac{\mathrm{RMSE}_j}{\mathrm{RMSE}_j^{\mathrm{pers}}}. \tag{9}$$

4. Results

The presented methods have been compared to the dataset described in Section 2. The number of samples in the dataset and their division for training and testing are summarized in Table 5.

In all the performed tests, three different lookback times have been adopted. This corresponds to a different number of lookback time steps used in each sample of the dataset. The information of these tests are summarized in Table 6.

Table 5. Summary of the information related to the construction of the dataset for the regression problem (see Figures 3 and 5).

Parameter	Definition	Values
N	Number of samples	103,740
N_{tr}	Number of training samples	82,992
N_{te}	Number of test samples	20,748
F	Number of features	5

Table 6. Summary of the different lookback time tested.

Lookback Time ΔT_B	Lookback Time Steps B
2 h	24
4 h	48
8 h	96

4.1. Cross-Validation Performance

We start the analysis from the cross-validation results. Table 7 summarizes the maximum, average, and standard deviation of the cross-validation score. The size of the population on which these quantities are calculated is the number of iterations in the hyperparameter search, which is the number of different combinations of parameters explored.

Table 7. Summary of cross-validation results. For each setting of ΔT_B, the maximum, average, and the standard deviation of the cross-validation score (R^2) are reported.

	Cross-Validation Score								
	Maximum			Average			Standard Deviation		
Lookback Time	2 h	4 h	8 h	2 h	4 h	8 h	2 h	4 h	8 h
Linear regression	0.78	0.78	0.78	0.78	0.78	0.78	0.0019	0.0023	0.0008
Lasso	0.77	0.77	0.77	0.07	0.07	0.07	0.2300	0.2301	0.2301
Random Forest	0.79	0.78	0.74	0.78	0.77	0.74	0.0031	0.0022	0.0027
MLP	0.79	0.79	0.78	0.70	0.70	0.69	0.1705	0.1560	0.1402
KNN	0.64	0.53	0.40	0.60	0.48	0.35	0.0403	0.0509	0.0588

Each quantity is informative about a different aspect of the algorithm performance. The maximum score is the criterion used to select the best set of training parameters, therefore it can be expected that an increase of the best CV score with ΔT_B should correspond to higher performance on the test set as well. None of the tested algorithms showed such an increase. Results on the test set reported in the next section confirm the validity of this conclusion on our data set. The number of predictors is already $BF = 120$ for $\Delta T_B = 2$ h, which is rather large. Indicators such as the feature importance provided by the Random Forest algorithm suggest that only a small fraction of the inputs effectively play a role in the prediction outcome. Therefore, adding even more predictors by increasing ΔT_B has no benefit and was found to actually decrease the performance in some cases.

The average value and standard deviation of the CV scores are useful indicators of the robustness of the regression algorithms with respect to hyperparameter selection. The Lasso and MLP algorithms have a much greater variability compared to Random Forest. We remark that the cross-validation score can also assume negative values, which justified the low average value observed for the Lasso. KNN is also fairly robust but has poor prediction performance compared to the others. Note that in the case of linear regression there are no training parameters to be tuned, hence all the observed variability is due to the different splittings of cross-validation.

4.2. Test Set Performance

We now examine performance on the test set, also including the two LSTM architectures (see Appendix A), which were not subjected to cross-validation.

We focus on the RMSE metric, shown in Figure 7. At the shortest forecast horizon (f_h) of 5 min, Linear Regression and Lasso outperformed more flexible regression algorithms. The situation is reversed for $f_h = 60$ min. The KNN algorithm performed poorly, suggesting that it is not well-suited for this specific data and framing of the problem. Nevertheless, it shows more promising performances if the MAE metric is considered (see Figure 8). The LSTM algorithms did not offer any significant performance advantage over Random Forest and MLP in terms.

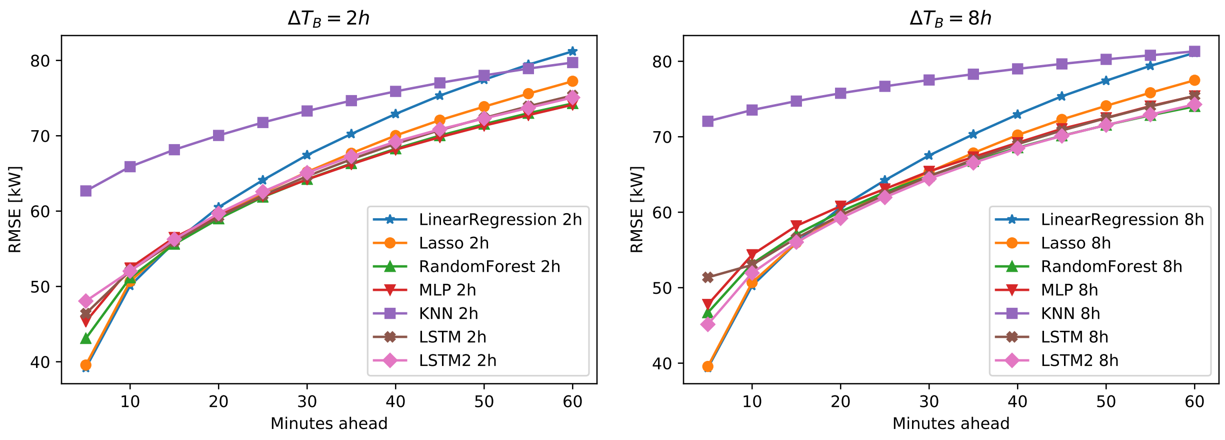

Figure 7. Plots of RMSE$_j$ with $j = 1, \cdots, 12$, corresponding to forecast horizons from 5 to 60 min ahead, for $\Delta T_B = 2$ h (**left**) and $\Delta T_B = 8$ h (**right**).

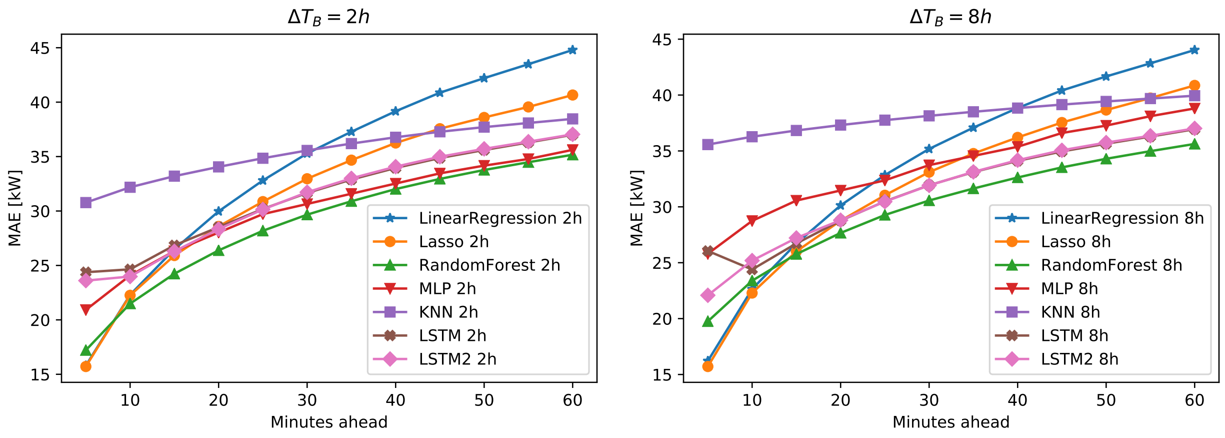

Figure 8. Plots of MAE$_j$ with $j = 1, \cdots, 12$, corresponding to forecast horizons from 5 to 60 min ahead, for $\Delta T_B = 2$ h (**left**) and $\Delta T_B = 8$ h (**right**).

It is worthwhile to consider additional error metrics since they highlight different aspects of the regression performance. For instance, the RMSE tends to penalize much more large deviations with respect to MAE. The latter is shown in Figure 8 and suggests a different ranking between the algorithms. It is again apparent that no significant improvements were obtained by increasing ΔT_B.

The mean bias error (MBE) as defined in Equation (6) preserves the sign of the deviation between predicted and actual values. Therefore, it is informative on potential systematic prediction bias. In Figure 9 one can observe that, in contrast with other indicators, MBE is almost constant as a function of the forecast horizon, except for a weak dependence in the case of Linear Regression. The two algorithms that perform worse in terms of RMSE (Linear regression and KNN) are also those with the largest bias.

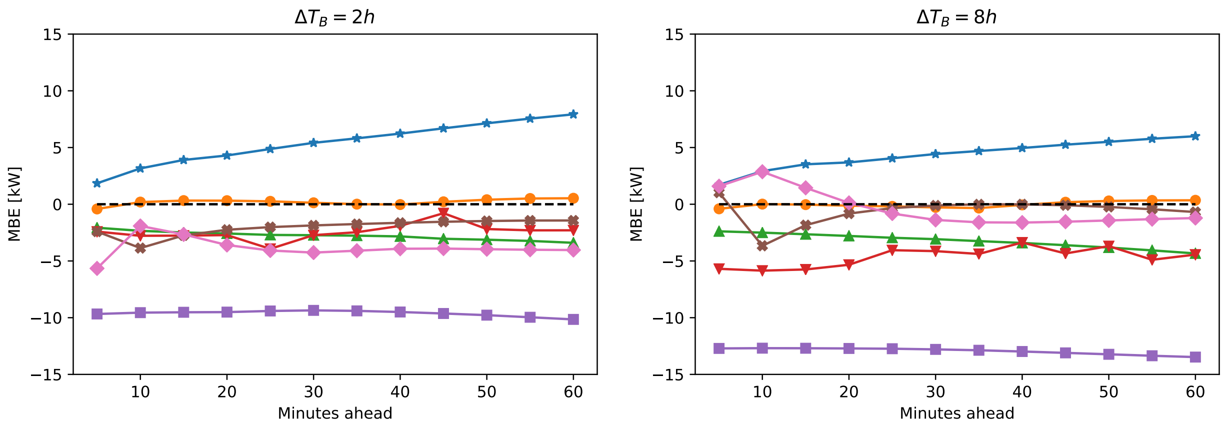

Figure 9. Plots of MBE_j with $j = 1, \cdots, 12$, corresponding to forecast horizons from 5 to 60 min ahead, for $\Delta T_B = 2$ h (**left**) and $\Delta T_B = 8$ h (**right**). To preserve the readability of the plot, the legend is not reported and follows the same convention in Figure 7.

Finally, we examine the Skill Score (defined in Equation (9)) against the persistence model defined in Section 3. As pointed out in a recent review [32], typical skill scores obtained in the literature against persistence by state of the art data-driven forecasts are in the range of SS = 0.3–0.4. In Figure 10 we have thus superimposed to the results reference line at SS = 0.3 to guide the eye. Unsurprisingly considering its definition, the ranking in skill between the algorithms is the same as the one suggested by the RMSE metric. Overall results suggests that a hybrid algorithm using a linear model for the first 5 min and a more flexible model for longer forecasts horizons could be considered to further improve performance. With respect to the peak plant power, the average RMSE reported is at the level of 5% for the best performing algorithms, with a MAE of the order of 3%. Table 8 summarizes the predictive performance of the hourly average power output.

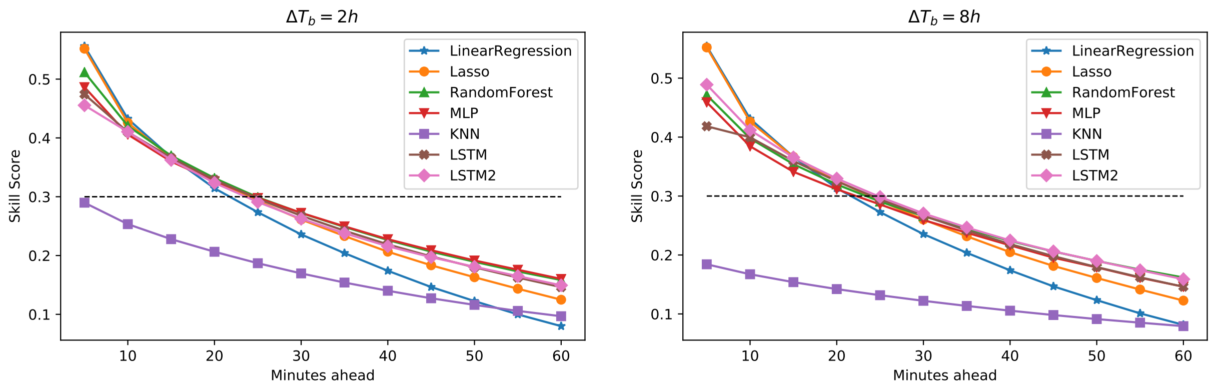

Figure 10. Plots of SS_j with $j = 1, \cdots, 12$, corresponding to forecast horizons from 5 to 60 min ahead, for $\Delta T_B = 2$ h (**left**) and $\Delta T_B = 8$ h (**right**). A dashed line at $SS = 0.3$ is inserted to guide the eye.

Table 8. Summary of the hourly average power output predictive performance, quantified by the metrics RMSE_{1h}, MAE_{1h}, MBE_{1h}, and R^2_{1h} defined in Section 3.

	RMSE_{1h} [kW]			**MAE_{1h} [kW]**			**MBE_{1h} [kW]**			**R^2_{1h}**		
Lookback Time	**2 h**	**4 h**	**8 h**	**2 h**	**4 h**	**8 h**	**2 h**	**4 h**	**8 h**	**2 h**	**4 h**	**8 h**
Linear regression	50.8	50.7	50.7	27.9	27.7	27.6	5.40	5.12	4.37	0.968	0.969	0.968
Lasso	48.4	48.5	48.5	25.6	25.6	25.7	0.20	-0.01	-0.02	0.971	0.971	0.971
Random Forest	47.0	47.8	47.9	22.3	22.8	23.1	−2.77	−3.79	−3.24	0.973	0.972	0.972
MLP	46.8	48.8	48.5	23.0	25.5	26.5	−2.44	−1.10	−4.66	0.973	0.971	0.971
KNN	58.5	61.2	63.7	29.1	30.7	31.9	−9.62	−10.88	−12.94	0.958	0.954	0.950
LSTM	47.4	47.8	47.7	24.2	24.0	24.1	−2.03	−0.11	−0.60	0.972	0.972	0.972
LSTM2	47.8	47.4	46.9	23.9	24.6	24.2	−3.84	−5.32	−0.40	0.972	0.972	0.973

This is one of the possible ways to quantify the performance over the whole forecast interval. One can find in confirmation of some of the trends highlighted by cross-validation scores. For example, the Random Forest shows a decrease in performance as ΔT_B increases. On the other hand, the LSTM2 architecture is the only one to show an appreciable improvement. This behavior could have been expected from the design of the LSTM network, as described in Section 3. Nevertheless, we have verified that further increasing ΔT_B to 12 h did not increase performance of LSTM2 and also produced a significant rise in training times.

To better analyze the connection between the error metrics of Table 8, it is useful to compare the full distribution of the residuals ($\delta = P_{1h} - \hat{P}_{1h}$) on the test set. The average of this distribution is, by definition, the MBE, whereas its variance $\sigma^2 = \text{Var}(\delta)$ is related to the RMSE as $\sigma^2 = RMSE^2 - MBE^2$. Therefore, qualitative analysis of the probability distributions can offer further insight on such metrics. For instance, Figure 11 (left panel) compares the two algorithms with the largest MBE and RMSE. Negative bias of the KNN mostly arises from the heavier tail of the distribution on the negative side. In the right panel, two of the best performing algorithms are compared and display a very similar distribution. This qualitative outlook should reinforce the conclusion that residual bias is small compared to the spread of the predictions and should not play any major role in practical applications.

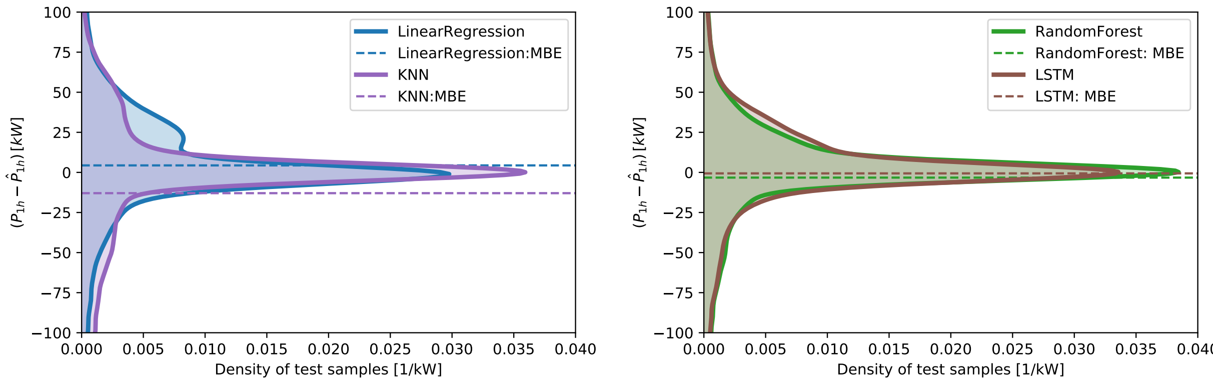

Figure 11. Distributions of the residuals of the actual and predicted average power over the hour, namely ($P_{1h} - \hat{P}_{1h}$). Continuous probability density functions (PDF) were obtained from the histograms via the kernel density estimation [33] to enhance the readability of the plots. For the same reason, only two pairs of algorithms were selected for comparison. Dashed lines represent the MBE, which is the average value of the PDF. $\Delta T_b = 8$ h was set in all four cases.

To offer some more qualitative insight on the predictive performance, in Figure 12 we show scatter plots of the actual vs. predicted power at different forecasts horizons. In keeping with the notation of

Equation (3), examining a forecast horizon of 5 min corresponds to selecting P_1 and $\hat{P}_1$, and similarly for the other horizons.

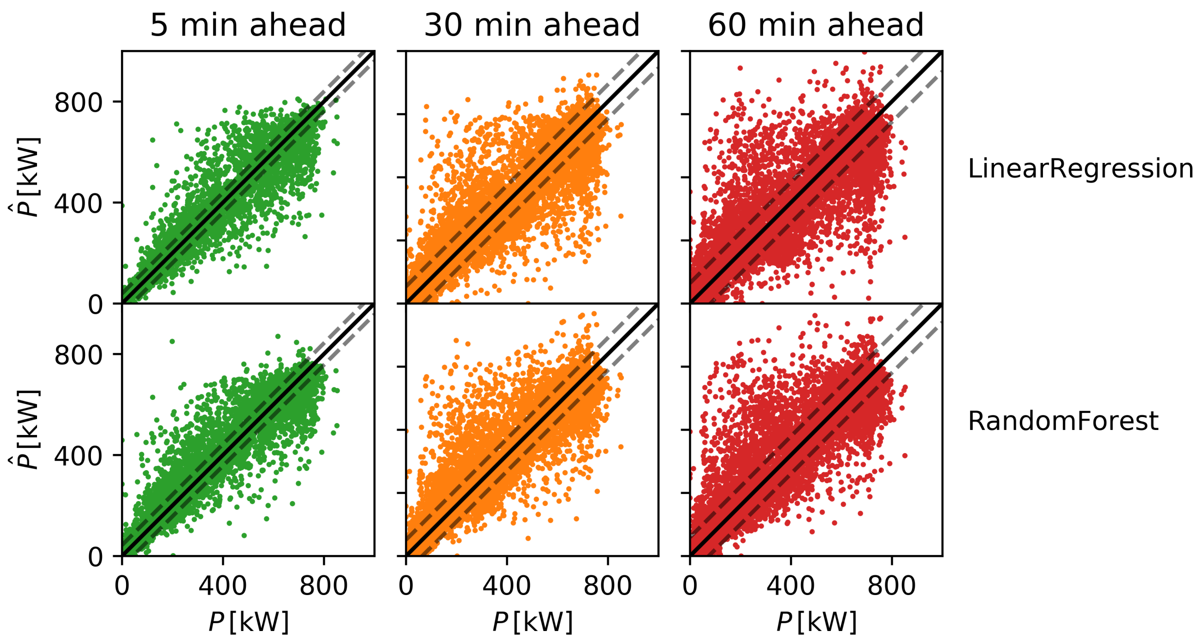

Figure 12. Scatter plots comparing actual (P) and predicted ($\hat{P}$) for different forecasts horizons and algorithm. Dashed lines shifted by ±RMSE from the identity line were inserted to give visual indication of the spread of the data. $\Delta T_B = 2$ h was selected for both algorithms.

Figure 13 shows a comparison of predicted and measured output on three full days, using the Random Forest algorithm with $\Delta T_B = 2$ h. These days were selected by computing the RMSE on each day of the test set for the three selected values of the forecast horizon. Then the distribution of the error was divided in three quantiles. One day for each quantile was then randomly selected, to represent cases where the prediction performs increasingly worse (from top to bottom in Figure 13). Prediction errors during the selected days are also reported in Table 9.

Table 9. Predictive performance during the day in terms of the indicators RMSE, MAE, R^2, for the three representative days reported in Figure 13.

	RMSE [kW]			**MAE [kW]**			**R^2**		
f_h [min]	**5**	**30**	**60**	**5**	**30**	**60**	**5**	**30**	**60**
Day 1	22.0	34.9	38.8	10.3	18.6	22.5	0.995	0.988	0.985
Day 2	40.5	59.7	71.1	21.3	33.9	40.9	0.924	0.836	0.767
Day 3	60.5	85.1	104.1	29.0	44.1	55.2	0.882	0.766	0.649

Plots of this kind must be interpreted assuming that the algorithm is in continuous operation to provide updated predictions based on new measurements every 5 min. Then one can compare P with the one predicted 5, 30, and 60 min before, as shown in Figure 13. The regressor seems to be able to predict rather rapid oscillations in the power output in a short term forecasting scenario ($f_h = 5$ min), while it has a tendency to overshoot for longer lead times. It is interesting to note that the magnitude rightmost peak in Day 2 was heavily underestimated even for $f_h = 5$ min. On the other hand, the small rise appearing after the power dropping to zero for several time steps in Day 3 was captured, in spite of the anomalous nature of the event. Note that the flat behavior of the power output in the central part of Day 1 (close to clear sky conditions) is due to the solar tracking system. It is interesting to note that the algorithms provided satisfactory performance by only making use of global horizontal irradiation. No information on solar tracker (such as the time dependence of panel angle) were explicitly fed to the models. The capability to learn complex patterns, reducing physical modeling efforts, is one of the main advantages of a data-driven approach.

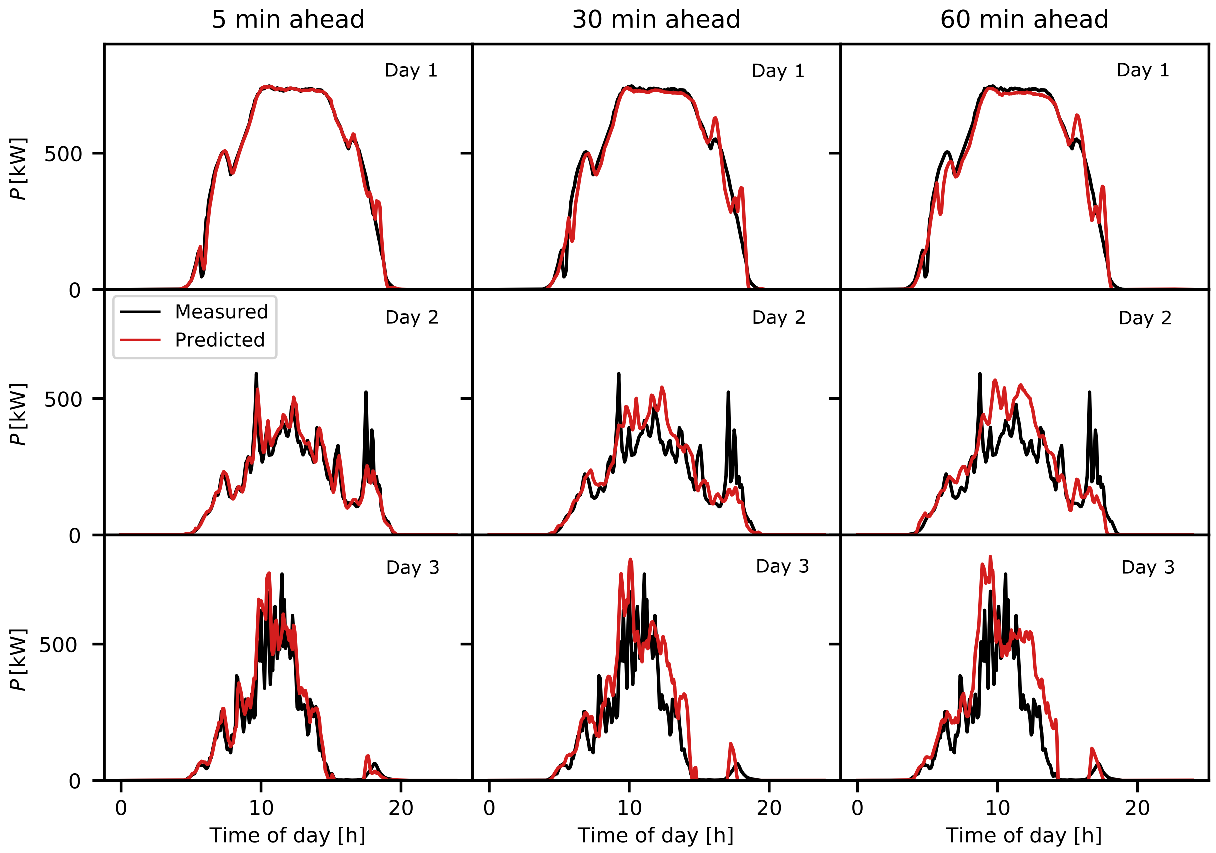

Figure 13. Comparison of predictive performance on three selected days, and three forecasts horizon. Black line (measured) is the actual power, whereas the red line (predicted) is the power which was predicted 5, 30, or 60 min before, respectively from left to right.

5. Conclusions

In this study, several well-know machine learning and deep learning algorithms were compared for the purpose of intra-hour forecasting of the power output of a 1 MW industrial-scale photovoltaic plant; the dataset implemented cover 1 year of measurements.

Each algorithm was tested over a large set of training parameters and standard machine learning practices, such as cross-validation were applied. We maintain that this is approach was required for a fair assessment of the performance.

The results highlighted some peculiar features of each regression technique: the Random Forest algorithm, with appropriate selection of the lookback time, was generally the best performing according to all the figures of merit we considered. The strength of this algorithm also lies in its robustness with respect to tuning of the training parameters. This is a particularly desirable feature for industrial applications, where the greater variability of the MLP would require fine-tuning of the parameters for each application on which the algorithm has to be deployed. On the other hand, the ultimate predictive performance of the two algorithms were similar. It must also be highlighted that for short forecast horizons Lasso and Linear regression tend to match or outperform the more flexible algorithms, thus, a hybrid solution for prediction could be a promising development. For the case in hand, we did not find significant benefits in LSTM networks, although they were the only algorithms to show an improvement when increasing the lookback time. This suggests that more complex LSTM architectures could be able to outperform conventional algorithms, provided that suitable hardware to handle very deep neural networks are available in the target application. There are many variables characterizing PV power forecasting efforts [2]: forecast horizon, forecast interval, type of input data, time resolution, size of the target plant, and geographical region. Therefore, it is often impossible to find a benchmark that shares all these characteristics against which new results can be compared directly [32,34].

The results achieved in this paper are related to a single dataset; to reduce the site-dependence of the obtained results, a comparison across different datasets should be performed. Future work will extend the here-exposed analysis to datasets belonging to different site conditions.

Nevertheless, we have identified several studies relevant to contextualize our findings in the present state of the art. For instance LSTM and CNN algorithms were shown to outperform MLP on nowcasting ($f_h = 1$ min) using sky images as additional input. The maximum reported skill score was 0.21 against persistence [35]. Using sky camera features, prediction of 10 to 30 min ahead was reported with a forecast skill up to 0.43 [36]. A skill score of 0.1 was reported for up to 30 min-ahead forecasting of PV power using only spatio-temporal correlation between neighboring PV plants [37]. A recent study [38] investigated several nowcasting algorithms in a microgrid setting (750 W PV power), finding the Random Forest as the best performing. Other studies aim at the forecasting of solar irradiance rather than the PV power directly. In this setting, forecast skills up to 0.3 [39] and 0.15 [40] were reported for intra-hour predictions.

Our approach to data preprocessing prioritized simplicity in the implementation and the identification of the smallest possible set of useful predictors. This is motivated by the need of minimizing the number of sensors to be installed and monitored for effective forecasting. For example, it is remarkable that quality predictions were obtained for a solar-tracking plant by making use of only the global horizontal irradiance. The capability to learn complex patterns without resorting to physical modeling is the main features of data-driven approaches, and was achieved by all the tested algorithms. Naturally, more complex regressors, such as Random Forest should always be preferred for optimal performance. Deep learning tools such as LSTM networks are very promising but might require dedicated hardware (GPUs) or larger training datasets to show significant performance advantage when deployed in industrial applications. This study suggests an effective data-driven workflow for intra-hour PV power forecasting, covering from feature selection to the identification of the best performing algorithms. Such an approach could be of interest for practical applications.

Author Contributions: Conceptualization: A.A., S.S.; Formal analysis: S.S., A.A.; Methodology: S.S.; Validation: A.A., S.L., M.M., A.N., E.O.; Software: S.S.; Supervision: A.A., S.L., M.M., A.N., E.O.; Writing—original draft: S.S.; Writing—review and editing: A.A., S.L., M.M., A.N., E.O.

Funding: This research received no external funding.

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

Appendix A

Table A1 details the range of values used in hyperparameter search. For each algorithm, *m* combinations of training parameters were randomly sampled from the available range, in order to make the search more efficient. We stress that this sampling was performed only once, therefore for all values of

ΔT_B the same selection of parameters was used. The number m was tuned for each algorithm in order to maintain a manageable computing time for the cross-validation and search. Computation was performed on a dual Xeon E5-2680 machine, featuring 16 physical cores and 32 logical threads.

Table A1. Summary of the range of training parameters searched for each algorithm.

Random Forest		**Lasso**	
n_estimators	300, 400	alpha	[0, 0.2]
max_depth	10, 20, 100, auto	selection	random, cyclic
min_samples_split	2, 10, 50	warm_start	true, false
min_samples_leaf	4, 10, 50		
MLP		**KNN**	
activation	relu, tanh, logistic	n_neighbors	10, 20, 200, 300
alpha	[0, 1]	selection	random, cyclic
learning_rate	adaptive, constant		
hidden_layer_size	5, 55, 155		
n_hidden_layers	2, 3		

The notation $[a, b]$ indicates uniform sampling in the $[a, b]$ range for real-valued parameters. The parameter names refer to the scikit-learn implementation of the algorithms. The reader can refer to the official documentation [13] for a precise definition. The LSTM and LSTM2 regressors were implemented using the Keras library with the Tensorflow backend. The first employed two 200 units network acting as the encoder and decoder. LSTM2 featured an encoder network with $5B/2$ units, and two stacked layers with $5B/3$ and $5B/6$ units respectively, as the decoder. The coefficients were tuned manually and the size of the network was adapter to the input size via the B dependence. The LSTM2 is thus able to outperform LSTM for longer lookback times.

References

1. IEA. Renewables 2018—Market Analysis and Forecast from 2018 to 2023. 2018. Available online: https://www.iea.org/renewables2018/ (accessed on 13 September 2019).
2. Antonanzas, J.; Osorio, N.; Escobar, R.; Urraca, R.; Martinez-de-Pison, F.; Antonanzas-Torres, F. Review of Photovoltaic Power Forecasting. *Sol. Energy* **2016**, *136*, 78–111, doi:10.1016/j.solener.2016.06.069. [CrossRef]
3. Cros, S.; Buessler, E.; Huet, L.; Sébastien, N.; Schmutz, N. The Benefits of Intraday Solar Irradiance Forecasting to Adjust the Day-Ahead Scheduled PV Power. In Proceedings of the Solar Integration Workshop 2015, Brussels, Belgium, 19–20 October 2015.
4. Wan, C.; Zhao, J.; Song, Y.; Xu, Z.; Lin, J.; Hu, Z. Photovoltaic and Solar Power Forecasting for Smart Grid Energy Management. *CSEE J. Power Energy Syst.* **2015**, *1*, 38–46, doi:10.17775/CSEEJPES.2015.00046. [CrossRef]
5. Chow, S.K.; Lee, E.W.; Li, D.H. Short-Term Prediction of Photovoltaic Energy Generation by Intelligent Approach. *Energy Build.* **2012**, *55*, 660–667, doi:10.1016/j.enbuild.2012.08.011. [CrossRef]
6. Chow, C.W.; Belongie, S.; Kleissl, J. Cloud Motion and Stability Estimation for Intra-Hour Solar Forecasting. *Sol. Energy* **2015**, *115*, 645–655, doi:10.1016/j.solener.2015.03.030. [CrossRef]
7. Lorenzo, A.T.; Morzfeld, M.; Holmgren, W.F.; Cronin, A.D. Optimal Interpolation of Satellite and Ground Data for Irradiance Nowcasting at City Scales. *Sol. Energy* **2017**, *144*, 466–474, doi:10.1016/j.solener.2017.01.038. [CrossRef]
8. Li, Y.Z.; He, L.; Nie, R.Q. Short-Term Forecast of Power Generation for Grid-Connected Photovoltaic System Based on Advanced Grey-Markov Chain. In Proceedings of the 2009 International Conference on Energy and Environment Technology, Guilin, China, 16–18 October 2009; pp. 275–278, doi:10.1109/ICEET.2009.305. [CrossRef]

9. Ogliari, E.; Dolara, A.; Manzolini, G.; Leva, S. Physical and Hybrid Methods Comparison for the Day Ahead PV Output Power Forecast. *Renew. Energy* **2017**, *113*, 11–21, doi:10.1016/j.renene.2017.05.063. [CrossRef]
10. Inman, R.H.; Pedro, H.T.; Coimbra, C.F. Solar Forecasting Methods for Renewable Energy Integration. *Prog. Energy Combust. Sci.* **2013**, *39*, 535–576, doi:10.1016/j.pecs.2013.06.002. [CrossRef]
11. Dolara, A.; Leva, S.; Manzolini, G. Comparison of Different Physical Models for PV Power Output Prediction. *Sol. Energy* **2015**, *119*, 83–99, doi:10.1016/j.solener.2015.06.017. [CrossRef]
12. Tina, G.M.; Marletta, G.; Sardella, S. Multi-layer thermal models of PV modules for monitoring applications. In Proceedings of the 2012 38th IEEE Photovoltaic Specialists Conference, Austin, TX, USA, 3–8 June 2012; pp. 002947–002952, doi:10.1109/PVSC.2012.6318203. [CrossRef]
13. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
14. Pedro, H.T.; Coimbra, C.F. Assessment of Forecasting Techniques for Solar Power Production with No Exogenous Inputs. *Sol. Energy* **2012**, *86*, 2017–2028, doi:10.1016/j.solener.2012.04.004. [CrossRef]
15. Meyers, B.; Hoffimann, J. Short time horizon solar power forecasting. *Neural Netw.* **2017**, *3*, 2340.
16. Ineichen, P.; Perez, R. A New Airmass Independent Formulation for the Linke Turbidity Coefficient. *Sol. Energy* **2002**, *73*, 151–157, doi:10.1016/S0038-092X(02)00045-2. [CrossRef]
17. Holmgren, W.F.; Hansen, C.W.; Mikofski, M.A. Pvlib Python: A Python Package for Modeling Solar Energy Systems. *J. Open Source Softw.* **2018**, *3*, 884, doi:10.21105/joss.00884. [CrossRef]
18. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning: With Applications in R*, corrected at 8th printing ed.; Springer Texts in Statistics; Springer: New York, NY, USA; Heidelberg, Germany; Dordrecht, The Netherlands; London, UK, 2017.
19. Mellit, A.; Pavan, A.M. 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* **2010**, *84*, 807–821, doi:10.1016/j.solener.2010.02.006. [CrossRef]
20. Zamo, M.; Mestre, O.; Arbogast, P.; Pannekoucke, O. A Benchmark of Statistical Regression Methods for Short-Term Forecasting of Photovoltaic Electricity Production, Part I: Deterministic Forecast of Hourly Production. *Sol. Energy* **2014**, *105*, 792–803, doi:10.1016/j.solener.2013.12.006. [CrossRef]
21. Husein, M.; Chung, I.Y. Day-Ahead Solar Irradiance Forecasting for Microgrids Using a Long Short-Term Memory Recurrent Neural Network: A Deep Learning Approach. *Energies* **2019**, *12*, 1856, doi:10.3390/en12101856. [CrossRef]
22. Santosa, F.; Symes, W.W. Linear Inversion of Band-Limited Reflection Seismograms. *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 1307–1330, doi:10.1137/0907087. [CrossRef]
23. Hastie, T.; Tibshirani, R.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer Series in Statistics; Springer: New York, NY, USA, 2017.
24. Ho, T.K. Random Decision Forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282, doi:10.1109/ICDAR.1995.598994. [CrossRef]
25. Altman, N.S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.* **1992**, *46*, 175–185, doi:10.1080/00031305.1992.10475879. [CrossRef]
26. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780, doi:10.1162/neco.1997.9.8.1735. [CrossRef]
27. Chollet, F. Keras 2015. Available online: https://keras.io/ (accessed on 27 November 2019)
28. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org (accessed on 27 November 2019).
29. Bao, W.; Yue, J.; Rao, Y. A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long-Short Term Memory. *PLoS ONE* **2017**, *12*, e0180944, doi:10.1371/journal.pone.0180944. [CrossRef]

30. Tang, Y.; Xu, J.; Matsumoto, K.; Ono, C. Sequence-to-Sequence Model with Attention for Time Series Classification. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 503–510, doi:10.1109/ICDMW.2016.0078. [CrossRef]
31. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:cs/1412.6980.
32. Yang, D. A Guideline to Solar Forecasting Research Practice: Reproducible, Operational, Probabilistic or Physically-Based, Ensemble, and Skill (ROPES). *J. Renew. Sustain. Energy* **2019**, *11*, 022701, doi:10.1063/1.5087462. [CrossRef]
33. Rosenblatt, M. Remarks on Some Nonparametric Estimates of a Density Function. *Ann. Math. Stat.* **1956**, *27*, 832–837, doi:10.1214/aoms/1177728190. [CrossRef]
34. Raza, M.Q.; Nadarajah, M.; Ekanayake, C. On Recent Advances in PV Output Power Forecast. *Sol. Energy* **2016**, *136*, 125–144, doi:10.1016/j.solener.2016.06.073. [CrossRef]
35. Zhang, J.; Verschae, R.; Nobuhara, S.; Lalonde, J.F. Deep Photovoltaic Nowcasting. *Sol. Energy* **2018**, *176*, 267–276, doi:10.1016/j.solener.2018.10.024. [CrossRef]
36. Chu, Y.; Pedro, H.T.; Kaur, A.; Kleissl, J.; Coimbra, C.F. Net Load Forecasts for Solar-Integrated Operational Grid Feeders. *Sol. Energy* **2017**, *158*, 236–246, doi:10.1016/j.solener.2017.09.052. [CrossRef]
37. Elsinga, B.; van Sark, W.G. Short-Term Peer-to-Peer Solar Forecasting in a Network of Photovoltaic Systems. *Appl. Energy* **2017**, *206*, 1464–1483, doi:10.1016/j.apenergy.2017.09.115. [CrossRef]
38. Oneto, L.; Laureri, F.; Robba, M.; Delfino, F.; Anguita, D. Data-Driven Photovoltaic Power Production Nowcasting and Forecasting for Polygeneration Microgrids. *IEEE Syst. J.* **2018**, *12*, 2842–2853, doi:10.1109/JSYST.2017.2688359. [CrossRef]
39. Pedro, H.T.; Coimbra, C.F.; David, M.; Lauret, P. Assessment of Machine Learning Techniques for Deterministic and Probabilistic Intra-Hour Solar Forecasts. *Renew. Energy* **2018**, *123*, 191–203, doi:10.1016/j.renene.2018.02.006. [CrossRef]
40. Chu, Y.; Coimbra, C.F. Short-Term Probabilistic Forecasts for Direct Normal Irradiance. *Renew. Energy* **2017**, *101*, 526–536, doi:10.1016/j.renene.2016.09.012. [CrossRef]