Technologies For Information Systems

# T.I.S. Project

*Submitted By :*
Luca Dondoni

# Contents

## 1 Introduction

In this short report we are going to analyze and discuss the results obtained by working on a simple data-set, adopting different models to perform prediction tasks. In this first section the structure of the data-set as well as the objectives of this work are presented.
The full code we used to get the numerical results and figures that will follow is available here.

The data-set that is given presents the following structure (for clarity purposes only few samples are displayed):

| Day | Month | Year | Hour | Minutes | Floor-Nr | AvgT | AvgH | AvgP |
|---|---|---|---|---|---|---|---|---|
| 27 | 2 | 2,018 | 17 | 30 | 3 | 3.2 | 60 | 99 |
| 27 | 2 | 2,018 | 17 | 30 | 1 | 3.2 | 60 | 87 |
| 27 | 2 | 2,018 | 17 | 45 | 3 | 3 | 63 | 101 |
| 27 | 2 | 2,018 | 17 | 45 | 1 | 3 | 63 | 94 |
| 27 | 2 | 2,018 | 18 | 0 | 3 | 2.8 | 64 | 104.3 |
| 27 | 2 | 2,018 | 18 | 0 | 1 | 2.8 | 64 | 106.8 |
| 27 | 2 | 2,018 | 18 | 15 | 3 | 2.7 | 68.7 | 106 |
| 27 | 2 | 2,018 | 18 | 15 | 1 | 2.7 | 68.7 | 101.7 |

Each sample represents the average electric power consumption *AvgP* of a building (measured in Watt), given the floor number *Floor-nr*, at the given date and time. In particular, the samples were collected every quarter of an hour and only on the first and third floor of the building. In each sample we can also find the *AvgT* and *AvgH* values, that respectively denote the outside average temperature (measured in Celsius degrees) and the outside average humidity percentage. The whole data-set is composed of 145640 samples that correspond to slightly more than 2 years of measurements, from the $6^{th}$ of February 2017 to the $10^{th}$ of April 2019.
The objective of the work is to predict the average electric power that will be consumed in the following $n$ quarters of an hour on a specific floor, given the current average electric power consumption on that floor, the outside temperature and humidity, the date and the current hour.

## 2 Data Pre-Processing

Given the specification just mention in Section 1 we have to re-arrange the data-set to properly obtain the desired results.

## 2.1 Duplicates Sample Removal

We noticed that the data-set contains some samples that were collected two or more times, before proceeding with the analysis it is necessary to remove the copies and keep just one of them. This job can be easily done by calling the method remove_duplicated_samples() present in the class DataProcessor.

## 2.2 Adding Next Value Of A Given Column

The basic case of our problem is to predict the value of $AvgP$ of the next sample (in other words, we want to predict the average power that will be consumed in the next quarter of an hour). In order to simplify our work with the data-set we can add another column which will contain, for each sample, the value of $AvgP$ of the following one. The method add_next_values(column="AvgP", steps_ahead=1) present in the class DataProcessor will take care of this job for us.

## 2.3 Adjusting Values Taken From Other Samples

Adding values from the following samples, like we just did in the previous subsection, can give raise to some inaccuracies in case missing values are present. In simple words, consider a sample $X_k$ collected at 17.30 and the following sample $X_{k+1}$ collected at 22.00 on the same date: is now clear that assigning the next value of $AvgP$ for the sample $X_k$ equal to the current value $AvgP$ of the sample $X_{k+1}$ is incorrect, since they are not as far as we expect (a quarter of an hour), but they are actually 4 hours and an half far in this example. To mitigate this issue without adding new samples, we can both interpolate or extrapolate these critical values.

1. If the gap (that is measured in terms of number of missing values) between the current sample $X_k$ and the following one $X_{k+1}$ is sufficiently small, interpolating these critical values with the ones of the previous and the following samples can be a good idea;

2. If the gap between the current sample $X_k$ and the following one $X_{k+1}$ is reasonably too high, extrapolating these critical values with the ones relative to the two previous samples is better;

3. Summarizing, small gap $\implies$ interpolation will get a bigger weight on the result w.r.t. extrapolation; big gap $\implies$ extrapolation will get a bigger weight on the result w.r.t. interpolation.

According to point 3, we can compute the "adjusted" value $V_k$ belonging to a feature $V$ of a given sample $X_k$ as a weighted mean between the extrapolation and interpolation approaches:

$$V_k = W_i * Inter(V_{k-1}, V_{k+1}) + W_e * Extra(V_{k-2}, V_{k-1})$$

where the functions $Inter(.)$ and $Extra(.)$ give us the interpolated and extrapolated current value $V_k$ basing on the values, of the same feature, belonging to the near samples. These two contributions are then weighted according to the length of the gap between $X_k$ and $X_{k+1}$ by means of the weights $W_i$ and $W_e$. To compute the latter weights we can use the following formulas:

$$W_i = \frac{1}{1 + e^{gap-2}}$$

$$W_e = \frac{1}{1 + e^{2-gap}}$$

As expected, $W_i$ tends to zero if the gap is big, while $W_e$ tends to one. On the other hand, $W_i$ tends to one if the gap is small, while $W_e$ tends to zero. Finally, if the gap is exactly equal to two, the two contributions are equally weighted. (Obviously, $W_i + W_e = 1$).

The DataProcessor will perform the whole task for us to adjust the column added in the previous sub-section by calling the corresponding method Adjust_next_values(column="AvgP_next_value_1").

## 2.4 Outliers Removal

Outliers are present in almost all the data-sets. The techniques that can be used to deal with them are a lot, in this project techniques like the one just described in the previous sub-section or merely the interpolation will be used. To state if a value $A_k$ relative to the feature $A$ of the $k^{th}$ sample is an outlier we can:

1. Recognize $A_k$ as a "certain" outlier, an average temperature AvgT equal to -400° is clearly an outlier, just to make an example;
2. Check if the value $A_k$ is enough below the $25^{th}$-quantile of the feature $A$ or enough above the $75^{th}$-quantile of the feature $A$. A possible approach could be to find the inter-quantile space as $I_q = A.quantile(0.75) - A.quantile(0.25)$, then use this value to state if $A_k$ is an outlier as follows:
   - **If** $A_k < A.quantile(0.25) - \alpha * I_q \rightarrow A_k$ is an outlier;

- **Else If** $A_k > A.quantile(0.75) + \alpha * I_q \rightarrow A_k$ is an outlier;
- **Else** $A_k$ is not an outlier.

By tuning the parameter $\alpha$ we can change our tolerance in marking values as outliers.

In this project we are going to correct the values of the outliers by applying the same technique described in the subsection 2.3, by adjusting only those values that are "sure" outliers. In the provided data-set the "sure" outliers we found were $AvgT < -50°$, $AvgH < 0$, $AvgP = 0$.

## 2.5 One-Hot

Some of the features of this data-set are characterized by a cyclical behaviour, like the *Month* feature. For the models that we will use for prediction a *Month* value equal to 12 seems very far from a *Month* value equal to 1, when they actually are not. To remove this problem we can apply the one-hot technique by selecting a feature name and calling the corresponding method on the DataProcessor Class: column_to_one_hot(column="Month", possibilities=...), where the possibilities parameters specifies all the possible values that can be assumed by that feature, for the *Month* feature we pass [1, 2, ..., 12].

## 2.6 Data-Set Enrichment

We can also add some other information to our current data-set that are not present, but can be inferred. One of them can be the addiction of the days of the week. Since we know that the $6^{th}$ February 2017 was Monday, we can add all the other days only by looking at the dates. This job can be easily done by calling the method add_days_of_the_week() of the class DataProcessor that will add a new column to the data-set containig the corresponding day of the week. A similar could have been done by adding, for each day, the average power consumption for that day, obtained by averaging all the values of the feature *AvgP*. We discarded this option since a day is composed by 24 hours * 4 quarter of an hour = 96 samples, therefore it would have been not so meaningful for our models (by the way, the method that computes this result is still present in the class DataProcessor).

# 3 Features Selection

In this section we are going to select the features that will be used by our models to make the predictions. In particular we are looking for features that are somehow correlated with the values that we want to predict, that is *AvgP*.

## 3.1 *AvgP* vs *AvgT*

Since the data-set is describing the average electric power consumption in a building, we expect that the lower is the outside average temperature, the higher will be the power consumption, since there is the need to warm up the internal rooms. Similarly, the hotter is the outside, the higher the average power consumption will be in order to properly cool the facility. Figure 1 and Figure 2 show respectively how the average consumed power varies as the outside average temperature changes on floor 1 and floor 3. As can be noticed on the floor 3 there is the evidence of a human behaviour, characterized by peaks in power corresponding to very low and pretty high outside temperature. On the other hand, on the floor 1 the average consumed power is almost constant, except for very low temperatures (please note that in Figure 1 the *AvgP* values range between 60 and 80 Watt, unlike on the other floor, where the values range in a wider interval. This fact confirms that on the floor 3 there is more human interaction with the electric system, which causes more variance in the values of *AvgP*).

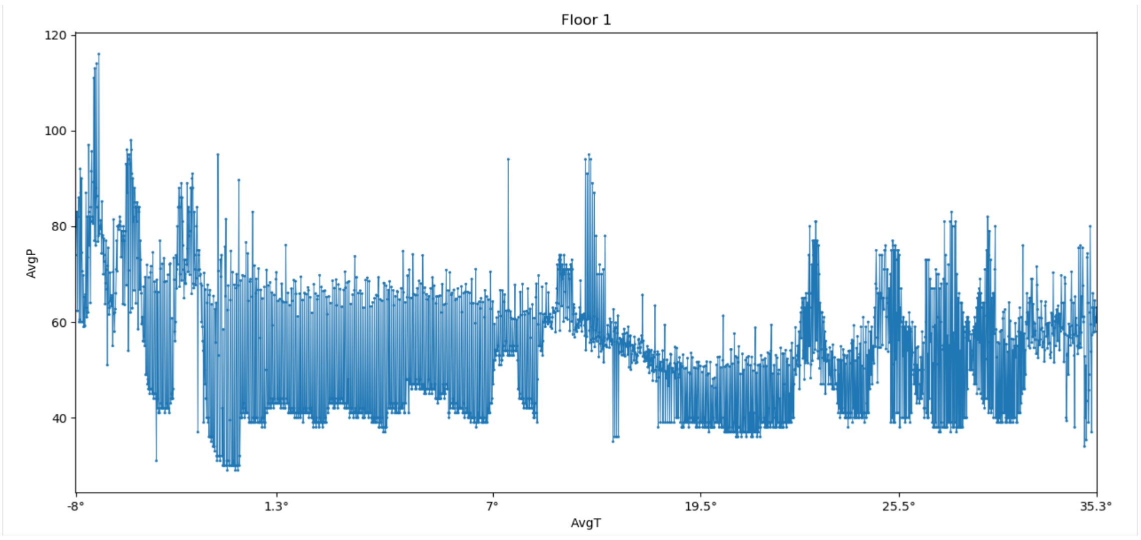

**Figure 1:** *The average consumed power as a function of the outside average temperature captured on the floor 1 and summarized by a point-plot.*

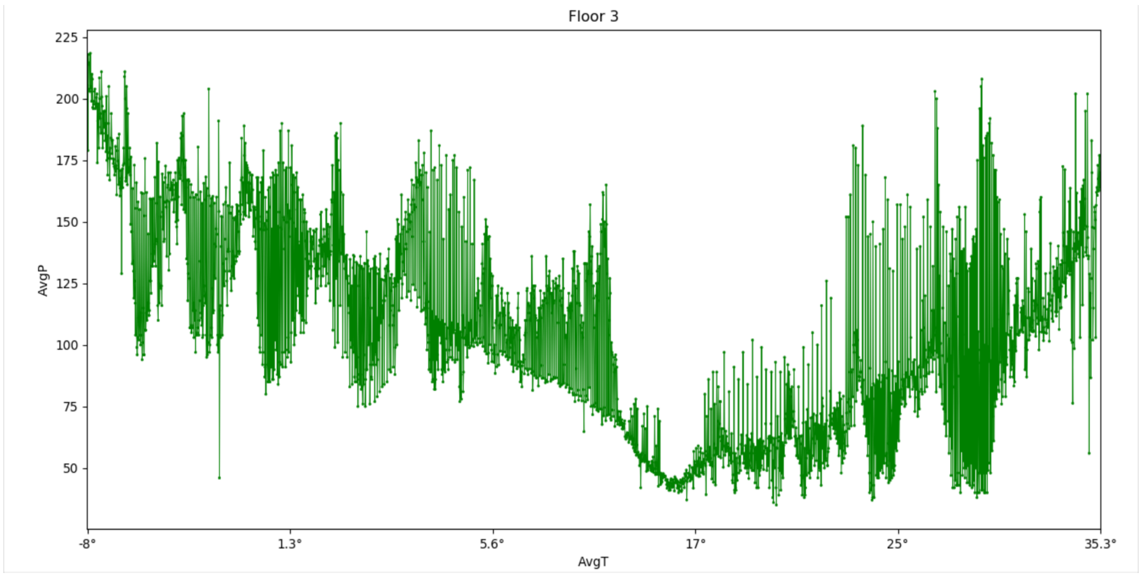

**Figure 2:** *The average consumed power as a function of the outside average temperature captured on the floor 3 and summarized by a point-plot.*

In both cases, the temperature can be used by the models to better infer the feature that has to be predicted, mostly for floor 3.

## 3.2 *AvgP* vs *Month*

As we expect, in summer and winter month the consumed average power should be higher. Figure 3 and Figure 4 summarize the values of *AvgP* for each *Month*.

- **Floor 1**: As can be derived from Figure 1, the only moment in which we can see a greater average power consumption is when the temperature is low. This is reflected in the corresponding box-plot by slightly higher values in correspondence of colder months.
- **Floor 3**: In this other scenario, we can see (as expected by looking at Figure 2) that also during hotter months the consumed power is higher.

In both cases, also the *Month* could be a meaningful input for our models, given the correlation we just found out with this plots.

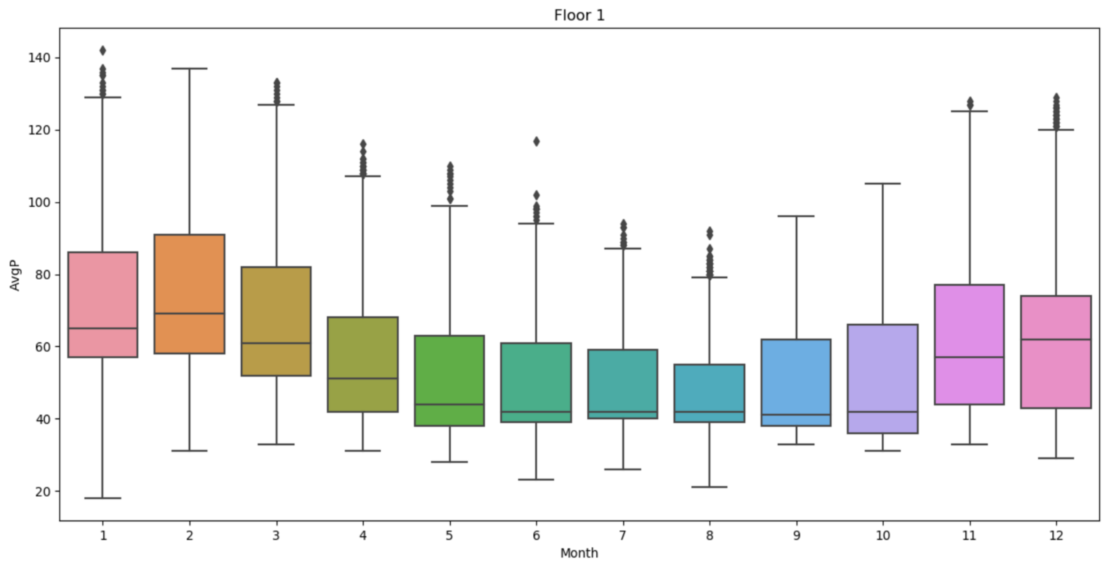

**Figure 3:** *The average consumed power as a function of the Month captured on the floor 1 and summarized by a box-plot.*

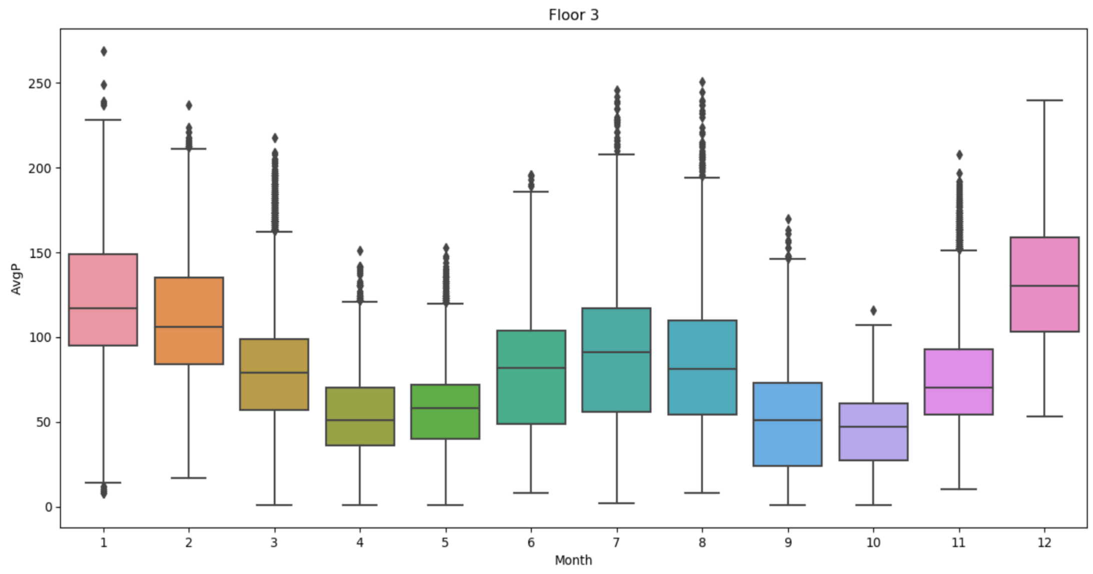

**Figure 4:** *The average consumed power as a function of the Month captured on the floor 3 and summarized by a box-plot.*

## 3.3 *AvgP* vs *AvgH*

In this sub-section we are going to use point-plots to check if there is an actual correlation between the average consumed power and the outside humidity percentage.

Figure 5 and Figure 6 shows the just mentioned plots. In the former we are not able to spot any evident correlation between the two features; in other words, except for few peaks, the shape is almost flat.

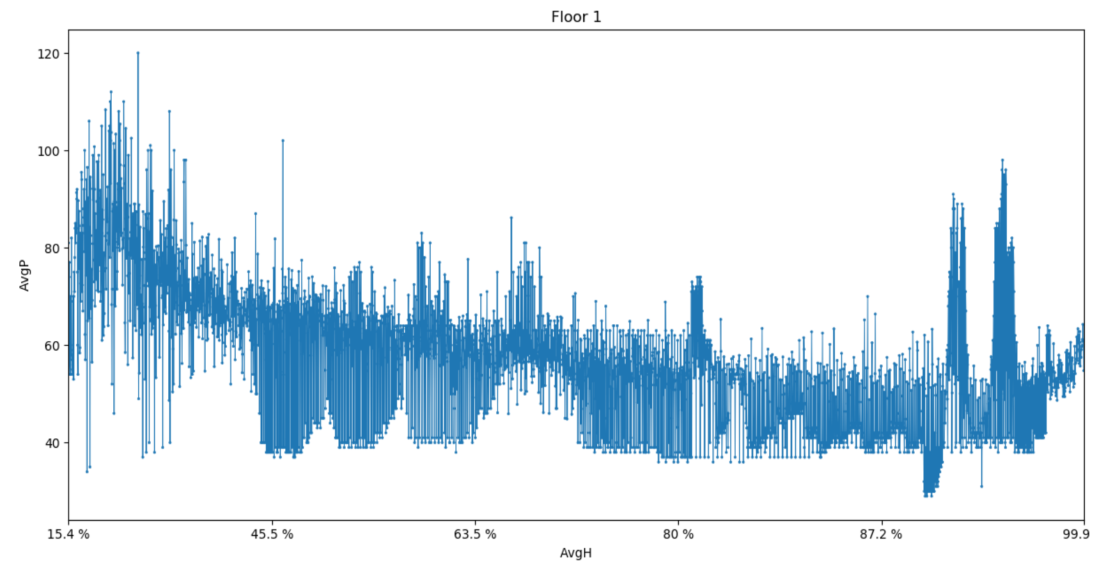

**Figure 5:** *The average consumed power as a function of the outside average humidity percentage captured on the floor 1 and summarized by a point-plot.*

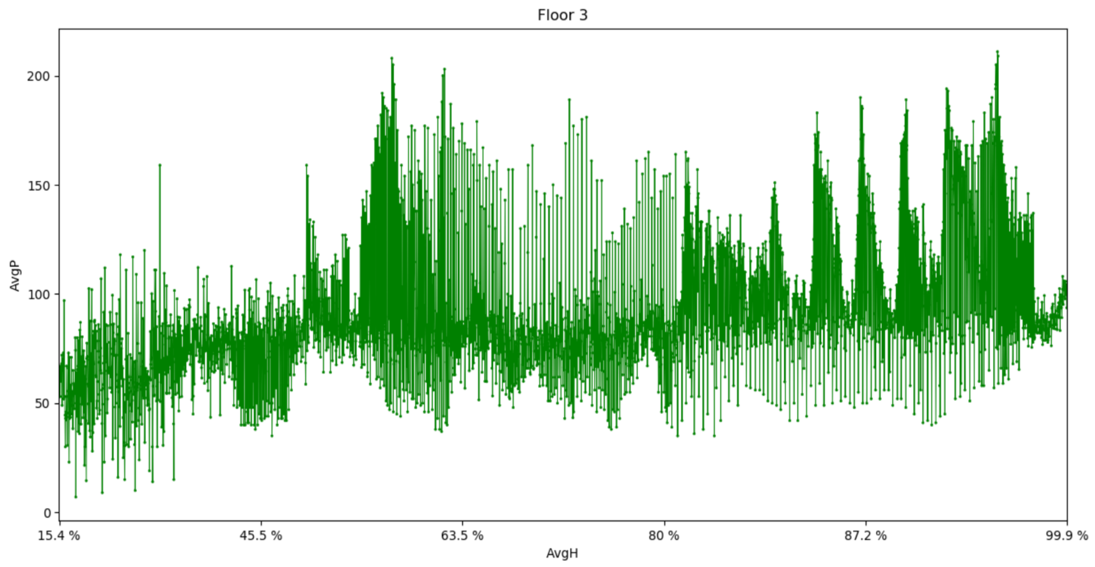

**Figure 6:** *The average consumed power as a function of the outside average humidity percentage captured on the floor 3 and summarized by a point-plot.*

In the latter we can spot an higher variance in terms of values assumed by *AvgP* but, likewise the previous case, no evidence of a clear correlation between the two features. For these reasons we decided to discard *AvgH* as an input feature for our models (we tried also with it, but the results we obtained were way worst, confirming what we described here).

## 3.4 *AvgP* vs *Hour*

We may think that the average consumed power is higher in certain moment of the day rather than others. To eventually spot this behaviour we can use box-plots to see how *AvgP* values are distributed for each *Hour* of the day.

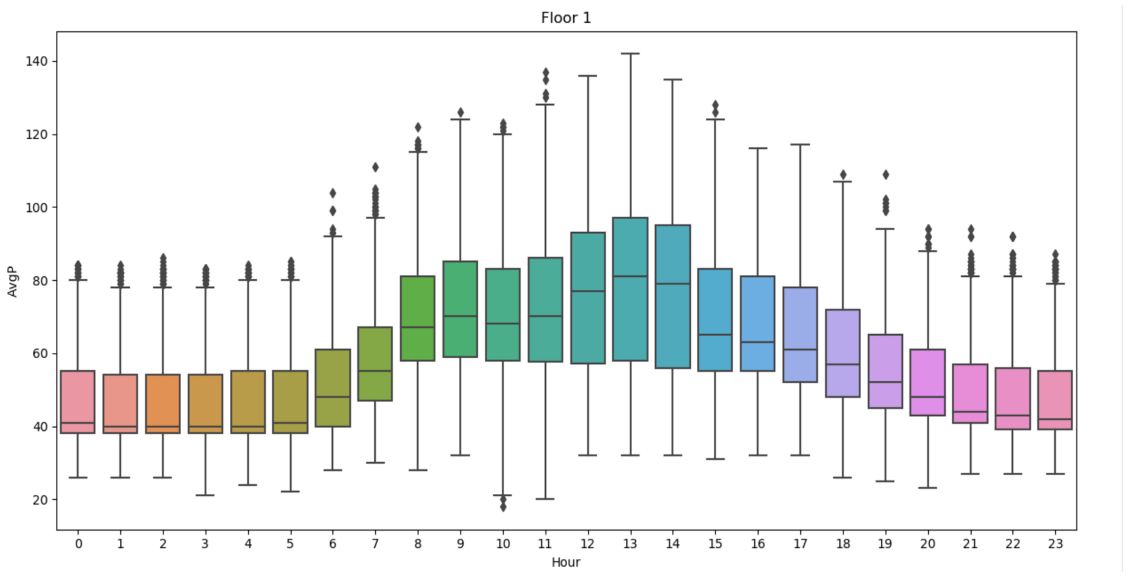

**Figure 7:** *The average consumed power as a function of the day hour, captured on the floor 1 and summarized by a box-plot.*

In both Figure 7 and Figure 8 we can see an evident correlation between the two features. More precisely:

- On the floor 1 the higher values of average consumed power concentrate between 8 a.m. and 6 p.m.. Most probably the electric system is turned of on this floor during the night;
- On the floor 3 the higher values of *AvgP* are located between 7 a.m. and 11 a.m., 4 p.m. and 8 p.m., which is something that recall a typical working schedule.

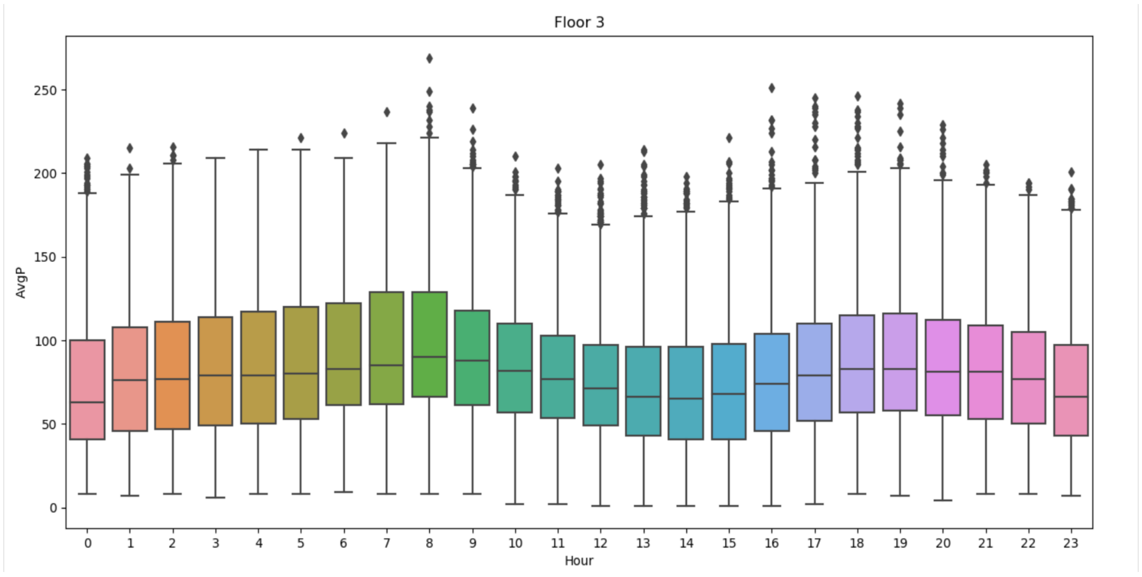

**Figure 8:** *The average consumed power as a function of the day hour, captured on the floor 3 and summarized by a box-plot.*

The *Hour* feature can then be used to more properly predict the values of the feature *AvgP*, therefore it will be part of the inputs for our models.

## 3.5 *AvgP* vs *Day Of The Week*

It could be possible that some day of the week the average consumed power is higher w.r.t. other days. We can then use the days of the week that we added to our dataset in Subsection 2.6. Figures 9 and 10 shows how the average consumed power is distributed over the days. For the third floor there is no correlation, whereas there is a low electric usage during the week end for the first floor. In conclusion, we will use the one-hot days-of-the-week input only in case we are predicting the *AvgP* values for the first floor.

# 4 Basic Prediction

In this section we will take a look at the result obtained by using different kind of models. For each model we are going to use a set of the inputs discussed in the previous section in order to predict the average consumed power after a quarter of an hour. We will use two models per kind, one for each floor.

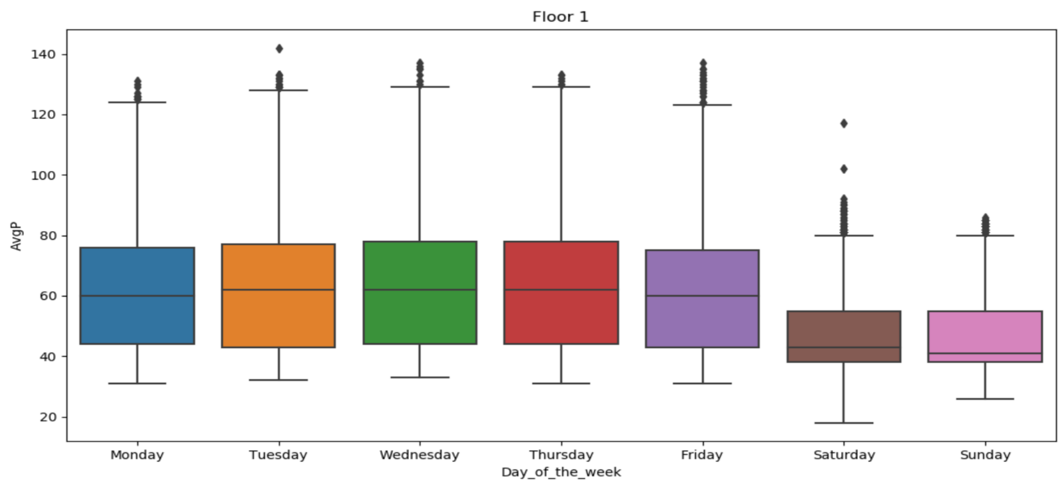

**Figure 9:** *The average consumed power as a function of the day of the week, captured on the floor 1 and summarized by a box-plot.*

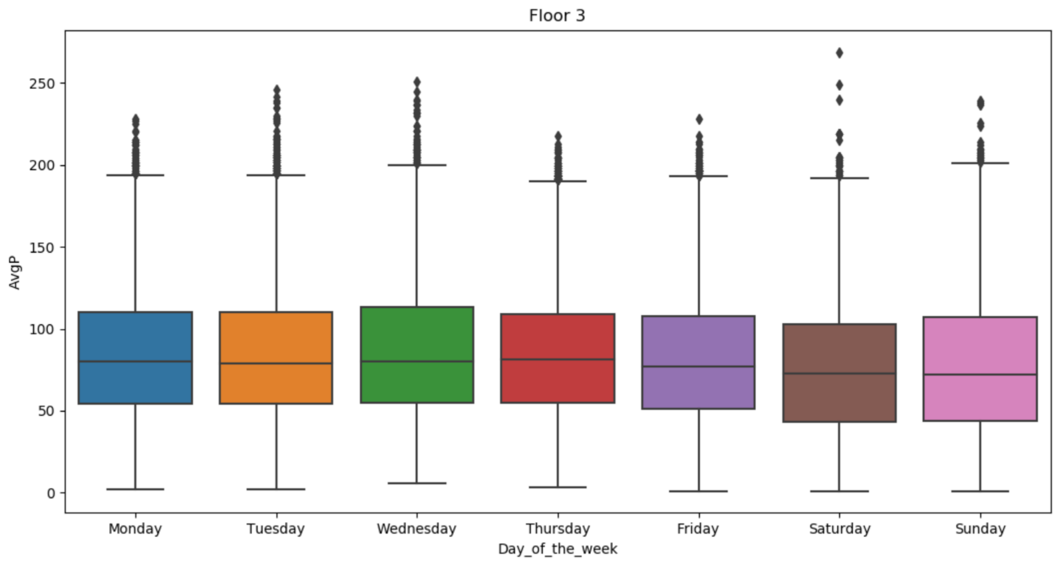

**Figure 10:** *The average consumed power as a function of the day of the week, captured on the floor 3 and summarized by a box-plot.*

## 4.1 Polynomial Regression

In this first subsection we are going to analyze the result obtained by a polynomial regression, with different polynomial degrees. We built two separated models for the two different floors. A generic input sample $X_k^i$ for our $3^{rd}$-floor-model has the following structure:

$$X_k^i = [Hour_k,\ AvgT_k,\ AvgP_k,\ [Month\_1\_Hot]_k]$$

whereas for the second model this other one:

$$X_k^i = [Hour_k,\ AvgT_k,\ AvgP_k,\ [Month\_1\_Hot]_k,\ [Week\_days\_1\_hot]_k]$$

at both we associate the relative output that the model will learn to predict:

$$X_k^o = [AvgP_{k+1}]$$

where $[Month\_1\_Hot]_k$ and $[Week\_days\_1\_hot]_k$ are sets of features derived by the one-hot technique described in Subsection 2.5 for respectively the *Month* and *Day_of_the_week* features.

| **Experimental Results Polynomial Regression Basic Prediction** | | | | |
|---|---|---|---|---|
| **Poly Degree** | **Ridge-Lasso** | $\alpha$ | **Floor 1-M.a.p.e.** | **Floor 3-M.a.p.e.** |
| 1 | — | — | 2.936 % | 8.570 % |
| 2 | — | — | 3.041 % | 8.232 % |
| 2 | Lasso | 1.0 | 2.940 % | 8.240 % |
| 2 | Ridge | 0.75 | 3.015 % | 8.229 % |
| 2 | Ridge | 1.0 | 3.016 % | 8.229 % |
| 3 | Lasso | 1.0 | 3.129 % ! | 8.426 % ! |
| 4 | Lasso | 0.5 | 3.068 % | 8.129 % ! |

**Table 1:** *Experimental result obtained by means of a polynomial regression with different settings. If column "ridge-lasso" is not present we mean simple polynomial regression without any regularization technique. This table refers to the case in which the training set is the 80% and the validation set is the remaining 20% of the original data-set. The symbol "!" is present when the model weights did not converge to a local minimum by using gradient descent.*

Table 1 shows the mean absolute percentage error on the validation set (20 % of the whole data-set) for both models with different configurations. The better performance for the floor 3 is achieved by the most complex model. Moreover, the latter's weights did not converge to a local minimum, therefore the error could be further reduced. As could have been expected, the m.a.p.e. error on the first floor is way lower if compared with the one of the third floor; this is due to the fact that the *AvgP* feature of the first floor is characterized by a less variance and it presents a semi-constant behaviour in case of mid and high temperatures. For the first floor, then, even a simple model performs well, for the third floor a slightly more complex model is required.

## 4.2 Gradient Boosting Regression

The second kind of model we are going to use is the gradient boosting regressor. We define the two different model inputs and the corresponding output as we have done in Subsection 4.1. The following table reports the result obtained with this kind of model with different configurations:

| **Experimental Results Gradient Boosting Regressor Basic Prediction** | | | | |
|---|---|---|---|---|
| **Num. of Trees** | $\nu$ | **Max Depth** | **Floor 1-M.a.p.e.** | **Floor 3-M.a.p.e.** |
| 50 | 0.1 | 8 | 2.956 % | 8.444 % |
| 100 | 0.1 | 8 | 3.034 % | 8.281 % |
| 100 | 0.2 | 12 | 3.104 % | 8.547 % |
| 100 | 0.5 | 10 | 3.253 % | 8.984 % |
| 500 | 0.1 | 8 | 3.270 % | 8.667 % |
| 500 | 0.2 | 16 | 3.403 % | 8.879 % |
| 1500 | 0.1 | 8 | 3.462 % | 9.116 % |
| 5000 | 0.1 | 8 | 3.764 % | 10.088 % |

**Table 2:** *Experimental result obtained by means of a gradient boosting regressor with different configurations. This table refers to the case in which the training set is the 80% and the validation set is the remaining 20% of the original data-set.*

As can be noticed, even with a very simple model (only 50 trees) we can get sufficiently good performances for the first floor. As the complexity of the model increases, the mean absolute percentage errors on the two floors increases, but the result we obtain are acceptable anyway: in some sense the models are not overfitting. This is due to the internal structure of a g. boosting regressor: even if a single tree overfits on a particular set of samples, its contribution is weighed (by means of the parameter $\nu$ in the table) and the final prediction is given by the contributions of all the trees, including the other that are unlikely overfitting on the same set of samples.

## 4.3 Random Forest Regression

In this subsection we will use a third kind of model, a random forest regressor, to predict the value $AvgP_{k+1}$ of each sample. As usual, we define the two different inputs and the corresponding output as we have done in Subsection 4.1. Table 3 shows the experimental result given by a random forest regressor with different configurations (number of the trees, bootstrapped data-set, max depth of every tree).

| **Experimental Results Random Forest Regressor Basic Prediction** | | | | |
|---|---|---|---|---|
| **Num. Trees** | **Bootstrap** | **Max Depth** | **Fl. 1-M.a.p.e.** | **Fl. 3-M.a.p.e.** |
| 100 | Yes | 8 | 3.033 % | 8.225 % |
| 100 | No | 8 | 3.211 % | 8.433 % |
| 500 | Yes | 12 | 3.189 % | 8.199 % |
| 500 | No | 12 | 3.624 % | 8.770 % |
| 1000 | Yes | 8 | 3.036 % | 8.232 % |
| 1000 | Yes | 16 | 3.355 % | 8.445 % |
| 1000 | No | 16 | 4.303 % | 9.887 % |

**Table 3:** *Experimental result obtained by means of a random forest regressor with different configurations. This table refers to the case in which the training set is the 80% and the validation set is the remaining 20% of the original data-set.*

As expected, the more complex is the model, the less accurate the predictions will be. In particular, we can see that without bootstrapping the data-set the performance of both models get worse. This can be explained by thinking about how the trees are built.
Without the bootstrapping step, all the trees will be built starting from the same data-set; since the criterion to evaluate the quality of a split of an internal node (mean squared error criterion in case of a random forest regressor) is the same for all the trees, all of them will have pretty much the same structure (node splits, leaves...). As a consequence, we are gaining precision since we are building trees with the best structure for the training set, but we are loosing generality on the validation set $\rightarrow$ the validation error will increase.
With the bootstrapping step every tree is built starting from a different data-set; therefore they will have an average different structure and, as a consequence, they can keep a level of generality to avoid overfitting on the validation set $\rightarrow$ the validation error will decrease. Obviously, a model which is to complex for the task will end up to overfit on the training-set anyway.

## 4.4 Long Short Term Memory Prediction

In this Subsection we will deal with a different approach to make predictions. Unlike what the previous models do, with LSTM we are not mapping a set of inputs to an output, but we are trying to predict the next element of one or more time series given as inputs. We can think about an LSTM network as a model that takes as input a sequence of ten numbers and tries to predict the next one, basing on a training-set from which it has learnt how to do it. In this new scenario we define as our generic input sample $X_k^i$ a set of three sequences of $\delta + 1$ elements:

1. A sequence of the last $\delta + 1$ average consumed powers (that are, the average consumed powers in the last $\delta + 1$ quarters of an hour);
2. A sequence of the last $\delta + 1$ average outside temperatures;
3. A sequence of the last $\delta + 1$ hours at which the values in the two previous sequences were collected.

The last element of each sequence is the current value of *AvgP, AvgT* and *Hour*. The structure of each input for our network is then the following:

$$X_k^i = \begin{bmatrix} AvgP_{k-\delta} & AvgP_{k-\delta+1} & ... & AvgP_{k-1} & AvgP_k \\ AvgT_{k-\delta} & AvgT_{k-\delta+1} & ... & AvgT_{k-1} & AvgT_k \\ Hour_{k-\delta} & Hour_{k-\delta+1} & ... & Hour_{k-1} & Hour_k \end{bmatrix}$$

At each input we can associate the corresponding output that the model will learn to predict:

$$X_k^o = [AvgP_{k+1}]$$

Given this specification we can try to improve a little bit the results obtained by the previous approach.

| **Experimental Results LSTM Basic Prediction** | | | | | | |
|---|---|---|---|---|---|---|
| **Hid. Layers** | **LSTM nr.** | **Batch** | **Epochs** | $\delta$ | **Floor 1** | **Floor 3** |
| 2 | [90, 90] | 100 | 100 | 10 | 2.651 % ! | 7.367 % ! |
| 1 | [10] | 100 | 100 | 10 | 2.869 % | 7.546 % |
| 3 | [20, 20, 20] | 100 | 100 | 5 | 2.854 % ! | 7.306 % ! |
| 3 | [32, 16, 8] | 200 | 200 | 8 | 2.665 % | 7.426 % |

**Table 4:** *Experimental result obtained by means of a LSTM network with different configurations. This table refers to the case in which the training set is the 80% and the validation set is the remaining 20% of the original data-set. The error values marked with the "!" symbol have been obtained with the Early Stopping regularization technique.*

Table 4 shows the experimental results obtained by using different network structure. Working with time series provides an improvement in terms of validation error w.r.t. the other technique. As can be noticed, even the results that are not subject to any regularization technique appear to be acceptable. This can be due to the fact that, during the training phase, even using a very complex model, the error on the training set decrease very slowly and at a given point stop decreasing. In some senses even a complex model is not able to reduce the error on the training set close to zero. This may be caused by a wide inconsistency in the training-set or to some outliers that we did not recognize.

## 5 Chain Predictions

In this and the following last two sections we will make the step that will complete our work. So far we have solved the simple problem to make a single prediction of the value of *AvgP* after a quarter of an hour. With a simple process we can extend this approach to a more general one, which allows us to predict for the following $N$ quarters of an hour. Given a generic sample $X_k$:

1. As we have done so far we can predict the value of *AvgP* for the next quarter of an hour $\rightarrow AvgP^*_{k+1}$;
2. Given the current date and time we can infer the *Hour* after a quarter of an hour $\rightarrow Hour^*_{k+1}$;
3. Given the current date and time we can infer the *Month* after a quarter of an hour $\rightarrow Month^*_{k+1}$
4. By putting beside the original model another model that, starting from the current sample $X_k$, is able to predict the value of *AvgT* after a quarter of an hour, we can also obtain the value $\rightarrow AvgT^*_{k+1}$.

We are then able to predict the entire following sample $X^*_{k+1}$:

$$X^*_{k+1} = [Hour^*_{k+1},\ AvgT^*_{k+1},\ AvgP^*_{k+1},\ [Month\_1\_Hot]^*_{k+1}]$$

By recursively repeating this process $N$ times we can predict the next $N$ values for *AvgP* obtaining a chain of predictions:

$$[AvgP^*_{k+1},\ AvgP^*_{k+2},\ AvgP^*_{k+3},\ ...,\ AvgP^*_{k+N}]$$

and finally by comparing this $N$ predictions with the real values for *AvgP* in the validation set we can estimate our validation error (Please note that we don't need to train again the models to perform this process, since the models just have to be able to predict the next value of *AvgP*. We have instead to train new models to predict also the temperature after a quarter of an hour).

## 6 Chain Predictions With LSTM Networks

In this last Section we will explain how to deal with chain predictions in case we are working with time-series and LSTM networks. Unlike what we did in Section 5 to perform this kind of task, the approach here is a little more complex. Although also here we are able to predict the next value of *AvgP*, *AvgT* and *Hour* to entirely predict the next sample, we have to take into account that every sample, in this scenario, keeps also track of the past values of *AvgP*, *AvgT* and *Hour*. Therefore, when we build a new sample, the sample that will follow it has to keep track of the former's values. To clearly understand of how chain prediction is done for the time series case please refer to Figure 11.

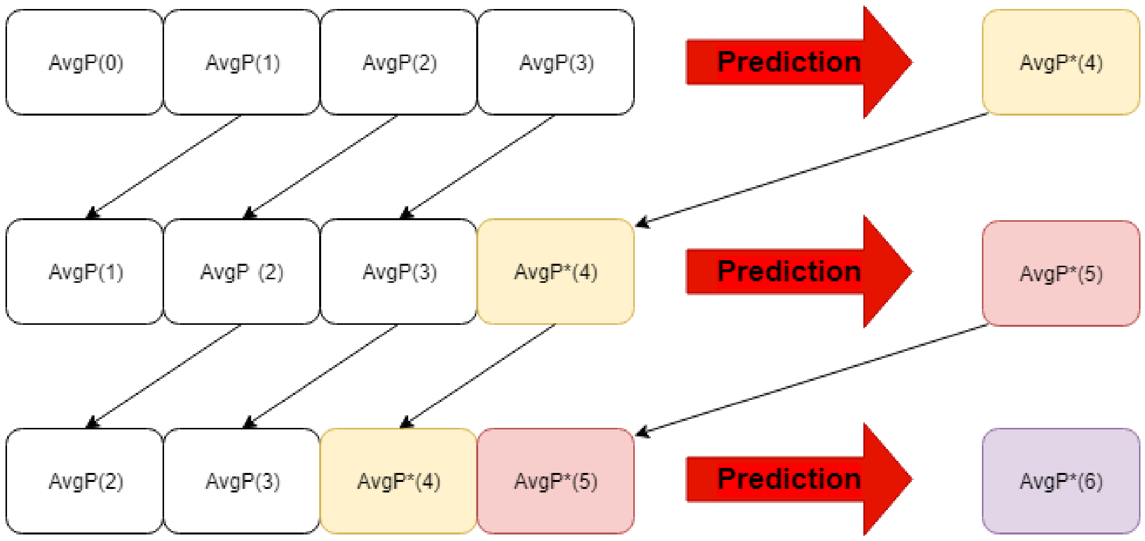

**Figure 11:** *Chain prediction process in case of time series analysis. In particular, here we can see a chain prediction of three elements for the time series of avgP. In pratice this process is done for each input series of the network.*

## 7 Chain Prediction Results

Now we are ready to make a comparison with the different kind of models we presented so far in terms of chain predictions capability. Figure 12 shows how the mean absolute percentage error varies as the number of chain predictions we perform changes. As can be noticed the LSTM predictor performs very well in the beginning, while in the end it overcomes the error of its competitors.

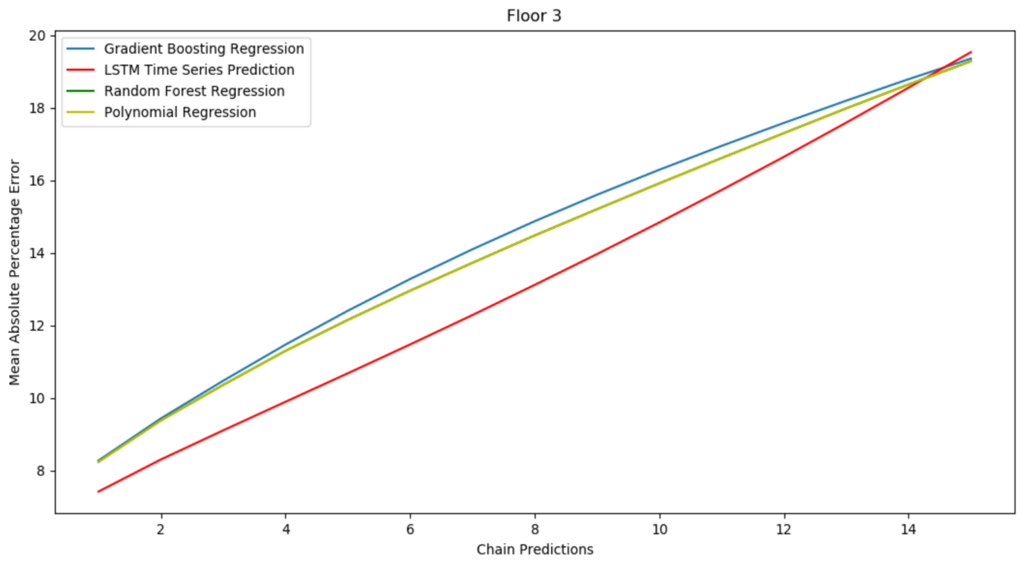

**Figure 12:** *Mean absolute percentage error as a function of the number of chain predictions concerning the floor 3. The performances of the random forest regressor and the polynomial regressor are the same.*

This can be explained by thinking about how the process of chain prediction works for time series: recalling Figure 11 it is clear that the more chain predictions we perform, the more the following predicted sample's series will contain values that are affected by an accumulated error. If the LSTM network is trying to predict the next element of a sequence that is full of partially wrong values, the error will increase; whereas if the time series contains only some element that are subject of a partial error the prediction will be more accurate. Since the LSTM network we used to enrich Figure 11 works with time-series which are 10 elements long, chain predictions longer than 10 elements cause the model to get worse results.
Another option could have been to design the LSTM network such that its outcome is not a single value, but a set of $K$ values that represent the prediction of the average power consumption for the next $K$ quarters of an hour, by doing this we avoid the error to accumulate by perform chain predictions.
With this setting the best performances are achieved by both the Random Forest and the Polynomial models, since they present more scalability even with long chain predictions.

Figure 13 shows the same results for what concerns the floor 1. The performances of a LSTM network here are the worst. This can be due to the fact that it is a very complex model (we used two layers of 90 LSTMs each with sequences of 11 elements as input) to be used for this simpler task. The performances of the other approaches

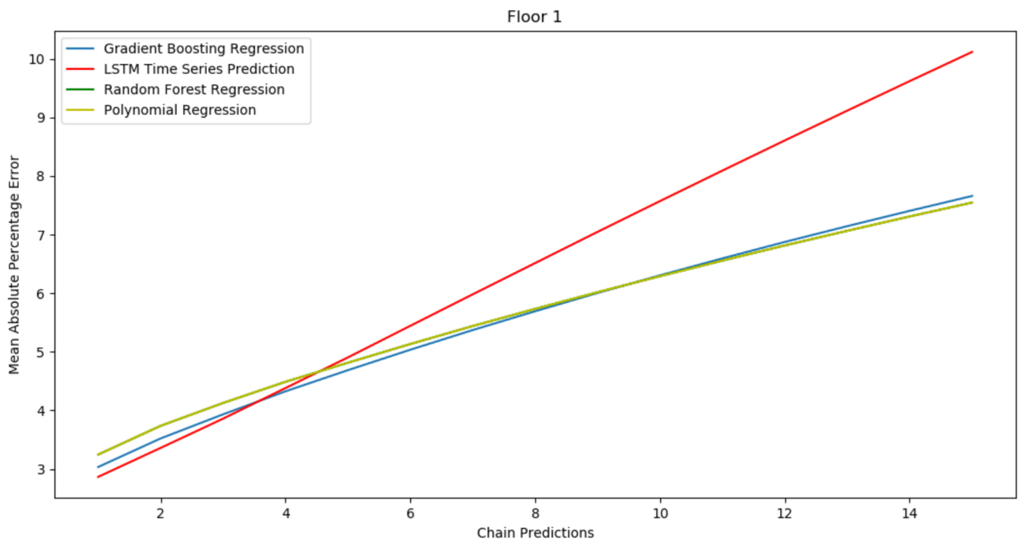

**Figure 13:** *Mean absolute percentage error as a function of the number of chain predictions concerning the floor 1. The performances of the random forest regressor and the polynomial regressor are the same.*

remanin a valid options to be adopted for chain predictions of this floor.