Performance Evaluation, Homework 4

Joël M. Fonseca 227334, Francis Damachi 217575

April 15, 2019

1 Introduction

As for any time series forecasting project, it is important to understand the nature of data we are dealing with by identifying the parameters that best characterize it. On one hand, this allows a good understanding of the time series while on the other hand, it allows the choice of certain models to be justified or the results of others to be better understood.

1.1 Data exploration

Figure 1 shows the entire time series. It is composed of hourly measurements of the energy consumption taken from October 5^{th} 00:00 of 2014 until May 31^{th} 23:00 of 2015. This corresponds to a signal composed of 5736 data points.

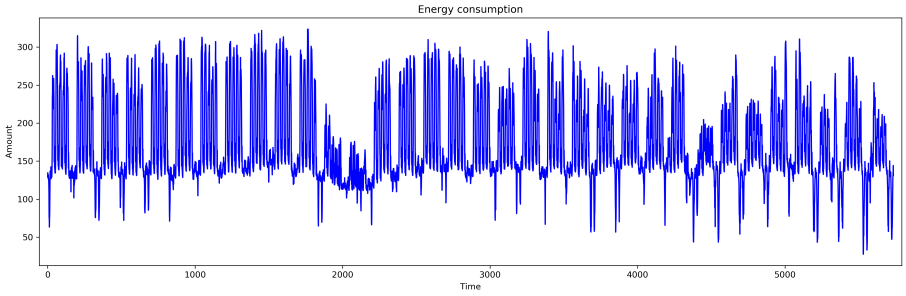

Figure 1: Complete time series visualization.

We can observe a change in consumption on public holidays or during more traditional holidays like Christmas or Easter, corresponding roughly to timestamps 2000 and 4500 respectively. Figure 2 shows a zoomed version of the signal for the month of October. Notice the difference between weekdays and weekend days.

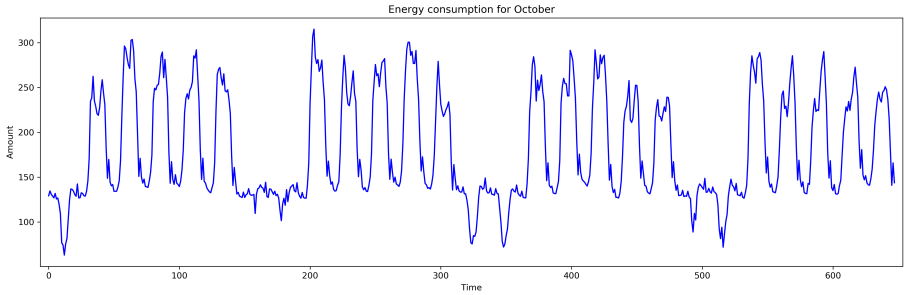

Figure 2: October time series visualization.

Already from the nature of the data and from the frequency of the measurements we can tell that two frequencies will be of interest: 24 and 168. The first frequency comes from the fact that we have hourly measurements and the second one simply corresponds to the duration of one week (7x24). These frequencies can characterize the seasonality component of the time series. Indeed, when dealing with such data one always tries to decompose it with three main components: trend, seasonality (and/or cycles) and noise. Figure 3 shows the classical additive seasonal decomposition of the signal with the frequency of the seasonality set to $f = 168$.

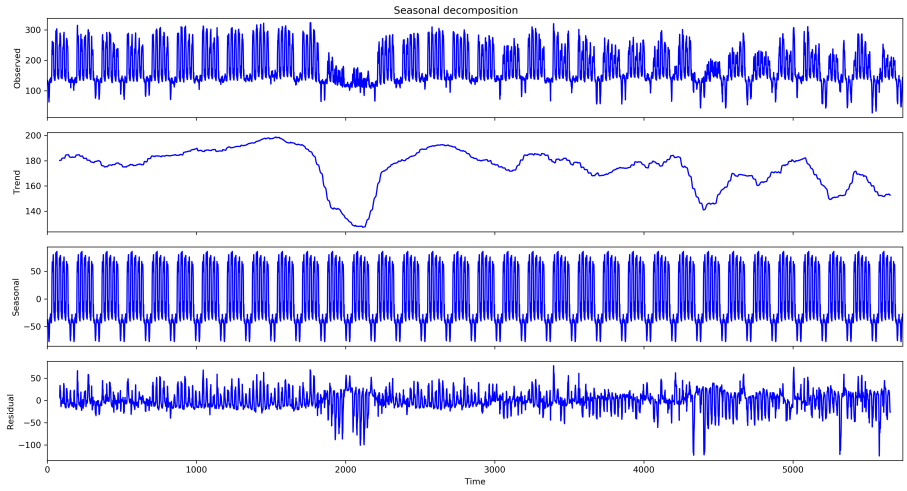

Figure 3: Seasonal decomposition using additive model and f=168.

We can see that this decomposition is based on some assumptions that do not necessarily match with the behaviour of our data. First, the trend estimate tends to over-smooth rapid spikes in the data. Again, if we look at Christmas and Easter for example, we can observe large values in the residuals at those particular times. Second, we assume that the seasonal component is constant,

this might be wrong. As we just said, time series can be particularly be unusual during a small period of time. Hence, we see that this decomposition is not robust enough to deal with these kinds of values. More sophisticated techniques will be tested in the following section.

2 Model Fitting

We split the data into train and test sets. Test set represents 10% of the data and represents approximately three weeks. To compare the performance of the different models we used the Root Mean Squared Error (RMSE) and the Mean Average Error (MAE) which are the most widely used measures when comparing forecast techniques.

At the end of each model subsection you will find a prediction plot against the target prediction and a table with both aforementioned metrics for the training and test sets.

2.1 Stationarity check

A stationary time series is a time series that does not get "old", i.e. let X_t be the energy consumption at time t, then for any $s > 0$, $X_{t_1}, X_{t_2}, ..., X_{t_n}$ has the same distribution as $X_{t_1+s}, X_{t_2+s}, ..., X_{t_n+s}$. In other words, no matter where you look with a fixed window size, you should observe pretty much the same behaviour. The trend and seasonality components can affect the stationarity of the time series. Stationarity is an important property that needs to be satisfied in order to accurately describe the data at all time points of interest. Whether we need to apply an ordinary or a seasonal differencing can be subjective. Several criterias based on hypothesis tests are used to assess if the time series is stationary. This will greatly help us determine whether further differencing is required to achieve stationarity.

These are based on unit root test. Intuitively, these tests attempt to quantify how much the time series is characterized by its trend. Therefore the Augmented Dickey-Fuller (ADF) test posseses 2 hypotheses. The null hypothesis states that the time series is not stationary. This simply corresponds to the fact that the time series possesses a unit root. To the contrary, if we reject the null hypothesis, the alternate hypothesis suggests a unit root does not exist. Hence, we can probably conclude that the system is stationary. We can summarize this explanation with the following condition: for a p-value > 0.05, we fail to reject the null hypothesis and for a p-value ≤ 0.05, we can reject the null hypothesis. The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test is a very simalar hypothesis test to ADF and it is done once again to reinforce our assumption of stationarity.

From both tests of Table 1, we can reject the null hypothesis, and conclude that our assumption stating the time series is stationary holds. Consequently, no further differencing is required.

Unit Root Test	*p-value*
Augmented Dickey-Fuller	5.178e-15
Kwiatkowski-Phillips-Schmidt-Shin	0.0131

Table 1: Hypothesis test for null hypothesis that the time series is not stationary.

2.2 Exponential Smoothing

The exponential smoothing techniques use weighted averages of past observations where the weights decay exponentially as the observations get older. We tested a more sophisticated technique that exploits three smoothing equations: the Holt-Winter's method.

Each smooting equation can be fined tuned using its respective parameters: α, β, γ. These equations are briefly shown below:

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}), \tag{1}$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}, \tag{2}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)(s_{t-m}), \tag{3}$$

$$y_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}, \tag{4}$$

The smoothing equations stand for the level l_t (unique smoothing equation for the simple exponential smoothing method), the trend b_t and the seasonal component s_t.

2.3 Holt-Winter's model

We tested three variations of Holt-Winter's model. The first one is the default model: no trend or seasonality are specified. In fact, this corresponds to a simple exponential smoothing model. Results of the predictions are shown in Figure 4. Afterwards, the two other models we tried have an additive trend component and a seasonality period set to 168. We also allowed the trend to be damped, i.e., meaning that it will not only be increasing or decreasing. The only difference between the two is their seasonal component, we tested the additive versus the multiplicative one. Results of the predictions are shown in Figure 5 and 6 respectively.

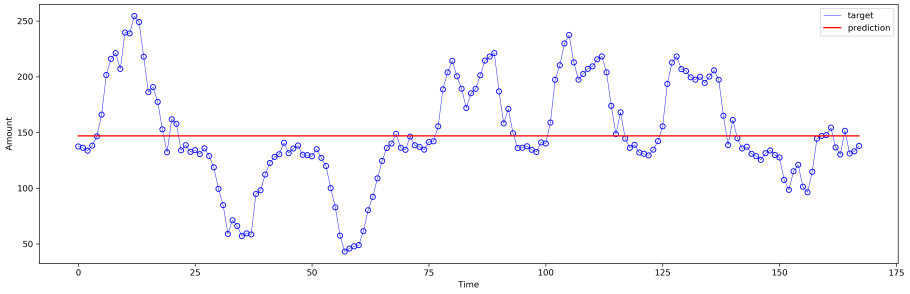

Figure 4: Predictions for the simple exponential smoothing model.

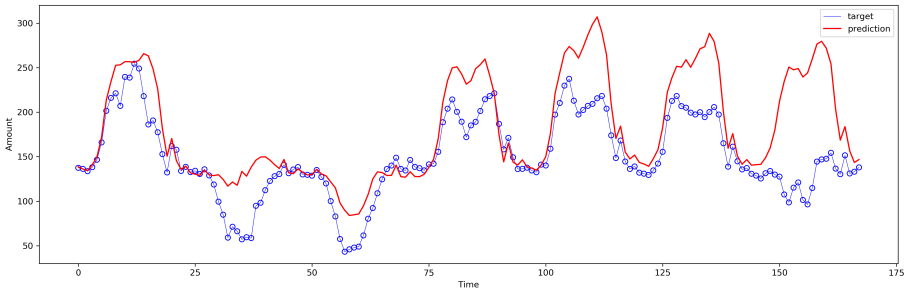

Figure 5: Predictions for the additive seasonal component model.

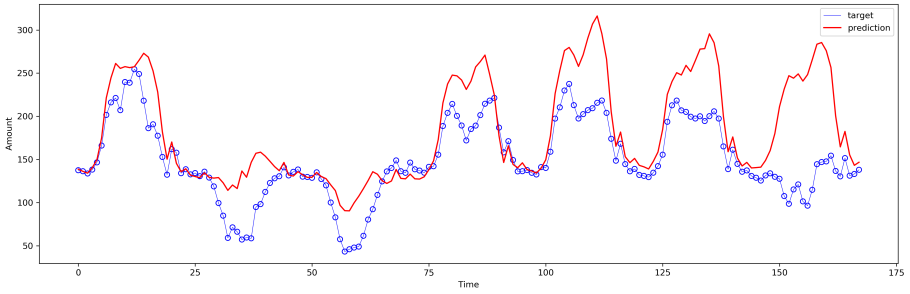

Figure 6: Predictions for the multiplicative seasonal component model.

While the first model has a poor fit due to its incapacity to infer the trend, we observe that there is no significant difference between the additive and multiplicative models.

Model	*RMSE*		*MAE*	
	Training	Test	Training	Test
Simple Exponential Smoothing	18.078	51.357	13.031	38.217
HW additive	9.512	40.144	6.349	26.800
HW multiplicativ	9.650	42.439	6.523	28.766

Table 2: Summary of RMSE and MAE for training and test sets for the HW models.

2.4 ARMA model

The ARMA models represent the other main class of models that are used for time series forecasting. However, they differ from the exponential smoothing techniques as they focus on explaining the autocorrelation in the data rather than seperate and describe the components that constitute it like the trend and the seasonality.

ARMA models are the combination of an autoregressive (AR) and a moving average (MA) model. It is generally possible to deduce the order for each of those models by using the autocorrelation function (ACF) and the partial autocorrelation function (PACF) applied to the data as shown in Figure 7. We would generally take the position of the last two significant lags of the ACF and PACF plots. In this way, we would have a good guess for the order of the AR and MA model. However, for the purpose of learning and also since it becomes difficult to fit a high order AR or MA model in our machines, we will first test an ARMA(2,9) model and analyse the residuals.

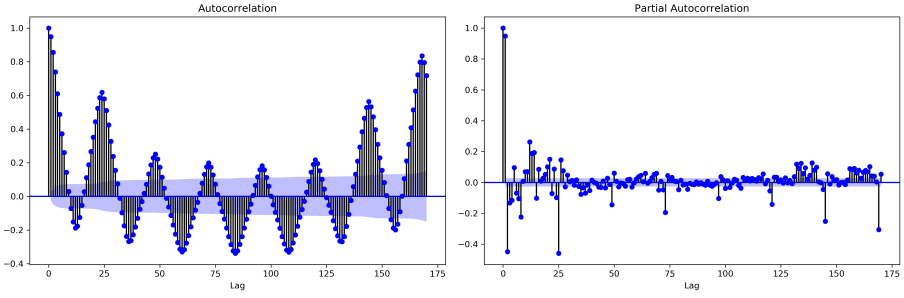

Figure 7: ACF and PACF plots for the entire time series.

The QQ-plot and the ACF and PCAF plots for the ARMA(2,9) model can be found in the Figure 8 and 9 respectively. We clearly managed to explain some of the correlations compared to Figure 7 but we are still far from the residuals being iid and normally distributed, which is what an ARMA model assumes for the error term. The ACF plot of the ARMA(2,9) shows that there is still a lot of correlation every 24 lag. Hence, it would be appropriate to select a $p > 24$

for the AR model.

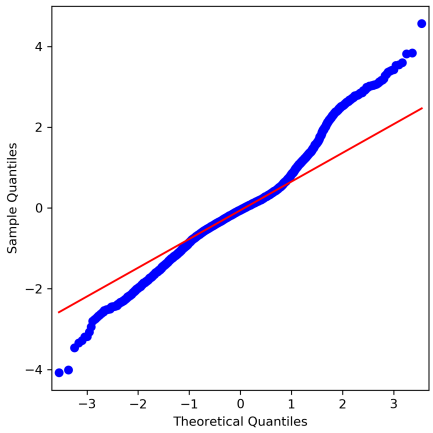

Figure 8: QQ-plot for the residuals of the ARMA(2,9) model.

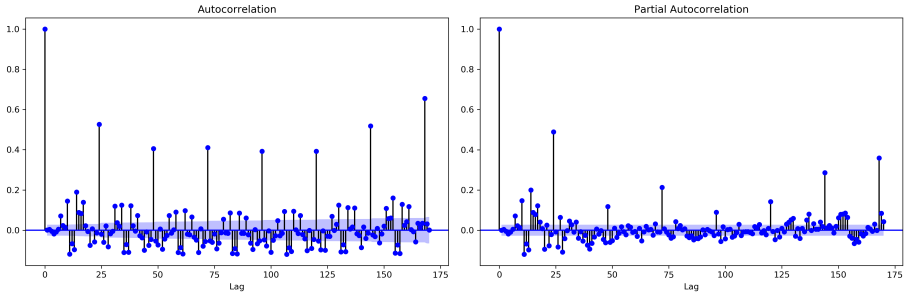

Figure 9: ACF and PACF for the ARMA(2,9) model.

Figure 10 and 11 show the the QQ-plot for the residuals and the ACF and PACF plots of an ARMA(30,0) model. Notice that this is equivalent to an AR(30) model. We should expect a less pronounced value every 24 lag, as now the model is able to explain those correlations. Observe that we get the expected behaviour.

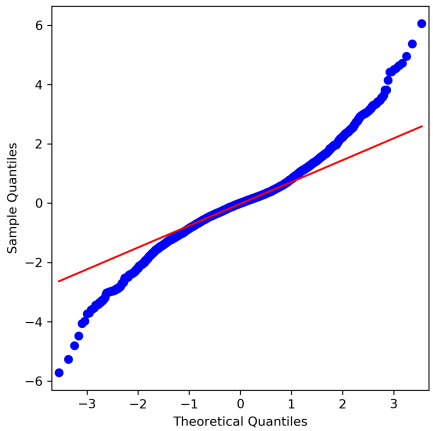

Figure 10: QQ-plot for the residuals of the ARMA(30,0) model.

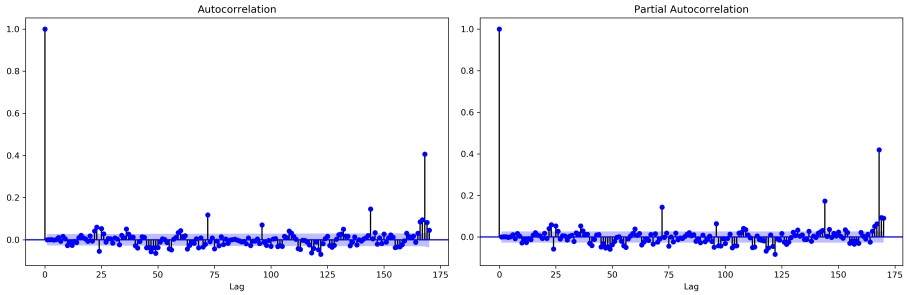

Figure 11: ACF and PACF plots for the ARMA(30,0) model.

We can see that with the order for the AR model set to $p = 30$ we have better results concerning autocorrelation. However, there is still room for improvement. Indeed, we still see a pike at lag 168, which corresponds to a week in frequency. Hence, the following step would be to set the order at $p > 168$ to check if we would get the expected behaviour for the residuals. Unfortunately we were not able to run such job on our machines. As a compensation we propose to visualize the predictions for the ARMA(2,9) and ARMA(30,0) in Figure 12 and 13 respectively.

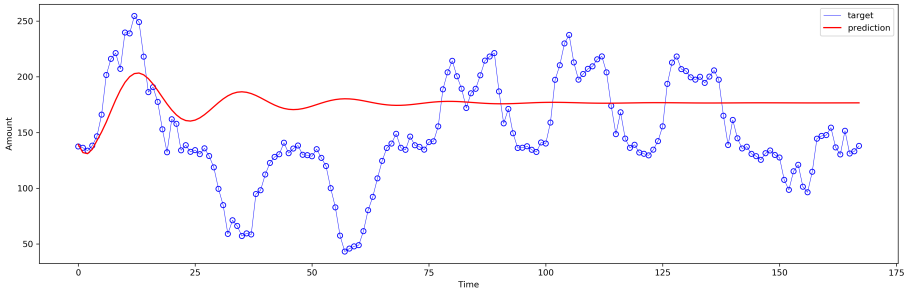

Figure 12: Predictions for the ARMA(2,9) model.

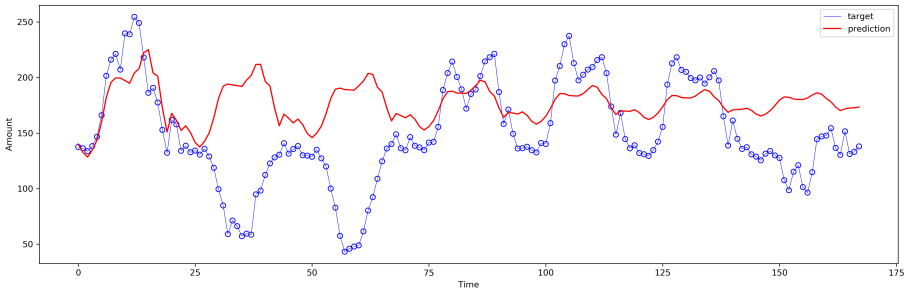

Figure 13: Predictions for the ARMA(30,0) model.

Model	*RMSE*		*MAE*	
	Training	Test	Training	Test
ARMA(2,9)	14.849	55.310	10.761	45.548
ARMA(30,0)	11.846	55.223	8.482	43.761

Table 3: Summary of RMSE and MAE for training and test sets for the ARMA models.

2.5 RNN model

Recurrent Neural Network (RNN) model is a more recent model used for time series forecasting. It became quite popular with all the hype around deep learning in the last years. We tested a Long Short-Term Memory (LSTM) model that is more advanced than a vanilla RNN.

The LSTM is more powerful than the basic RNN due to the RNN suffering from the vanishing gradient problem. This is a phenomena which occurs during back-propagation. When the error is derived with respect to parameters of the weight matrix, these quantities tend to get smaller as we traverse backwards in

the network. This results in the event in which the model is unable to learn long term dependencies as we go closer to the input. The inability to learn long term dependencies defeats the purpose of a recurrent neural network. Fortunately, an LSTM solves this issue so we opt to experiment with it.

A common LSTM unit is composed of a cell, an input and output gate as well as a forget gate. These components allow to handle data sequentially and learn complex patterns thanks to their hidden states throughout the entire process. The architecture of the model we tested is only composed of two layers of LSTM cells where each has 128 hidden layers and a dropout activation. The goal was not to perform a huge gridsearch on all the parameters but rather get an idea of how a simple LSTM model trained a reasonable amount of time can perform with respect to more traditional techniques like the exponential smoothing or the ARIMA techniques. All the details about the parameters of the model like the learning rate, the optimizer (Adam), the criterion (MSE) or the length of the input (168) that were used can easily be found on the folder of interest. Figure 14 shows the predictions obtained in log scale.

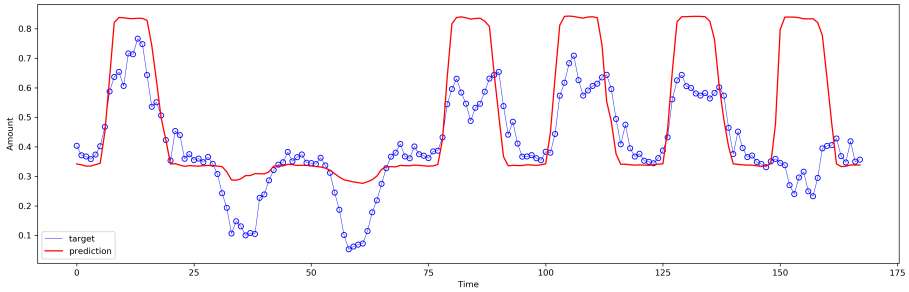

Figure 14: Predictions for the LSTM model in log scale.

Model	*RMSE*		*MAE*	
	Training	Test	Training	Test
LSTM	*0.058*	58.825	*0.047*	43.264

Table 4: Summary of RMSE and MAE for training and test sets for the LSTM model. Training results are presented in log scale.

2.6 LR model

Finally, a last attempt was done using a Linear Regression (LR) model where we used the hour, the day of the week and if we are in holiday or not as the input features. Interestingly, this model is not as bad as it might seem. Although it is simple, it can easily be modified to achieve better perfomance. As an example, we could have considered the last n differenced lag values or even

linked information from another database such as the temperature. Figure 15 shows the predictions.

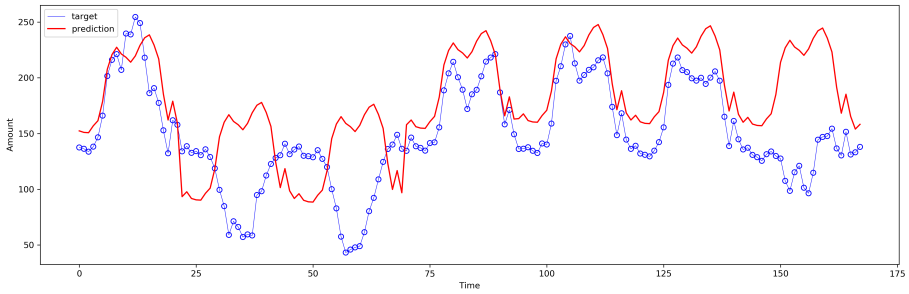

Figure 15: Predictions for the LR model.

Model	*RMSE*		*MAE*	
	Training	Test	Training	Test
LR	34.787	47.159	27.996	37.053

Table 5: Summary of RMSE and MAE for training and test sets for the LR model.

3 Summary

Table 6 gives a summary of the metrics for all the models we tested. Again, we would like to emphasize the fact that we have favored an exploratory approach for this homework. Indeed, each model in its own can be further developed in order to obtain better results. The Holt-Winter's additive model appears to be the preferred choice among all. Hence, we will use this model for computing the prediction intervals of the predefined isolated hours.

Model	*RMSE*		*MAE*	
	Training	Test	Training	Test
Simple Exponential Smoothing	18.078	51.357	13.031	38.217
HW additive	9.512	40.144	6.349	26.800
HW multiplicativ	9.650	42.439	6.523	28.766
ARMA(2,9)	14.849	55.310	10.761	45.548
ARMA(30,0)	11.846	55.223	8.482	43.761
LSTM	*0.058*	58.825	*0.047*	43.264
LR	34.787	47.159	27.996	37.053

Table 6: Summary of RMSE and MAE for training and test sets for all models. Results in *italic* are in log scale.

Table 7 shows the prediction intervals. We used the seasonal naive forecast technique when defining the prediction intervals. It can be defined as follows:

$$y_h \pm 1.96\sigma_h \tag{5}$$

where σ_h is defined as

$$\sigma_h = \sigma\sqrt{(h-1)//m} \tag{6}$$

and where σ is the standard deviation of the residuals, h is the h-step forecast and m is a seasonal period. We fixed it at $m = 24$ as the predictions are very accurate every 24 samples. Notice also that this technique assumes that the residuals follow a Gaussian distribution. This is not correct but as we did not manage to make the bootstrap work efficiently, we chose a technique which finds similar results.

Isolated hours	*Prediction Interval*	
	Lower bound	Upper bound
Friday June 5th 2015 11h-12h	211.538	294.916
Friday June 5th 2015 21h-22h	215.136	298.514
Saturday June 6th 2015 15h-16h	92.635	183.971

Table 7: Prediction intervals for the isolated hours from the HW additive model.

4 Contest

We would like to participate to the contest using the same model. Instructions are very straightforward and can be found at the very end of the attached jupyter notebook.