Electricity Price Forecasting

Raffaele Anselmo[1]

Abstract
The aim of this project is the definition, development and validation of different forecasting models in order to predict the daily price of Electricity in the Italian Market. The goal is the daily prediction over a 11 months horizon using 9-years price data with ARIMA, UCM and Machine Leaning models.

Keywords
Time Series — Forecasting — ARIMA — UCM — Machine Learning — Data Science

[1] *Università degli studi di Milano-Bicocca, CdLM DataScience*

Contents

Introduction

"The Information Society is better seen as a neo-manufacturing society in which raw materials and energy have been superseded by data and information, the new digital gold and the real source of added value" [1]

The value of Data has been exploited in various fields, from NLP to industrial processes optimization, but the most trivial example of their power can be founded in Time Series Forecasting.

As per definition, Forecasting is a technique for predicting the future as accurately as possible, given all the information available, including historical data and knowledge of any other events that might impact the forecast. [2]

Different Forecasting techniques have been developed from the early '900 and the recent growth of data generation, along with the increase of computational capabilities, has increased the attention in this topic.

The aim of this project is to explore the capabilities of 3 forecasting methods families, ARIMA models, UCM and Machine Learning derived models, to predict, given 10 years of historical daily data, the price of electricity in the Italian Market over a 11 months horizon. The data are aggregated in the dataset with an unknown aggregation criterion.

The paper is divided as follows:

- In the first section ARIMA, UCM and 2 Machine Learning models, KNN and Recurrent Neural Networks, are introduced;
- In the second section the prediction task is presented.
- The third section is focused on ARIMA models implementation and evaluation;
- The fourth section treats about the development of Unobserved Component Models;
- The Machine Learning models are applied and evaluated in the fifth section;
- In the sixth section the models obtained were compared with respect to training and validation performances. The best model for each family will be deployed for the prediction on (unknown) Test Set.

Lastly, the results achieved in the analysis are commented with focus on the most important aspects emerged from the survey.

1. Models Overview

1.1 ARIMA

ARMA (Autoregressive Moving Average) models [3] are the most general class of linear models for forecasting a time series through the Box & Jenkins approach (1970). The main idea of this approach is that the model is adjusted towards the reality, in opposition to the classic approach where the reality was supposed to go towards the model.

The Process is seen as the interaction of two components:

- **Autoregressive Component** $\phi_p(B)$ treats the long-period dependencies, it is always invertible and it is stationary if the characteristic equation roots lie outside the unitary circle
- **Moving Average Component** $\theta_q(B)$ treats the short-term dependencies, it is always stationary and it is invertible if the characteristic equation roots lie outside the unitary circle

If d roots of ARMA characteristic equation lies on the unitary circle, then the process is called ARIMA of order q, where I stands for Integrated.

If there is any seasonal dependency in the process, it would be modelled through the seasonal differences. The process is called SARIMA(P,D,Q) and the polynomes $\Phi_p^s(B)$ and $\Theta_q^s(B)$.

In most real cases, a time series presents either a seasonal period memory, either a non seasonal memory. The model that has both characteristics has the form:

$$ARIMA(p,d,q)(P,D,Q)_S$$

$$\phi_p(B)\Phi_p(B)\Delta^d\Delta_S^D y_t = k + \theta_q(B)\Theta_Q(B)\varepsilon_t$$

The identification of the model with Box & Jenkins procedure is based on the global autocorrelation function:

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}$$

where γ is the autocovariance function.

And on the partial autocorrelation function:

$$\alpha(h) = \frac{Q_h}{P_h)}$$

where P is the Toeplitz Matrix of order h and Q is the same matrix, where the last column is substituted by the vector $(\rho_1, \rho_2, \ldots, \rho_h)$

The first function holds all the information of the memory of the stationary process, while the second one is defined as the correlation of the stationary process at lag h, excluding all the intermediary linear correlations.

Before proceedings with the identification of the AR order (with PACF) and MA order (with ACF), it would be necessary to make the time series stationary. If the non-stationarity is at mean-level, the (seasonal) differences are the best choise, while if the non-stationarity is at variance-level, the Box & Cox transformations are needed.

The last steps of the identification process are the model evaluation and its application.

1.2 UCM

The Unobserved Components Models [4] (known also as Structural Time Series Models) are models where the time series is seen as sum of some unobserved components, typically trend , cycle, seasonality and white noise:

$$Y_t = \mu_t + \psi_t + \gamma_t + \varepsilon_t$$

Although it seems like the classical approach, the UCM components are defined as stochastic processes, allowing statistical testing, the use of regressors and the derivation of the whole distribution of future outcomes.

- The **Trend** component is responsible for mean's variations on the long run.

  $$\mu_t = \mu_{t-1} + \beta_{t-1} + \eta_t$$

  $$\beta_t = \beta_{t-1} + \zeta_t$$

  where η_t and ζ_t are WN with 0 mean and variances σ_η^2 and σ_η^2, respectively.

  The stochastic or deterministic nature of its two components, Level and Slope, determines the particular type of trend (local linear trend, Random Walk, Random Walk with drift, Integrated Random Walk).

- The **Cycle** component can be used for modeling periodic oscillations different from seasonality (typical of economic context):

  $$\begin{bmatrix}\psi_t\\ \psi_t^*\end{bmatrix} = \rho\begin{bmatrix}cos\lambda & sin\lambda\\ -sin\lambda & cos\lambda\end{bmatrix}\begin{bmatrix}\psi_{t-1}\\ \psi_{t-1}^*\end{bmatrix} + \begin{bmatrix}k_t\\ k_t^*\end{bmatrix}$$

  It is stationary, thanks to the dumping factor ρ, that dumpens the effect of the shocks.

  The cycle component is the first variable ψ_t, the second one, ψ_t^* is used only for the construction of the component.

- The **Seasonality** component can be modelled using two alternative forms: the stochastic dummy form:

  $$\gamma_t = -\sum_{i=1}^{s-1}\gamma_{t-i} + \omega_t$$

  and the stochastic trigonometric form:

  $$\begin{bmatrix}\gamma_{j,t}\\ \gamma_{j,t}^*\end{bmatrix} = \begin{bmatrix}cos(2j\pi/s) & sin(2j\pi/s)\\ -sin(2j\pi/s) & cos(2j\pi/s)\end{bmatrix}\begin{bmatrix}\gamma_{j,t-1}\\ \gamma_{j,t-1}^*\end{bmatrix} + \begin{bmatrix}\omega_{j,t}\\ \omega_{j,t}^*\end{bmatrix}$$

  From the application point of view, the trigonometric stochastic form tends to evolve more gradually than the dummy stochastic form because it is represented by a MA process, that is smoother than the white noise that represents the dummy form.

The **Kalman Filter** is the algorithm used to make inference on unobserved components

1.3 Machine Learning models

Two Machine Learning models have been developed, the KNN regression and a LSTM (Long Short-Term Recurrent) Neural Network.

Compared with classical statistical models, computational intelligence methods exhibit interesting features, such as non-linearity or the lack of an underlying model, that is, they are non-parametric [5].

- **KNN** is a simple algorithm that stores a collection of examples (consisting of a vector the historical values of the time series and his associated target values) and, given a new example, KNN finds its k most similar examples (called nearest neighbors) according to the Euclidean distance. Then, the prediction is performed as an aggregation of the target values associated with its nearest neighbors.

  Two strategies have been utilized to perform the prediction: The MIMO (multiple Input Multiple Output) strategy, that is characterized by the use of a vector of target values, and the recursive strategy, an iterative one-step ahead strategy, similar forecasting method of ARIMA and Kalman filter.

- The **LSTM**[6] is a Recurrent Neural Network (so with shared parameters) that reduces the effect of vanishing gradients during the backpropagation with connection weights that may change at every time step. It is composed of cells which include several operations. An internal state variable is passed and modified into the cells through the Operation Gates (Forget Gate, Input Gate, Output Gate).

2. Prediction task

The time series (1) consists of daily data from 01/01/2010 to 12/31/2018 (3287 observation) without NULL values.

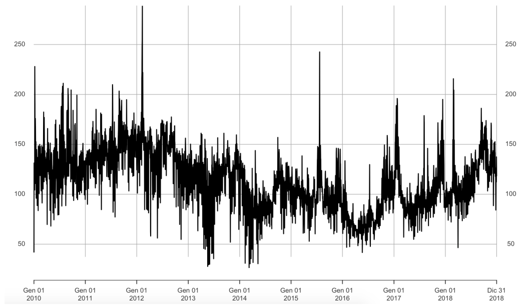

Figure 1. Electricity Price time series

The values range in a wide interval, so a logarithmic transformation is applied to the time series.

The predictions are requested for the interval 01/01/2019 - 11/31/2019 (1 to 334 step ahead). Test set data are unknown, so a partition of the training set has been used for the validation. In order to validate the algorithms on the same horizon of the test set, the validation set has been identified as the last 334 observation of the training set (almost 90% - 10% split), consisting on the interval 02/01/2018 - 12/31/2018.

The evaluations and comparison of the different forecasts are in terms of Mean Absolute Error (MAE), that gives a direct economical interpretation of the forecast loss (when applied to the original scale), also for non-statistician experts (usually the requester of the forecast). It is also one of the few metrics available for the LSTM RNN.

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

3. ARIMA forecasting

The first operations to identify the model are conducted in order to make the time series stationary. The non stationarity in variance has been corrected with the logarithmic transformation, while for the non stationarity in mean, a seasonal difference is applied. The resulting time series is smoother, but not as much as desired. However, the Augmented Dicky Fuller test confirms the stationarity.

The ACF (2) never decays and the PACF (3) has an alternate course and decays after almost 50 lags but reappears at seasonal lags. This suggests the presence of both AR and MA components, maybe a ARIMA(2,1,1)(1,1,1). Unfortunately, the R package TSA doesn't deal with lags greater than 365, so a seasonal model for daily data can't be developed.

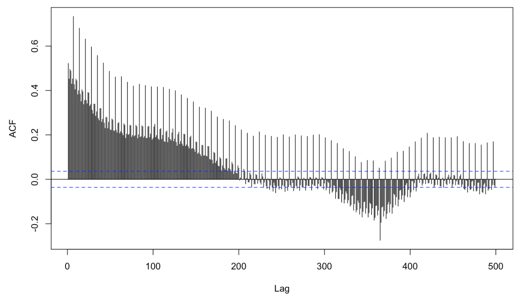

Figure 2. ACF

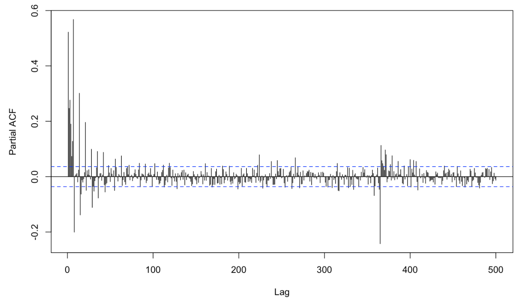

Figure 3. PACF

The *auto.arima* function has been used to identify a model that could be fitted. It select the best models with respect to the AIC score and it gave back a ARIMA(5,1,3) model.

The ARIMA forecast on the validation set (4) has good performances for the firsts delays, but it becomes a straight line after few steps ahead, losing all information about seasonality, cycle, etc. Its Validation MAE is equal to 0.236.

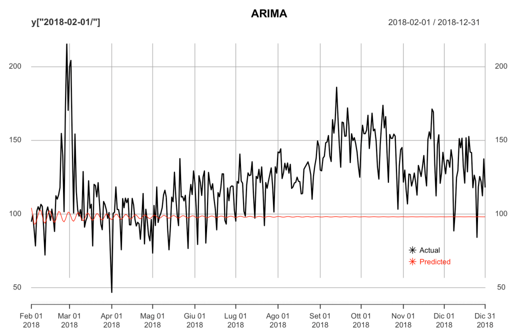

Figure 4. ARIMA Forecast

4. UCM forecasting

The Unobserved Components Models have been developed with KFAS package [7].

External regressors, representing different holidays like Christmas or Easter, have been included in the models.

The identification of the best UCM has been conducted iteratively, from the simplest model with Local Linear Trend and daily stochastic dummy seasonality of order 7, to more complex models with economic cycle and intra-annual daily stochastic trigonometric seasonality.

The following components have been evaluated during the iterations:

- LLT vs IRW
- Stochastic Seasonality vs Deterministic Seasonality
- Cycle vs Seasonality vs Cycle and Seasonality

Initial diffuse conditions have been avoided for computational reasons. The mean of the time series has been used as initial condition for the level, while the variance of the components has been initialized with the variance of the process.

The estimation has been made with the Kalman Filter with state smoothing and signal filter (that includes the one-step ahead predictions)

The results of the models on validation set are summarized in Table 1:

Table 1. UCM comparison

Model	Components	Val-MAE
1	LLT + DUMMY SEAS (7)	0.255
2	IRW + CYCLE + DUMMY SEAS (7)	0.273
3	IRW + CYCLE + DET TRIG SEAS (365)	0.349
4	LLT + DUMMY SEAS (7) + TRIG SEAS (365)	0.309
5	IRW + DUMMY SEAS (7) + TRIG SEAS (365)	1.906
6	IRW + CYCLE + DUMMY SEAS (7) + DET TRIG SEAS (365)	0.322
7	IRW + CYCLE + DUMMY SEAS (7) + TRIG SEAS (365)	0.323
8	LLT + CYCLE + DUMMY SEAS (7) + TRIG SEAS (365)	0.158

The 8^{th} model, with Local linear Trend, Cycle, Stochastic Seasonality (dummy) of order 7 and Stochastic Seasonality (trigonometric) of order 365 (where only the firsts 16 harmonics have been included) has the best performances on the validation set (5), with a MAE of only 0.158.

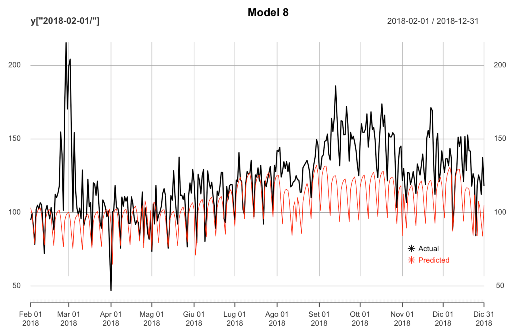

Figure 5. Model 8 Forecast

This model will be deployed for the test set prediction (after a new training where the training set includes also the validation set).

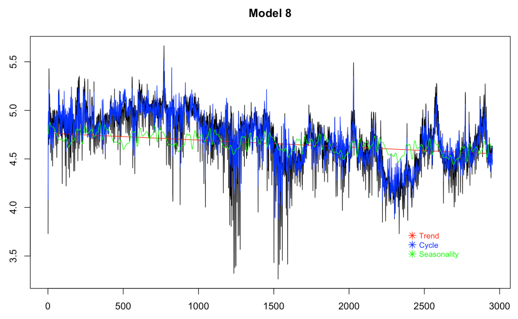

Figure 6. Model 8 Components

5. ML forecasting

5.1 KNN regression

The KNN models have been developed with tsfknn package [5].

The first model uses the MIMO strategy and it is extremely fast to train and make the predictions. It uses 1:1500 lags and 3 nearest neighbors to perform the prediction.

The second model uses the recursive strategy, this means that only one-step ahead forecasts are performed iteratively to forecast all the 334 future observations (the model uses previous predictions instead of historical values when these are unavailable). The parameters lags and number of nearest neighbors are the same as the previous model.

The results are pretty similar between the two strategies (7) and they are summarized in table 2.

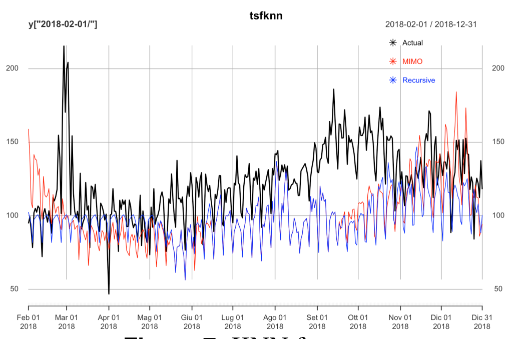

Figure 7. KNN forecasts

Table 2. KNN comparison

Model	Val-MAE
MIMO	0.243
Recursive	0.229

5.2 LSTM

The Long Short Term Memory Recurrent Neural Network has been developed with the help of Keras package [8].

A generator function has been used to transform the data in a suitable form for the LSTM. In particular, every batch contains 128 records with a *lookback* (the past values taken in account) equals to 1500 and 334 future target values. The validation data has been defined with the same generator function with some adjustments in order to obtain the same validation window as the previous models.

The model architecture consists of one lstm layer with 32 units, tanh activation function 0.1 dropout rate and 0.3 recurrent dropout rate and one dense output layer with 334 neurons (equals to target horizon).

Once compiled with *rmsprop* optimizer and *mae loss* function, the model is trained over 10 epoch with 50 steps per epoch.

The validation MAE (≈ 0.26) is similar to the training MAE (≈ 0.20) (9).

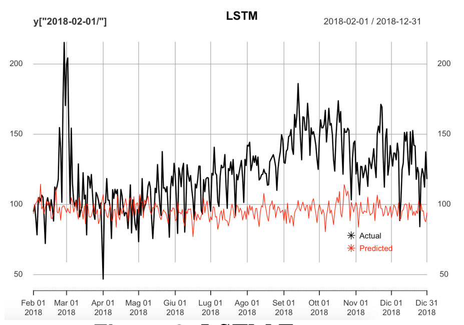

Figure 8. LSTM Forecast

6. Results comparison

Training and Validation errors are summarized in Table 3:

Table 3. RMSE comparison

Model	Train-MAE	Val-MAE
ARIMA	0.110	0.236
UCM 1	0.073	0.255
UCM 2	0.062	0.273
UCM 3	0.106	0.349
UCM 4	0.064	0.309
UCM 5	0.080	1.906
UCM 6	0.063	0.322
UCM 7	0.063	0.323
UCM 8	1.9e-15	0.158
KNN M	NA	0.243
KNN R	NA	0.229
LSTM	0.20	0.26

The best Unobserved Component Model has the best performances in terms of MAE both on training and validation set. The performances of Arima forecast are good but, as we saw in the analysis, it has not any information about the cyclic oscillation that are found in the original time series.

The training errors for UCM are seen as the difference between the smoothed values and the actual ones.

The comparison of training performances for the KNN models are not available, due to the fact that this algorithm perform the prevision directly in the future as the aggregation of past similar values.

The comparison training-validation for LSTM over the 10 epochs is visible in Figure 9.

For the test set predictions, the ARIMA model, UCM 8 and KNN recursive will be deployed.

7. Conclusions

The aim of this paper was to explore the peculiarities of each of the three forecasting methods developed. Starting from the ARIMA forecasting, it has not proved entirely satisfactory, although its error measures are not so bad. The difficulties to deal with daily data in R contributed to the lack of specialization on the models. ARIMA models seemed to be more

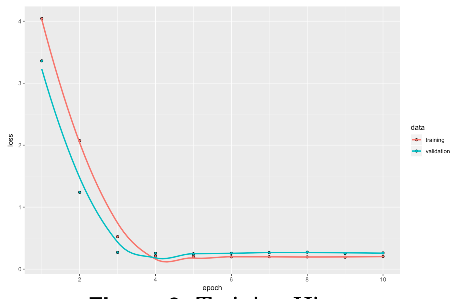

Figure 9. Training History

suitable for an high level time aggregation (e.g. week or month) and more regular time series.

The UCM models have revealed very tricky in the construction. The freedom of KFAS package allow to develop ad-hoc models with particular update functions and parameters to be estimated. Also, the easily addiction of external features is a great feature of UCM.

The two Machine Learning models revealed different properties. The KNN regressions are extremely fast to train and predict, and they need almost 4 or 5 rows of code to be implemented, that makes these models extremely easy, without a great loss of performances. The use of other distance metrics, that weights more the latest observation than the remote ones, can reveal interesting results.

The LSTM RNNs are deeper models, that can be fine tuned with respect of many hyper-parameters, with the drawback of an extremely long time of training (almost 1 hour in this case). The feeling is that with more computational power, more data and the right hyper-parameters tuning, the RNN can outperforms the other forecasting methods.

Finally, the analysis revealed that ARIMA, UCM and machine learning models are able (with some differences on the performances) to predict the future values of the Italian Market electricity price.

References

[1] L. Floridi. *The Fourth Revolution: How the Infosphere is Reshaping Human Reality*. OUP Oxford, 2014.

[2] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[3] E.B. Dagum. *Analisi delle serie storiche: modellistica, previsione e scomposizione*. Collana di Statistica e Probabilità Applicata. Springer, 2001.

[4] Matteo Pelagatti. *Time Series Modelling with Unobserved Components*. 07 2015.

[5] https://cran.rapporter.net/web/packages/tsfknn/vignettes/tsfknn.html.

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

[7] Jouni Helske. KFAS: Exponential family state space models in R. *Journal of Statistical Software*, 78(10):1–39, 2017.

[8] https://cran.r-project.org/web/packages/keras/index.html.