Abstract

Forecasting energy demand is an important task in electric power industry. This study aims to forecast in advance at each day, the 24 hours of the electrical energy consumption of the next day in Spain, so we are in the context of multi-step time series forecasting. To do so, explanatory variables such as the electricity price and the weather were used. To take into account the problems related to dimensionality and multicollinearity, we performed Ridge regression, LASSO and regression based on Factorial Analysis of Mixed Data (FAMD). In addition, machine learning and deep learning models were used to reach our goal. Apart from the deep learning models, the remaining models were evaluated using Walk Forward Validation (WFV) instead of Cross Validation. Among our model, we fund that Convolutional Neural Network and the Random forest with multiple output give better results so future work could combine both, but their RMSE (Root Mean square Error) were very far from the one of European Network of Transmission System Operators for Electricity (ENTSOE) that we took as reference. Limits of this work have been discussed and some solutions have been proposed.

Key words : Energy demand, Multi-step time series forecasting, Walk Forward Validation.

Contents

Introduction

Energy is a leading economic sector. Whether it is for industrial activities, particularly to operate machinery, transport, access to communication technologies, or the provision of essential services such as education and health, the continuous availability of energy is necessary for the smooth running of these sectors. Thus, energy contributes to changes in society and to the improvement of the living conditions of the population. And this is why energy occupies a de facto dominant place in the Sustainable Development Goals (SDOs) i.e., ensure access for all to reliable, sustainable, and modern energy services at an affordable cost. However, the energy services offered obviously depend on the energy demand. The challenge then become the forecasting of the energy demand on the markets in order to ensure a safe, reliable and affordable energy supply.

The forecasting of demand plays an essential role in the electric power industry and this forecasting can be made in the short term (minutes, hours, day) to medium and long term (weeks, months, years), in a microscopic level (individual consumer) to macroscopic level (country, region). For example, been able to better forecast the 24 hours (short term) ahead energy demand can allow to reduce the need for standby reserve and plants which often pollute, and to proactively manage increasing amounts of variable sources (Donti, 2019). In turn that will contribute to an efficient network. Better long-term forecasts can help system operators and investors determine where and when variable plants should be built. Even if load forecasting has been a research topic for decades, there is again nowadays a rising research trends in electricity load forecasting due to the eminent changes in the grid, the integration of new technologies such as the intermittent renewable generation, distributed generation, smart meters, and electric vehicles, etc.

According to Hu (2020), there is a rich state of the art of methods to forecast energy demand which can be divided into statistical models (purely empirical models where inputs and outputs are correlated using statistical inference methods, ARIMA model for example); grey models (they combine a partial theoretical structure with empirical data to complete the structure); artificial intelligence models (Deep/Machine Learning). The latter are known to have highly improved the accuracy, robustness, precision, and the generalization ability of the conventional time series forecasting tools.

Furthermore, the short-term prediction of electricity demand needs to take into account multiple historical climate and load time series for example, as well as market conditions such as generation capacity stocks (European Network of Transmission System Operators for Electricity: ENTSOE, 2019). The number of variables can therefore increase considerably, with potential collinearity, which might adversely affect model accuracy. A special attention should therefore be paid to the selection of variables.

Our goal in the current study is to forecast in advance at each day, the 24 hours of the electrical energy consumption of the next day in Spain and try to do better than the ENTSOE if possible. The dataset we use is from Kaggle website and contains 4 years (2015-2018) of electrical consumption, generation, pricing, and weather data for Spain. To do this, we will opt on the one hand for parametric econometric methods and on the other hand for machine learning techniques (Random Forest, KNN, XGBoost SVM, and neural networks). Our predictions did not reach the accuracy level of the ENTSOE and the multioutput Random forest and the Convolutional Neural Network were our "best" models. We highlight different limits of our work and give different possibilities to improve the result, for example by proposing to more focus on the nonlinear relationship between the electricity consumption and the temperatures, because weather fundamentally drives both variable generation and electricity demand, apply different other transformations to the data rather than the standardization that has been used, and also test a large set of hyperparameters for the model and this goes through the optimization of the code we used since a some parts were from scratch and under-optimized.

The rest of this report is structured in 3 chapters: the first aims to present a literature review on the study problem, in the second, we will present the materials and methods that will be used and the last focus on the results. Before concluding, we will compare our different methods .

1 Literature review

Many authors have tried to model and to predict energy demand with almost different approaches. In our work, we will present four of these documents.

First, to model the demand for electric energy Filik et al., (2009) proposed a mathematical model in three forms: short-term prediction (demand from one hour to one week), medium-term prediction (demand from one week to one year) and long-term prediction (for demands of more than one year). They believe that their work was unique because it attempts to make the long-term forecasting with an hourly accuracy. And to build the final model, they combined the total yearly variations, marginal weekly variations, and hourly residual variations. They results showed on the test data for the years (2002 to 2005) the MAPE going from 3.03 to 7.30 percent and the RMSE going from 613.8 to 1763.5.

Then, Kolotroni et al., (2010) described a method for the prediction of heating and cooling energy demand for buildings within the Urban Heat Island. They used a back-propagation ANN model which included hourly air temperature, relative humidity, cloud cover, wind speed and global solar radiation and it shows high accuracy for monthly average temperature. But one of the main factors affecting heating and cooling loads and thus the effect of urbanization on energy demand was here the distance from physical centre or thermal centre of London.

Garshasbi et al., 2016, developed a double strategy for the prediction of the instantaneous and cumulative net energy balances with a hybrid genetic algorithm and a Monte Carlo simulation approach. This strategy allows them to predict instantaneous and cumulative net energy balances and the energy consumption and generation patterns in a cluster with buildings. On of the mains points was the fact that the model with this approach was dynamic and performed better with large data.

More recently, Spyros et al., (2018) compare the performance of machine learning forecasting methods with traditional statistical ones. These machine learning forecasting methods are: Multilayer perceptron, Bayesian Neural Network, Radial Basis Function, Generalized Regression Neural Networks (also called kernel regression), K-Nearest Neighbor regression, CART regression tree, Support Vector Regression, and Gaussian Processes. Their results showed that the accuracy of the machine learning models appears to be lower than the statistical models. But the complex forecasting methods have improved their accuracy considerably over time.

So, it is obvious that the choice of model affect the outcome. This is what is underlined by the Koloktroni et al. 2009 (cited by Koloktroni et al. 2010) when they said that for the model, it is important to select an appropriate algorithm which fits the specific purpose of the problem to get the best performance for the simulation and prediction. In our study, we considered several methods (Factorial analysis for mixed data, Ridge regression, LASSO, GETS, Random Forest, XGBoost, Support Vector Regression, Convolutionnal neural network, Mutilayer perceptron, Long Short-Term Memory ...) presented below to predict hourly energy demand. To complete the available data (hourly energies values and price, weather variables), our study also included dummies variables: holidays dummy, month dummies and dummies that capture the fluctuation of the total load.

2 Materials and methods

2.1 Data used

Our database is relating to hourly energy demand values from January 2015 0 am to December 2018 11 pm and is composed of two parts: the first one is "energy dataset" with 29 variables and 35064 observations and, contains the hourly energies values and price (actual and forecast) and the second one "weather features" contains the weather variables related to 5 major cities (Madrid, Barcelona, Seville, Bilbao, Valencia) in Spain.
Link: `https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather/download`

We describe the variables in table 2.1 and 2.2.

Table 2.1: Energy dataset

Variables	**Description**
Time	Time (Central Europe Time)
generation biomass, generation fossil brown coal/lignite, generation fossil coal-derived gas, generation fossil gas, generation fossil hard coal, generation fossil oil, generation fossil oil shale, generation fossil peat, generation geothermal, generation hydro pumped storage aggregated, generation hydro pumped storage consumption, generation hydro run-of-river and poundage, generation hydro water reservoir, generation marine, generation nuclear, generation other, generation other renewable, generation solar, generation waste, generation wind offshore, generation wind onshore	Represents the amount of electricty generated by the different energy sources (biomass, fossil,water, wind, solar...) in MegaWatt-hour (MWh)
forecast solar day ahead, forecast wind offshore eday ahead, forecast wind onshore day ahead	Represents the forescast amount of electricty in MWh
total load forecast	Represents the forescast total amount of electricty in MWh
total load actual	Represents the actual total amount of electricty in MWh
price day ahead	The price of energy one day ahead in €/MWh
price actual	The actual price of energy in €/MWh

Table 2.2: Weather features

Variables	**Description**
dt_iso	Time
city_name	Name of the city
temp, temp_min, temp_max, pressure, humidity, wind_speed, wind_deg, rain_1h, rain_3h, snow_3h, clouds_all, weather_id, weather_main, weather_description, weather_icon	The temperatures are in Kelvin (K), the pressure is in hectoPascal (hPa), wind_speed is in mph. 'rain_3h' and 'rain_1h' columns are supposed to provide information respectively about the precipitation (i.e. rain) of the last 3 hours in mm and the last 1 hour, snow_3h measures the snowfall, weather_main, weather_description and weather_id : they give a qualitative description of the weather at the given hour, but weather_description gives more details about the weather and weather_id is numeric. weather_icon is a qualitative variable and its meaning is not all clear.

Note that the descriptive analyses allow us to generate others dummy and lagged variables. Among the variables in the energy dataset, some have no values and other have only 0 as value. We deleted them and we try to interpolate data where there are missing values. Only a small part of our input data will be noisy, and it will not affect performance noticeably. Furthermore, variables like day-ahead forecasts for the total load, the solar energy and the wind energy are not used in our analysis, since they just represent the prediction made in advance. Concerning the weather feature, since we can reject the hypothesis of independence between "weather_id" and "weather_description" (after a chi square test), we keep just one variable between "weather_id", "weather_description" , "weather_main" and one hot encode it. We also deleted weather_icon. Also, the weather information is regarding 5 different cities so we have to make sure that the number of records in energy dataset is equal to the number of records in each city for the weather_features. To do so, we drop the duplicates by considering the city name and the time (dt_iso) as identifiers. After this we merge the two databases (energy and weather) according to the date and the hour.

The data is comprised of 5 major cities: Madrid, Barcelona, Seville,Valencia and Bilbao. Our electricity power demand data is for Spain as a country. So the weather data will need to represent the whole country. To do this we will take a weighted average of the cities features values based on population size in each city. This is a little weird but, since the demand of a given town will probably depend on the size of this town in terms of population and the weather conditions in this town (for example a heat wave hitting a densely populated city is likely to see more total energy demanded than a smaller city).

But if we can take the weighted average for numeric variables, what about the weather_main variable which is a qualitative (12 categories) and available for each town? One solution would be to keep them intact but when we dichotomize them, we will have too many dummy variables coming from this single variable. So, we decided to remove it from our analyses hoping that the other meteorological variables would give a good indication of the weather.

2.2 Description of the methods used

To predict the hourly energy demand, we used several approaches, supervised, unsupervised and selections methods. In this section, we present these methods.

2.2.1 Principal Components Regression

Principal component regression (PCR) is a regression analysis technique that is based on principal component analysis (PCA) and is an alternative method to multiple linear regression when facing multicollinearity. Multicollinearity leads to high variance, which means that we can be far from the true value of the model. Multicollinearity deteriorates OLS estimator quality. Multicollinearity often arises in many real-world applications where the set of explanatory variables has the nearly linear dependence, or the sample size is smaller than the variable size. This can be avoided by using the selected Principal Components in place of the original variables since the Principal Components are uncorrelated. The main purpose of Principal Component Regression is to estimate the values of a response variable at the basis of selected Principal Components of the explanatory variables. Secondly, the dimensionality of the regressors is reduced by taking only a subset of Principal Components for prediction.

However, when there is a mix of quantitative and qualitative variables, it is not very wise to deal directly with PCA data that mix columns of numerical (continuous) values and indicators from the one hot encoding. Since the dispersions are different, there is a strong likelihood that the results will be biased (Rakotomalala, 2012). In this case a factorial analysis of mixed data (FAMD) is more adapted. This is more general and acts as PCA quantitative for variables and as MCA for qualitative variables. FAMD consists in normalizing as in PCA the quantitative variables and then, the 0/1 coding of the dummy variables is transformed into 0/x coding where "x" is calculated from the frequencies of the modalities. Then an unstandardized PCA is run to conduct the analysis. In this study, we will regress on the main components resulting from the FAMD.

2.2.2 Ridge Regression

Ridge regression is a 'shrinkage' or 'penalized smoothing' method in regression analysis. It imposes a penalty on the squared Euclidean norm of the regression coefficients to counterbalance OLS' tendency to overadapt to the idiosyncrasies of the data at hand, a tendency that typically leads to a shrinkage of the R^2 for prediction relative to the R^2 for fitting. Estimate are defined as:

$$\beta_{Ridge} = (X'X + \lambda I_N)^{-1} X'Y \tag{2.1}$$

In ridge regression, variables are standardized. Shrinkage allows to reduce the grid of the values that taken by the estimated parameters. It is useful to avoid the impact of variables with high variances. λ is the penalty parameter that controls the size of the coefficient and the amount of regularization.

Ridge improves OLS once the value of λ is increasing, the flexibility of the Ridge regression fit decreases, leading to a smaller variance but a higher bias where in OLS situation, when λ is equal to 0, there is no bias but when λ is increasing, the shrinkage of the ridge coefficient leads to a reduction in the variance of the predictions.

2.2.3 LASSO

Lasso : Lasso (Least Absolute Shrinkage and Selection Operator) regression method is like the Ridge regression, but with variables selection (Ridge and PCR do not select variables). Lasso coefficients are the solutions to the L1 optimization problem:

$$minimize(y - X\beta)'(y - X\beta) s.t \sum_{j=1}^{p} |\beta_j| \leq t \tag{2.2}$$

This is equivalent to the loss function:

$$(y - X\beta)'(y - X\beta) + \lambda||\beta_j||_1 \tag{2.3}$$

λ, the tuning parameter controls the amount of regularization and $\lambda = 0$ implies no shrinkage.
The penalty force some of the coefficients estimates to be exactly equal to zero when λ is very large, therefore it does variable selection. Note that unlike the ridge regression, the β_{lasso} has no closed form. In this study, regarding the way that we make the prediction and the way that we evaluate the different models, using a large set of λ and using the lasso for each of them in order to see what λ could give a good prediction, might be time consuming to. So, we decide to employ a method that could give us directly the information criterion (IC) on the training, and this lambda which minimize the IC is taken as the best LASSO. A method that can provide us directly in scikit-learn the IC for LASSO is the LassoLarsIC. So that is what we use.

LARS: The least-angle regression was developed by Efron, Hastie, Johnstone and Tibshirani (2004). At the first step it identifies the variable most correlated with the response. Rather than to fit this variable completely, LARS moves the coefficient of this variable continuously toward its OLS value (causing its correlation with the evolving residual to decrease in absolute value). As soon as another variable "catches up" in terms of correlation with the residual, the process is paused. The second variable then joins the active set, and their coefficients are moved together in a way that keeps their correlations tied and decreasing. This process is continued until all the variables are in the model and ends at the full OLS fit. To get the entire Lasso path, the LARS require the following modification :

> If a non zero coefficients hits zero, drop its variable from the active set of variables and recompute the current least square directions

The Lasso modification of the LARS is an effective way of computing the solution to any Lasso problem. The "tuning" parameter (hyper-parameter) for a LARS fit is the number of steps K and to choose K we use Information Criteria or Cross Validation.

2.2.4 GETS

According to Clarke (2014) in "General-to-specific modeling in Stata", General to specific (GETS) modelling is a prescriptive way to select a parsimonious and model from a large set of real-world variables and enables the researcher to avoid unnecessary ambiguity or adhoc decisions. This process involves the definition of a general model that contains all potentially important variables and then, via a series of stepwise statistical tests, the removal of empirically "unimportant" variables to arrive at the proposed specific or final model. GETS modelling can be summarized into 3 steps:

- Formulation of the general unrestricted model (GUM). The GUM should pass the diagnostic tests (normality, ARCH, Heteroscedasticity, Chow. . .)
- Backwards elimination of insignificant regressors along several paths. When an explanatory variable is removed, a joint significance test and the diagnostic tests are made
- Choice of the best terminal model according to a fit criterion (AIC, BIC, for example)

This model has two main properties. First, all the relevant regressors in the starting model (GUM) will be retained in the final model. Second, the average proportion of irrelevant regressors retained relative to the actual number of irrelevant variables will tend , where is the chosen significance level for t-tests.

2.2.5 Hierarchical Ascending Classification

Hierarchical methods are clustering methods which aim to construct a dendrogram, given a distance d, and they can be ascendant ("bottom-top") or descendant ("top-down"). The distance between two clusters may be defined in several ways, the most popular being single average complete linkage. For ordinal data one may use the Manhattan distance and for nominal data, mutual information may be used. Hierarchical Ascending Classification algorithm is an unsupervised method.

2.2.6 Kmeans

Kmeans is the most widely used method of clustering. It works often with dissimilarity matrix and the number k of clusters need to be fixed in advance. There exist some heuristics to estimate k. We need also to estimate the quality of a partition and to compare partitions. Often, we need to deal with qualitative datasets, numeric, or mixed functional data, image data. Kmeans algorithm is an unsupervised method.

Principle

- Choose arbitrary centers
- Repeat the two following steps until the centers do not change.
  - Step 1 assign each observation to the closest center,
  - Step 2 compute the new center of each class using the observations assigned to that class
- Repeat the previous steps n times and keep the partition with the minimum within-cluster sum of squares given by,

  $\sum_{i=1}^{n} \sum_{j=1}^{k} ||X_i - c_j||^2 1_{i \in G_j}$, where G_j is the jth group and cj is the corresponding center.

2.2.7 K-Nearest Neighbour

The K-Nearest Neighbours is an algorithm used in both regression and classification. It is a supervised machine learning method and the idea of this algorithm is to be based on the K neighbors closest to the point studied in order to predict the class or the group to which it belongs. K represents the number of neighbors.

To make a prediction, the K-NN algorithm will base itself on the entire dataset. Indeed, for an observation, which is not part of the dataset, that we want to predict, the algorithm will look for the K instances of the dataset closest to our observation. Then for these K neighbors, the algorithm will use their output variable y to calculate the value of the variable y of the observation we want to predict.

Note that if K-NN is used for the regression, it is the mean (or median) of the y variables of the K closest observations that will be used for the prediction and if K-NN is used for classification, the mode of the variables y of the closest K observations will be used for prediction.

2.2.8 Random Forests

Random forest is a supervised learning algorithm. It constructs bootstrap samples of the data and leave the out-of-bag (OOB) sample aside. For each node of the tree, it selects the optimal split searching over only log (p) variables among the p ones, selected randomly. The random features are random linear combination of the selected variables at each node.

Properties

- Each tree has a low bias (but high variance)
- Trees are not correlated and this results in high performances
- The correlation is defined to be the one computed between trees' predictions over OOB samples.
- Computational complexity is reduced and parallelization is possible

2.2.9 XGBoost

XGBoost (Extreme gradient boosting) is a supervised machine learning algorithm. Boosting is an algorithm that builds a family of models that are then aggregated by a weighted average of the estimates (in the case of regression). Each model is an adaptive version of the previous one by giving more weight, in the next estimation, to poorly fitted or poorly predicted observations. It concentrates its efforts on the observations that are the most difficult to fit while the aggregation of all models reduces the risk of over-fitting.
The main idea behind gradient boosting is to combine many simple models. And because each tree only provide good predictions on part of the data, more and more trees are added to iteratively improve performance.

2.2.10 Support Vector Machine

SVM (Vector Machine Support) is a supervised algorithm used in regression and classification. The idea is to find the hyperplane that optimally separates the data while maximizing the margin. But it may not be possible to find a boundary that perfectly separates hyperplanes. In this case we may consider hyperplanes which do not perfectly separate the data. The hyperplane is defined as the line which allows to classify the data correctly. The vector supports are the values of the dataset that are close to the hyperplane. Some data is linearly separable while others are linearly non-separable. When a dataset is non-linearly separable, kernel functions should be used. The kernel technique provide a kernel that will allow data to be separated with non-linear decision limits such as the polynomial kernel and the radial basic function.

2.2.11 The forecasting in the previous method (from PCR to SVM)

Since we are working with time series, the data are not yet in the form of (X,y) data. So we need to transform them into this form. Also, note that we want to predict the total load for the next 24 hours starting from 0 am to 11 pm for each days, so this is a problem of multi-step forecasting. There are mainly three ways to tackle this problem :

1. **Direct Multi-step Forecast Strategy**

   The direct method involves developing a separate model for each forecast time step. For example, to predict the total load for the next two hours, we would develop a model for predicting the total load for the first hour and a separate model for predicting the total load for the second hour. If we use just the lags of total load as explanatory variables; the direct multi-step forecast could be written like this:

   $$\hat{y_{t+1}} = model1(y_t, y_{t-1}, \ldots, y_{t-n}) \tag{2.4}$$

   $$\hat{y_{t+2}} = model2(y_t, y_{t-1}, \ldots, y_{t-n}) \tag{2.5}$$

   This method is computationally expensive and since separate models are used, it means that there is no opportunity to model the dependencies between the predictions, such as the prediction on hour 2 being dependent on the prediction in hour 1, as it is often the case in time series. In addition, this formulation could imply to use different value for the hyperparameter of a same model estimated for each hour.

2. **Recursive Multi-step Forecast**

   The recursive strategy involves using a one-step model multiple times where the prediction for the prior time step is used as an input for making a prediction on the following time step. In this setting, the formulation of previous example becomes:

   $$\hat{y_{t+1}} = model(y_t, y_{t-1}, \ldots, y_{t-n}) \tag{2.6}$$

   $$\hat{y_{t+2}} = model(y_{t+1}, y_t, \ldots, \ldots, y_{t-n+2}, y_{t-n+1}) \tag{2.7}$$

   Because predictions are used in place of observations, the recursive strategy allows prediction errors to accumulate such that performance can quickly degrade as the prediction time horizon increases.

3. **Multiple Output Strategy**

   The multiple output strategy involves developing one model that can predict the entire forecast sequence in a one-shot manner. With this strategy, the formulation of the previous example of forecasting the total load for the next 2 hours becomes:

   $$(\hat{y_{t+1}}, \hat{y_{t+2}}) = model(y_t, y_{t-1}, \ldots, y_{t-n}) \tag{2.8}$$

   Multiple output models are more complex as they can learn the dependence structure between inputs and outputs as well as between outputs.

To forecast the total load of the next 24 hours, initially, we consider as predictors the last 72 hours (of course this is arbitrarily, we can consider more or less than that, by relating on the cross correlations) of all variables (price of electricity, the total load itself, the meteorological variables, and even the aggregation of fossil energy generation and renewable energy generation). But when doing that, we encounter several memory issues and the execution of some methods lasted too much. That push us to make many simplifications. So, for the selection and ML methods previously described, we finally use:

- the previous 72 hours of the total load
- the meteorological variables (temperature, humidity ...) at the time for which we want to forecast here we suppose that thanks to weather journal we can know in advance the next times weather. This assumption somewhat arbitrary, but plausible allows us not to have to also use the lagged values of this meteorological variables.
- The price of electricity forecast a day ahead by the Spanish Transmission Service Operator (Red Electric España). To make a realistic forecasting we think that we can't use the actual price at time t to predict the load at time t, because in real world, this will not be known in advance. The same observation is possible for the quantities of energy generated, but in their case, these are variables not used in the literature we have reviewed.
- The dummy variables (holidays dummy, month dummies and other dummies that capture the fluctuation of the total load, we will explain in exploratory analysis)

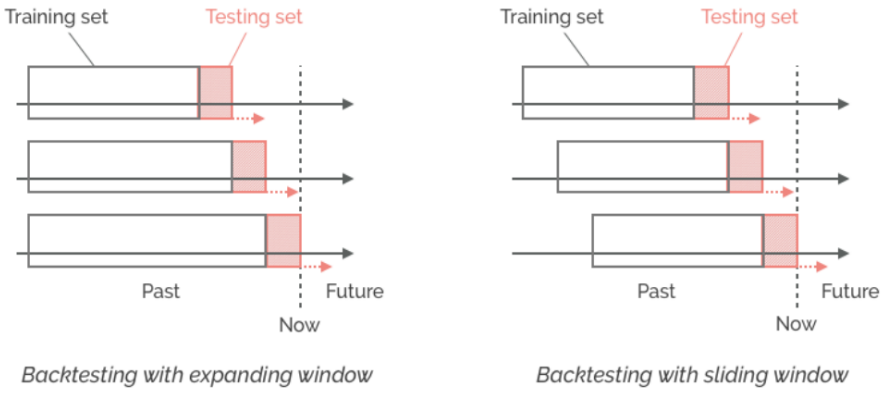
In the case of variable selection methods, we will use the recursive approach to make the forecasting and in the case of ML we will use both recursive and multiple output (or direct multi-step). Since Deep Learning models are able to map input data directly to an output vector, the Multiple Output Strategy will be used in their case.

2.2.12 Walk forward validation (WFV)

As the goal of time series forecasting is to make accurate predictions about the future, the fast and powerful methods that we rely on in machine learning, such as using train-test splits and k-fold cross validation, do not work since we are manipulating time series data. This is because they ignore the temporal components inherent in the problem. There are different methods which can be used to evaluate machine learning models on time series problems, but the standard method used is the Walk-Forward Validation (that we will denote by WFV). In this one, the model may be updated at each time step new data are received. This would give the model the best opportunity to make good forecasts at each time step and we can evaluate our models under this assumption. To apply the WFV, we have to choose:

- the minimum number of observations required to train the model. This may be thought as the window width if a sliding window is used.
- the type of window: sliding or expanding window. This means that we should decide whether the model will be trained on all data it has available (expanding window) or only on the most recent observations (sliding window). In our case, we use the expanding window because it was simpler to code.

The following figure illustrate the expanding and the sliding window:



Backtesting with expanding window *Backtesting with sliding window*

Source:https://www.datapred.com/blog/the-basics-of-backtesting

In the case of the recursive estimation: the models are trained and evaluated with the following way:

- Initially the models are trained on 80% of the data and during the training, the total load at time t is explained by explanatory variables mentioned above. So, the general model estimated is:

$$totalloadactual_t = f(total\_load\_actual_{t-1}, \ldots, total\_load\_actual_{t-72}, Price\_ahead_t, Meteo_t, Dummies_t) + \epsilon_t \tag{2.9}$$

  Where f is a non-parametric function or a parametric linear function depending on the model used, and ϵ_t is the error term. $Meteo_t$, $Dummies_t$ are respectively a set of meteorological variables and the set of dummy variables. This equation is similar to the one estimated in Thouvenot et al (2015) where an additively separable semi-parametric model was estimated.

- For the forecasting and the evaluation: the last 72 hours of total load are combined with the values of forecast price, meteorological and dummies (all at 0 am for example) to make the prevision of the total load at 0h. Then, the prevision of 0 am is put in the last 72 hours of total load actual and they are combined to the values of forecast price, meteorological and dummies variables all at 1 am, and the prevision of total load at 1 am is made. We repeat the process until 11 pm. Then, the actual or real value of the total load from 0 am to 11 pm, and also the value of the other variables from 0 am to 11 pm are added to the training set. After that, a new model is fitted and the prevision of the total load from 0 am to 11pm is made like we described above. We end up with a vector of total load forecast by the model that will be compared to the total load actual and we compute a global RMSE (Root Mean Square Error) on the whole period of the test set and also a RMSE for each hour.

As remainder RMSE$= \sqrt{1/n \sum_{t=1}^{n} (y_t - \hat{y}_t^2)}$, where y_t and $\hat{y}_t$ are the total load actual and the forecast total load at time t.
Note that the methodology described here seems like a chained multi-output estimation, but we could not employ the RegressorChain class in the scikit-learn library, because it would not be possible to keep strictly at each time step, the 72 priors timesteps, in fact it will enlarge the number of prior timesteps. Maybe it is possible, but we do not investigate it. In the case of multi output strategy, we slightly modify the structure of the previous model. The model estimated for KNN and Random Forest (optionally for XGBOOST) is:

$$\begin{gathered}(total\_load\_actual_t, total\_load\_actual_{t+1}, \ldots, total\_load\_actual_{t+23}) = \\ \text{f}(\text{total\_load\_actual}_{t-1}, \ldots, total\_load\_actual_{t-72}, Price\_ahead_t, \ldots, Price\_ahead_{t+23}, Meteo_t, \ldots, \\ \text{Meteo}_{t+23}, Dummies_t, \ldots, Dummies_{t+23}) + (\epsilon_t, \ldots, \epsilon_{t+23})\end{gathered}$$

Where f could be a non-parametric function or a parametric linear function depending on the model used, and $(\epsilon_t, \ldots, \epsilon_{t+23})$ the vector of error term.

Here the forecasting for the next 24 hours of total load, is made directly since an observation of the dependent variable is a vector. After the prediction, the true values are added to the training set, another model is fitted and then the prediction for the following day (the 24 hours of the next day) is made, and so on... Of course, this methodology is also possible for Ridge and LASSO, but since we will be using information criteria to select the best model in these cases, we do not use it, because the formula of these criteria could differ from one dimensional case to multi-output cases, and we do not investigate it. Concerning the SVR (Support Vector Regression), note that the multi-output estimation is not possible in its case, we employ both recursive and direct multi-step (by using MultiOutputRegressor class in the scikit-learn library) but we did not we were not able to obtain a result as it lasts all the time, maybe we should spend more time to find hyperparameters to make it works.

It is worth noting that, while several authors agree that k-fold cross validation should not be used on time series (otherwise there is a risk of predicting the past with information from the future), Bergmeir, Hyndman and Koo (2017) in "A note on the validity of cross-validation for evaluating autoregressive time series prediction" claim that : "When purely (non-linear, nonparametric) autoregressive methods are applied to forecasting problems, as is often the case (e.g., when using Machine Learning methods), the aforementioned problems of CV are largely irrelevant, and CV can and should be used without modification, as in the independent case." In the future it will be useful to investigate in this direction, especially since a package (called forecastML) based on this paper is available and implements all the methods we have mentioned (direct, recursive, multioutput estimates) and applied by hand.

2.2.13 Multilayer perceptron

Multilayer perceptrons (MLPs), also called (vanilla) feed-forward neural networks, or artificial neural networks are a kind of a generalization of linear models performing multiple stages of processing to come to a decision. Commonly used for pattern recognition, classification and prediction, MLP can mimic almost any functional relationship. Multi-Layer Perceptron (MLP) networks's topology includes three types of layers: input, hidden, and output layers. The input layer nodes correspond to individual data sources and the output nodes correspond to the desired classes of information (Hasan et al., 2020). The information in this kind of network goes from the inputs to the output. In the network, each neuron is connected to the one (neuron) following it and the connection can have different weights. As Behm et al., (2020) stated, a model with a small number of hidden nodes already has good generalization properties and with increasing number of nodes, the danger of overfitting increases (which is an obstacle to generalization of the results). But in comparisons with other methods, MLP appears to be more robust and less sensitive to outliers.

2.2.14 Convolutionnal Neural Network

Convolutional neural networks are commonly used when dealing with image or video data, but they can also be used for sequence data like times series. Broadly speaking, a CNN is made up of a sequence of Convolutional layers, each of them is followed by a pooling layer and finally there are fully connected layers. The convolutional layers are kind of hidden layers where each neuron is connected to only a subset of input neurons. So, in the case of time-series (1-dimensional objects): each hidden neuron is connected to a contiguous region of successive values for example: values at t 2, t 1, t, t + 1, t + 2. This is done by sliding a filter, or weight matrix, over the input and at each point computing the dot product between the two (this is the convolution operation). As result all the values in the output feature map share the same weights: the benefit is that the number of parameters is reduced. Since a convolution layer is made of a many filter, repeating the convolution operation for each kernel increases the dimension of the output of the layers, and the pooling layers that follow the convolutional ones, reduce this dimension by carrying out aggregating operation (like max of mean) on the output. This output can then be flattened and feed into the fully connected layers which are just MLP and the prediction will be made. In this process, convolutional layers act as features extraction methods, and have shown efficacity in many predictive tasks.

2.2.15 LSTM

Long Short-Term Memory (LSTM) is a kind of RNN which succeeds to keep memory for a period of time by adding a "memory cell."The memory cell is mainly controlled by "the input gate," "the forgetting gate," and "the output gate." The input gate activates the input of information to the memory cell, and the forgetting gate selectively obliterates some information in the memory cell and activates the storage to the next input. Finally, the output gate decides what information will be outputted by the memory cell. The LSTM network structure is illustrated below (see appendix). Each box represents different data, and the lines with arrows mean data flow among these data.
The recognition procedure of LSTM begins with a set of input sequences $x = (x_1, x_2, \ldots, x_t)$ and finally outputs a set of $y = (y_1, y_2, \ldots, y_t)$, which is calculated according to the following equations:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + b_i) \tag{2.10}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + b_o) \tag{2.11}$$

$$f_t = 1 - i_t \tag{2.12}$$

$$tc_t = g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{2.13}$$

$$c_t = f_t \odot c_{t-1} + i \odot tc_t \tag{2.14}$$

$$mt = o_t \odot hc_t \tag{2.15}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{2.16}$$

In these equations,

- i means the input gate, and o and f are the output gate and the forget gate, respectively.
- tc is the information input to the memory cell, and c includes cell activation vectors, and m is the information the memory cell outputs.
- b is the bias (bi the input gate bias vector)
- and g and h are the activation function of cell inputting and cell outputting, respectively, regarded as *tanh* and *linear* in most of the models.
- the circle with a dot inside is the point multiplication in a matrix.
- ϕ is the activation function of the neural network output.

- W represents weight matrices (e.g., W_{ix} *represents the weight matrix from input x to the input gate i)*

3 Results of the study

3.1 Exploratory analysis

3.1.1 Summary of the data

1. **Load distribution**

Figure 3.1: Total load actual

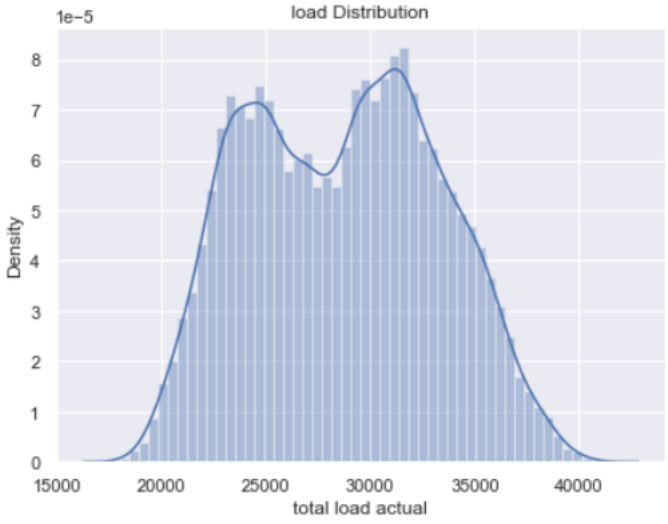


Figure 3.2: Boxplot of Load per year

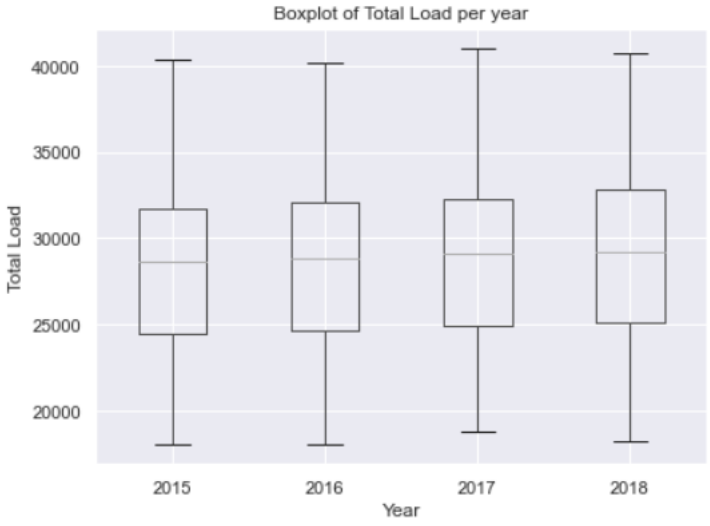


Figure 3.1 shows that the total load has a bimodal distribution and Figure 3.2 shows that the dispersion of the total load did not change significantly from one year to another. This also appears in the graph of the evolution of the total load across the whole period and there are periods of high demand and periods of low demand in each year. For the sake of a clarity, it is not reported here, see the appendices. We will be focusing on a small period to have a better look on the evolution of the total load.

Figure 3.3: Actual Hourly Total load (Zoomed - 2 Weeks)

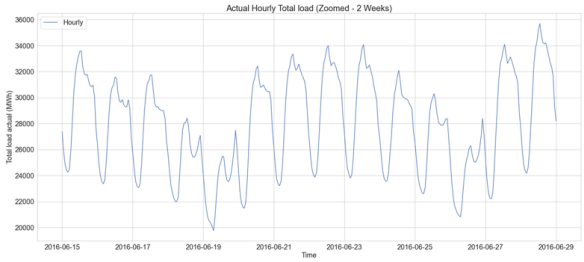


Figure 3.4: Actual Hourly Total load (Zoomed - 1 Day)

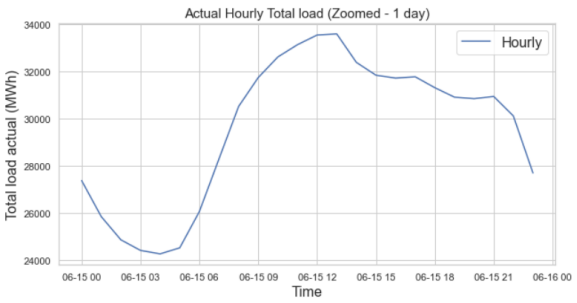


Figure 3.3 and Figure 3.4 show respectively the hourly total load actual from 15/06/2016 (Monday) at 00:00 up to 29/06/2016 (Monday) at 23:00, (i.e. two weeks of data) and the evolution of the same variable just at the 15/06/2016. We can observe that there are many patterns and periodicities, such as:

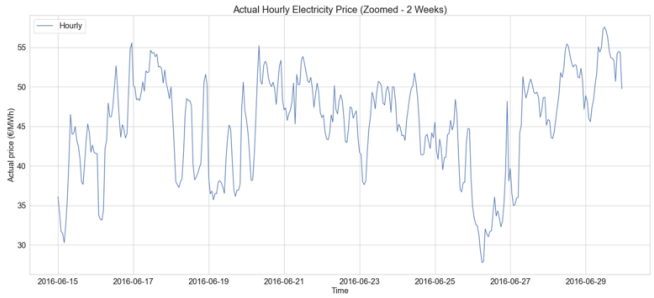
- A periodicity from week to week, as the load evolution across the first and the second week is globally the same. In addition the load tends to be higher during business days and lower during weekends and especially during sundays.

- An intraday periodicity, as the load is higher during certain hours of day and lower after.

Later on, we will investigate more these patterns and we will generate features which contain these kinds of information.

2. **Price evolution**

Figure 3.5: Actual Hourly Electricity Price (Zoomed - 2 Weeks)



From fig 3.5, we can have the same conclusion for the price as for the load, but the price is more volatile.

3. **Load Profile and Shape**
A load profile is a graph of the variation in the electrical load versus time. A load profile will vary according to customer type (typical examples include residential, commercial and industrial), temperature and holiday seasons.

Figure 3.6: Boxplot of the total Load per month

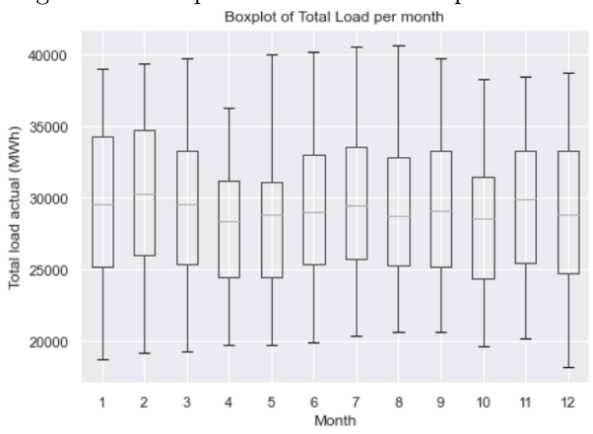


Figure 3.7: Total load per month

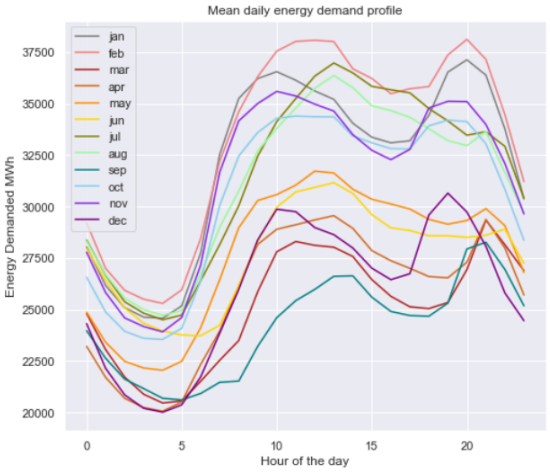
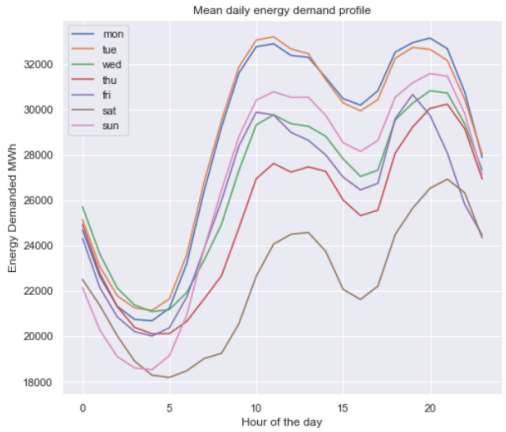


Figure 3.8: Demand profile per day



We can see in Figure 3.6 that the median actual load is highest in January, February, March, Jully and November (months 1,2,3,7 and 11). The first 3 months correspond to winter and Jully corresponds to summer, so it is worth looking closer at temperature data to see if there is a correlation. Furthermore, in most months (Figure 3.7) the evolution of energy consumption is as follows: from midnight to 6 a.m. energy consumption decreases, from 7 a.m. to around 1 p.m. energy consumption increases sharply and then decreases until around 4 p.m. before increasing again until 8 p.m., then it decreases again. This evolution coincides with the working hours in Spain, where traditionally a typical working day tends to be from around 8.30am or 9am (or 10a.m) to around 1.30 pm and then from 4.30pm or 5pm to around 8pm. These two moments of the day are separated by the traditional siesta. The same pattern appears in the demand profile per day (Figure 3.8) where we can also see, as it could be expected, that the total load is higher in the first two days of the week. We also try to see if unsupervised algorithms can find the same patterns.

4. **Autocorrelation and partial autocorrelation**

Figure 3.9: Autocorrelation and partial autocorrelation of the load

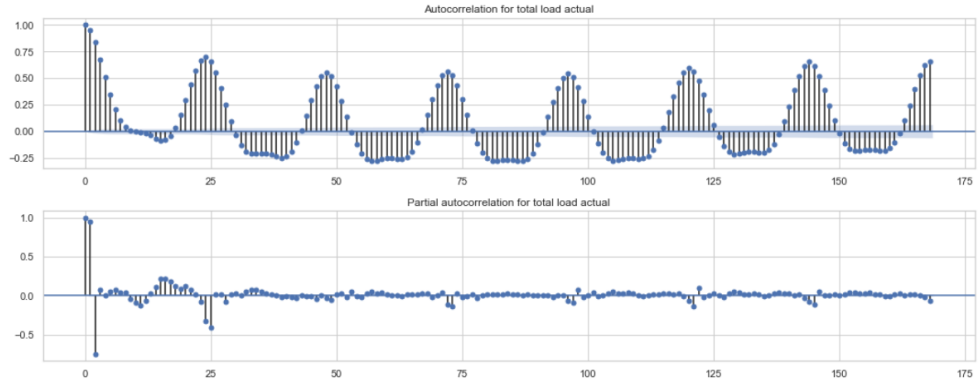
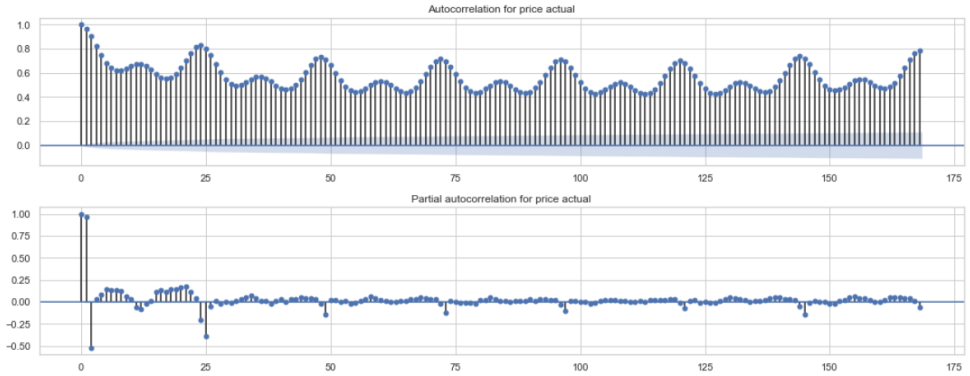


Figure 3.10: Autocorrelation and partial autocorrelation of the price



The partial autocorrelation graphs show that the load and the price at time t are correlated with their values at the two last hours t-1 and t-2, and also with their past values at 24 h, 25h,72h ...

5. **Stationnarity tests on the load values**

Figure 3.11: Stationnarity tests ADF and KPSS results resp.

```
ADF Statistic: -22.206085
p-value: 0.000000
#Lags used: 52
Critical Value (1%): -3.430537
Critical Value (5%): -2.861623
Critical Value (10%): -2.566814
```

```
KPSS Statistic: 0.981730
p-value: 0.010000
#Lags used: 52
Critical Value (10%): 0.347000
Critical Value (5%): 0.463000
Critical Value (2.5%): 0.574000
Critical Value (1%) : 0.739000
```

Fig 3.11 presents the results of the stationnarity tests on the log of load. In both tests (ADF H0 : presence of unit root H1 : no unit root, KPSS: H0 : No unit root; H1: Unit rot) we reject the hypothesis of unit root in the log of load at the level of 5%.

6. **Relationship between the total load and the temperature**

Figure 3.12: Temperature and Energy consumption

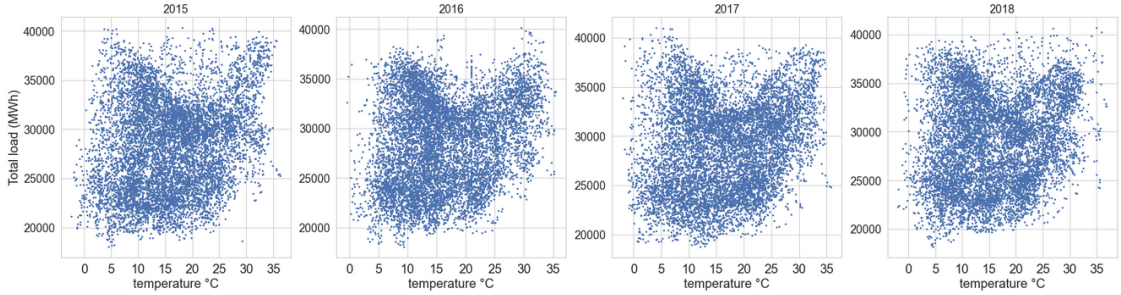


Fig 3.12 which presents the relationship between the total load and the temperature appears to have a U-shaped form for every year.

7. **Correlation matrix**

Figure 3.13: Correlation Matrix

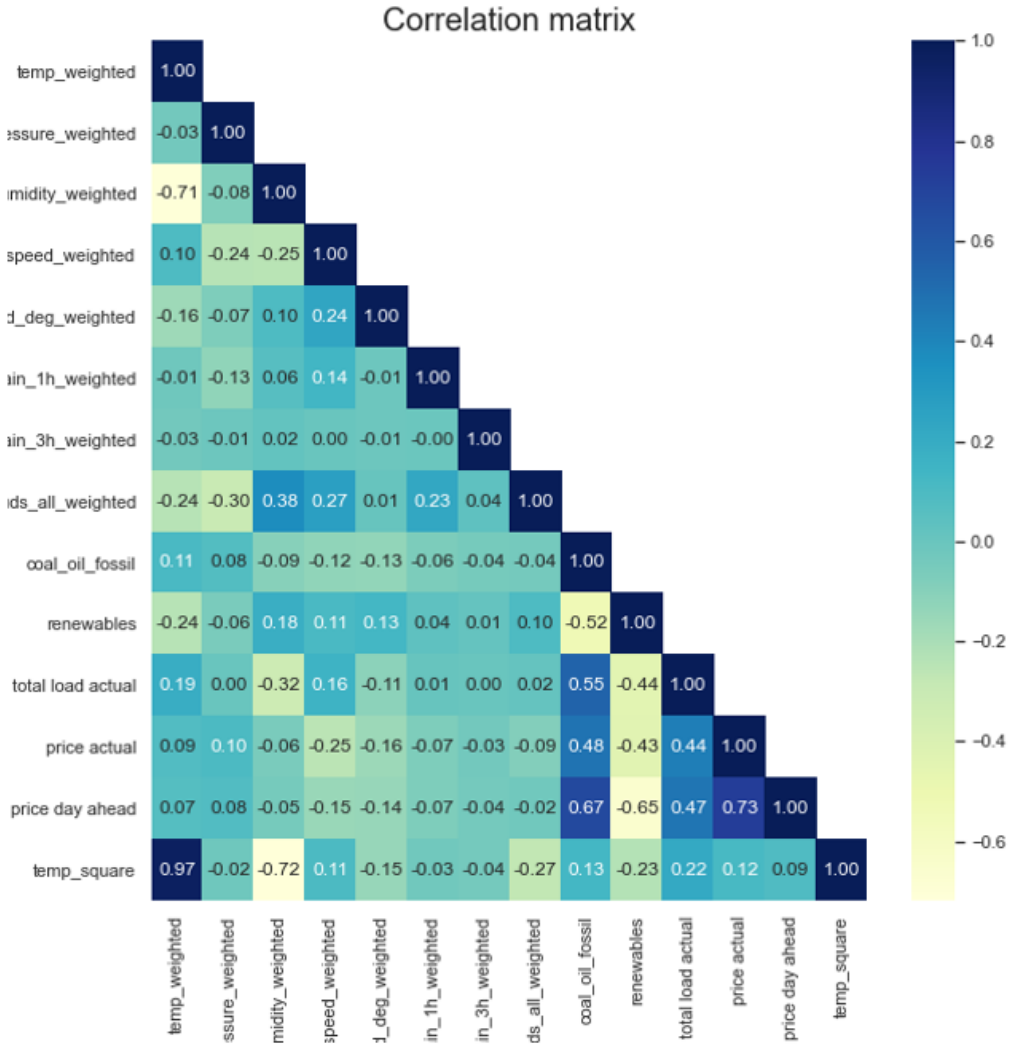


Figure 3.12 that present the correlation matrix. It shows that :

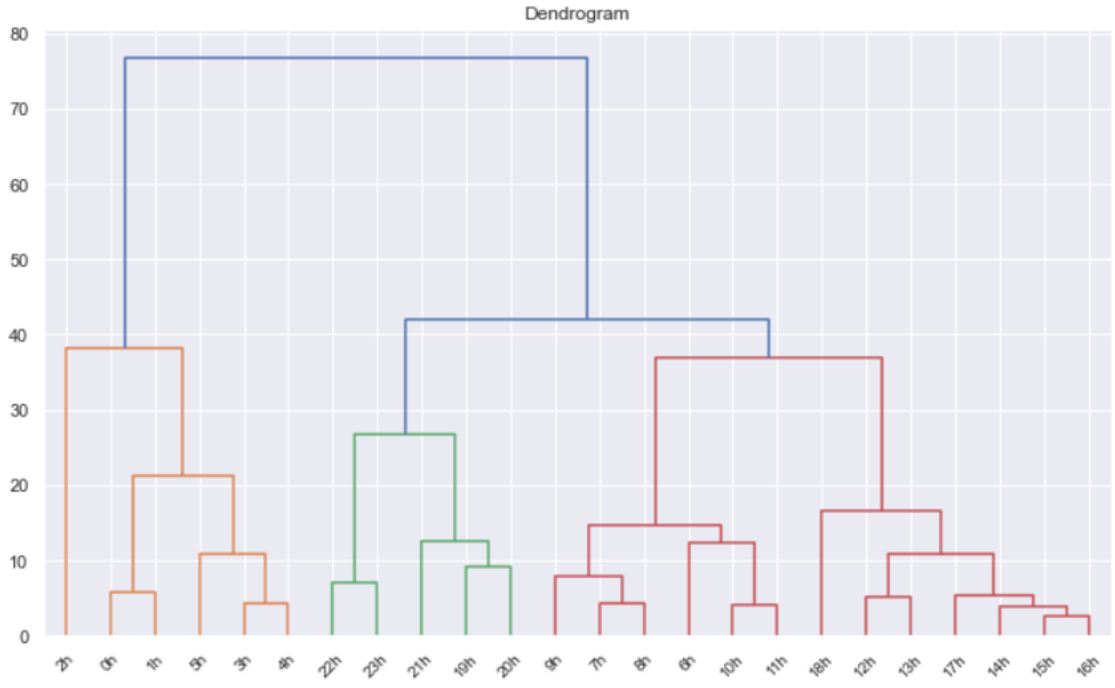
- there is not a strong linear correlation between total load actual and the climatic variables (temperature, humidity, rain, clouds),
- the energy generated from renewable sources (renewables) are negatively correlated with the total load actual,

- the price actual and its proxy price forecast a day ahead, whereas the total energy load and the energy generated from non renewables sources (coal_oil_fossil), are positively correlated with the electricity price.

3.1.2 Unsupervised methods

1. **Hierarchical Ascending Classification**

Figure 3.14: Dendrogram



The dendrogram suggests mainly 2 groups : the day (6h - 23h) and the night (0h - 5h). But we could also say 3 groups as depicted by the colors orange, green and red.

2. **Kmeans**

Figure 3.15: Elbow Method For Optimal k

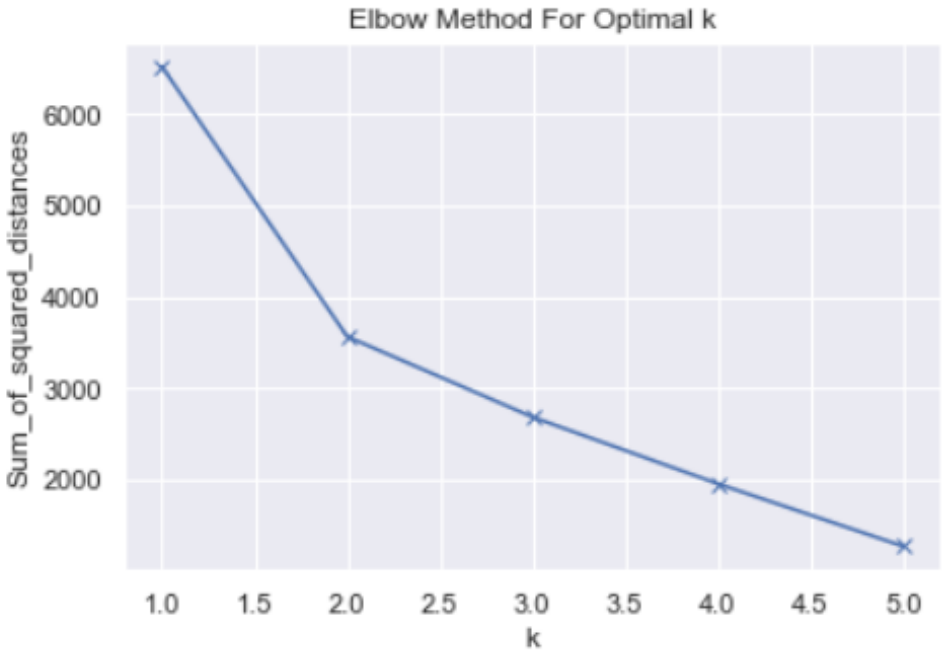


Figure 3.14 suggests that k = 2 groups is the optimal value. This is in line with the result of the hierarchical ascending classification (the Kmeans prediction of the groups of each hour is the same as the prediction of the ascending classification).

Based on this exploratory analysis, we have created the following dummy variables:

- dummy variables for each month, since the total load could significantly vary from a month to another due to seasonality.
- following Romero et al. (2018), we create a dummy for holidays: this dummy indicates if each day is a public holiday or not.
- dummy_work: indicates if the time is between 9 am and 1 pm or between 5 pm and 8 pm (the bounds are included)
- dummy_siesta : indicates if the time is between 2 pm and 4 pm (bound are included)
- dummy_night : indicates if this is the night or the day
- dummy_jour_ouvra: indicates if the day is between Monday and Friday.
- dummy_samedi: indicates Saturday.
- dummy_diman : indicates Sunday.

We also include the square of the temperature.

3.2 Results of the predictive analysis

General settings

As stated in the introduction, our goal in the current study is to forecast the 24 hours of the electrical energy consumption in Spain in advance and try to do better than the ENTSOE (European Network of Transmission System Operators for Electricity: ENTSOE, please visit this web site to see their forecasting : https://transparency.entsoe.eu/dashboard/show).

- Since we do not have a human error reference, we will compare the results of our models to the RMSE of ENTSOE prediction and consider it as the Bayes optimal error (theoretical limit that represents the smallest error with respect to which a human or a system cannot do better in predicting a target variable Y, given the explanatory variables X). At each new day, the ENTSOE publishes its forecasting of the total load for the next 24 hours.
- Our data cover the period of 01 January 2015 00:00 am to 31 December 2018 11 pm, so we have 35064 observations. The data is split into 80% for the training set (01/01/2015 0 am to 15/03/2018 11 pm) and 20% for the test set (16/03/2018 0 am to 31/12/2018 11 pm). Note that we could lose the first observations due to the lags and the right dates may slightly differ after lagging. One very important thing is that given the description of the WFV approach in section 2.2.11, this implies that in order to evaluate a given model, the number of times the model has to be re-estimated is equal to the number of days in the test set (the model is updated to estimate the next 24 hours). This implies a long computation time, especially when the interval in which to search for the optimal value of a hyperparameter is wide or when we want to test different values for several hyperparameters. Therefore, we will try to avoid searching for the optimal values of hyperparameters in wide intervals with the WFV.
- The training set is standardized, and the transformation is applied to the test set. So, after the prediction, we rescale the forecasting by applying the inverse of the standardization.
- Given the range of date of the test set, the RMSE of the ENTSOE prediction is computed for the same range; thus, this RMSE is 378.31 MWh.
- Let us also remind that the supervised method that we will discuss below (except deep learning model) use as explanatory variables: 72 lagged values of total load actual, 19 dummies (see the different dummies in the subsection 3.1) and 10 numerical variables including the price forecast and 9 meteorological variables : temp_weighted, temp_square pressure_weighted,humidity_weighted,wind_speed_weighted, wind_deg_weighted, rain_1h_weighted,rain_3h_weighted, clouds_all_weighted.

3.2.1 Principal Components Regression (not with PCA but with FAMD)

Since we have dummy variables in our data, applying a principal component regression is not suitable. This is because the dispersions are different for the dummies and for the numerical variables and the results of a principal component regression are likely to be biased (Rakotomalala, 2012). Thus, we use factorial analysis of mixed data (FAMD, in FactoMineR) and we fit it on the train set, apply the transformation on the test set, get the coordinates of the observations on the different factorial axes, and for each number of factorial axes that was possible the Walk Forward Validation (WFV) is run to determine the optimal number of factorial axis. It is worth nothing that the lagged values of total load are not included in the FAMD, so the FAMD is fitted on the 29 remaining variables. This is because if these lagged values were included, we will not be able to do the recursive estimation due to identification issue.

Figure 3.16: Eigenvalues histogram

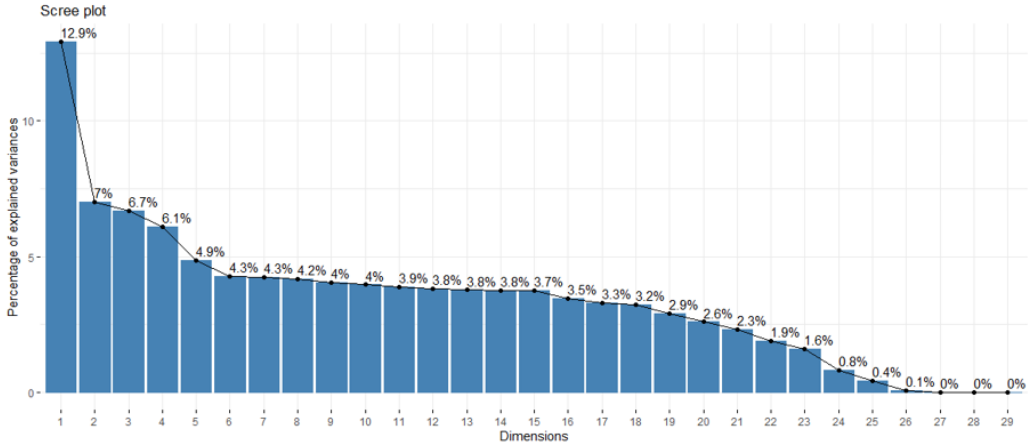
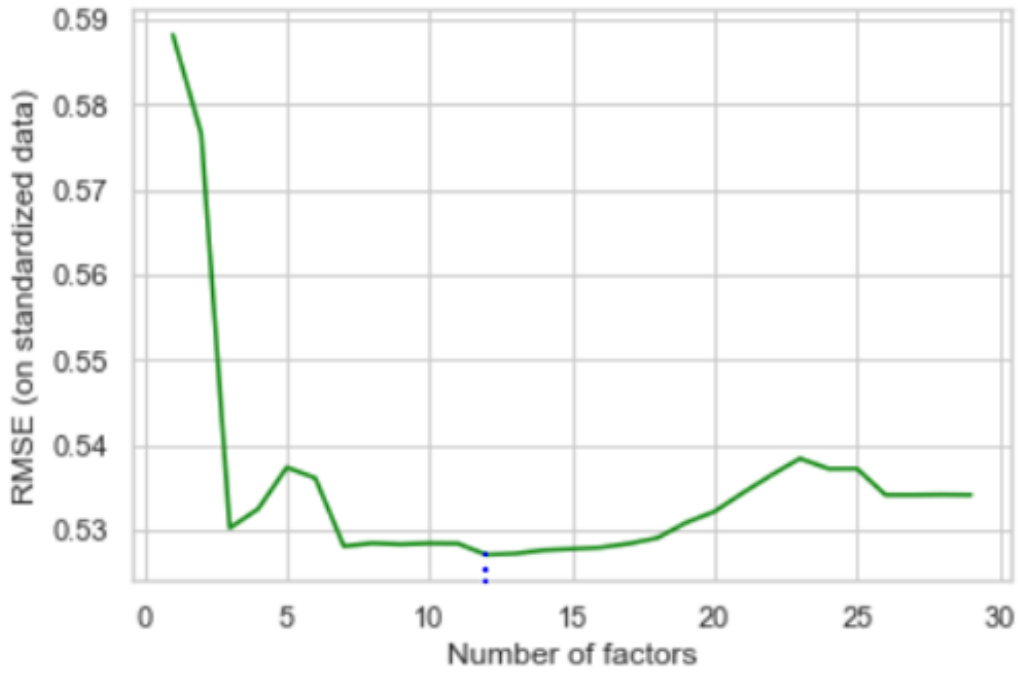


Figure 3.16 shows the plot of the eigenvalues/variances against the number of dimensions. There are 29 eigenvalues (10 for numerical variables and 19*(2-1) for dummies). About 20% of the information contained in the data used is captured by the first two factorial axes, and 80% of the information is held by the first 16. we can see an elbow between the first and the second dimension and another one between the 4th and the 5th dimension.

Figure 3.17: Evolution of RMSE with the number of factors



To choose the optimal number of factors that should be included in the regression to make the forecasting, the WFV is achieved for each possible number factorial axes going from 1 to 29. A given regression in this case include the 72 previous hours of total load and the principal components. Figure 3.17 shows that the minimum RMSE (com-

puted on the standardized data) is reached when the principal components is 12. So instead of using 29 variables with the lagged variables in the regression, we will use the first 12 principal components. The RMSE is then computed after rescaling the prediction and we found a RMSE of 2419.21 MWh, meaning that the predictions deviate on average from the actual values by 2419.21 MWh on the whole test set., that is very far from the ENTSOE prediction.

Note that one drawback of this method that we used, is like a drawback of PCR (*rincipal component regression), which is that: "There is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response." James, Witten, Hastie and Tibshirani (2017). One could envisage a Partial Least Square (PLS).

3.2.2 Ridge Regression

To choose the best value of lambda , we proceed by the efficient implementation methods which consist in applying the following algorithm:
For i =1,. . . , N, deduce λ_i from df_λ with

$$df_\lambda = \sum_{j=1}^{N} \frac{d_j^2}{d_j^2 + \lambda_i} = i \tag{3.1}$$

For λ_i , i =1,. . . , N, compute $\hat{\beta}_{\lambda_i}^{Ridge}$, i =1,. . . , N

Where N is the number of variables, λ the tuning parameter, $d_j^2, (j = 1, \ldots, N)$ the eigenvalues of the matrix X'X.

The value of lambda is computing by solving the equation (3.1) with Newton-Raphson algorithm, and for each lambda the ridge regression is estimated. Since the total number of variables to be used here in the recursive forecasting is 101 (72 lagged values of total load, 19 dummies and 10 numerical variables including the price forecast and 9 meteorological variables), we have 101 different values for the λ, going from 2.93 to 2118552.21. Regarding the number of λ that we have and the way we designed the Walk Forward Validation (WFV), computing a RMSE after the WFV for each lambda values in order to look for the best λ, will be very time consuming. To avoid this, the BIC (Bayesian information criteria) of ridge regression for each λ is computed on the training set and the optimal λ is the one which minimizes the information criterion. The graph of the ridge coefficients paths is available in appendices.

Figure 3.18: Evolution of BIC

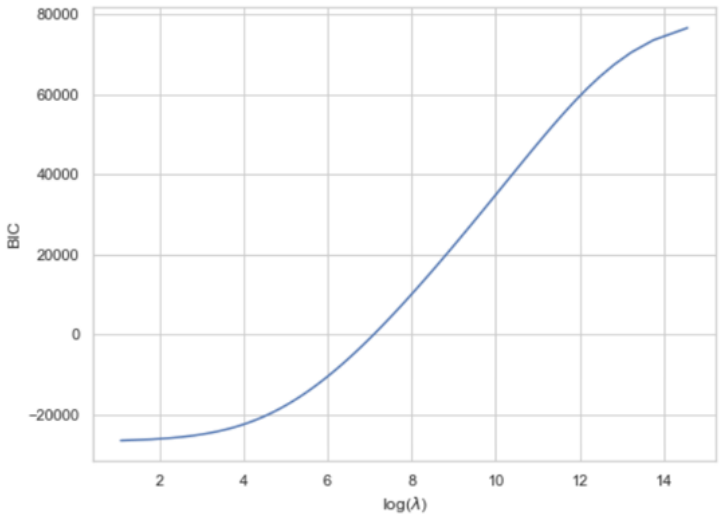


Figure 3.18 shows the BIC for each ridge regression with the corresponding lambda. The more lambda grows, the more the degree of freedom will drop toward 0 and the BIC increases. The penalty which minimizes the BIC is the lowest: 2.93; and based on this value the RMSE for the forecasting using the WFV is 2451.43 MWh, meaning that the predictions deviate on average from the actual values by 2451.43 MWh. This is very big for the electrical industry, where accuracy is essential, otherwise the grids will not be efficient.

Table 3.1: Some coefficients in the "best" ridge regression

The 10 highest coef in absolute value				**The top 5 coefficients closest to 0 in absolute value**	
positive		**negative**			
y_lag_1	1.304947	**y_lag_2**	-0.424584	rain_1h_weighted	0.000096
y_lag_24	0.277665	**y_lag_25**	-0.331012	wind_speed_weighted	0.000385
y_lag_71	0.178031	**y_lag_70**	-0.110411	wind_deg_weighted	0.000425
y_lag_27	0.116285	**y_lag_72**	-0.097462	pressure_weighted	0.000778
y_lag_23	0.103550	**y_lag_52**	-0.094752	june	0.000848

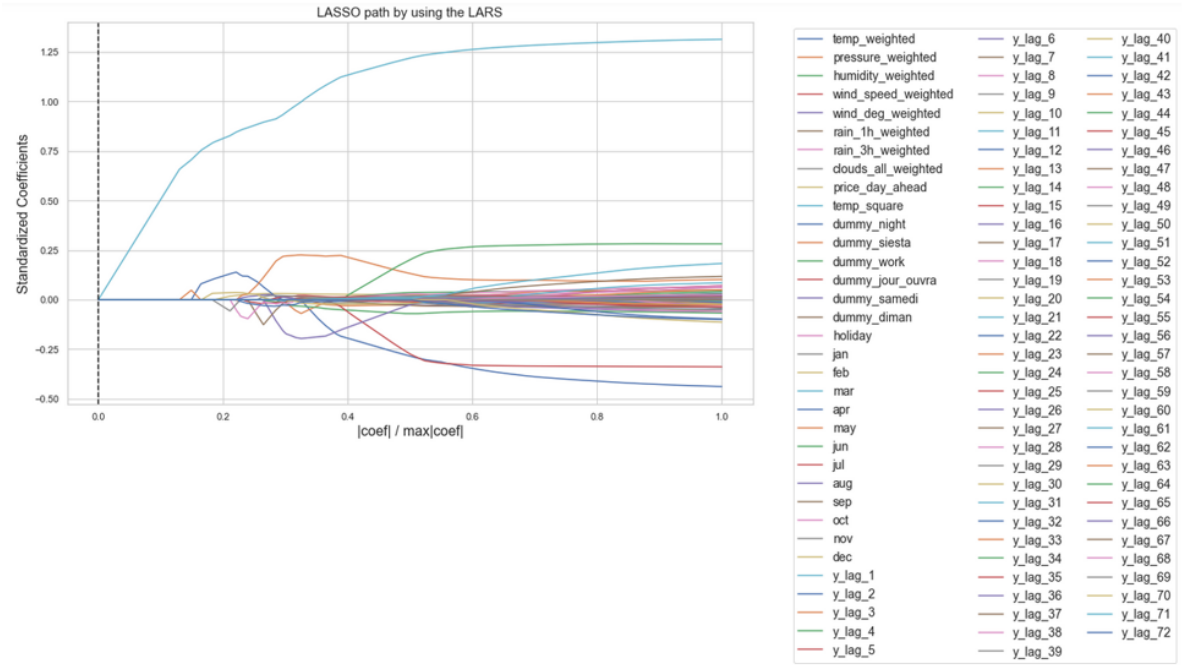
The y_lag variables are the lagged values of the total load.

Table 3.1 shows the 10 highest coefficients in absolute value and the top 5 coefficients closest to 0. Remember that in this case of the recursive forecasting, we are explaining the total load at a hour t, by the remaining variables. The ridge has shrunk some variables meteorological that clearly do not directly impact the energy demand. As shown in the autocorrelation graphs, the demand at a given hour is hugely impacted by the demand at the previous hour (y_lag_1), the demand at the same hour at the previous day (y_lag_24) and the demand at the same hour 3 days before (y_lag_72).

3.2.3 Lasso LARS (Least Angle Regression)

We use the LassolarsIC class of scikit learn to fit Lasso model with Lars using BIC (or AIC) for model selection. The pursued objective was the same: avoid using a range of value of the penalty, compute the RMSE by using the WFV, in order to find the optimal penalty, because the WFV was time consuming. Instead, LassolarsIC class offers us the possibility to fit the LASSO model and select the optimal hyperparameter regarding the one which minimizes the BIC. This allows to select the value of the regularization parameter by making a trade-off between the goodness of fit and the complexity of the model. The regularization parameter had been chosen in a range of 3.84e-08 to 5.68e-03 and the optimal value was 3.28e-05. Lasso path is the following:

Figure 3.19: Lasso path



In Figure 3.19, we observe that at first the lasso results in a model that contains only the total load of the previous hour (y_lag_1). Then y_lag_23 enter the model and after the remaining variables enter the model.

Figure 3.20: Non-null coefficients of "best" LASSO

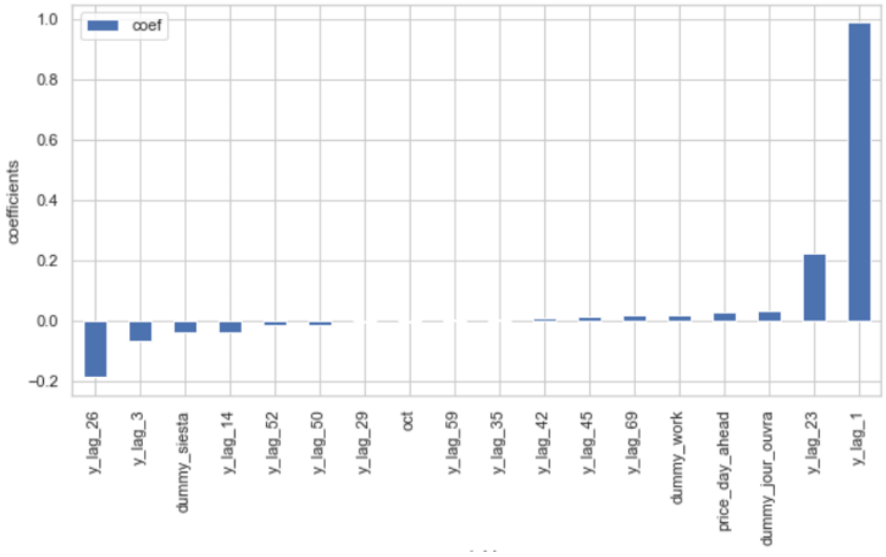


Figure 3.20 shows the non-null coefficients with the optimal regularization parameter. As the ridge, the Lasso selects the demand at the previous hour (y_lag_1) and the y_lag_23 as the highest which influence positively the demand at hour t, but in addition the Lasso also selects the dummies related to the business hour (dummy_work), the working days (dummy_jour_ouvra) and also the proxy of the actual price of electricity at hour t (price_day_ahead) to be positively correlated with the demand at hour t. We can also see that the dummy indicating time between 2 pm and 4 pm (dummy_siesta) i.e. break time, has a negative effect on the demand. To our surprise, the Lasso sets the coefficients of all the meteorological variables to 0, while we were expecting to see them (at least the temperature) impacting the energy consumption. Perhaps the relationships between the energy demand and the meteorological variables are more complex than a simple linear or U-shaped relationship and could combine some interactions effects. With the selected regularization parameter, we compute the RMSE with the WFV and obtain 2533.67 MWh, worse than the ridge.

3.2.4 General to Specific (GETS) modeling

Using the Stata general to specific modeling approach, the following GUM (General Unrestricted Model) has been specified:

$$total\_load\_actual_t = \alpha_0 + \sum_{i=1}^{72} \alpha_i * total\_load\_actual_{t-i} + \alpha_{73} * Price\_ahead_t + Meteo_t * \beta + Dummies_t * \gamma + \epsilon_t \quad (3.2)$$

This is same the model in Ridge or in LASSO, but without the regularization term, as the GETS will find itself the appropriate model. GETS modeling assumes there exists at least one specification within the GUM that is a statistically valid representation of the DGP: the Local Data Generating Process). Equation (3.2) is estimated on the training set and with robust standard errors (because in a first setting we encountered heteroscedastic errors). The GUM failed 3 of 4 misspecification tests:

- Doornik-Hansen test rejected normality of errors.
- The test for ARCH (AutoRegressive Conditional Heteroskedasticity) components is not rejected.
- The in-sample Chow test rejects equality of coefficients

And this GUM performs poorly. Only the Out-of-sample Chow test for equality of coefficients was not rejected. When we took the variables that were selected by this GETS at the level of 5% and compute the RMSE we get a very bad result (the selected model is available in appendices). This bad performance could be explained by the fact that there are arch effect and possible structural break since the test for ARCH components is not rejected and the in-sample Chow test rejects equality of coefficients. Therefore, we tried to solve these problems by using the gets package of R because it offers the possibility to detect outliers and structural breaks (like in Autometrics) but in addition we can specify a GUM with arch effect. « The gets package, at the time of writing, is the only statistical software that offers GETS modeling of the conditional variance of a regression, in addition to GETS modeling of the mean of a regression, and indicator saturation (IS) methods for the detection of breaks of outliers structural breaks in the mean of a regression using impulses (IIS), step (SIS; see Castle, Doornik, Hendry, and Pretis 2015) as well as trend indicators

(TIS).» Pretis, Reade and Sucarrat (2018). We attempted to use this package and specify that we want to achieve a GETS modeling with arch effect (the package modelize the logarithm of the conditional variance) with step-indicator saturation, but we got memory issue (see the error in the appendices). To solve this, it would be possible to reduce the number of variables but doing so, we will no longer use the initial GUM variables. Hence, we no longer continue this investigation.

3.2.5 Summary of the result of models of variable selection or dimension reduction

The following table present the summary of the result of models of variable selection or dimension reduction.

Table 3.2: Summary of the result of models of variable selection or dimension reduction

Models	**Reg on FAMD**	**Ridge**	**Lasso**	**ENTSOE**	**Persistance 1 Day**
RMSE (MWh)	2419.21	2451.43	2533.67	378.31	3507.52

Between the different selection method that we used, it is the regression on the factorial axes of the FAMD, which performs better, but we are very far from the ENTSOE result. All the models perform better than a simple persistence 1 day model (using the observed values from the previous day as the prediction of the next day). Please, see the appendices for more details on the RMSE per hour for the different models.

Figure 3.21: RMSE per day of week and per hour of models of variable selection or dimension reduction

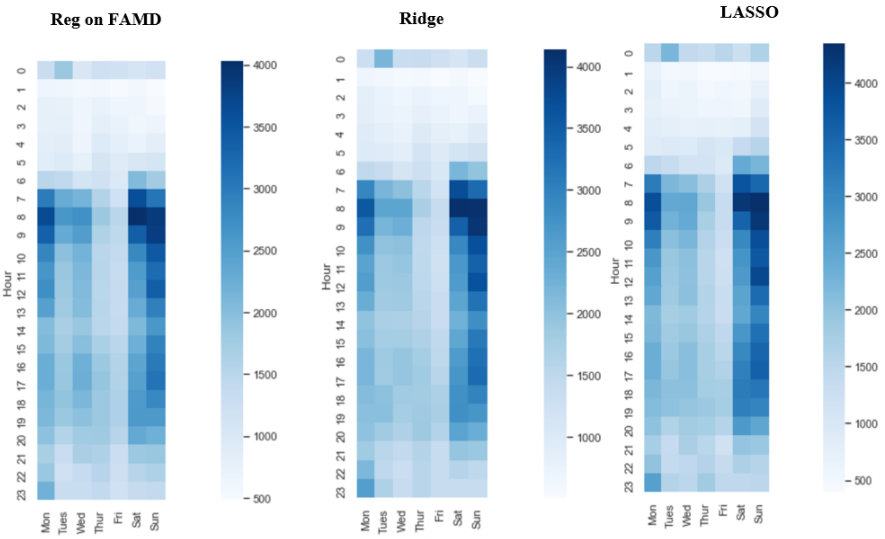


Figure 3.21 depicts the RMSE per day and per hour of selection or reduction models. The errors in the 3 models are quite similar. We can see that the predictions are better during the night (RMSE about 500 MWh between 23 pm to 5 am), probably because in night the demand is more stable, and then the models perform better. The models have good prediction in the middle of the week (Thursday and Friday). The total load at Saturday and Sunday seem to be badly predicted with a huge error at 8 am. A graph of the prediction for the first two week is available in appendices.

3.2.6 K-Nearest Neighbors (KNN), Random Forest (RF), Gradient Boosting (XGBOOST), Support Vector Regression (SVR)

In front of the myriad of hyperparameters intended for each method and given the Walk Forward Validation WFV run time it takes for just one of these models we have not performed a gridsearch of the hyperparameters, due to time constraint. Instead, we consider just one of the important hyperparameters for each method and look for an optimal value for this hyperparameter out of at most four possible (arbitrary) values. Furthermore, let us remind that in the multioutput estimation, the output is a vector of R^{24}, as there are 24 hours. The settings that have been used are the following:

- **KNN** : It is the number k of neighbors which is varied. Both in the recursive and multioutput estimation $k =$ 5,10,15 or 20.
- **RF**: We vary the number of trees. In the recursive estimation n_tree = 10 or 15. In the multioutput estimation n_tree = 100, 150 or 200. This huge difference is because the multioutput estimation is faster, and so we can increase the number of trees. Normally to do a good comparison we should also set n_tree = 10 or 15 in the

multioutput case. Note that due to computation time, the numbers of trees are very reduced compare to what can be seen usually.

- **XGBOOST**: Like in RF the number of trees is varied; here we just implement the recursive estimation. We also tried the direct multistep but with no results. The n_tree = 50 or 100. We let the loss function on the default setting, which is the squared error, but for robustness, squared error (least squares) is not the best criteria, a huber lost that combines both least squares and least absolute deviation loss should be better.
- **SVR**: Initially, we were using the SVR class of scickit learn with the RBF kernel (radial basis function), but it was hard to get its results, as the number of observations in our training set (28008 in the recursive case), largely exceed the limit of 10000 sample required. Now, we use the LinearSVR, let the loss function to the default: -insensitive (L_1loss, or precisely an approximation to Huber's loss function that enables a sparse set of support vectors to be obtained unlike the quadratic and Huber cost functions). The error-epsilon is set to 0 (default) and 0.2. Note that a direct multi-step forecasting is performed with the same settings.

The results of these methods are the following:

Table 3.3: RMSE of some machine learning methods on the test set

	KNN		**RF**		**XGBOOST**	**Linear SVR**	
	Recursive	Multioutput	Recursive	Multioutput	Recursive	Recursive	Direct multi-step
RMSE (MWh)	2286.32	2271.13	2433.98	2124.78	2375.63	2994.35	2996.88
"best" hyper-parameter	k = 20	k = 5	n_tree = 10	n_tree = 200	n_tree = 50	ε=0.2	ε=0.2

Table 3.3 shows that it is the Random Forest with 200 trees in multioutput case which performs the best with a RMSE of 2124.78 MWh, this could be due to a better handling of non-linearity and interactions between variables. But compare to the ENTSOE results (378.31 MWh), we are still very far from allowing to the grid to be efficient as the error is very large. Note that the execution time of the RF method for just the 3 different number of trees in multioutput case with the WFV lasted more than 11 hours, in the direct multi-step of XGBoost and Linear SVR (just 2 values for epsilon) cases, it lasted more than 1 day ; so we should try to optimize the code used and parallelized the execution if it is possible.

The best KNN in multioutput case performs slightly better than KNN in recursive case, this may be possible because in the recursive estimation the error in the prediction increases as far as the step or the horizon increases, and the multioutput estimation takes into account the relationship between the component of the output. This is important here, since we saw that the demand at the current depends on the demand of the previous hour and 24 hours before. Both the recursive and direct-multistep linear SVR have the worse results, which is far from the others, this may be due to the convergence issue that occurred during the different trainings of this method, despite the increase in the number of iterations. All the models perform better than a persistence 1 day model (3507.52 MWh) which use the observed values from the previous day as the prediction of the next day. For more details on the RMSE per hour for the different models, please, see the appendices.

Figure 3.22: RMSE per day of week and per hour of 'best' of Machine learning methods used:

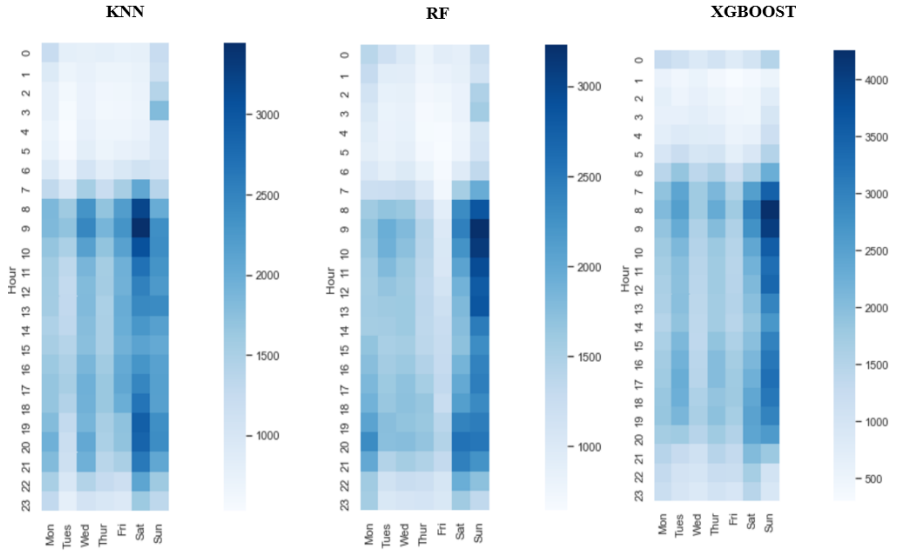
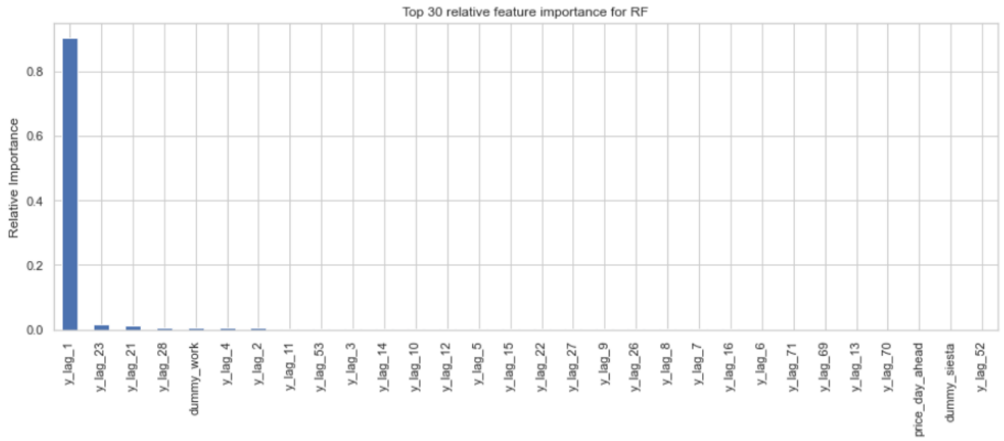


Figure 3.22 depicts RMSE per day of week and per hour of 'best' of Machine learning methods used. As in the selection and reduction models, the ML models predict well in night. KNN failed to correctly predict the total load of Saturday and precisely at 8 and 9 am and predict well Tuesday and Thursday. RF predicts quite well comparatively to the others model the whole week except the Sunday particularly at 9. XGBOOST also failed at Sunday. A graph of the prediction for the first two week is available in appendices. One of the drawback of the multioutput estimation might be the loss of interpretability (of course one can compute the variables importance, but we are not very sure about its interpretation), so we will focus on the variables importance of the recursive RF (RF with single output).

Figure 3.23: RMSE per day of week and per hour of 'best' of Machine learning methods used:



Again, the total load at the previous hour is the most important factor in explaining the next. Apart from the lagged variable, the work dummy and the price are also important in the explanation of the total load. But what about the meteorological variables? Maybe the fact that we have used the weighted average of these variables in different towns leads to their insignificance.

3.2.7 Deep Learning Models

Here we change the methodology that we used so far. Indeed, since neural networks are generally slow to train, we do not update the models after a new prediction, so the models are static. Put it simply, we no longer use the walk forward validation, we just evaluate the model by using out of sample observations. The data is split into 65% for the training set (01/01/2015 0 am to 29/01/2017 11 pm), 15% for the validation set (30/01/2017 00 am to 14/03/2018 11 pm) and 20 % for the test set (15/03/2018 0 am to 31/12/2018 11 pm). The models use the information of the 30 previous days to predict the total load the 24 hours of total load for the next day. Of course, this can be a hyperparameter for the models, we can consider the 3 or 7 or 14 previous days for example, 30 is used to consider possible monthly seasonality. As a result, it becomes possible to use variables such as the actual price (instead of the price forecast ahead), the actual amount of renewable and fossil energies generated. As in previous models we are using the standardized data and one additional preprocessing was the windowing of the datasets: this allows to have a vector

of past timesteps that will be used to predict on a target vector of future steps. As a result, for a mini-batch size of 256 for example its dimension will be: [256, 24 ×30, number of variables =32] for the explanatory variables and $[256, 2430, 1]$ for the target one.

General setup

- Learning rate: 1e-4
- Loss : mean squared error
- Optimizer: Adam
- Batch size = 256, epochs = 200
- Early Stopping with patience =10 and Reduce learning rate by 0.1 when MSE on validation set has stopped improving for patience =5 and save the best model only.

Architecture

The following are the architectures that we kept after some hyperparameters tuning:

1. Multi-layer perceptron (MLP)
    - Flatten the input
    - 1st dense layer with 256 neurons with ReLU activation, followed by a dropout of 0.2 to avoid overfitting
    - 2nd dense layer with 256 neurons with ReLU activation, followed by a dropout of 0.2,
    - Output layer: dense layer with 24 neurons.
2. Convolutional Neural Network (CNN)
    - 1D convolutional layer with 64 filters of size 6 with ReLU activation,
    - 1D MaxPooling with pool_size = 2,
    - 1D convolutional layer with 64 filters of size 3 with ReLU activation,
    - 1D MaxPooling with pool_size = 2 and then flatten the output
    - Dropout with a rate of 0.2
    - Dense layer with 128 neurons without activation, then a second dropout with a rate of 0.2
    - Dense layer with 24 neurons without activation
3. Recurrent Neural Network based on Long short-term memory (LSTM)
    - LSTM layer of 72 neurons with ReLU activation function and return the full sequence,
    - LSTM layer of 72 neurons with ReLU activation function and return the last output
    - Flatten the previous output and apply a dropout with a rate = 0.2
    - Dense layer with 128 with ReLU, and apply a dropout with a rate = 0.2
    - Dense layer with 24 neurons without activation

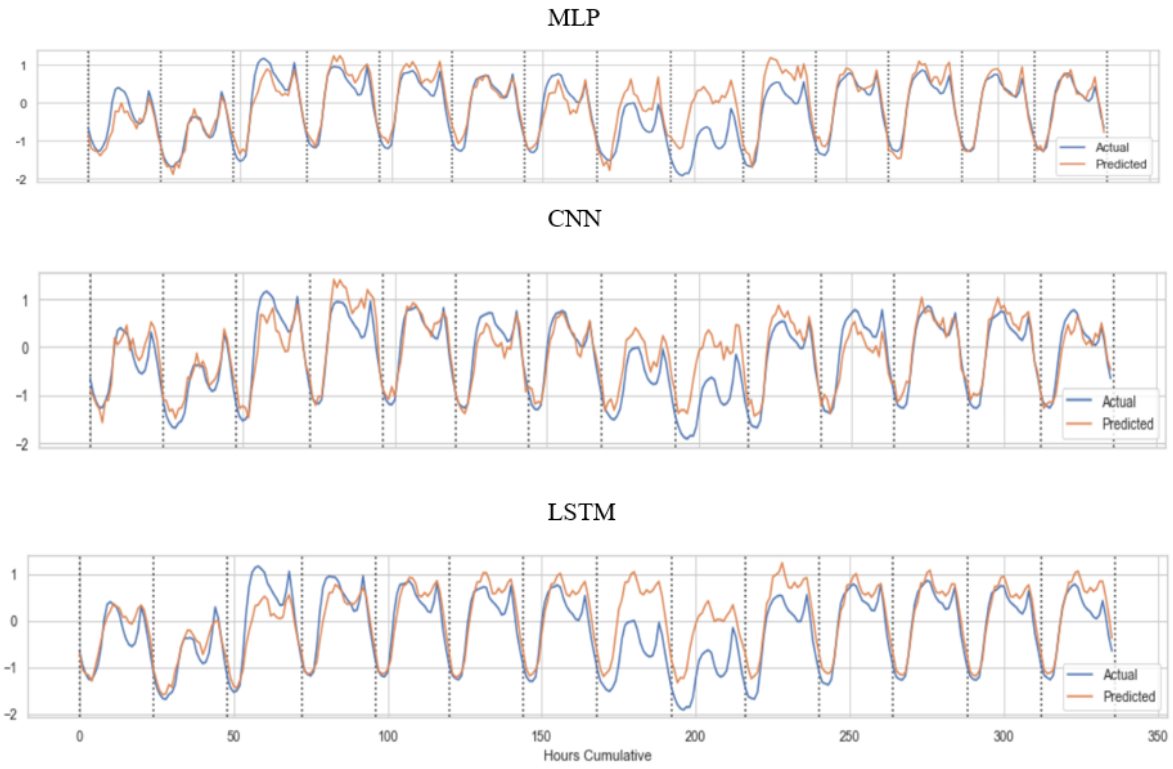
The CNN-LSTM encoder-decoder architecture is described in our notebook.
Here are the results we got:

Table 3.4: RMSE of some deep learning models on the test set

Models	**MLP**	**CNN**	**LSTM**	**ENTSOE**	**Persistance 1 Day**
RMSE (MWh)	2707.02	2596.30	2819.22	378.31	3507.52

We are still very far from the ENTSOE standard, we were expecting to get better result with the LSTM, but it was the last; other configurations should be tried. The CNN is the better between these 3 models. We cannot directly compare the result of these deep learning models with the previous ones because of the difference in the way they are evaluate (deep learning models are evaluated with out of sampling whereas the previous are based on walk forward).

Figure 3.24: Predictions of the First Two Weeks of test set after lagging: 14/04/2018 – 27/04/2018 Ordinate : total load actual standardized; abscissa : cumulative hours



We can see that the models tend to replicate the different changes in the total load actual, but they are not accurate, the CNN-LSTM encoder-decoder smooths but it is not quite accurate (see appendices)

3.3 Discussion and limits of this work

Table 3.5: RMSE of the different "best" models

Models	**Reg on FAMD**	**RF (multi)**	**CNN**	**ENTSOE**
RMSE (MWh)	2419.21	2124.78	2596.30	378.31

In models of variable selection or dimension reduction, we have seen that it is the regression on the principal components of FAMD which perform the best, and in the case of the other models of machine learning, it is the multiple output Random forest which perform the best. Since the two methods have been evaluated in the same way (i.e., with the Walk Forward Validation: WFV), we can say that it is the Random forest which perform the best. Deep learning models were not evaluated with the WFV, but with out of sample observations, and the "best" was CNN. We cannot say anything about which is the best model between these two models (RF and CNN) without evaluating them in the same way. So, we retain these two models as our best models. A possible future work would be to combine these models to see if we can reach better performance. One of our goal was to try to reach the level of performance of the ENTSOE (European Network of Transmission System Operators for Electricity), but we came very far from this objective. Using CNN plus ANN (artificial neural network) del Real et al. (2020) got a slightly better result than the reference Réseau de Transport d'Electricité (RTE, French transmission system operator). Furthermore, one could ask, how the ENTSOE does it forecasting, please see: "Demand forecasting methodology" ENTSOE, 2019. A figure of the process is given in appendices.

Before wrapping up this work, it is worth noting some limits of this work.

- One important limit of this work it to have not veritably do a gridsearch over several hyperparameters values, this was due to time constraint, and the walk forward validation that we implement by hand (from scratch) is slow, one should optimize it,
- The way the data are scaled may impact the result, hence, one should try different type of normalization: MaxAbs scaler, Robust scaler, MinMax scaler and evaluates the different models that, even data without transformation

can be tried. For example, Romero et al. (2018) found that it was data without scaling and random forest which allowed to have better prediction for the price of electricity in Spain,

- One could also investigate other possible transformations of the data, for example: the logarithm, deseasonalization and detrending,
- The weigthed average of the different meteorological variables that we used might have impact negatively the results also, maybe use the data in their initial form would have give better result,
- The non-linear effect of the temperature should be correctly modelized to hope having better result. Again, in Romero et al. (2018), their evaluate 6 different configurations for the nonlinear effect of the temperature, we just use the square here,
- Different lagged values could be tried, in this study we just used the 72 previous and the 30 previous days, one could try 1 day up to 30 days to see what seems to work better. All this put together can quickly increase the number of models that would be trained and evaluated, parallelization is therefore welcome,
- Try other metrics, MAPE, MAE for example, to see if the best model choose by the RMSE are still the same,
- Search for other variables that can also explain the national energy demand.
- We have not dealt with them here, but it is normal to use ARIMA or SARIMA models when manipulating time series. Instead, we provide in the notebook forecasting with Facebook Prophet, a well-known package used in time series forecasting. It models time series using additive model on the components of the time series (the trend, the seasonality, the holidays and the error term) like a GAM (generalized additive model).

Conclusion

In our study, we forecast the 24 hours of the electrical energy consumption in Spain in advance using econometric methods including variable selection methods (Least Angular Regression, General to specific modeling or GETS), dimension reduction and shrinkage methods (Regression based on Factorial Analysis of Mixed Data and Ridge). We also used machine learning methods (Support Vector Regression, KNN...) and Deep learning methods (Convolutional Neural Network, Long Short Term Memory) with the objective to do better than the ENTSOE. Our models have not achieved the performance of the ENTSOE (RMSE =378.31MWh) and the multioutput Random forest and the Convolutional Neural Network appears to be our "best" models wih respectively an RMSE of 2124.78 MWh and 2596.30 MWh (these models were evaluated differently, so a direct comparison of them is not possible at this stage). These two models are both multioutput model, so they were able to find relationship between the component of the output and to have better result. Most of the models predict well in night, probabily because the demand is stable, but they get stuck especialy the Sunday, and the reason why this, is not very clear. Furthermore, these poor results highlight a limit of our work because a gridsearch over several hyperparameters values of the models could have led to better results. For better prediction, it could be interesting to find the best non linear relationships that caracterise energy demand and weather variables, it could also be interesting to identify other variables that can help to model the electrical energy consumption such as income (like the GDP, but this could not be got at hourly granularity), unusual periods of intense activity, solar irradiance (as Behm et al., 2020), etc.

Bibliography

1. Á. Romero, J. R. Dorronsoro, J. Díaz. Day-Ahead Price Forecasting for the Spanish Electricity Market, International Journal of Interactive Multimedia and Artificial Intelligence, (2018), http://dx.doi.org/10.9781/ijimai.2018.04.008
2. Alejandro J. del Real, Fernando Dorado, Jaime Durán (2020). Energy Demand Forecasting Using Deep Learning: Applications for the French Grid. Energies 2020, 13, 2242; doi:10.3390/en13092242
3. Ashfaq Ahmad, Nadeem Javaid, Abdul Mateen, Muhammad Awais, Zahoor Ali Khan (2019). Short Term Load Forecasting in Smart Grids: An Intelligent Modular Approach. Energies 2019, 12, 164; doi:10.3390/en12010164
4. Christian Behm, Lars Nolting, Aaron Praktiknjo, 2020, How to model European electricity load profiles using artificial neural networks, Applied Energy, Volume 277, 115564, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.115564.
5. Christoph Bergmeir, Rob J. Hyndman b, Bonsoo Koo (2017). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics and Data Analysis 120 (2018) 70–83.
6. Damian Clarke (2014). General-to-specific modeling in Stata. The Stata Journal (2014) 14, Number 4, pp. 895–908.
7. European Network of Transmission System Operators for electricity. Demand forecasting methodology, 2019.
8. Felix Pretis, J. James Reade, Genaro Sucarrat (2018). Automated General-to-Specific (GETS) Regression
9. Hanen Borchani, Gherardo Varando, Concha Bielza, Pedro Larrañaga (2015). A survey on multioutput regression. WIREs Data Mining Knowl Discov 2015, 5:216–233. doi: 10.1002/widm.1157
10. Hastie T., Tibshirani R., Friedman J., "Elements of statistical learning", Springer, corrected 12th, January 2017.
11. Hu, H.;Wang, L.; Peng, L.; Zeng, Y.-R. Effective energy consumption forecasting using enhanced bagged echo state network. Energy 2020, 197, 1167–1178. https://doi.org/10.1016/j.energy.2019.116778
12. Makridakis S, Spiliotis E, Assimakopoulos V (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS ONE 13(3): e0194889. https://doi.org/10.1371/journal.pone.0194889
13. Maria Kolokotroni, Michael Davies, Ben Croxford, Saiful Bhuiyan, and Anna Mavrogianni. A validated methodology for the prediction of heating and cooling energy demand for buildings within the urban heat island: Case-study of london. Solar Energy, 84(12):2246 – 2255, 2010.
14. Modeling and Indicator Saturation for Outliers and Structural Breaks.Journal of Statistical Software, August 2018, Volume 86, Issue 3. doi: 10.18637/jss.v086.i03
15. Muller, A. C., Guido, S. (2017). Introduction to machine learning with Python: A guide for data scientists.
16. Ricco Rakotomalala (2012). Analyse factorielle des données mixtes – Comparaison des logiciels Tanagra et R.
17. Ricco Rakotomalala (2018). Régression Lasso sous Python. Utilisation du package « scikit-learn ».
18. Samira Garshasbi, Jarek Kurnitski, and Yousef Mohammadi. A hybrid genetic algorithm and monte carlo simulation approach to predict hourly energy consumption and generation by a cluster of net zero energy buildings. Applied Energy, 179:626 – 637, 2016.
19. Sean J. Taylor, Benjamin Letham (2018). Forecasting at Scale. https://doi.org/10.7287/peerj.preprints.3190v2
20. Sildir, Hasan; Aydin, Erdal; Kavzoglu, Taskin. 2020. "Design of Feedforward Neural Networks in the Classification of Hyperspectral Imagery Using Superstructural Optimization" Remote Sens. 12, no. 6: 956. https://doi.org/10.3390/
21. Ümmühan Basaran Filik, Ömer Nezih Gerek, and Mehmet Kurban. A novel modeling approach for hourly forecasting of long-term electric energy demand. Energy Conversion and Management, 52(1):199 – 211, 2011.
22. Vincent Thouvenot, Audrey Pichavant, Yannig Goude, Anestis Antoniadis, and Jean-Michel Poggi (2015). IEEE. doi :10.1109/TPWRS.2015.2504921
23. Yuehui Zhao Peng Wang Jun Zhang Tianyi Liu, Shuangsang Fang. Implementation of training convolutionnal neural networks. University of Chinese Academy of Sciences, 2015.

Web sites

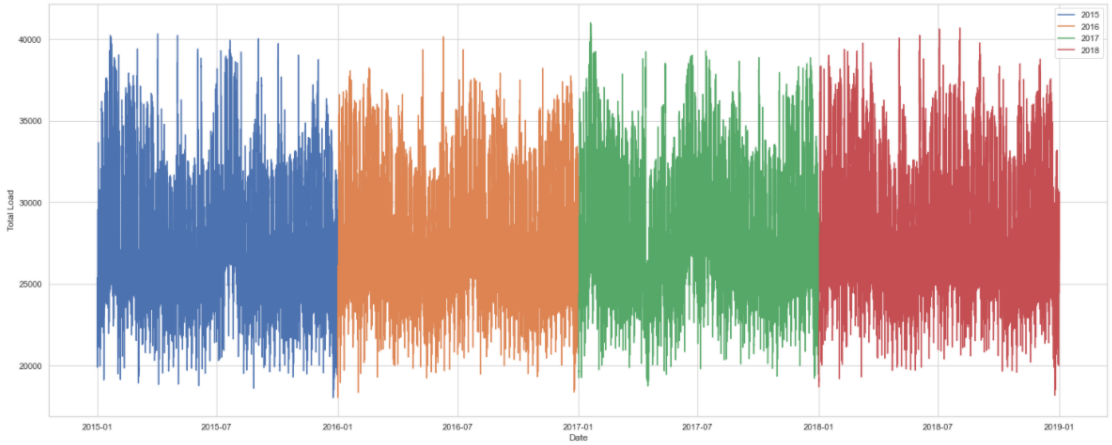
1. https://cran.r-project.org/web/packages/forecastML/vignettes/package_overview.html
2. https://github.com/nicholasjhana/short-term-energy-demand-forecasting
3. https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/
4. https://machinelearningmastery.com/multi-output-regression-models-with-python/
5. https://machinelearningmastery.com/multi-step-time-series-forecasting-with-machine-learning-models-for-household-electricity-consumption/
6. https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/

Appendices

A Data and Methods

1. Exploratory analysis

Figure A.1: Yearly actual load



2. LSTM

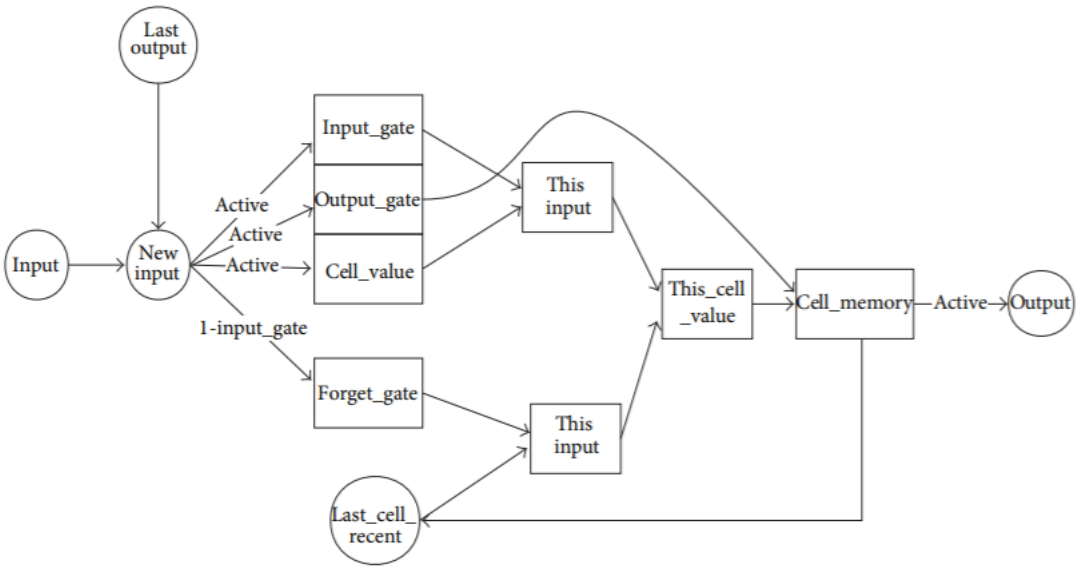
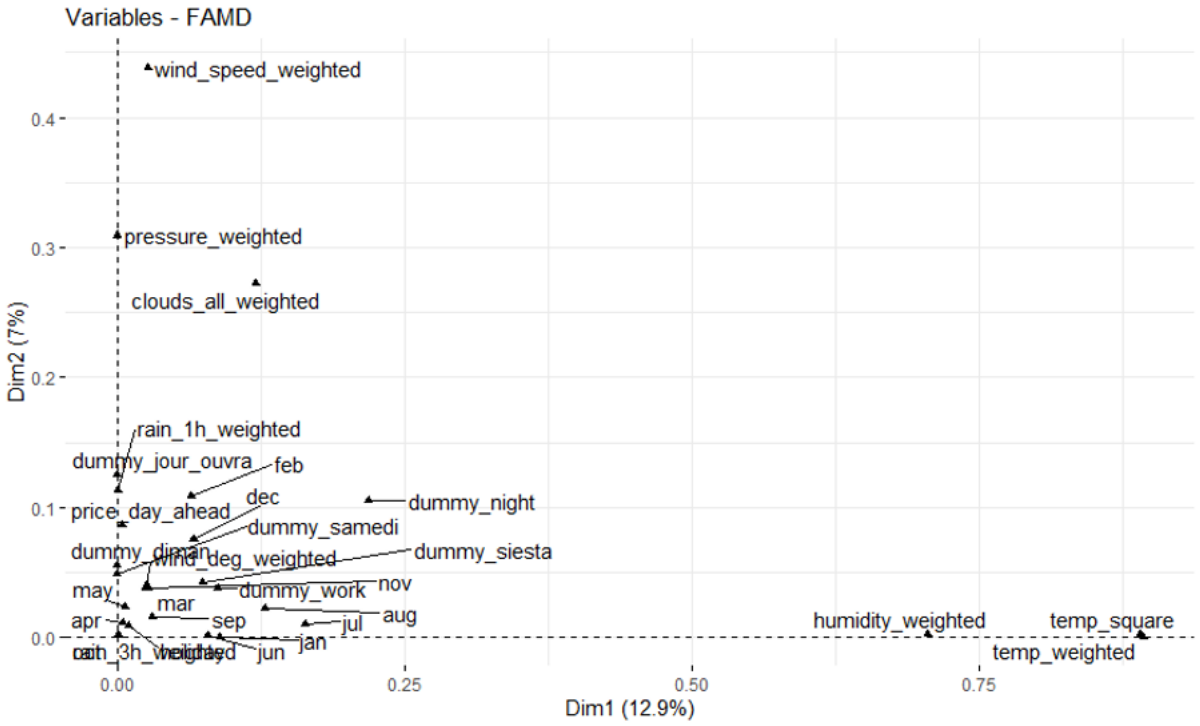


FIGURE 1: The LSTM network structure.

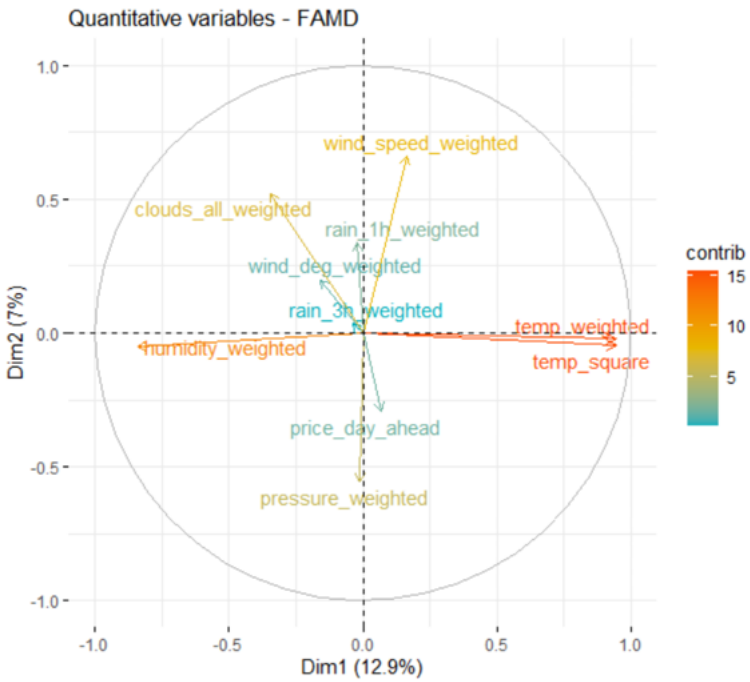
Source:Google images

B Results

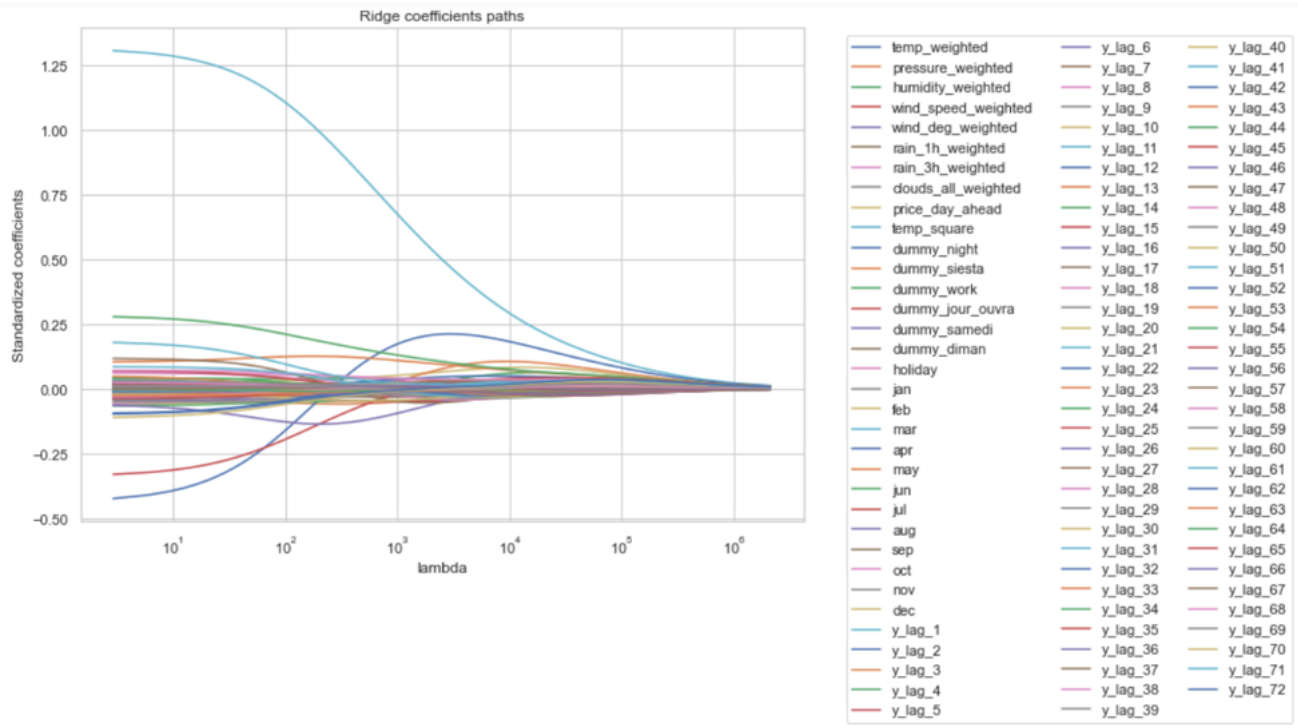
1. FAMD - Graph of variables



2. Correlation circle

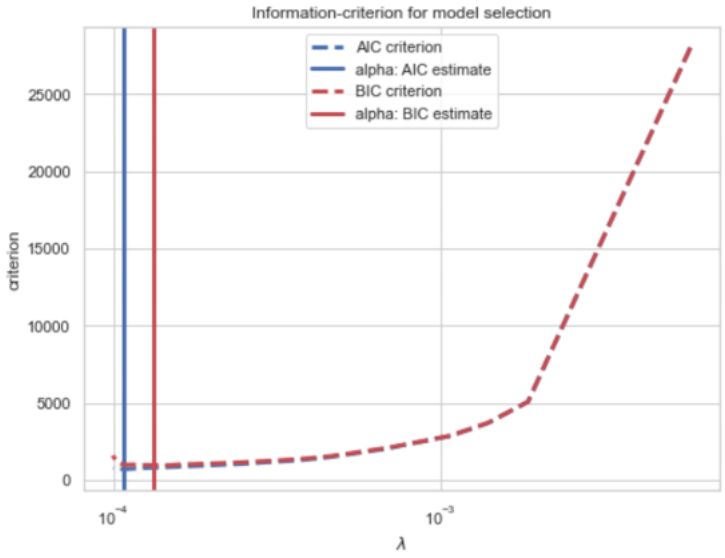


3. Ridge coefficient path



As the lambda increases, the coefficients are shrunk towards 0, but they will not be worth 0.

4. The evolution of the criteria for Lasso LARS



5. GETS
   Model obtained in Stata:

```
Doornik-Hansen test rejects normality of errors in the GUM.
Respecify using nodiagnostic if you wish to continue without specification tests. This option should be used with c
aution.

The test for ARCH components is not rejected.
Respecify using nodiagnostic if you wish to continue without specification tests. This option should be used with c
aution.

The in-sample Chow test rejects equality of coefficients
Respecify using nodiagnostic if you wish to continue without specification tests. This option should be used with c
aution.

The GUM fails 3 of 4 misspecification tests.  Out-of-sample Chow test for equality of coefficients not rejected.  T
his GUM
performs poorly. Care should be taken in interpretation.

Specific Model:
note: dummy_night omitted because of collinearity

Linear regression                               Number of obs     =     28,008
                                                F(58, 27949)      =  46574.42
                                                Prob > F          =    0.0000
                                                R-squared         =    0.9781
                                                Root MSE          =    .14817
```

total_load_actual	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf.	Interval]
temp_weighted	-.0159584	.0044478	-3.59	0.000	-.0246762	-.0072405
humidity_weighted	-.0032703	.0015407	-2.12	0.034	-.0062902	-.0002504
clouds_all_weighted	.0026669	.0011298	2.36	0.018	.0004524	.0048813
price_day_ahead	.0237938	.001448	16.43	0.000	.0209557	.0266319
temp_square	.015985	.0041161	3.88	0.000	.0079172	.0240528
dummy_night	0	(omitted)				
dummy_siesta	-.0285916	.0053197	-5.37	0.000	-.0390185	-.0181646
dummy_work	.0529554	.0040039	13.23	0.000	.0451076	.0608032
dummy_jour_ouvra	.0344537	.0025436	13.55	0.000	.0294681	.0394392
jan	.0137351	.0036081	3.81	0.000	.0066629	.0208072
feb	.0182178	.0037374	4.87	0.000	.0108922	.0255433
mar	.0104692	.003849	2.72	0.007	.0029249	.0180134
jun	.0081839	.0035769	2.29	0.022	.0011731	.0151947
jul	.0172421	.0039458	4.37	0.000	.0095082	.0249761
oct	-.0119902	.0033917	-3.54	0.000	-.0186382	-.0053423
y_lag_1	1.315828	.0282376	46.60	0.000	1.26048	1.371175
y_lag_2	-.4394911	.0410594	-10.70	0.000	-.5199694	-.3590127
y_lag_3	.0480178	.020865	2.30	0.021	.0071214	.0889142
y_lag_4	.0409088	.0098592	4.15	0.000	.0215843	.0602333
y_lag_5	-.0329014	.0082245	-4.00	0.000	-.0490218	-.016781
y_lag_6	-.0172712	.0074809	-2.31	0.021	-.0319342	-.0026082
y_lag_8	.0485807	.0044374	10.95	0.000	.0398832	.0572783
y_lag_14	-.0906862	.0035019	-25.90	0.000	-.0975501	-.0838224
y_lag_18	.0644776	.0041715	15.46	0.000	.0563013	.0726538
y_lag_20	-.0531649	.0069462	-7.65	0.000	-.0667798	-.03955
y_lag_21	.023788	.0071962	3.31	0.001	.0096832	.0378929
y_lag_23	.0911802	.0126549	7.21	0.000	.066376	.1159843
y_lag_24	.2862402	.0270241	10.59	0.000	.2332717	.3392088
y_lag_25	-.3388417	.0306082	-11.07	0.000	-.3988353	-.2788481
y_lag_26	-.0652059	.0219493	-2.97	0.003	-.1082276	-.0221841
y_lag_27	.1231558	.0119221	10.33	0.000	.0997878	.1465237
y_lag_28	-.0599841	.01045	-5.74	0.000	-.0804665	-.0395016
y_lag_29	-.032539	.0100316	-3.24	0.001	-.0522015	-.0128766
y_lag_30	.0179853	.0078519	2.29	0.022	.0025953	.0333754
y_lag_32	-.0324311	.0053661	-6.04	0.000	-.0429489	-.0219134
y_lag_34	.027881	.0038725	7.20	0.000	.0202907	.0354714
y_lag_38	.0216797	.0044448	4.88	0.000	.0129677	.0303917
y_lag_40	-.0179035	.0061395	-2.92	0.004	-.0299372	-.0058698
y_lag_41	.023085	.0059717	3.87	0.000	.0113801	.0347899
y_lag_44	-.034137	.0073326	-4.66	0.000	-.0485093	-.0197646
y_lag_45	.0746208	.0109543	6.81	0.000	.0531499	.0960917
y_lag_46	-.0510129	.013038	-3.91	0.000	-.0765679	-.0254579
y_lag_47	-.0386089	.0135592	-2.85	0.004	-.0651855	-.0120323
y_lag_48	.0759747	.0178484	4.26	0.000	.040991	.1109585
y_lag_49	-.0654407	.0142461	-4.59	0.000	-.0933638	-.0375176
y_lag_51	.0736826	.0093173	7.91	0.000	.0554202	.0919449
y_lag_52	-.0995948	.0111033	-8.97	0.000	-.1213578	-.0778318
y_lag_53	.0617053	.0082167	7.51	0.000	.0456001	.0778104
y_lag_55	-.03394	.0047986	-7.07	0.000	-.0433454	-.0245346
y_lag_57	.0186339	.0068297	2.73	0.006	.0052473	.0320205
y_lag_58	-.033983	.0089132	-3.81	0.000	-.0514534	-.0165127
y_lag_59	.0353908	.0060527	5.85	0.000	.0235273	.0472544
y_lag_62	.0162345	.0059694	2.72	0.007	.0045341	.0279349
y_lag_63	-.0301421	.00602	-5.01	0.000	-.0419415	-.0183426
y_lag_65	.0146999	.0042025	3.50	0.000	.0064629	.022937
y_lag_68	.0294225	.0046497	6.33	0.000	.0203088	.0385361
y_lag_70	-.1063732	.010256	-10.37	0.000	-.1264754	-.086271
y_lag_71	.1818279	.0141596	12.84	0.000	.1540743	.2095814
y_lag_72	-.1005726	.0076839	-13.09	0.000	-.1156334	-.0855118
_cons	-.046111	.0031385	-14.69	0.000	-.0522626	-.0399595

Memory error encountered for GETS in R

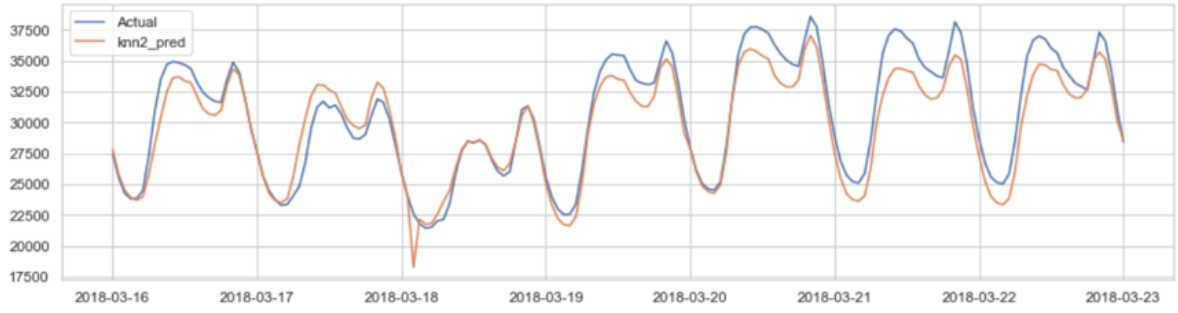
```
> library(zoo)
> library(gets)
> train_r <- read.csv("C:/Users/jeane/3as/train_r.csv")
> train_r = zoo(train_r, order.by = train_r[, "time"])
> (sis <- isat(train_r[,1], t.pval = 0.05,mxreg = train_r[,2:103],vcov.type= "white",arch.LjungB=list(lag = 1))) ##estimate
 an AR(72) including the remaining regressor and the SIS (step-indicator saturation), test arch in the variance
Erreur : impossible d'allouer un vecteur de taille 5.8 Go
> (sis <- isat(train_r[,1], t.pval = 0.05,ar = c(1:72),mxreg = train_r[,2:30],vcov.type= "white",arch.LjungB=list(lag = 1)))
 ##estimate an AR(72) including the remaining regressor and the SIS (step-indicator saturation), test arch in the variance
Erreur : impossible d'allouer un vecteur de taille 5.8 Go
> ##Note that in the first sis above, we normally already have the AR(72)
```

The *isat* function undertakes multi-path indicator saturation to detect outliers and mean-shifts using impulses (IIS), step-shifts (SIS), or trend-indicators (TIS). Indicators are partitioned into blocks and selected over at a chosen level of significance (t.pval) using the getsm function. The getsm function is for multi-path General-to-Specific (GETS) Modelling of an AR-X model (the meanspecification) with log-ARCH-X errors (the log-variance specification),whereas getsv does the same for the log-variance specification. See https://cran.r-project.org/web/packages/gets/gets.pdf and also Automated General-to-Specific (GETS) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks : Pretis, Reade and Sucarrat (2018) doi:10.18637/jss.v086.i03 or https://www.jstatsoft.org/article/view/v086i03

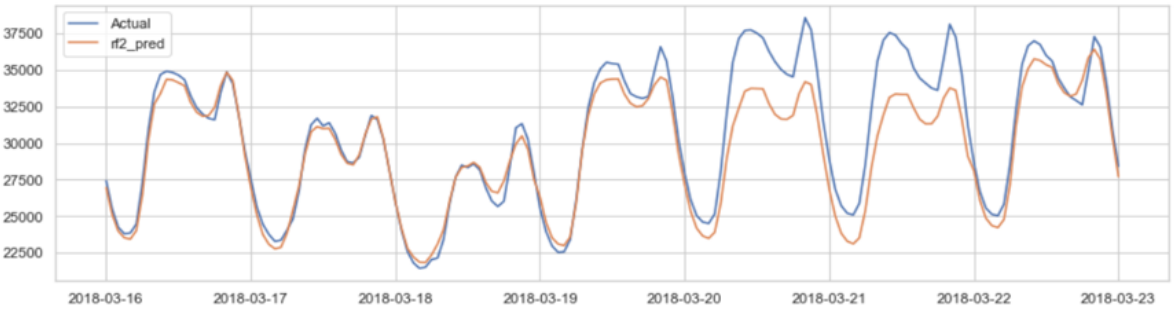
6. RMSE per hour

Hour	Lin_reg	Ridge	Reg_on_FAMD	LASSO	KNN	RF	XGBOOST	KNN2	RF2	MLP	CNN	LSTM
0	875,49	869,32	875,37	725,93	1042,67	654,48	679,12	1198,27	1351,40	1424,16	1434,76	1494,92
1	1013,27	1005,63	1006,73	825,91	1006,39	775,29	795,98	1069,41	1206,63	1533,66	1419,22	1304,35
2	1023,37	1018,17	1000,27	873,69	993,49	880,33	894,24	1934,84	1971,38	1624,42	1351,91	1306,09
3	1103,52	1099,42	1060,00	1000,33	1031,43	1031,02	1048,06	988,29	1099,72	1664,50	1398,16	1369,02
4	1240,98	1238,54	1166,11	1287,77	1187,28	1334,98	1373,89	1007,12	1089,59	1702,38	1331,32	1474,14
5	1727,40	1726,85	1662,73	1819,46	1624,64	1994,57	2048,65	1115,71	1138,71	1905,95	1814,64	1994,65
6	2771,68	2772,40	2741,51	2873,99	2467,36	2961,69	2889,13	1691,00	1517,10	2774,59	2838,91	3014,23
7	3440,92	3441,79	3444,46	3582,78	2992,42	3558,33	3412,51	2761,08	2331,83	3387,05	3368,68	3663,57
8	3431,29	3432,18	3444,74	3603,63	3000,38	3535,27	3347,24	3236,11	2776,10	3561,79	3474,49	3754,60
9	3094,49	3095,54	3092,97	3258,83	2814,83	3146,96	3029,00	2984,43	2694,38	3392,12	3270,53	3479,71
10	2834,59	2835,47	2825,50	2975,15	2613,67	2876,12	2772,49	2683,89	2525,47	3250,22	3106,15	3276,64
11	2835,94	2836,31	2809,96	2939,33	2647,60	2785,44	2760,44	2544,18	2375,20	3203,21	3074,89	3269,60
12	2661,71	2662,47	2645,15	2773,75	2516,77	2695,44	2614,24	2622,26	2408,80	3167,51	2958,11	3329,87
13	2507,10	2508,13	2467,37	2630,30	2386,65	2511,08	2477,10	2483,24	2284,76	3115,84	2831,56	3222,05
14	2604,15	2605,67	2556,46	2732,02	2493,88	2602,69	2587,48	2338,52	2175,27	3062,28	2917,22	3280,83
15	2793,81	2795,02	2768,68	2915,43	2678,16	2835,26	2772,87	2506,11	2301,60	3135,72	3021,11	3324,09
16	2878,41	2879,44	2862,58	2998,42	2747,02	2898,11	2867,86	2641,93	2414,93	3128,52	3132,44	3475,90
17	2891,67	2892,30	2857,35	3026,31	2806,83	2871,84	2903,28	2688,09	2454,75	3125,96	3075,95	3418,05
18	2856,35	2856,80	2825,46	2944,41	2854,41	2879,05	2806,59	2697,07	2580,25	3191,96	3073,86	3358,06
19	2649,61	2649,74	2614,06	2644,10	2742,73	2702,11	2529,73	2788,92	2672,42	3033,95	2864,18	3087,55
20	2307,33	2306,80	2300,35	2278,70	2337,73	2268,75	2080,87	2619,34	2460,16	2710,53	2441,73	2700,80
21	2163,72	2163,99	2104,79	2169,21	1988,60	1787,29	1760,83	2161,38	2073,67	2173,93	2108,69	2301,17
22	2121,57	2122,48	1971,35	2168,12	1681,89	1507,76	1559,11	1759,61	1830,42	1709,29	1764,72	2035,61
23	2093,93	2095,71	1916,48	2224,52	1637,22	1566,47	1595,60	1420,42	1534,40	1569,65	1650,83	1846,02

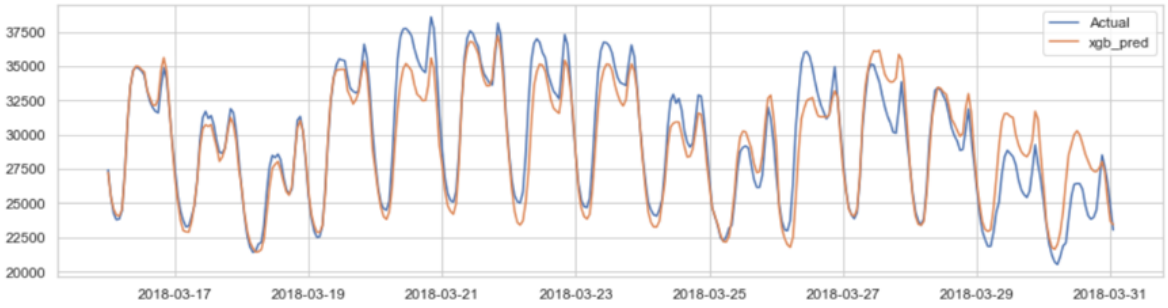
Forecasting of the "best" KNN for the first week on the test set (ordinate: total load actual)



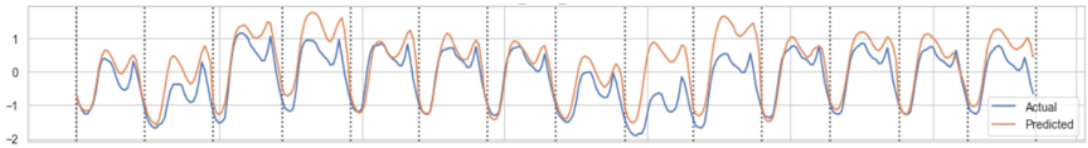
Forecasting of the "best" RF for the first week on the test set (ordinate: total load actual)



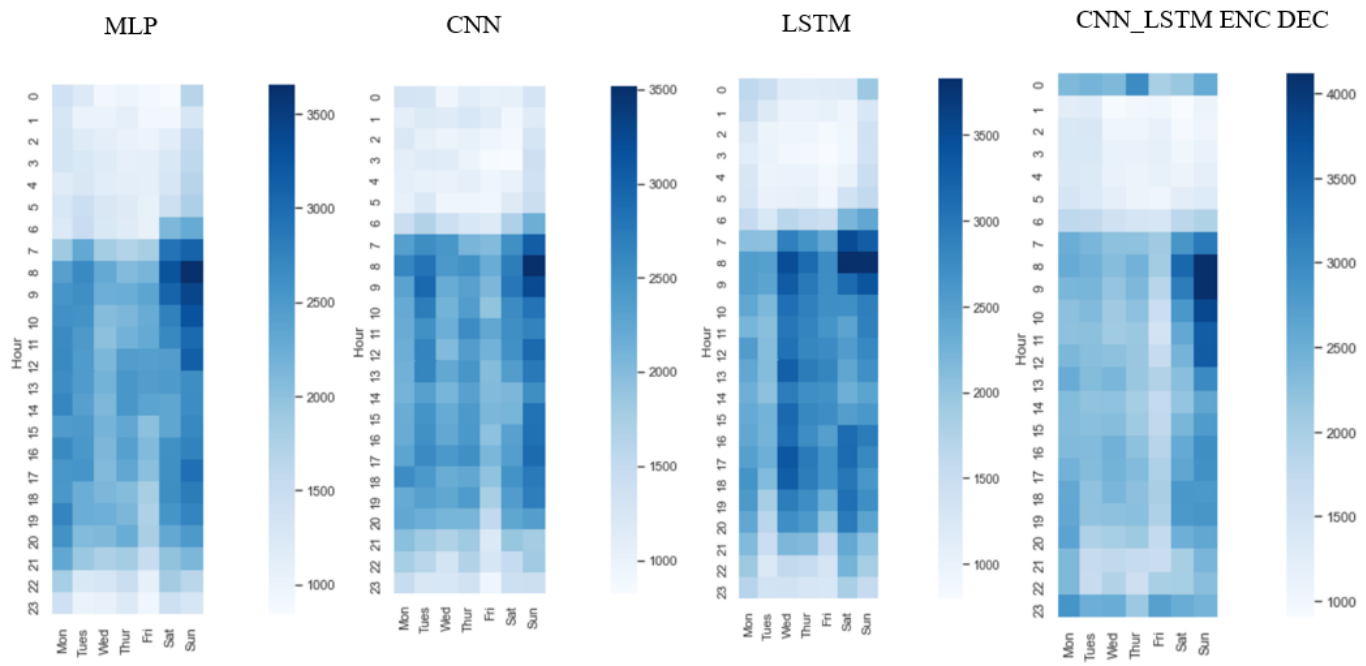
Forecasting of the "best" XGBOOST for the first week on the test set (ordinate: total load actual)



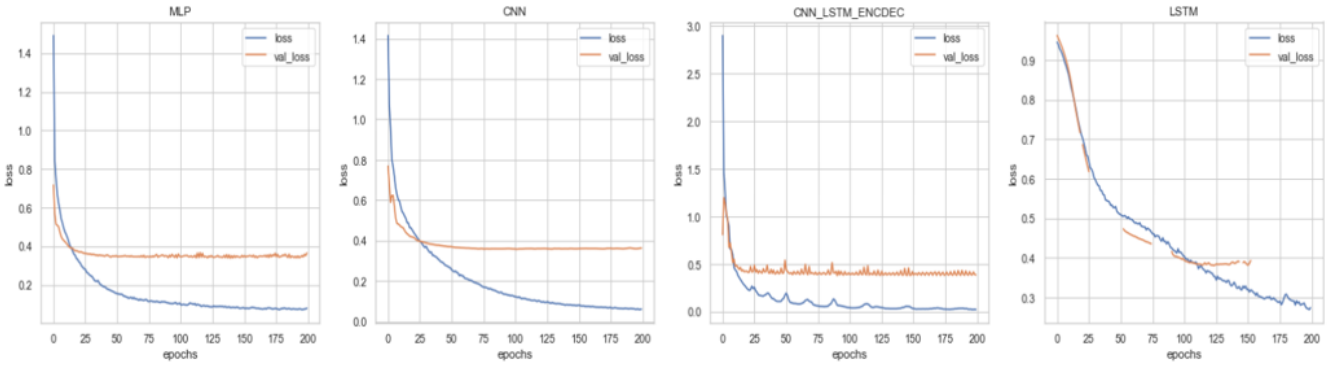
CNN-LSTM encoder-decoder Predictions of the First Two Weeks of test set after lagging: 14/04/2018 – 27/04/2018. Ordinate : total load actual standardized; abscissa : cumulative hours



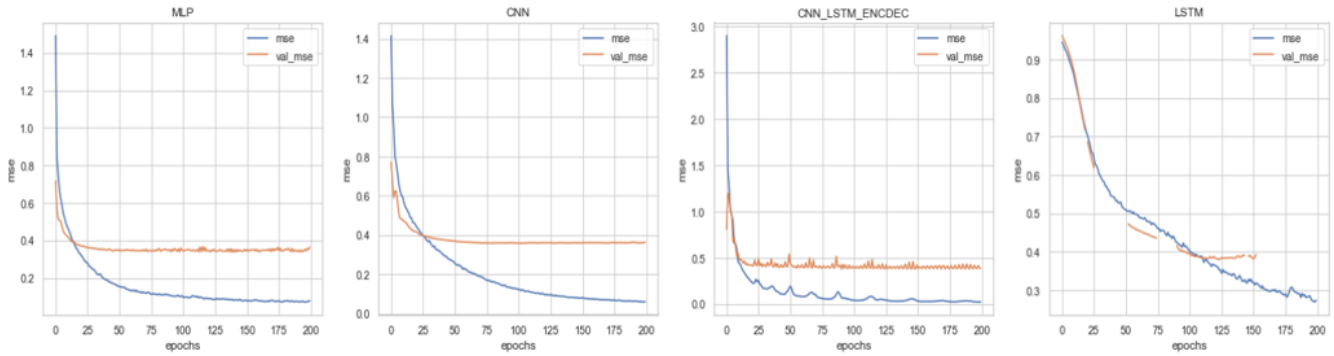
8. RMSE per day of week and per hour for deep learning models



9. Losses Curves

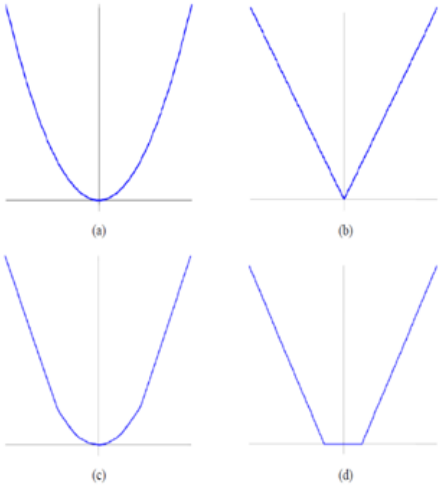


10. MSE Curves



11. Different loss functions mentioned in this work

Figure 2 Loss functions, (a) quadratic (b) Laplace (c) Huber (d) ε-intensive (see online version for colours)



- Squared error loss function:

$$L = \frac{1}{2}(y_i - m(x_i))^2$$

for which the gradient vector is the residuals $r_i = y_i - m(x_i)$

- Absolute error loss function, or Laplacian:[30]

$$L = |y_i - m(x_i)|$$

→ median of the conditional distribution . . . robust regression

- Huber loss function: a robust alternative to absolute error loss,

$$L = \begin{cases} \frac{1}{2}(y_i - m(x_i))^2 & |y_i - m(x_i)| \leq \delta \\ \delta(|y_i - m(x_i)| - \delta/2) & |y_i - m(x_i)| > \delta \end{cases}$$

Emmanuel Flachaire (2020). Econometrics & Machine Learning. course materials

$$\varepsilon - \text{intensive/insensitive: } L = \max\{0, |y_i - m(x_i)| - \varepsilon$$

Azadeh, A., Boskabadi, A.
and Pashapour, S. (2015) 'A unique support vector regression for improved modelling and forecasting of short-term gasoline consumption in railway systems', *Int. J. Services and Operations Management*, Vol. 21, No. 2,
pp.217–237.

12. ENTSOE forecasting methodology, (2019)