



UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in Ingegneria Elettronica

**Previsione di serie temporali sul consumo di energia elettrica e utilizzo delle
relative predizioni in un modello di simulazione**

Electric load time series forecasting and relative predictions on simulation model

Relatore: Chiar.mo

Prof. Stefano Squartini

Tesi di Laurea di:

Francesco Romagnoli

Correlatore:

Dott. Sebastian Schmitt

A.A. 2019 / 2020

CONTENTS

INTRODUCTION.....	4
Part 1: REGRESSION ALGORITHMS AND SIMULATION MODEL	6
1.1 Machine Learning Algorithms.....	6
1.2 Linear Models	7
1.3 Ensemble Models.....	10
1.4 Recurrent Neural Network	13
1.5 MPC Simulation Model	20
Part 2: PROJECT DESCRIPTION	22
- Analysis of time series z35	24
2.1 RNN Models	25
2.1.1 RNN – Test n. 1.....	26
- Structure and description of RNN model.....	29
2.1.2 RNN – Test n.2	40
2.1.3 RNN – Final Test.....	49
2.2 Linear Regression Models.....	117
2.2.1 Linear Regression Models – Preliminary Test.....	118
2.2.2 Linear Regression Models – Final Test.....	138
2.3 Ensemble Regression Models	199
2.3.1 Ensemble Regression Models – Preliminary Test.....	202
2.3.2 Ensemble Regression Models – Final Test.....	231
2.4 Multioutput Regression for Linear and Ensemble Models.....	242
2.5 Conclusions on Best Models	256
2.6 Generalization of Best Models.....	259

2.7 EMS MPC Simulation Model	271
CONCLUSION.....	294
APPENDIX: RIASSUNTO PROGETTO	296
BIBLIOGRAPHY	303

INTRODUCTION

Honda Research Institute Europe GmbH (HRI-EU) was founded in Europe in 2003 and the central focus of the institute is research into Computational Intelligence, Optimization and Robotics.

During my traineeship at HRI-EU, my research was focus in the field of time series forecasting applied to electric load case.

Time series forecasting is an important area of machine learning because there are so many prediction problems that involve a time component. A normal machine learning dataset is a collection of observations, while a time series dataset adds an explicit order dependence between observations, represented by time dimension. This additional dimension is both a constraint and a structure that provides a source of additional information.

In this topic, electric load time series forecasting represent a crucial task in the next future. The progressive replacement of fossil fuels in a wide energy demand case with the using of electricity, like cooling or heating of buildings and transport with the spreading of electric vehicles, make electric power demand predicions fundamental in order to develop a smart grid structure at any level, from electric industry to facilities and private houses. Electricity load forecasting allows to make this exponentially growing of using of electricity energy sustainable, with smart management of the energy resources to cover the electric demand and savings costs.

The objective of the research is to investigate different time series forecasting algorithms applied to electric consumption of a facility, including the estimation of some confidence bounds, such as minimal and maximal load values as well as the expected load standard deviation.

The predictions of the most promising time series algorithms are applied to a simulation model of an Energy Management System (EMS) based on model-predictive-control (MPC) method in an optimization scenario with the objective of reducing energy costs while maintaining optimal air-condition temperature.

The simulation framework used for testing the predictions of the models realized was provide by a PhD student at HRI-EU, Thomas Schmitt, that I want to thank for kindly allowing me to use it.

Part 1: REGRESSION ALGORITHMS AND SIMULATION MODEL

1.1 Machine Learning Algorithms

Machine learning is the study of computer algorithms that improve automatically through experience. Machine learning algorithms build a model on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. The approaches of machine learning techniques are traditionally divided into three broad categories:

- Supervised Learning: training data are composed by example inputs and their desired outputs, provided by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised Learning: no labels or desired outputs are given to the machine learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: the algorithm interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the algorithm is provided feedback that's analogous to rewards, which it tries to maximize.

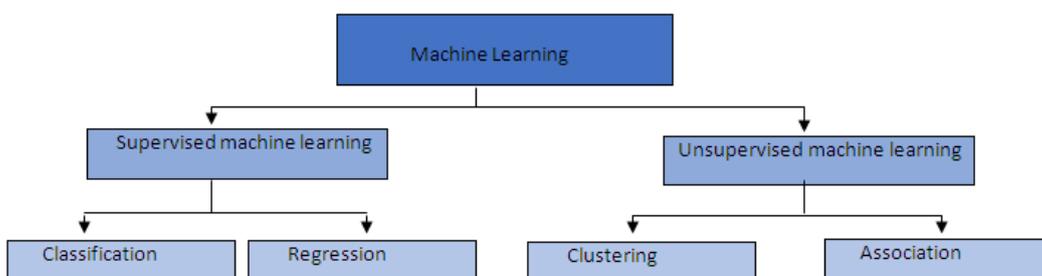


Figure 1. Machine learning algorithms with corresponding tasks

Supervised learning problems can be further grouped into Classification and Regression problems. The main difference between them is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).

In machine learning, classification algorithms attempt to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).

In this case, y is a category that the mapping function predicts. If provided with a single or several input variables, a classification model will attempt to predict the value of a single or several conclusions.

On the other hand, regression algorithms attempt to estimate the mapping function (f) from the input variables (x) to numerical or continuous output variables (y).

In this case, y is a real value, which can be an integer or a floating point value. Therefore, regression prediction problems are usually quantities or sizes.

Between the most common regression algorithms, there are Linear models, Ensemble models and Recurrent Neural Network.

1.2 Linear Models

Linear regression algorithms attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

Linear Regression: Single Variable

$$\hat{y} = \beta_0 + \beta_1 x + \epsilon$$

Linear Regression: Multiple Variables

$$\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p + \epsilon$$

Equation 1. Linear Regression Univariate and Multivariate case

To estimate the values of the coefficients we can use Ordinary Least Squares (OLS). OLS chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function of the independent variable.

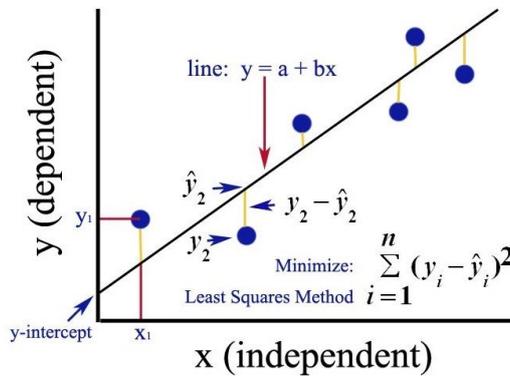


Figure 2. Ordinary Least Squares

We can use another process for optimizing the values of the coefficients by iteratively minimizing the error of the model on training data. This process is called as Gradient Descent. It works by starting with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients

are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

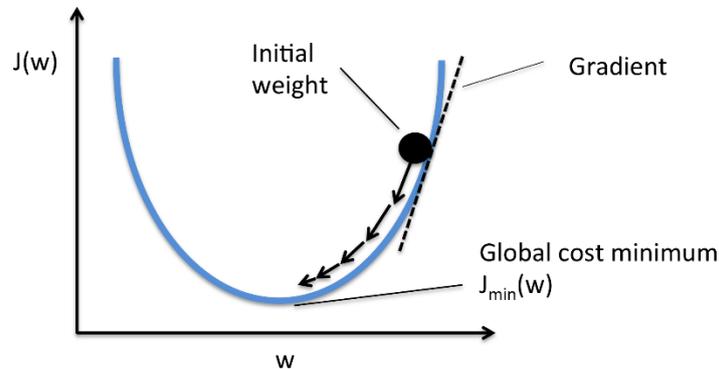


Figure 3. Gradient Descent for Linear Regression

The most important assumption of a linear regression model is that the errors are independent and normally distributed.

In fact every regression model inherently has some degree of error since we can never have prediction 100% accurately. More importantly, randomness and unpredictability are always a part of the regression model. Hence, a regression model can be explained as:

$$\text{Response} = \text{Deterministic} + \text{Stochastic}$$

The deterministic part of the model is what we try to capture using the regression model. Ideally, our linear equation model should accurately capture the predictive information. Essentially, what this means is that if we capture all of the predictive information, all that is left behind (residuals) should be completely random and unpredictable, i.e stochastic. Hence, we want our residuals to follow a normal distribution. And that is exactly what we look for in a residual plot. A few characteristics of a good residual plot are as follows:

- It has a high density of points close to the origin and a low density of points away from the origin
- It is symmetric about the origin

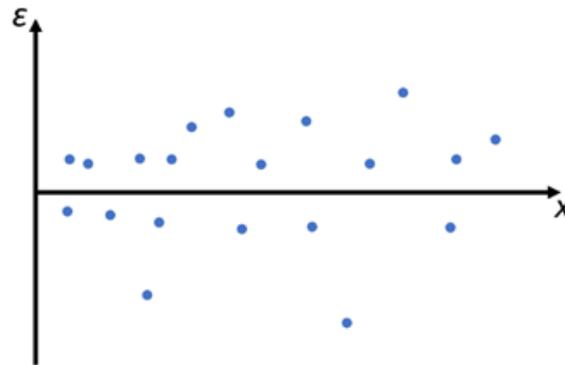


Figure 4. Example of good residual plot

1.3 Ensemble Models

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model.

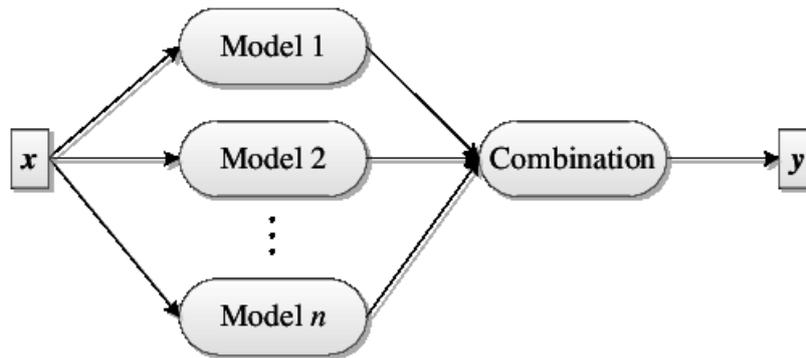


Figure 5. Ensemble model

The two principal methods for combining base models in an ensemble model are:

- **Bagging:** In this case we fit the different base models independently from each others, in parallel, so it is possible to train them concurrently, then they combine following some kind of deterministic averaging process.
- **Boosting:** in this case the different combined base models are no longer fitted independently from each others. The idea is to fit models iteratively such that the training of model at a given step depends on the models fitted at the previous steps, in sequential way.

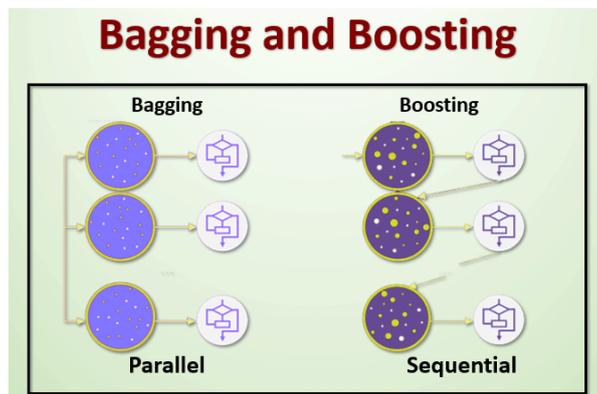


Figure 6. Bagging and Boosting techniques

In general, we can say that bagging will mainly focus at getting an ensemble model with less variance than its components whereas boosting will mainly try to produce strong models less biased than their components (even if variance can also be reduced).

The objective of these models is to find the best tradeoff between bias and variance. A model with high variance will be overfitted, so it will have low quality predictions on unseen data because is too much fitted on training data, whereas a model with high bias will have low quality predictions because it didn't learn to associate well input and output.

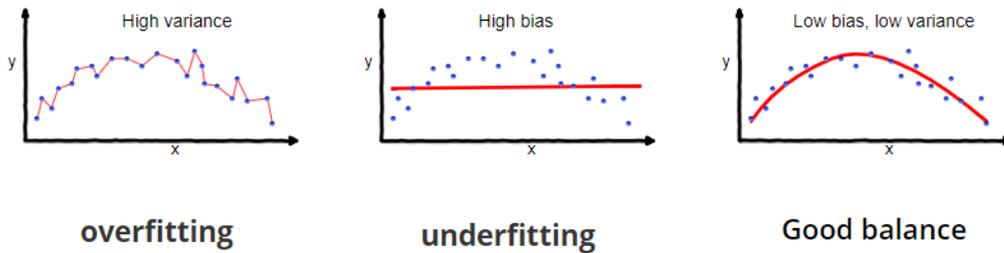


Figure 7. Overfitting and Underfitting

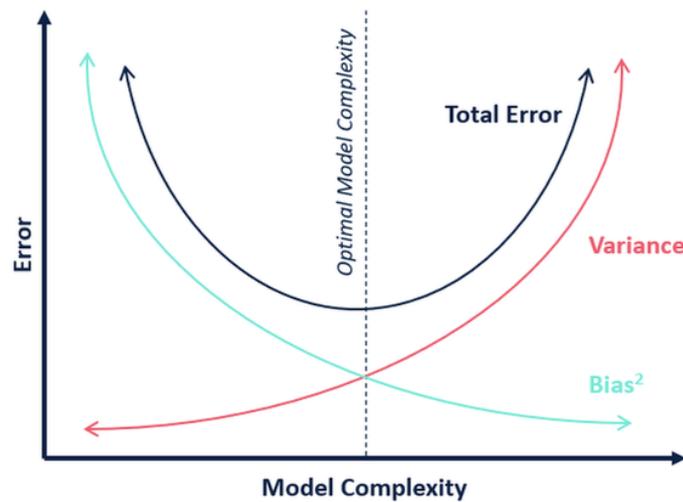


Figure 8. Bias-Variance Tradeoff

1.4 Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks (ANN) where connections between nodes form a directed graph along a temporal sequence. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

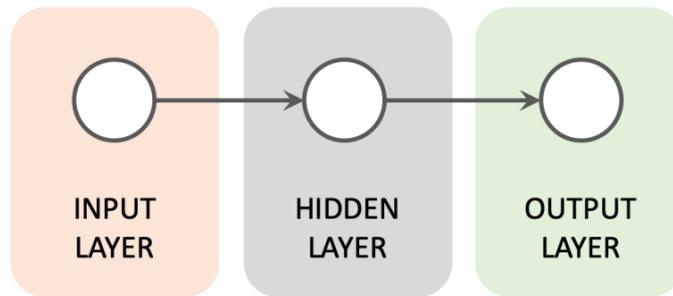


Figure 9. Neural Network - FeedForward Architecture

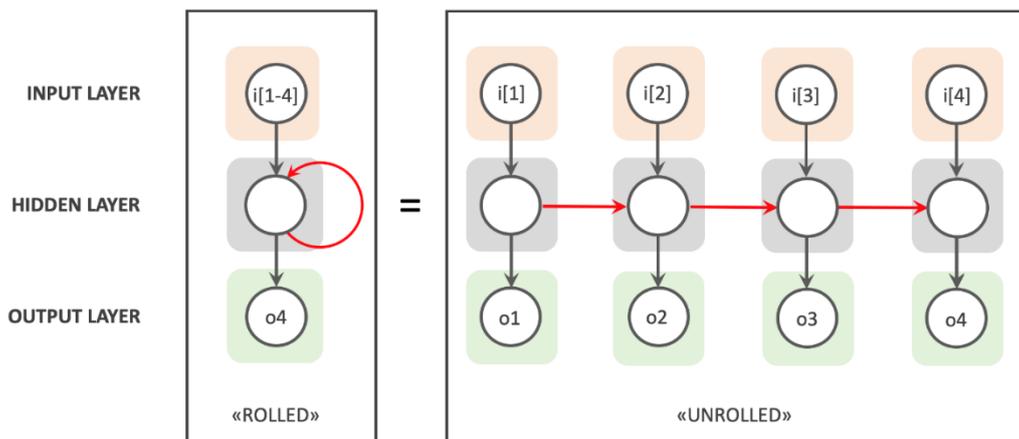


Figure 10. Neural Network – Recurrent Architecture

Unlike Feed Forward Neural Networks, RNNs support processing of sequential data by the addition of a loop. This loop allows the network to step through sequential input data while persisting the state of nodes in the Hidden Layer between steps (we have a sort of working memory).

As we can see in the unrolled representation of RNN of figure 10, RNNs loop is presented as a chain of identical Feed Forward ANNs. These ANNs are identical,

sharing the same structure, weights and activation functions. The red arrows show how the working memory for each iteration is passed to the next.

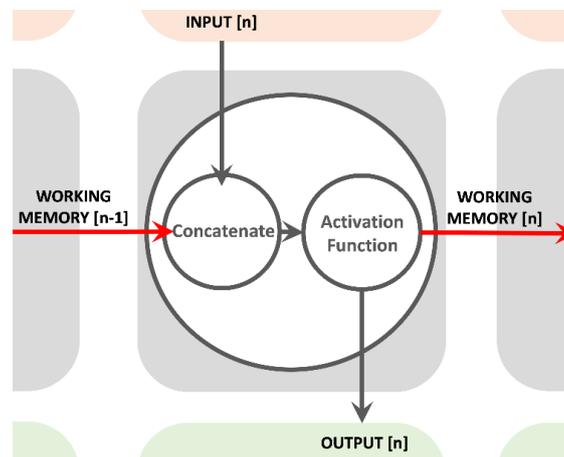


Figure 11. Internal Operations in a Unit of RNN

In figure 11 we show how works a hidden node of RNN. It concatenates it's current input and working memory from the previous iteration, before passing the result on to an activation function. The output from the activation function is then both sent to the output layer and forwarded on to the next iteration of the RNN, as the working memory of the node.

One of the major limitation of standard RNNs is that the working memory struggles to retain longer term dependancies. In fact the working memory of RNNs can forget information from early in sequence. This behaviour is due to the Vanishing Gradient problem, and can cause problems when early parts of the input sequence contain important information. The Vanishing Gradient problem is an issue with back-propagation and Gradient Descent, in particular for RNN it is the width of the unrolled RNN that causes problems. Back-propagation over multiple time steps results in ever diminishing error gradients making both Gradient Descent and weight optimisation difficult.

A variant of RNNs that solve the Vanishing Gradient problem is Long Short Term Memory Network (LSTM). As standard RNN, LSTM loops through sequences of data, persisting and aggregating the working memory over multiple iterations, but it also adds to the network some extra component than standard RNN.

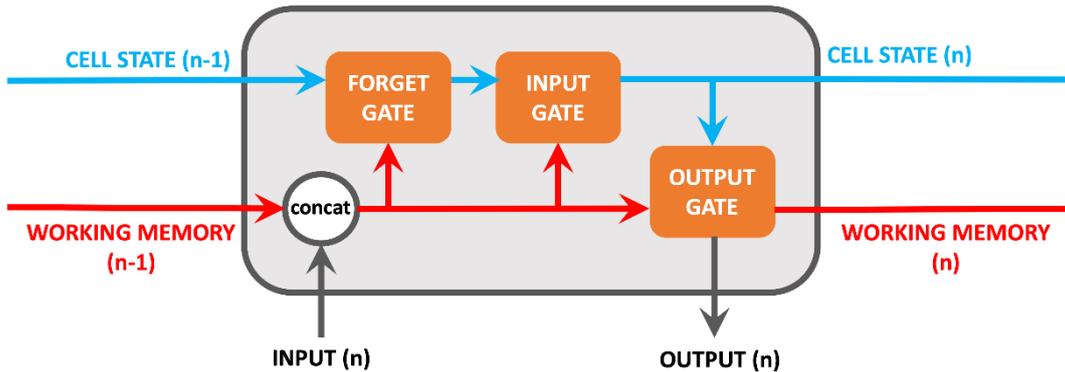


Figure 12. Internal Operations in a Unit of LSTM

Considering figure 12, input and working memory work as standard RNN. The new component added include:

- Cell State: its role is to act as a long-term memory, persisting information (if required) over all iterations of the node. The Cell State can be modified to remove unnecessary information or to keep important information. As such it ensures that important information from early iterations will not be lost over long sequences.
- Forget Gate: The role of the Forget Gate is to decide what information should be removed from the Cell State. This is done by performing calculations on the concatenated working memory-input value and then applying these to the Cell State.
- Input Gate: The Input Gate decides what information should be added to the Cell State, always based on the concatenated working memory-input value.

- Output Gate: The Output Gate decides on what working memory this node will output, based on calculations on the current Cell State and the concatenated working memory-input value.

The output from the node include: the Cell State, so the long-term memory from current iteration, the working memory from current iteration and the output value which will be used for current iterations prediction. Output value is the same as working memory.

We can describe the learning process of a Neural Network

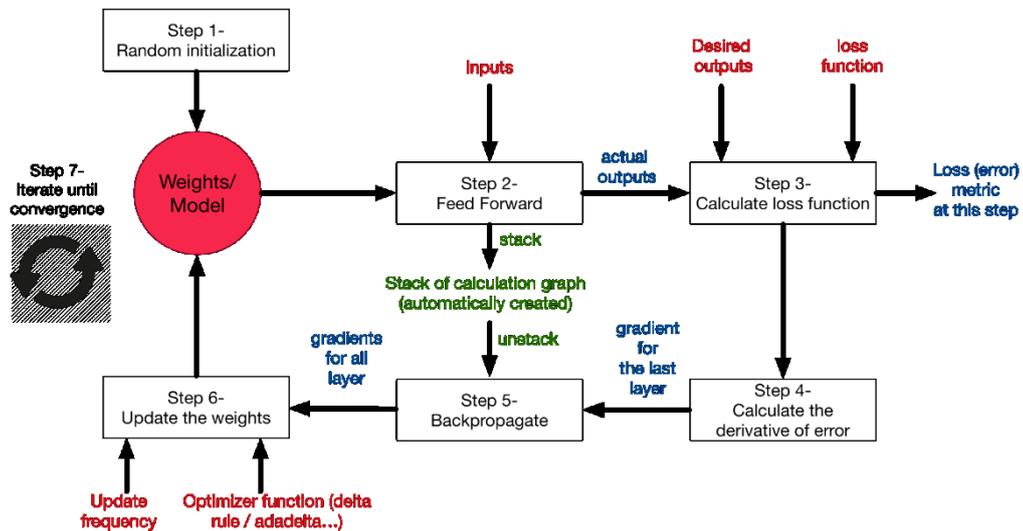


Figure 13. Learning process of a Neural Network step-by-step

First step consist in the random initialization of the model, so we assigned randomly a value for each weights of the model. In second step we check the model performance, so we start from the input, pass it through the network layers and calculate the actual output of the model; this step is called forward-propagation, because the calculation flow is going in the natural forward direction from the input to output. In third step, we have the actual output of our randomly initialized neural network and, on the other hand, we have the

desired output we would like the network to learn, so we have to calculate the loss function that measures the difference between predicted output and desired output. Loss function can be convex or non-convex:

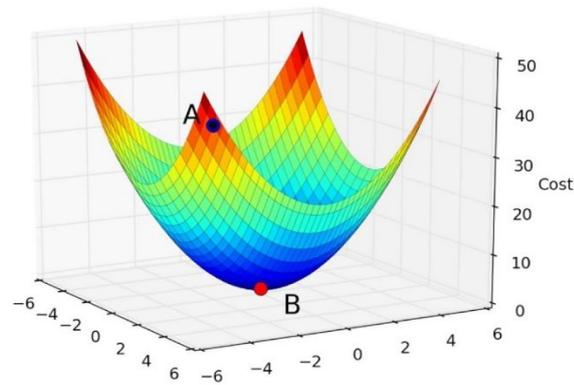


Figure 14. Convex Loss Function

Convex Loss function are characterized by a single global minimum, so, considering figure 14, at first iteration of the model we are at point A and at the end of learning process we want to reach point B that represent the global minimum of the Loss function.

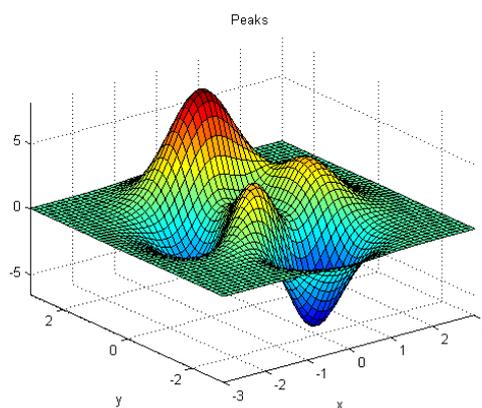


Figure 15. Non-Convex Loss function

Non-Convex Loss function are characterized by one or more local minimum and a global minimum. The considerations made for Convex Loss function are valid, but the problem to use Non-Convex Loss function is that it can reach local minimum instead of global minimum at the end of learning process.

In fourth step, to minimize the Loss function we can use any optimization technique that modifies the internal weights of neural networks. Differentiation is the process used to optimize the weights, it deals with the gradient of the Loss function.

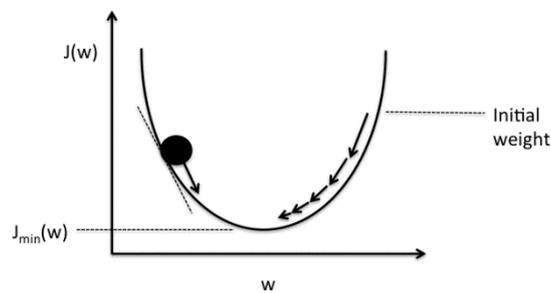


Figure 16. Schematic of Gradient Descent

Now we can describe fifth step of learning process. Usually Neural Networks are composed by more than one layer between input and output, in order to reach more variations in the functionality of the neural network, so the output produced will be a composition on function applied to the input

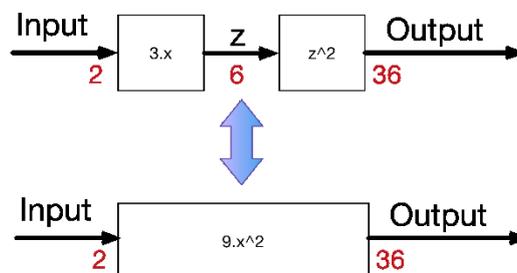


Figure 17. Composition of function used in input to produce the output

As we know, derivative is decomposable, so, starting from Loss function, if we know how to calculate its derivative and the derivative of each function of the composition, we can back propagate the error from the end to the start. The constraint to use back-propagation of errors is that activation functions used in the layers of the model are differentiable.

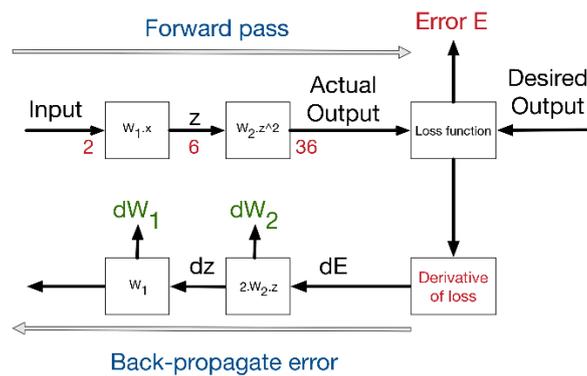


Figure 18. Diagram of Forward and Backward paths

In sixth step, we use an optimizer to update the weights of the models in order to minimize the Loss function at next iteration. Optimizer use a parameter called learning rate to optimize the values of the weights of the model.

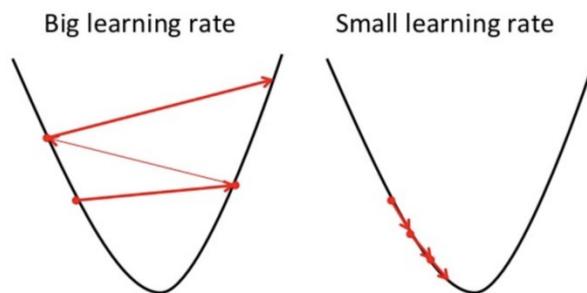


Figure 19. Learning rate

From figure 19, we can observe that learning rate influence the speed which Loss function reach the minimum.

Big learning rate means that weights of the models are drastically modified at each iteration, so there are high chance that Loss function never converge to the global minimum, but always remain around it. With small learning rate we have better probability that Loss function converge to the minimum, but slower. A disadvantage of using small learning rate is that if the Loss function is Non-Convex, there is the possibility that it converges to local minimum instead of global minimum.

At final step, we have to repeat the process described from step two to step six until we reach the convergence to minimum of Loss function.

1.5 MPC Simulation Model

Model predictive control (MPC) is an advanced method of process control that is used to control a process while satisfying a set of constraints. Model predictive controllers rely on dynamic models of the process, most often linear empirical models obtained by system identification. The main advantage of MPC is the fact that it allows the current timeslot to be optimized, while keeping future timeslots in account. This is achieved by optimizing a finite time-horizon, but only implementing the current timeslot and then optimizing again, repeatedly. MPC has the ability to anticipate future events and can take control actions accordingly.

Model Predictive Control (MPC) are a multivariable control algorithm that uses:

- an internal dynamic model of the process
- a cost function over the finite-horizon
- an optimization algorithm minimizing the cost function using the control input

MPC is based on iterative, finite-horizon optimization of a plant model. At time t the current plant state is sampled and a cost minimizing control strategy is computed for a relatively short time horizon in the future: $[t, t + T]$. Specifically, an online or on-the-fly calculation is used to explore state trajectories that emanate from the current state and find a cost-minimizing control strategy until time $t + T$. Only the first step of the control strategy is implemented, then the plant state is sampled again and the calculations are repeated starting from the new current state, yielding a new control and new predicted state path. The prediction horizon keeps being shifted forward.

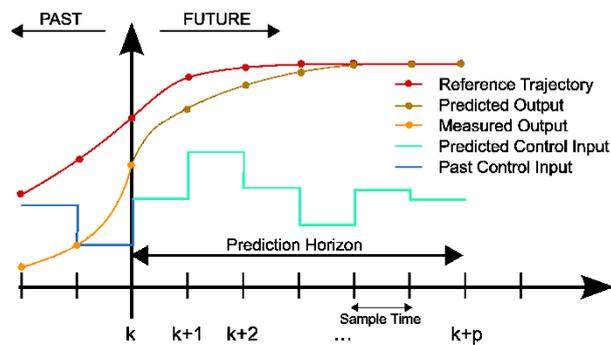


Figure 20. Example of discrete MPC scheme

Part 2: PROJECT DESCRIPTION

The objective of the thesis is to find the optimal models to predict Electric Load time series values of HRI facility and its features and apply the predictions obtained from those models in an Energy Management System simulation model to verify how the predictions influence the control output in order to achieve the best tradeoff between ambient temperature and energy costs.

We investigated three major types of models: Recurrent Neural Network, Linear Regression models and Ensemble models.

The start point of our research is “load_15min_avrg_monitoring_server_v2.csv” dataset that contain different features inherent electric power load measured at HRI facility. More precisely, time series z35 is the feature we want to predict with our models, it contains 29188 electric load power values sampled every 15 minutes from 2017-09-30 at 23:15:00 to 2018-08-01 at 00:00:00.

From time series z35, we also extract and predict some useful features to have a confidence bound where the predictions of electric load probably fall, these features are: standard deviation of z35, variance of z35, minimal value of z35 and maximal value of z35.

The methodology used to find the optimal models for every feature we have to predict is:

- Analysis of time series z35;
- Preprocessing of the data;
- Compare between Univariate and Multivariate models, to verify if the features added in Multivariate models can improve the prediction performance; in particular for RNN the approach used is to choose a base structure of the model and after have found the right optimizer and cost

function, we compare prediction results between different combinations of the features, to verify their influence on the predictions.

- For Linear Regression and Ensemble models, we compare results between different kind of machine learning algorithms with default configuration of the hyperparameters. After we have found the most promising models we apply a Grid Search process to them to optimize the hyperparameters considering as reference the prediction of z35 time series in Univariate case. Then we use the optimized configuration found also on the other models realized in Univariate and Multivariate case for predicting z35 and its features.
- After we have found for each type of regression models the best models for predicting z35, z35 variance, z35 standard deviation, z35 minimal value and z35 maximal value, we compare them and based on error metrics score we decide which model between RNN, Linear and Ensemble is the best for predicting those features.

Then we describe the process for generalizing the best models found for predicting other time series related to electric load power.

At the end we use the predictions of two different electric load time series and the prediction of their minimal and maximal values in the simulation framework to investigate their influence on the output produced.

- Analysis of time series z35

First of all, we have to observe time series z35, because we can find some pattern information that can be useful for preprocessing of the data.

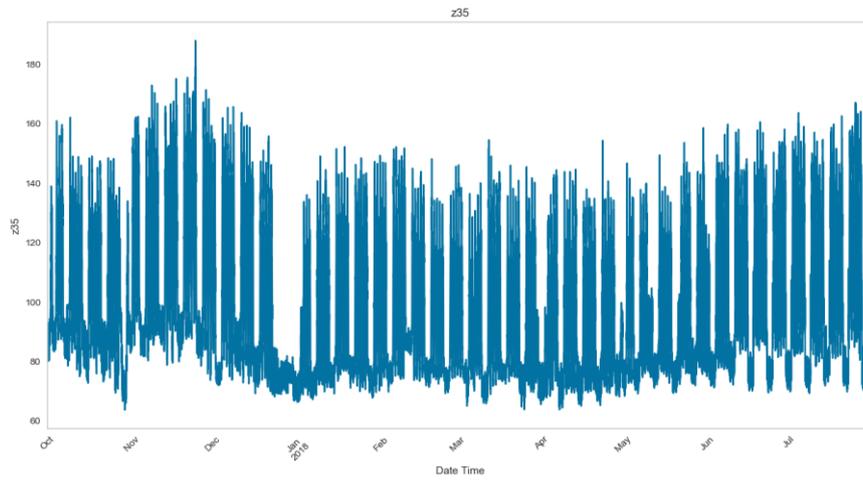


Figure 21. Time Series z35

From figure 21, we can observe a constant level of trend of the values of the time series and the presence of a seasonality. We can decompose z35 in Trend, Seasonal and Residual component to verify if this assumptions are true.



Figure 22. Trend, Seasonal and Residual component of z35

From figure 22, Seasonal component of z35 time series is characterized of a weekly seasonality. This information will be useful for the choice of number of timesteps to provide as input for the models.

All of the models tested, from RNN to Linear and Ensemble Regression model, are multistep models, it means that their predictions are formed of more than one timestep in the future.

2.1 RNN Models

The first model we want to investigate is the Recurrent Neural Network (RNN). Our RNN is based on LSTM (Long Short-Term Memory) layers and we used Keras library with Tensorflow Backend to implement it. The model request as input a 3D Tensor with shape (batch_size, n_timesteps, n_features) so we have to manipulate data from time series z35 to get it of this size.

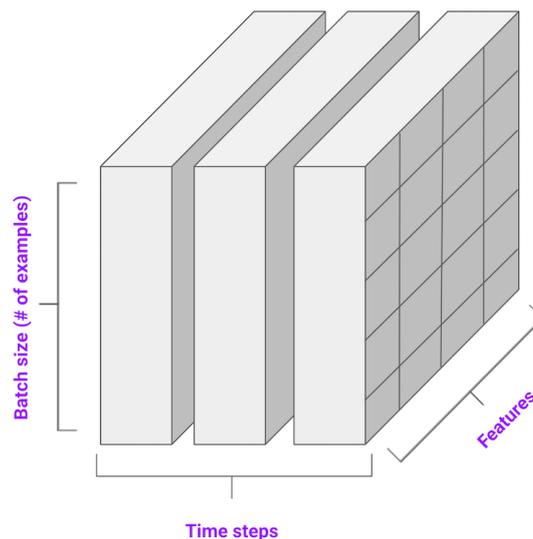


Figure 23. 3D Input Tensor

As we observed, time series z35 has a weekly seasonality and its values are sampled every 15 minutes, it means we have four samples per hour and a total of $4 \times 24 \times 7 = 672$ samples per week. We choose to consider this interval of data because if the time series has really a weekly seasonality, the informations the model needs are inside this time window of 672 steps.

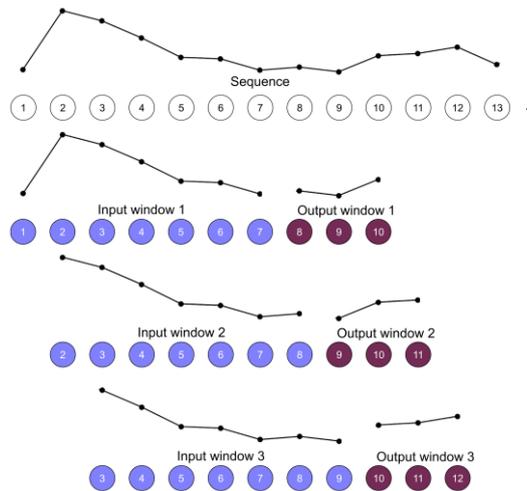


Figure 24. Example of using time window on dataset for *past_history* and *future_target*

2.1.1 RNN – Test n. 1

The first RNN model realized use a time window of 168 timesteps for *past_history* (equal to *n_timesteps* dimension of 3D Input Tensor) and a time window of 96 timesteps as *future_target*, so the model is trained to predict 96 timesteps of z35 in the future when we provide it a window of *past_history* of 168 timesteps of z35. We choose 168 timesteps for *past_history* because, in the interval of 672 timesteps of a week, we consider only one sample per hour, so $672/4=168$ timesteps, in this way our first RNN model will be less expensive in computational terms in training phase. This time window formed by *past_history*

timesteps and future_target timesteps is shift along all z35 dataset to create the samples to provide to the model.

The dimension n_features of the 3D Input Tensor depend if the model is Univariate or Multivariate. For this RNN model in Multivariate case we use some other features inside dataset “load_15min_avrg_monitoring_server_v2.csv” that are sampled at same frequency of z35. This features are:

- Ambient Temperature;
- Wetbulb Temperature;
- Relative Humidity;
- Air Pressure.

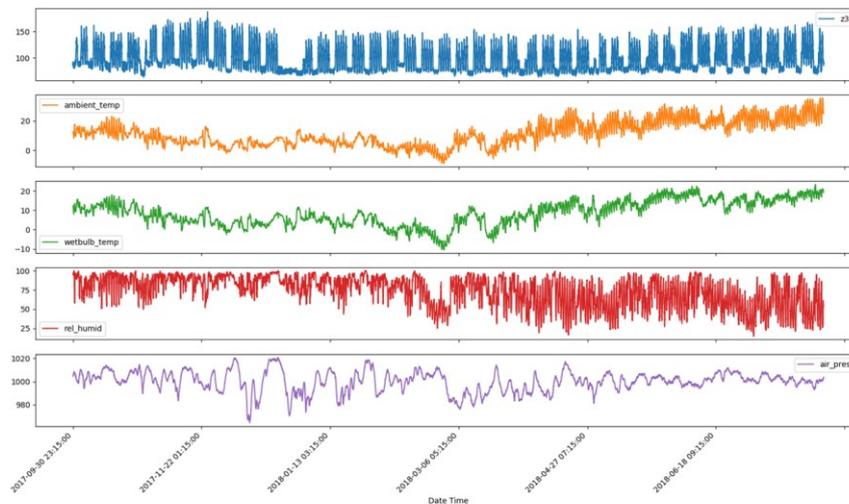


Figure 25. RNN Multivariate – Features used

The features used in Multivariate case have different range of values, so to make the model give the same importance to all the features, we have to apply a technique of scaling values. In this model we apply standardization, it uses mean μ and standard deviation σ of each feature to have values with zero mean and unit standard deviation.

$$x_{standardized} = \frac{x_i - \mu}{\sigma}$$

Equation 2. Standardization

RNN model used supervised learning and mini-batch gradient descent technique in training phase, so for every batch of data provided to the model that contains the samples of past_history window and corresponding future_target window, it make predictions and calculate the loss function to minimize and then it optimize the weights using an optimizer in combination with Backpropagation technique.

The dimension batch_size of 3D Input Tensor means how many samples of past_history window there are inside every batch.

In summary, the steps to preprocessing time series data are:

- Divide z35 dataset and the extra features used in Multivariate case in training and test set. We use the first 25000 values as training set and the remaining 4188 values as test set.
- Apply the time window of past_history and future_target along training set and test set. We use one timestep per hour for past_history window, so we consider only 168 timesteps of the 672 timesteps of a week, and use 96 timesteps of the next day for future_target window.
- For training set, we reorganize every couple of past_history window and corresponding future_target window in a buffer of dimension 10000, we shuffle the data to prevent overfitting and group them in batch of dimension 256. After we applied the time window to training set, we have a total of $25000 - 672 = 24328$ samples of past_history and corresponding future_target, so if we use a buffer size of 10000 we discard a total of 14328 samples. From one side it's an advantage in terms of complexity, because there are less samples for training, but from the other side it can be a problem if the model is underfitted with the amount of data presented. Also the size of the batch influence the

training of the model, because large batches lead to more accurate model but with slow iteration during training instead of having small batches.

- For test set, we only reorganize data in batch of dimension 256, without using any buffer.

- [Structure and description of RNN model](#)

RNN is a sequential model formed by different layers that interpose between 3D Tensor Input and Output prediction, we report the constructor of the model.

Constructor of RNN model (same for Univariate and Multivariate case):

```
multi_step_model = tf.keras.models.Sequential()

multi_step_model.add(tf.keras.layers.LSTM(32, return_sequences=True,
input_shape=x_train_multi.shape[-2:]))

multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))

multi_step_model.add(tf.keras.layers.Dense(96))

multi_step_model.compile(optimizer=tf.keras.optimizers.RMSprop(clipvalue=1.0)
, loss='mae')
```

The model is formed by two LSTM layers with a last layer Dense that produces the output prediction. Now we describe how each layer works:

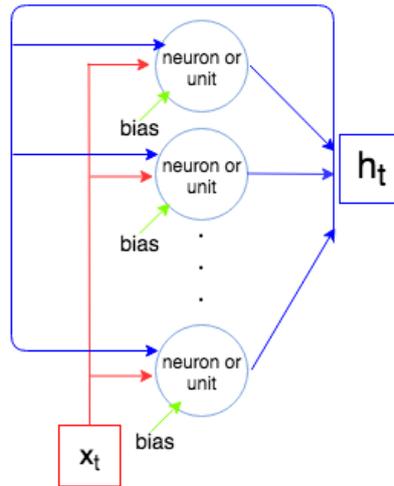


Figure 26. LSTM Layer Structure

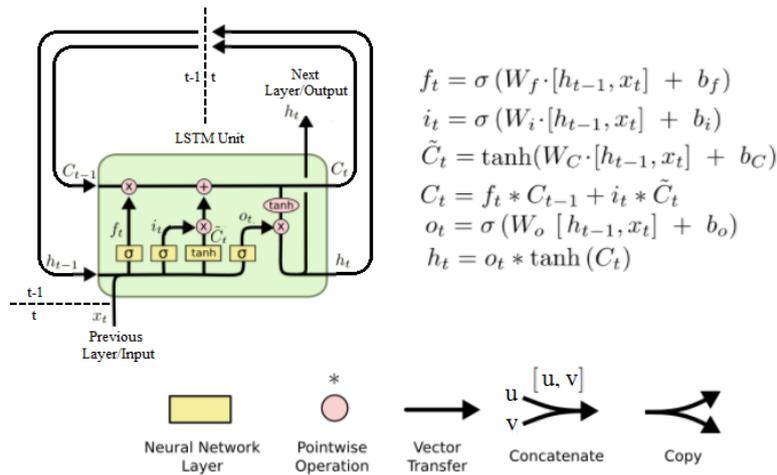


Figure 27. LSTM Unit

- The first LSTM layer is composed by 32 units and receive the Input 3D Tensor previously described of dimension (batch_size=256, n_timesteps=168, n_features), so every element of the batch that correspond to a window of past_history is a 2D Tensor of dimension (n_timesteps=168, n_features). Referring to figure 26, we have an input vector x_t to each unit of the layer with t assuming values from 1 to 168 (n_timesteps) and we calculate and save the output produced h_t , it means that at beginning we will have as input a vector x_1 with dimension (1, n_features) applied to each unit that produces an output h_1 with dimension (1, 32) and so on for every timesteps of the input sequence to

iterate until we have a final output h_t of dimension (168, 32). In this case we report in the final output of the layer all the output iterations because we set the attribute of the LSTM layer `return_sequences=True`

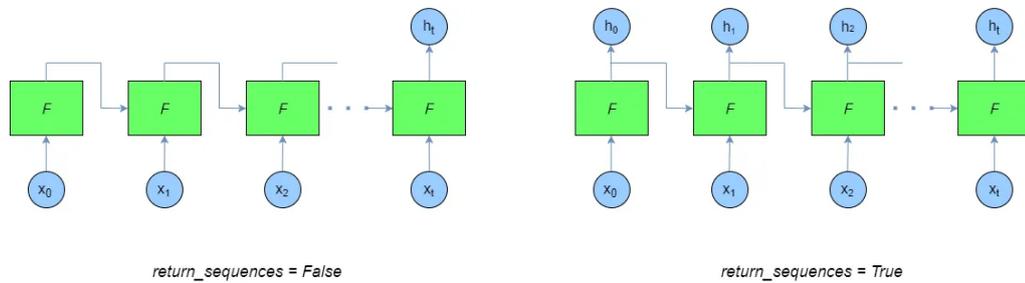
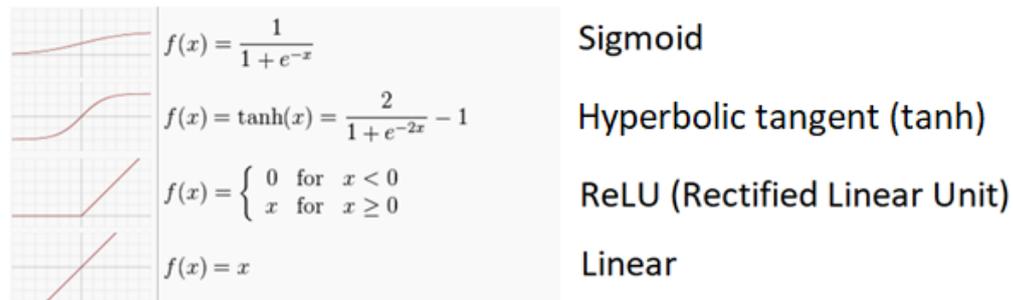


Figure 28. Attribute `return_sequences` LSTM layer

We repeat this procedure for every elements in the batch, so the final output of the first LSTM layer is a 3D Tensor of dimension (256, 168, 32).

- The second LSTM layer is composed by 16 unit and receive the output 3D Tensor of the previous layer. This layer works as the other LSTM layer with the differences that it uses the activation function ReLU for Cell State and Hidden State instead of hyperbolic tangent (function tanh figure 27) and report in output only the last output for every input sequence iterated. In this case we have an input vector x_t of dimension (1, 32) with t assuming values from 1 to 168 and we save only the final output h_t of dimension (1, 16). We repeat this process for every element in the batch, so at the end we will have a 2D output tensor of shape (batch_size, 16).
- Last layer Dense is formed by 96 units and calculate the output prediction of the model, each units use a linear activation function to produce the output. For every elements in the batch we have an input vector of dimension (1, 16) applied to each units of the layer, so the output will have shape (1, 96) that correspond to a future_target window predicted. We repeat this process for all the elements in the batch, so we will have a final output of shape (256, 96).

The activation function used in the layers are:



Equation 3. Activation functions

The number of parameters trained in each layer of the model, can be obtained with this formulas:

$$\text{LSTM layer: } 4 * out * (inp + out + 1)$$

$$\text{Dense layer: } out * (inp + 1)$$

Where *out* indicate the number of unit of the layer, *inp* indicate the last dimension of input tensor (in first LSTM it's equal to *n_features*, in second LSTM layer it's equal to 32 and in Dense layer it's equal to 16). Addend 1 indicate bias term and factor 4 in LSTM layer indicates that we have four parameter in each unit.

- First LSTM layer: $4 * 32 * (5 + 32 + 1) = 4864$
- Second LSTM layer: $4 * 16 * (32 + 16 + 1) = 3136$
- Dense layer: $96 * (16 + 1) = 1632$

At the end we have a total of $4864+3136+1632=9632$ parameters to train in the RNN model.

In training phase we have to specify the number of epochs, so the number of times we passed the entire training dataset to the model; this attribute is set to 10. In RNN model we have to specify also the attribute *steps_per_epoch* which

determine how many batches the model has to iterate to consider finished an epoch; for this model the attribute is set to 200, it means that the model iterate more times the entire training dataset to end an epoch because train set is formed by $\text{buffer_size}/\text{batch_size} = 10000/256 = 39,0625$ batch that is lower than 200.

Loss function chosen that we have to minimize during training is Mean Absolute Error (MAE) and the optimizer used to achieve this is RMSprop.

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

$y_i = \text{target variable}$; $\hat{y} = \text{predicted variable}$

Equation 4. MAE Loss Function

For each Parameter w^j

(j subscript dropped for clarity)

$$\nu_t = \rho \nu_{t-1} + (1 - \rho) * g_t^2$$

$$\Delta \omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta \omega_t$$

η : *Initial Learning rate*

ν_t : *Exponential Average of squares of gradients*

g_t : *Gradient at time t along w^j*

Equation 5. RMSprop algorithm

In RMSprop algorithm, exponential moving average of square of gradient at current iteration depends on exponential moving average of square of gradient at previously iteration and on square of gradient of loss function, multiplied by factor $\rho \in [0, 1]$ (good default value 0,9). Hyperparameter ρ takes into account previously iterations, higher is its value and greater importance is acquired by past iterations instead of current value of gradient.

After we have trained the model, we verify the performance of the prediction using data in test set and calculate the following error metrics score:

- R^2 Score (Best Score = 1.0):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

- Explained Variance Score (Best Score = 1.0)

$$EVS = 1 - \left(\frac{Var[y - \hat{y}]}{Var[y]} \right)$$

- Mean Absolute Error (Best Score = 0)

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

- Median Absolute Error (Best Score = 0)

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, \dots, |y_n - \hat{y}_n|)$$

- Mean Squared error (Best Score = 0)

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

$y_i = target\ value$; $\hat{y} = predicted\ value$; $\bar{y} = mean\ value$

The first two error metrics score (R^2 and EVS) are correlated to the variance of the prediction and are equal when the mean of residues is null:

$$Var[y - \hat{y}] = \frac{\sum((y - \hat{y}) - \overline{(y - \hat{y})})}{n} \text{ if } \overline{(y - \hat{y})} = 0 \rightarrow EVS = R^2$$

The following two error metric scores are correlated with absolute value of the residues and the last one is correlated to square of value of the residues. In presence of outliers in time series, error metrics correlated to absolute value of the residues are more robust than metrics correlated to square of value of the residues.

We report now the results of the predictions of the RNN model in Univariate and Multivariate case. In this first RNN model, error metrics score are calculated on standardized data and using only one batch of test set, so using 256 predictions.

- **RNN Univariate (z35)**

Input shape $x = (256, 168, 1)$

True_target shape $y = (256, 96)$

Predicted_target shape $\hat{y} = (256, 96)$

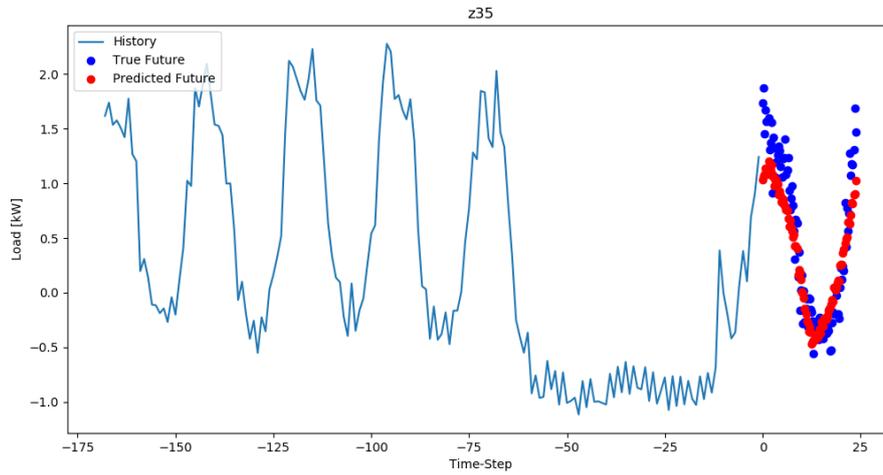


Figure 29. Example of one RNN model prediction using the batch of test set where we calculate error metrics

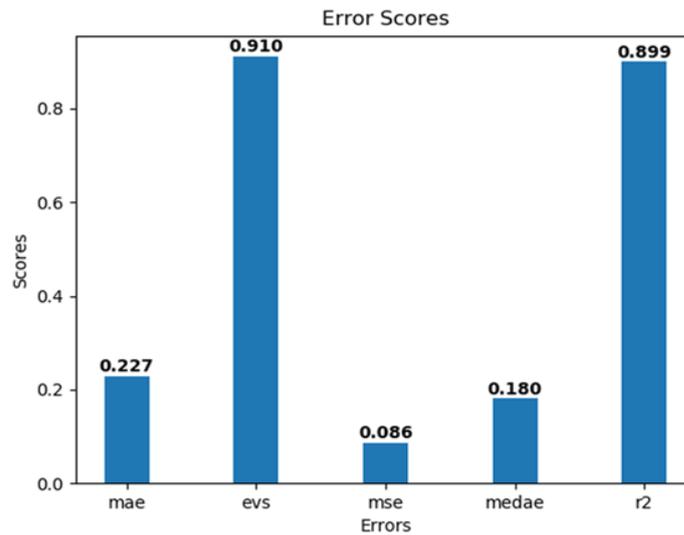


Figure 30. Error metrics score calculate on the batch of test set

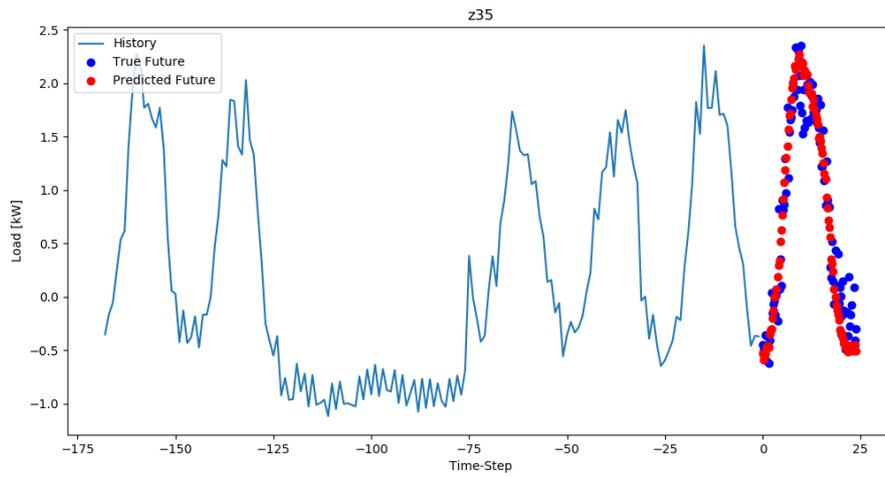


Figure 31. Example of one RNN model prediction using another batch of test set where we calculate error metrics

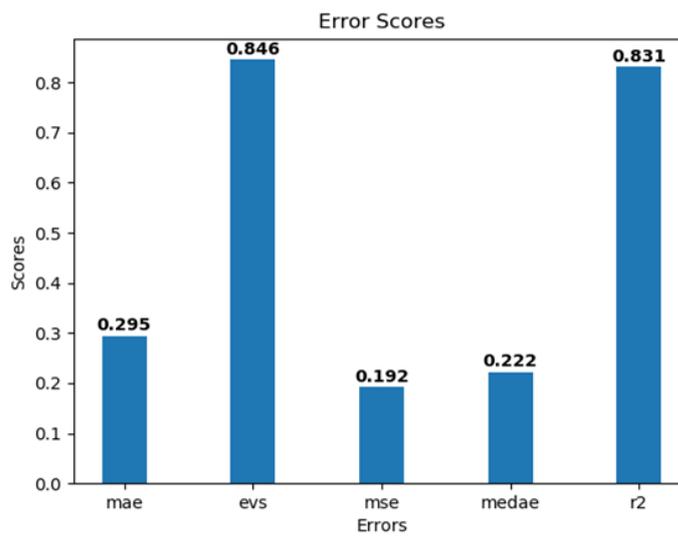


Figure 32. Error metrics score calculate on the batch of test set

- **RNN Multivariate (z35, ambient_temp, wetbulb_temp, rel_humid, air_press)**

Input shape $x = (256, 168, 5)$

True_target shape $y = (256, 96)$

Predicted_target shape $\hat{y} = (256, 96)$

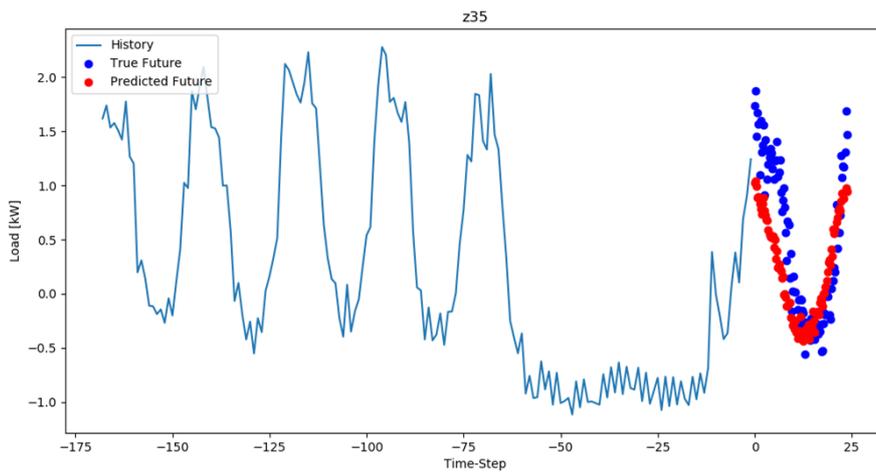


Figure 33. Example of one RNN model prediction using the batch of test set where we calculate error metrics

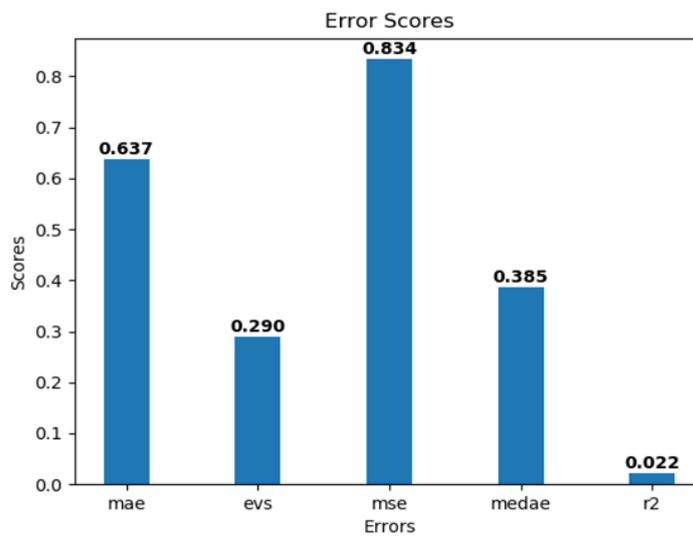


Figure 34. Error metrics score calculate on the batch of test set

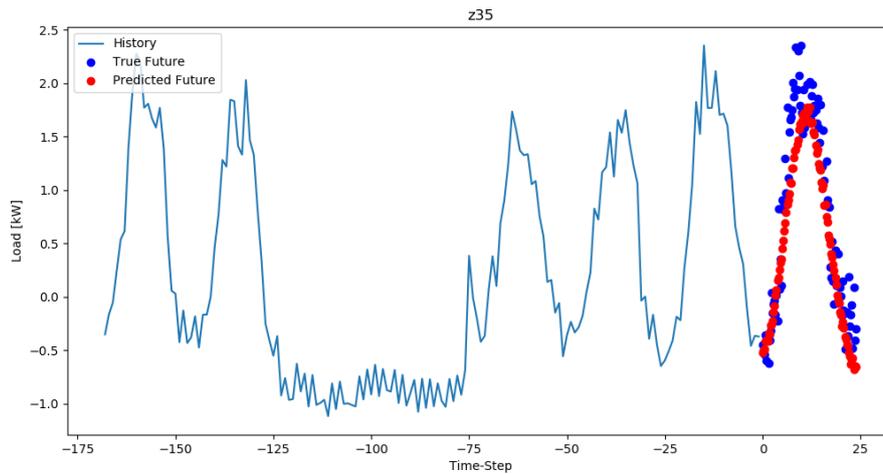


Figure 35. Example of one RNN model prediction using another batch of test set where we calculate error metrics

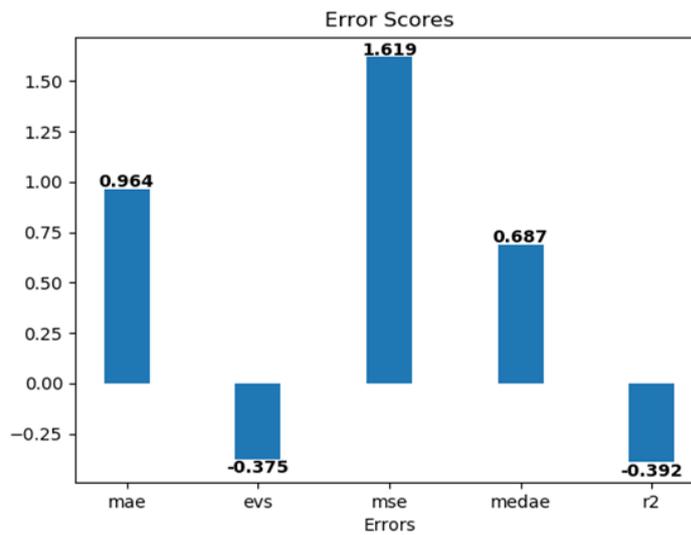


Figure 36. Error metrics score calculate on the batch of test set

From this first comparison based on error metrics score, we can observe that RNN Univariate model has better prediction performance than Multivariate model, so it means that extra features added don't provide more informations to the model, but they are seen as noise.

2.1.2 RNN – Test n.2

In the first RNN Multivariate model we have used some of the other features saved in dataset “load_15min_avrg_monitoring_server_v2.csv”. Now for predicting z35 future values in Multivariate case we want to use some features extracted directly from z35 time series, this technique is properly called feature extraction. The features extracted are mean (z35_hour_mean), standard deviation (z35_hour_std) and variance (z35_hour_var) calculated every hour, so using 4 timesteps of z35 per time. For this model we use all of the 672 timesteps of a week for past_history window, and not only one timestep per hour like in the previous models, because electric load can significantly change in less than an hour, so we use all the informations provided by time series z35. The future_target window is always formed by 96 timesteps of the next day.

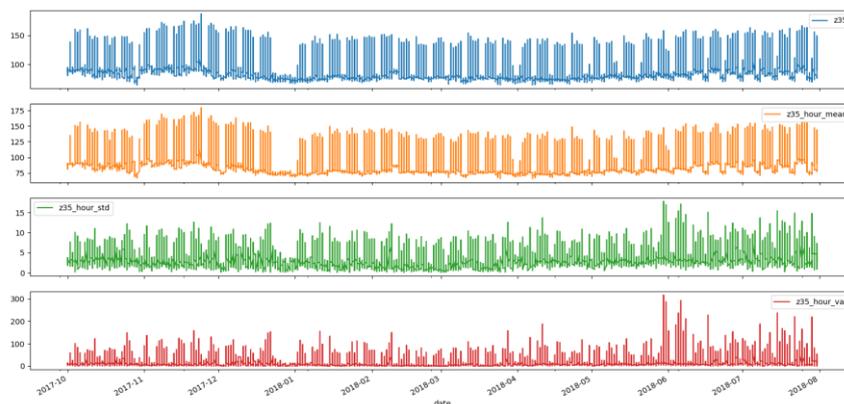


Figure 37. Time Series z35, z35_hour_mean, z35_hour_std, z35_hour_var

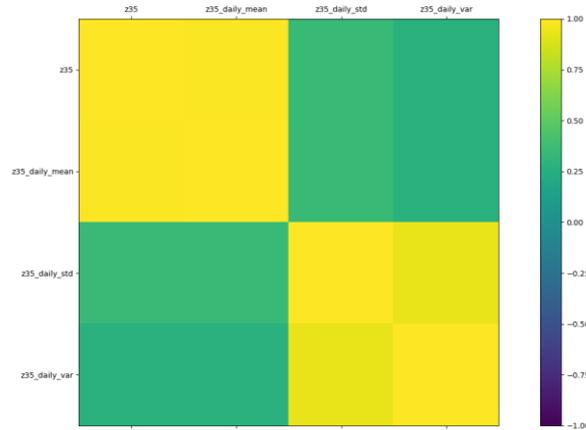


Figure 38. Correlation between features

We have also discarded from original z35 time series four samples to make it more clear, in order to have a total of 29184 sampled every 15 minutes from 2017-10-01 at 00:00:00 to 2018-07-31 at 23:45:00, in summary we have $29184/96=304$ days of values of electric load power. In this case we divide z35 dataset in 20000 samples for training set, 5000 samples for validation set and the remaining 4184 samples for test set. We use also a validation set in this model, it allows to calculate the loss function to minimize using unseen data at the end of every epoch of training, in order to have an idea of the performance prediction before the end of the training phase. During training, for validation set we have to specify the attribute *validation_steps* that set the number of batch the model has to iterate before calculate loss function, in our case it's set to 50. The other difference from the previous model is the size of the batch that is set to 128 instead of 256, this choice is made for faster training because now we use all of the 672 samples of a week, so the sequences in input to iterate are bigger.

In summary, the characteristics of the model are:

- 29184 samples for every features used in the model, with 20000 samples for training set, 5000 samples for validation set and 4184 sample for test set.
- Scaling of the values using standardization
- Past_history window of 672 timesteps and future_target window of 96 timesteps
- Buffer_size = 10000 and batch_size = 128
- Number of epochs = 10; steps_per_epoch = 200; validation_steps = 50

The structure of the RNN model is the same of the previous model, with two LSTM layers and a final layer Dense. We have also added RMSE error metric score to verify the performance of the prediction on test set:

- Root Mean Squared error (Best Score = 0)

$$RMSE = MSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$$

We report now the results of the predictions of the RNN model. Error metrics score are calculated on standardized data and using only one batch of test set, so using 128 predictions.

- **RNN Multivariate (z35, z35_hour_mean, z35_hour_std, z35_hour_var)**

Input shape $x = (128, 672, 4)$

True_target shape $y = (128, 96)$

Predicted_target shape $\hat{y} = (128, 96)$

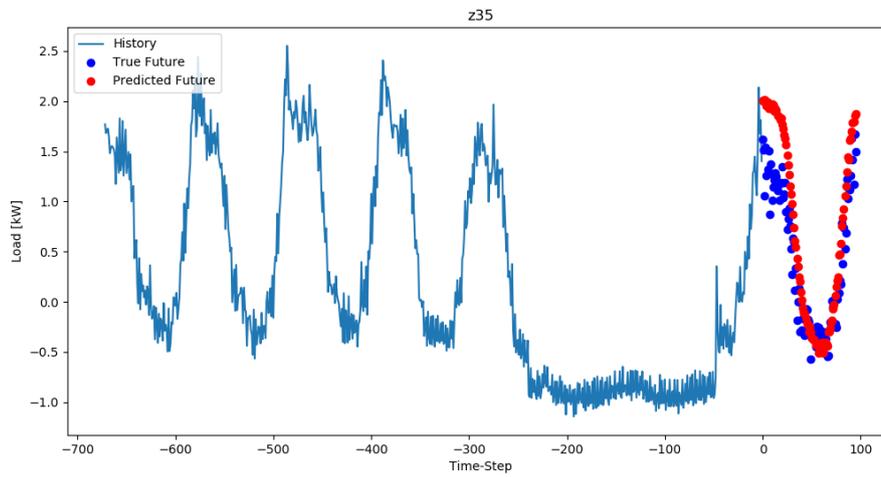


Figure 39. Example of one RNN model prediction using the batch of test set where we calculate error metrics

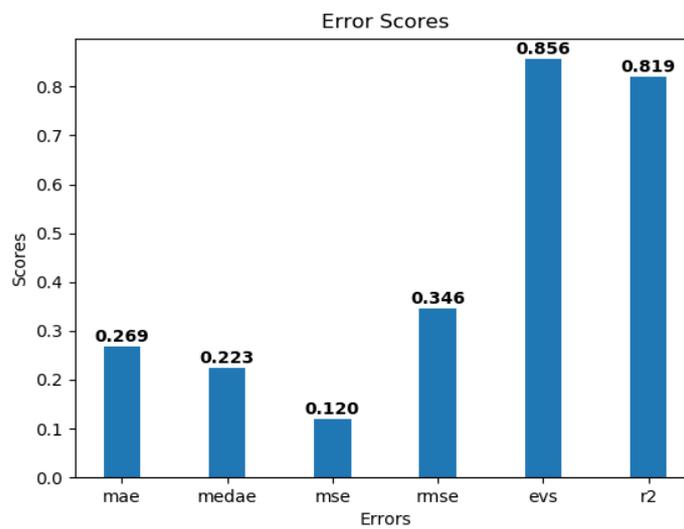


Figure 40. Error metrics score calculate on the batch of test set

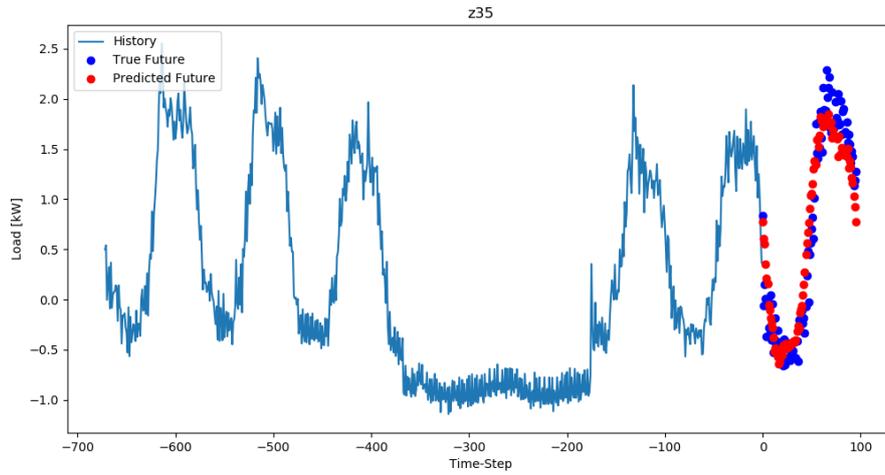


Figure 41. Example of one RNN model prediction using another batch of test set where we calculate error metrics

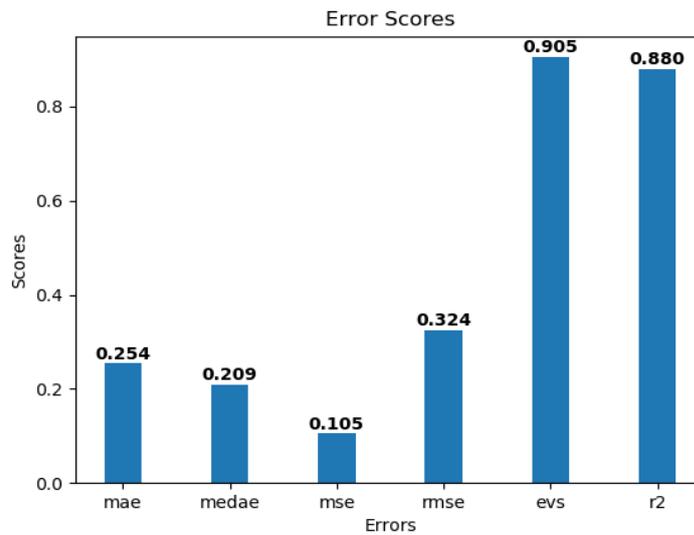


Figure 42. Error metrics score calculate on the batch of test set

Comparing figures 34 and 36 of error metrics score of model RNN Multivariate (z35, ambient_temp, wetbulb_temp, rel_humid, air_press) and figures 40 and 42 of error metrics score of model RNN Multivariate (z35, z35_hour_mean, z35_hour_std, z35_hour_var), we can see an improvement in prediction

performance in this last model, with results comparable to those obtained with the RNN Univariate model. It means that the choice of these features extracted from z35 time series gives to RNN Multivariate model an improvement in terms of prediction performance. The difference to notice is that in RNN Multivariate model with features extracted from z35, we calculate error metrics score using 128 predictions instead of 256 predictions used in the first two RNN models.

We have observed that time series z35 is characterized by a weekly seasonality, with a different behaviour between weekdays and weekend days, as shown in the figure:

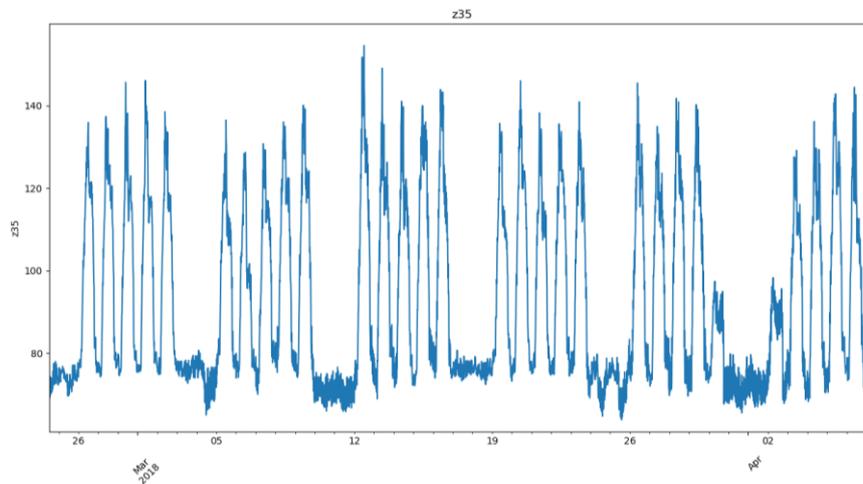


Figure 43. Behaviour of z35 time series during weeks

So we have create a time series called “weekday” where we coded the seven days of a week with number that goes from 0 to 6, where 0 represent Monday and 6 represent Sunday, because we want to investigate if the correlation between z35 and weekday time series can improve the performance of the predictions like the features extracted in the previous Multivariate model.

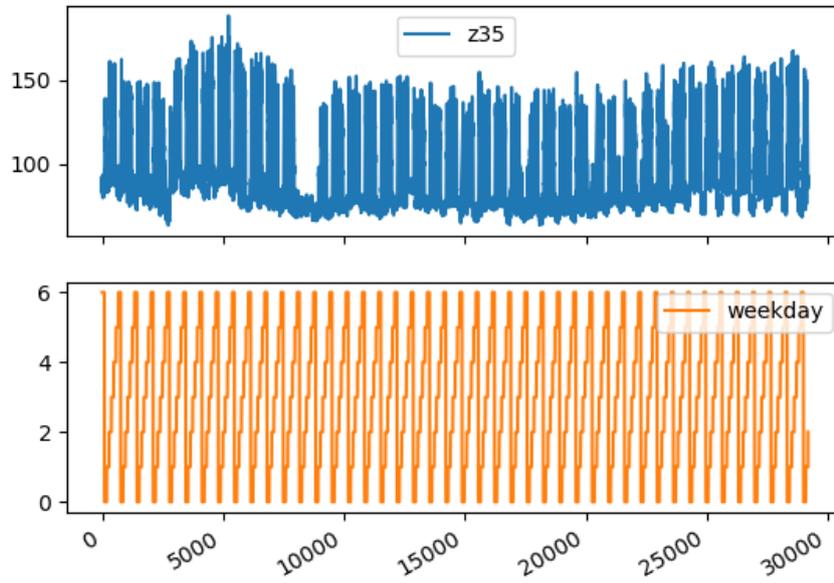


Figure 44. z35 and weekday Time Series

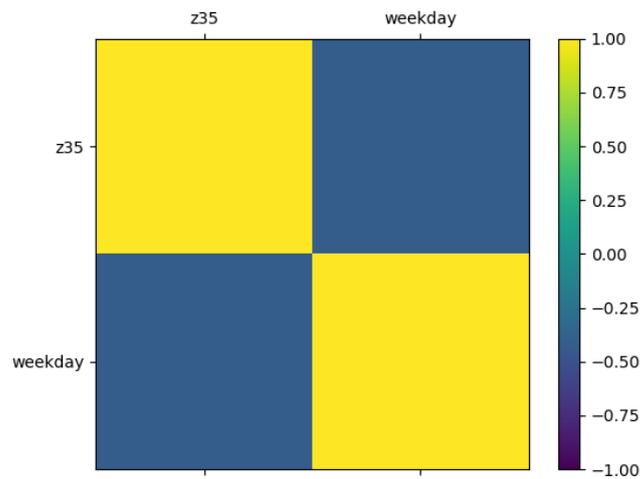


Figure 45. Correlation between features

The structure and the characteristics of this RNN Multivariate model are the same of the model RNN Multivariate (z35, z35_hour_mean, z35_hour_std, z35_hour_var) with the only difference of the features used, so also here the error metrics score are calculated on 128 predictions.

- **RNN Multivariate (z35, weekday)**

Input shape $x = (128, 672, 2)$

True_target shape $y = (128, 96)$

Predicted_target shape $\hat{y} = (128, 96)$

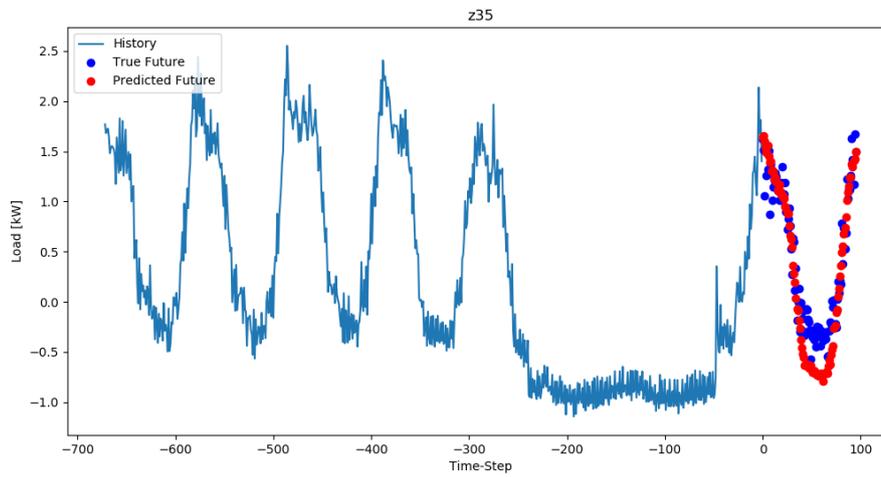


Figure 46. Example of one RNN model prediction using the batch of test set where we calculate error metrics

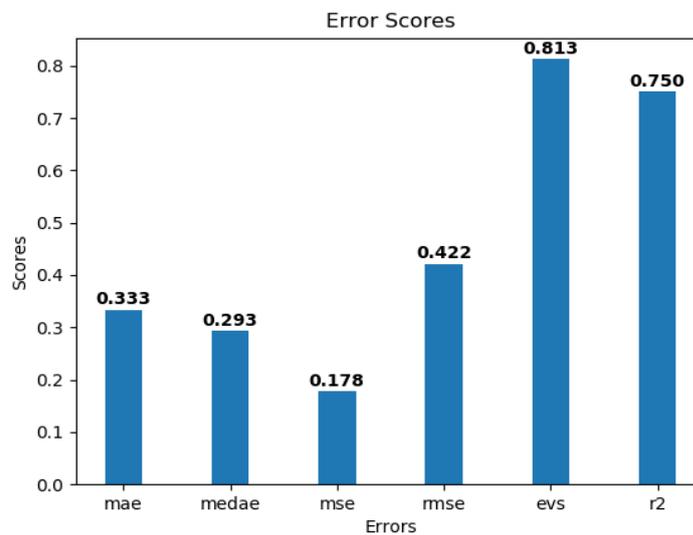


Figure 47. Error metrics score calculate on the batch of test set

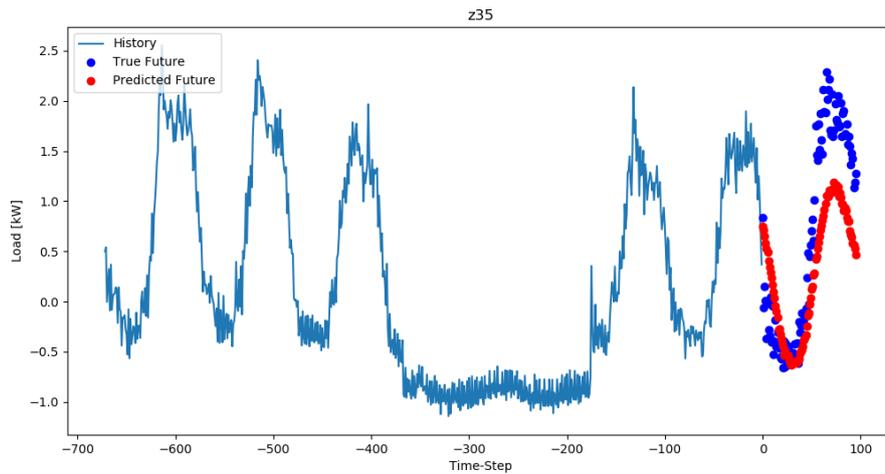


Figure 48. Example of one RNN model prediction using another batch of test set where we calculate error metrics

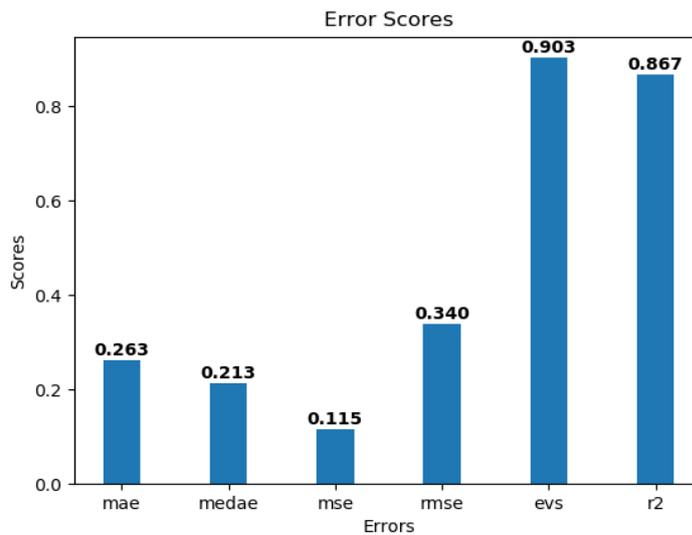


Figure 49. Error metrics score calculate on the batch of test set

From observation of error metrics score of figures 47 and 49, we can conclude that also using time series weekday in Multivariate model can improve the accuracy of the predictions. From correlation plot between features of figures 38 and 45, respectively of RNN Multivariate (z35, z35_hour_mean, z35_hour_std,

z35_hour_var) and RNN Multivariate (z35, weekday), we can notice that we obtained an improvement in Multivariate models using in the first case features with a direct correlation with time series target z35 and in the second case using a feature with inverse correlation with z35.

2.1.3 RNN – Final Test

We observed from previous RNN Multivariate models that the features extracted from z35 and the time series weekday can improve the performance of the models. Now we will test different combinations of the features to verify which one have the most accurate prediction performance. The features extracted from time series z35 for this test are: z35_daily_mean; z35_daily_std; z35_6hours_std; z35_daily_var; z35_6hours_var, z35_daily_min; z35_6hours_min; z35_daily_max; z35_6hours_max. In this case we have also time series related to minimal and maximal values of z35 and the time interval used to calculate all of the time series extracted is daily (96 samples) or of six hours (24 samples), because using a time interval of only one hour (4 samples) like in the previous model is not too much significative for this kind of features.

The objective of the research is also to find a confidence bound for the predictions of electric load power, so after we have found the best model for predicting z35 future values, we will test different models to predict some of the feature extracted from z35 that are: z35 standard deviation, z35 variance, z35 minimum value and z35 maximum value.

The differences in the configuration from the previous models tested regarding the technique of scaling of the values, the optimizer and the loss function that the model try to minimize during training and the calculation of error metrics score.

In these models we use the normalization of values between 0 to 1 as technique of scaling, because it lead to a better generalization of the models trained.

$$x_{normalized} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Equation 6. Normalization

Indeed another objective of this research is to investigate the generalization of the most promises models for predicting the features of interest. For generalization of the models we intend using the models trained on time series z35 to other time series related to electric power load. These time series have characteristics similar to z35, like weekly seasonality, but can differ for the range of the values assumed. In this case using normalization technique instead of standardization is better for the generalization of the models because with normalization we will always have scaled values between 0 and 1 for all the time series we test. Instead if we consider z35 and another electric load time series with different range of values and apply standardization to both of them, we still obtain different range of scaled values and the models trained on z35 applied to the other time series can have problem to predict well this new values, because are values never seen during training.

The optimizer and the loss function used in these new models are Adam (Adaptive moving Average) and MSE (Mean Squared Error). We choose Adam as optimizer instead of RMSprop because it allows to lead the minimum of the cost function faster, this happen because when it updates the weights of the models during training it uses both the exponential moving average of squares of gradients (like RMSprop) and the exponential moving average of gradients. The step size used to update each weight of the model is determined in an adaptive way, in fact if the gradient does not vary much the step size will be greater, while it will be less when the gradient changes rapidly.

For each Parameter w^j
(j subscript dropped for clarity)

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\text{Step-size} \Rightarrow \Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : Initial Learning rate

g_t : Gradient at time t along ω^j

ν_t : Exponential Average of gradients along ω_j

s_t : Exponential Average of squares of gradients along ω_j

β_1, β_2 : Hyperparameters

Equation 7. Adam algorithm

Hyperparameters β_1 and β_2 control the decay rate of these moving average, in other words they indicate at each iteration how much the old gradients contribute in the calculation of the current moving average. β_1 and β_2 values are generally set equal to 0,9 and 0,99 respectively. Hyperparameter ϵ is generally set equal to $1 * e^{-10}$.

The last difference regard how many predictions we use for calculate the error metrics score. Instead of using only one batch of data set to calculate errors, now we use all data available in test set, so we have a more accurate score on how much are correct the predictions.

In summary, we report all the models tested for predicting the five features of interest:

- Prediction of **z35**:
 - Feature used in Univariate models:
 1. z35
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);

3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);
- Prediction of **var** (z35 variance):
 - Feature used in Univariate models:
 1. z35_6hours_var
 2. z35_daily_var
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

 - Predictions of **std** (z35 standard deviation):
 - Feature used in Univariate models:
 1. z35_6hours_std
 2. z35_daily_std
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);

 - Predictions of **min** (z35 minimum value):
 - Feature used in Univariate models:
 1. z35_6hours_min
 2. z35_daily_min
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Prediction of **max** (z35 maximum value):
 - Feature used in Univariate models:
 1. z35_6hours_max
 2. z35_daily_max
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

The mainly differences between the features used in the models regarding the time interval considered (one day or six hours) and the choice to use standard deviation or variance of z35, because are features that give to the model approximately the same information being standard deviation the square root of variance. For the features extracted that we want to predict for having a confidence bound of the values of electric load power that are: std, var, min and max, we consider a time interval of a day or of six hours based on which model gives the best prediction performance.

The layers that form the RNN are the same of the previous models and we use always a time window of past_history of 672 timesteps and future_target of 96 timesteps. We report the constructor of the model.

Constructor of RNN model (same for Univariate and Multivariate case):

```
multi_step_model = tf.keras.models.Sequential()

multi_step_model.add(tf.keras.layers.LSTM(32, return_sequences=True,
input_shape=x_train_multi.shape[-2:]))

multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))
```

```
multi_step_model.add(tf.keras.layers.Dense(96))
```

```
multi_step_model.compile(optimizer='adam', loss='mse')
```

Now we report the results of the predictions with the error metrics score of the list of models trained for each features of interest.

- Prediction of **z35**

- *Univariate Models*

1) *Model name: "z35_uni";*

Feature used: z35

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

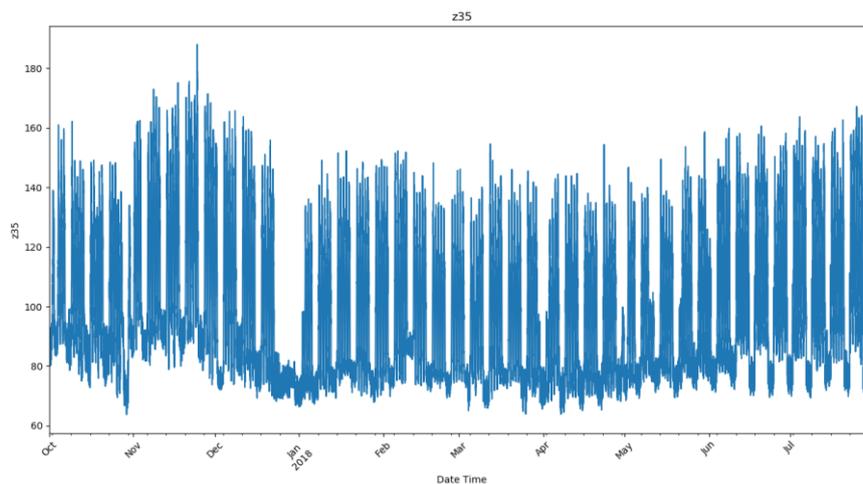


Figure 50. Time Series z35

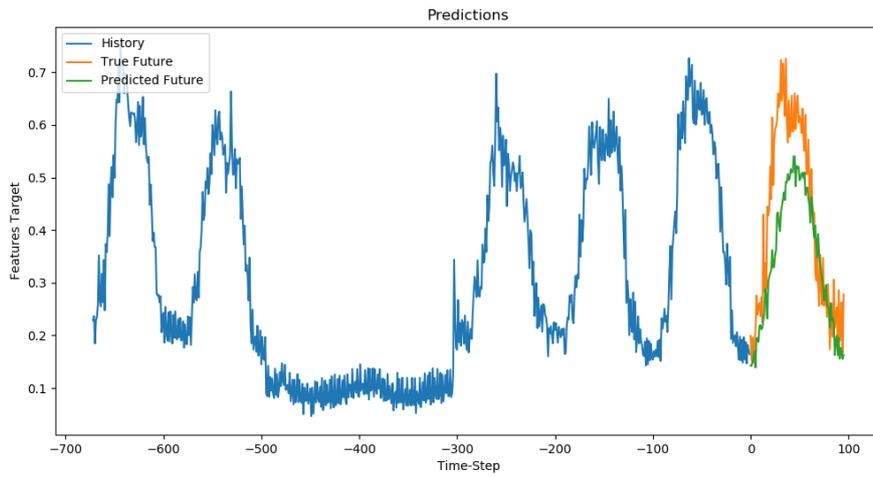


Figure 51. Example of one RNN model prediction

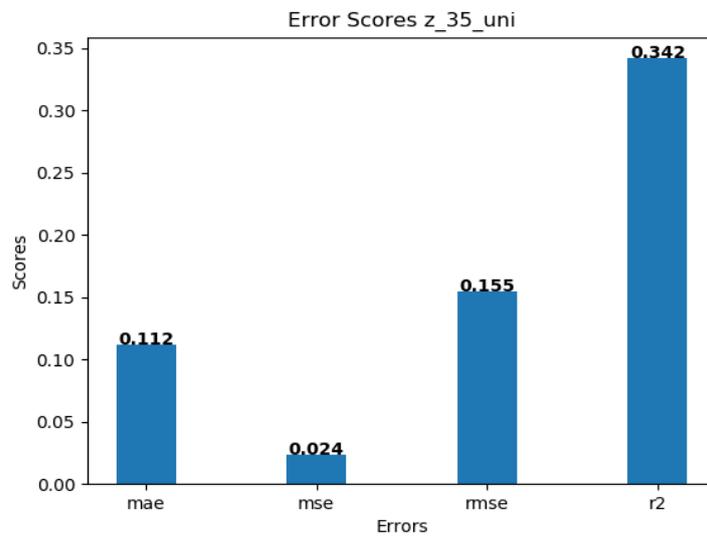


Figure 52. Error metrics score calculated on all batches of test set

- *Multivariate Models*

1) *Model name: "z35_6h_std_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

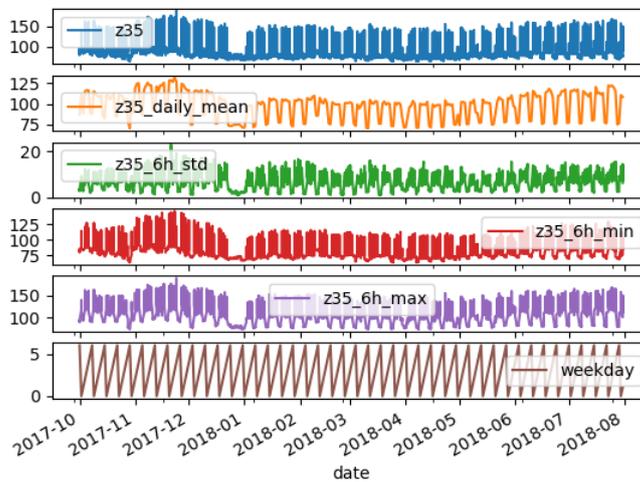


Figure 53. RNN Model Multivariate n.1 – Features used

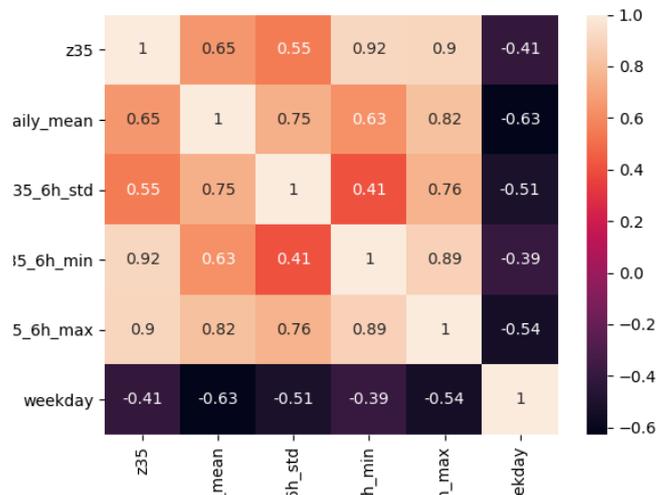


Figure 54. Correlation between features

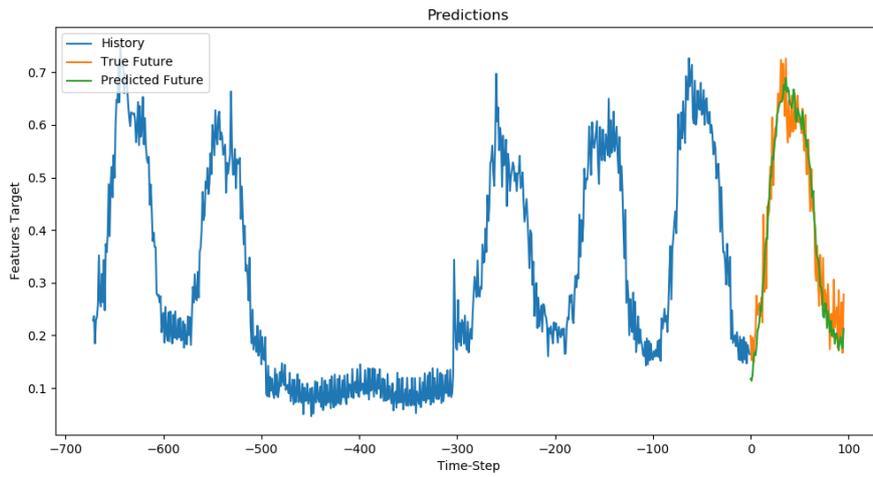


Figure 55. Example of one RNN model prediction

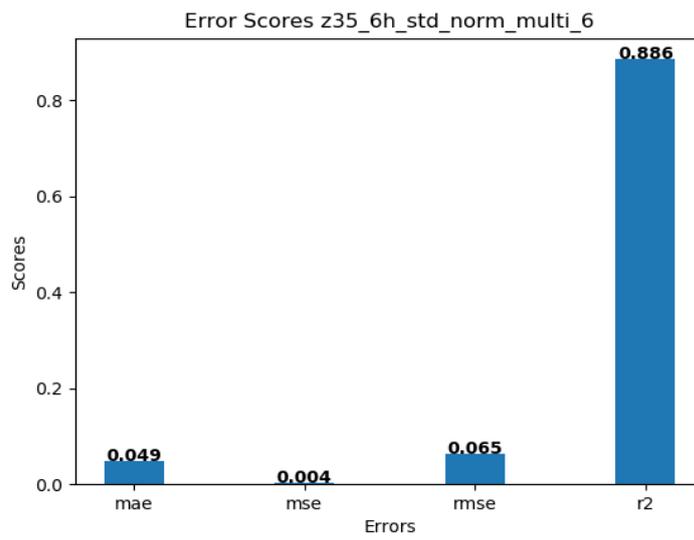


Figure 56. Error metrics score calculated on all batches of test set

2) Model name: "z35_d_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

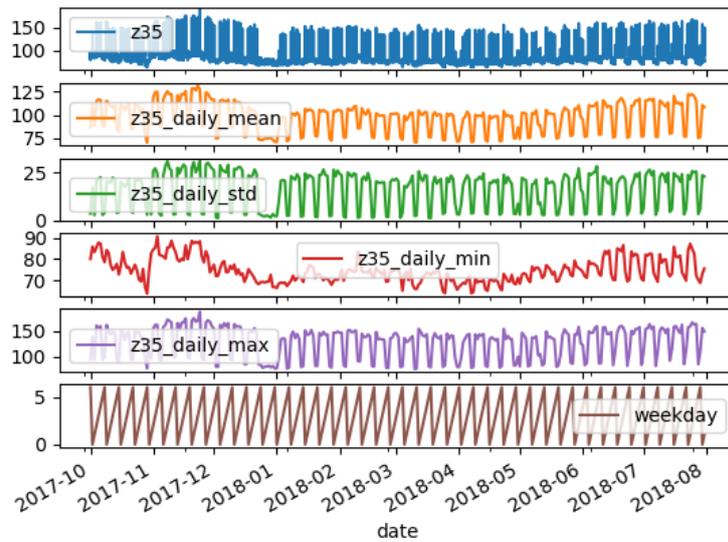


Figure 57. RNN Model Multivariate n.2 – Features used

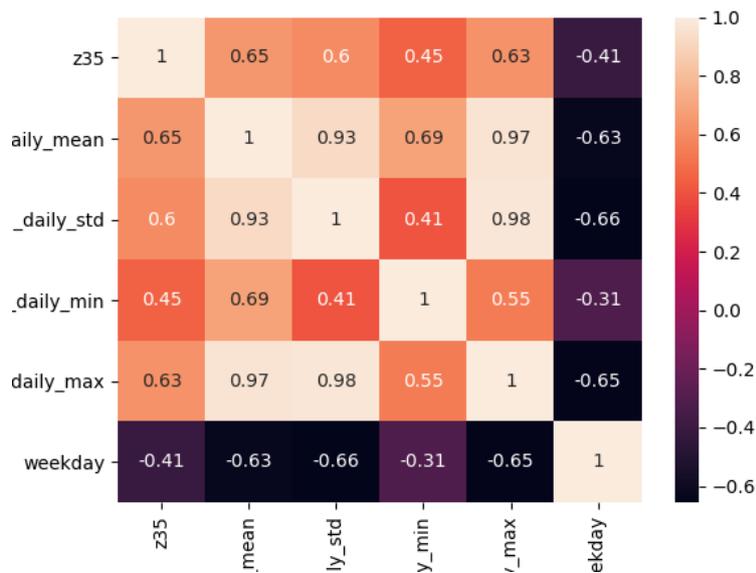


Figure 58. Correlation between features

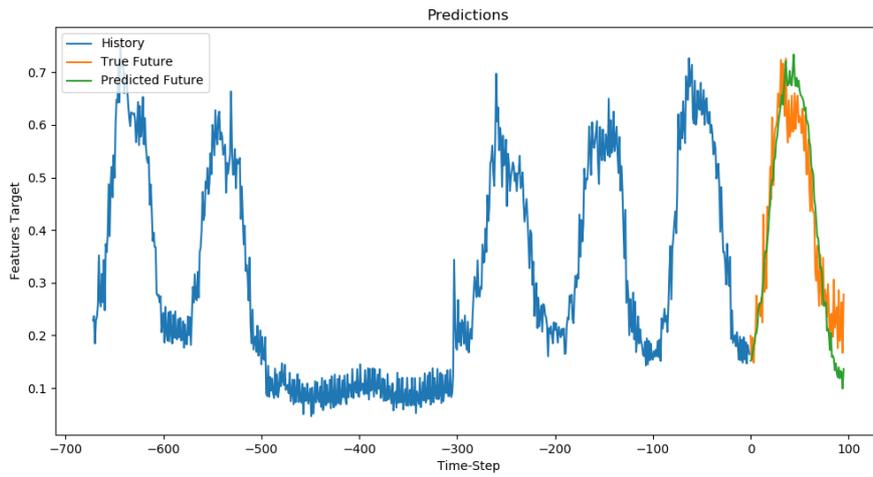


Figure 59. Example of one RNN model prediction

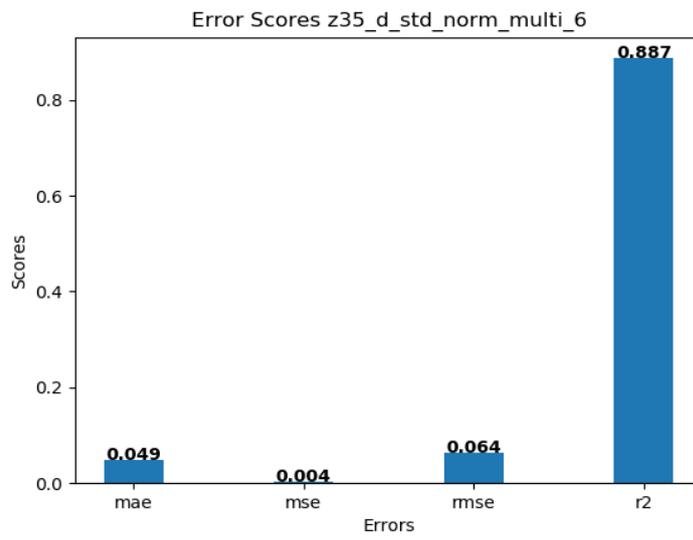


Figure 60. Error metrics score calculated on all batches of test set

3) Model name: "z35_6h_var_norm_multi_6";

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

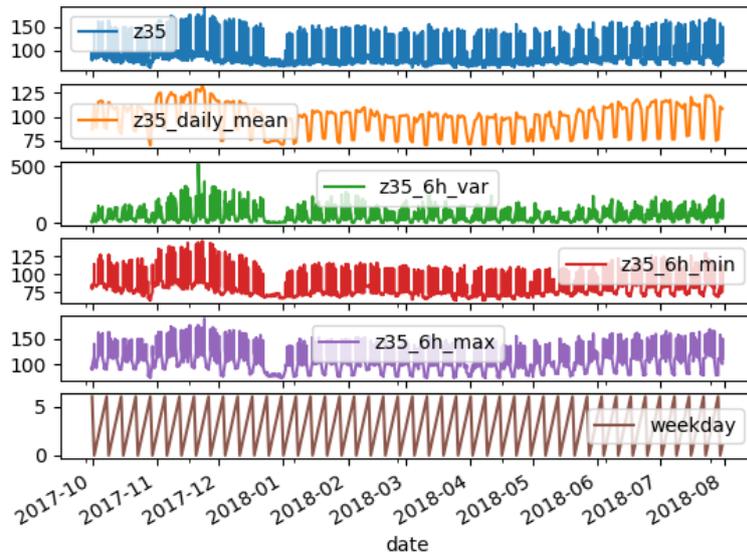


Figure 61. RNN Model Multivariate n.3 – Features used

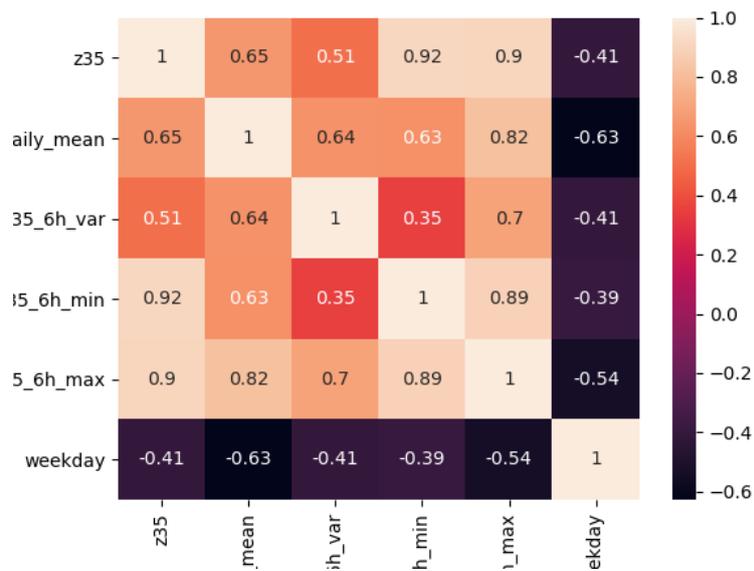


Figure 62. Correlation between features

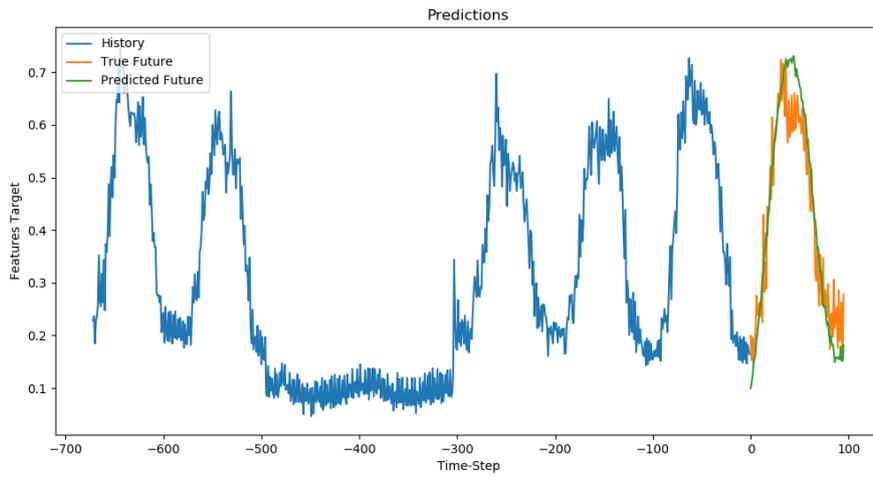


Figure 63. Example of one RNN model prediction

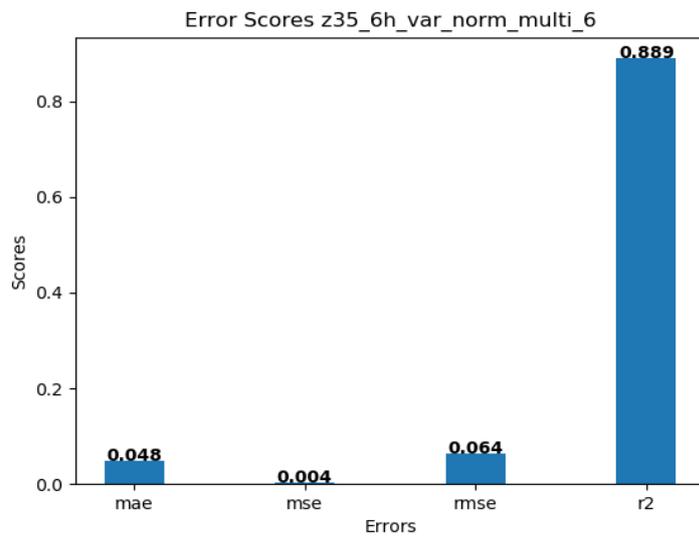


Figure 64. Error metrics score calculated on all batches of test set

4) *Model name: "z35_d_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

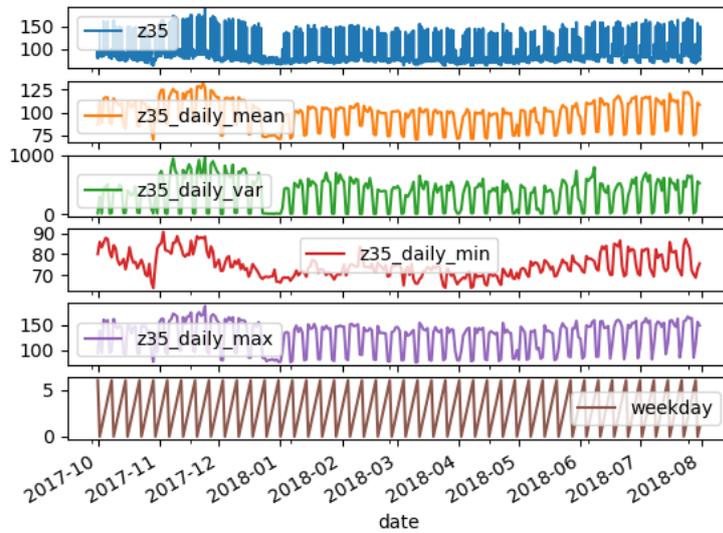


Figure 65. RNN Model Multivariate n.4 – Features used

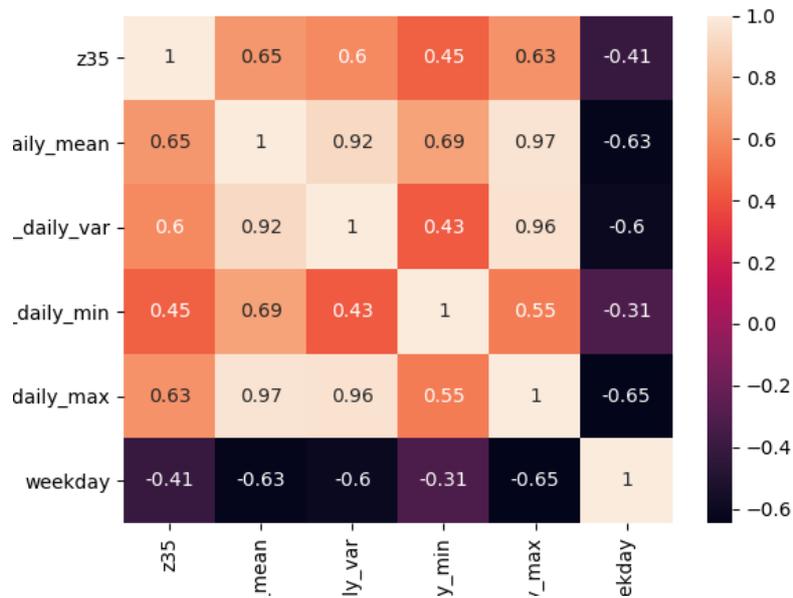


Figure 66. Correlation between features

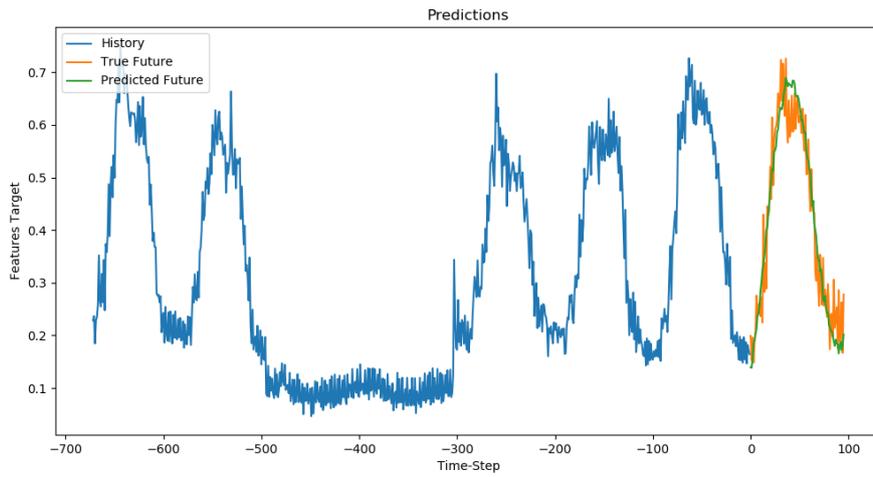


Figure 67. Example of one RNN model prediction



Figure 68. Error metrics score calculated on all batches of test set

Now we report the plot comparison between the five models tested (one Univariate and four Multivariate) of the error metrics score R2, MAE and RMSE, to verify which model obtained the best results for predicting z35 time series. We have also reported in the last two columns of each plot the results obtained with two other Multivariate models not described, where we use standardized values of the features and RMSprop as optimizer of the model with MAE as cost function to minimize during training. In first of this extra models we use four features (z35, z35_daily_mean, z35_daily_var, weekday) and in the second model we use six features (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday) with always the same structure of the layers of the model used until now. We have reported also them only to compare their results with the results obtained with the new configuration of the models using normalization as scaling and Adam as optimizer with MSE cost function, we must however consider as benchmark only the error score R2, because the errors MAE and RMSE are obviously bigger for the fact that with normalization we have only values between 0 and 1 instead of standardization where the range of values changes depending on mean and standard deviation factors.

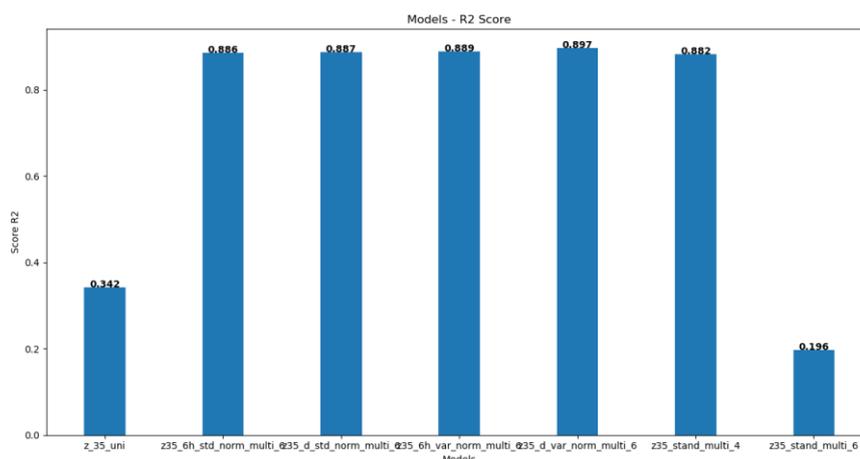


Figure 69. R2 Scores z35 models tested

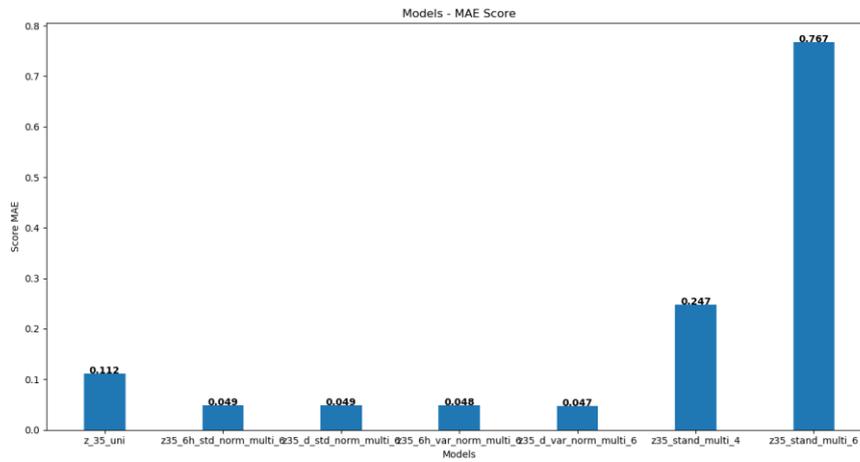


Figure 70. MAE Scores z35 models tested

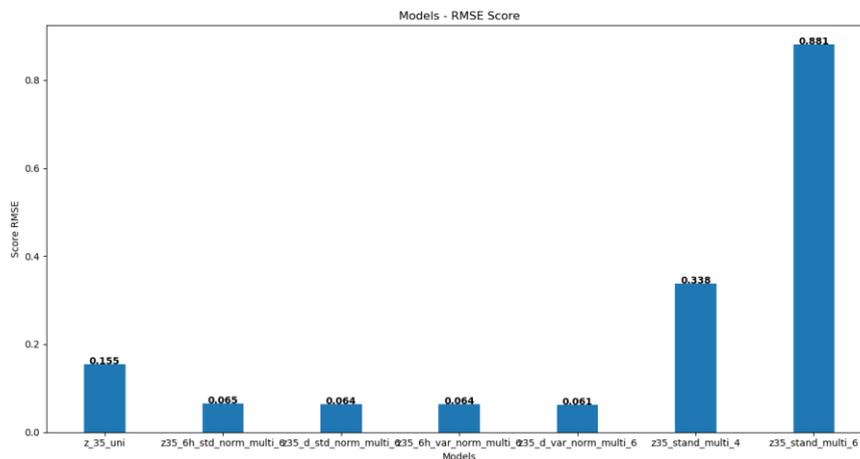


Figure 71. RMSE Scores z35 models tested

Considering figure 69 with the R2 scores and comparing the results of the two standardized Multivariate models with the four normalized Multivariate models we can observe that normalized models have better scores, so they show more accurate predictions. Considering all figures (69, 70 and 71) we can observe that Multivariate models greatly improve prediction performance comparing to the Univariate model tested. The model that show the best performance (represented by the fifth column from the left in the figures) is:

Model name: "z35_d_var_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

So we choose this model as the best RNN model for predicting z35 time series.

- Prediction of **var**

- *Univariate Models*

1) *Model name: "6h_var_uni";*

Feature used: z35_6h_var

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

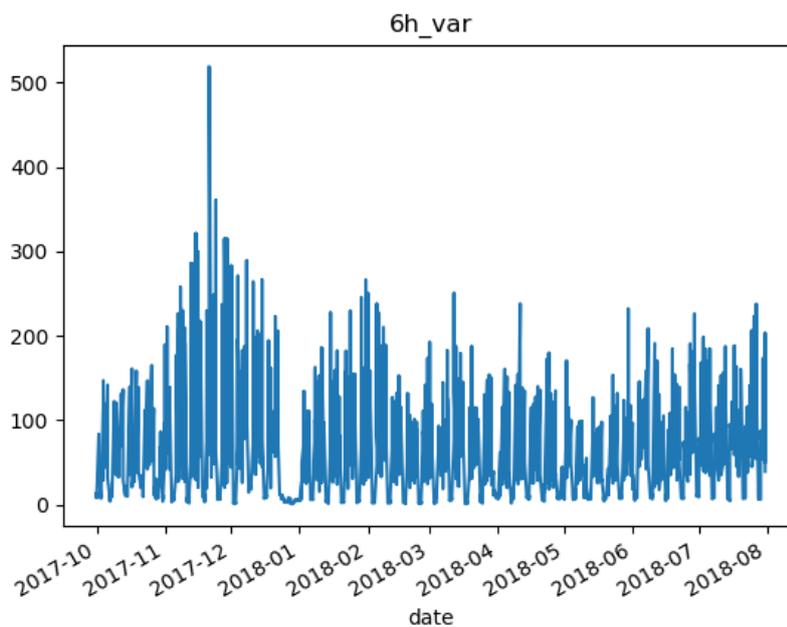


Figure 72. Time series z35_6h_var

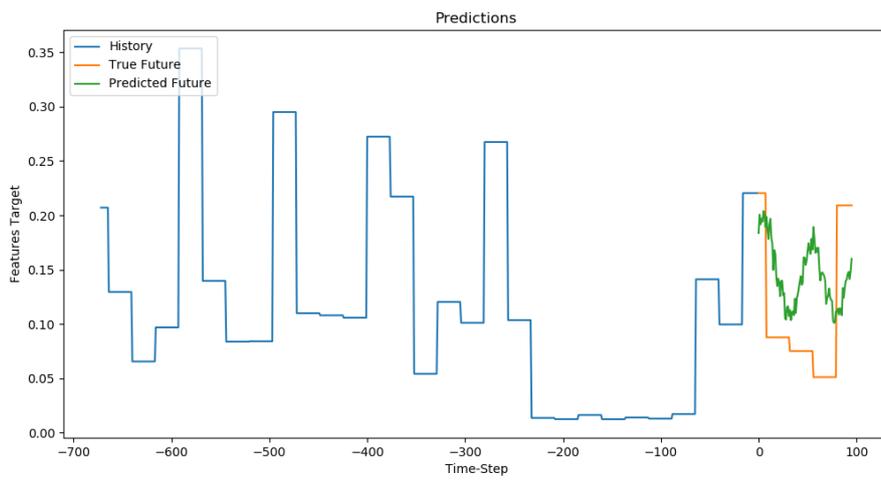


Figure 73. Example of one RNN model prediction

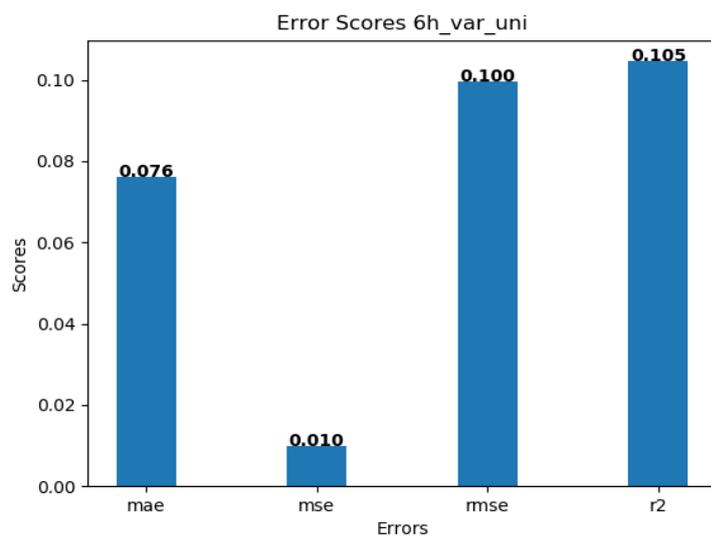


Figure 74. Error metrics score calculated on all batches of test set

2) *Model name: "d_var_uni";*

Feature used: z35_daily_var

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

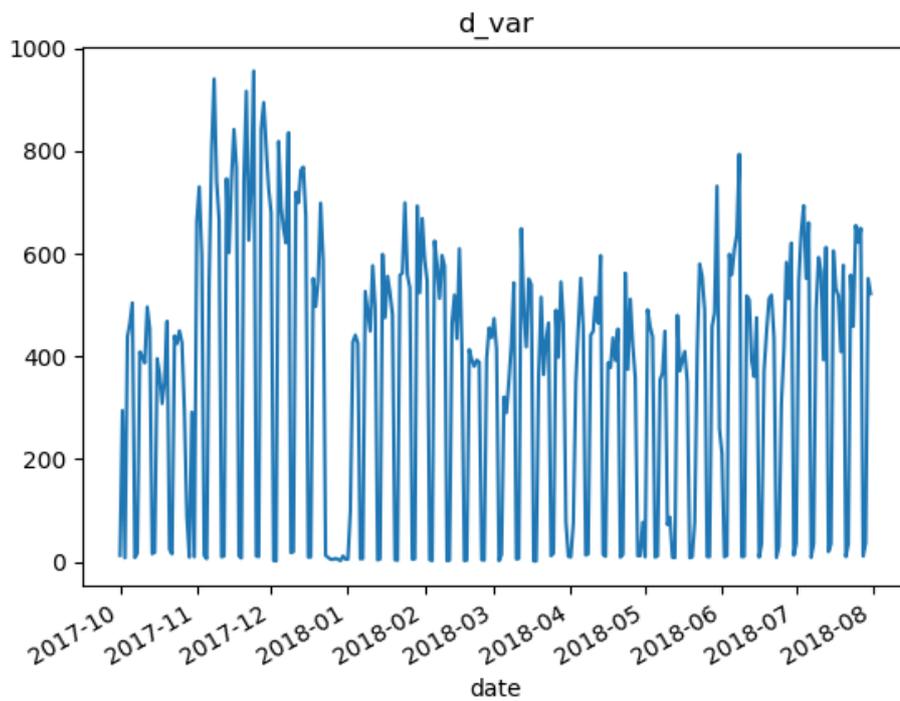


Figure 75. Time series z35_daily_var

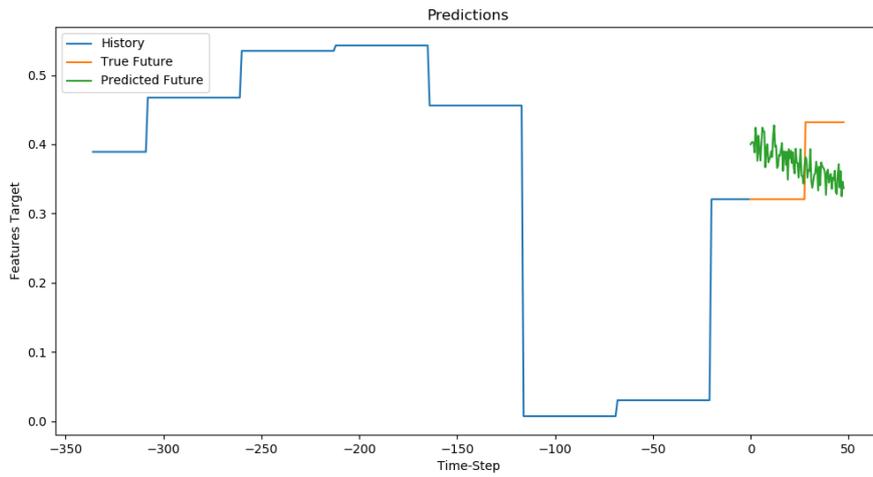


Figure 76. Example of one RNN model prediction

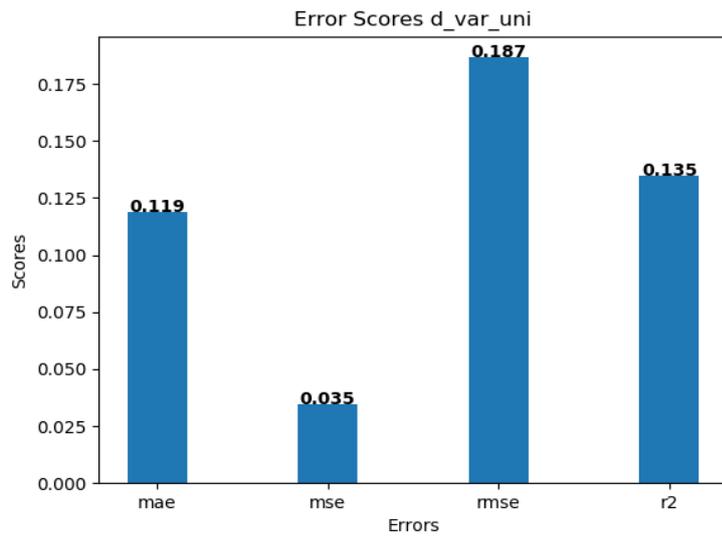


Figure 77. Error metrics score calculated on all batches of test set

- *Multivariate Models*

1) *Model name: "6h_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

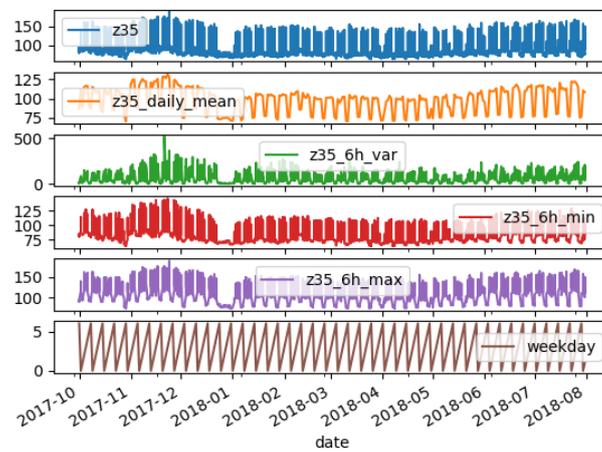


Figure 78. RNN Model Multivariate n.1 – Features used

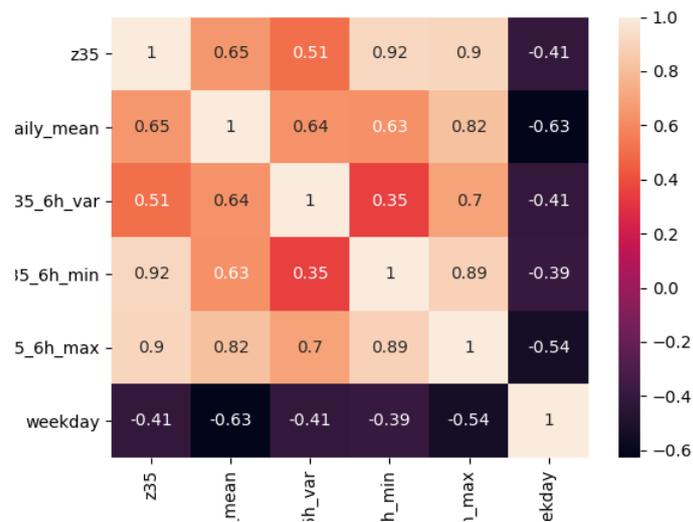


Figure 79. Correlation between features

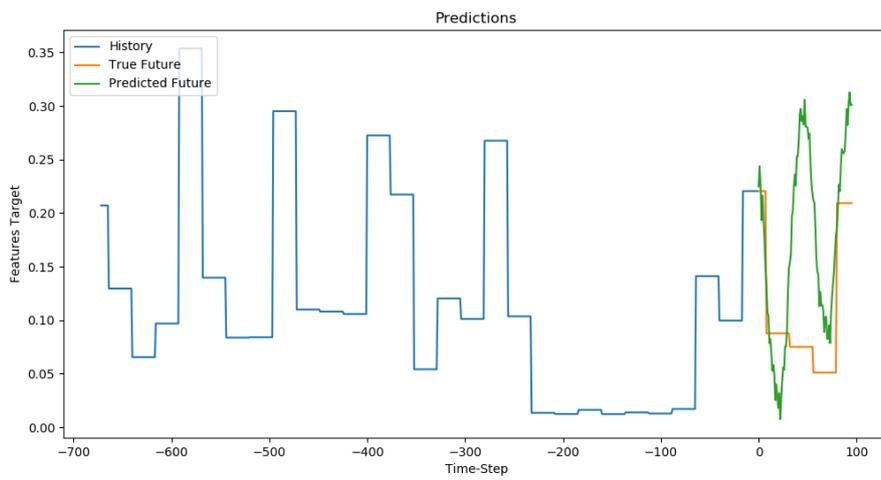


Figure 80. Example of one RNN model prediction

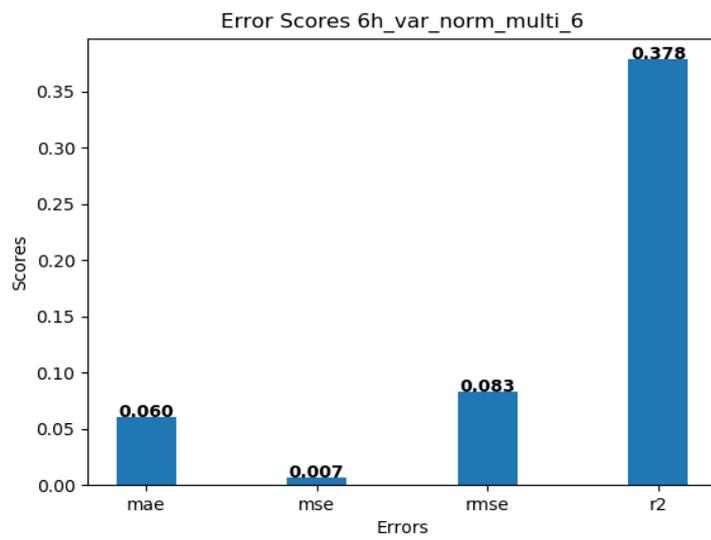


Figure 81. Error metrics score calculated on all batches of test set

2) Model name: "d_var_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

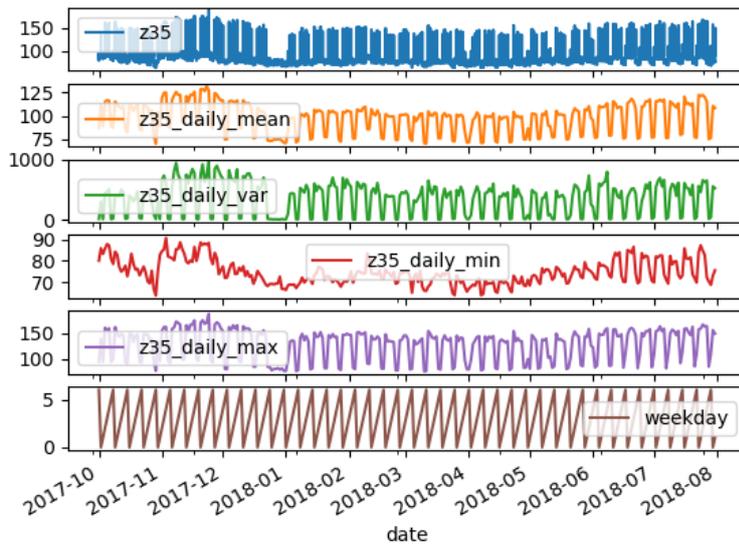


Figure 82. RNN Model Multivariate n.2 – Features used

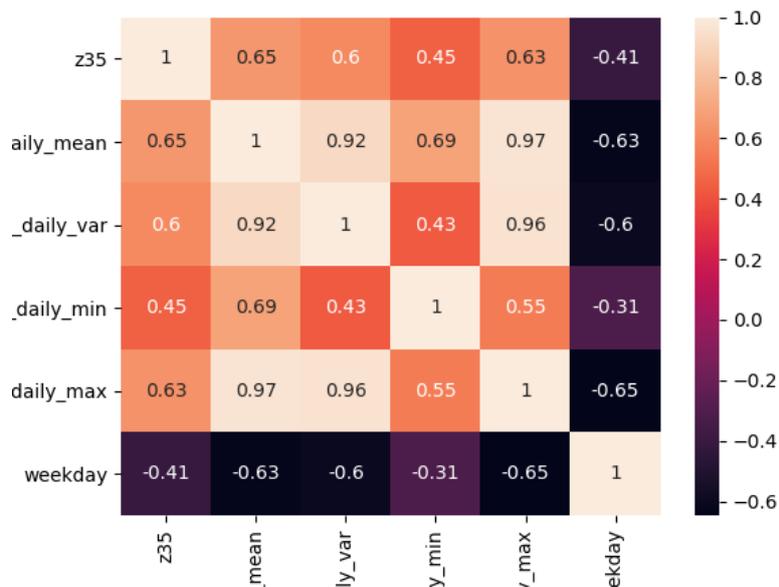


Figure 83. Correlation between features

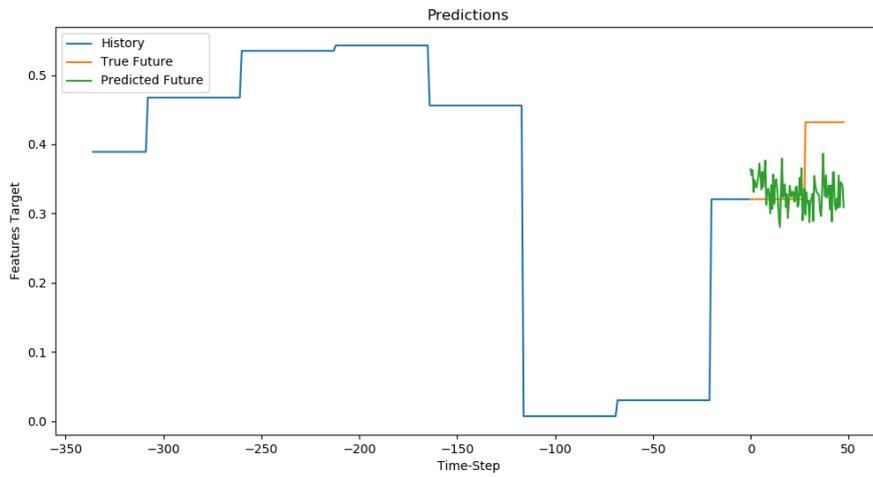


Figure 84. Example of one RNN model prediction

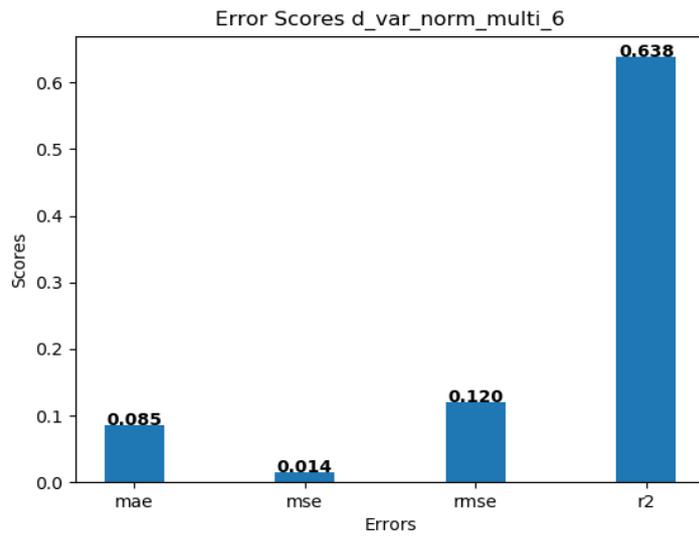


Figure 85. Error metrics score calculated on all batches of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between the four models tested (two Univariate and two Multivariate), to verify which model obtained the best results for predicting z35 variance time series. Also here we have reported in the last two columns of each plot the results obtained with two other Multivariate models not described, where we use standardized values of the features and RMSprop as optimizer of the model with MAE cost function for predicting z35 variance. In first of this extra models we use four features (z35, z35_daily_mean, z35_daily_var, weekday) and in the second model we use six features (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday) with always the same structure of the layers of the model. We have to consider as benchmark only the error score R2 for the same reason illustrated for z35 models about comparing normalized and standardized values.

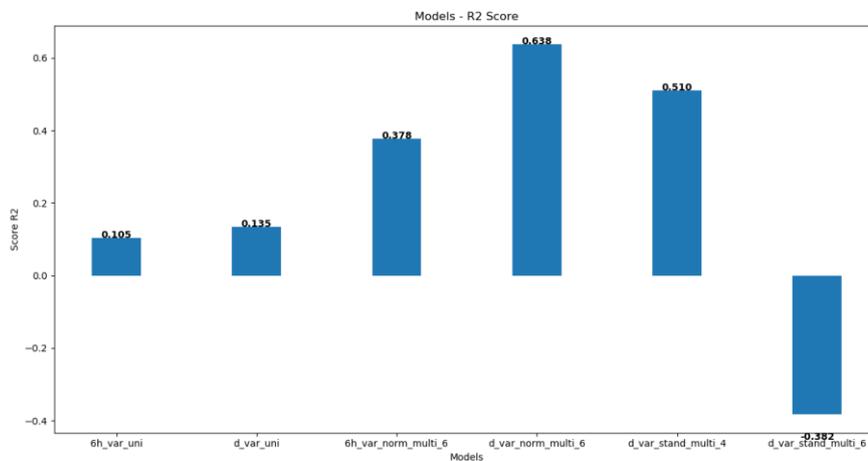


Figure 86. R2 Scores z35 variance models tested

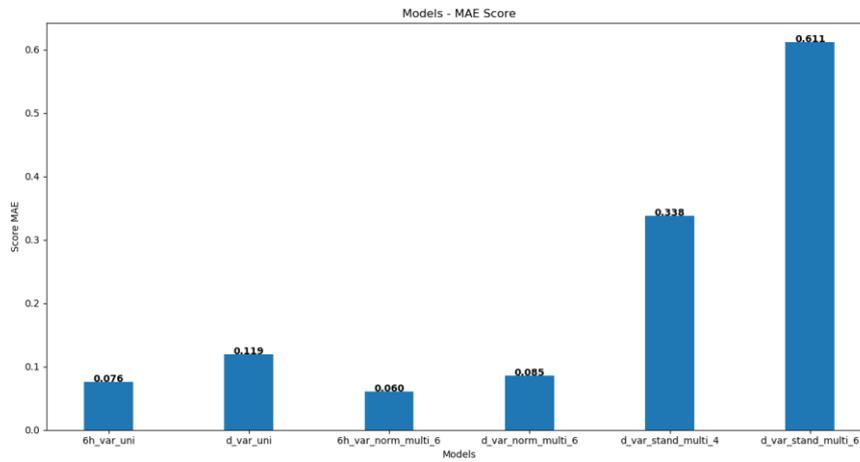


Figure 87. MAE Scores z35 variance models tested

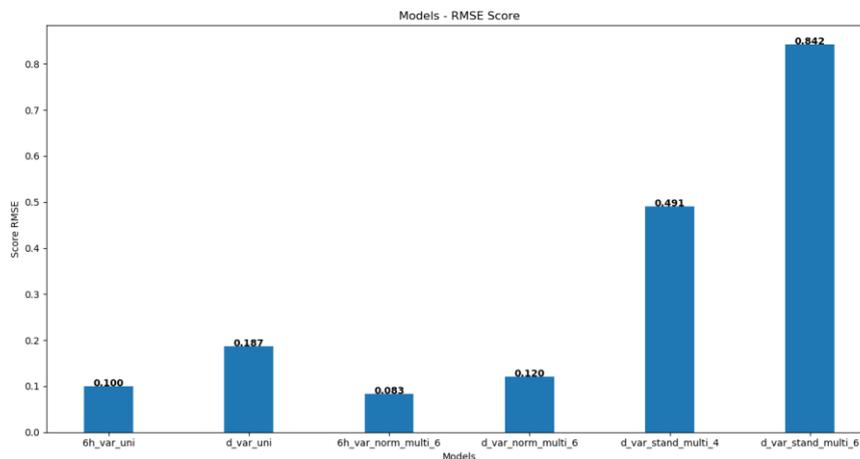


Figure 88. RMSE Scores z35 variance models tested

Also in this comparison we can observe the same results obtained in the comparison made between z35 models. For R2 score, normalized models have better performance than standardized models. Considering all the error scores and the same interval of time for z35 variance time series (one day or six hours), RNN Multivariate models can predict z35 variance values in more accurate way than Univariate models. The model that show the best performance (represented by the third column from the left in the figures) is:

Model name: "6h_var_norm_multi_6";

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

So we choose this model as the best RNN model for predicting z35 variance time series.

-
- Prediction of **std**
 - *Univariate Models*
-

1) *Model name: "6h_std_uni";*

Feature used: z35_6h_std

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

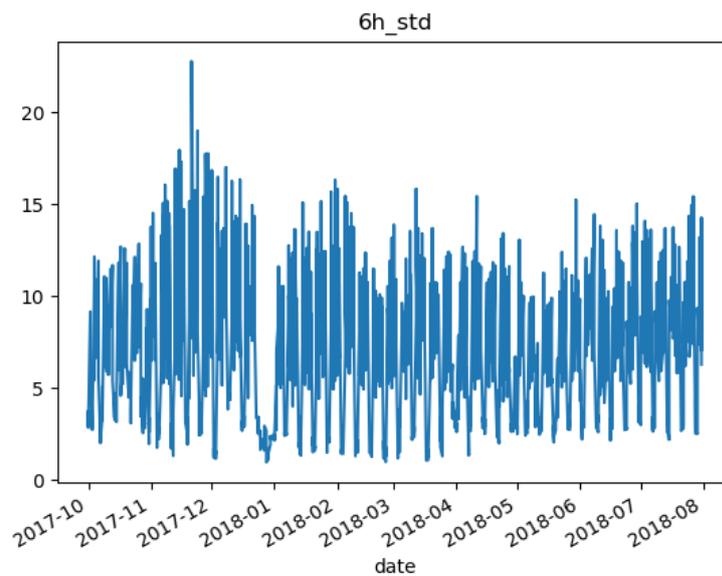


Figure 89. Time series z35_6h_std

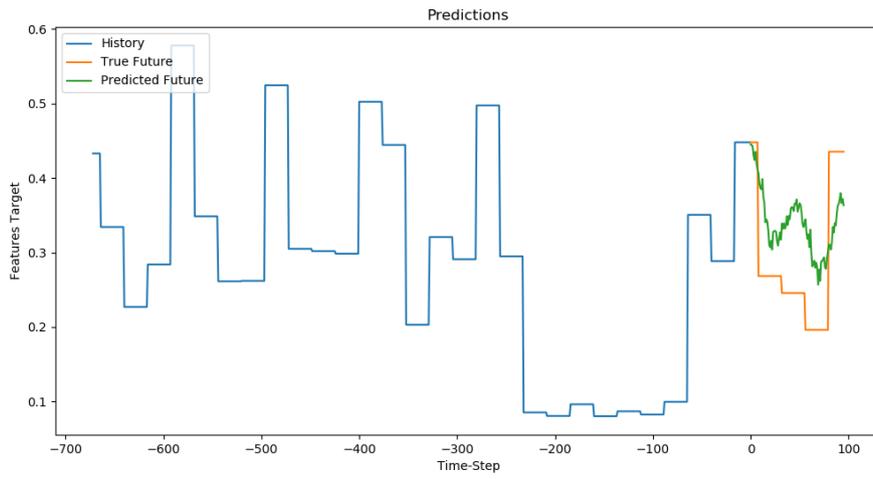


Figure 90. Example of one RNN model prediction

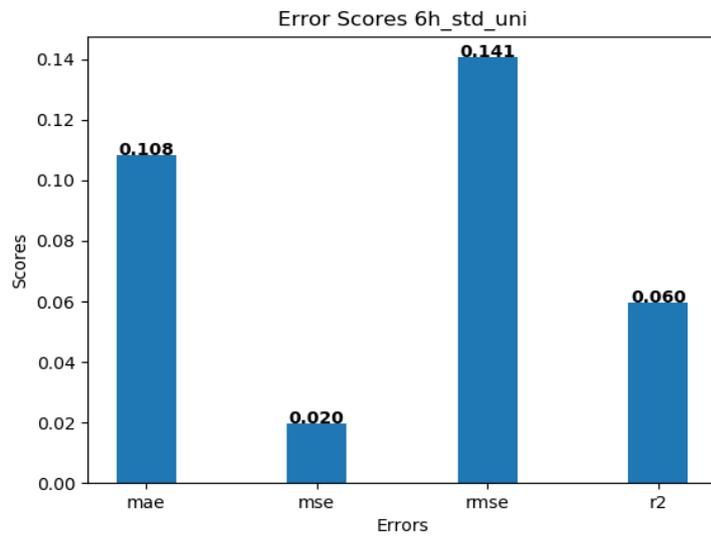


Figure 91. Error metrics score calculated on all batches of test set

2) Model name: "d_std_uni";

Feature used: z35_daily_std

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

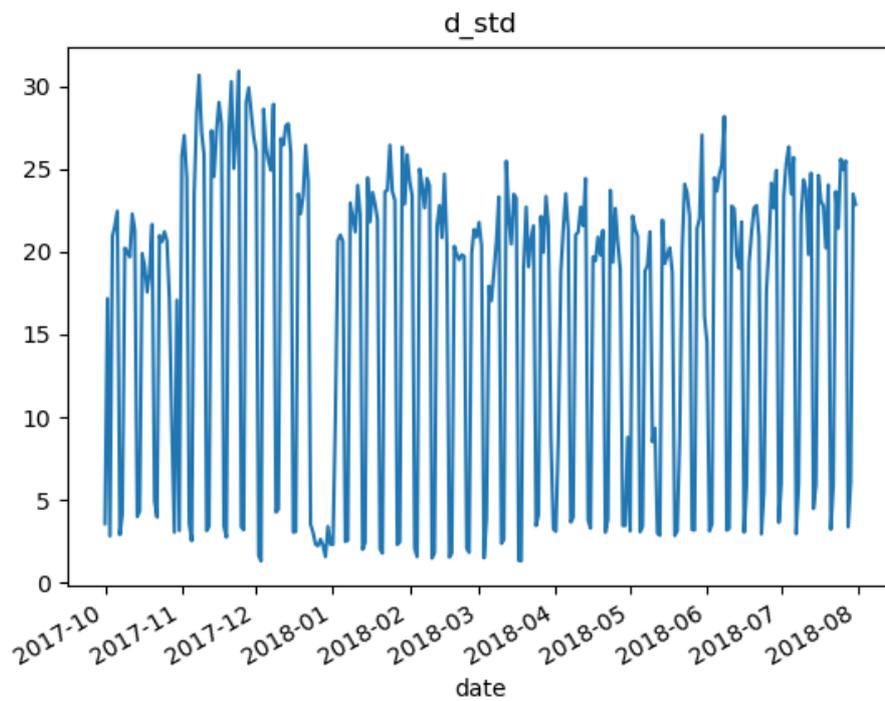


Figure 92. Time series z35_daily_std

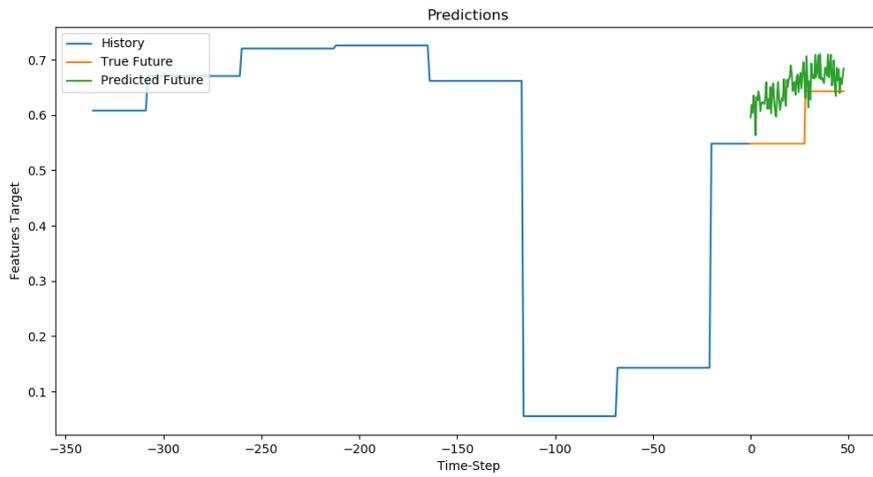


Figure 93. Example of one RNN model prediction

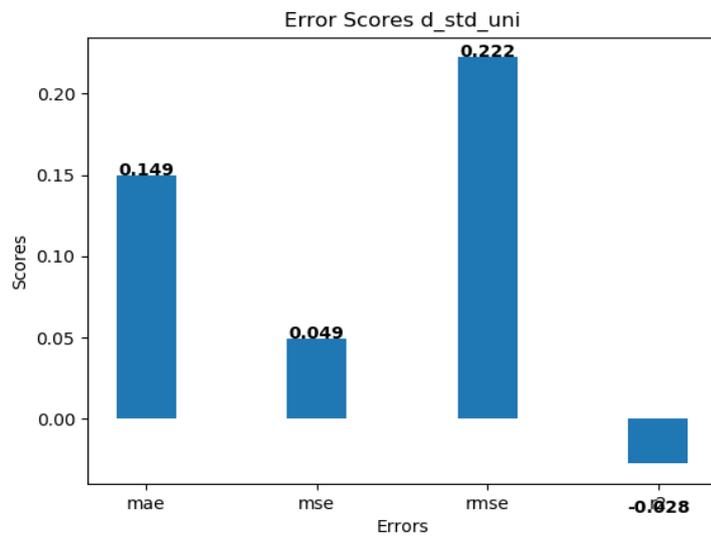


Figure 94. Error metrics score calculated on all batches of test set

- *Multivariate Models*

1) *Model name: "6h_std_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

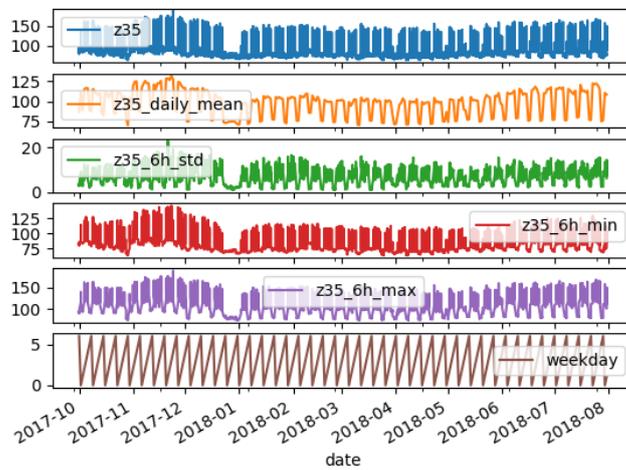


Figure 95. RNN Model Multivariate n.1 – Features used

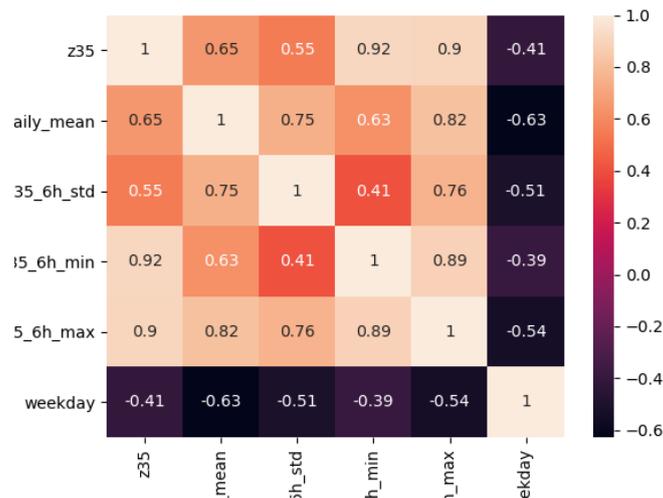


Figure 96. Correlation between features

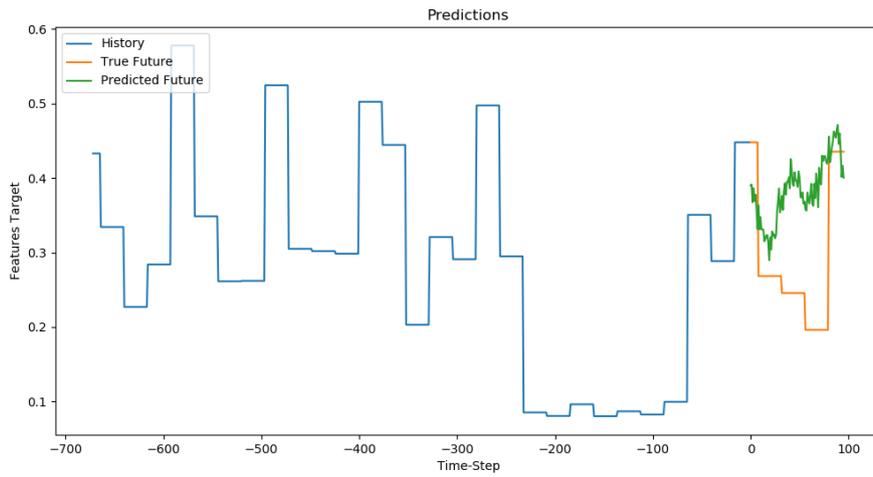


Figure 97. Example of one RNN model prediction

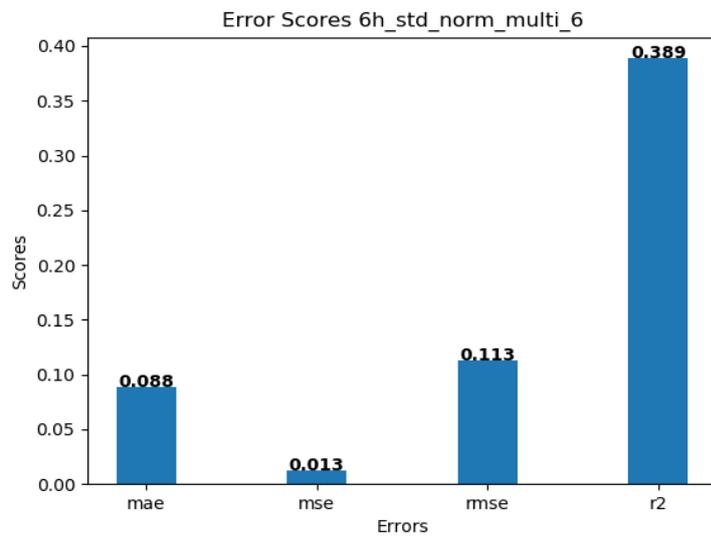


Figure 98. Error metrics score calculated on all batches of test set

2) Model name: "d_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

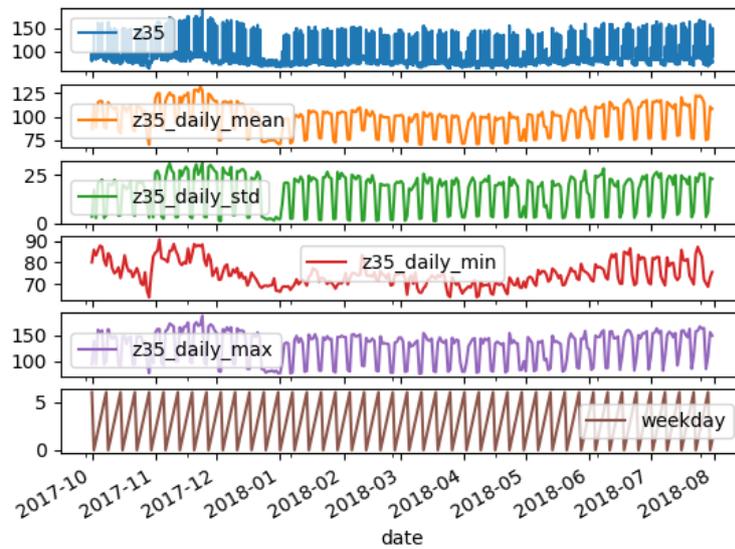


Figure 99. RNN Model Multivariate n.1 – Features used

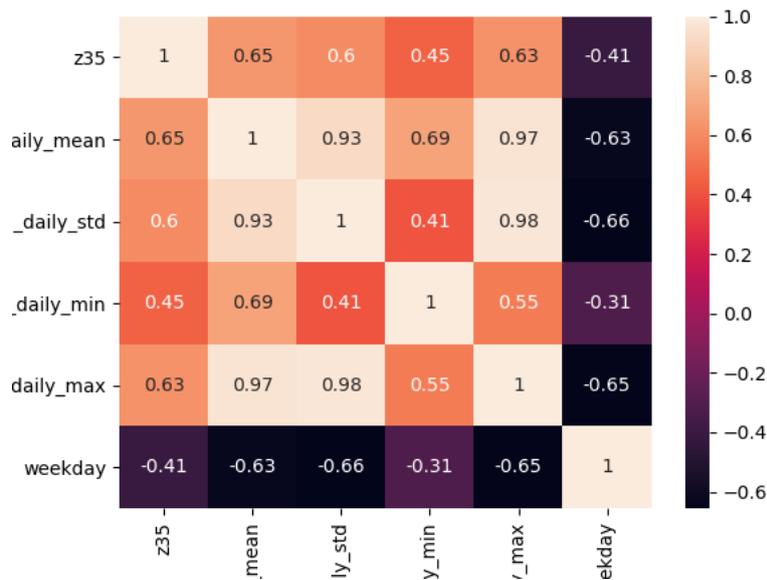


Figure 100. Correlation between features

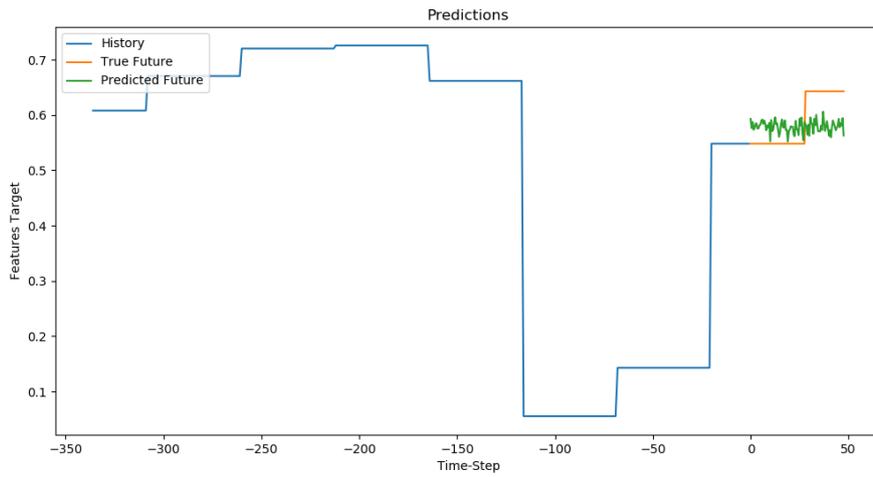


Figure 101. Example of one RNN model prediction

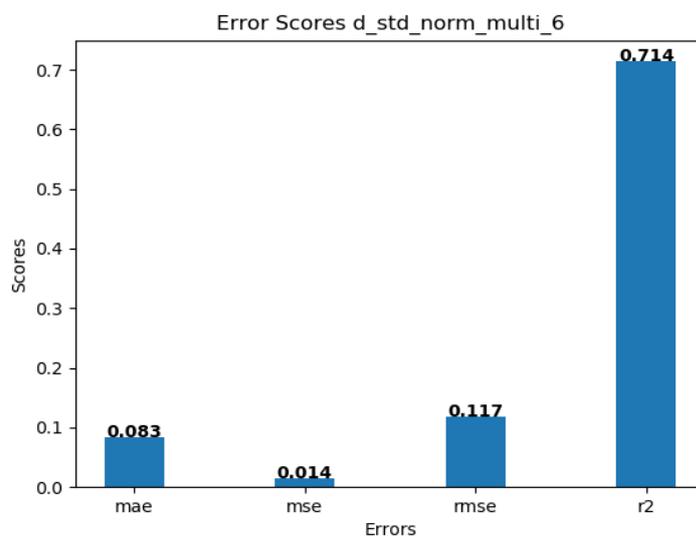


Figure 102. Error metrics score calculated on all batches of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between the four models tested (two Univariate and two Multivariate), to verify which model obtained the best results for predicting z35 standard deviation time series.

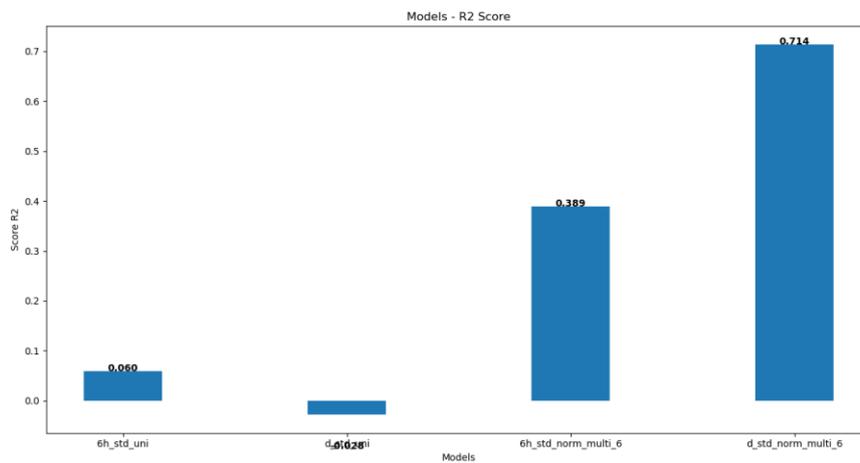


Figure 103. R2 Scores z35 standard deviation models tested

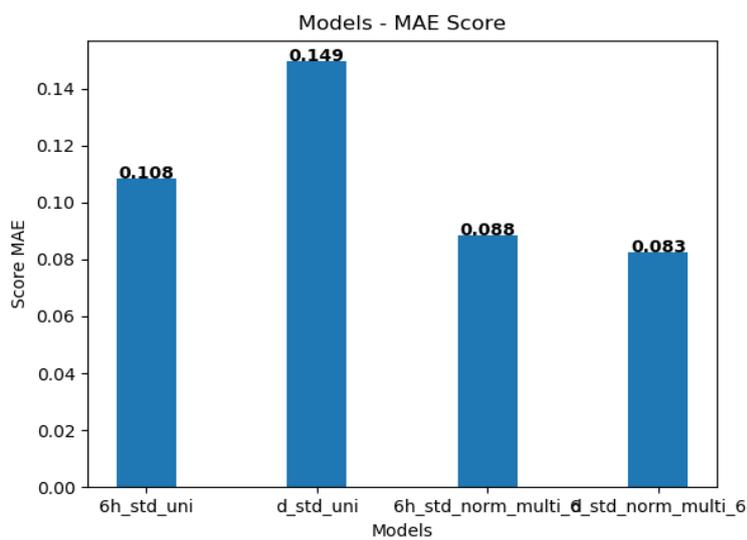


Figure 104. MAE Scores z35 standard deviation models tested

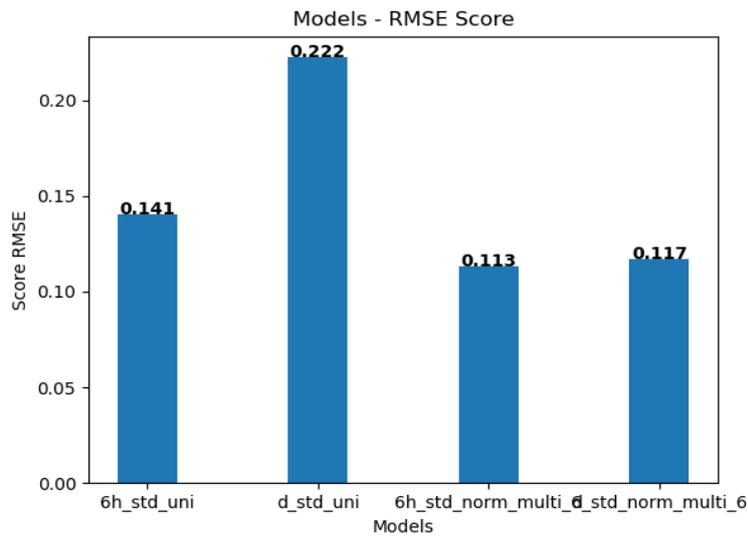


Figure 105. RMSE Scores z35 standard deviation models tested

Considering figures 103, 104 and 105, we can observe that also for predicting the standard deviation of z35, the models in Multivariate case have better prediction performance than the models in Univariate case. The model that show the best performance based on the results of the three error metrics score (represented by the last column from the left in the figures) is:

Model name: "d_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

So we choose this model as the best RNN model for predicting z35 standard deviation time series.



- Prediction of **min**
 - *Univariate Models*
-

1) *Model name: "6h_min_uni";*

Feature used: z35_6h_min

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

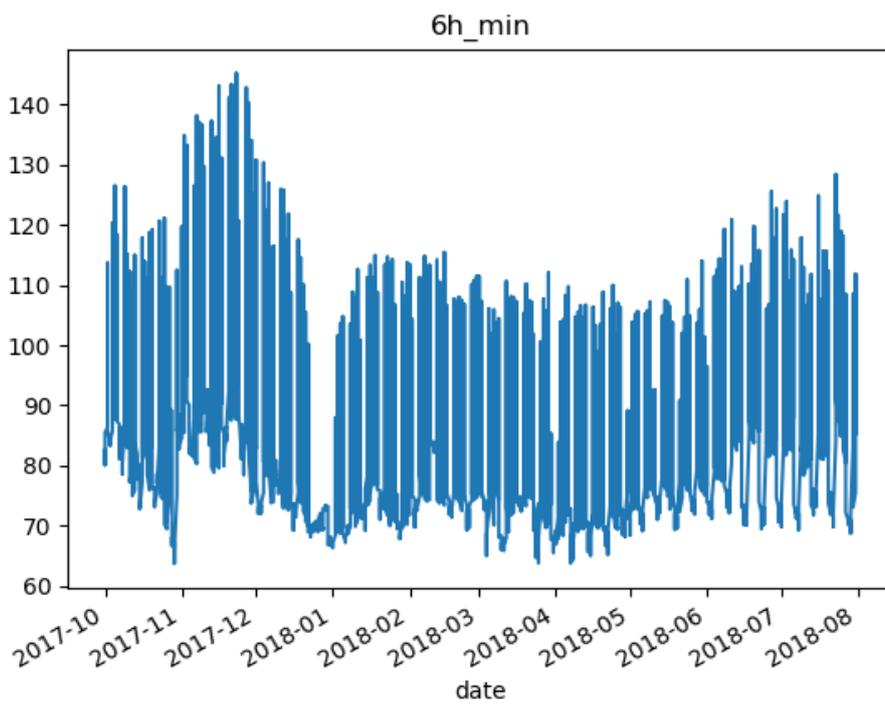


Figure 106. Time series z35_6h_min

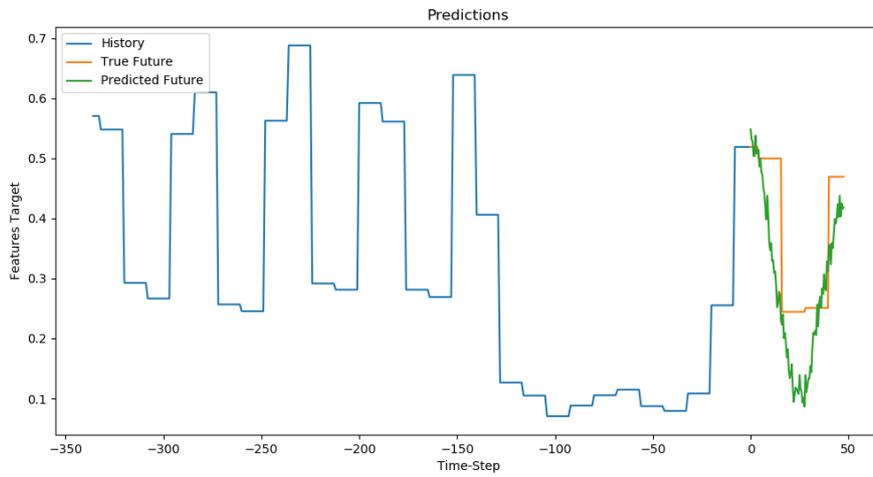


Figure 107. Example of one RNN model prediction

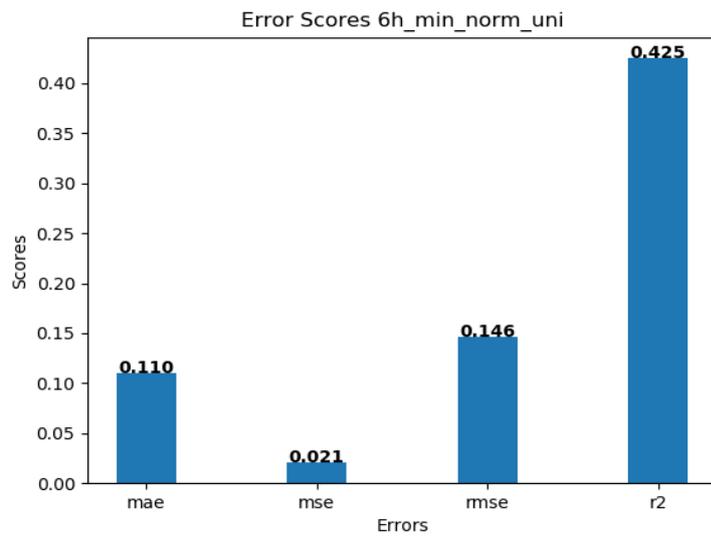


Figure 108. Error metrics score calculated on all batches of test set

2) Model name: "d_min_uni";

Feature used: z35_daily_min

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

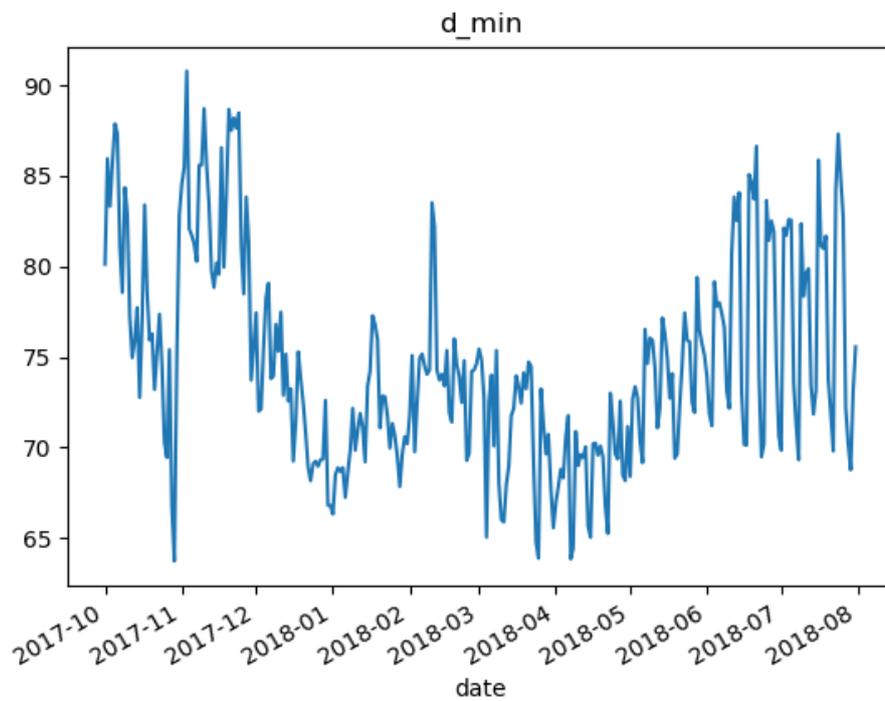


Figure 109. Time series z35_daily_min

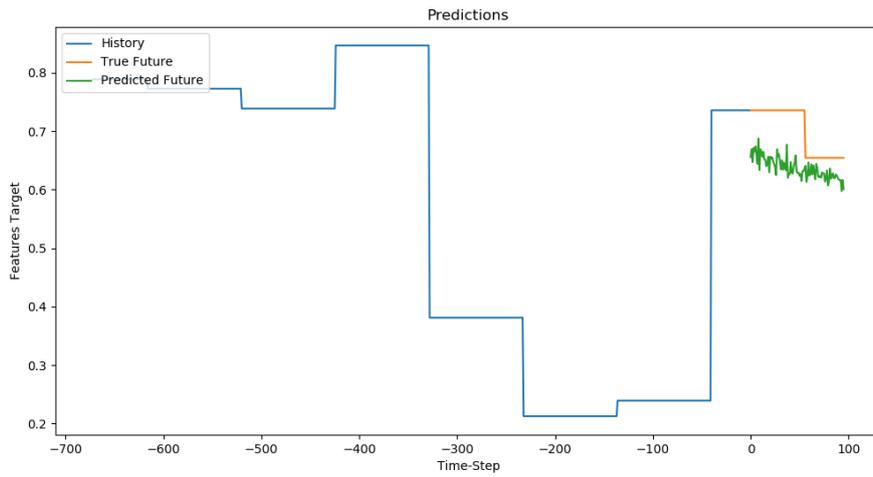


Figure 110. Example of one RNN model prediction

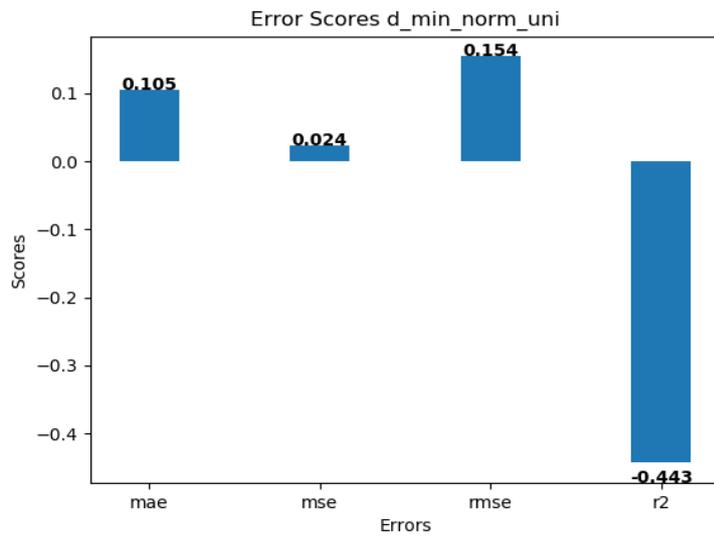


Figure 111. Error metrics score calculated on all batches of test set

- *Multivariate Models*

1) *Model name: "6h_min_6h_std_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

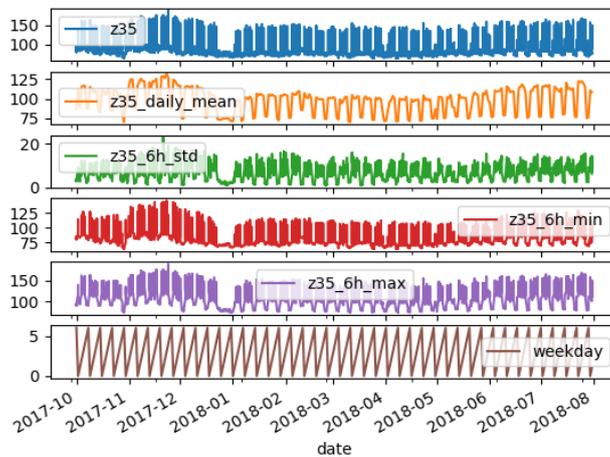


Figure 112. RNN Model Multivariate n.1 – Features used

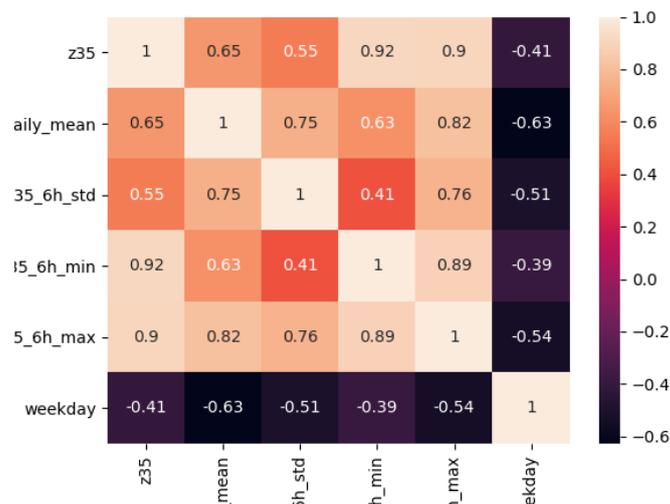


Figure 113. Correlation between features

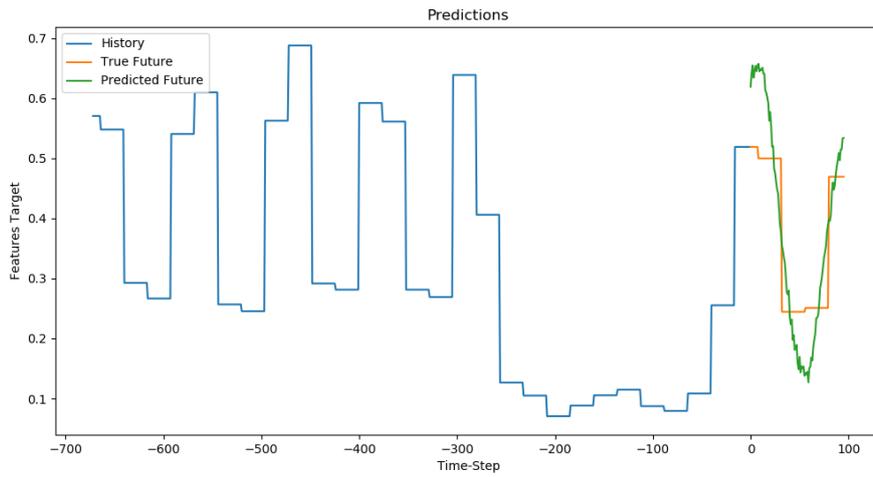


Figure 114. Example of one RNN model prediction

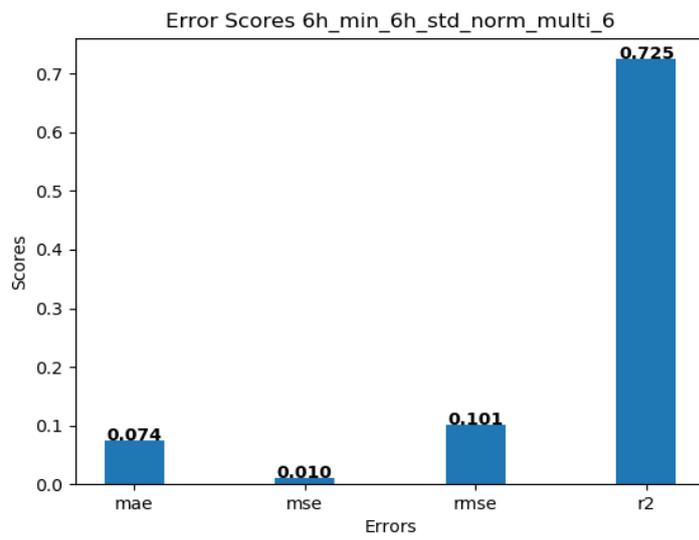


Figure 115. Error metrics score calculated on all batches of test set

2) Model name: "6h_min_d_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

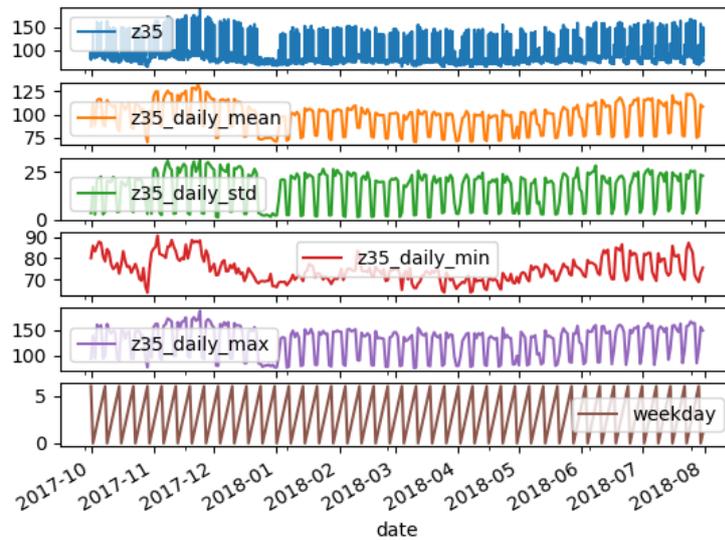


Figure 116. RNN Model Multivariate n.2 – Features used

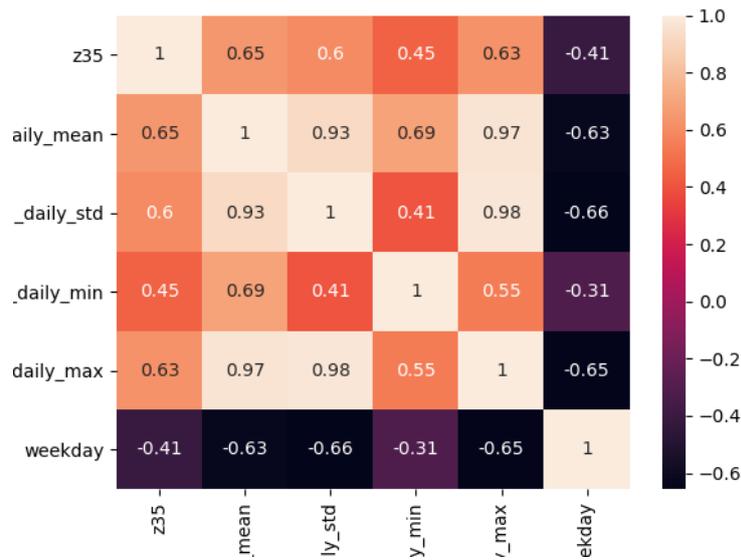


Figure 117. Correlation between features

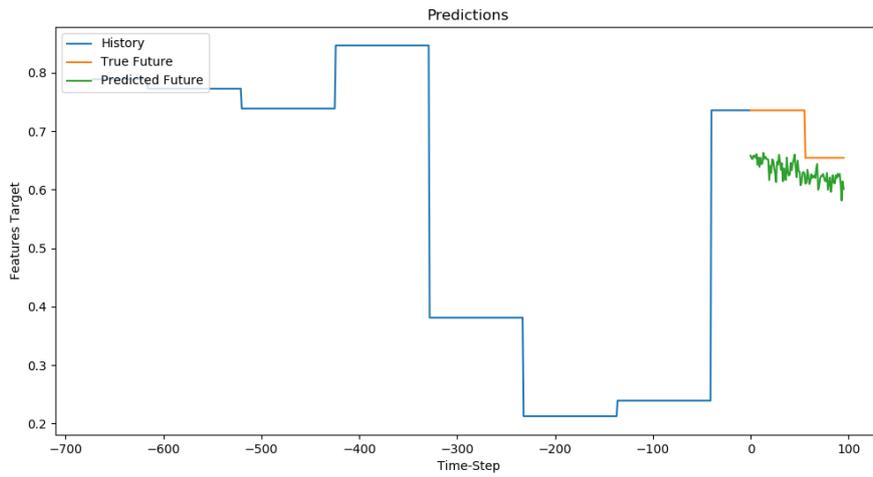


Figure 118. Example of one RNN model prediction

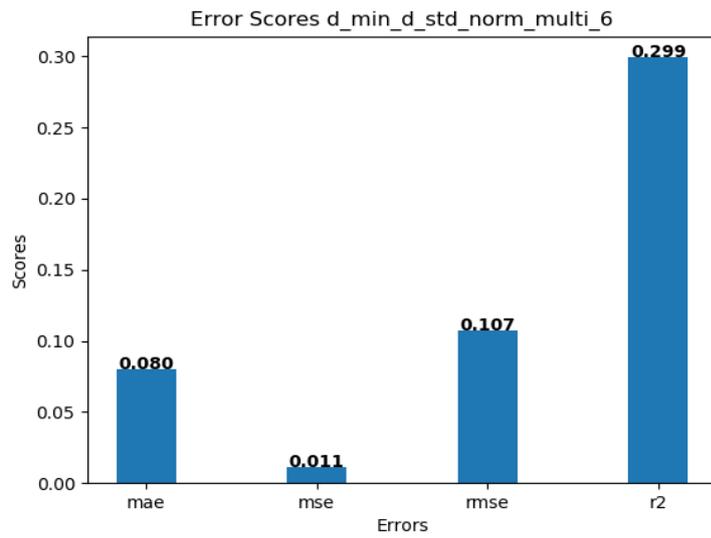


Figure 119. Error metrics score calculated on all batches of test set

3) *Model name: "6h_min_6h_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

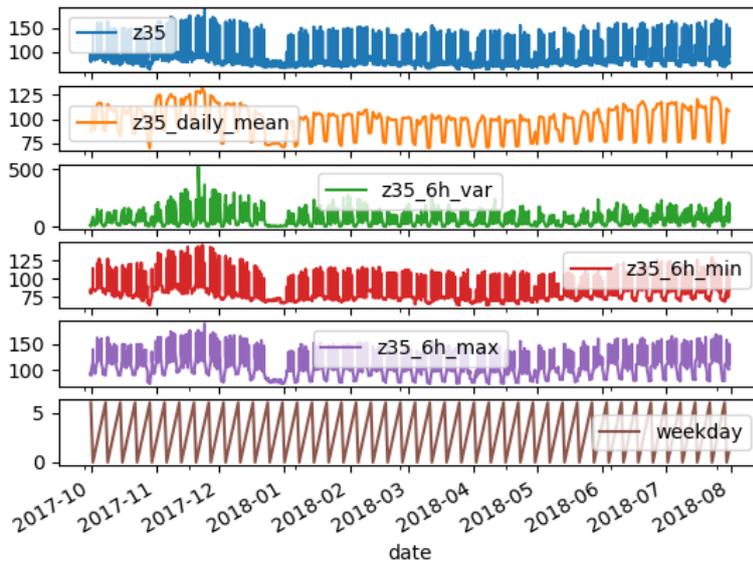


Figure 120. RNN Model Multivariate n.3 – Features used

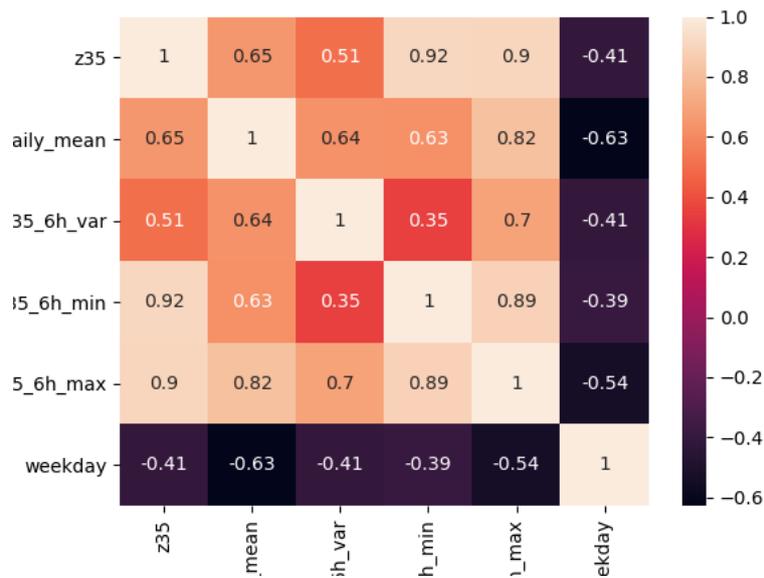


Figure 121. Correlation between features

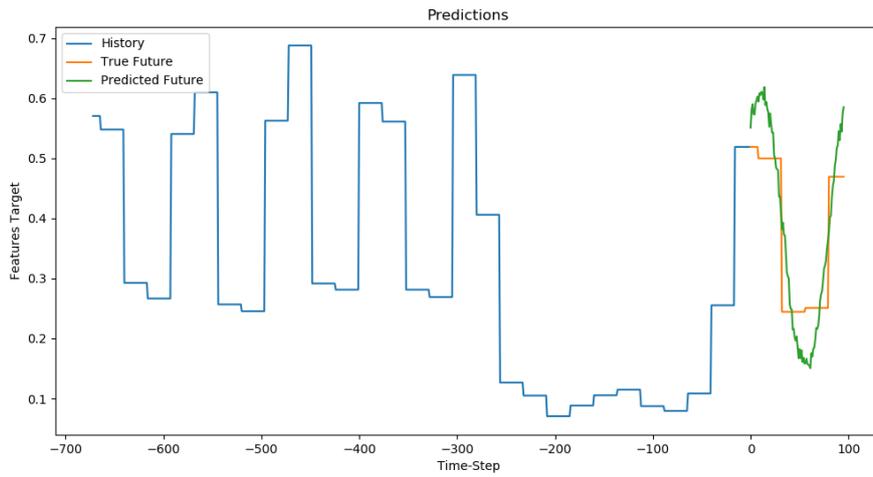


Figure 122. Example of one RNN model prediction



Figure 123. Error metrics score calculated on all batches of test set

4) *Model name: "6h_min_d_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

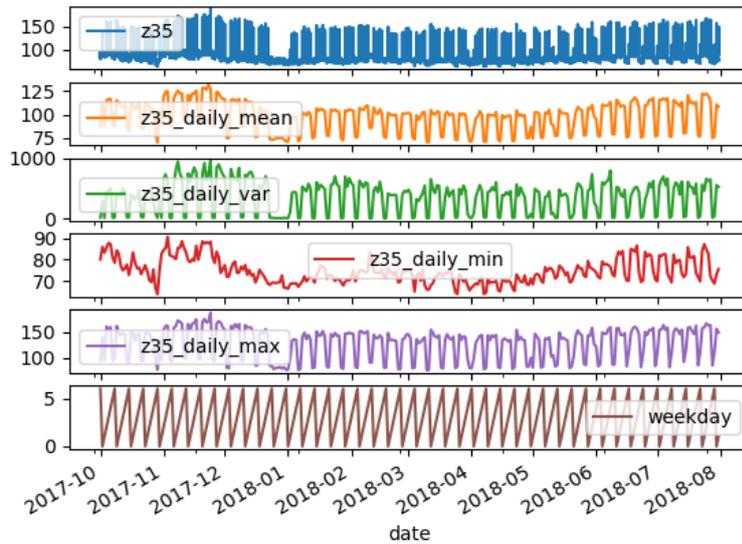


Figure 124. RNN Model Multivariate n.4 – Features used

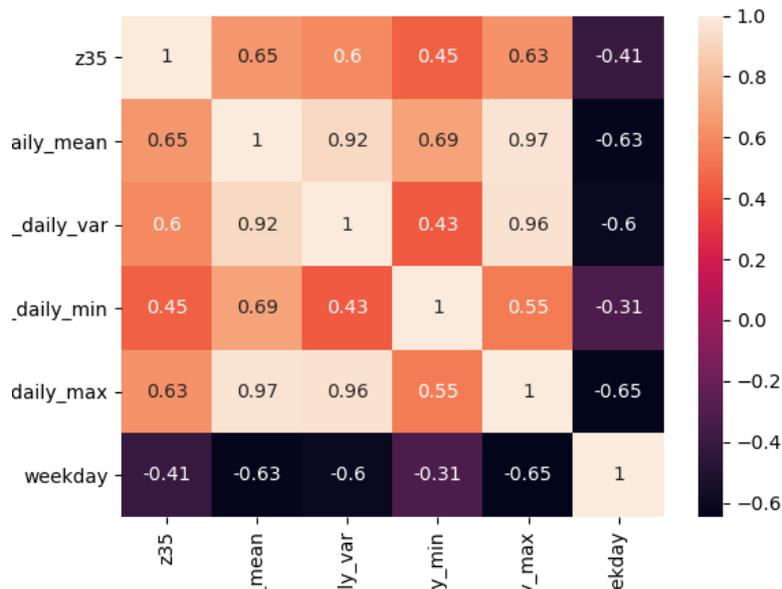


Figure 125. Correlation between features

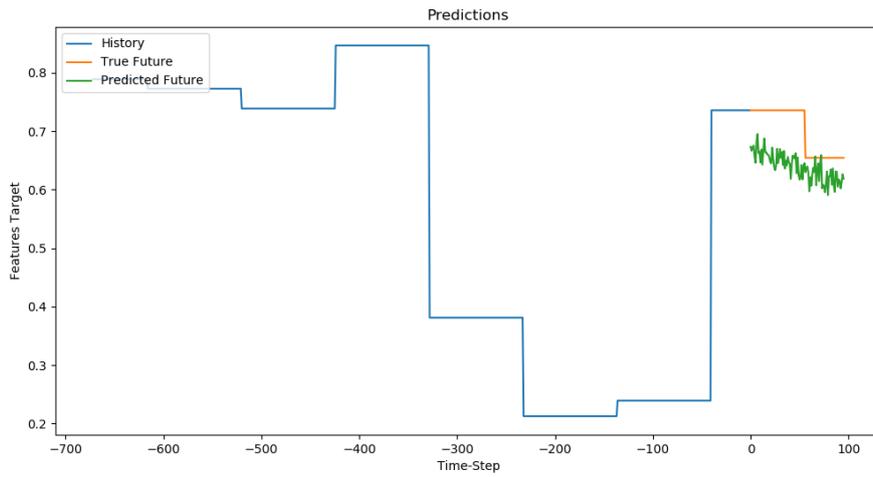


Figure 126. Example of one RNN model prediction

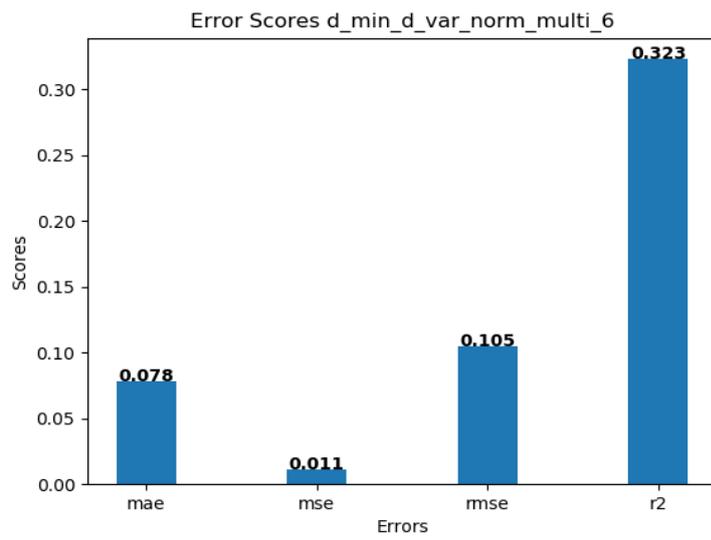


Figure 127. Error metrics score calculated on all batches of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between the six models tested (two Univariate and four Multivariate), to verify which model obtained the best results for predicting z35 minimal value time series. We have reported in the third column from the left of the following figures another Univariate model not described, where we use always normalized values of the feature z35_min but using RMSprop as optimizer, with cost function MAE to minimize during training; the structure of the layers of the model is always the same.

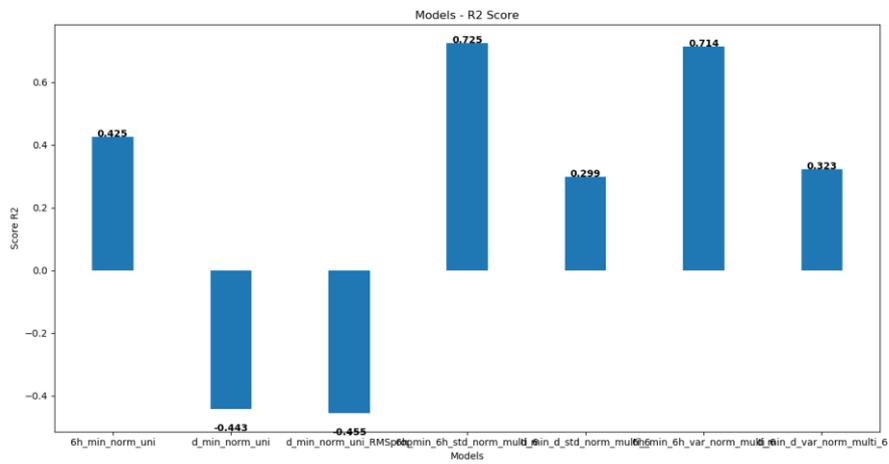


Figure 128. R2 Scores z35 minimal value models tested

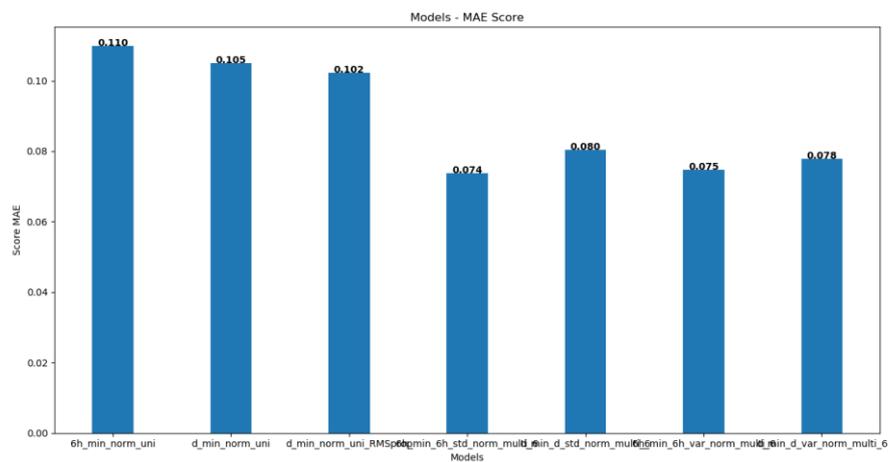


Figure 129. MAE Scores z35 minimal value models tested

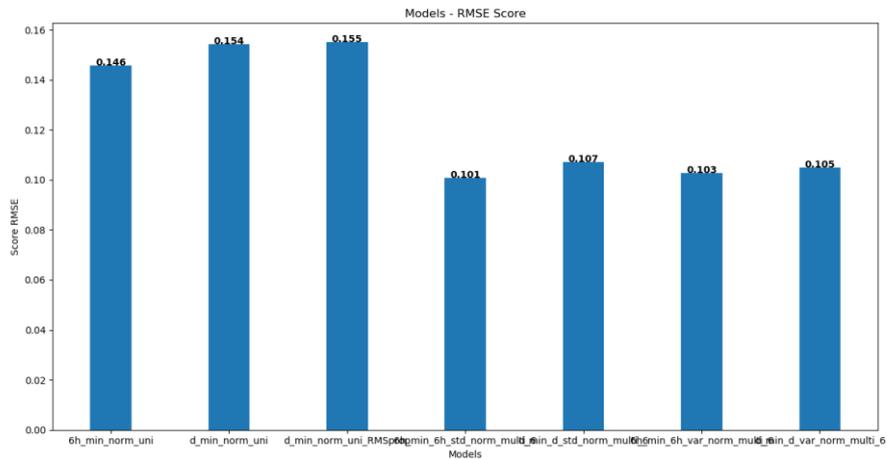


Figure 130. RMSE Scores z35 minimal value models tested

We can observe that also here, considering the same base model with the same structure of the layers, Multivariate models have better performance for predicting z35 minimal value than Univariate models, independently from which time interval we choose for the feature of interest (six hours or one day). Based on time interval, for predicting z35_min values we have the better performance using six hours than one day. The model that show the best performance based on the results of the three error metrics score (represented by the fourth column from the left in the figures) is:

Model name: "6h_min_6h_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

So we choose this model as the best RNN model for predicting z35 minimal values time series.

- Prediction of **max**
 - *Univariate Models*
-

1) *Model name: "6h_max_uni";*

Feature used: z35_6h_max

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

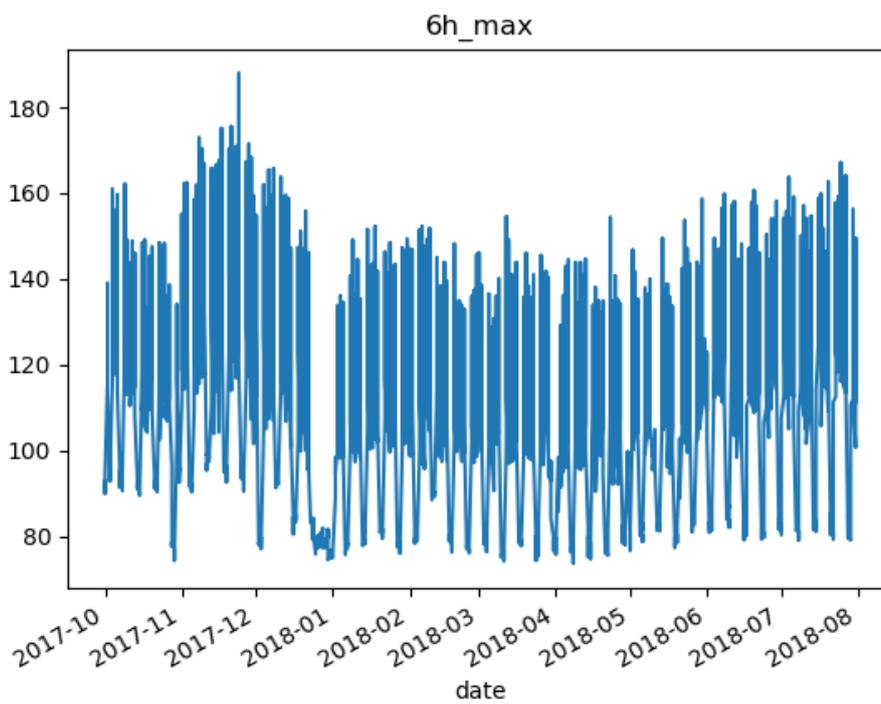


Figure 131. Time series z35_6h_max

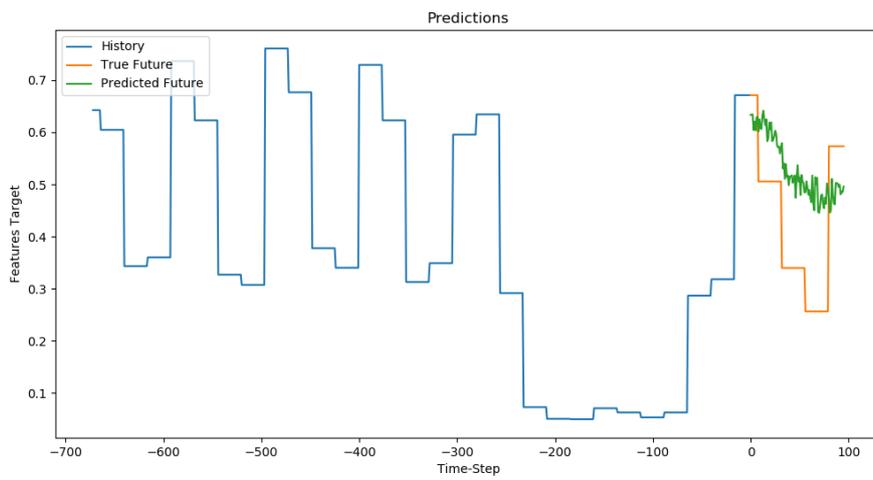


Figure 132. Example of one RNN model prediction

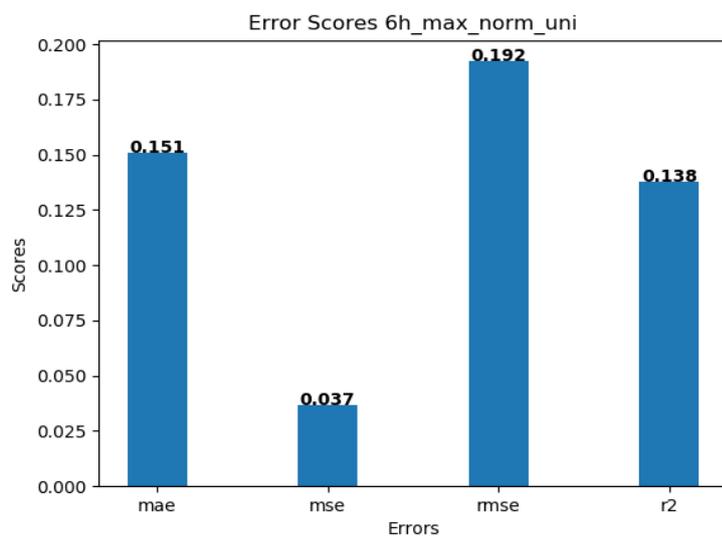


Figure 133. Error metrics score calculated on all batches of test set

2) Model name: "d_max_uni";

Feature used: z35_daily_max

- Input shape $x = (256, 672, 1)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

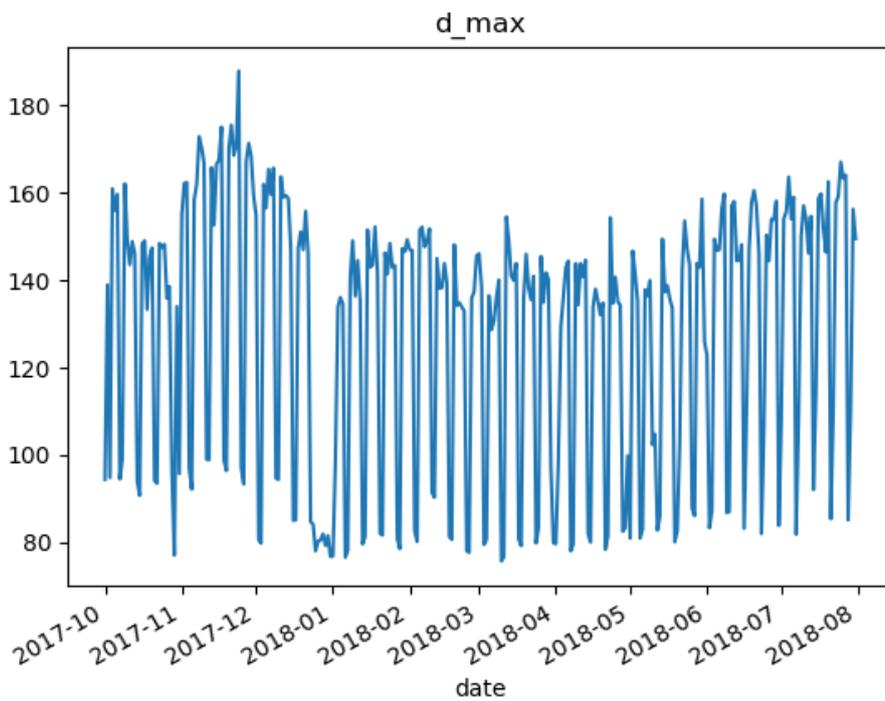


Figure 134. Time series z35_daily_max

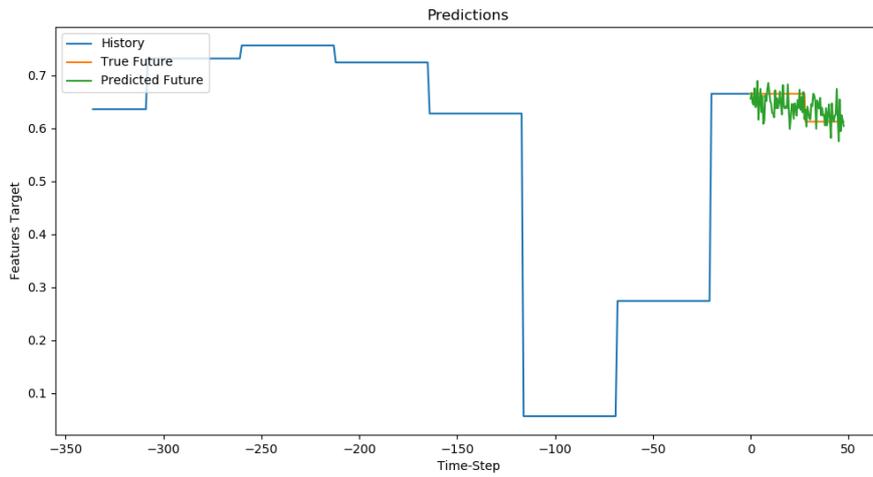


Figure 135. Example of one RNN model prediction

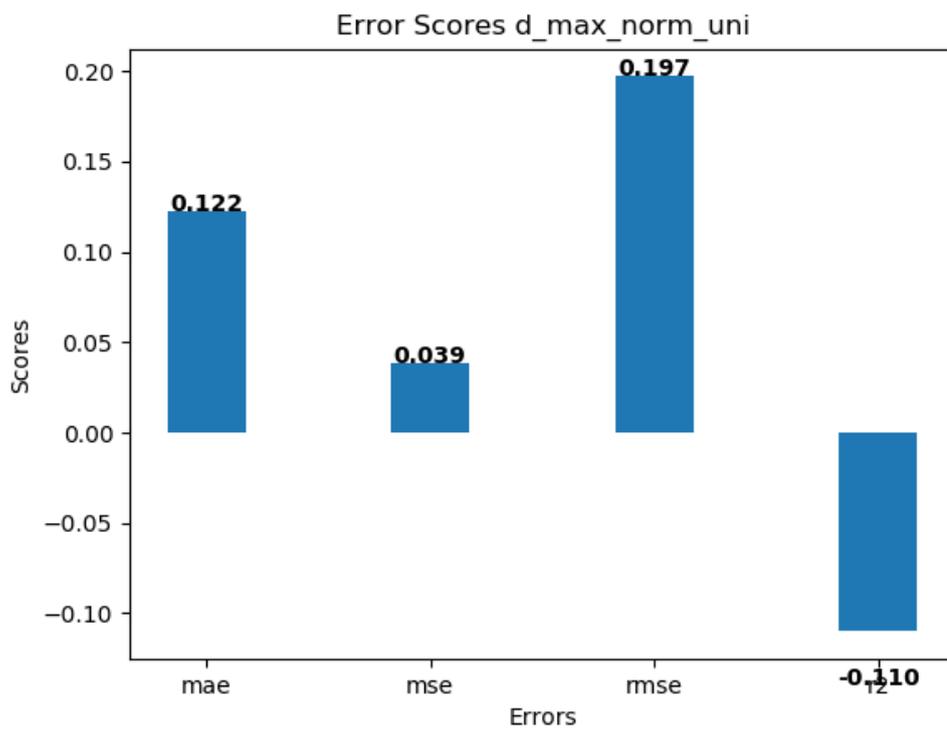


Figure 136. Error metrics score calculated on all batches of test set

- *Multivariate Models*

1) *Model name: "6h_max_6h_std_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

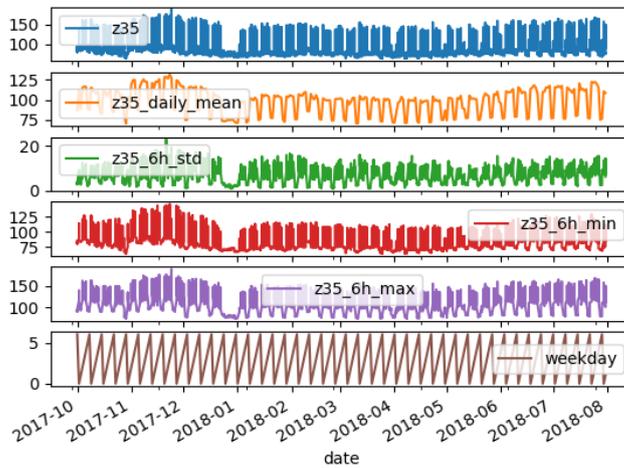


Figure 137. RNN Model Multivariate n.1 – Features used

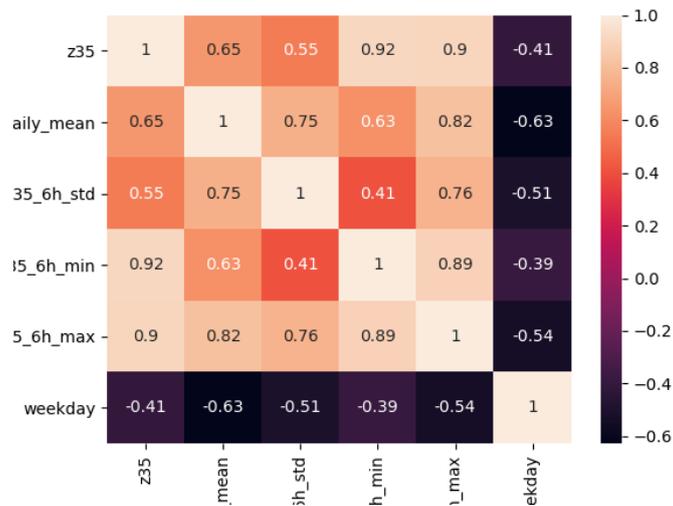


Figure 138. Correlation between features

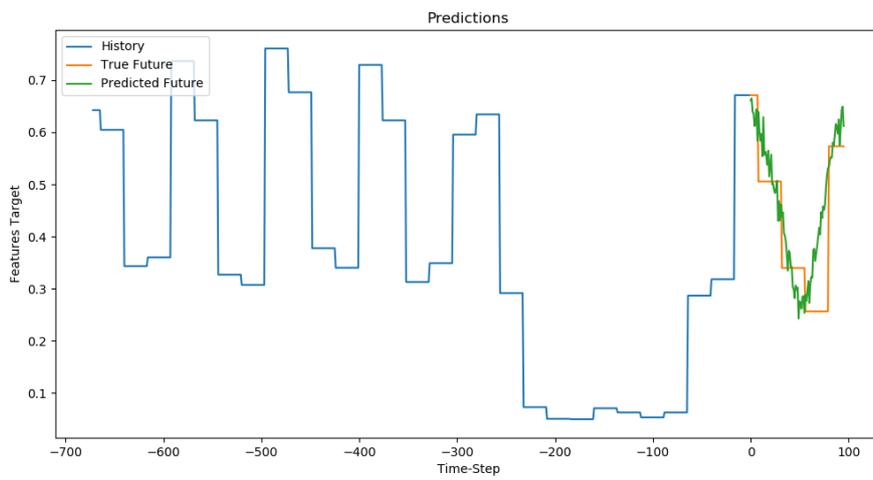


Figure 139. Example of one RNN model prediction

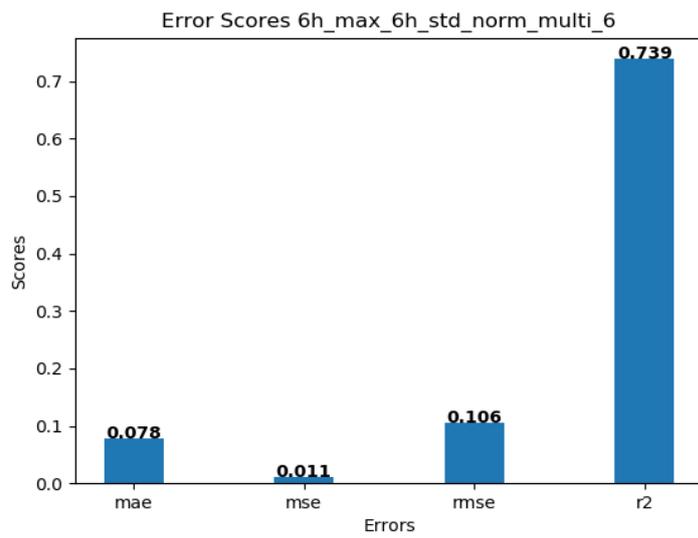


Figure 140. Error metrics score calculated on all batches of test set

2) Model name: "d_max_d_std_norm_multi_6";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

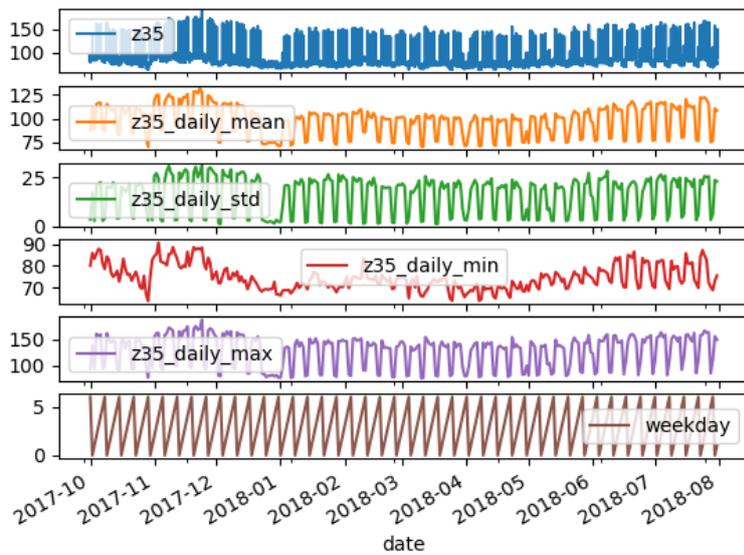


Figure 141. RNN Model Multivariate n.2 – Features used

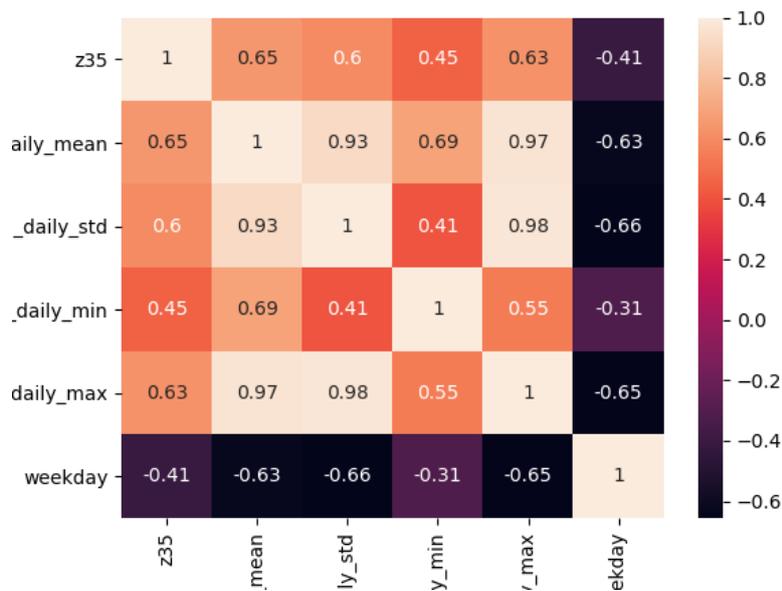


Figure 142. Correlation between features

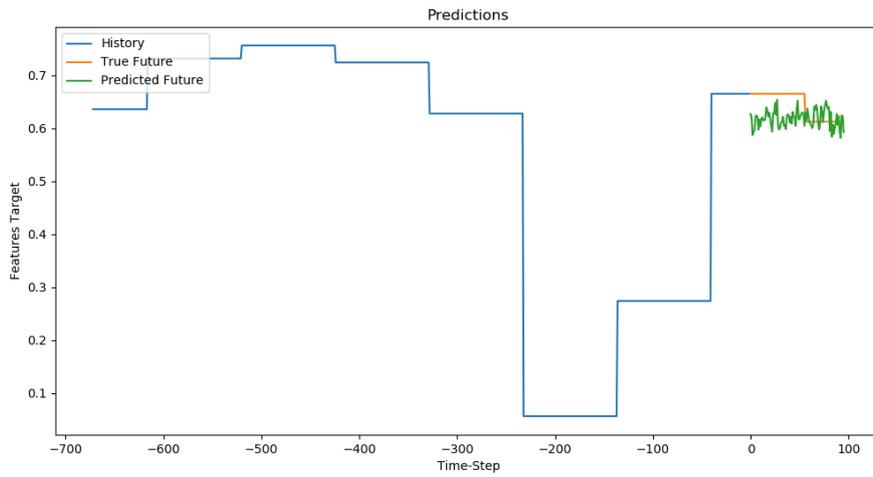


Figure 143. Example of one RNN model prediction

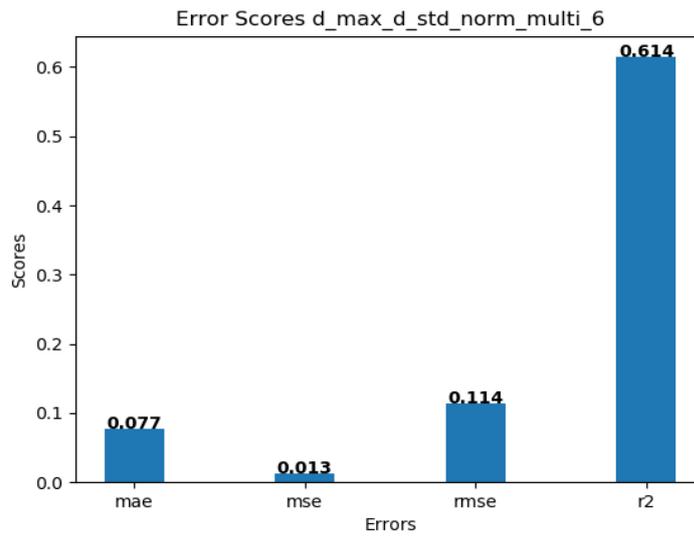


Figure 144. Error metrics score calculated on all batches of test set

3) *Model name: "6h_max_6h_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

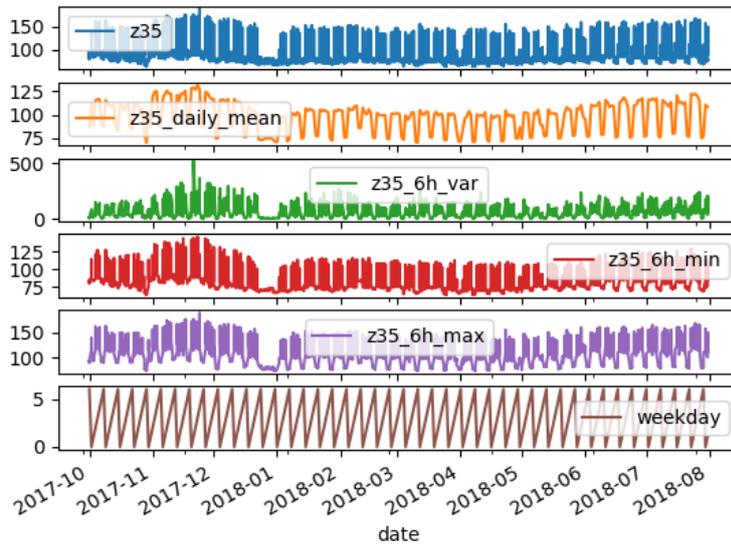


Figure 145. RNN Model Multivariate n.3 – Features used

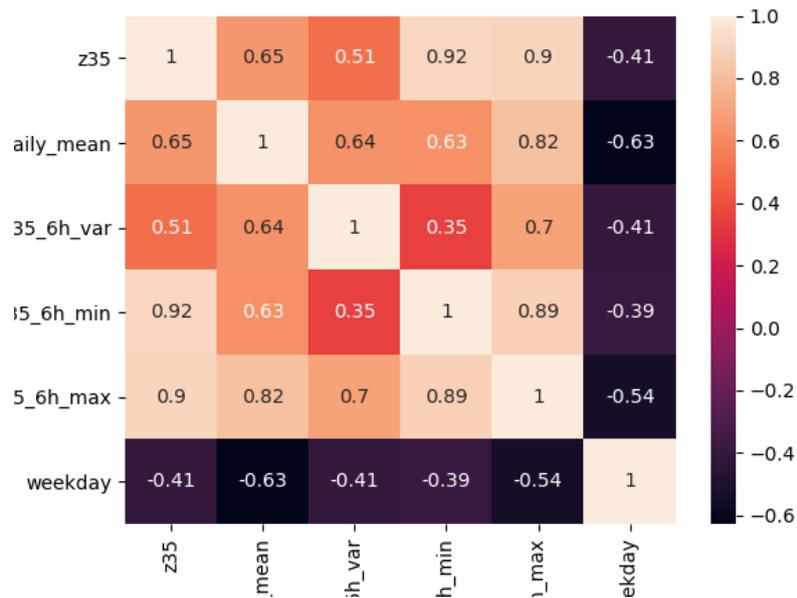


Figure 146. Correlation between features

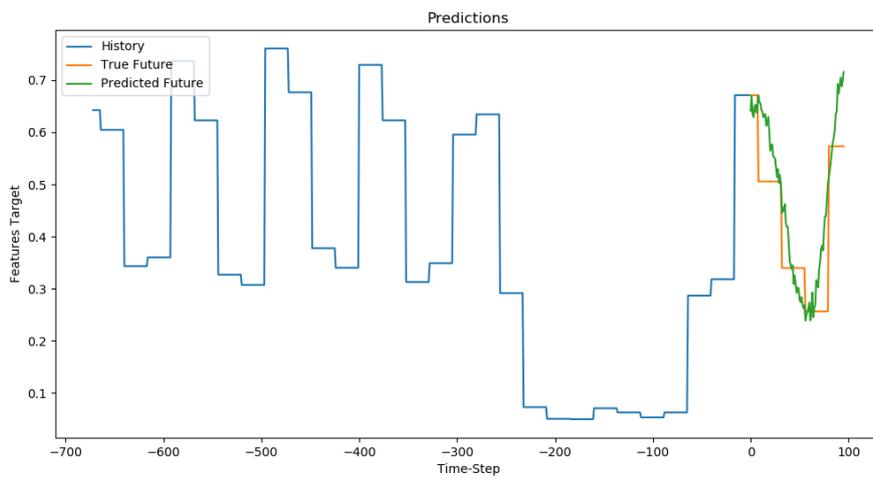


Figure 147. Example of one RNN model prediction

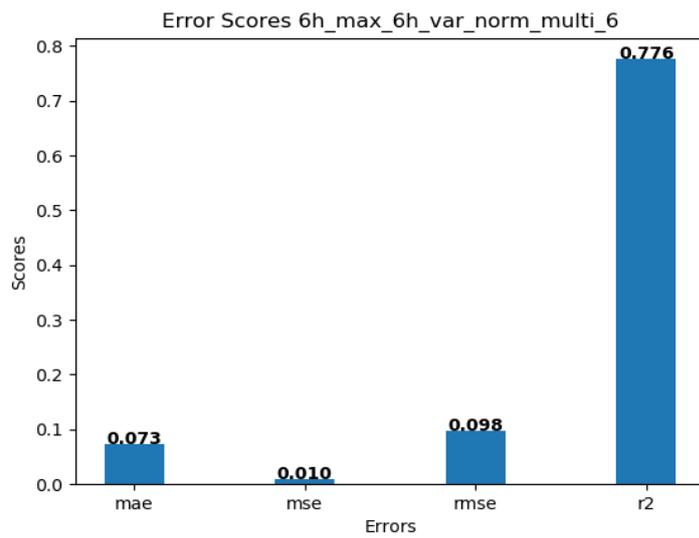


Figure 148. Error metrics score calculated on all batches of test set

4) *Model name: "d_max_d_var_norm_multi_6";*

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (256, 672, 6)$
- True_target shape $y = (256, 96)$
- Predicted_target shape $\hat{y} = (256, 96)$

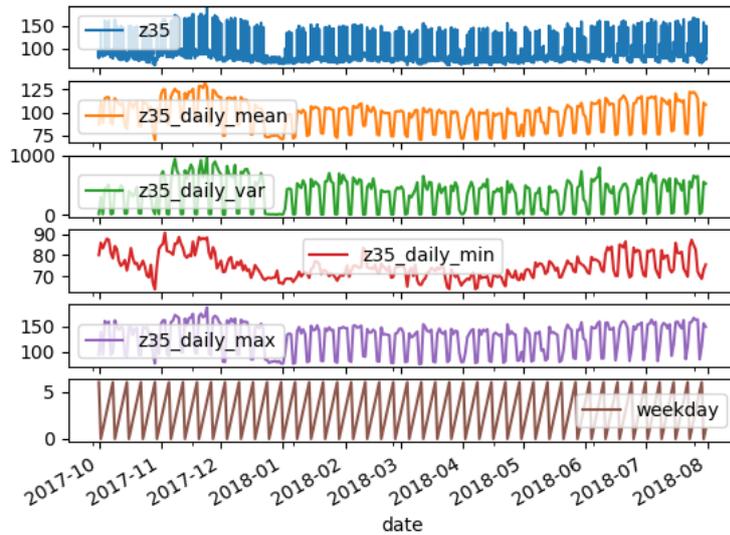


Figure 149. RNN Model Multivariate n.4 – Features used

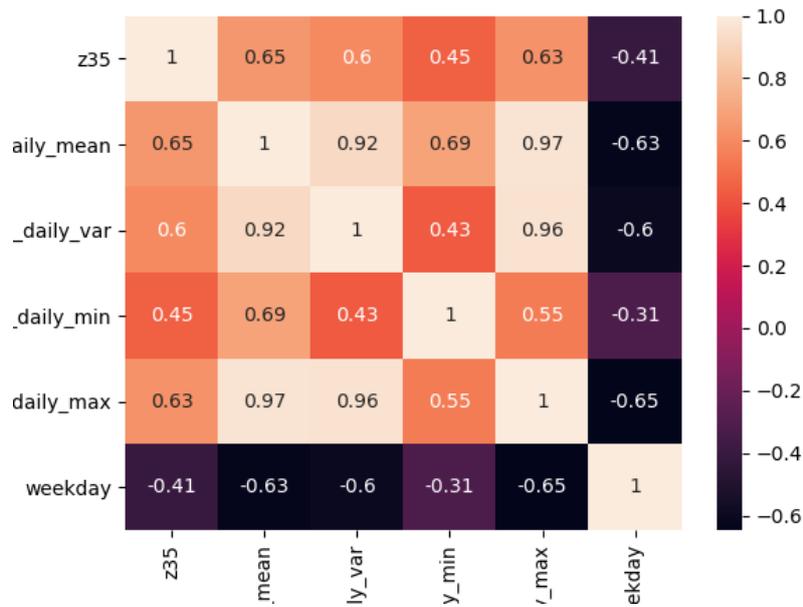


Figure 150. Correlation between features

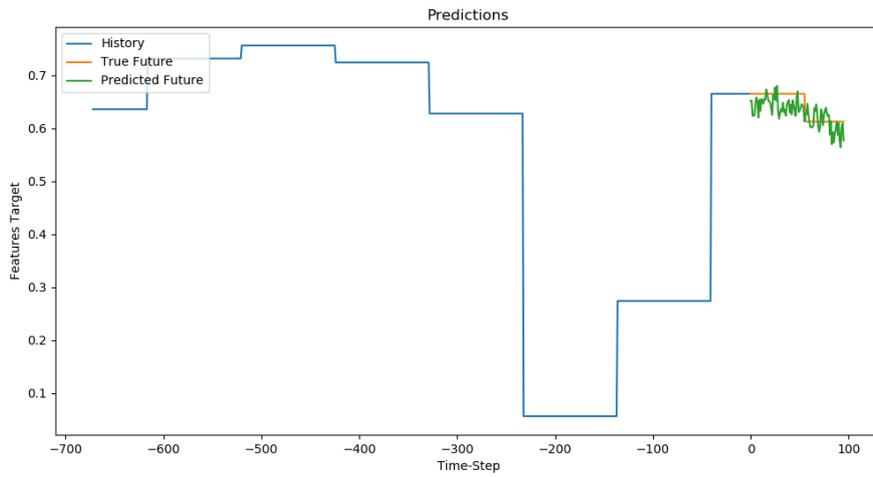


Figure 151. Example of one RNN model prediction



Figure 152. Error metrics score calculated on all batches of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between the six models tested (two Univariate and four Multivariate), to verify which model obtained the best results for predicting z35 maximal value time series. Like we did for the models of z35_min, we have reported in the third column from the left of the following figures another Univariate model not described, where we use always normalized values of the feature z35_max but using RMSprop as optimizer, with cost function MAE to minimize during training; the structure of the layers of the model is always the same.

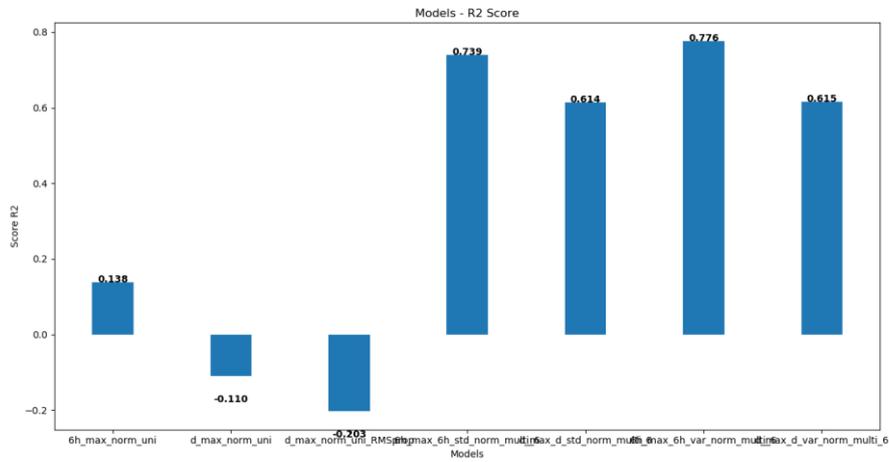


Figure 153. R2 Scores z35 maximal value models tested

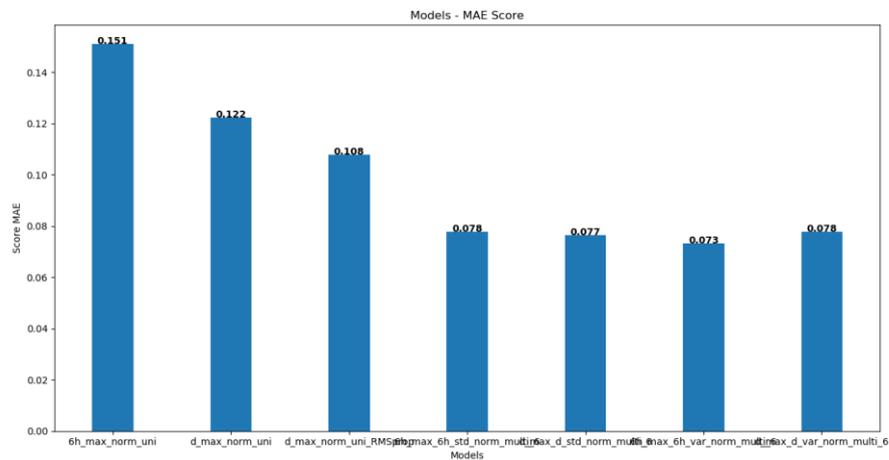


Figure 154. MAE Scores z35 maximal value models tested

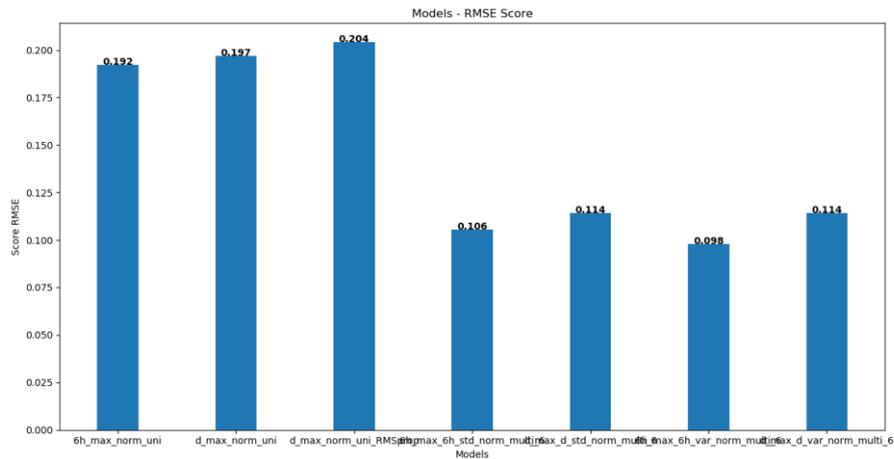


Figure 155. RMSE Scores z35 maximal value models tested

Based on error metrics score, we have another confirm that RNN Multivariate models have better performance than Univariate models, also for predicting z35 maximal value, independently from which time interval we choose for the feature of interest (six hours or one day). Based on time interval, for predicting z35_max values we have the better performance using six hours than one day. The model that show the best performance based on the results of the three error metrics score (represented by the sixth column from the left in the figures) is:

Model name: "6h_max_6h_var_norm_multi_6";

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

So we choose this model as the best RNN model for predicting z35 maximal values time series.

In summary, we report the list of the best models found for predicting the features of interest that are: z35, z35 standard deviation, z35 variance, z35 minimal value and z35 maximal value.

FEATURE PREDICTED	CASE AND FEATURES USED	MODEL
z35	MULTIVARIATE (z35, d_mean, d_var, d_min, d_max, weekday)	RNN (z35_d_var_norm_multi_6)
z35 Standard Deviation (d_std)	MULTIVARIATE (z35, d_mean, d_std, d_min, d_max, weekday)	RNN (d_std_norm_multi_6)
z35 Variance (6h_var)	MULTIVARIATE (z35, d_mean, 6h_var, 6h_min, 6h_max, weekday)	RNN (6h_var_norm_multi_6)
z35 Minimum value (6h_min)	MULTIVARIATE (z35, d_mean, 6h_std, 6h_min, 6h_max, weekday)	RNN (6h_min_6h_std_norm_multi_6)
z35 Maximum value (6h_max)	MULTIVARIATE (z35, d_mean, 6h_var, 6h_min, 6h_max, weekday)	RNN (6h_max_6h_var_norm_multi_6)

Table 1. Best RNN models for predicting z35 and its features

We report also the plots of error metrics R2, MAE and RMSE of the five best models chosen, one for each features of interest. In each plot, the first column represent the error score of z35 best model, the second column of z35 standard deviation best model, the third column of z35 variance best model, the fourth column of z35 minimal value best model and the fifth column of z35 maximal value best model.

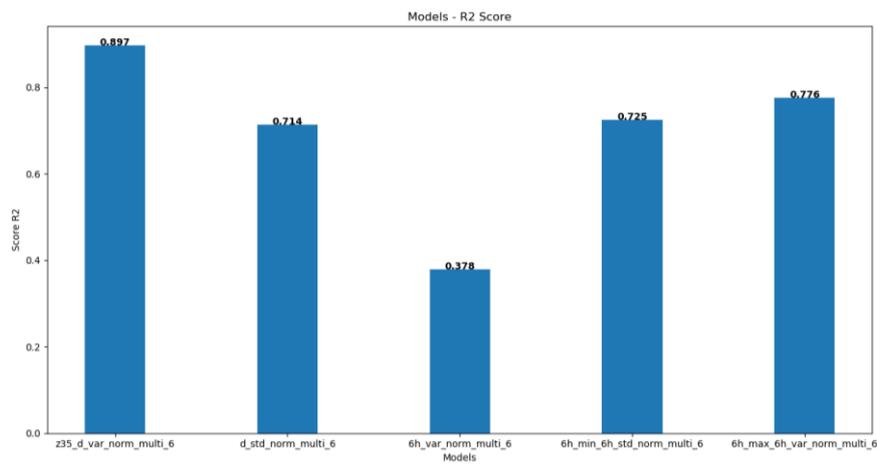


Figure 156. R2 Scores best RNN models

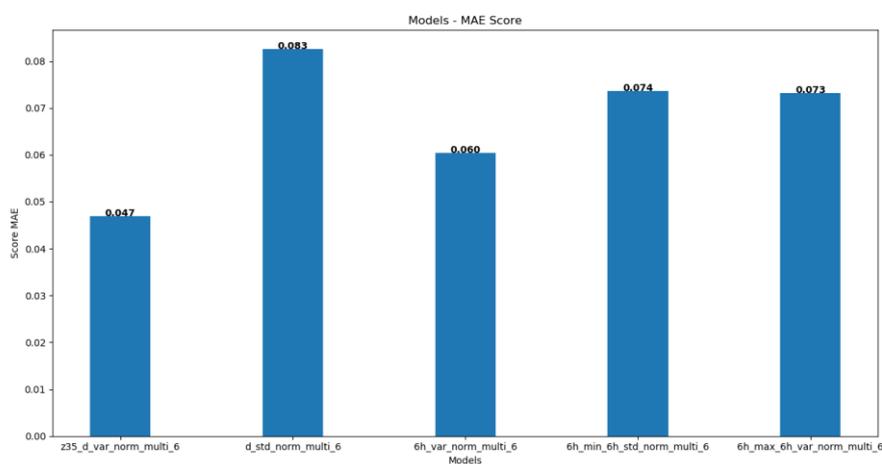


Figure 157. MAE Scores best RNN models

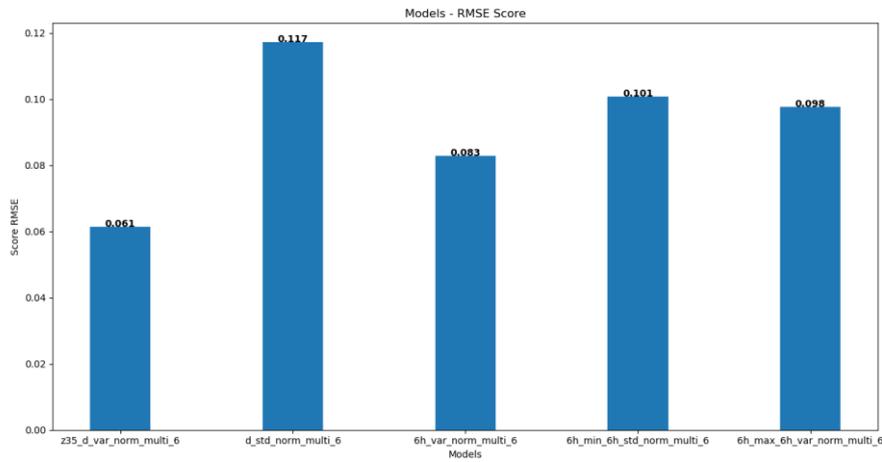


Figure 158. RMSE Scores best RNN models

From our approach to find the best models for predicting z35 and its features using RNN models, so consider a base model and see how the preprocessing of the data and the features choose in Multivariate model can affect the prediction performance, we can conclude that Multivariate models improve the accuracy of the prediction then using Univariate models, as proof at the end of the tests we have found only Multivariate models as best models for predicting the features of interest.

The analysis of the time series to predict, the choice of the features to use in Multivariate models and the preprocessing of the data form a very important part in the design of a time series forecasting model. A characteristic that we can make in evidence for RNN model is properly the fact that Multivariate models have a boost in prediction performance than Univariate models, but only if the features have some correlation with the feature target of the forecasting, because as we observed in the first RNN Multivariate model tested for predicting z35 using ambient temperature, wetbulb temperature, relative humidity and air pressure, we obtained a better score of the error metrics using Univariate model.

2.2 Linear Regression Models

The second family of models tested for predicting electrical load time series belongs to Linear Regression algorithms. Those algorithms are implemented using Scikit-learn library in Python IDE. In linear models, the result of the prediction is a linear combination of the features used, so we will have:

$$\hat{y}(w, x) = w_0 + w_1x_1 + \dots + w_px_p$$

where $w = (w_1, \dots, w_p)$ are the weights of the model and w_0 is the bias term, while $x = (x_1, \dots, x_p)$ represent the features used in the model.

The consideration made after the analysis of the time series z35 are valid also for Linear Regression models, so we will always use a time window of `past_history` of a week given the weekly seasonality of the time series. The difference from RNN models is that for the number of timesteps of `future_target` to predict we will use a time window composed of 672 samples of a week, that's because, for how the algorithm is designed, it needs the same number of samples from input to output, so if the model use a time window of `past_history` of 672 timesteps as input, it have to predict a time window of `future_target` of 672 timesteps as output.

In RNN models we used approximatively always the same base model (the structure of the layers was always the same, we have changed only the optimizer and the cost function), focusing more on how the prediction performance are influenced by preprocessing of the data and the choice of the features to use. With linear models we use also a technique for hyperparameters optimization of the models, called Grid Search. Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. We search the optimal configuration of the hyperparameter considering as reference the Univariate model for predicting z35 time series and then applying this configuration also to the Multivariate models

and the models for predicting standard deviation, variance, minimal and maximal values, because these features are extracted directly from z35 time series.

2.2.1 Linear Regression Models – Preliminary Test

Linear Regression models that we are going to test and compare are Linear Regression (LR), Ridge Regression (RR), Lasso (L), ElasticNet (EN), Stochastic Gradient Descent Regressor (SGDR) and Passive Aggressive Regressor (PAR). We are going to verify which configuration of hyperparameters is better for predicting z35 using Grid Search method in Univariate model, and then apply this optimal configuration obtained to a Multivariate model for predicting z35 and to others models for predicting some features extracted from z35, that are: daily variance, daily minimum value and daily maximum value. In this models we use standardization of the data as scaling technique. At the end we report in a plot an example of the prediction of z35 combining with the predictions of the others features of interest that give to us the confidence bound where the future values of z35 probably fall.

Each Linear Regression models needs as input a 2D tensor of shape ($n_samples$, $n_features$) and produce as output a vector of dimension ($n_samples$) because in our case we always want to predict only one feature. In this case the preprocessing of the data consist in:

- Dividing the 29184 samples of z35 time series in training and test set, using 25000 samples for training set and the remaining 4184 samples for test set.
- Applying standardization of the values (the scale factors mean and standard deviation are always to calculate on samples of training set and then applying to all dataset).

- Creating the samples for the model, using the time window of past_history of 672 timesteps and future_target of 672 timesteps, shifting it one timesteps per time along all dataset.
- After create the samples for the model, we have data variables of dimension (n_samples, n_timesteps=672, n_features), so we have to apply a flattening operation in the first two dimensions to obtain the 2D tensor input needed from the model of dimension (n_samples*n_timesteps, n_features).

Now we are going to describe how works each models tested and showing the results of the predictions with the calculate of error metrics considering all the data in test set.

- **Linear Regression (LR)**

In Linear Regression (LR) model we don't use the Grid Search process to tuning the hyperparameters because we don't have parameters to optimize. This model, at each iteration during training, updates the weights associated to each input trying to minimize the residual sum of squares (RSS).

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j * x_{ij} \right)^2$$

Equation 8. Residual Sum of Squares

Model name: "LR_uni";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

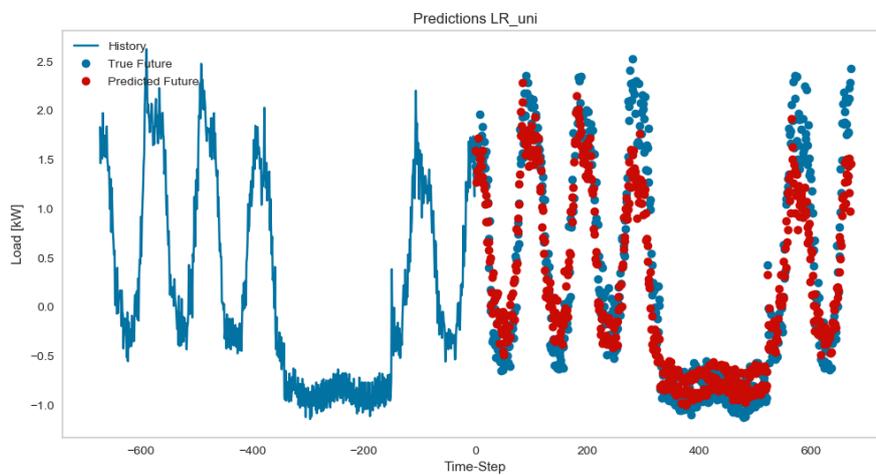


Figure 159. Example of one LR model prediction

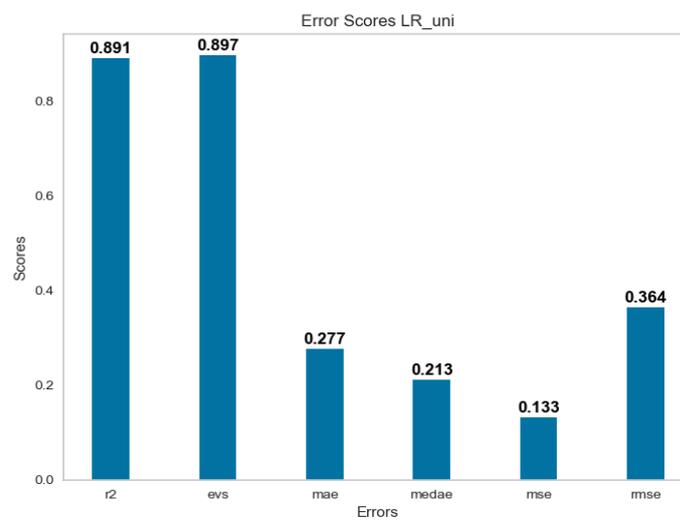


Figure 160. Error metrics score calculated on all samples of test set

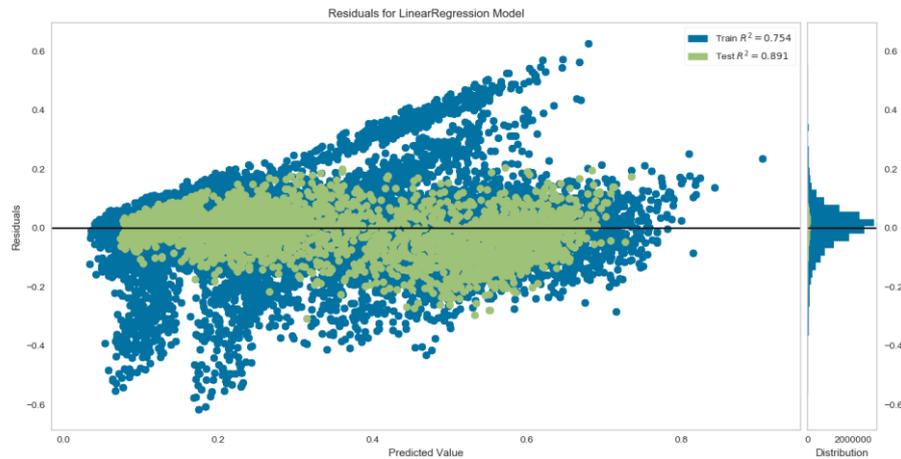


Figure 161. Residuals plot Linear Regression model

In this case we reported also the residuals plot of Linear Regression model. We can observe that the distribution of the residuals in Train and Test set is a normal distribution, it means Linear models are good for predicting this kind of data.

- Ridge Regression (RR)

In Ridge Regression (RR) model, during training phase it updates the weights associated to the inputs trying to minimize a cost function formed by the residual sum of squares (RSS) plus a L2 regularization function (related to the square of the weights)

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j * x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2$$

Equation 9. Cost Function Ridge Regression model

In this case, we apply Grid Search method to find the optimal value for the regularization factor λ , so we optimize the parameter that affect the penalty added to the model. In case we obtain as best value $\lambda = 0$, the model reduces to

a simple Linear Regression, because it will use RSS as cost function. Instead if $\lambda \rightarrow \infty$ the penalty introduced at each iteration is very high and it lead some weights to be almost null, though they are not effectively excluded from the model. Since we have found a parameter $\lambda = 0$ after Grid Search, we don't consider this model because it's equal to the previous Linear Regression model.

- **Lasso (L)**

In Lasso (L) model, the process to update the weights associated to the inputs it's similar to Ridge Regression, with the difference that we use a L1 regularization function (related to absolute value of the weights), instead of L2.

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j * x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j|$$

Equation 10. Cost Function Lasso model

Also in this case, with Grid Search method we want to optimize the regularization factor λ . If $\lambda = 0$ the model reduces to a simple Linear Regression, while if $\lambda \rightarrow \infty$ the presence of absolute operator make some weights to be null, so it permit to make a selection of the variable and can exclude some of them from the model. Since we have found also here a parameter $\lambda = 0$ after Grid Search, we don't consider this model because it's equal to the previous Linear Regression model.

- **ElasticNet (EN)**

In ElasticNet (EN) model, the cost function is formed by residual sum of squares plus L1 and L2 regularization terms.

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j * x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right) =$$

$$= RSS + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$

Equation 11. Cost Function ElasticNet model

In this case, with Grid Search method we are going to optimize the parameters λ and α (values from 0 to 1), where α represent the L1 ratio. In case $\lambda = 0$, the model reduces to simple Linear Regression. If $\alpha = 0$ the model have a penalty of type L2, while if $\alpha = 1$ the model have a L1 penalty (become equal to Lasso model). Also in this model we have found a parameter $\lambda = 0$ after Grid Search, so we don't consider this model because it's equal to the previous Linear Regression model.

- **Stochastic Gradient Descent Regressor (SGDR)**

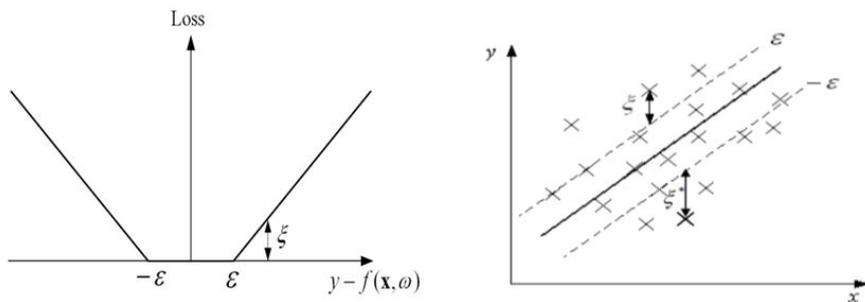
In Stochastic Gradient Descent Regressor (SGDR) model, the cost function is minimized using Gradient Descent technique, so at each iteration it calculate the gradient of the cost function and use it to update the weights of the model based on learning rate. Also in this case we can add to cost function a regularization term that can be L1, L2 or both types. The hyperparameters we want to optimize with Grid Seach method are: regularization factor (hyperparameter alpha), cost function (loss), penalty and learning rate.

$$\left\{ \begin{array}{l} \alpha = [0,1; 0,01; 0,001; 0,0001; 0,00001; 0,000001] \\ \text{loss} = [\text{squared loss}; \text{huber}; \text{epsilon insensitive}] \\ \text{penalty} = [L2; L1; \text{ElasticNet}] \\ \text{learning rate} = [\text{constant}; \text{optimal}; \text{invscaling}] \end{array} \right.$$

After Grid Search process, we obtain these optimal values for the hyperparameters:

$$\left\{ \begin{array}{l} \alpha = [0,01] \\ \text{loss} = [\text{epsilon insensitive}] \\ \text{penalty} = [L1] \\ \text{learning rate} = [\text{constant}] \end{array} \right.$$

So the loss function ignores error of value less than epsilon (equal to 0,1) and it's linear past this threshold, the regularization factor (alpha) it's equal to 0,01 and the penalty used is L1 type. The learning rate used to update the weights is constant.



$$L_{\epsilon}(y, f(\mathbf{x}, \omega)) = \max(|y - f(\mathbf{x}, \omega)| - \epsilon, 0)$$

Figure 162. Epsilon insensitive Loss

Model name: "SGDR_uni";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

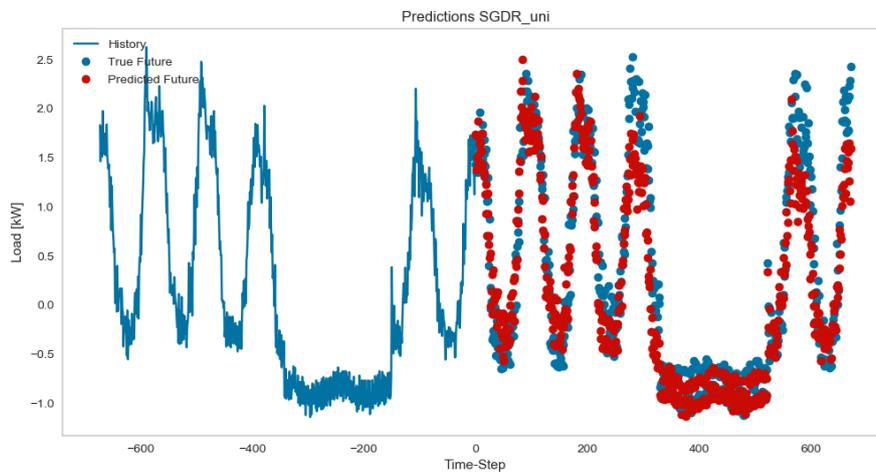


Figure 163. Example of one SGDR model prediction

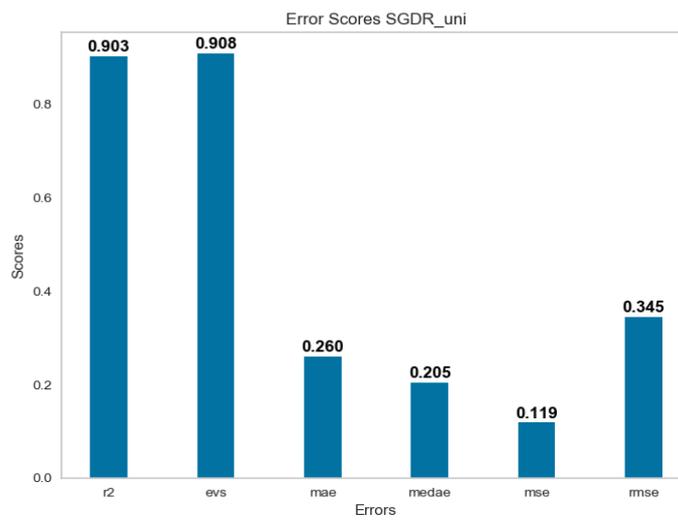


Figure 164. Error metrics score calculated on all samples of test set

- **Passive Aggressive Regressor (PAR)**

The last linear model tested is Passive Aggressive Regressor (PAR), that can be of type PA-1 or PA-2 based on which cost function it uses, PA-1 if cost function is 'epsilon_insensitive' and PA-2 if it's 'squared_epsilon_insensitive'. We already choose to use a PA-1 model, so what we are going to optimize with Grid Search method is the regularization parameter C

$$\{C = [0; 0,5; 1; 0,1; 0,01; 0,001]\}$$

We found as optimal value of regularization factor $C = 0,001$. Parameter epsilon of epsilon_insensitive cost function is set equal to 0,1.

Model name: "PAR_uni";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

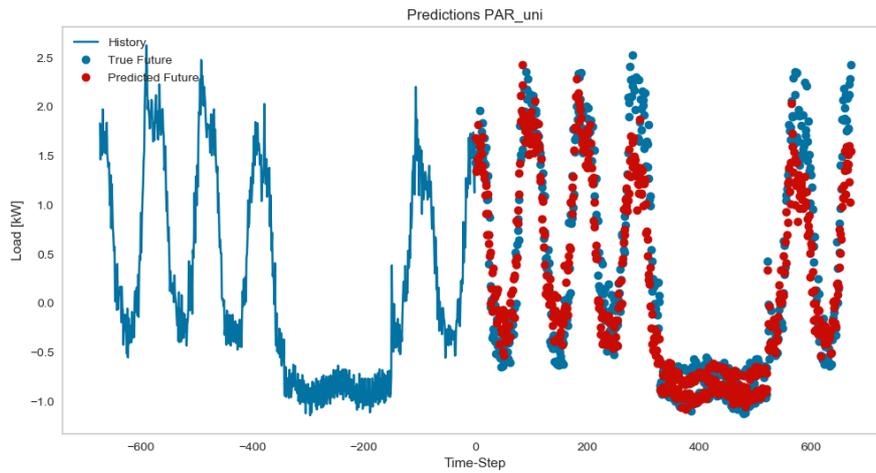


Figure 165. Example of one PAR model prediction

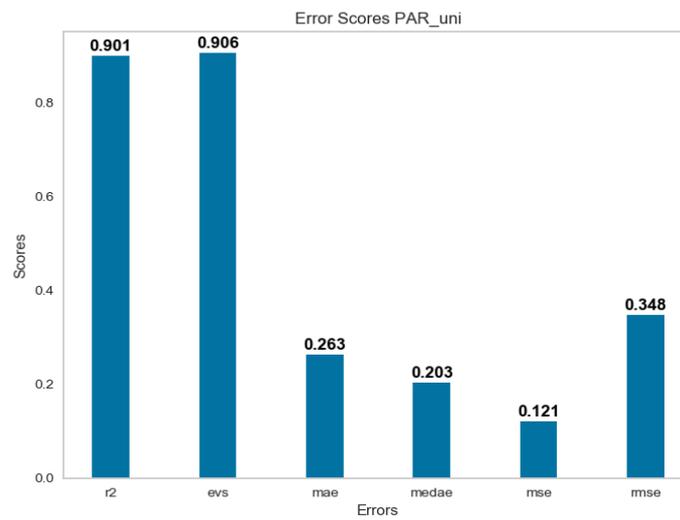


Figure 166. Error metrics score calculated on all samples of test set

In the following figures, we report the comparison plots of error metrics score R2, MAE and RMSE of each models, using also Ridge Regression, Lasso and ElasticNet models, to verify that with hyperparameter $\lambda = 0$ they are equal to Linear Regression model.

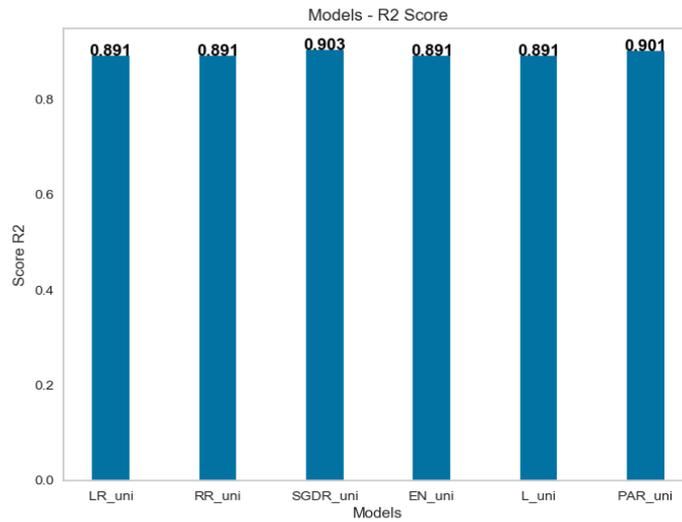


Figure 167. R2 Scores Linear Univariate z35 models tested

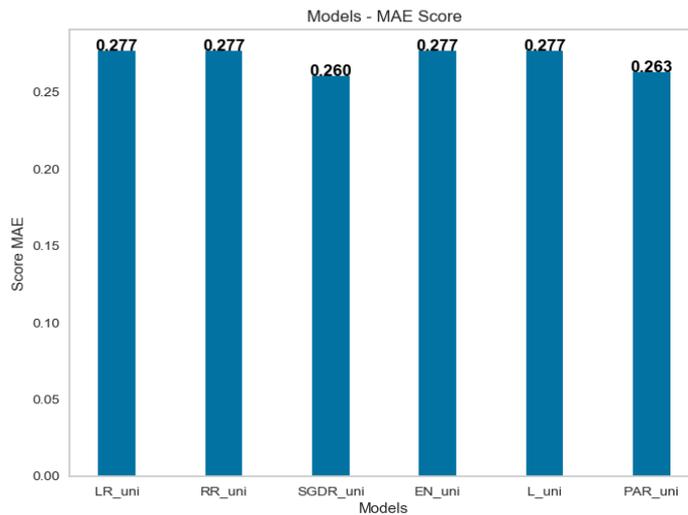


Figure 168. MAE Scores Linear Univariate z35 models tested

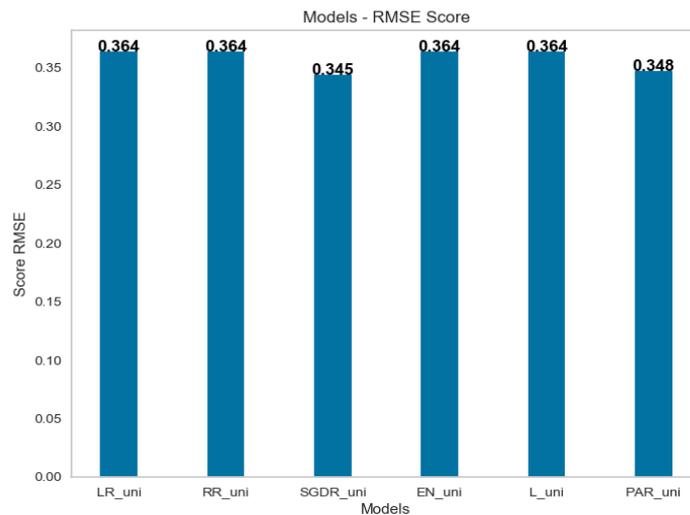


Figure 169. RMSE Scores Linear Univariate z35 models tested

We can observe that RR, L and EN models have the exact error scores of LR model. From this first test using models belonging to Linear Regression family, the best model to predict z35 in Univariate case is Stochastic Gradient Descent Regressor (SGDR).

We want also to test the results of the predictions of this models in Multivariate case, considering some features extracted directly from z35 that are: z35 daily mean, z35 daily variance, z35 daily minimal value and z35 daily maximal value; we use also the time series 'weekday' representing the weekday code of the day, for considering the different behaviour of z35 during weekday and weekend days. The configuration of the models realized is the same used in Univariate case. We want to verify how much the choice and the using of extra features affect the prediction performance of Linear models. We have chosen these extra time series in Multivariate case because are the combination of features used in the best RNN model found for predicting z35 values.

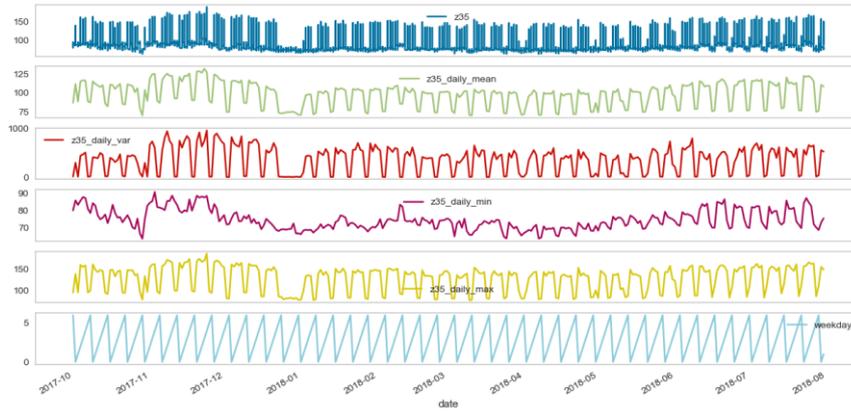


Figure 170. Linear Model Multivariate – Features used



Figure 171. Correlation between features

- *Multivariate Models*

Model name: "LR_multi";

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=6)$
- True_target shape $y = (n_timesteps=672, n_features=6)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

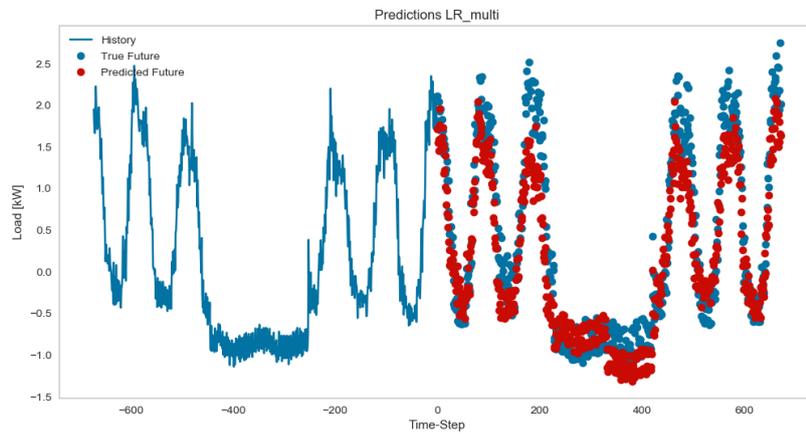


Figure 172. Example of one LR model prediction

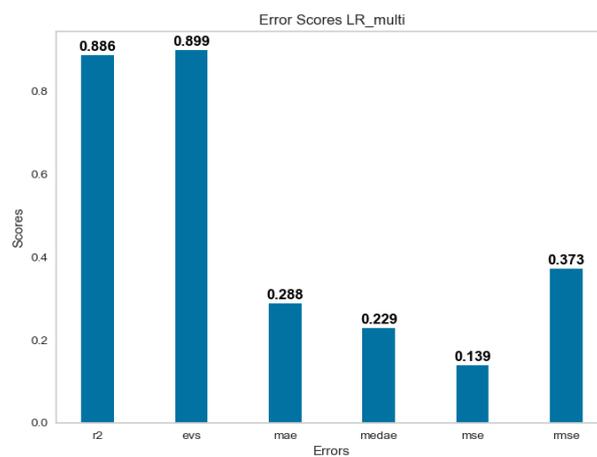


Figure 173. Error metrics score calculated on all samples of test set

Model name: "SGDR_multi";

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=6)$
- True_target shape $y = (n_timesteps=672, n_features=6)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

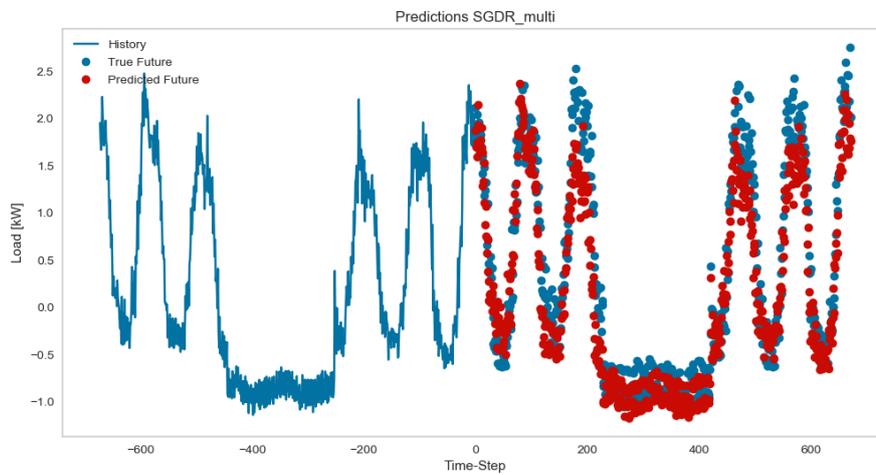


Figure 174. Example of one SGDR model prediction

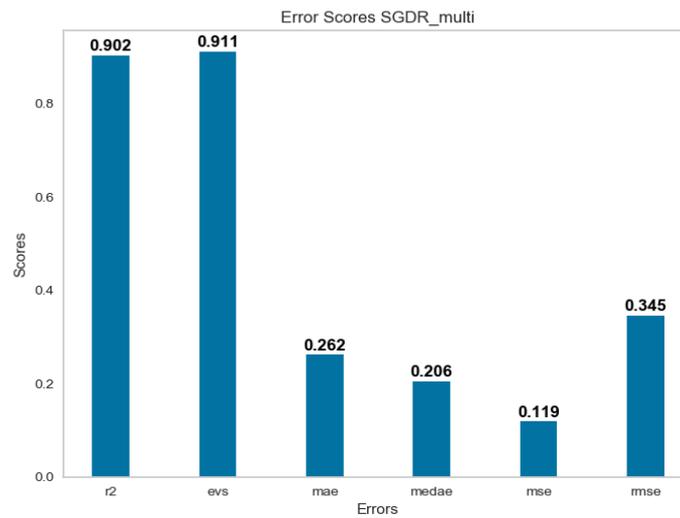


Figure 175. Error metrics score calculated on all samples of test set

Model name: "PAR_multi";

Features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=6)$
- True_target shape $y = (n_timesteps=672, n_features=6)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

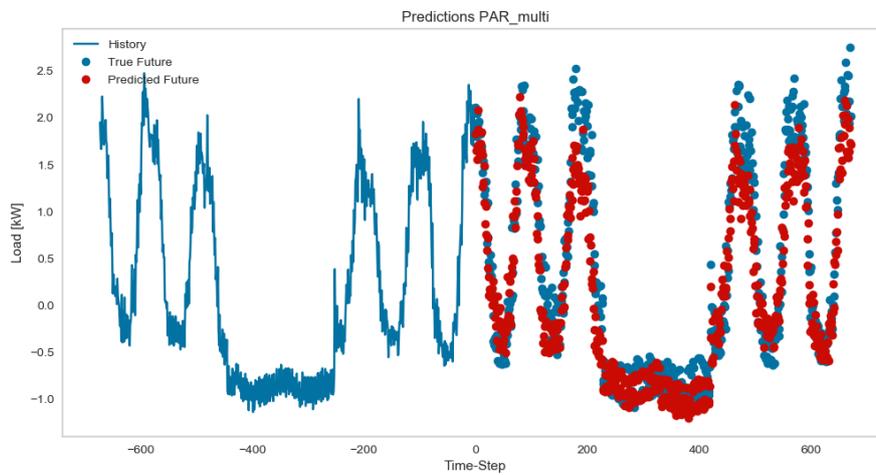


Figure 176. Example of one PAR model prediction

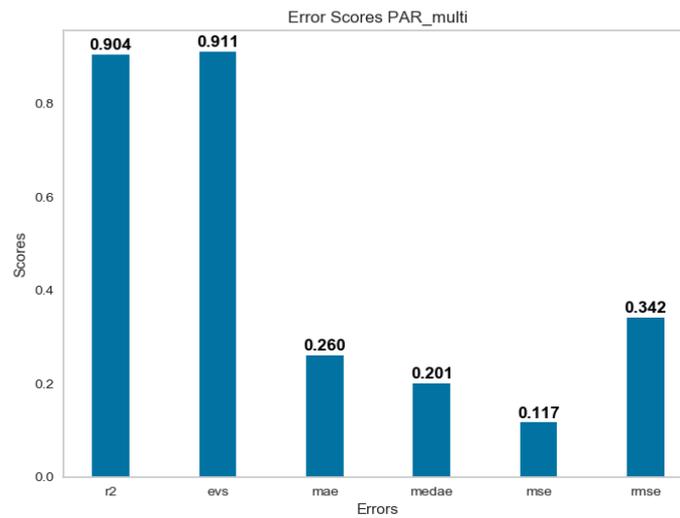


Figure 177. Error metrics score calculated on all samples of test set

As we done for Univariate model, we report the comparison plot of error metrics score R2, MAE and RMSE of each models, including also Ridge Regression, Lasso and ElasticNet models.

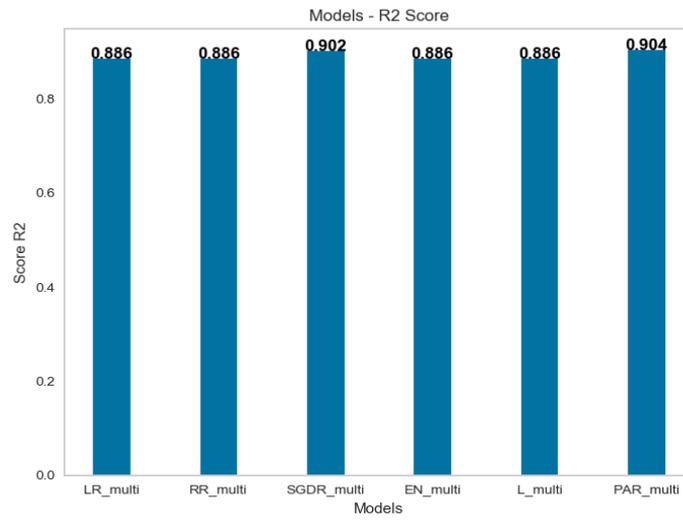


Figure 178. R2 Scores Linear Multivariate z35 models tested

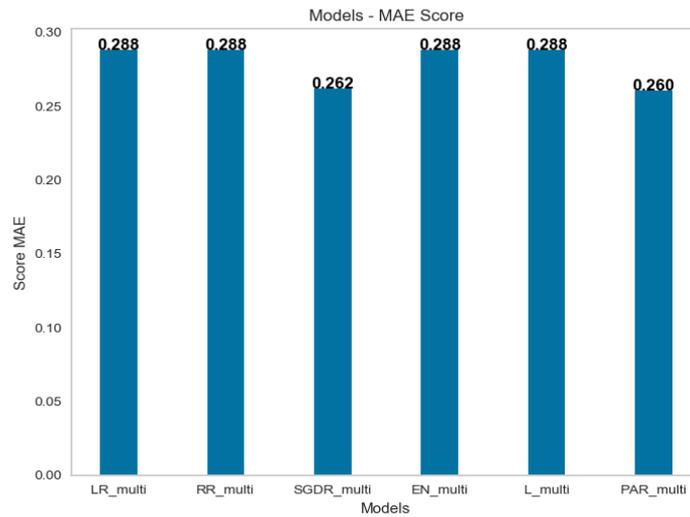


Figure 179. MAE Scores Linear Multivariate z35 models tested

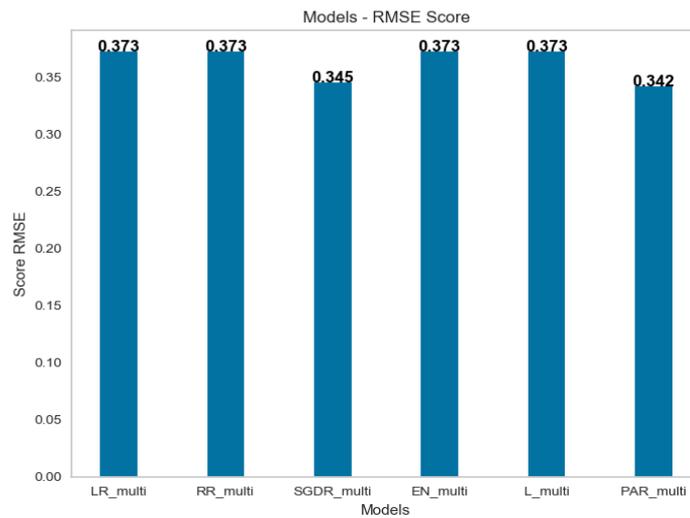


Figure 180. RMSE Scores Linear Multivariate z35 models tested

We can observe that RR, L and EN models have the exact error scores of LR model, so from now we only focus on LR, SGDR and PAR models. Based on the error metrics scores obtained, in Multivariate models we have a small performance improvement for PAR model and a small degrade of performance for LR and SGDR model. Considering both Univariate and Multivariate models, the model that shows the most accurate precision for predicting z35 is Passive Aggressive Regressor model in Multivariate case. Unlike RNN models where the using of extra features correlated to the time series target improved the prediction performance, for Linear Regression models we don't have significant improvement from Univariate to Multivariate case, indeed in some case we observe a degrade in performance.

Also for Linear regression models we have to find models that predicting electric load values with some confidence bound. For this purpose, we realized also models that predict the characteristics of interest of electric load time series, that are standard deviation, variance, minimal value and maximal value.

Before proceed in the next part to test all the models in Univariate and Multivariate case for finding which linear model has the best prediction performance like we have done for RNN models, we want to show as demonstration how would be the final prediction plot containing the predictions of each features of interest with data not scaled. Due to the fact that this is not the final test between the most promises Linear models but have only a demonstration purpose, starting from the combination of features used in the Linear Regression model in Multivariate case for predicting z35 just described (features used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday), we report directly the best models for predicting z35, z35_daily_var, z35_daily_min and z35_daily_max and the resulting final plot (the models for predicting the four features of interest are tested in Univariate and Multivariate case and have the same configuration of the model just described for predicting z35).

The list of best models is:

- z35: PAR Multivariate
- z35_d_var: SGDR Multivariate
- z35_d_min: LR Univariate
- z35_d_max: PAR Univariate

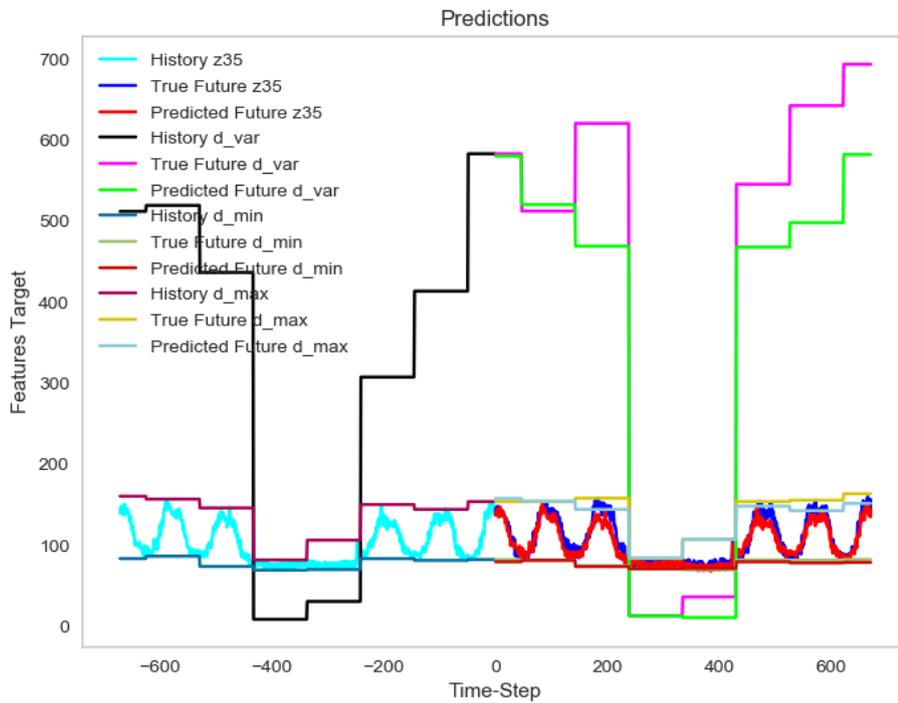


Figure 181. Example n.1 of the predictions of z35 and its features

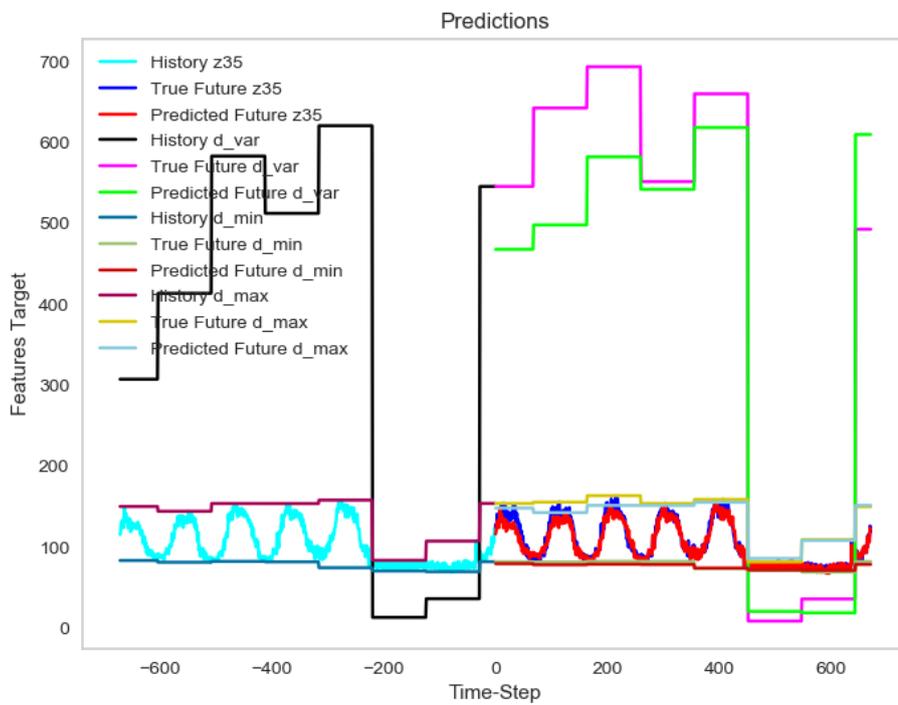


Figure 182. Example n.2 of the predictions of z35 and its features

2.2.2 Linear Regression Models – Final Test

In this test, we start from the approach used in previous analysis on Linear Models, so optimizing Linear models tested using Grid Search process for predicting z35 time series and applying this tuned models for predicting also the other time series of interest, with the assumption that being features extracted directly from z35 these models can predict well also these time series. Therefore Linear Regression models that we are going to test are: Linear Regression (LR), Stochastic Gradient Descent Regressor (SGDR) and Passive Aggressive Regressor (PAR). In this case we use normalization as scaling technique, because we want to generalize the models to other time series related to electric load power and, as described in RNN models, normalization generalizes better than standardization. The time window used is always of one week of past_history and one week of future_target, so 672 timesteps for both of them. For error metrics, we calculate R2, MAE and RMSE score of each models using all data in test set. In summary, the process to realize and test the models are:

- Dividing dataset of 29184 samples, in 25000 samples of train set and 4184 samples of test set.
- Applying normalization of values between 0 and 1. Scale factors are calculated on train set and then applied on all dataset.
- Applying the time window of 672 timesteps of past_history and 672 timesteps of future target to train and test set. We obtain data with dimension (n_samples, n_timesteps=672, n_features).
- Applying a flattening operation into first two dimensions of the data obtained after time window process for training the model, so data are format (n_samples*n_timesteps, n_features)
- Calculate error metrics score R2, MAE and RMSE using all data in test set.

We will test Linear Models in Univariate and Multivariate cases, using different combination of features, in the same way done for RNN model. The features we want to predict are: z35, z35 standard deviation, z35 variance, z35 minimal value and z35 maximal value. We use different time interval to calculate this features extracted from z35, this interval can be of one day or of six hours. In the models we use standard deviation or variance of z35 because they basically provides the same information.

The models tested are:

- Prediction of **z35**:
 - Feature used in Univariate models:
 1. z35
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Predictions of **std** (z35 standard deviation):
 - Feature used in Univariate models:
 1. z35_6hours_std
 2. z35_daily_std
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);

- Prediction of **var** (z35 variance):
 - Feature used in Univariate models:
 1. z35_6hours_var
 2. z35_daily_var

- Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Predictions of **min** (z35 minimum value):
 - Feature used in Univariate models:
 1. z35_6hours_min
 2. z35_daily_min
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Prediction of **max** (z35 maximum value):
 - Feature used in Univariate models:
 1. z35_6hours_max
 2. z35_daily_max
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

For each Univariate and Multivariate case, we create a Linear Regression (LR) model, a Stochastic Gradient Descent Regressor (SGDR) model and a Passive Aggressive Regressor (PAR) model; being the models tested the same used in

RNN models in terms of features used, we report for each of them only the results of error metrics R2, MAE and RMSE, without plot times series and correlation between them.

At the end, for each one of the five features we want to predict, we compare the error metrics score obtained using each model realized for that feature and choose the best one.

-
- Prediction of **z35**
 - *Univariate Models*
-

1) *Feature used: z35*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

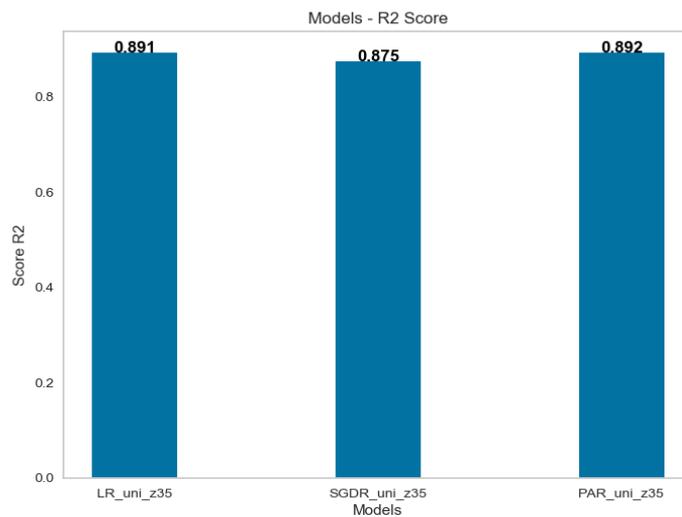


Figure 183. R2 Scores Linear models tested for predicting z35 Univariate

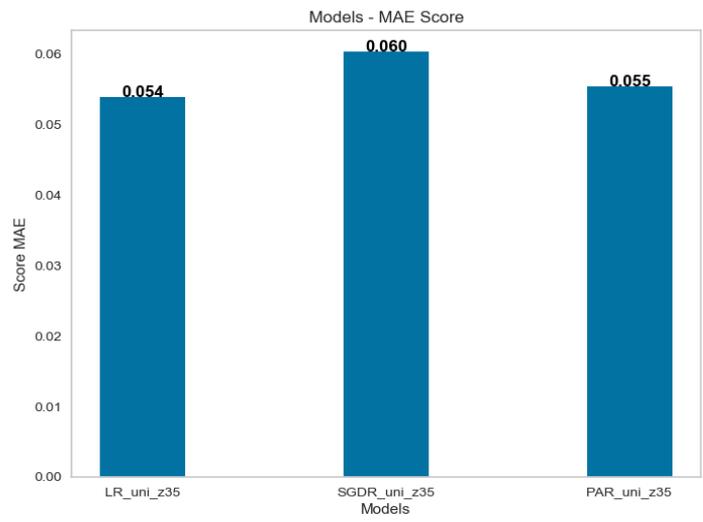


Figure 184. MAE Scores Linear models tested for predicting z35 Univariate

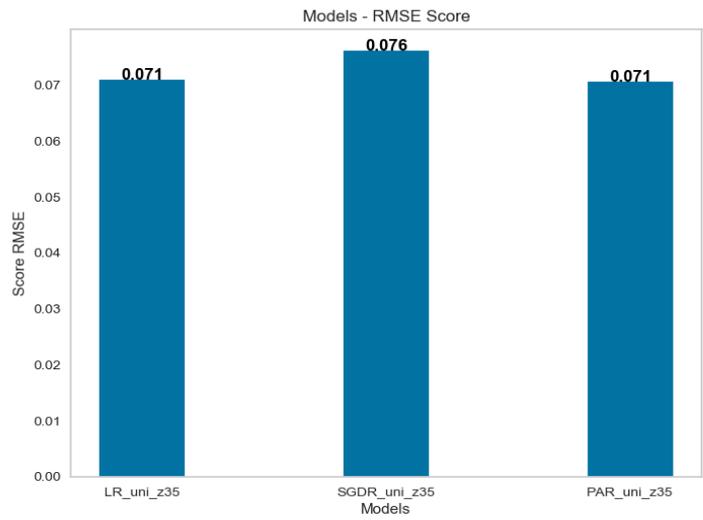


Figure 185. RMSE Scores Linear models tested for predicting z35 Univariate

- *Multivariate Models*

- 1) *Features used:* z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday
 - Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

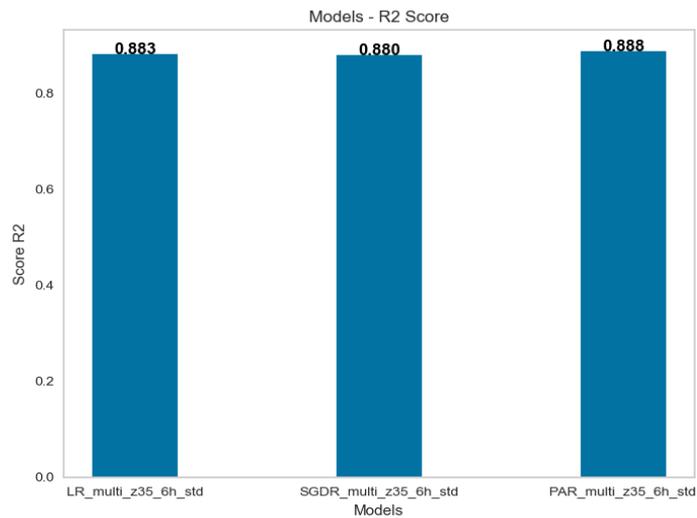


Figure 186. R2 Scores Linear models tested for predicting z35 Multivariate

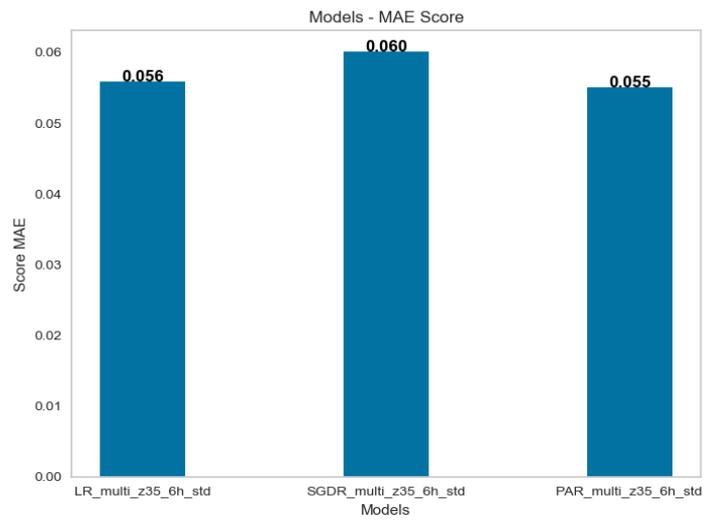


Figure 187. MAE Scores Linear models tested for predicting z35 Multivariate

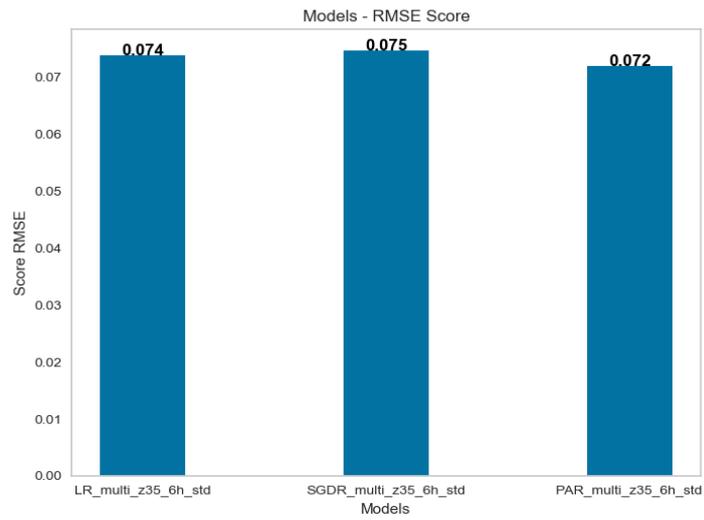


Figure 188. RMSE Scores Linear models tested for predicting z35 Multivariate

- 2) *Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

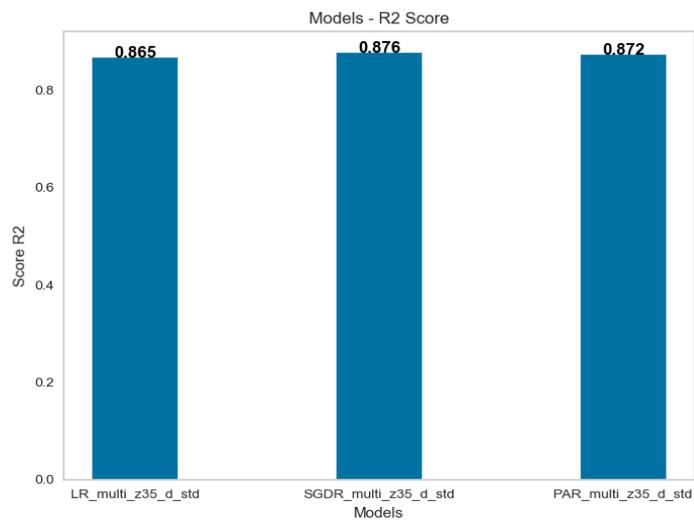


Figure 189. R2 Scores Linear models tested for predicting z35 Multivariate

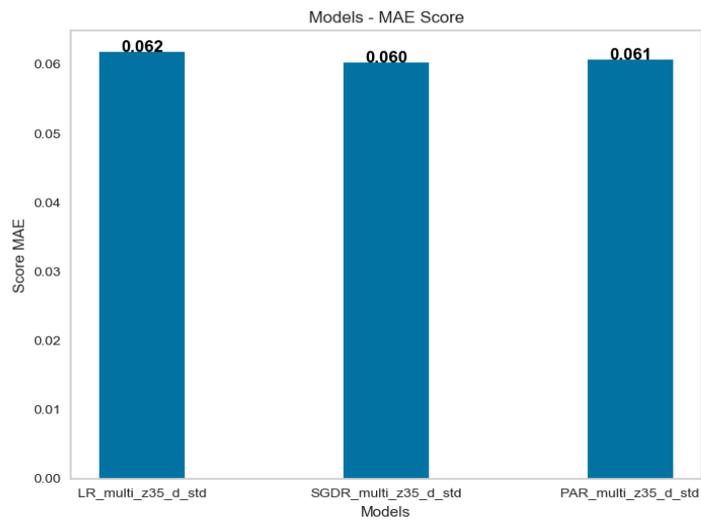


Figure 190. MAE Scores Linear models tested for predicting z35 Multivariate

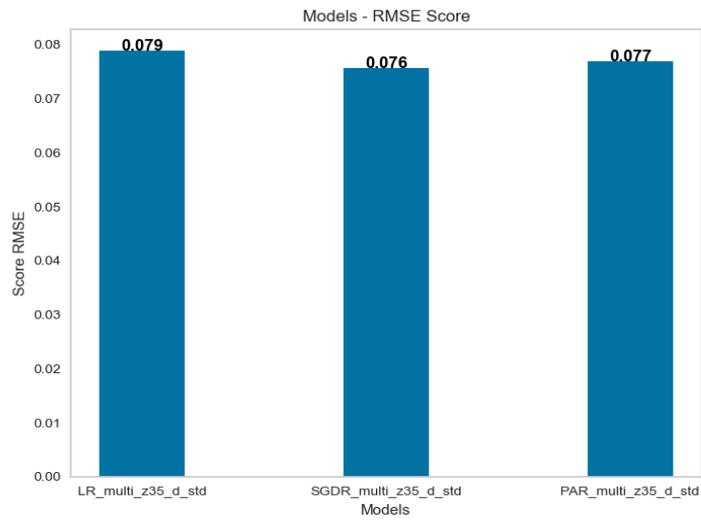


Figure 191. RMSE Scores Linear models tested for predicting z35 Multivariate

- 3) *Features used*: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

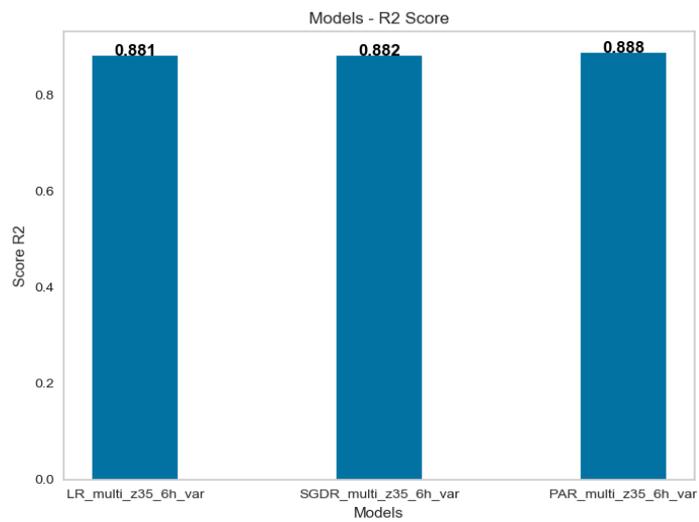


Figure 192. R2 Scores Linear models tested for predicting z35 Multivariate

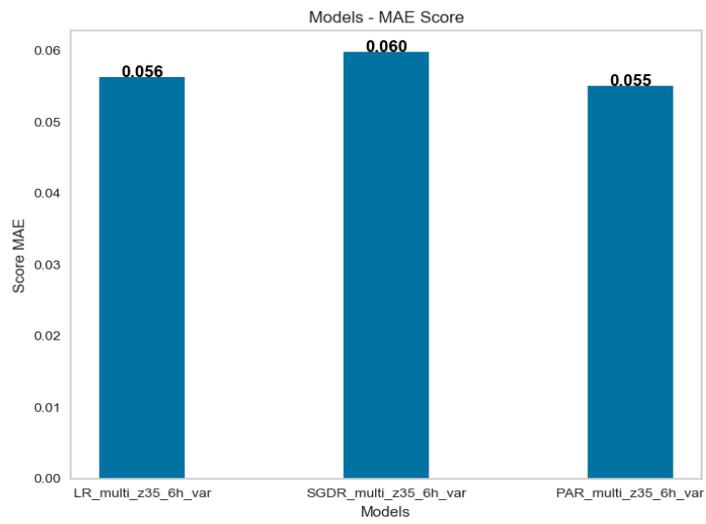


Figure 193. MAE Scores Linear models tested for predicting z35 Multivariate

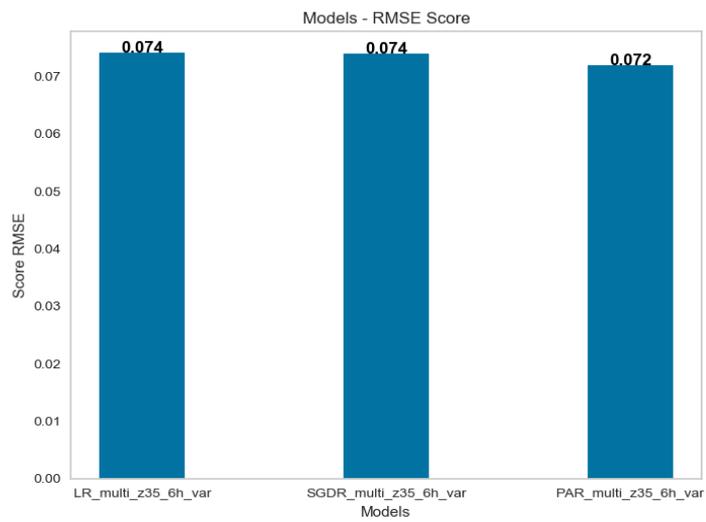


Figure 194. RMSE Scores Linear models tested for predicting z35 Multivariate

- 4) *Features used*: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

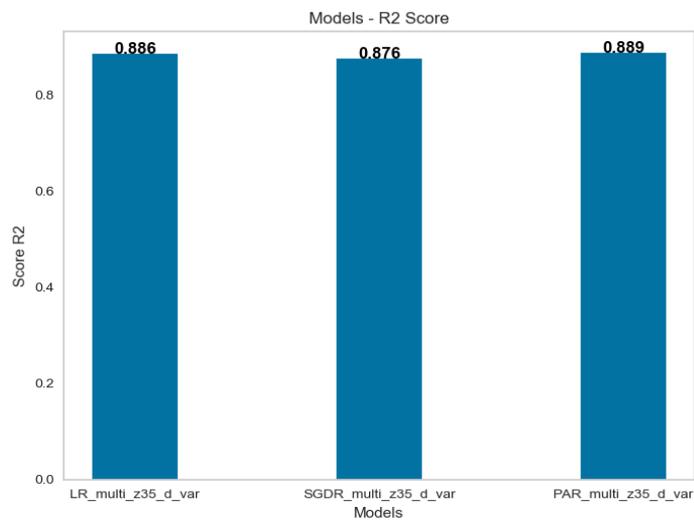


Figure 195. R2 Scores Linear models tested for predicting z35 Multivariate

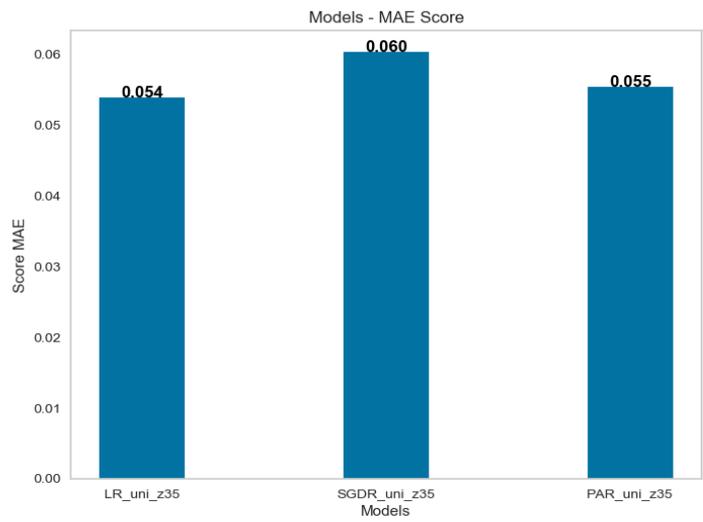


Figure 196. MAE Scores Linear models tested for predicting z35 Multivariate

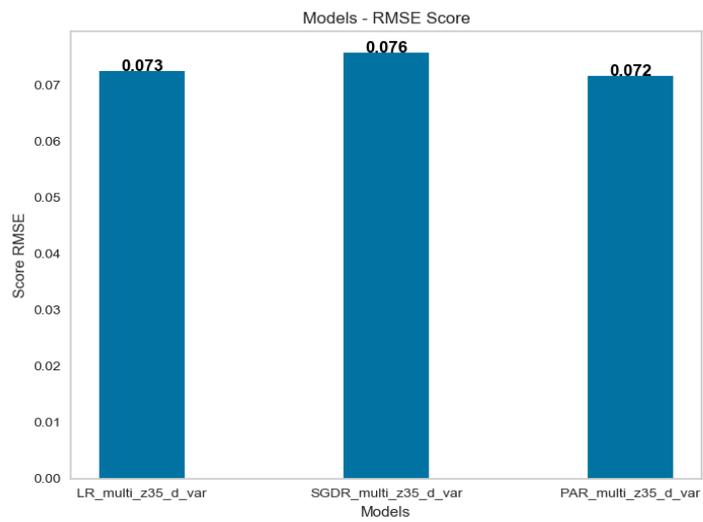


Figure 197. RMSE Scores Linear models tested for predicting z35 Multivariate

Considering all the error metrics score plot for each model realized to predict z35 (one Univariate and four Multivariate), the best model results to be the Linear Regression (LR) model in Univariate case:

Model name: "LR_uni";

Features used: z35

We report an example of prediction of z35 using the best model found:

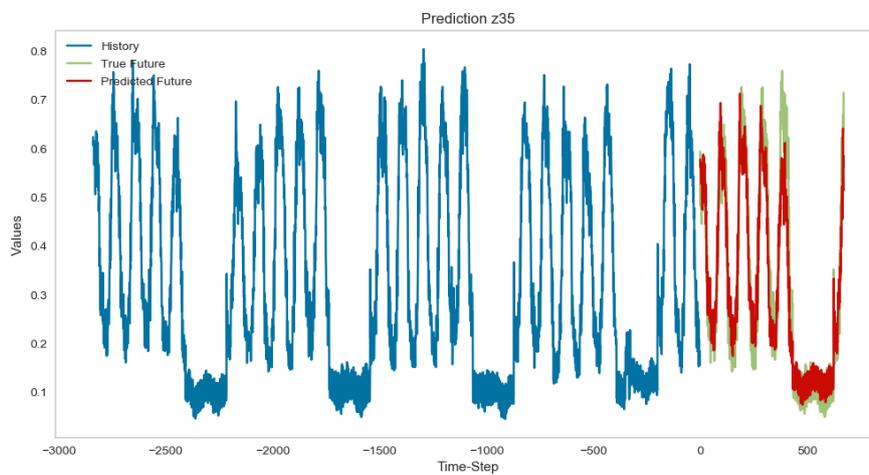


Figure 198. Example of one prediction of best Linear model for predicting z35

In the figure we have plotted a time window of past_history of z35 of 4 weeks than only one, but the prediction is made using only one week of past_history, so using the 672 timesteps before 0 point along x-axis.

- Prediction of **std**
 - *Univariate Models*
-

1) *Feature used: z35_6hours_std*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

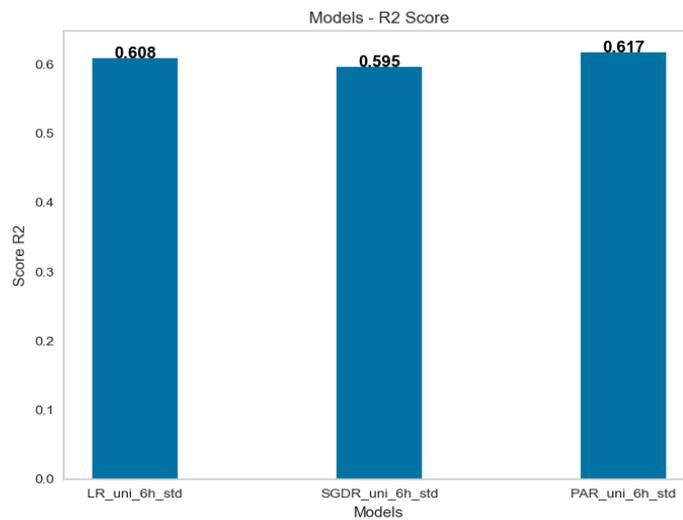


Figure 199. R2 Scores Linear models tested for predicting z35_std Univariate

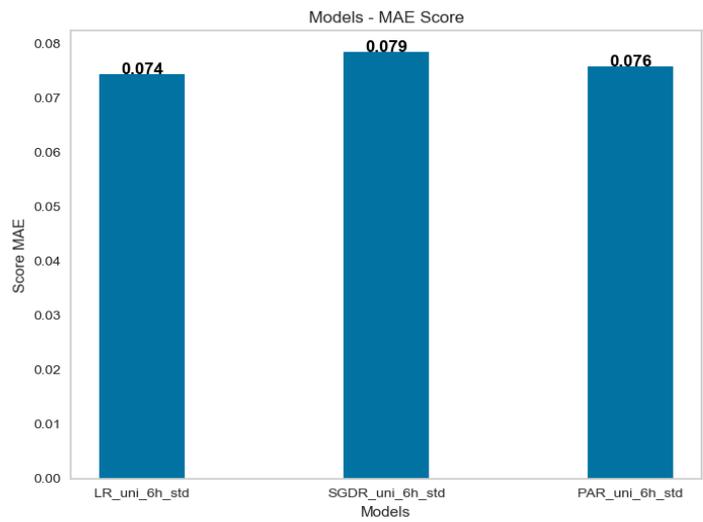


Figure 200. MAE Scores Linear models tested for predicting z35_std Univariate

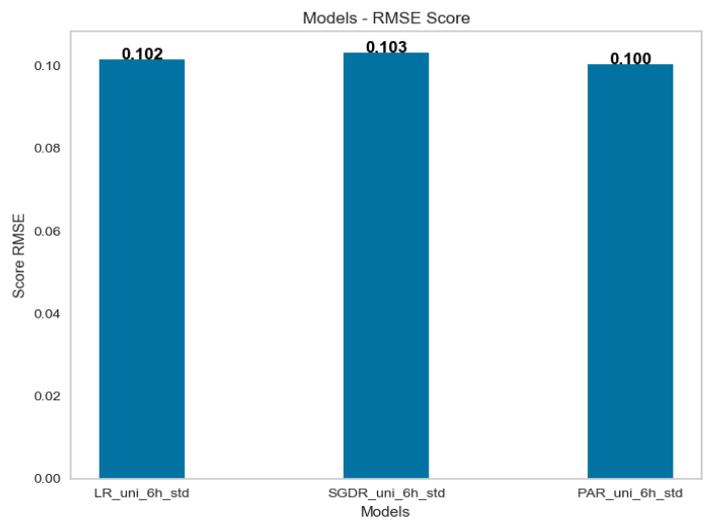


Figure 201. RMSE Scores Linear models tested for predicting z35_std Univariate

2) *Feature used: z35_daily_std*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

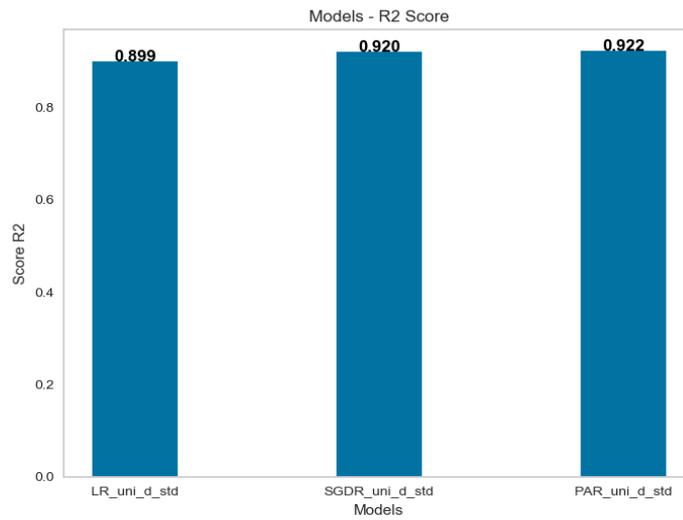


Figure 202. R2 Scores Linear models tested for predicting z35_std Univariate

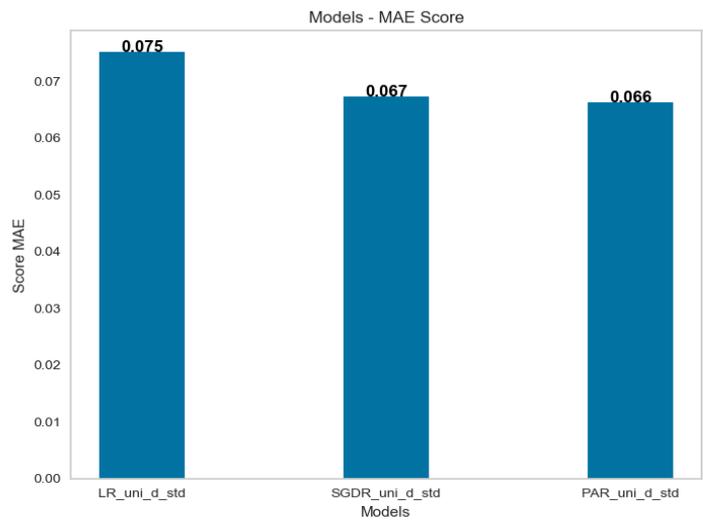


Figure 203. MAE Scores Linear models tested for predicting z35_std Univariate

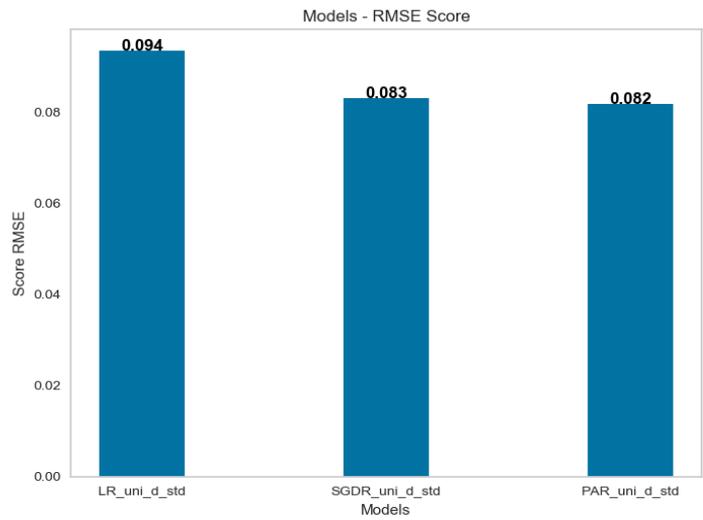


Figure 204. RMSE Scores Linear models tested for predicting z35_std Univariate

- *Multivariate Models*

- 1) *Feature used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday*
 - Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

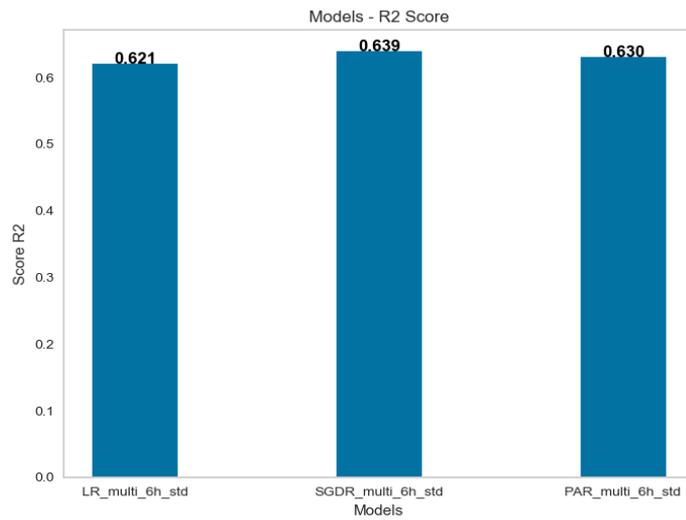


Figure 205. R2 Scores Linear models tested for predicting z35_std Multivariate

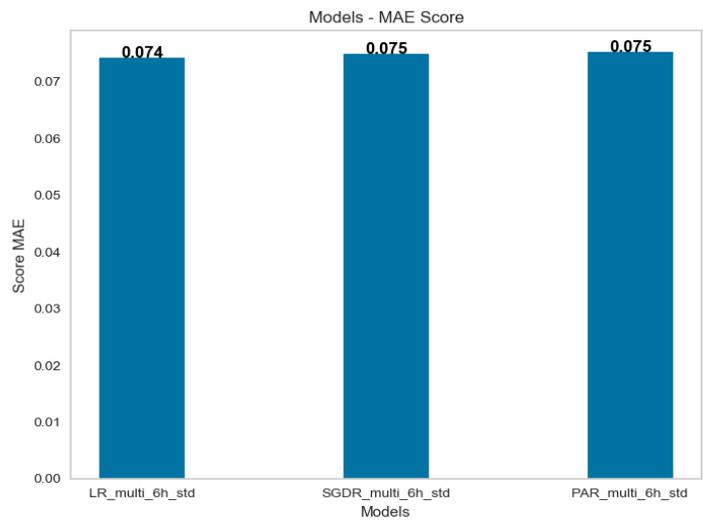


Figure 206. MAE Scores Linear models tested for predicting z35_std Multivariate

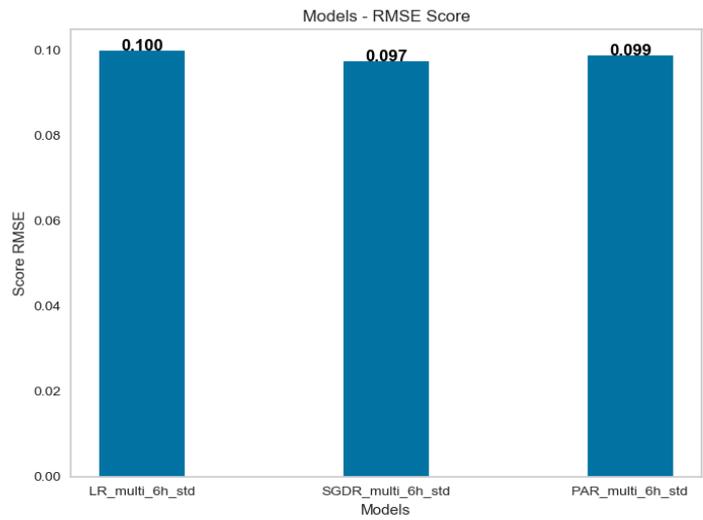


Figure 207. RMSE Scores Linear models tested for predicting z35_std Multivariate

- 2) *Feature used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

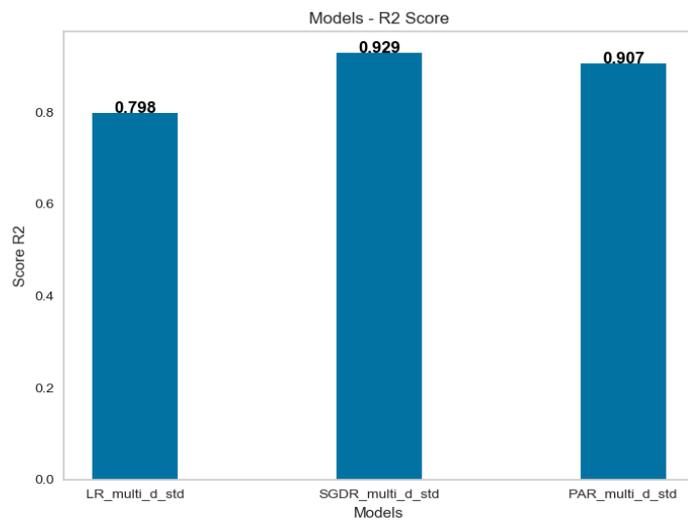


Figure 208. R2 Scores Linear models tested for predicting z35_std Multivariate

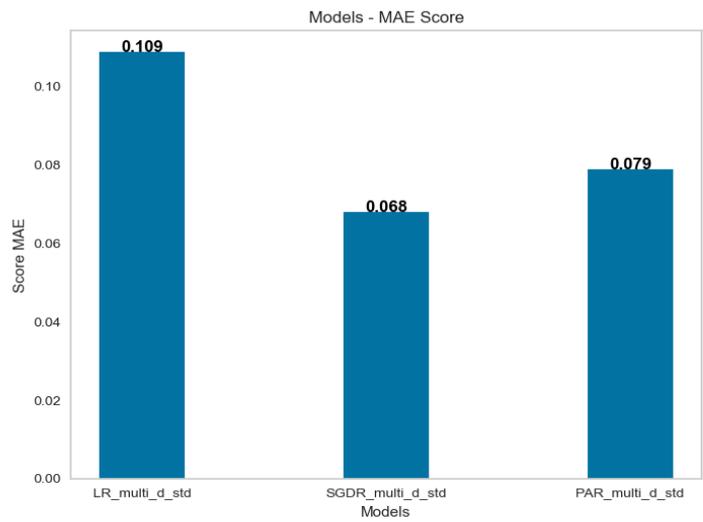


Figure 209. MAE Scores Linear models tested for predicting z35_std Multivariate

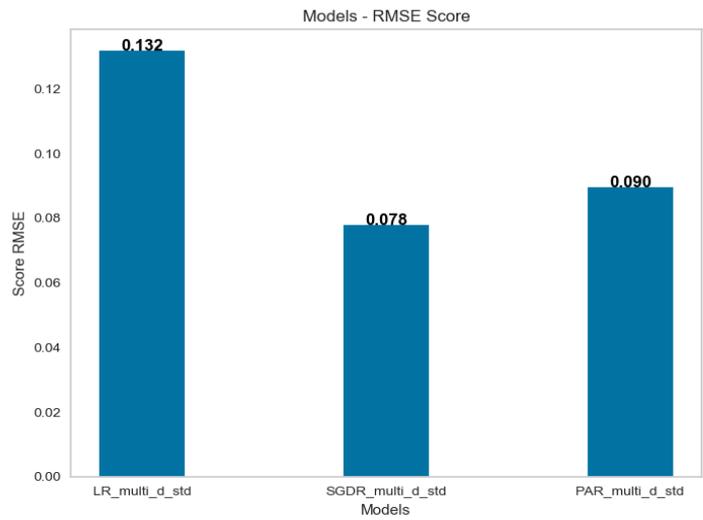


Figure 210. RMSE Scores Linear models tested for predicting z35_std Multivariate

Considering all the error metrics score plot for each model realized to predict z35 standard deviation (two Univariate and two Multivariate), the best model results to be the Stochastic Gradient Descent Regressor (SGDR) model in Multivariate case:

Model name: "SGDR_multi_d_std";

Features used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

We report an example of prediction of z35 standard deviation using the best model found:

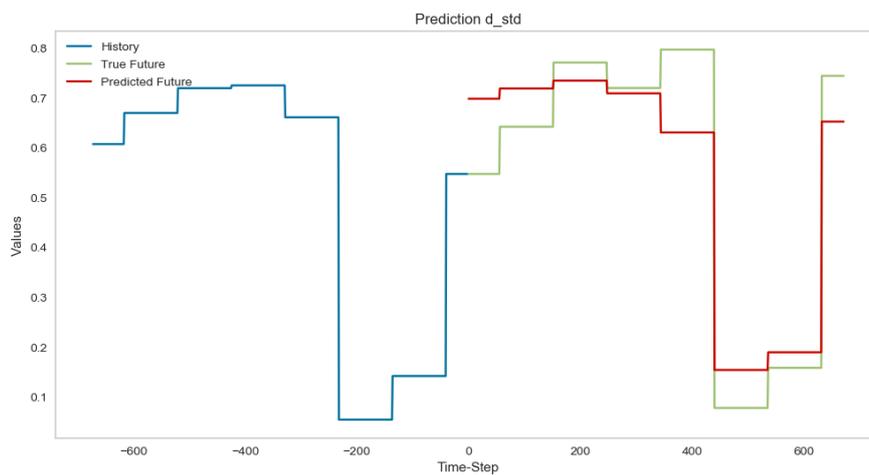


Figure 211. Example of one prediction of best Linear model for predicting z35_std

- Prediction of **var**
 - *Univariate Models*
-

1) *Feature used: z35_6hours_var*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

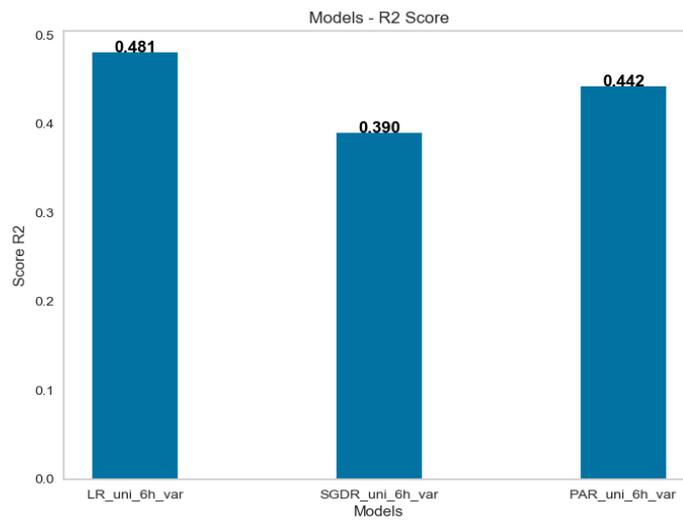


Figure 212. R2 Scores Linear models tested for predicting z35_var Univariate

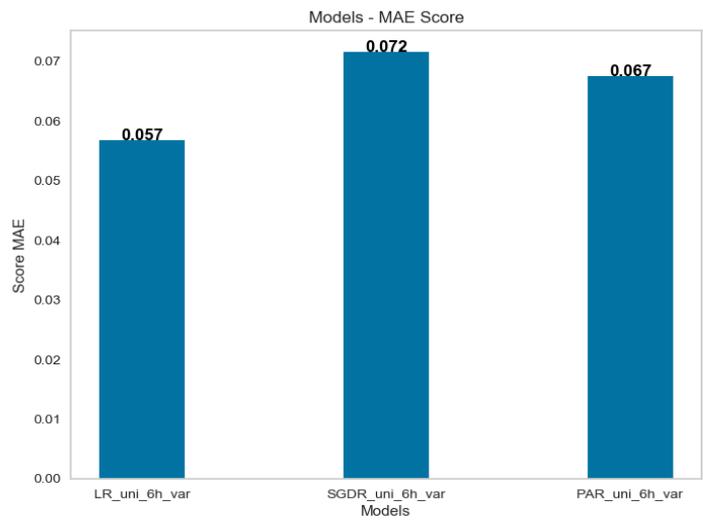


Figure 213. MAE Scores Linear models tested for predicting z35_var Univariate

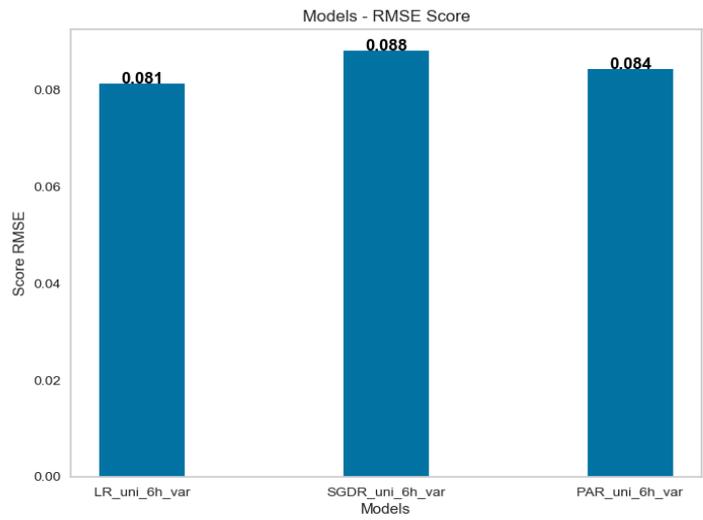


Figure 214. RMSE Scores Linear models tested for predicting z35_var Univariate

2) *Feature used: z35_daily_var*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

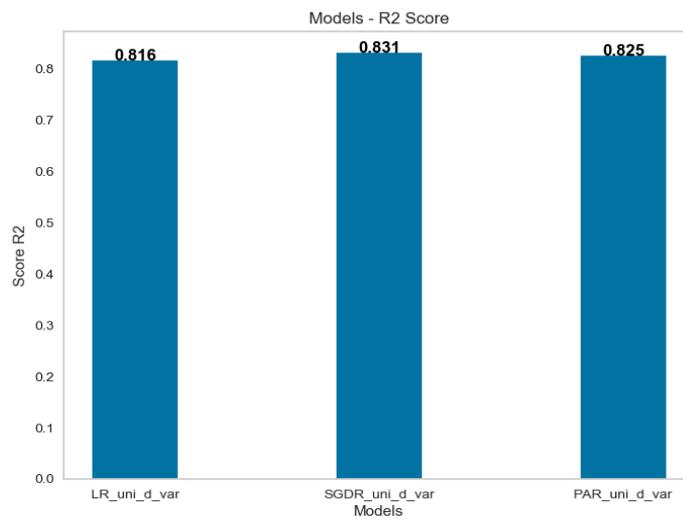


Figure 215. R2 Scores Linear models tested for predicting z35_var Univariate

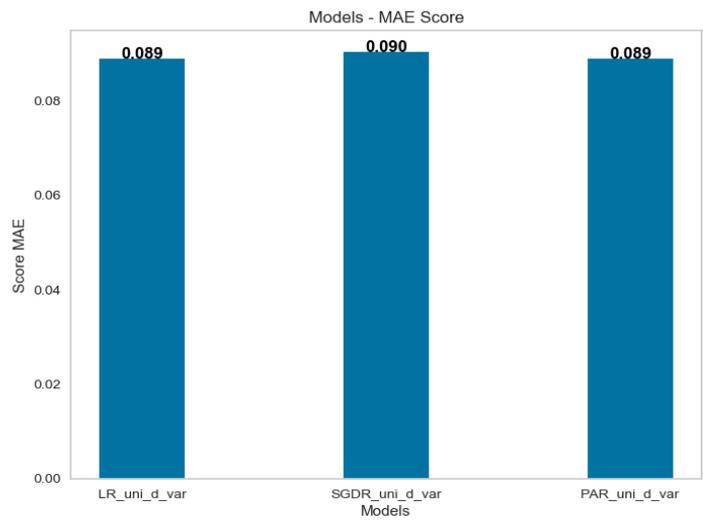


Figure 216. MAE Scores Linear models tested for predicting z35_var Univariate

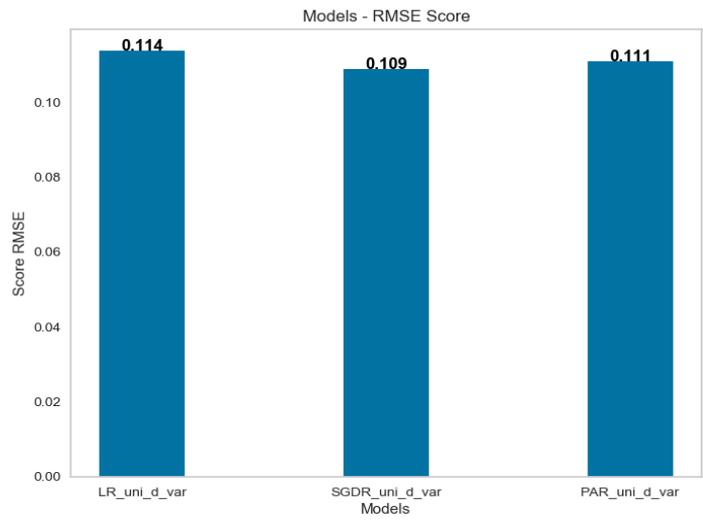


Figure 217. RMSE Scores Linear models tested for predicting z35_var Univariate

- *Multivariate Models*

- 1) *Feature used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

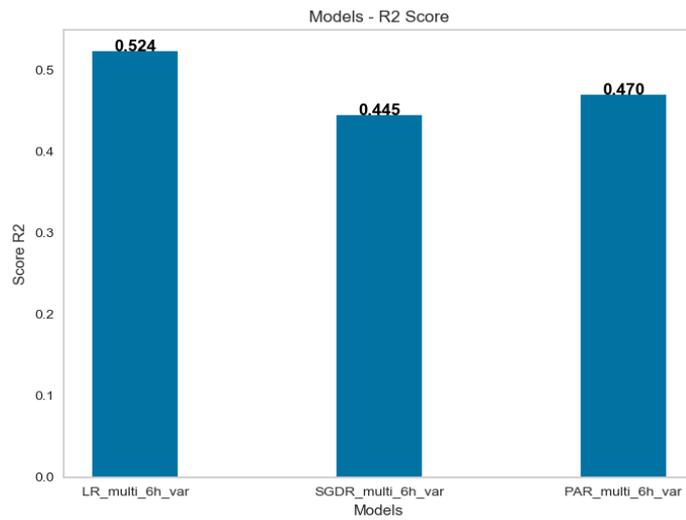


Figure 218. R2 Scores Linear models tested for predicting z35_var Multivariate

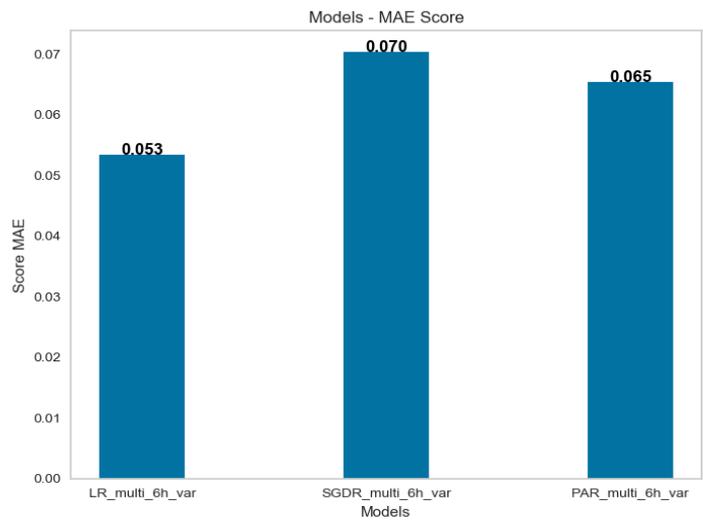


Figure 219. MAE Scores Linear models tested for predicting z35_var Multivariate

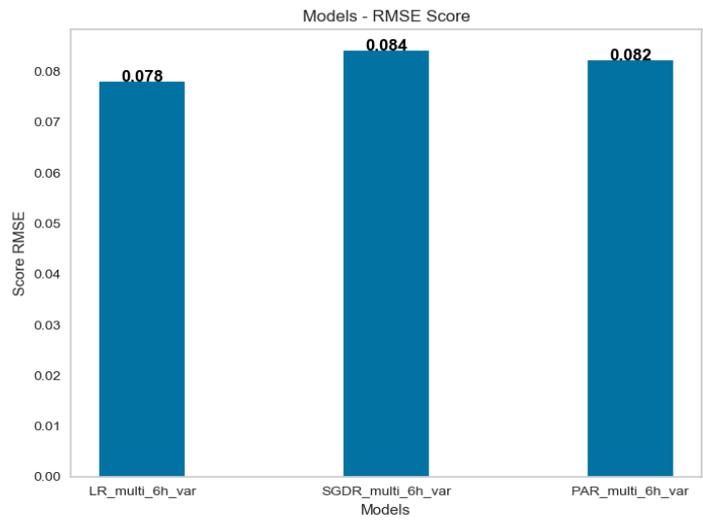


Figure 220. RMSE Scores Linear models tested for predicting z35_var Multivariate

- 2) *Feature used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

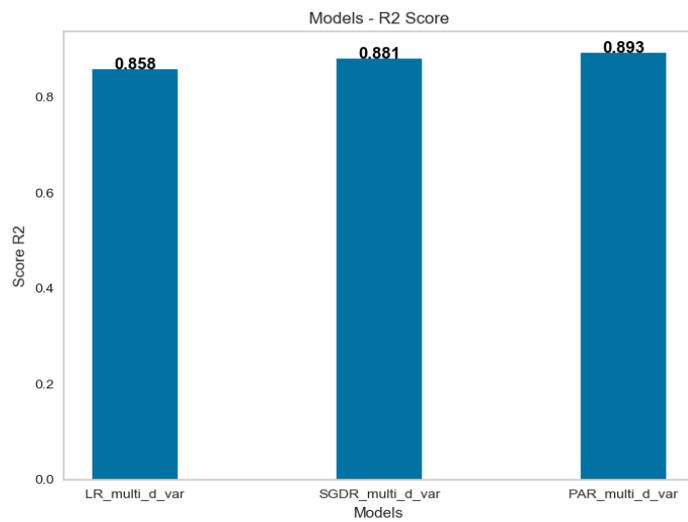


Figure 221. R2 Scores Linear models tested for predicting z35_var Multivariate

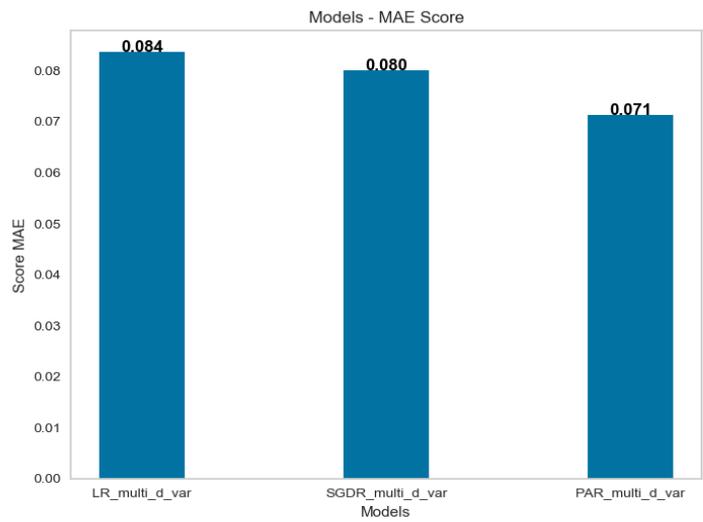


Figure 222. MAE Scores Linear models tested for predicting z35_var Multivariate

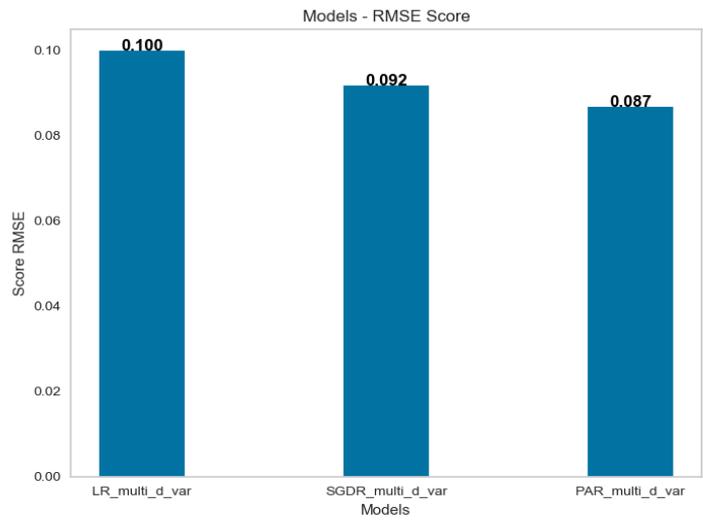


Figure 223. RMSE Scores Linear models tested for predicting z35_var Multivariate

Considering all the error metrics score plot for each model realized to predict z35 variance (two Univariate and two Multivariate), the best model results to be the Linear Regression (LR) model in Multivariate case:

Model name: "LR_multi_6h_var";

Features used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday

We report an example of prediction of z35 variance using the best model found:

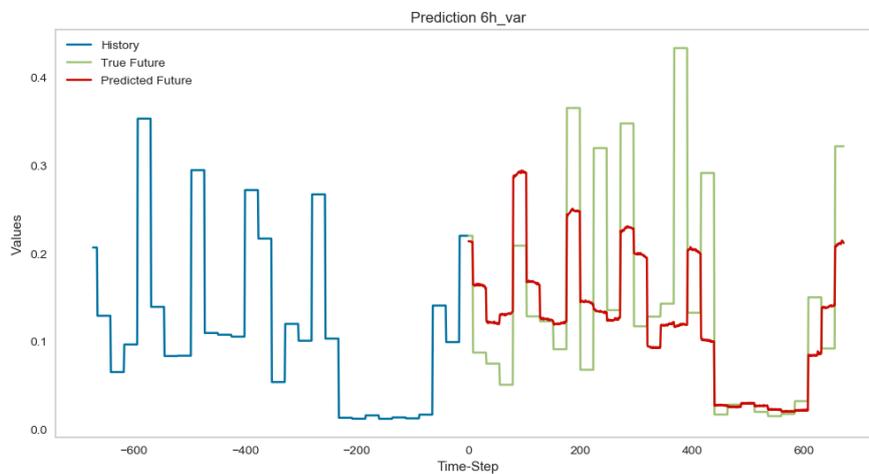


Figure 224. Example of one prediction of best Linear model for predicting z35_var



- Prediction of **min**
 - *Univariate Models*
-

1) *Feature used: z35_6hours_min*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

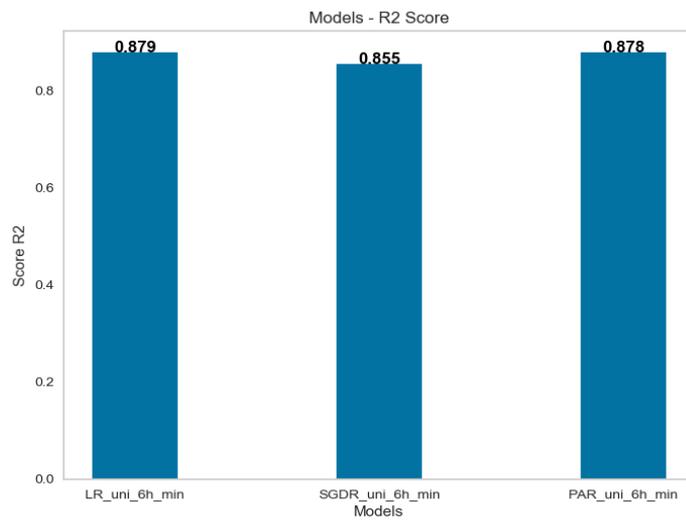


Figure 225. R2 Scores Linear models tested for predicting z35_min Univariate

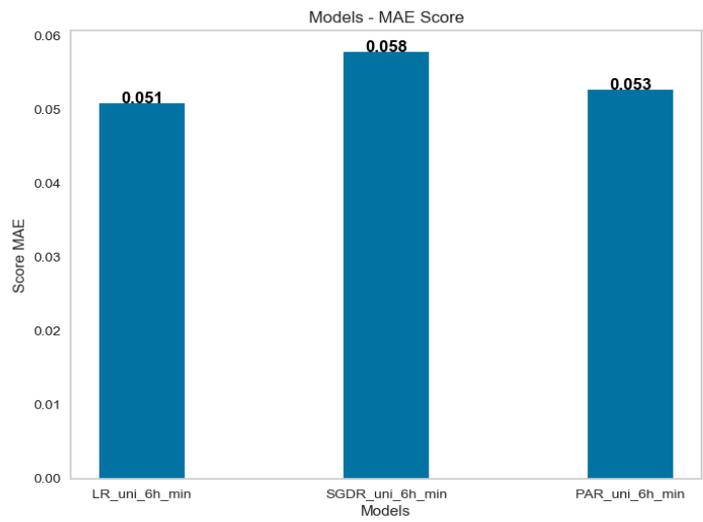


Figure 226. MAE Scores Linear models tested for predicting z35_min Univariate

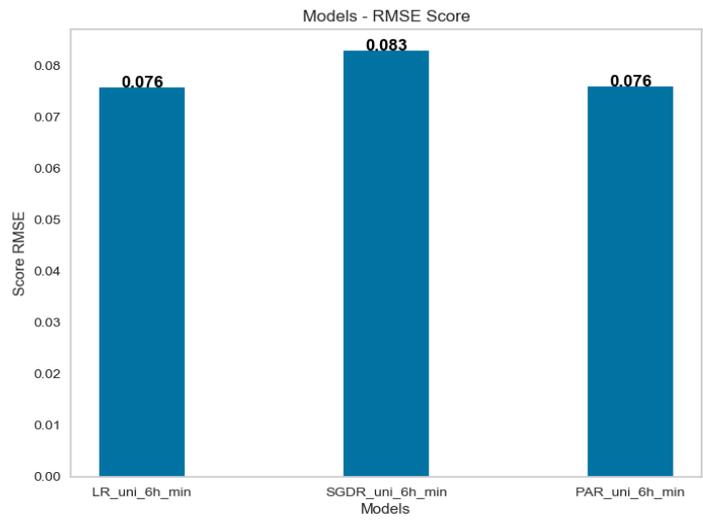


Figure 227. RMSE Scores Linear models tested for predicting z35_min Univariate

2) *Feature used: z35_daily_min*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

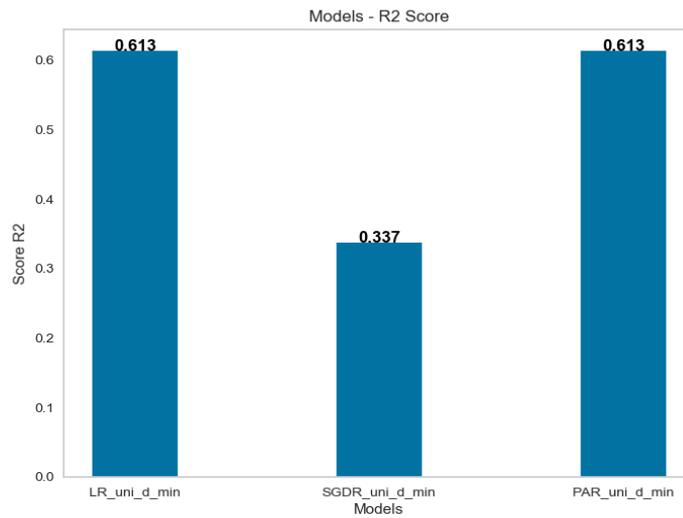


Figure 228. R2 Scores Linear models tested for predicting z35_min Univariate

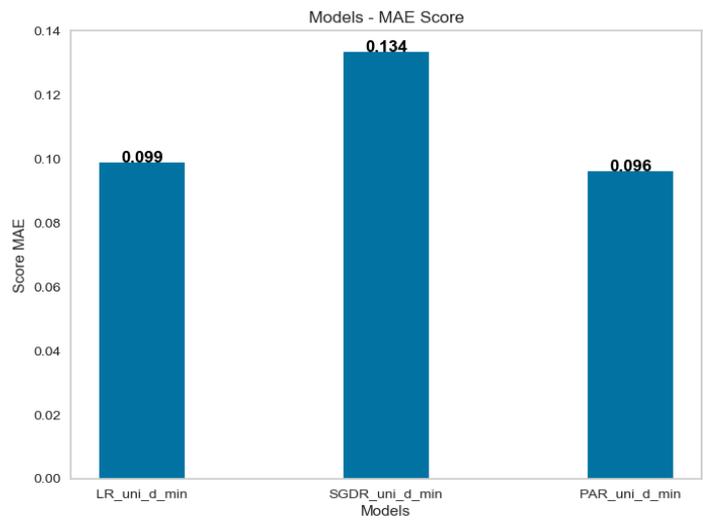


Figure 229. MAE Scores Linear models tested for predicting z35_min Univariate

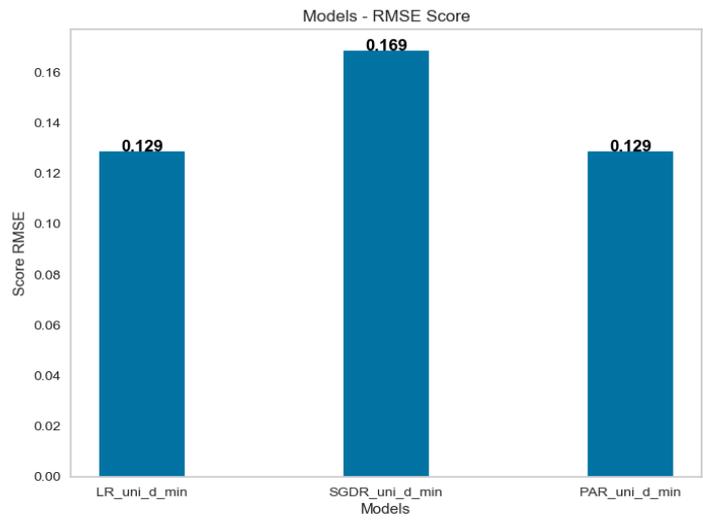


Figure 230. RMSE Scores Linear models tested for predicting z35_min Univariate

- *Multivariate Models*

- 1) *Feature used*: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday
 - Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

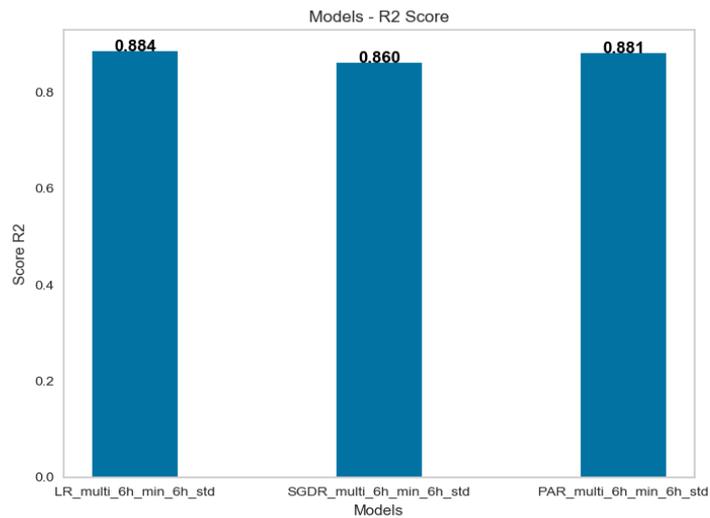


Figure 231. R2 Scores Linear models tested for predicting z35_min Multivariate

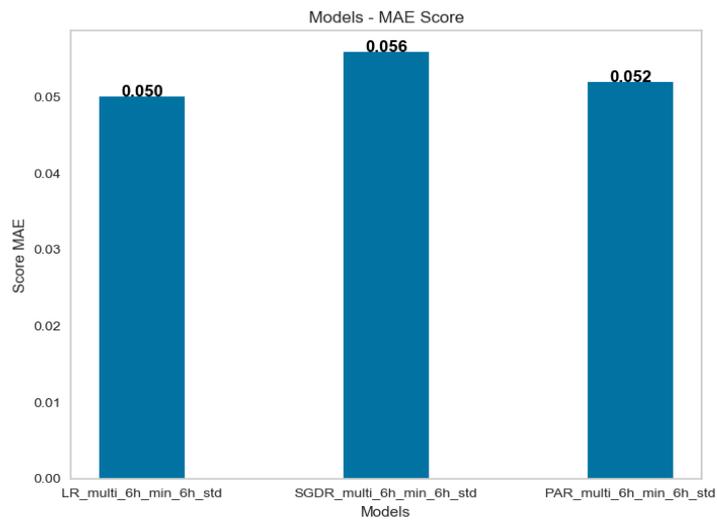


Figure 232. MAE Scores Linear models tested for predicting z35_min Multivariate

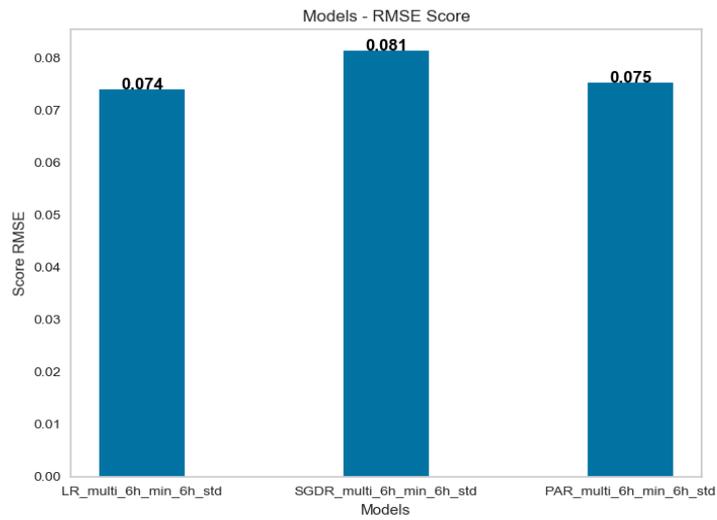


Figure 233. RMSE Scores Linear models tested for predicting z35_min Multivariate

- 2) *Feature used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

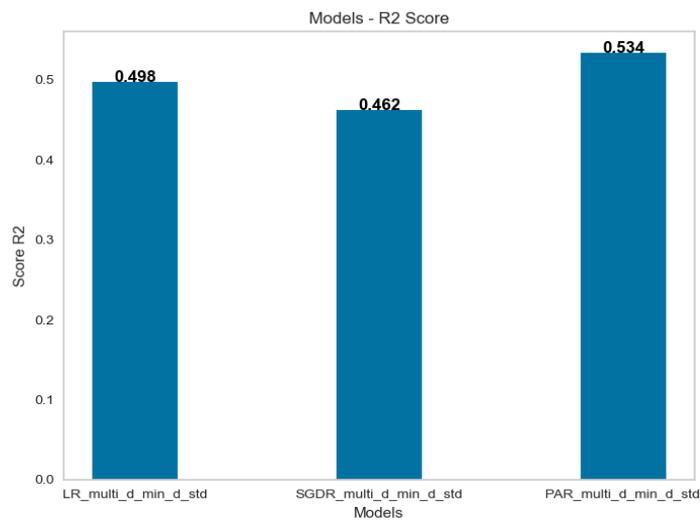


Figure 234. R2 Scores Linear models tested for predicting z35_min Multivariate

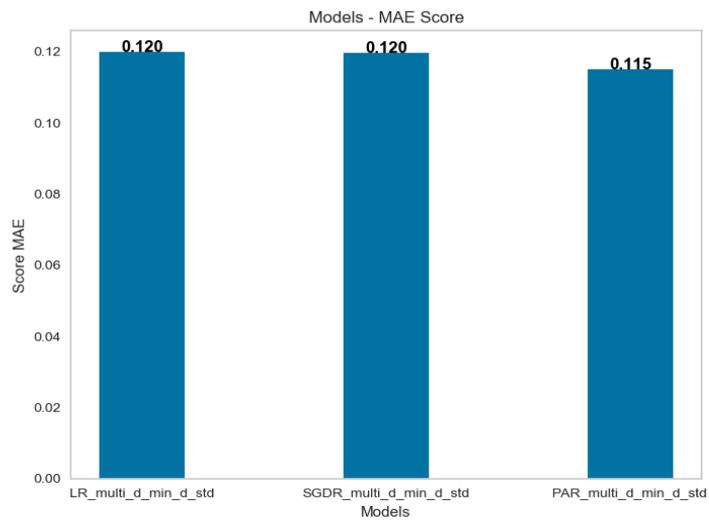


Figure 235. MAE Scores Linear models tested for predicting z35_min Multivariate

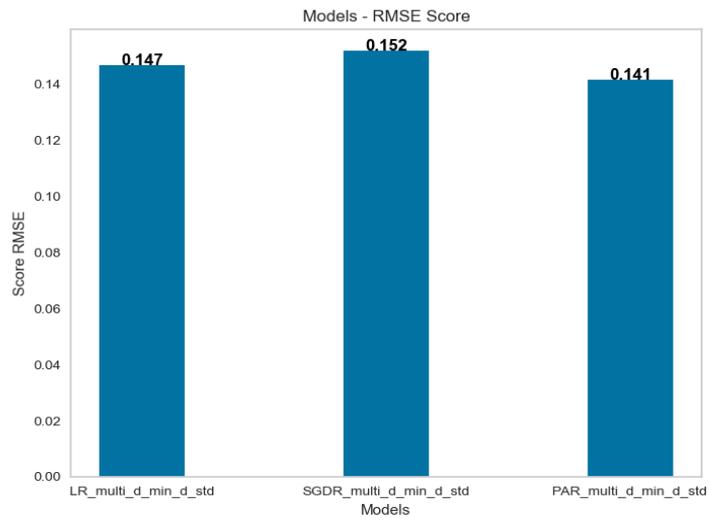


Figure 236. RMSE Scores Linear models tested for predicting z35_min Multivariate

- 3) *Feature used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

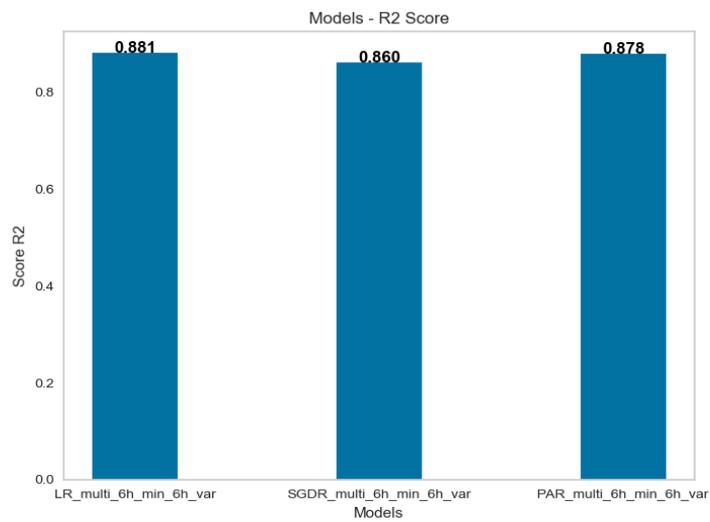


Figure 237. R2 Scores Linear models tested for predicting z35_min Multivariate

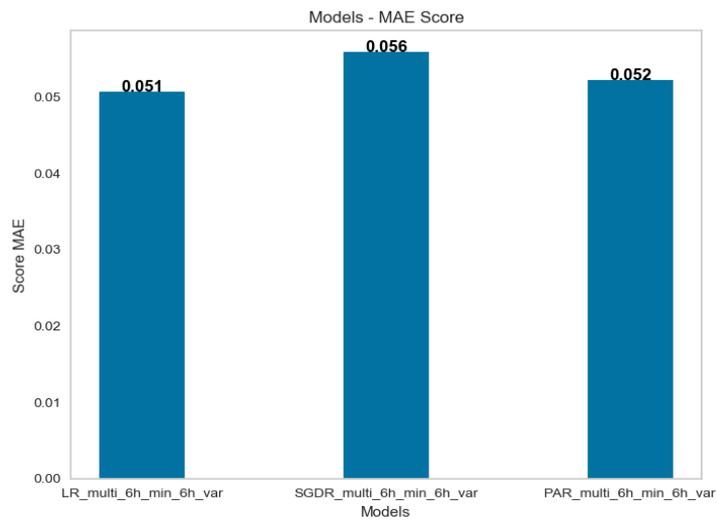


Figure 238. MAE Scores Linear models tested for predicting z35_min Multivariate

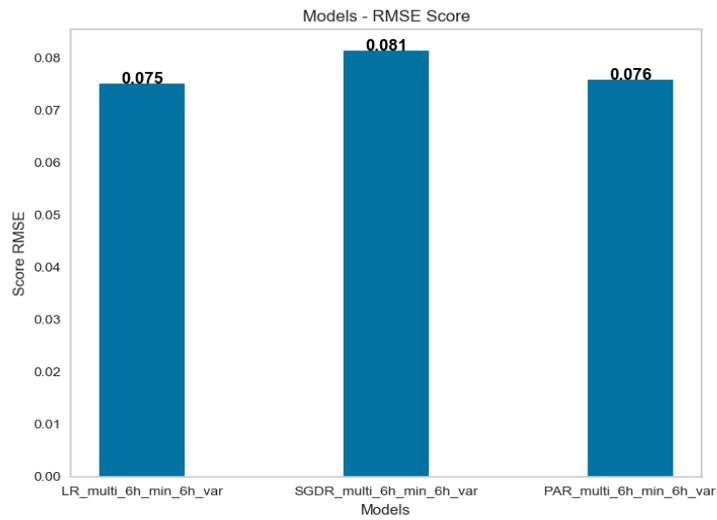


Figure 239. RMSE Scores Linear models tested for predicting z35_min Multivariate

- 4) *Feature used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

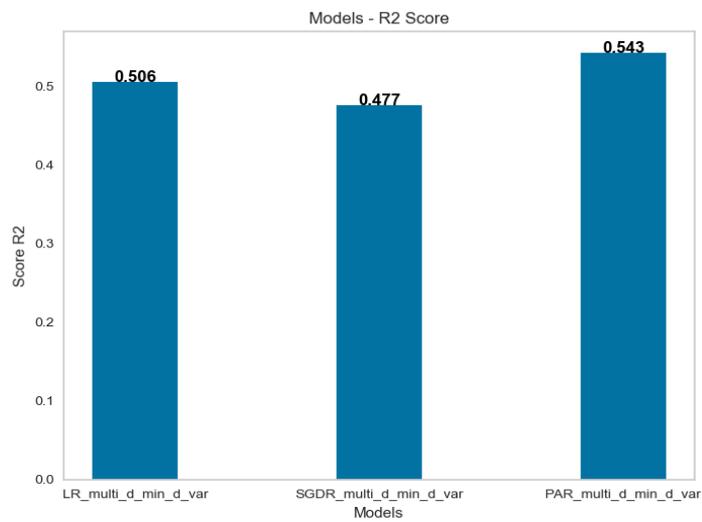


Figure 240. R2 Scores Linear models tested for predicting z35_min Multivariate

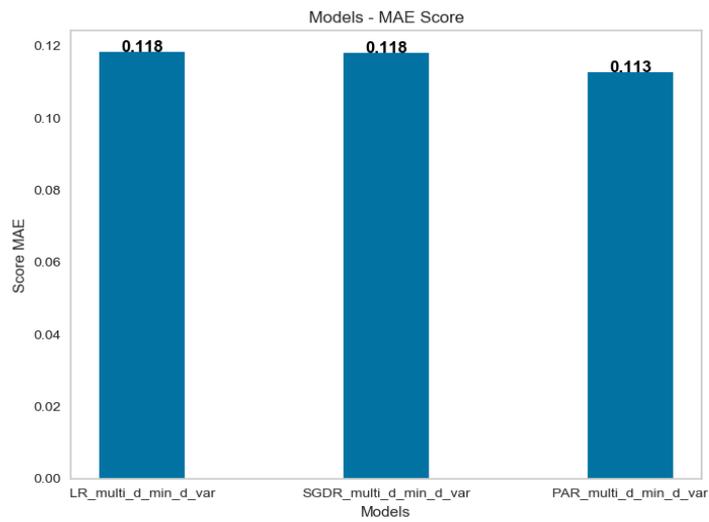


Figure 241. MAE Scores Linear models tested for predicting z35_min Multivariate

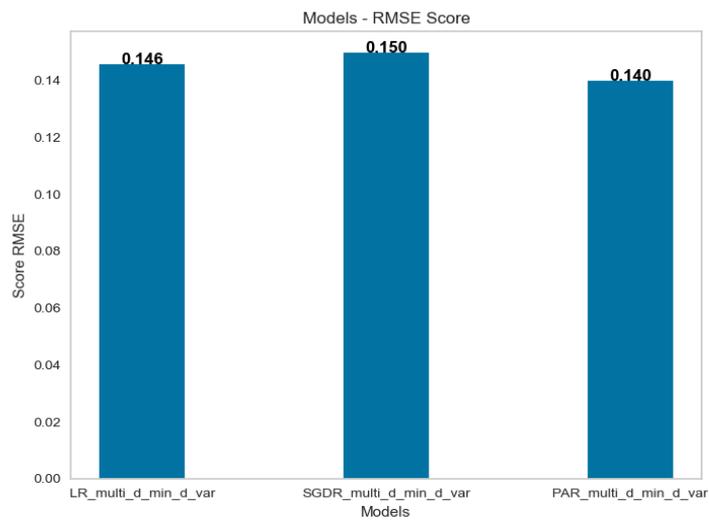


Figure 242. RMSE Scores Linear models tested for predicting z35_min Multivariate

Considering all the error metrics score plot for each model realized to predict z35 minimal value (two Univariate and four Multivariate), the best model results to be the Linear Regression (LR) model in Multivariate case:

Model name: "LR_multi_6h_min_6h_std";

Features used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday

We report an example of prediction of z35 minimal value using the best model found:

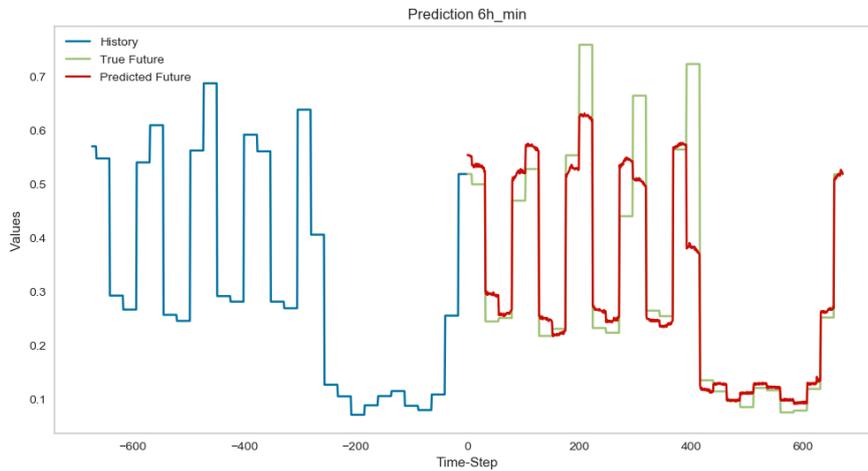


Figure 243. Example of one prediction of best Linear model for predicting z35_min

- Prediction of **max**
 - *Univariate Models*
-

1) *Feature used: z35_6hours_max*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

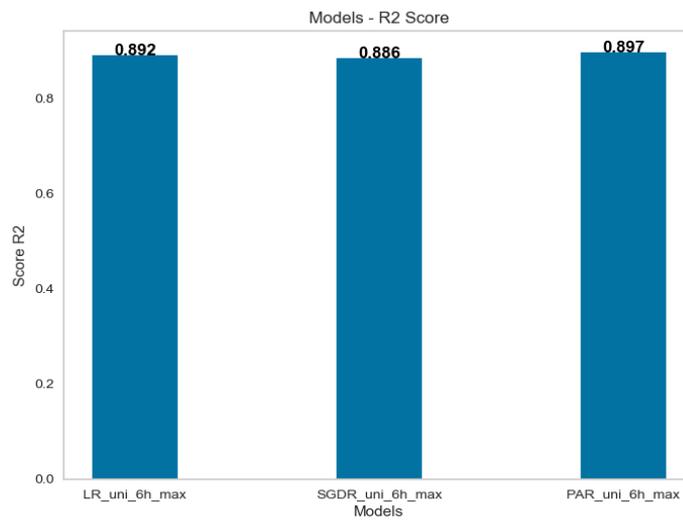


Figure 244. R2 Scores Linear models tested for predicting z35_max Univariate

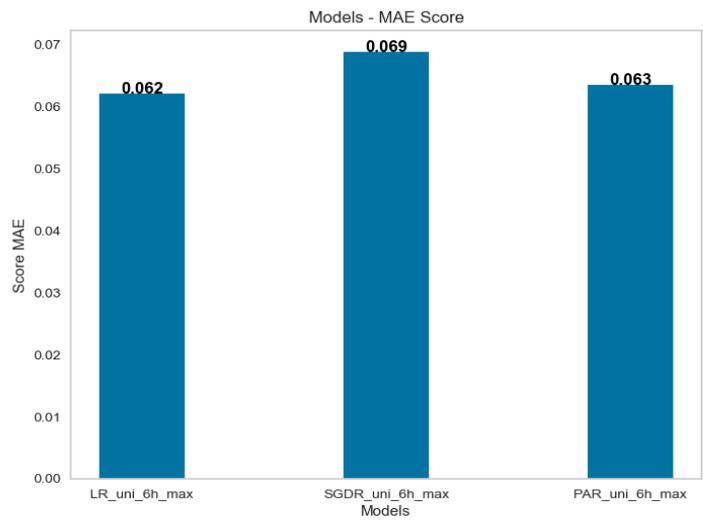


Figure 245. MAE Scores Linear models tested for predicting z35_max Univariate

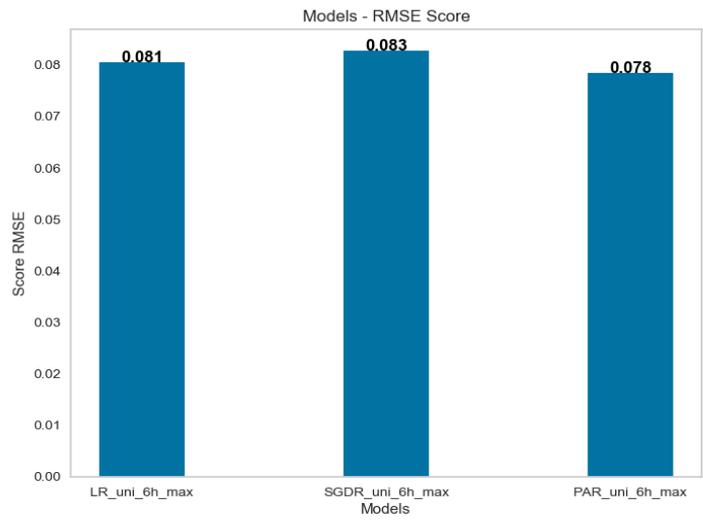


Figure 246. RMSE Scores Linear models tested for predicting z35_max Univariate

2) *Feature used: z35_daily_max*

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

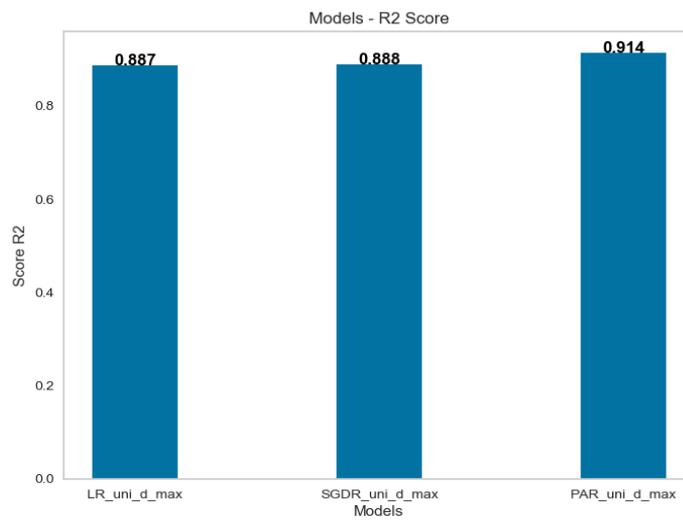


Figure 247. R2 Scores Linear models tested for predicting z35_max Univariate

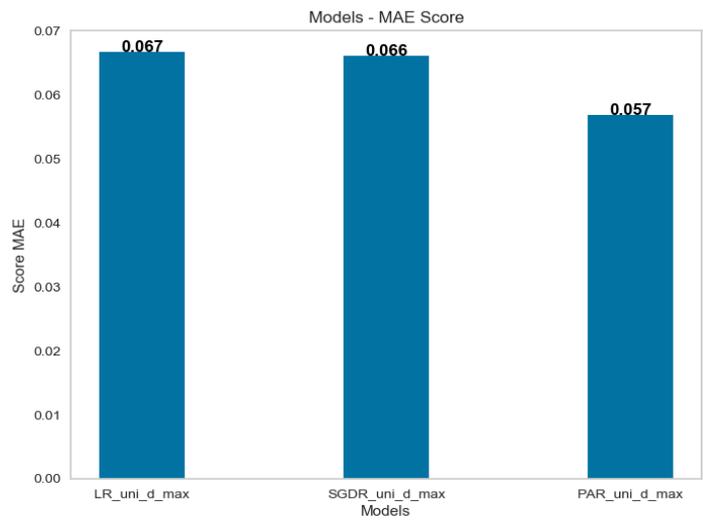


Figure 248. MAE Scores Linear models tested for predicting z35_max Univariate

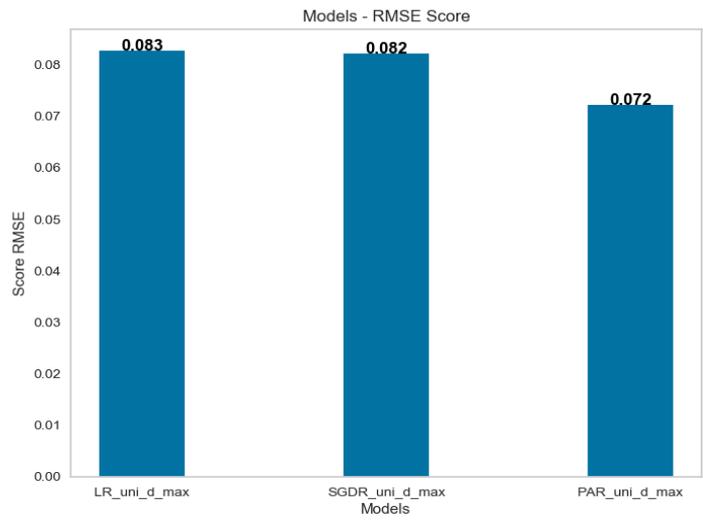


Figure 249. RMSE Scores Linear models tested for predicting z35_max Univariate

- *Multivariate Models*

- 1) *Feature used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday*
 - Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

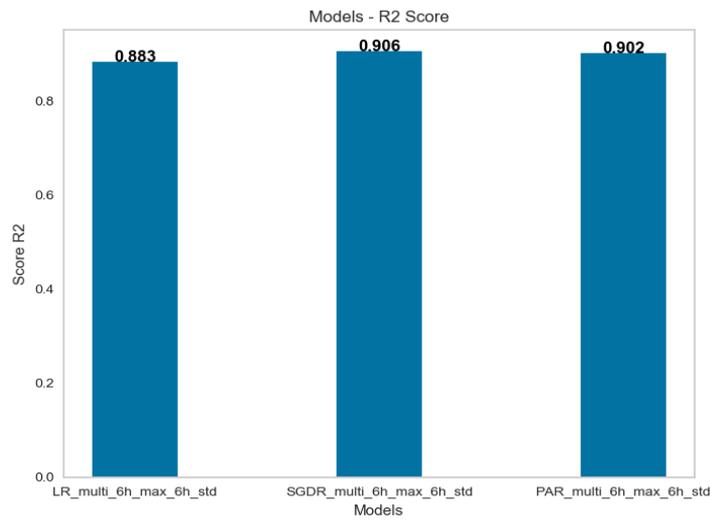


Figure 250. R2 Scores Linear models tested for predicting z35_max Multivariate

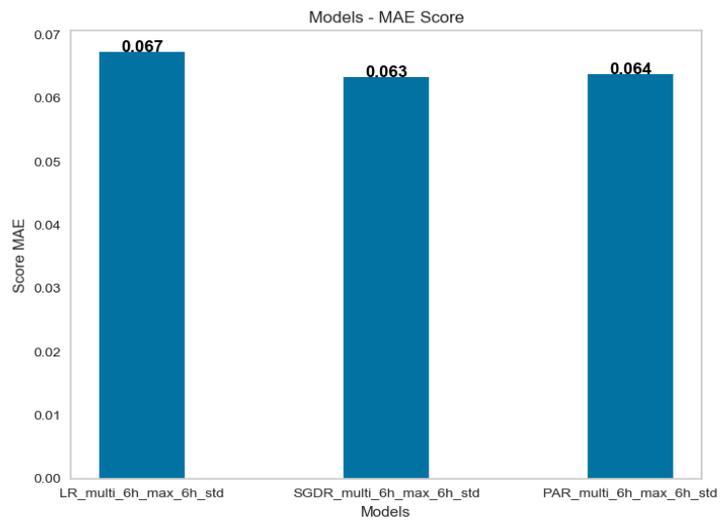


Figure 251. MAE Scores Linear models tested for predicting z35_max Multivariate

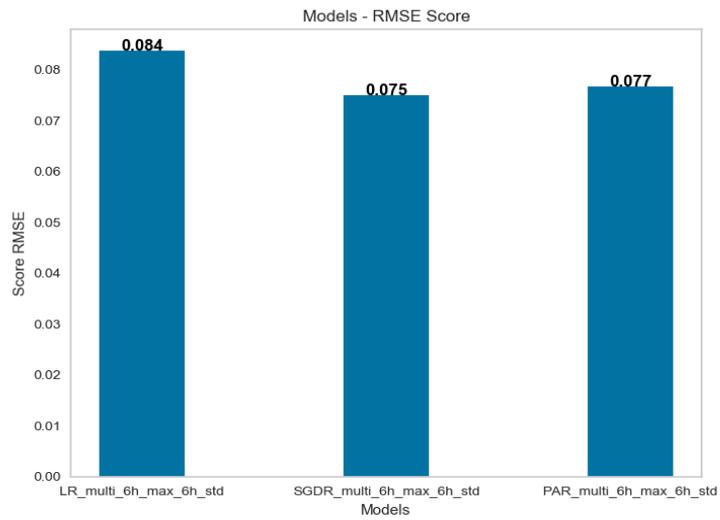


Figure 252. RMSE Scores Linear models tested for predicting z35_max Multivariate

2) *Feature used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday*

- Input shape $x = (n_timesteps=672, n_features=6)$
- True_target shape $y = (n_timesteps=672, n_features=6)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

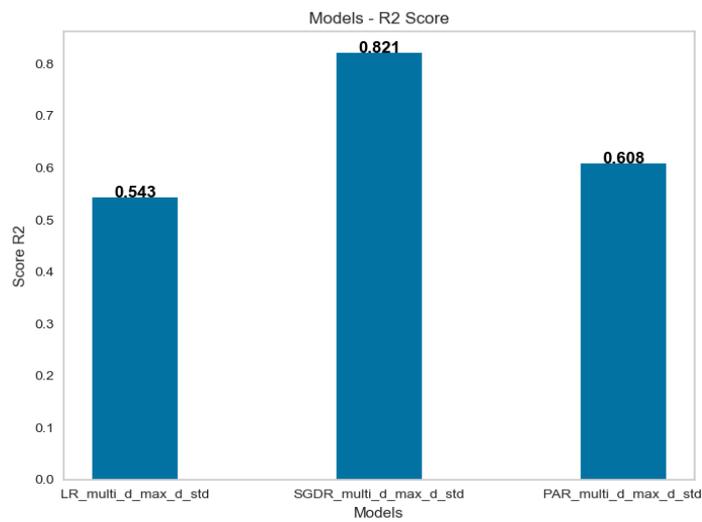


Figure 253. R2 Scores Linear models tested for predicting z35_max Multivariate

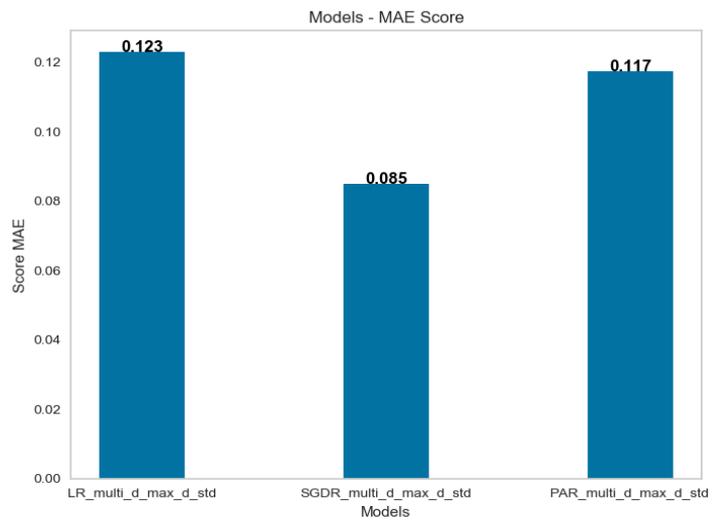


Figure 254. MAE Scores Linear models tested for predicting z35_max Multivariate

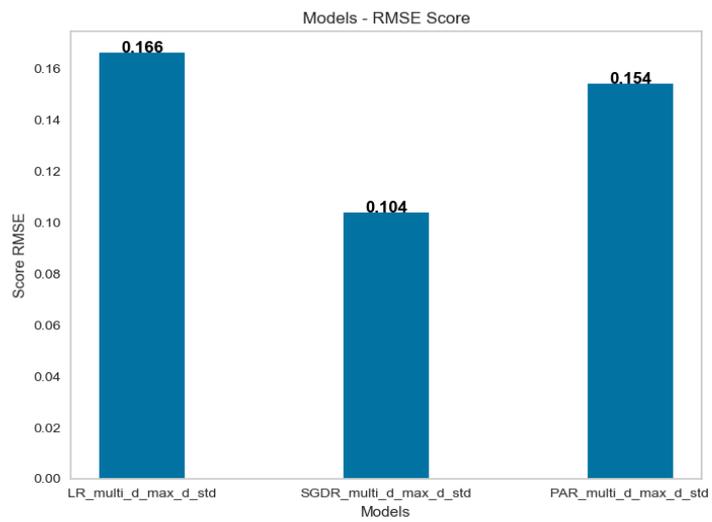


Figure 255. RMSE Scores Linear models tested for predicting z35_max Multivariate

- 3) *Feature used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

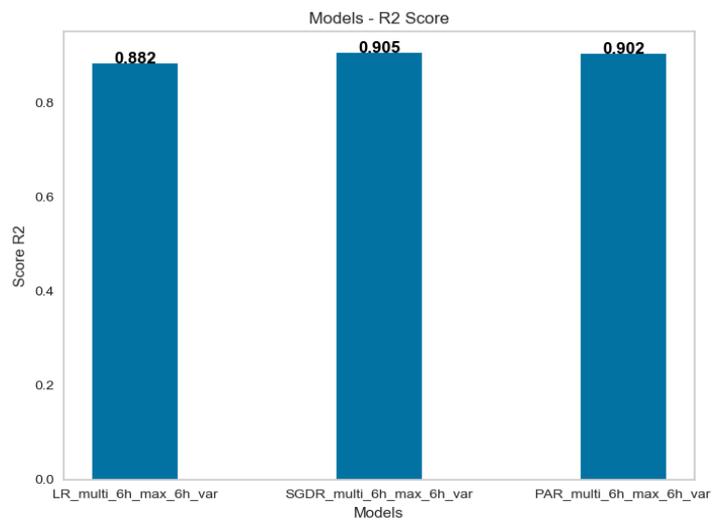


Figure 256. R2 Scores Linear models tested for predicting z35_max Multivariate

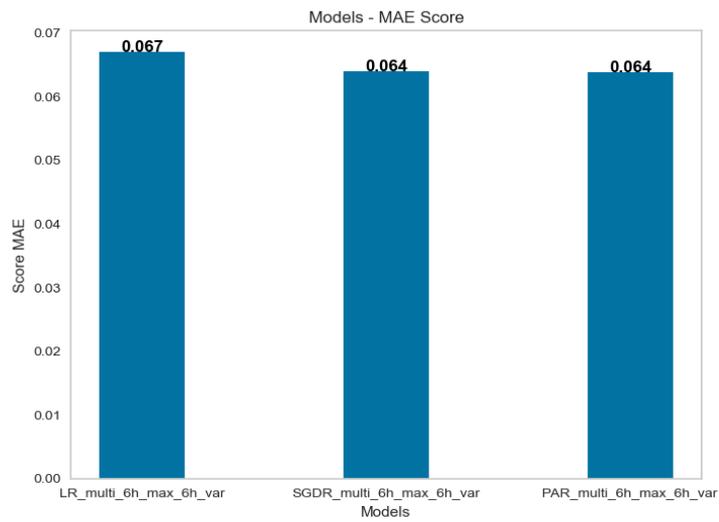


Figure 257. MAE Scores Linear models tested for predicting z35_max Multivariate

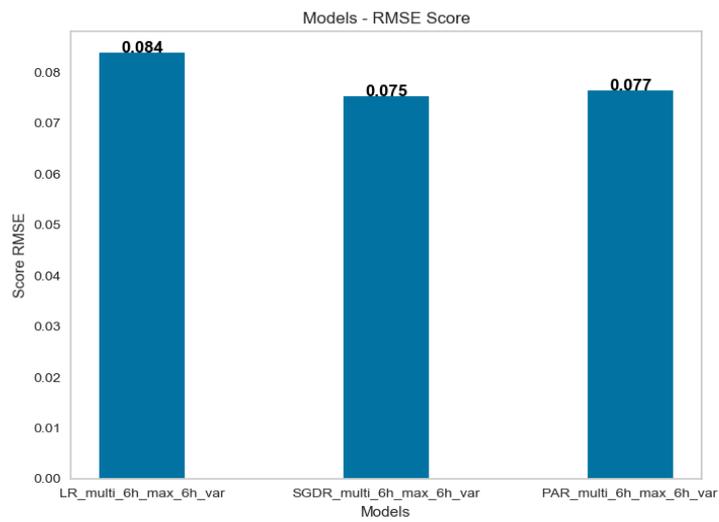


Figure 258. RMSE Scores Linear models tested for predicting z35_max Multivariate

- 4) *Feature used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday*
- Input shape $x = (n_timesteps=672, n_features=6)$
 - True_target shape $y = (n_timesteps=672, n_features=6)$
 - Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

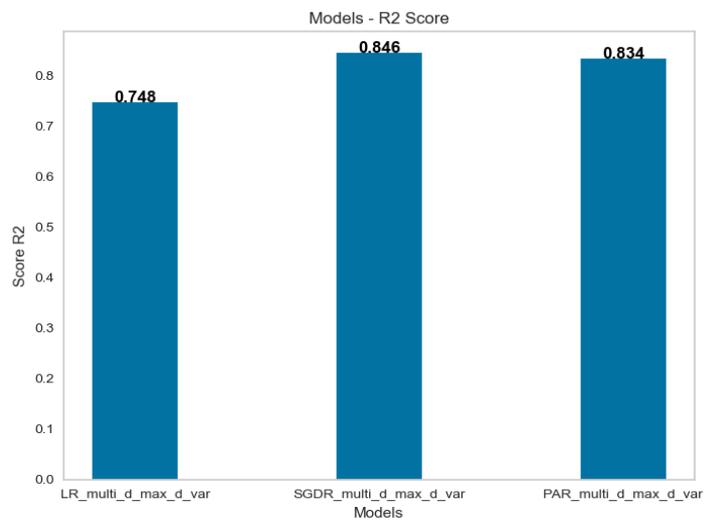


Figure 259. R2 Scores Linear models tested for predicting z35_max Multivariate

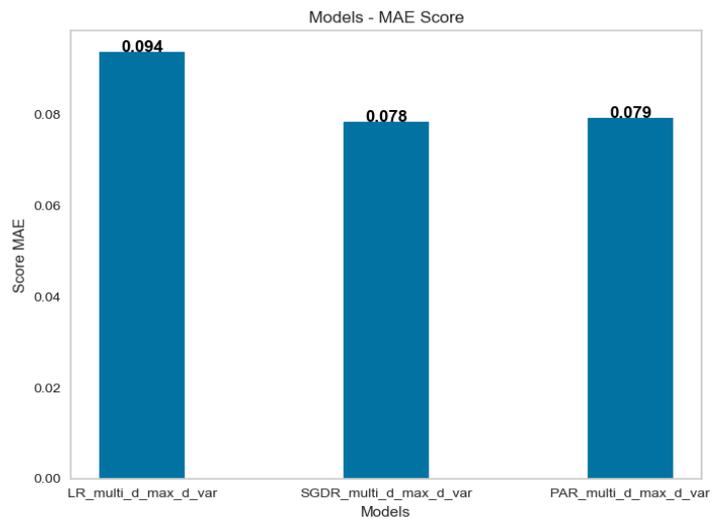


Figure 260. MAE Scores Linear models tested for predicting z35_max Multivariate

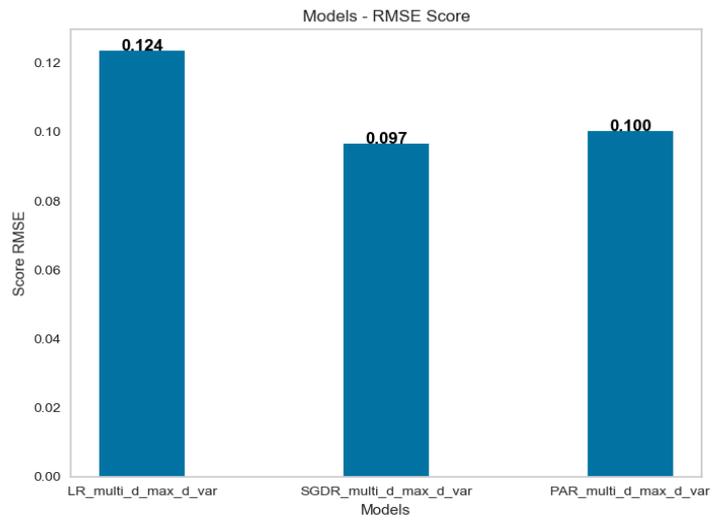


Figure 261. RMSE Scores Linear models tested for predicting z35_max Multivariate

Considering all the error metrics score plot for each model realized to predict z35 maximal value (two Univariate and four Multivariate), the best model results to be the Passive Aggressive Regressor (PAR) model in Univariate case:

Model name: "PAR_uni_d_max";

Features used: z35_daily_max

We report an example of prediction of z35 maximal value using the best model found:

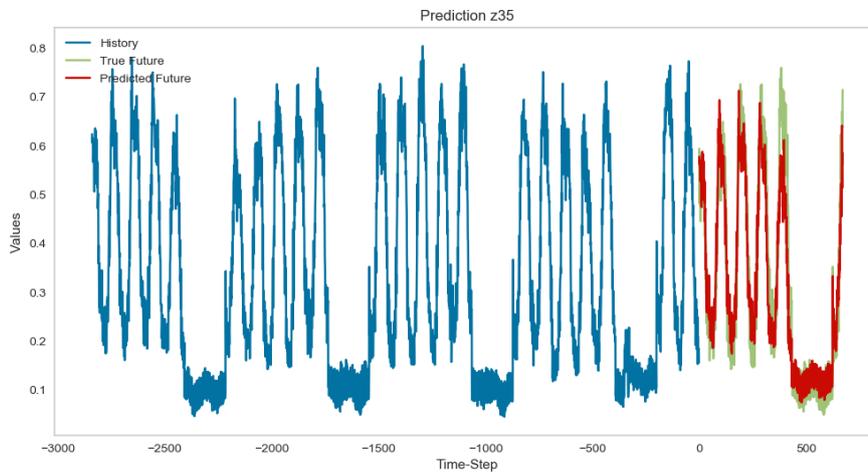


Figure 262. Example of one prediction of best Linear model for predicting z35_max

In the figure we have plotted a time window of past_history of z35_daily_max of 4 weeks than only one, but the prediction is made using only one week of past_history, so using the 672 timesteps before 0 point along x-axis.



After we have tested all the models and calculated the error metrics score, we can summarize the best Linear Regression models found for each features of interest:

FEATURE PREDICTED	CASE AND FEATURES USED	MODEL
z35	UNIVARIATE (z35)	Linear Regression (LR_uni_z35)
z35 Standard Deviation (d_std)	MULTIVARIATE (z35, d_mean, d_std, d_min, d_max, weekday)	Stochastic Gradient Descent Regressor (SGDR_multi_d_std)
z35 Variance (6h_var)	MULTIVARIATE (z35, d_mean, 6h_var, 6h_min, 6h_max, weekday)	Linear Regression (LR_multi_6h_var)
z35 Minimum value (6h_min)	MULTIVARIATE (z35, d_mean, 6h_std, 6h_min, 6h_max, weekday)	Linear Regression (LR_multi_6h_min_6h_std)
z35 Maximum value (d_max)	UNIVARIATE (d_max)	Passive Aggressive Regressor (PAR_uni_d_max)

Table 2. Best Linear models for predicting z35 and its features

We report also the plots of error metrics score R2, MAE and RMSE of the five best models chosen, one for each features of interest. In each plot, the first column represent the error score of z35 best linear model, the second column of z35 standard deviation best linear model, the third column of z35 variance best linear model, the fourth column of z35 minimal value best linear model and the fifth column of z35 maximal value best linear model.

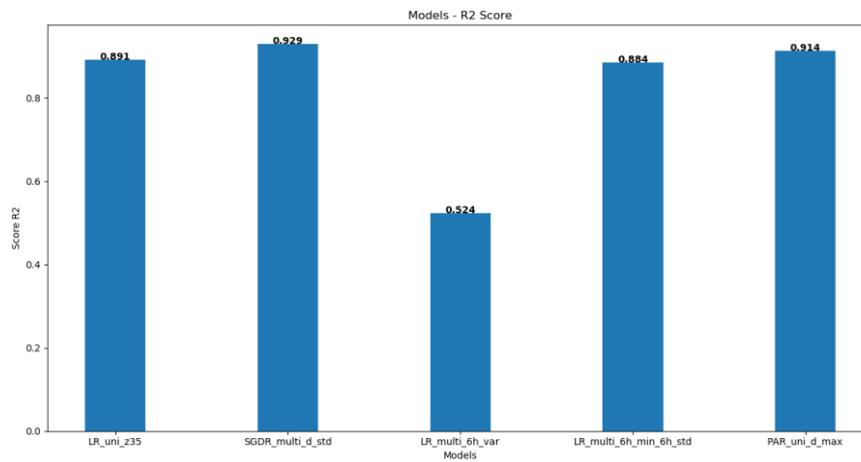


Figure 263. R2 Scores best Linear models

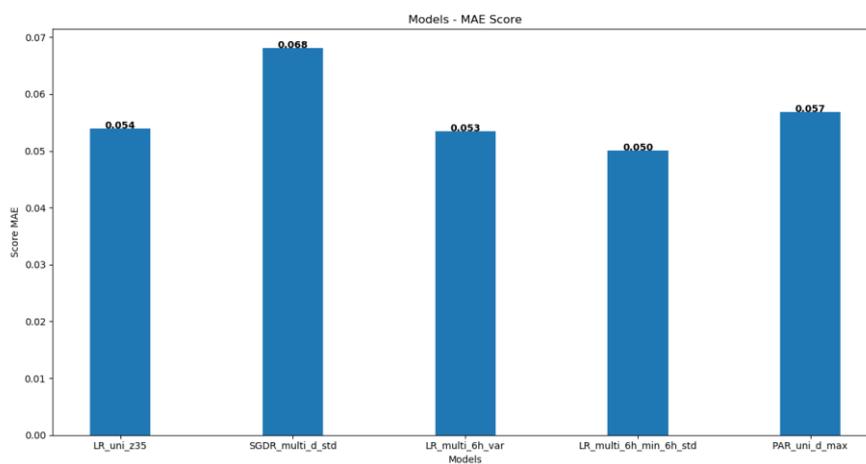


Figure 264. MAE Scores best RNN models

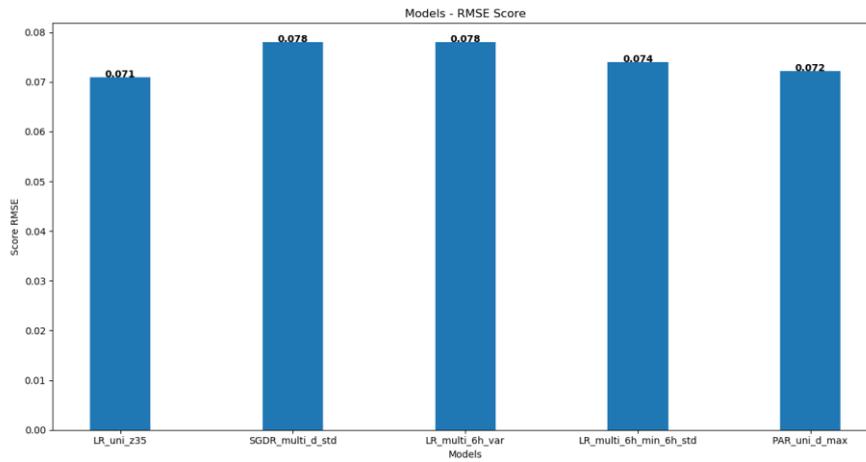


Figure 265. RMSE Scores best RNN models

From analysis of best Linear Regression models found for each of the features we want to predict, we can observe that the influence of the extra features added in Multivariate model is not too much noticeable like instead was in RNN models, this is proved from the fact that some of the best model found are Univariate and other are Multivariate. In some cases uses Multivariate model increase the prediction performance, in other it degrades. In general, considering features with same time interval (one day or of six hours), LR models shows a degrade of the performance in Multivariate case, but in some case (like for the best model for predicting z35_var and for predicting z35_min) they show a very small improvement than the Univariate model; instead SGDR models shows almost always a small improvement in Multivariate case than in Univariate case; for PAR models the considerations are the same of LR models.

2.3 Ensemble Regression Models

The last family of models tested for predicting electrical load time series belongs to Ensemble Regression algorithms. Those algorithms are implemented using Scikit-learn and xgboost library in Python IDE.

For ensemble models are valid the same considerations made for Linear Regression models regarding the choices for the size of `past_history` and `future_target` time window and the preprocessing of the data for training each ensemble models, because the models are implemented using the same library of Linear Regression models (considerations valid also for xgboost library).

In this case we will test some ensemble models with the default configuration of the hyperparameters to verify which model have the most accurate predictions, so to discard the less promising models, and focusing only on the best ones using also Grid Search method for hyperparameter optimization. Grid Search process is applied in the same way used for Linear Regression models, i.e. we will apply it on the model for predicting z35 in Univariate case and the configuration found for the hyperparameters is used also when the same kind of model is realized for predicting the other features of interest, because are features extracted directly from z35.

We can summarize the steps for preprocessing the data and calculating the error metrics score:

- Dividing dataset of 29184 samples, in 25000 samples for training set and 4184 samples for test set.
- Applying the scaling of the values using normalization between 0 and 1.
- Applying the time window of `past_hystory` of 672 timesteps and the time window of `future_target` of 672 timesteps along all train

and test set, shifting them one timestep per time, to create the data for the model.

- Flattening the first two dimensions of the training variable containing the samples for the model, from format (n_samples, n_timesteps=672, n_features) to format (n_samples*n_timesteps, n_features).
- Calculate of the error metrics score using all samples available in in test set.

Before starting the description of ensemble models tested, we report as reference the results of a first comparison plot of error metric R2 made between Linear Regression and Ensemble models in Univariate case for predicting z35, a test done at the beginning of the research with the purpose to have a panoramic of which kind of regression models have better fit on electric load time series target, where all the models use the default configuration of the hyperparameters. The preprocessing of the data and the calculating of error metrics score described for ensemble models are still valid, the only difference is that here we didn't use any technique of scaling of the values (we can do that because they are all Univariate models), but considering R2 as error metrics is not relevant because it's an error related to the variance of the predictions.

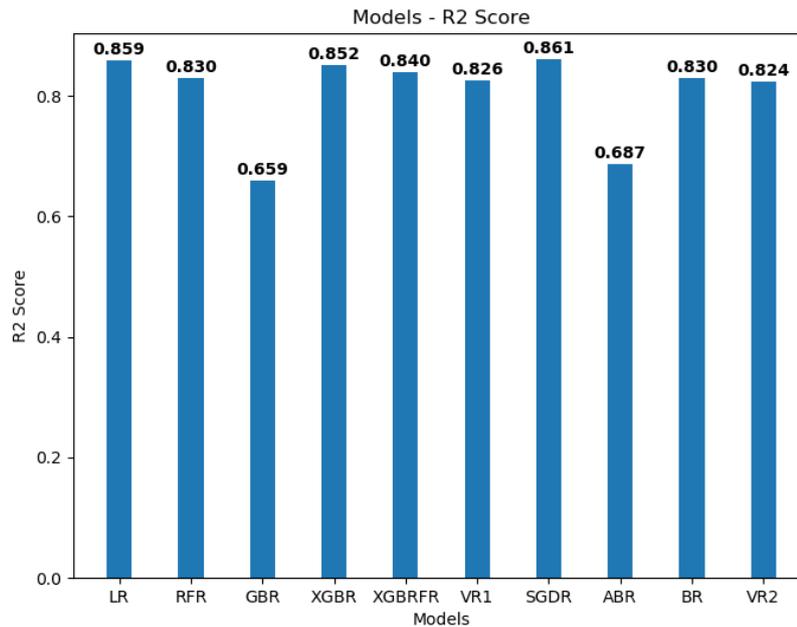


Figure 266. R2 error scores of Linear and Ensemble models tested

The models realized and tested for this comparison are: Linear Regression (LR), Random Forest Regressor (RFR), Gradient Boosting Regressor (GBR), Extreme Gradient Boosting Regressor (XGBR), Extreme Gradient Boosting Regressor with RFR as base model (XGBRFR), Voting Regressor (VR1 that use as base models: LR, RFR, GBR, XGBR and XGBRFR models previously defined), Stochastic Gradient Descent Regressor (SGDR), AdaBoost Regressor (ABR), Bagging Regressor (BR) and Voting Regressor (VR2 that use as base models: LR, RFR, GBR, XGBR, XGBRFR, SGDR, ABR and BR models previously defined). We can observe that the two Linear models (LR and SGDR) have the better prediction performance than all the other ensemble models in default configuration of the hyperparameters. From this comparison, the best among ensemble model is XGBR, therefore we will take it into account in the following discussion on ensemble models.

2.3.1 Ensemble Regression Models – Preliminary Test

As we know, in ensemble learning, weak learners (or base models) are used as building blocks for designing more complex models by combining several of them in order to create a strong learner (or ensemble model) that achieves better performances. We will use as base models: decision tree regressors, other ensemble models and the most promising kind of linear regression models previously found (LR, SGDR, PAR).

The models tested are: Random Forest Regressor (RFR), Extreme Gradient Boosting Regressor (XGBR), Bagging Regressor (BR, BR2, BR3), Adaboost Regressor (ABR, ABR2, ABR3) and Voting Regressor (VR, VR2, VR3, VR4). The different BR, ABR and VR models mean that we use different base models.

- **Random Forest Regressor (RFR)**

Random Forest Regressor use the Bagging technique to produce the final output prediction, combining the results of the predictions of different decision trees, each one fitted on various subsample of the dataset. The final prediction is the results of the average of the predictions of the single decision trees with the objective to improve the predictions accuracy and control overfitting. In fact Random Forest model tend to have a low variance than the single decision trees that form it, even at the cost of a small increase in bias.

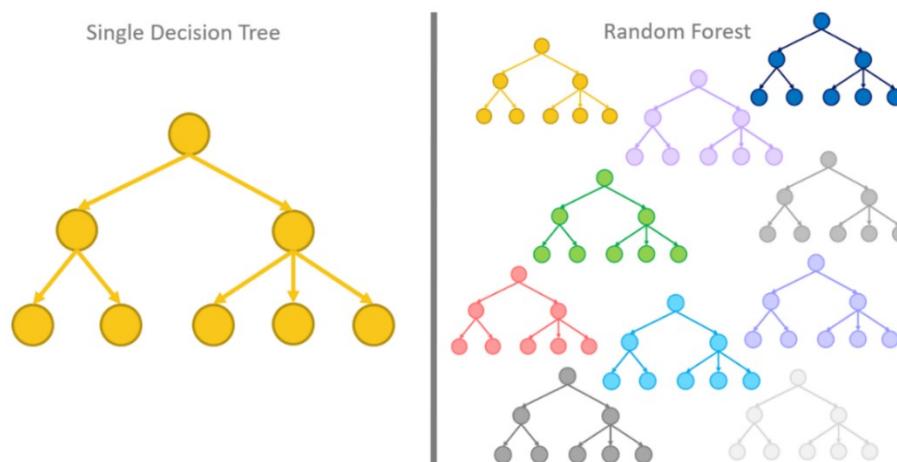


Figure 267. Random Forest Regressor model

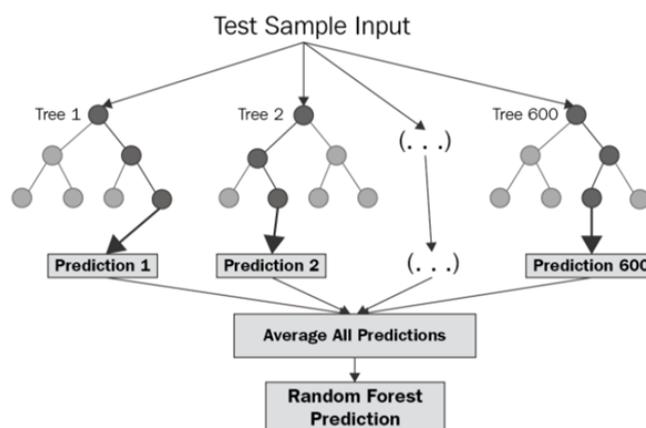


Figure 268. Final prediction of Random Forest Regressor model

For this model we have tested Grid Search method for optimization of hyperparameter $n_estimators$ and max_depth that indicate respectively the number of Decision trees that form the model and the maximum depth of each tree. Higher are those parameters and more accurate is the model, but on the other hand we have a considerable increase of computational time for training, with predictions results that are not too much better than the default configuration of the hyperparameters of the model. So at the end we have used the default configuration of it.

Model name: "RFR_uni_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

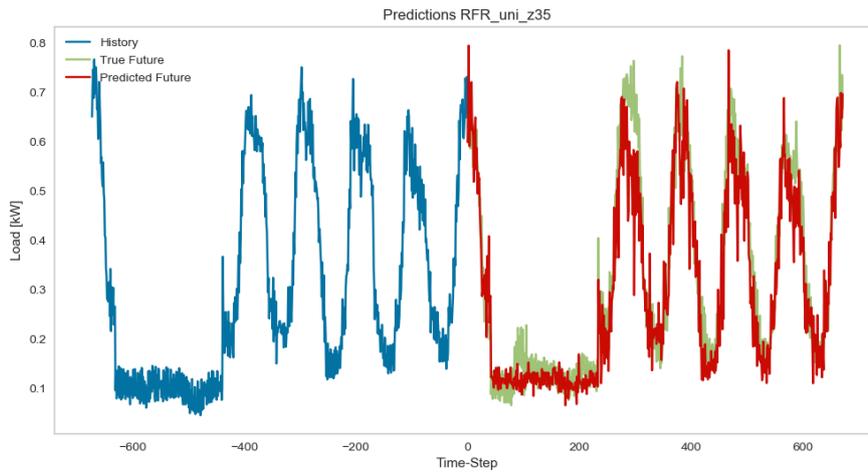


Figure 269. Example of one RFR model prediction

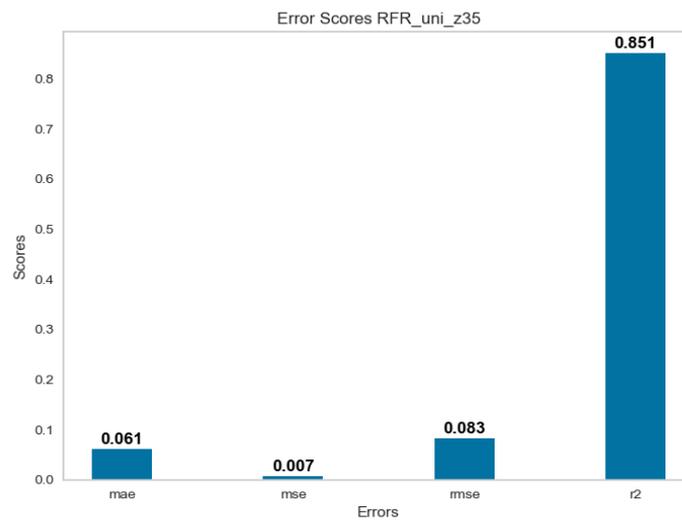


Figure 270. Error metrics score calculated on all samples of test set

- **Extreme Gradient Boosting Regressor (XGBR)**

Extreme Gradient Boosting Regressor use the Boosting technique to produce the final output prediction. In this case we have an additive model, where the decision trees that form the model are added one at a time and we use a gradient descent procedure to minimize the loss function. The minimization of cost function using Gradient Descent is made adding a tree into the model, so the procedure is to calculate the loss and then add a tree that reduces it (i.e. follow the gradient). We can do this by parameterizing the tree, then going to modify its parameters to reducing the residual loss. The output of the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. In Extreme Gradient Boosting we use a better regularization function than simple Gradient Boosting to improve the predictions accuracy and control overfitting.

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Equation 12. Cost Function XGBR model

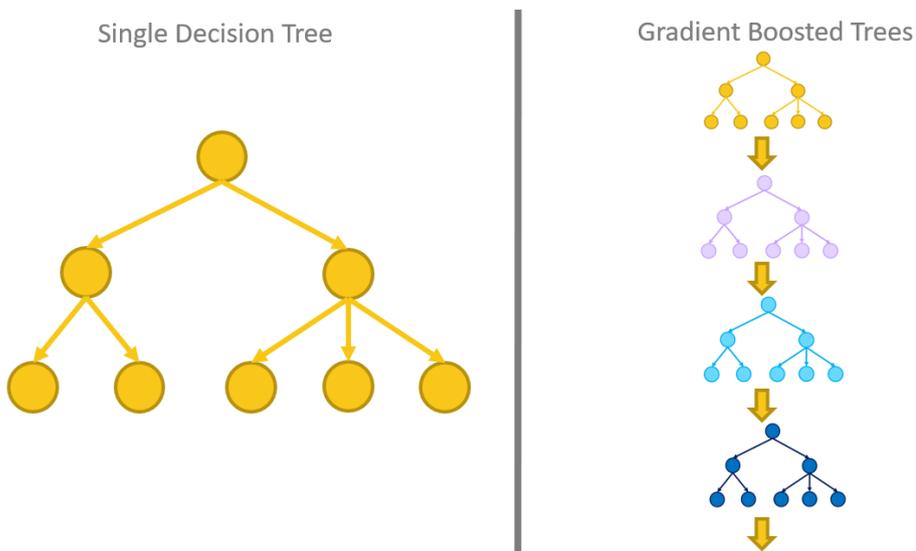


Figure 271. Extreme Gradient Boosting Regressor model

From the first comparison made between Linear and Ensemble models using default configuration of the hyperparameters (figure 266), XGBR resulted the best ensemble model for predicting z35 values. We apply the Grid Search method to two key hyperparameters: *learning_rate* and the regularization factor *alpha*

$$\begin{cases} \alpha = [1; 0,1; 0,01; 0,001] \\ \text{learning_rate} = [0,5; 0,1; 0,05; 0,01; 0,005] \end{cases}$$

After Grid Search process, we obtain these optimal values for the hyperparameters:

$$\begin{cases} \alpha = [1] \\ \text{learning_rate} = [0,5] \end{cases}$$

We can now describe the final configuration used of the main hyperparameters of the model: attribute *n_estimators* has been set equal to 10 and indicate the number of trees that form the model (we choose a not too high value because otherwise it would have required a lot of computational time to have to test different configurations to get the best model). The parameter *early_stopping_rounds* was set equal to 10 and indicate after how many trees added without improvements is necessary to stop the training; in our case this parameter doesn't affect the number of trees of the model, because we have set *n_estimators* equal to 10. The parameter *max_depth* is set equal to 10 and indicate the number of maximum split can have each tree. The parameter *subsample* is equal to 1 and indicate that at each iteration during training (i.e. at every tree added with Boost technique) the model use the entire training dataset. Attribute *objective* is set equal to *reg:squarederror*, so it means that the objective of the model is to minimize a quadratic type cost function, the error metrics used is RMSE. The regularization parameters *alpha* and *lambda*, respectively of type L1 and L2, are set equal to 1, making the model very conservative, so it means that it takes enough account of the previous iterations. The last parameter *learning_rate* is set equal to 0,5.

Model name: "XGBR_uni_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

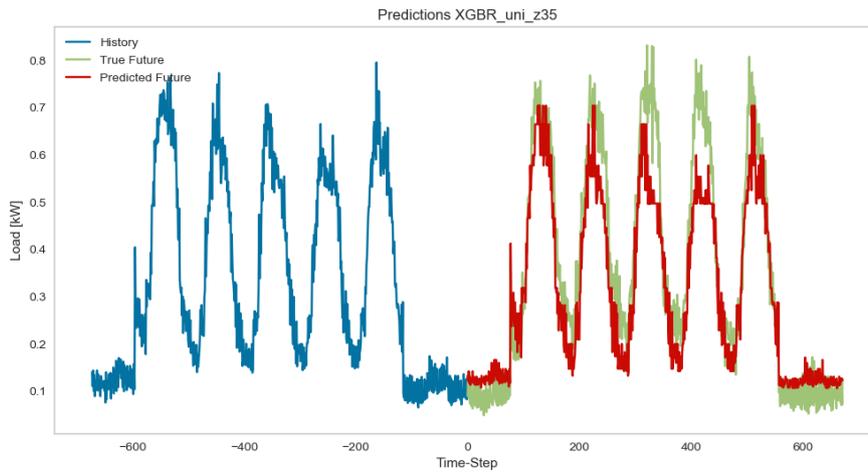


Figure 272. Example of one XGBR model prediction

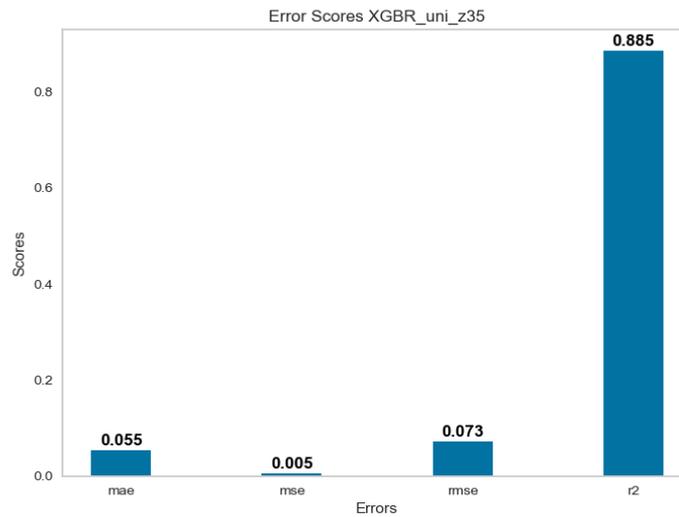


Figure 273. Error metrics score calculated on all samples of test set

- **Bagging Regressor (BR)**

Bagging Regressor model use the Bagging technique to produce the final output prediction, combining the results of the predictions of different base models, each one fitted on random subset of the dataset. The final prediction is the results of the average of the predictions of the single base models with the objective to improve the predictions accuracy and control overfitting. If using Decision trees as base models, it's behaviour is equal to the Random Forest Regressor model, like we can observe from error score R2 of models RFR and BR of figure 266.

For Bagging Regressor models we use the default configuration of the hyperparameters, because what we want to verify is the prediction performance using different kind of base models.

In the first model (BR), we use Decision tree regressor as base model and their number is equal to 10 (attribute `n_estimators`). The final prediction is the average of the predictions of the 10 base models.

Model name: "BR_uni_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

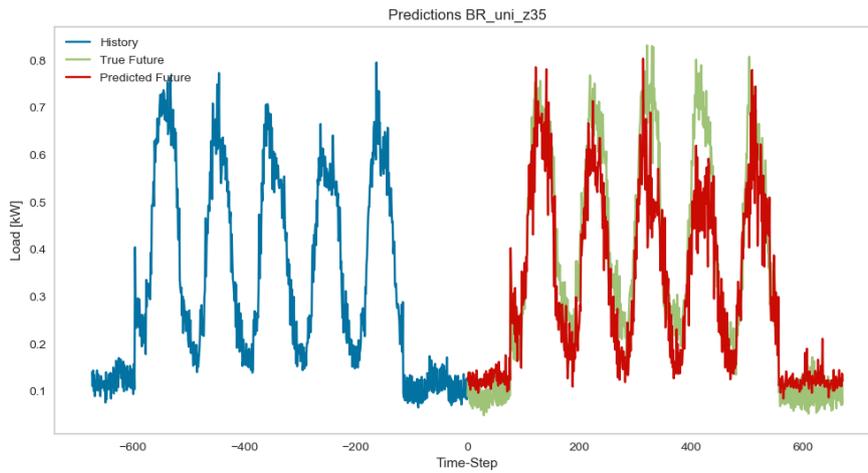


Figure 274. Example of one BR model prediction

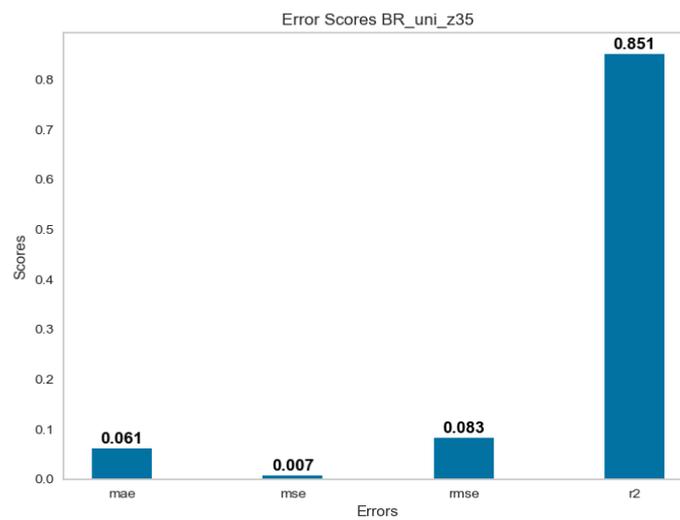


Figure 275. Error metrics score calculated on all samples of test set

The second Bagging Regressor model (BR2) uses as base models the XGBR models previously designed after applying Grid Search method. The number of base models is always set equal to 10.

Model name: "BR_uni_2_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

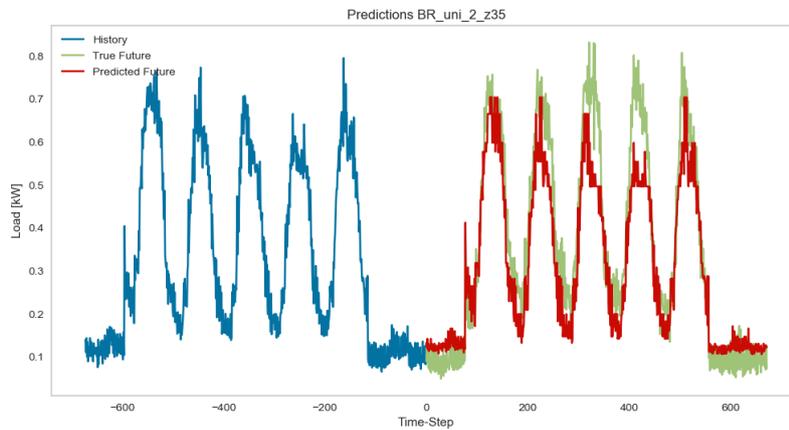


Figure 276. Example of one BR_2 model prediction

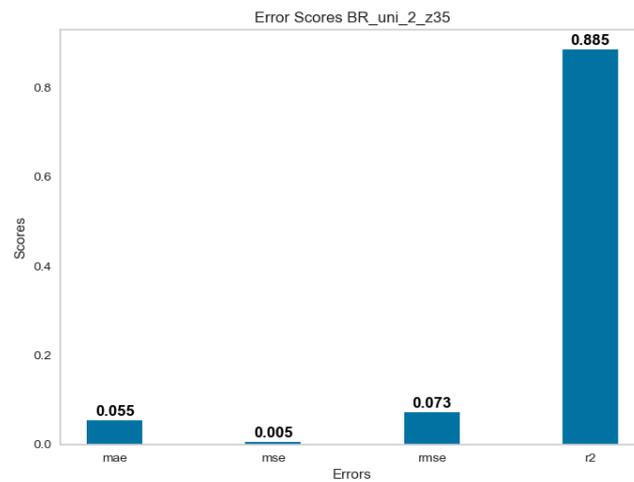


Figure 277. Error metrics score calculated on all samples of test set

The last Bagging Regressor model (BR3) uses always 10 base models which are Linear Regression (LR) models. We chosen Linear Regression (LR) model because it was the best linear model for predicting z35 in Univariate case.

Model name: "BR_uni_3_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

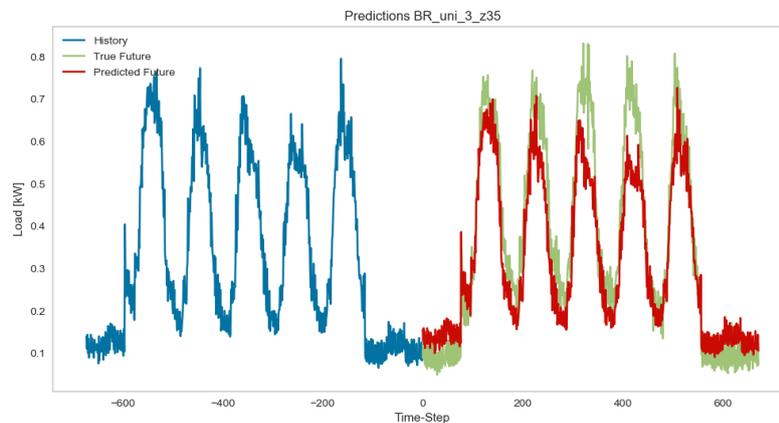


Figure 278. Example of one BR_3 model prediction

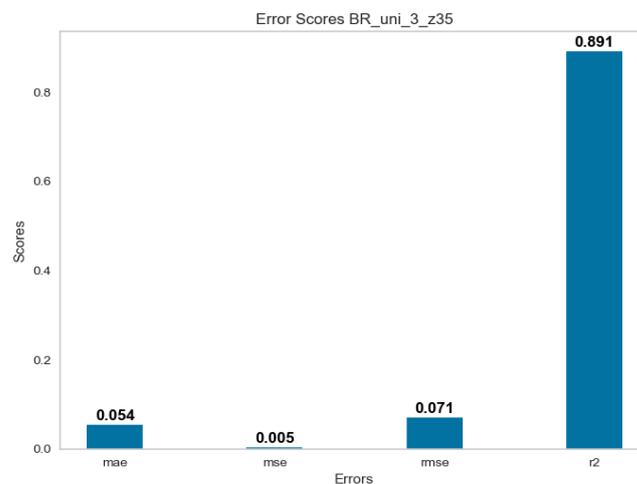


Figure 279. Error metrics score calculated on all samples of test set

- **AdaBoost Regressor (ABR)**

Adaboost Regressor model use the Boosting technique to produce the final output prediction. In this case, we have a first base model trained on the original dataset and there are subsequent copies of the base model that train on the same dataset as the previous one, but with the weights updated to the current prediction. So the sequence of models focuses more and more on learning complex cases, that is, on those training samples where the previous base models have made an incorrect prediction.

Also for AdaBoost Regressor models we use the default configuration of the hyperparameters, because what we want to verify is the prediction performance using different kind of base models.

In the first model (ABR), we use Decision tree regressor as base model and their number is equal to 50 (attribute `n_estimators`). The parameter `loss=linear` indicate the type of cost function used. The parameter `learning_rate` is set to 1.

Model name: "ABR_uni_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

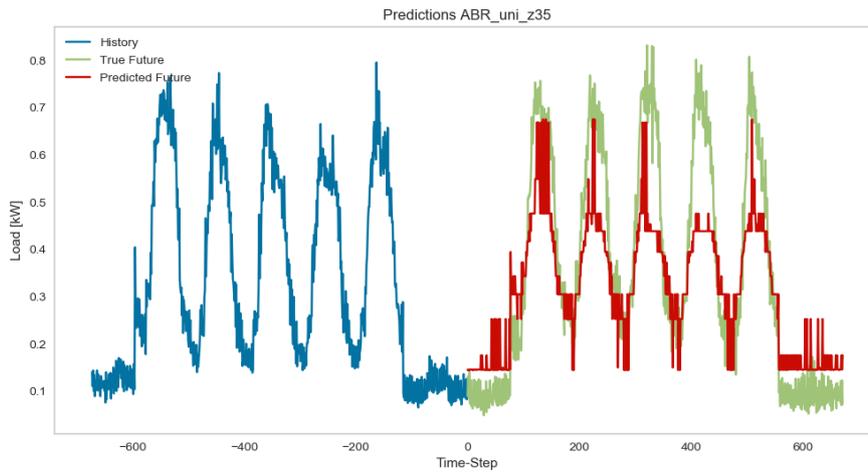


Figure 280. Example of one ABR model prediction

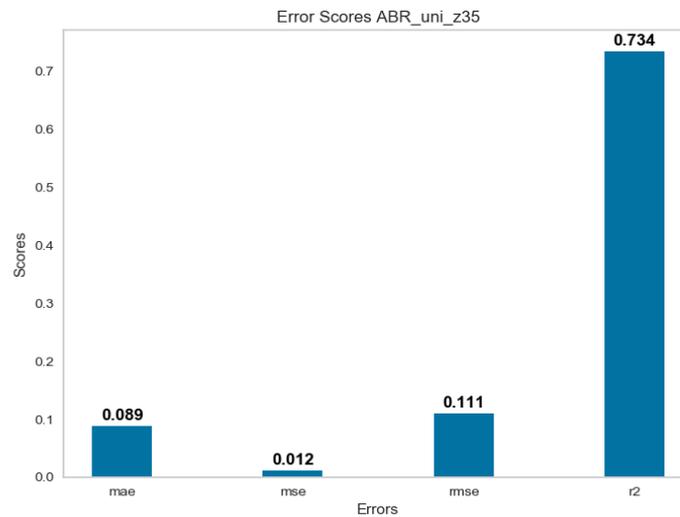


Figure 281. Error metrics score calculated on all samples of test set

The second AdaBoost Regressor model (ABR2) uses as base models the XGBR models previously designed after applying Grid Search method. The number of base models is always set equal to 50.

Model name: "ABR_uni_2_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

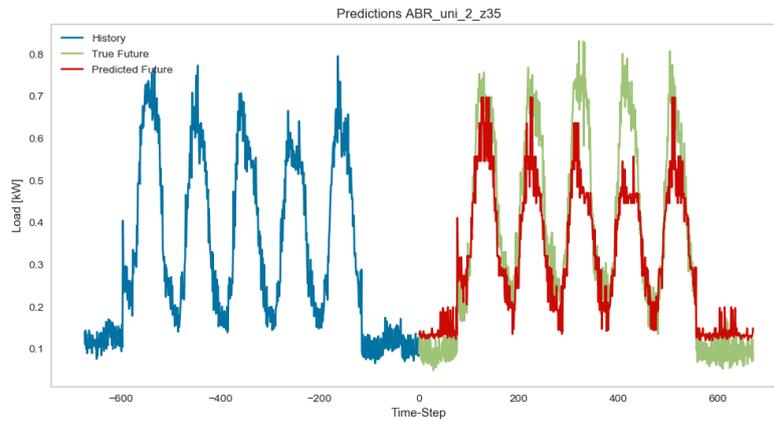


Figure 282. Example of one ABR_2 model prediction

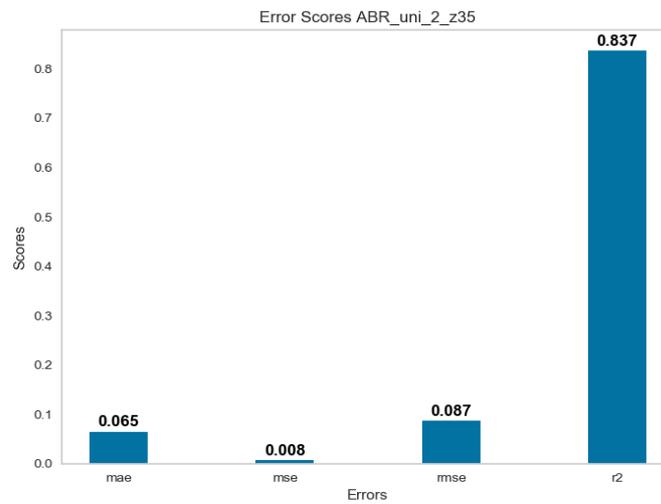


Figure 283. Error metrics score calculated on all samples of test set

The last AdaBoost Regressor model (ABR3) uses always 50 base models which are Linear Regression (LR) models. We chosen Linear Regression (LR) model for the same reason explained for model BR3, because it was the best linear model for predicting z35 in Univariate case.

Model name: "ABR_uni_3_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

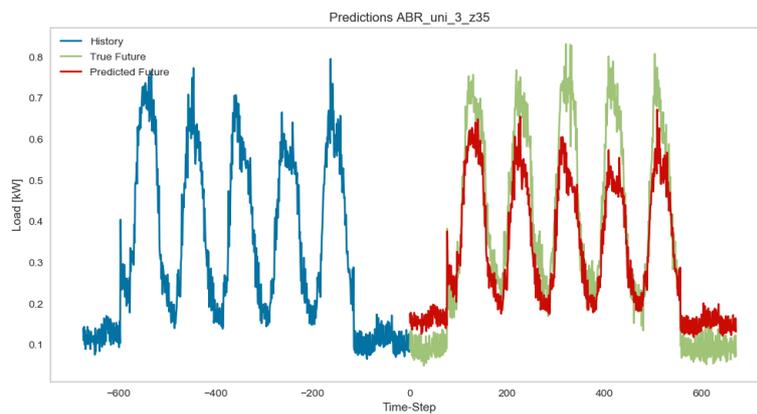


Figure 284. Example of one ABR_3 model prediction

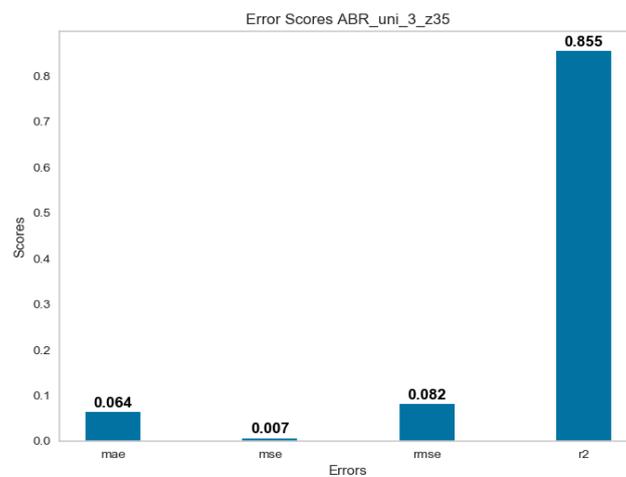


Figure 285. Error metrics score calculated on all samples of test set

Now before describing Voting Regressor models, we report the comparison plot between error metrics R2, MAE and RMSE of the ensemble models tested until now and of the three Linear Regression models (LR, SGDR and PAR) previously treated during analysis of best Linear models, in this way we can verify if there are some ensemble models with better predictions than Linear models.

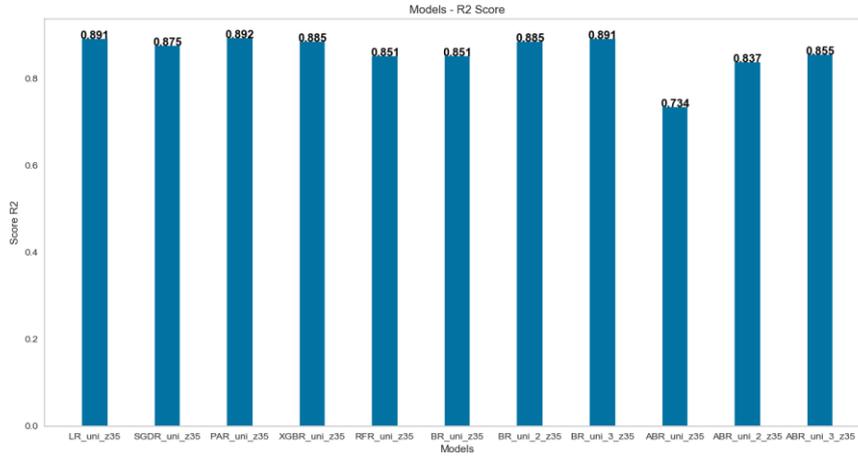


Figure 286. R2 Scores of Linear and Ensemble models tested for predicting z35 Univariate

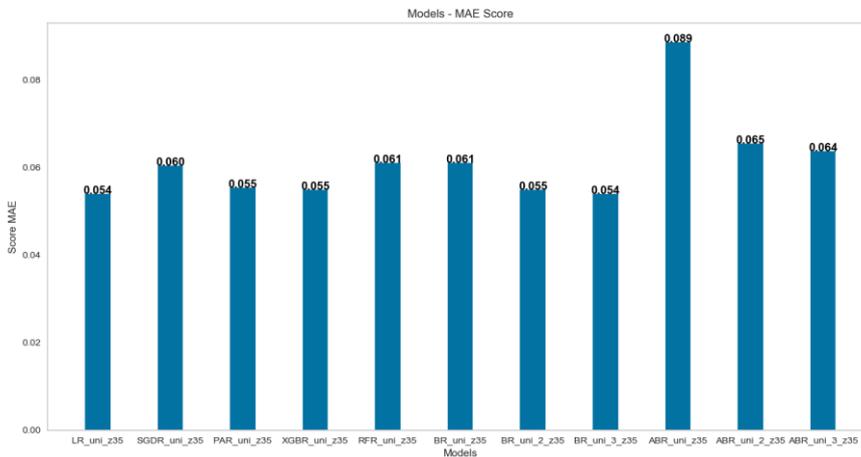


Figure 287. MAE Scores of Linear and Ensemble models tested for predicting z35 Univariate

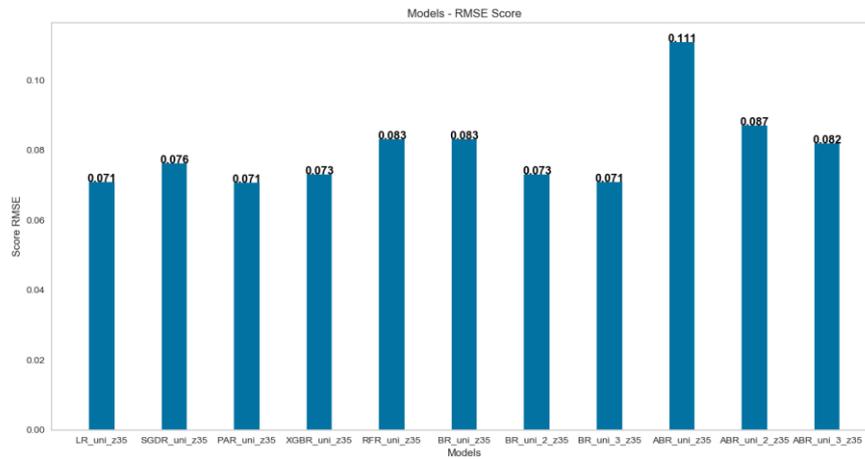


Figure 288. RMSE Scores of Linear and Ensemble models tested for predicting z35 Univariate

From this comparison we can observe that ensemble models don't improve the prediction performance than Linear models. The models RFR and BR are equivalent because BR uses Decision Trees as base models and have the same values of the parameters of RFR regarding the number of estimators and max depth of each tree. BR_2 and BR_3 models don't improve the performance of the base models, but they are equivalent, therefore it's unnecessary to use ensemble method. ABR results the worst model in term of error metrics score and ABR_2 and ABR_3 models get worse the prediction performance than base models.

- **Voting Regressor (VR)**

Voting Regressor model use the Bagging technique to produce the final output prediction. In this case different base models are trained on all dataset and the final prediction is the results of the average of the predictions of the single base models.

For Voting Regressor models we use the default configuration of the hyperparameters, because what we want to verify is the prediction performance using different kind of base models.

In the first Voting Regressor model (VR) we have as base models the ensemble models previously described that uses Decision Trees as base models that are: RFR, BR, XGBR, ABR.

Model name: "VR_uni_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

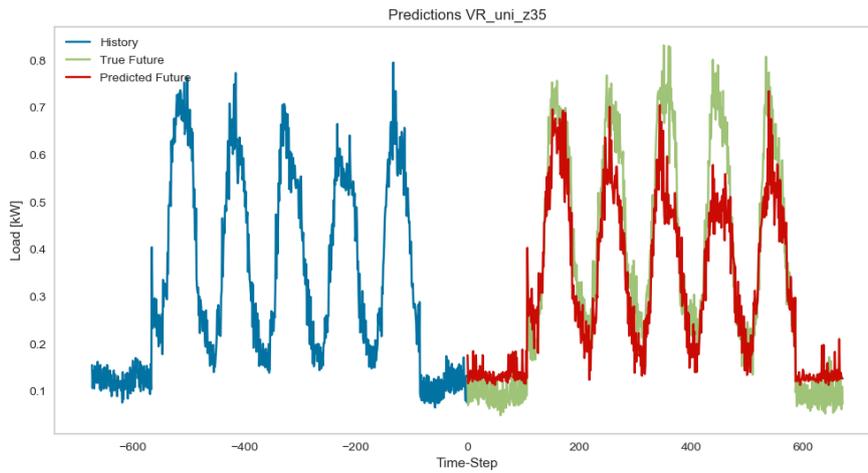


Figure 289. Example of one VR model prediction

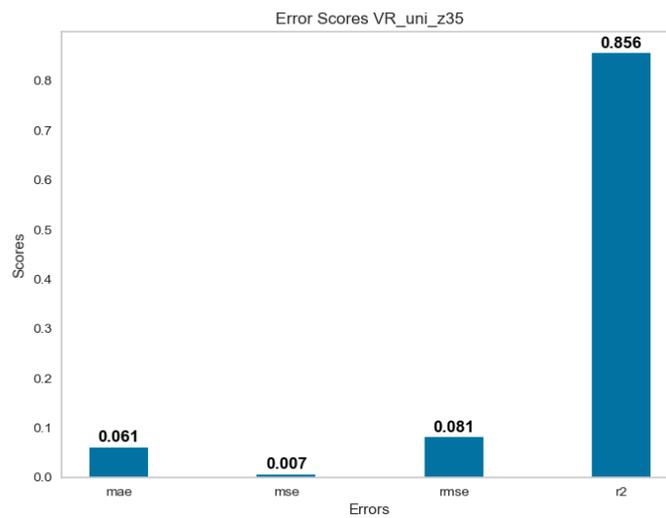


Figure 290. Error metrics score calculated on all samples of test set

We report also the comparison plot of the error metrics R2, MAE and RMSE between Voting Regressor (VR) and its base models.

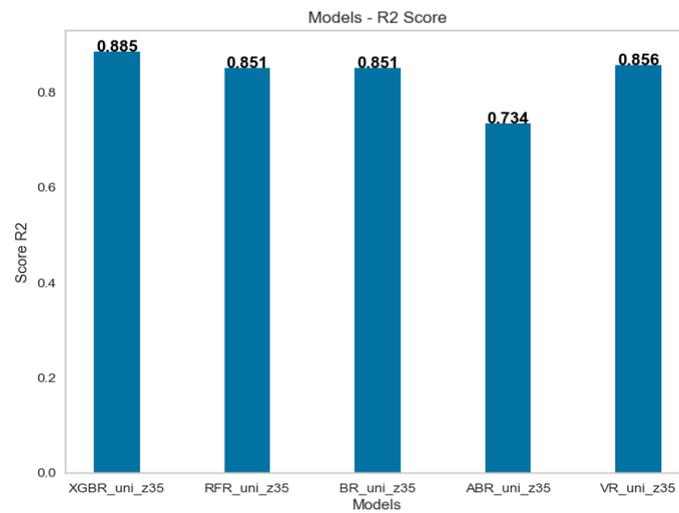


Figure 291. R2 Scores of Voting Regressor and its base models tested for predicting z35 Univariate

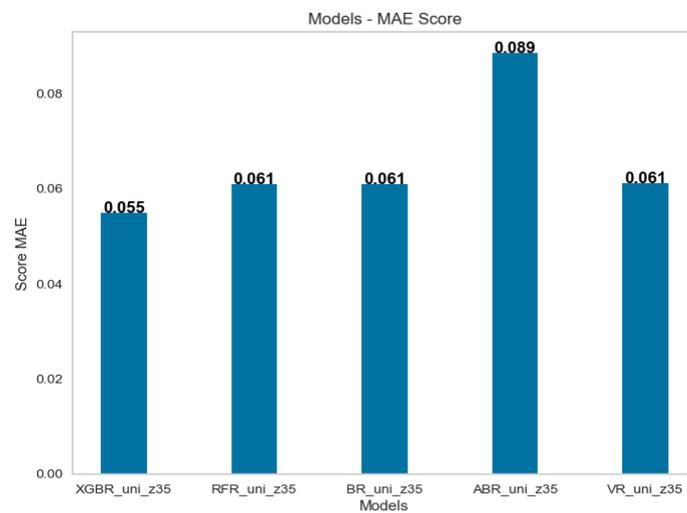


Figure 292. MAE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

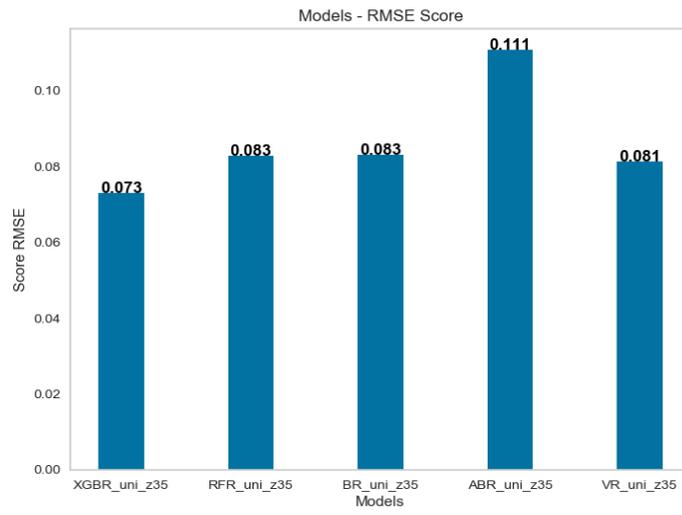


Figure 293. RMSE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

In the second Voting Regressor model (VR2) we use as base models the three Bagging Regressor models previously described (BR, BR2 and BR3).

Model name: "VR_uni_2_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

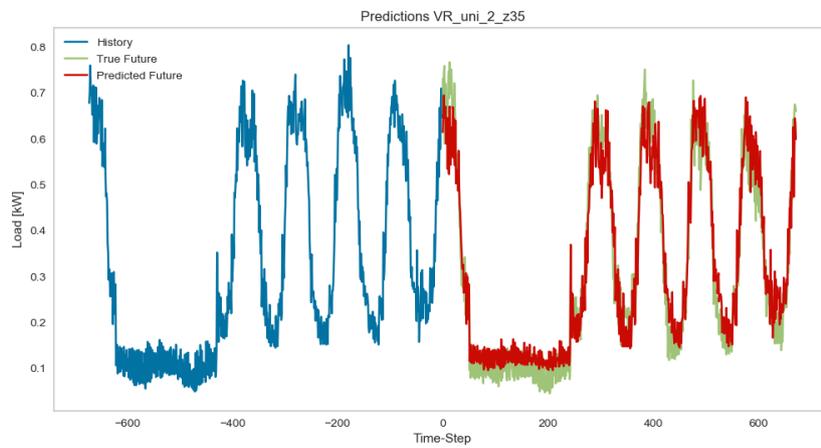


Figure 294. Example of one VR_2 model prediction

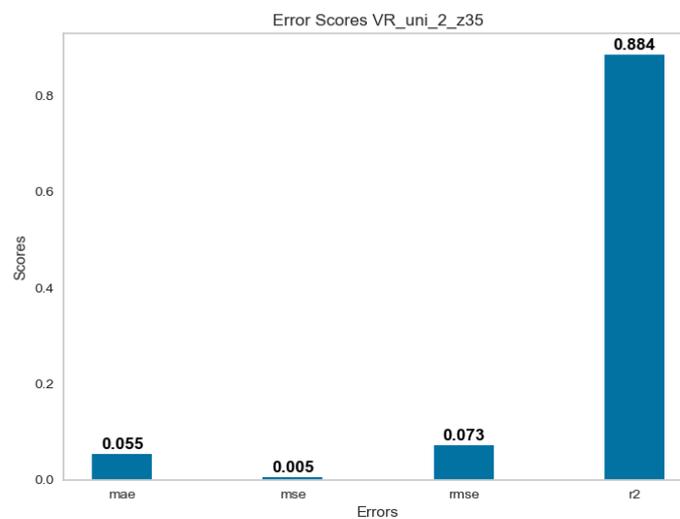


Figure 295. Error metrics score calculated on all samples of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between Voting Regressor (VR2) and its base models.

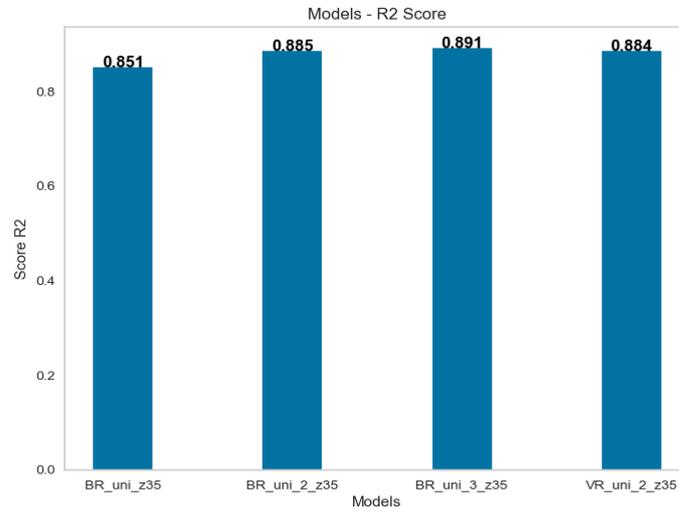


Figure 296. R2 Scores of Voting Regressor and its base models tested for predicting z35 Univariate

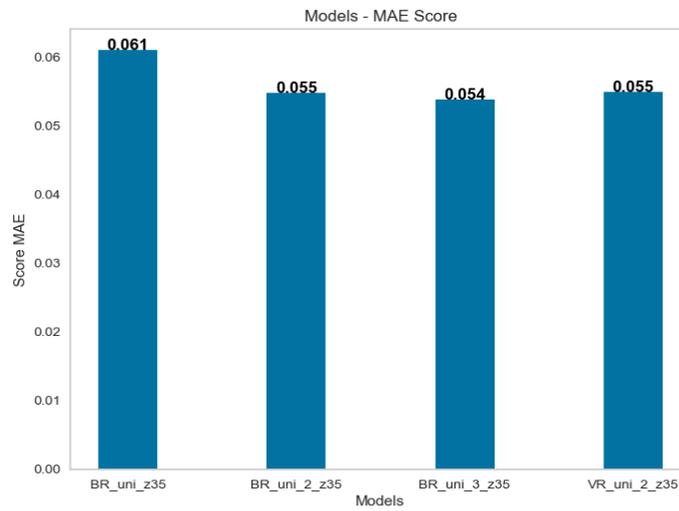


Figure 297. MAE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

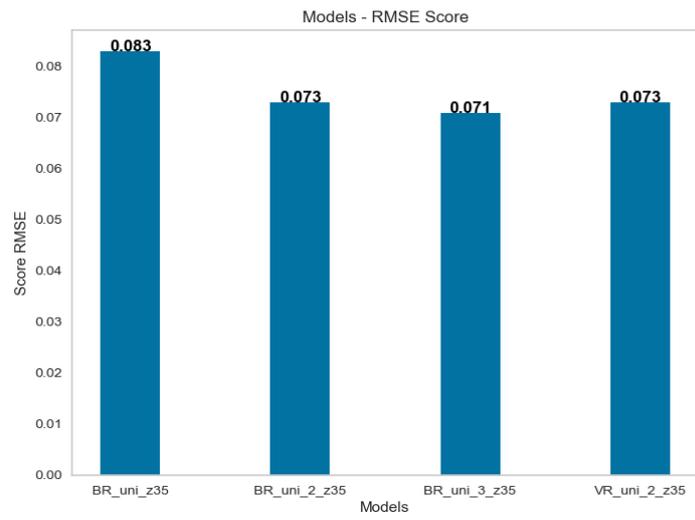


Figure 298. RMSE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

In the third Voting Regressor model (VR3) we use as base models the three AdaBoost Regressor models previously described (ABR, ABR2 and ABR3).

Model name: "VR_uni_3_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

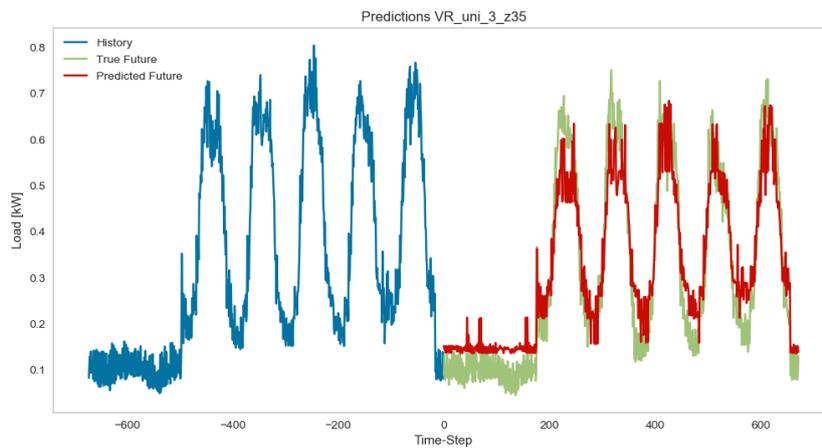


Figure 299. Example of one VR_3 model prediction

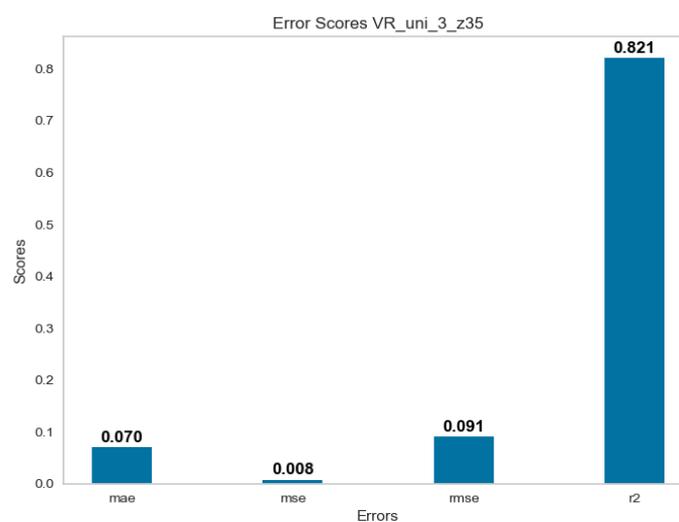


Figure 300. Error metrics score calculated on all samples of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between Voting Regressor (VR3) and its base models.

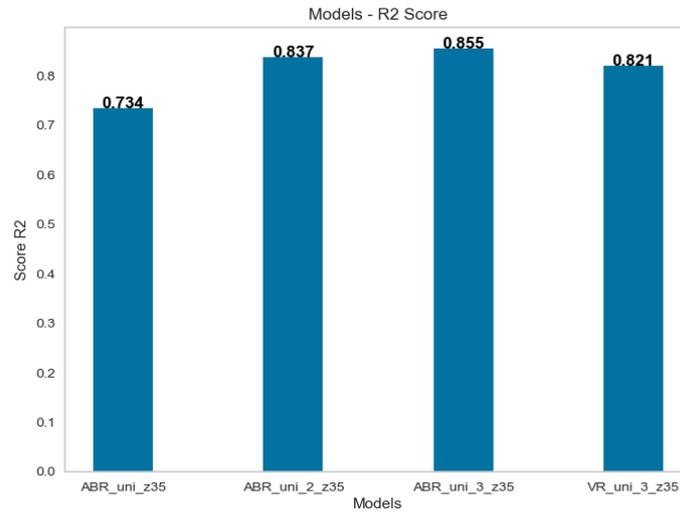


Figure 301. R2 Scores of Voting Regressor and its base models tested for predicting z35 Univariate

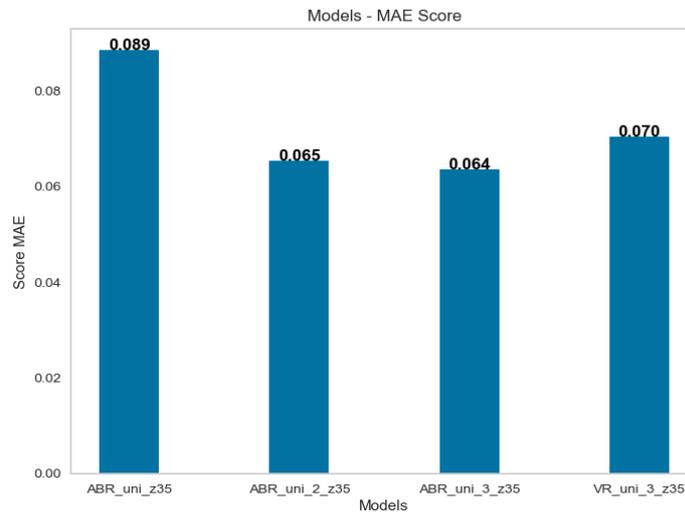


Figure 302. MAE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

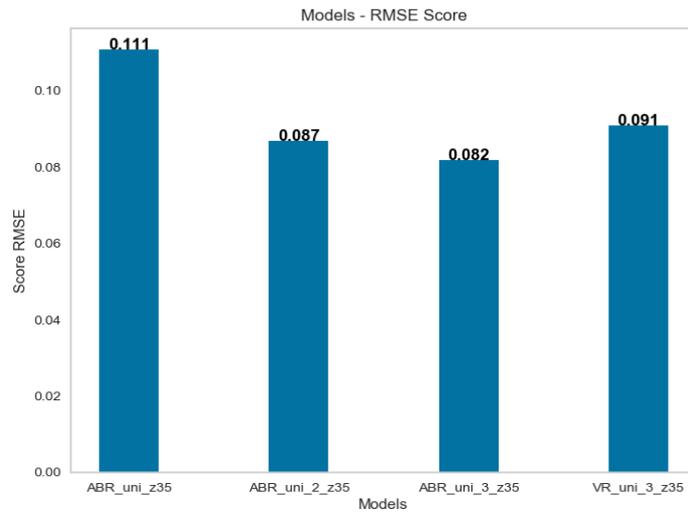


Figure 303. RMSE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

In the last Voting Regressor model (VR4) we use as base models the three Linear models previously tested (LR, SGDR, PAR).

Model name: "VR_uni_4_z35";

Features used: z35

Dimensions of Input and Output for each prediction:

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

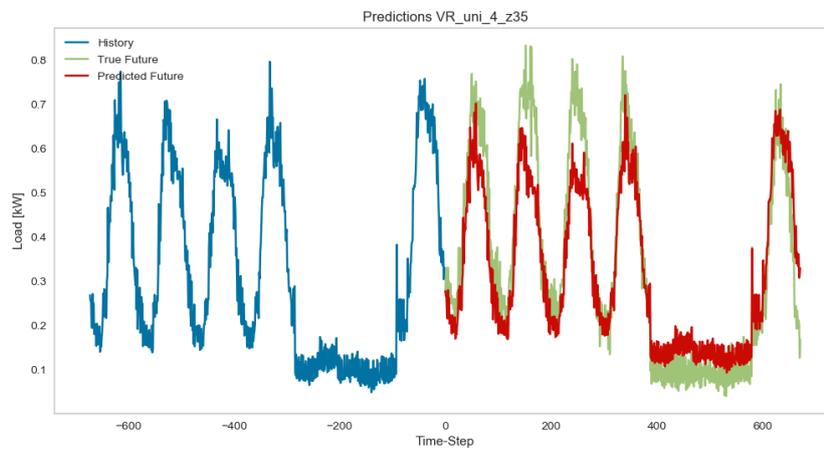


Figure 304. Example of one VR_4 model prediction

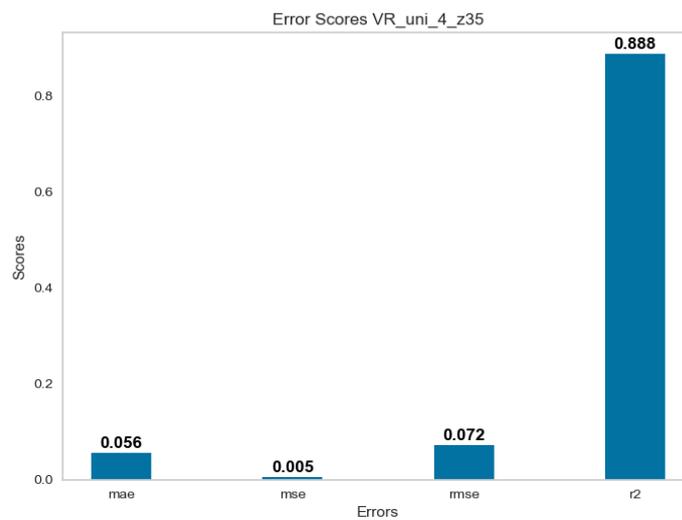


Figure 305. Error metrics score calculated on all samples of test set

We report the comparison plot of the error metrics R2, MAE and RMSE between Voting Regressor (VR4) and its base models.

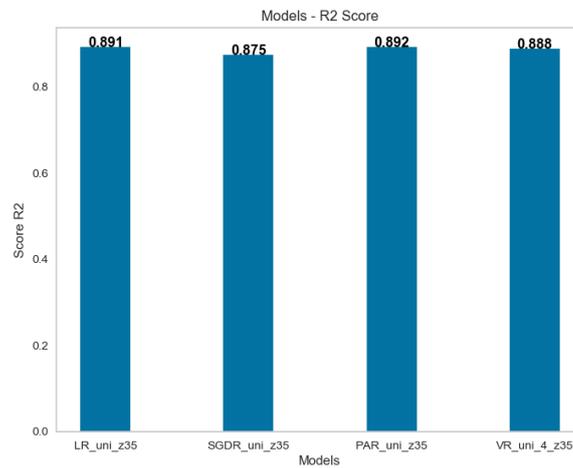


Figure 306. R2 Scores of Voting Regressor and its base models tested for predicting z35 Univariate

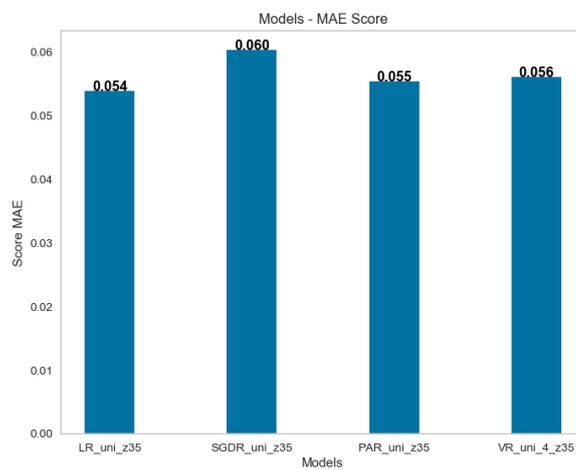


Figure 307. MAE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

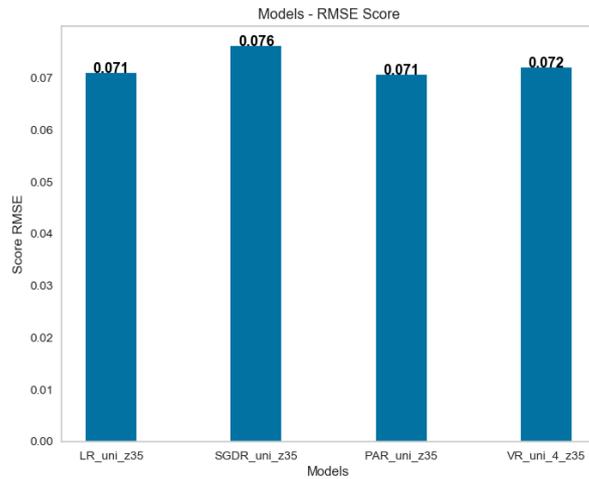


Figure 308. RMSE Scores of Voting Regressor and its base models tested for predicting z35 Univariate

From the analysis of Voting Regressor models tested, we can conclude that they don't improve the predictions of the base models used, more precisely being the final prediction an average of the predictions of the base models, Voting Regressor models improve the prediction performance of worst base models and degrade the performance of best base models. Because our objective is to find the best model for predicting z35 and its features extracted, Voting Regressor models are not the best choice, moreover they need an high computational time because they have to train each base model on the entire training dataset.

2.3.2 Ensemble Regression Models – Final Test

In this final test on ensemble models we follow the same procedure used for RNN and Linear Regression models, so we test different Univariate and Multivariate models to find the most promising ensemble models for predicting z35, z35 standard deviation, z35 variance, z35 minimal value and z35 maximal value. The combinations of features used in Multivariate case are the same used for RNN and Linear models, so we use always a time interval for the features extracted from z35 of one day or six hours. The preprocessing of the data is the same described in the previous test, using normalization between 0 and 1 of features values and using a time window of past_history of one week and a time window of future_target of always one week.

The ensemble models that we are going to test are (referring to the considerations of the previous test): Random Forest Regressor (RFR), Extreme Gradient Boosting Regressor (XGBR) and Bagging Regressor using XGBR model as base models (BR).

The models tested are:

- Prediction of **z35**:
 - Feature used in Univariate models:
 1. z35
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Predictions of **std** (z35 standard deviation):
 - Feature used in Univariate models:
 1. z35_6hours_std
 2. z35_daily_std
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);

- Prediction of **var** (z35 variance):
 - Feature used in Univariate models:
 1. z35_6hours_var
 2. z35_daily_var
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Predictions of **min** (z35 minimum value):
 - Feature used in Univariate models:
 1. z35_6hours_min
 2. z35_daily_min
 - Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

- Prediction of **max** (z35 maximum value):
 - Feature used in Univariate models:
 1. z35_6hours_max
 2. z35_daily_max

- Features used in Multivariate models:
 1. (z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday);
 2. (z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday);
 3. (z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday);
 4. (z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday);

For each Univariate and Multivariate case, we create a Random Forest Regressor (RFR) model, a Extreme Gradient Boosting (XGBR) model and a Bagging Regressor (BR) model that uses as base model the XGBR model; the configuration of the hyperparameters of the three models is the same used in the previous analysis for predicting z35 in Univariate case.

For simplify, in this case we report only the results of the best model found for each of the five features of interest.

- Prediction of **z35**
 - *Best model*
-

Model name: BR_multi_z35

Feature used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday

- Input shape $x = (n_timesteps=672, n_features=6)$
- True_target shape $y = (n_timesteps=672, n_features=6)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=6)$

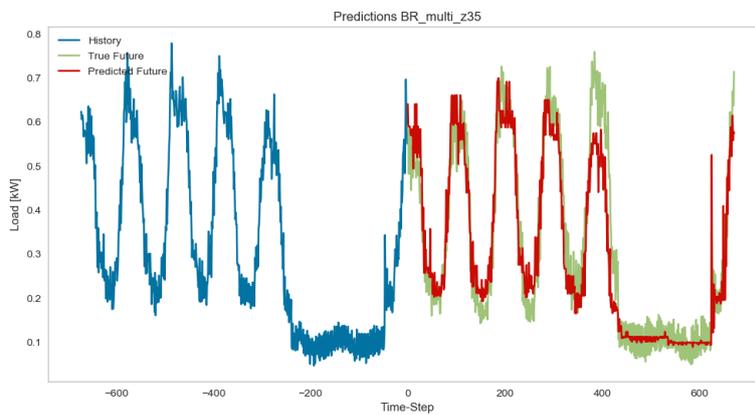


Figure 309. Example of one BR model prediction for predicting z35

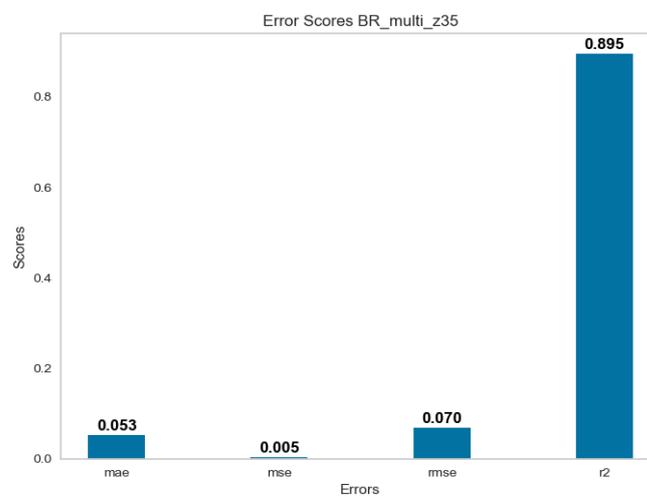


Figure 310. Error metrics score calculated on all samples of test set

- Prediction of **std**
 - *Best model*
-

Model name: XGBR_uni_6h_std

Feature used: z35_6hours_std

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

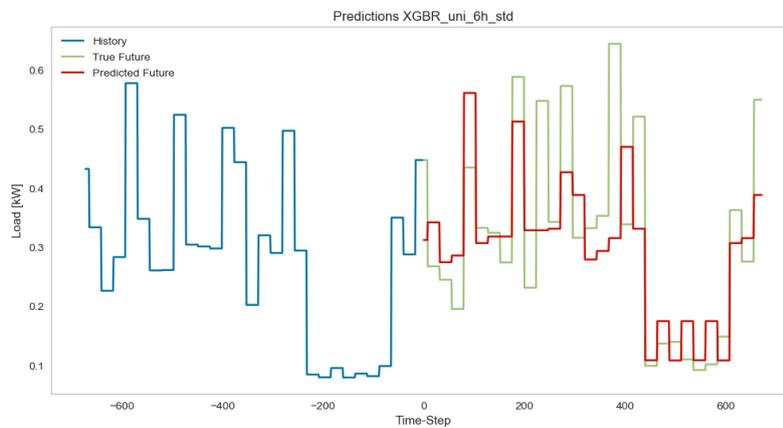


Figure 311. Example of one XGBR model prediction for predicting z35_std

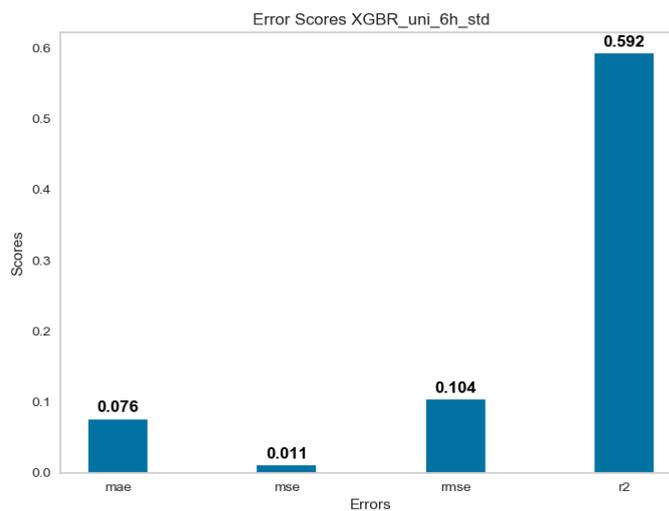


Figure 312. Error metrics score calculated on all samples of test set

- Prediction of **var**
 - *Best model*
-

Model name: XGBR_uni_6h_var

Feature used: z35_6hours_var

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

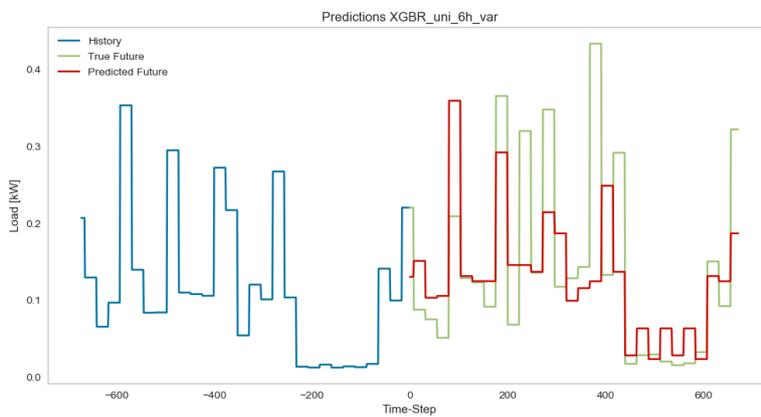


Figure 313. Example of one XGBR model prediction for predicting z35_var

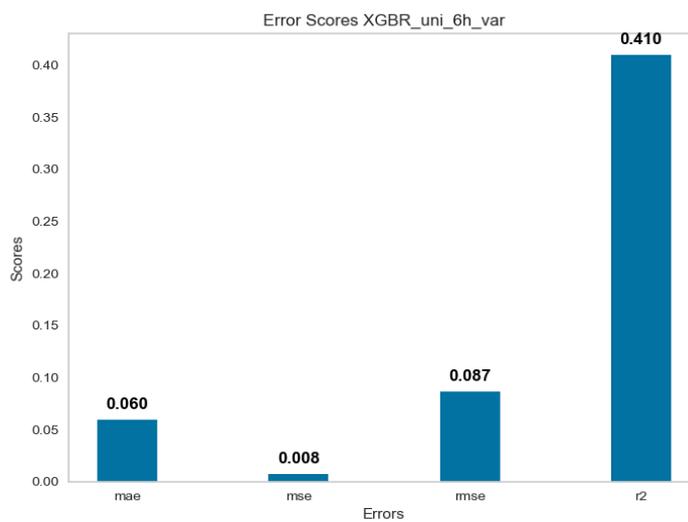


Figure 314. Error metrics score calculated on all samples of test set

- Prediction of **min**
 - *Best model*
-

Model name: BR_uni_6h_min

Feature used: z35_6hours_min

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

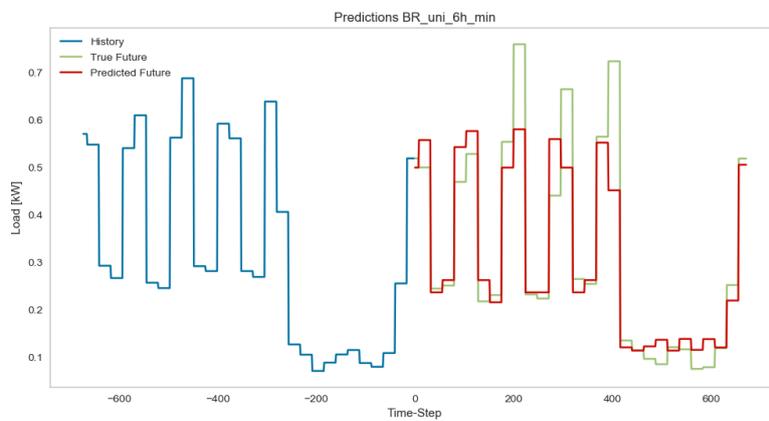


Figure 315. Example of one BR model prediction for predicting z35_min

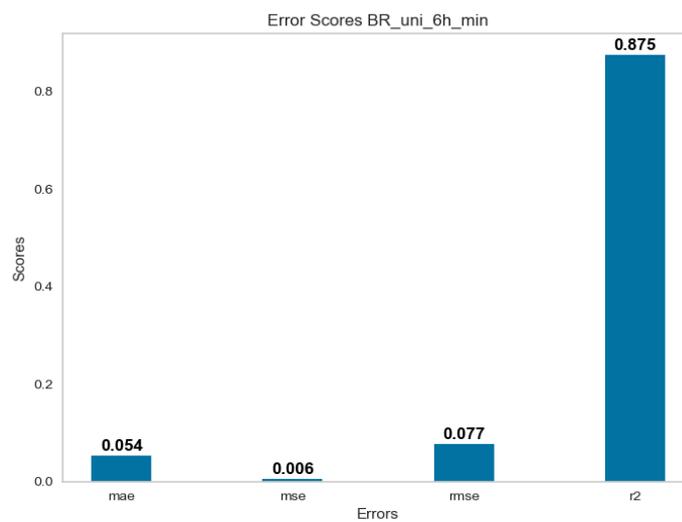


Figure 316. Error metrics score calculated on all samples of test set

- Prediction of **max**
 - *Best model*
-

Model name: BR_uni_6h_max

Feature used: z35_6hours_max

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

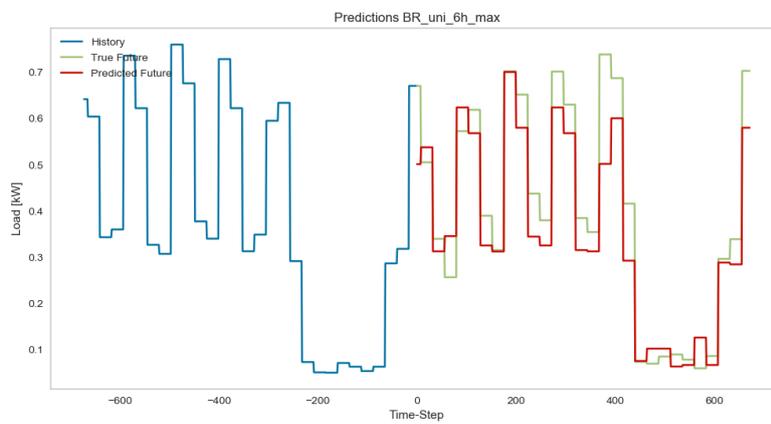


Figure 317. Example of one BR model prediction for predicting z35_max

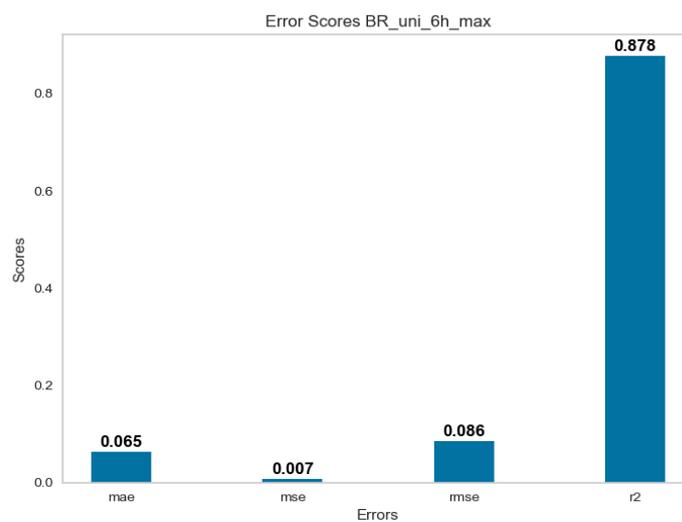


Figure 318. Error metrics score calculated on all samples of test set

We can summarize in a table the best Ensemble models found for each features of interest:

FEATURE PREDICTED	CASE AND FEATURES USED	MODEL
z35	MULTIVARIATE (z35, d_mean, d_std, d_min, d_max, weekday)	Bagging Regressor (BR_multi_z35_d_std)
z35_Standard Deviation (6h_std)	UNIVARIATE (6h_std)	Extra Gradient Boosting Regressor (XGBR_uni_6h_std)
z35_Variance (6h_var)	UNIVARIATE (6h_var)	Extra Gradient Boosting Regressor (XGBR_uni_6h_var)
z35_Minimum value (6h_min)	UNIVARIATE (6h_min)	Bagging Regressor (BR_uni_6h_min)
z35_Maximum value (6h_max)	UNIVARIATE (6h_max)	Bagging Regressor (BR_uni_6h_max)

Table 3. Best Ensemble models for predicting z35 and its features

We report also the plots of error metrics score R2, MAE and RMSE of the five best models chosen, one for each features of interest. In each plot, the first column represent the error score of z35 best ensemble model, the second column of z35 standard deviation best ensemble model, the third column of z35 variance best ensemble model, the fourth column of z35 minimal value best ensemble model and the fifth column of z35 maximal value best ensemble model.

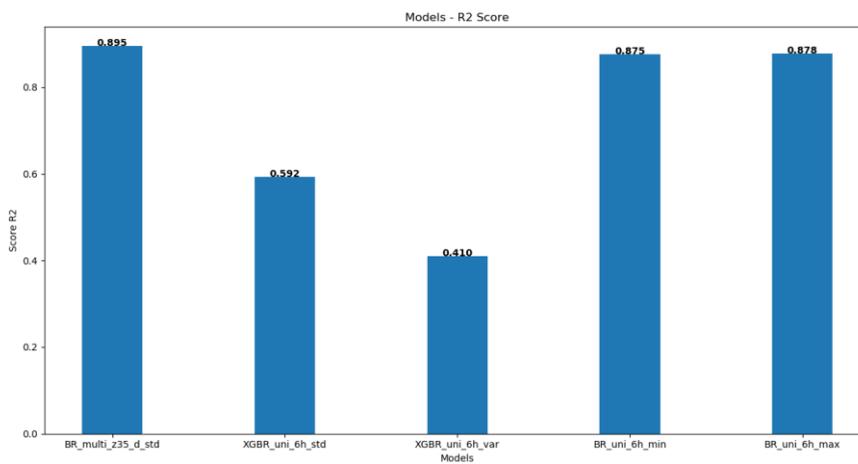


Figure 319. R2 Scores best Ensemble models

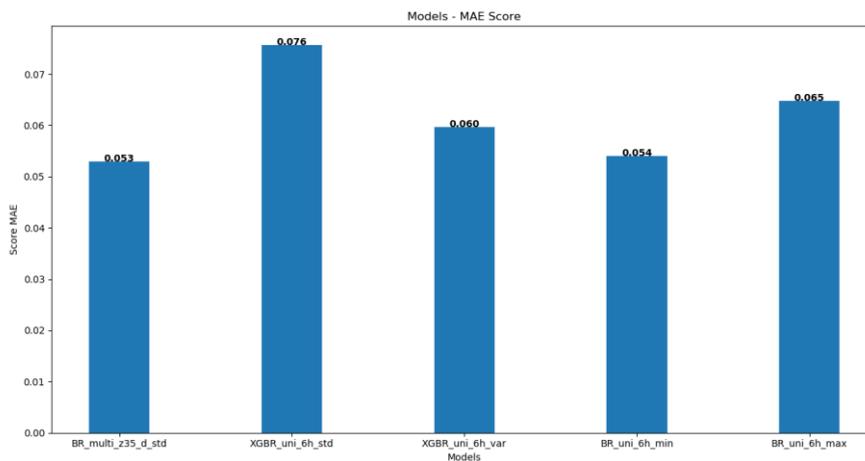


Figure 320. MAE Scores best Ensemble models

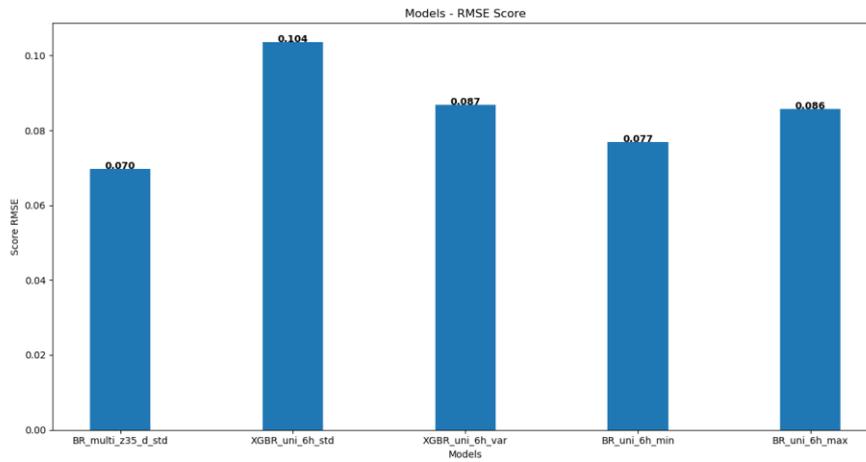


Figure 321. RMSE Scores best Ensemble models

From analysis of best Ensemble Regression models found for each of the features we want to predict, we can observe that also here as in Linear models, the influence of the extra features added in Multivariate models is not too much noticeable, this is proved from the fact that only the model for predicting z35 is a Multivariate model and it doesn't improve too much the prediction performance than the Univariate model (considering figures 277 and 310 for comparison on error metrics). As we observed also from the previous test, ensemble models that uses only Bagging method (like Random Forest Regressor) are not too much suitable for time series forecasting, in fact the only Bagging model used in the best models have as base estimators ensemble models that use Boosting technique. In Bagging models the final prediction that is the average of the predictions of each base models can increase the accuracy and control overfitting, but it never show a real significative increment in prediction performance.

2.4 Multioutput Regression for Linear and Ensemble Models

Normally for Multioutput Regression in time series forecasting we intend predicting two or more features in output than only one like in the models we have realized until now (obviously considering Multivariate case), e.g. realizing a model that can predict both z35 minimal value and maximal value. To understand, referring to Linear models realized, at each iteration during training we have one weight to fit in Univariate case and six weights to fit in Multivariate case (because we use six features in input), these are scalar weights and have the advantage to maintain a correlation between all the points in the time series. What we want to do is using Multioutput Regression not in the standard way to predict in output more than one features, but considering the number of timesteps in input as the number of features used in Multivariate model and the number of timesteps in output to predict as the number of features to predict in Multioutput Regression. It means having to use only Univariate models but we can have a finer dependency between input and output, with however a loss in terms of correlation between points. In this case we can also have different number of timesteps in input and in output, because the constraint in the model during training is on the number of samples uses in input and output that have to be equal and not on the number of features.

In practice, with the possibility to choice the number of timesteps in output, we want to use a time window of `past_history` of a week (672 timesteps) and a time window of `future_target` of a day (96 timesteps). Remembering how to implement Linear and Ensemble models from Scikit-learn library, we need an input of dimension $(n_samples, n_features)$ and an output, considering multioutput case, of dimension $(n_samples, n_features)$. Using Multioutput Regression like previously described, we will have an input of dimension $(n_samples, n_features=n_timesteps=672)$ and an output of dimension

(n_samples, n_features=n_timesteps=96). Considering Linear models it means that the relation between input and output will be:

$$y = wx \quad \text{where} \quad y = [96, 1]; \quad x = [672, 1]; \quad w = [96, 672]$$

So we will have to fit a matrix of weights and not only one weight like in the original Univariate model.

Now we want to verify if using this procedure can lead to an improvement in prediction compared to the original model. All the steps described for the original Linear and Ensemble models are still valid using Multioutput Regression, what change in this case is only the dimension of input and output where we don't apply the flattening operation.

We report the results of Linear Regression model (LR) and Extreme Gradient Boosting Regressor model (XGBR) for predicting z35 in Univariate case, with Multioutput Regression and in the original version.

- Linear Regression (LR)
 - *Multioutput Regression*
-

Model name: LR_uni_z35

Feature used: z35

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

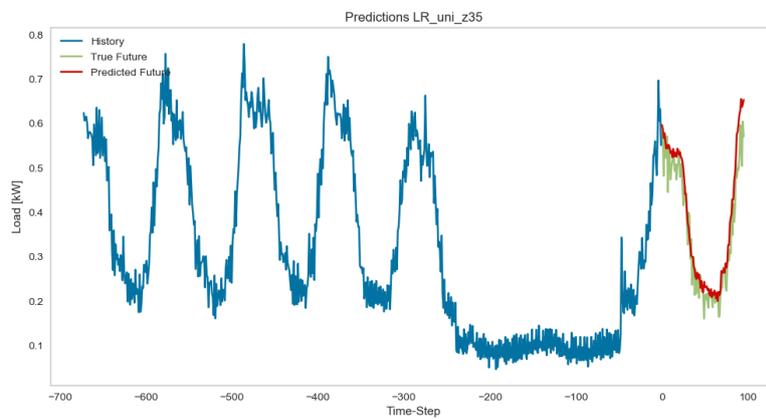


Figure 322. Example of one LR model prediction for predicting z35 with Multiouput

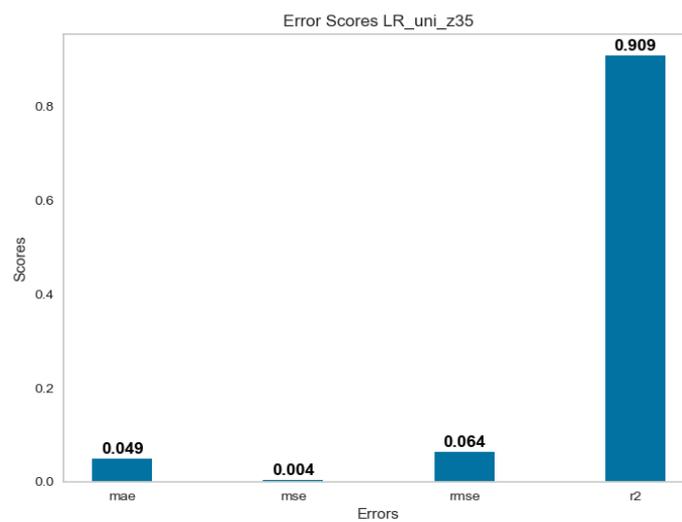


Figure 323. Error metrics score calculated on all samples of test set

- *No Multioutput Regression*

Model name: LR_uni_z35

Feature used: z35

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

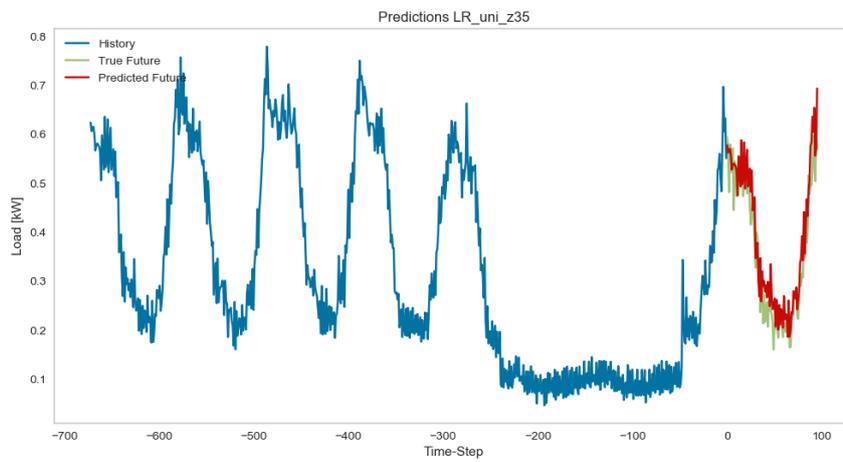


Figure 324. Example of one LR model prediction for predicting z35 without Multiouput

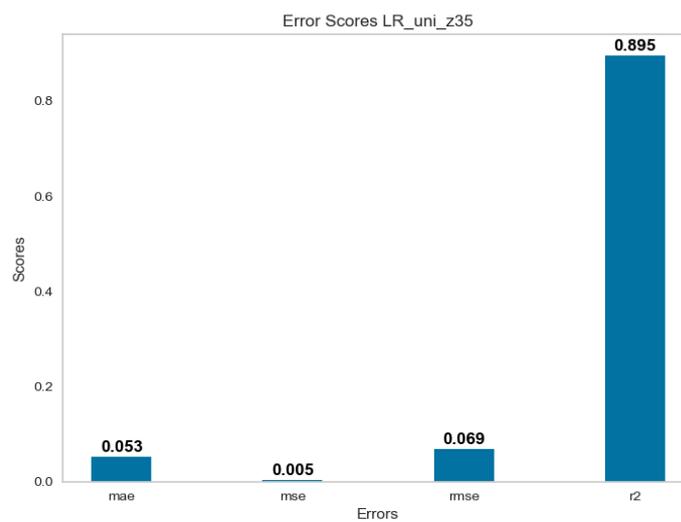


Figure 325. Error metrics score calculated on all samples of test set

- Extreme Gradient Boosting Regressor (XGBR)
 - *Multioutput Regression*
-

Model name: XGBR_uni_z35

Feature used: z35

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

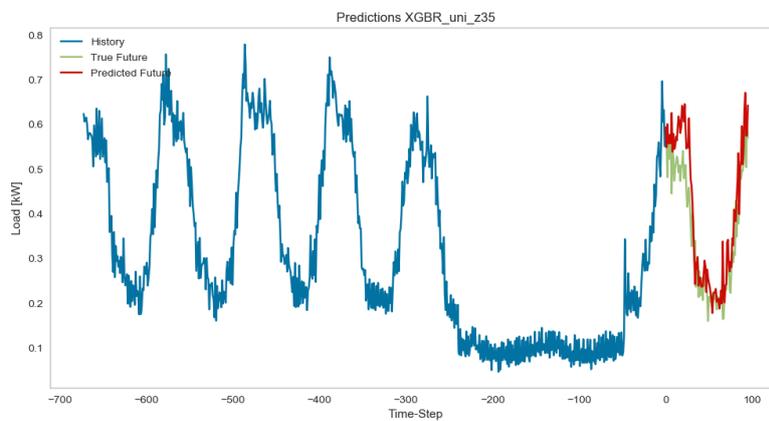


Figure 326. Example of one XGBR model prediction for predicting z35 with Multiouput

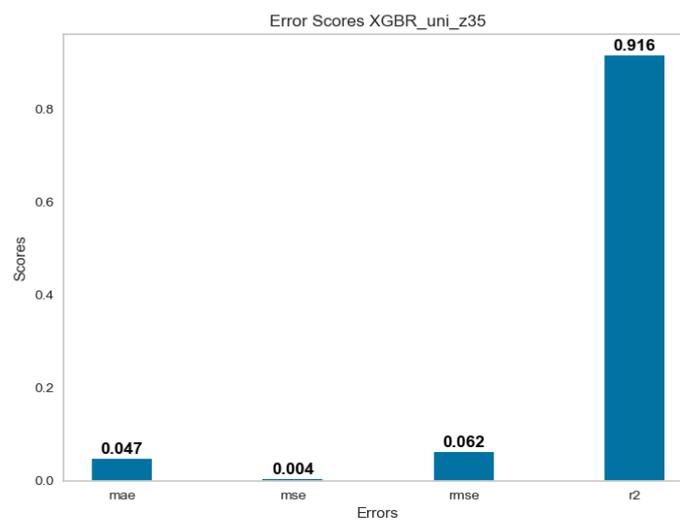


Figure 327. Error metrics score calculated on all samples of test set

- *No Multioutput Regression*

Model name: XGBR_uni_z35

Feature used: z35

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=672, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=672, n_features=1)$

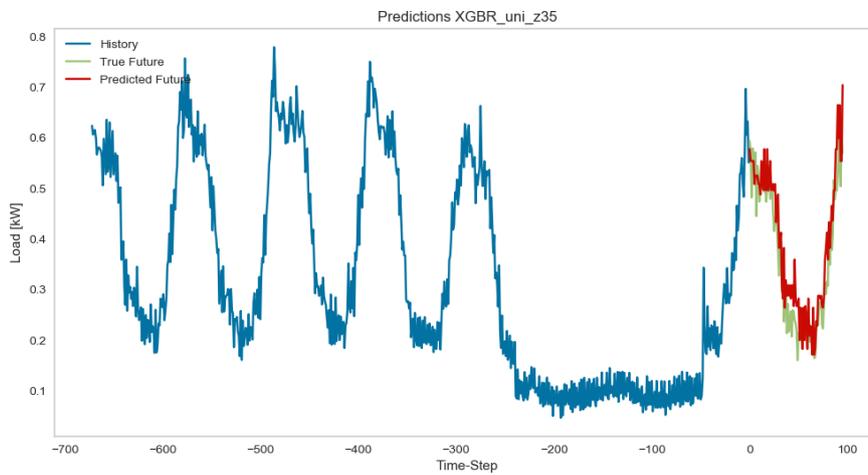


Figure 328. Example of one XGBR model prediction for predicting z35 without Multioutput

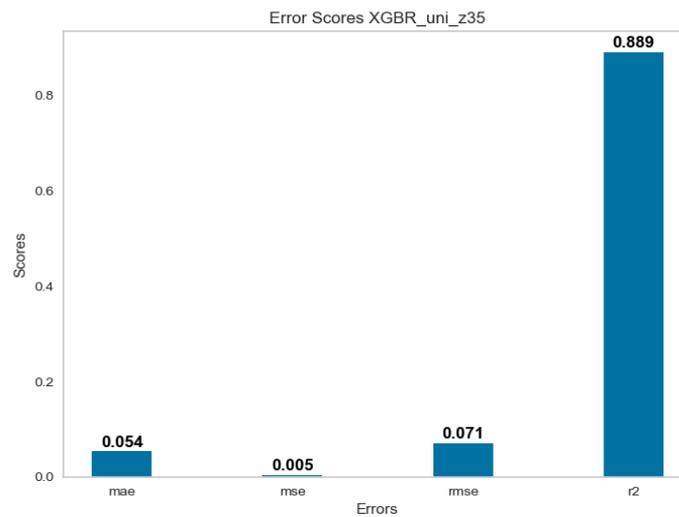


Figure 329. Error metrics score calculated on all samples of test set

From comparison of error metrics plot between Multioutput and no Multioutput Regression models, both for Linear Regression (LR) and Extreme Gradient Boosting Regressor (XGBR), we can observe that Multioutput models have better prediction performance than the original Univariate models. We can directly observe from the plots of the predictions that Multioutput model is characterized by a lower variance compared to the original model.

Now we want test Multioutput Regression procedure to the best models found between Linear and Ensemble models families for predicting z35 and the features extracted of interest in Univariate case, to verify if we can obtain a better model for predicting each feature using Multioutput Regression.

The best models for predicting z35 and its features in Univariate case between Linear and Ensemble regression, comparing relative error metrics scores, are:

- **z35**: Linear Regression (LR);
- **z35_daily_std**: Passive Aggressive Regressor (PAR);
- **z35_daily_var**: Stochastic Gradient Descent Regressor (SGDR);
- **z35_6hours_min**: Linear Regression (LR);
- **z35_daily_max**: Passive Aggressive Regressor (PAR).

The best models are all Linear models. Now we apply Multioutput procedure and test the prediction performance.

- Prediction of **z35**
 - *Multioutput Regression*
-

Model name: LR_uni_z35_multioutput

Feature used: z35

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

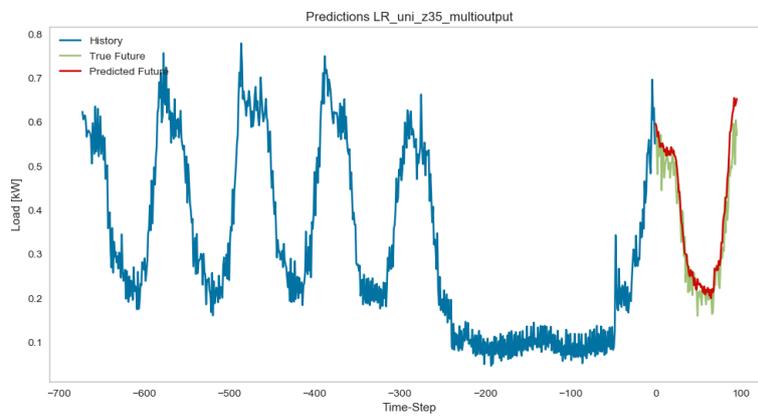


Figure 330. Example of one LR model prediction for predicting z35 with Multioutput

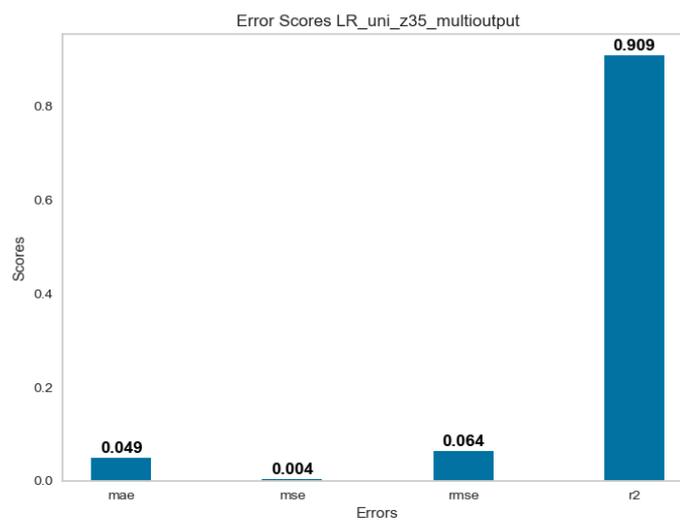


Figure 331. Error metrics score calculated on all samples of test set

- Prediction of **std**
 - *Multioutput Regression*
-

Model name: PAR_uni_d_std_multioutput

Feature used: z35_daily_std

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

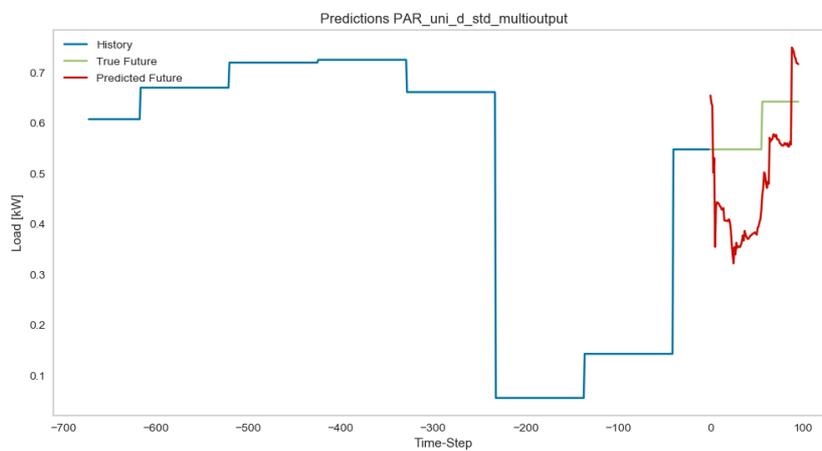


Figure 332. Example of one PAR model prediction for predicting z35_std with Multiouput

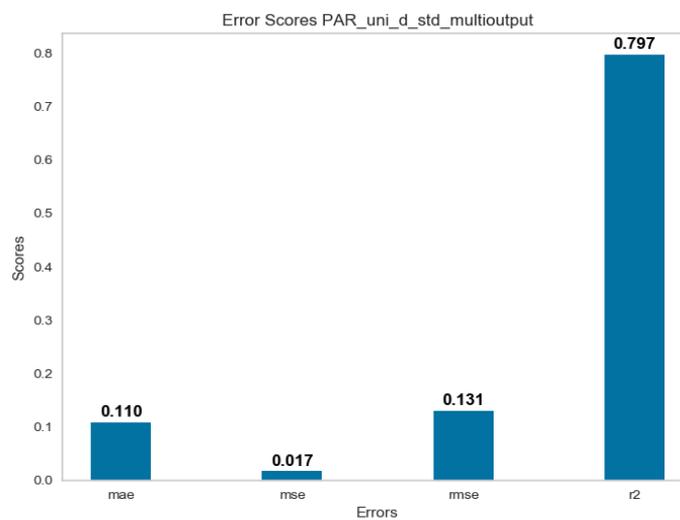


Figure 333. Error metrics score calculated on all samples of test set

- Prediction of **var**
 - *Multioutput Regression*
-

Model name: SGDR_uni_d_var_multioutput

Feature used: z35_daily_var

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

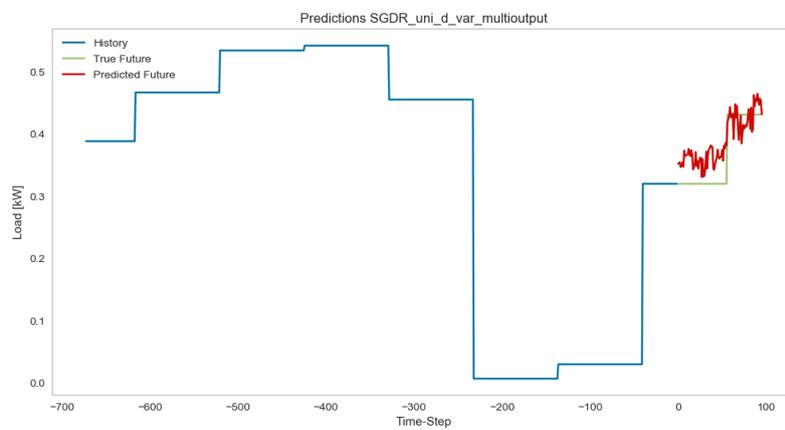


Figure 334. Example of one SGDR model prediction for predicting z35_var with Multioutput

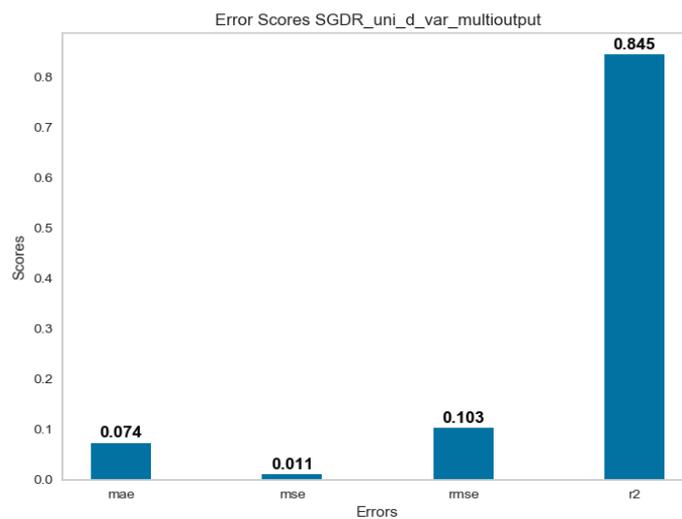


Figure 335. Error metrics score calculated on all samples of test set

- Prediction of **min**
 - *Multioutput Regression*
-

Model name: LR_uni_6h_min_multioutput

Feature used: z35_6hours_min

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

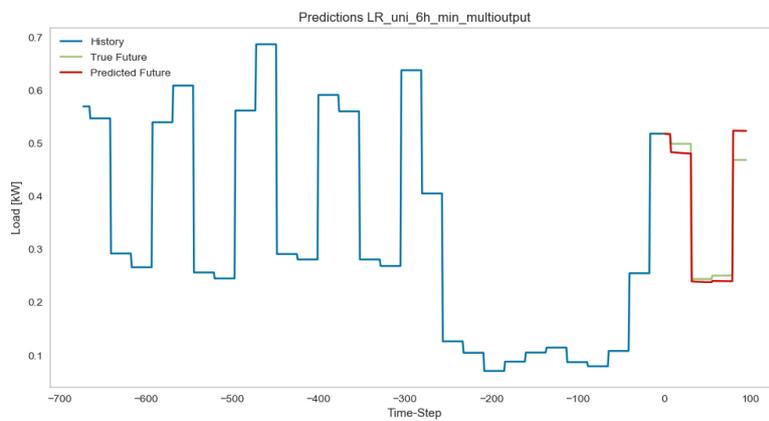


Figure 336. Example of one LR model prediction for predicting z35_min with Multiouput

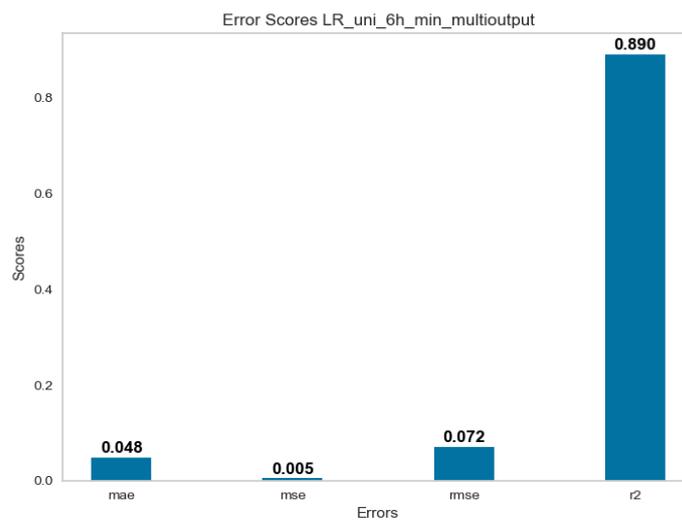


Figure 337. Error metrics score calculated on all samples of test set

- Prediction of **max**
 - *Multioutput Regression*
-

Model name: PAR_uni_d_max_multioutput

Feature used: z35_daily_max

- Input shape $x = (n_timesteps=672, n_features=1)$
- True_target shape $y = (n_timesteps=96, n_features=1)$
- Predicted_target shape $\hat{y} = (n_timesteps=96, n_features=1)$

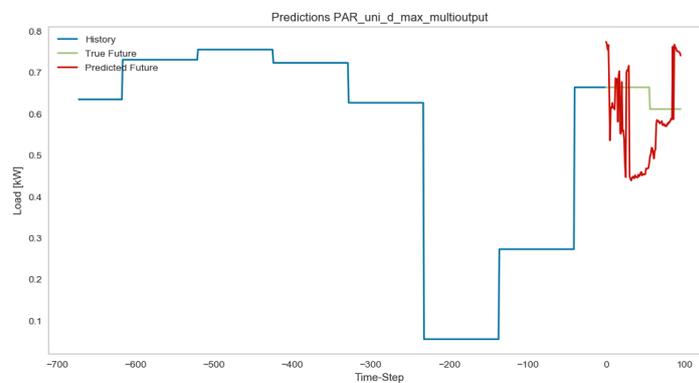


Figure 338. Example of one PAR model prediction for predicting z35_max with Multioutput

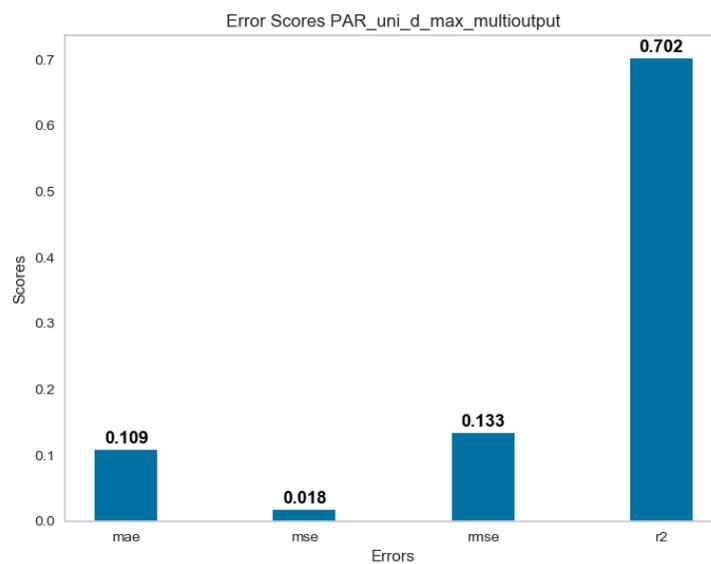


Figure 339. Error metrics score calculated on all samples of test set

We report also the plot of error metrics R2, MAE and RMSE of all the five models. In each plot, the first column represent the error score of z35 model, the second column of z35 standard deviation model, the third column of z35 variance model, the fourth column of z35 minimal value model and the fifth column of z35 maximal value model.

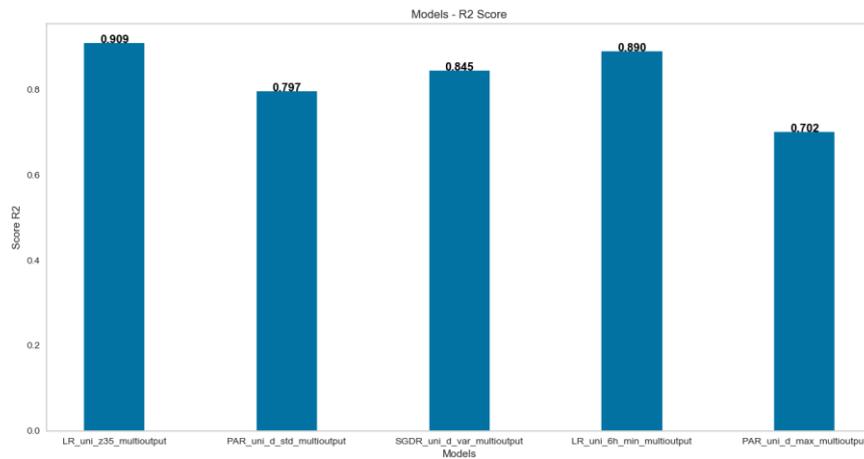


Figure 340. R2 Scores Multioutput Regression models

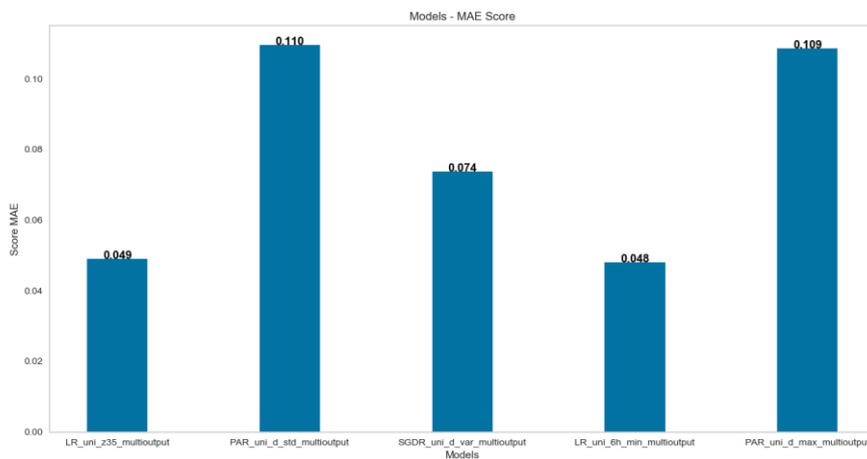


Figure 341. MAE Scores Multioutput Regression models

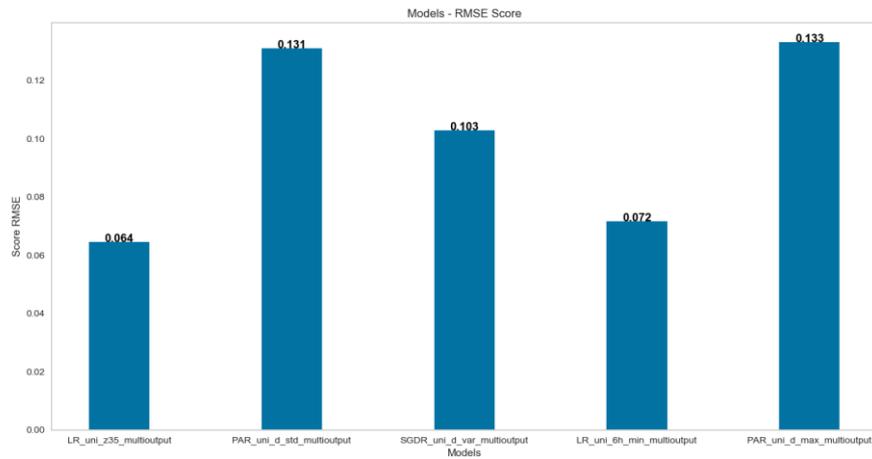


Figure 342. RMSE Scores Multioutput Regression models

Comparing the error metrics score of the Multioutput Regression model with the error metrics score of the best model found for predicting each features of interest belonging to families RNN, Linear and Ensemble, we can observe that Multioutput Regression models are not the best in any features of interest to predict. So we discard this procedure for finding the best models, moreover for applying it we use theoretically the Scikit-learn library in the wrong way and has the constraint of being an Univariate model. An improvement in the results using Multioutput regression can derive from implementation of a real Multivariate model.

2.5 Conclusions on Best Models

At this point, considering best models found for predicting z35 and its features belonging to the three different groups of regression models tested that are RNN, Linear and Ensemble models, we can obtain the best global models for predicting the five features of interest. We report a table with the error metrics score R2, MAE and RMSE of the best models for each group and feature to predict.

z35	MAE	RMSE	R2
LINEAR MODEL (LR_uni_z35)	0,054	0,071	0,891
ENSEMBLE MODEL (BR_multi_z35_d_std)	0,053	0,070	0,895
RNN MODEL (z35_d_var_norm_multi_6)	0,047	0,061	0,897
Standard Deviation	MAE	RMSE	R2
LINEAR MODEL (SGDR_multi_d_std)	0,068	0,078	0,929
ENSEMBLE MODEL (XGBR_uni_6h_std)	0,076	0,104	0,592
RNN MODEL (d_std_norm_multi_6)	0,083	0,117	0,714
Variance	MAE	RMSE	R2
LINEAR MODEL (LR_multi_6h_var)	0,053	0,078	0,524
ENSEMBLE MODEL (XGBR_uni_6h_var)	0,060	0,087	0,410
RNN MODEL (6h_var_norm_multi_6)	0,060	0,083	0,378
Minimum value	MAE	RMSE	R2
LINEAR MODEL (LR_multi_6h_min_6h_std)	0,050	0,074	0,884
ENSEMBLE MODEL (BR_uni_6h_min)	0,054	0,077	0,875
RNN MODEL (6h_min_6h_std_norm_multi_6)	0,074	0,101	0,725
Maximum value	MAE	RMSE	R2
LINEAR MODEL (PAR_uni_d_max)	0,057	0,072	0,914
ENSEMBLE MODEL (BR_uni_6h_max)	0,065	0,086	0,878
RNN MODEL (6h_max_6h_var_norm_multi_6)	0,073	0,098	0,776

Table 4. Best regression models for predicting z35 and its features

The best global regression model for predicting z35 and its features based on error metrics score (put in evidence in table 4) are:

- **z35:**

Model: RNN Multivariate

Model name: SGDR_multi_d_std

Feature used: z35, z35_daily_mean, z35_daily_std, z35_daily_min, z35_daily_max, weekday

- **z35_daily_std:**
Model: Linear Multivariate
Model name: z35_d_var_norm_multi_6
Feature used: z35, z35_daily_mean, z35_daily_var, z35_daily_min, z35_daily_max, weekday
- **z35_6hours_var:**
Model: Linear Multivariate
Model name: LR_multi_6h_var
Feature used: z35, z35_daily_mean, z35_6hours_var, z35_6hours_min, z35_6hours_max, weekday
- **z35_6hours_min:**
Model: Linear Multivariate
Model name: LR_multi_6h_min_6h_std
Feature used: z35, z35_daily_mean, z35_6hours_std, z35_6hours_min, z35_6hours_max, weekday
- **z35_daily_max:**
Model: Linear Univariate
Model name: PAR_uni_d_max
Feature used: z35_daily_max

We can observe that the best model for predicting z35 is the RNN model in Multivariate case while for predicting the features extracted from z35 that are standard deviation, variance, minimum and maximum value the best models are the Linear models. It probably depends on the type of time series to predict, because the features extracted from z35 have values that repeated for a certain number of timesteps, depending on the time interval used to calculate them, while z35 time series have different values at each timesteps, so a Linear model is a good choice for fitting the first type of time series, instead RNN model is suitable for time series like z35. We have also noticed that RNN models shows an improvement in prediction performance using the right combination of extra

features in Multivariate case, while it's not always true regarding Linear models. The further steps for having a more detailed research on which are the best models for predicting electric load time series and its features, considering the same three groups of regression models, are testing different configuration of the hyperparameters of RNN models (like number of unit in the layers, different structures of the layers, different values for the learning rate in Adam optimizer); for Linear models using Grid Search process also considering Multivariate case for predicting z35 and for the other features extracted, because in our approach we have assumed that the hyperparameter optimized for model z35 Univariate was suitable also for the other features because they derived from z35; for ensemble models we have to concentrate on models based on Boosting technique because for time series forecasting are the most promising than models based on Bagging technique.

2.6 Generalization of Best Models

For generalization of best models, we intend using the five best models found for predicting z35, z35 standard deviation, z35 variance, z35 minimal value and z35 maximal value, applied to a different time series always related to electric load power.

In dataset “load_15min_avrg_monitoring_server_v2.csv” besides z35 time series there are other electric load time series with values always sampled every 15 minutes from 2017-09-30 at 23:15:00 to 2018-08-01 at 00:00:00, with a total of 29188 samples.

Those time series showing the same characteristics of z35, like weekly seasonality, but have similar or different range of values. For this reason it was important training models on normalized data with values between 0 and 1, in this way we can use efficiently the best models found on these new time series, because after applying scaling on new data we will always have values between 0 and 1. Based on characteristics of the new electric load time series, we can adopt different approach for applying best models found. Theoretically we apply scaling technique calculating the scale factors on training data and then applying them on new unseen data. In this case these new data are not of the same times series, so the scale factors may have to be recalculated. In some cases could be more convenient retraining the same models (with the same configuration of the hyperparameters) but on the new time series data.

We will test generalization of best models found on these new electric load time series:

- Time series **z36**: data have a range of values similar to z35 time series;
- Time series **z32**: data have a range of values lower than z35 time series;

- Time series **z34**: data have a range of values a little upper than z35 time series and there is the presence of outliers;
- Time series **True_load**: this time series is not part of dataset “load_15min_avrg_monitoring_server_v2.csv”, but it’s the time series that will be used in the simulation model. It has values in a range upper than z35.

We report z35 time series with the plot of the distribution of the values, so we can analyze the differences between the original time series used for training the models and the time series used for generalizing them.

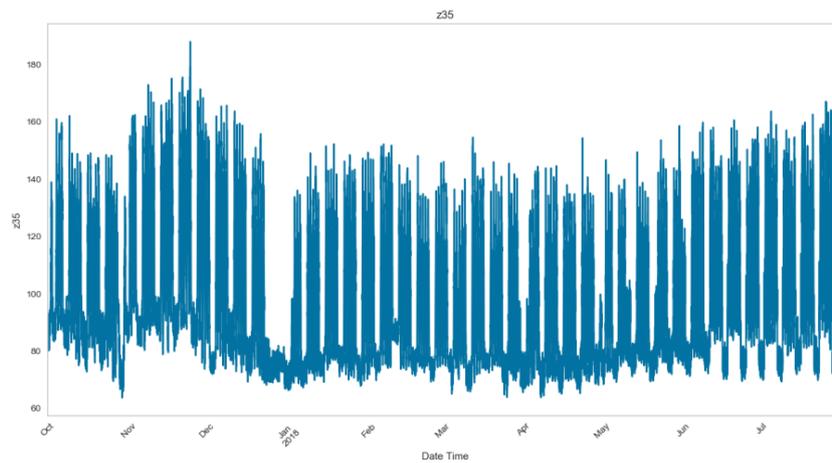


Figure 343. z35 Time Series

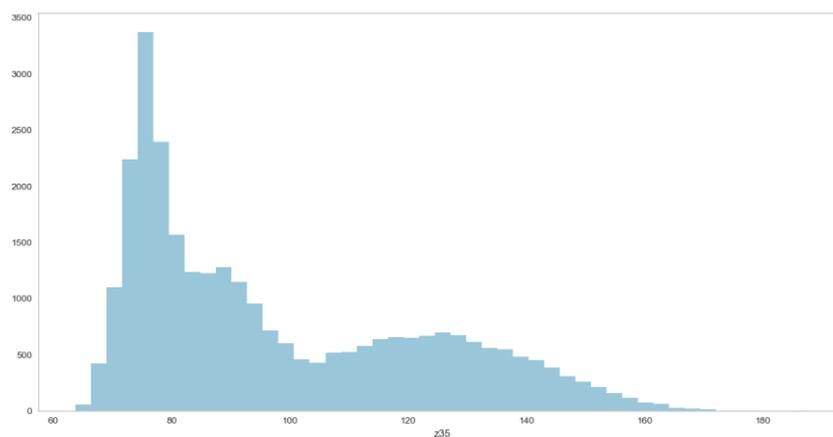


Figure 344. Plot of the distribution of values of z35

- Time series **z36**

Time series z36 contains 29188 values of electric load data of HRI facility, the values are sampled every 15 minutes for the same period of z35 time series. Like we have done for z35 time series, also in z36 time series we discard the first three values and the last values of electric load to have exactly 304 days of samples, so a total of 29184 values sampled from 2017-10-01 at 00:00:00 to 2018-07-31 at 23:45:00.

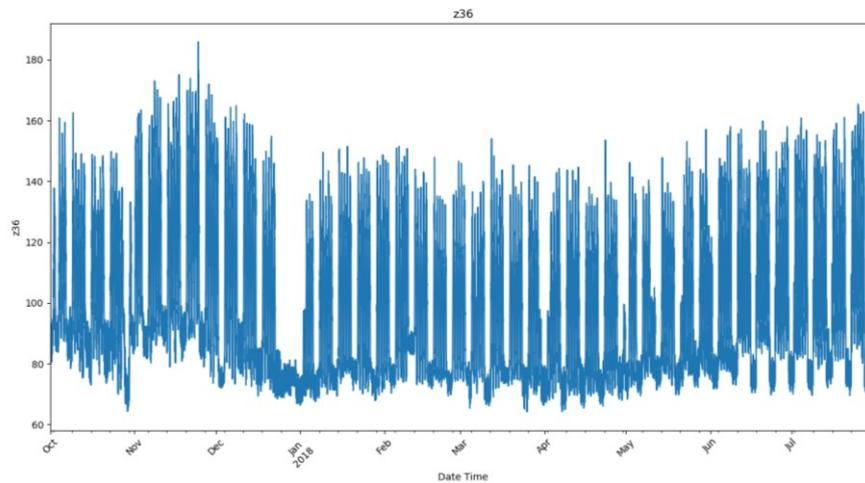


Figure 345. z36 Time Series

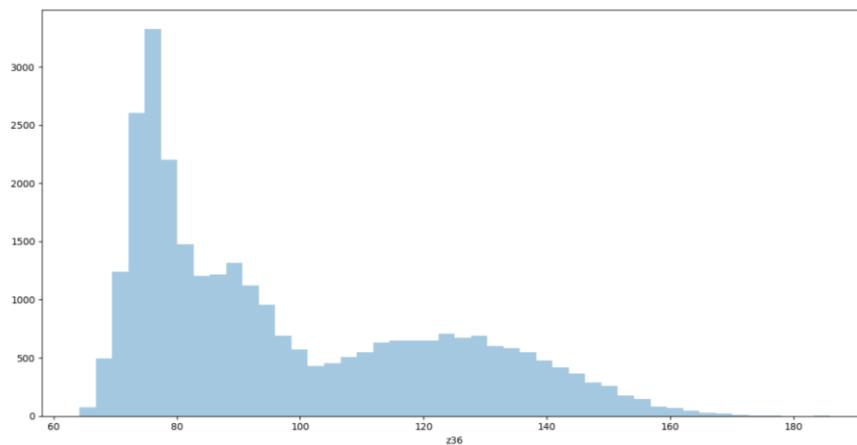


Figure 346. Plot of the distribution of values of z36

We can observe from figures that z36 time series is almost equal to z35 time series, with the same range and distribution of values. In this case we don't need to retrain the models on z36 time series. For applying normalization on z36 data and verifying the prediction performance of best models found we can follow two ways: using the same scale factors calculate on z35 or recalculating the scale factors on training data of z36 time series.

We report the error metrics score of each models in the two cases:

CASE 1) Prediction using z35 Scale Factors on z36 data	MAE	RMSE	R2
z36_d_var_norm_multi_6	0,047	0,061	0,897
SGDR_multi_d_std	0,116	0,175	0,698
LR_multi_6h_var	0,054	0,083	0,532
LR_multi_6h_min_6h_std	0,062	0,095	0,797
PAR_uni_d_max	0,101	0,155	0,676

Table 5. Case 1: Error metrics z36 models using z35 Scale Factors

CASE 2) Prediction using new Scale Factors on z36 data	MAE	RMSE	R2
z36_d_var_norm_multi_6	0,047	0,062	0,896
SGDR_multi_d_std	0,117	0,177	0,700
LR_multi_6h_var	0,058	0,090	0,531
LR_multi_6h_min_6h_std	0,062	0,095	0,797
PAR_uni_d_max	0,103	0,158	0,677

Table 6. Case 2: Error metrics z36 models using new Scale Factors

In tables we have put in evidence in which case the error metrics score of a model is better. As we could expected, the best choice is to use the same scale factors calculated for z35 time series for applying normalization, because the time series are very similar. The models used are trained on z35 time series data in each cases.

- Time series **z32**

Time series z32 contains 29188 values of electric load data of HRI facility, the values are sampled every 15 minutes for the same period of z35 time series. Also in z32 time series we discard the first three values and the last values of electric load to have exactly 304 days of samples, so a total of 29184 values sampled from 2017-10-01 at 00:00:00 to 2018-07-31 at 23:45:00.

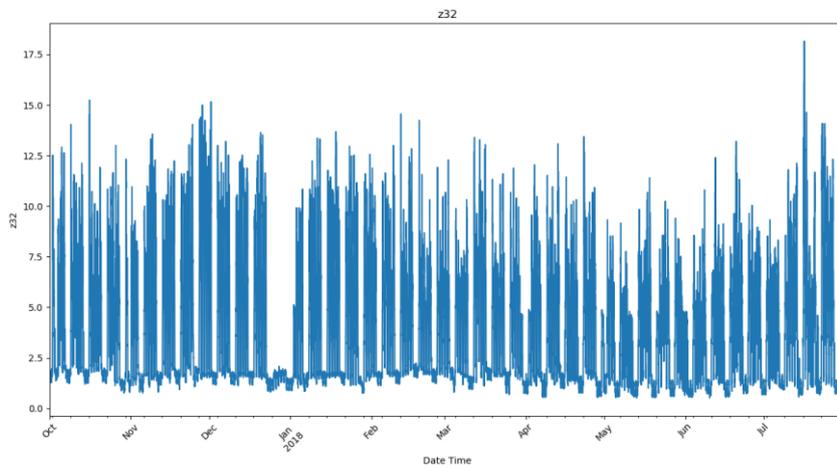


Figure 347. z32 Time Series

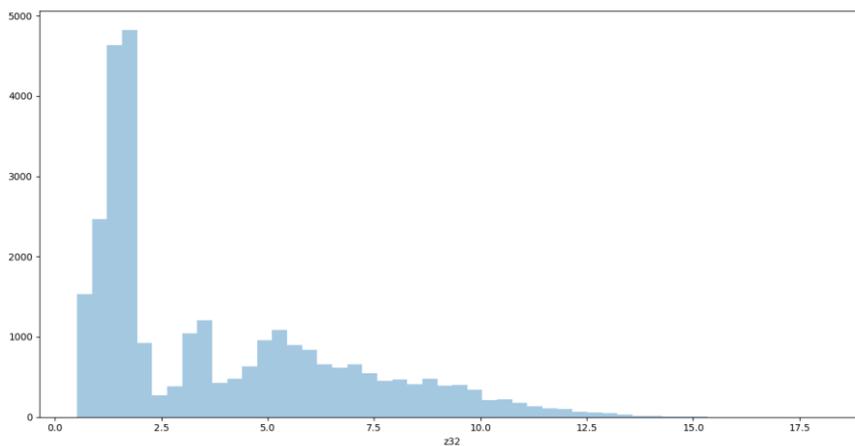


Figure 348. Plot of the distribution of values of z32

We can observe from figures that z32 time series has a lower range of values than z35 time series. z32 time series shows the same behaviour of z35 time series in weekday and weekend days and it's also characterized by a weekly seasonality of the values. The only difference is the range of values of the samples. In this case the best choice for generalizing the models is to recalculate the scale factors on z32 training data, because we can't use the scale factors of z35, then we use best model trained on z35 data and calculate the error metrics score.

We report the error metrics score of each models:

CASE - Prediction using new Scale Factors on z32 data	MAE	RMSE	R2
z32_d_var_norm_multi_6	0,084	0,107	0,509
SGDR_multi_d_std	0,137	0,184	0,549
LR_multi_6h_var	0,051	0,083	0,391
LR_multi_6h_min_6h_std	0,061	0,096	0,714
PAR_uni_d_max	0,121	0,170	0,579

Table 7. Error metrics z32 models using new Scale Factors

In case the prediction aren't enough accurate, we can always retrain the models on z32 time series, because it shows the same pattern and characteristics of z35 time series, so we don't have to change the configuration of each model.

- Time series **z34**

Time series z34 contains 29188 values of electric load data of HRI facility, the values are sampled every 15 minutes for the same period of z35 time series. Also in z34 time series we discard the first three values and the last values of electric load to have exactly 304 days of samples, so a total of 29184 values sampled from 2017-10-01 at 00:00:00 to 2018-07-31 at 23:45:00.

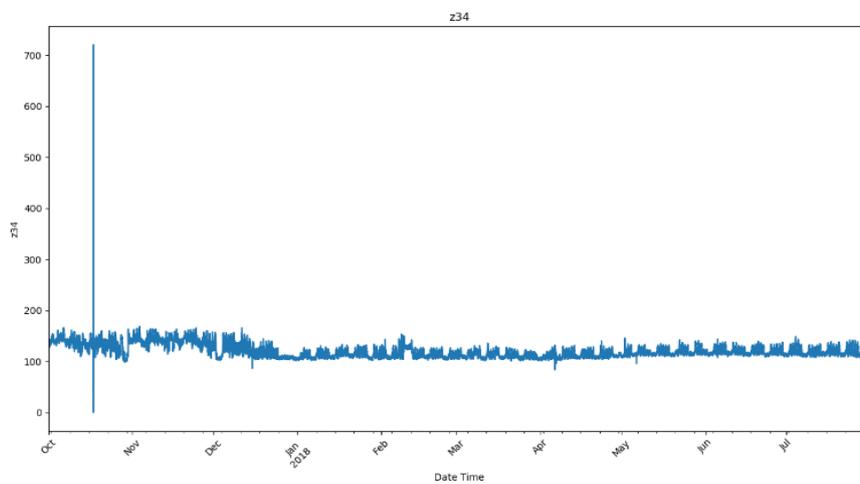


Figure 349. z34 Time Series

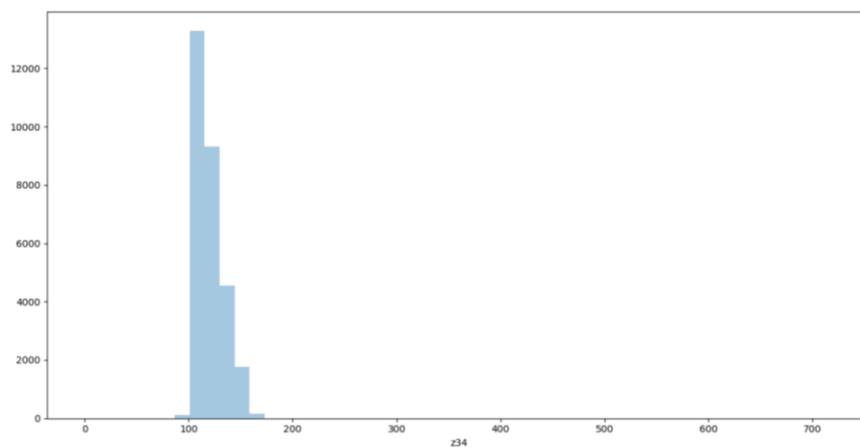


Figure 350. Plot of the distribution of values of z34

We can observe from plot of z34 time series the presence of outliers, in this case before applying normalization we have to remove them, because scale factors for normalization using minimum value and maximum value of the time series. We substitute outliers values with median value of the time series using standardization, so we set a threshold standardized value and if the absolute value standardized of the time series is bigger than this threshold is substituted with median value. We report the plot of z34 time series and the distribution of the values after removed the outliers.

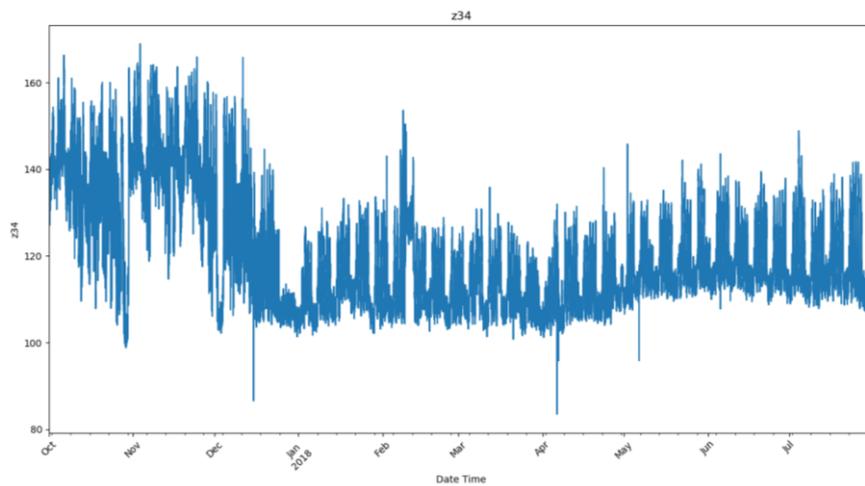


Figure 351. z34 Time Series after removed Outliers

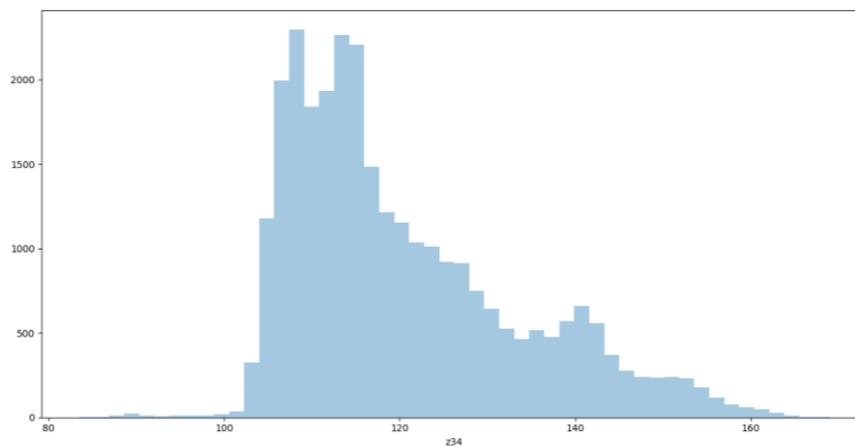


Figure 352. Plot of the distribution of values of z34 after removed Outliers

After removed outliers, we can observe that the range of values of z34 time series is a little upper than z35 time series. Also z34 time series shows the typical pattern of electric load time series with the weekly seasonality, so we can predict z34 and its features applying the best models trained on z35 and recalculating the scale factors for normalization on z34 time series.

CASE - Prediction using new Scale Factors on z34 data	MAE	RMSE	R2
z34_d_var_norm_multi_6	0,165	0,194	-3,875
SGDR_multi_d_std	0,140	0,177	0,179
LR_multi_6h_var	0,060	0,087	-0,053
LR_multi_6h_min_6h_std	0,067	0,106	0,391
PAR_uni_d_max	0,105	0,150	0,624

Table 8. Error metrics z34 models using new Scale Factors

Also here are valid the considerations made for time series z32, so if the prediction performance are not too much accurate, we can retrain the same configuration of the models on z34 time series.

- Time series **True_load**

Time series True_load contains 2976 values of electric load data of HRI facility, the values are sampled every 15 minutes from 2019-06-01 at 00:00:00 to 2019-07-01 at 23:45:00, so a total of 31 days of samples.

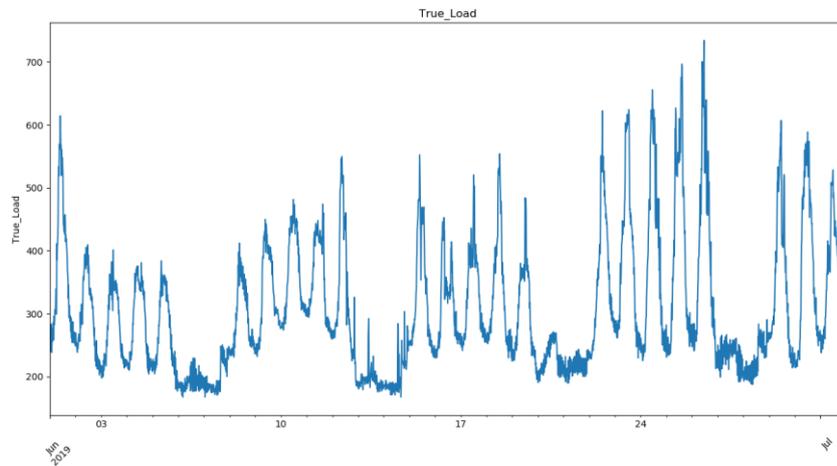


Figure 353. True_load Time Series

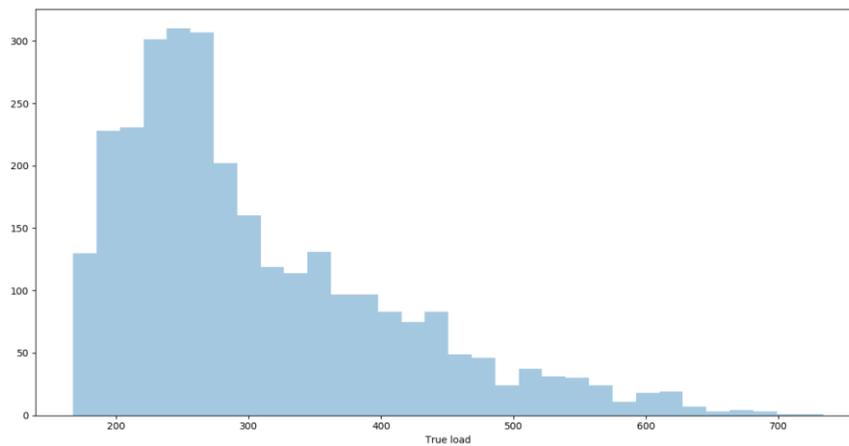


Figure 354. Plot of the distribution of values of True_load

We can observe that True_load time series has an upper range of values than z35 time series. True_load time series shows the same pattern of z35 time series in weekday and weekend days and it's also characterized by a weekly seasonality of the values. Also in this case the best choice for generalizing the models is to recalculate the scale factors on True_load training data, because we can't use the scale factors of z35, then we use best models trained on z35 data and calculate the error metrics score.

We report the error metrics score of each models:

CASE - Prediction using new Scale Factors on True_Load data	MAE	RMSE	R2
True_Load_d_var_norm_multi_6	0,132	0,182	-0,172
SGDR_multi_d_std	0,187	0,227	0,496
LR_multi_6h_var	0,109	0,187	0,290
LR_multi_6h_min_6h_std	0,132	0,191	0,525
PAR_uni_d_max	0,165	0,207	0,529

Table 9. Error metrics True_load models using new Scale Factors

In this case the choice of retrain the same models on True_load time series data can lead to worst prediction performance, because the number of samples of the time series True_load is approximatively ten times less than z35 time series, a number probably too low for training the models in efficient way and having an improvement in prediction performance.

2.7 EMS MPC Simulation Model

In this Simulation model we are going to use the predictions obtained from the best models found for predicting electric load time series and related features for testing how they affect the results of the simulations.

The EMS MPC Simulation Framework is a modular Matlab simulation framework for simulating the control of (monolithic) state space systems using model predictive control (MPC).

The framework was built with the use-case of an Energy Management System (EMS) in mind. This EMS is a central entity within an energy grid, which tries to control energy providers and consumer within its reach in an economically optimal way.

The framework is fully modular and the EMS-specific features are optional, such that it can be used to simulate any state space system controlled using any type of MPC method.

For using the simulation model in Matlab we have to install the Optimization Toolbox that provides functions for finding parameters that minimize or maximize objectives while satisfying constraints. The optimization toolbox used for model and solve our optimization problem is YALMIP with Gurobi as solver.

In the MPC model, the cost function to minimize is quadratic, so we use Gurobi as a solver of quadratic programming (QP) optimization problems. A quadratic cost function for optimization is given by (to minimize without violating constraints):

$$J = \sum_{i=1}^N w_{x_i} (r_i - x_i)^2 + \sum_{i=1}^N w_{u_i} \Delta u_i^2$$

Equation 13. Example of a quadratic cost function

Where:

x_i : i^{th} controlled variable (e.g. measured temperature)

r_i : i^{th} reference variable (e.g. required temperature)

u_i : i^{th} manipulated variable (e.g. actuator)

w_{x_i} : weighting coefficient reflecting the relative importance of x_i

w_{u_i} : weighting coefficient penalizing relative big changes in u_i

In our MPC model, we have two-objective optimization problem, so we have two objective function to be optimized simultaneously. Multi-objective optimization (or Pareto Optimization) is applied where optimal decisions need to be taken in the presence of tradeoff between two or more conflicting objectives. In fact for a nontrivial multi-objective optimization problem, no single solution exists that simultaneously optimizes each objective. A solution is called nondominated (or Pareto optimal) if none of the objective functions can be improved in value without degrading some of the other objective values.

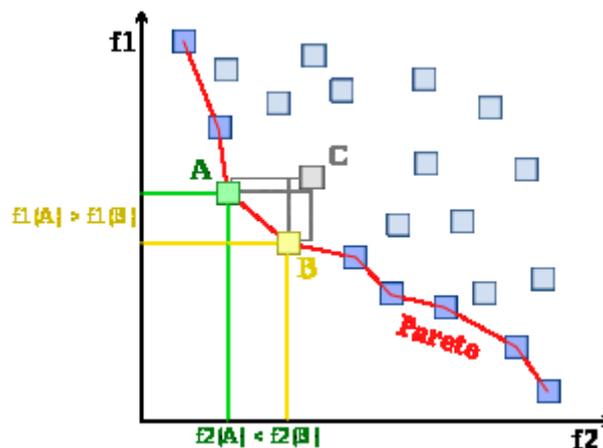


Figure 355. Example of Pareto Optimization – in Pareto frontier (in red) we have all the Pareto optimal solutions

Now we can describe how the simulation works. Every simulation we run have a time horizon of one day. In the Simulation Framework we use Pareto Optimization to find the best tradeoff between Temperature Deviation and Monetary costs, it means that we have a comfort value of Temperature (21°C) that we want to maintain for that day without too much deviation, meanwhile we want to minimize Monetary costs for Heating or Cooling, depending on outside Temperature, and also minimizing Monetary costs depending on Electric Power Demand (here is where we use the prediction data of the best models previously found). We will run the simulations in different scenario and observe how much we can save in terms of Monetary costs with optimization and we will see if in presence of a reduction of Monetary costs, we can maintain the target comfort temperature for all the day.

The predictions data that we use in the simulation model for electric power demand are:

- prediction of **z35** values, its **6hours_min** values and **daily_max** values using the best models previously described;
- prediction of **True_Load** values, its **6hours_min** values and **daily_max** values using the best models trained on **z35** dataset, so we have to generalize these models calculating the new scale factors for normalization on **True_load** dataset.

Each prediction dataset we will use, have a frequency of the values of 30 minutes and no longer of 15 minutes, so we will have 48 prediction per day. The prediction dataset create are:

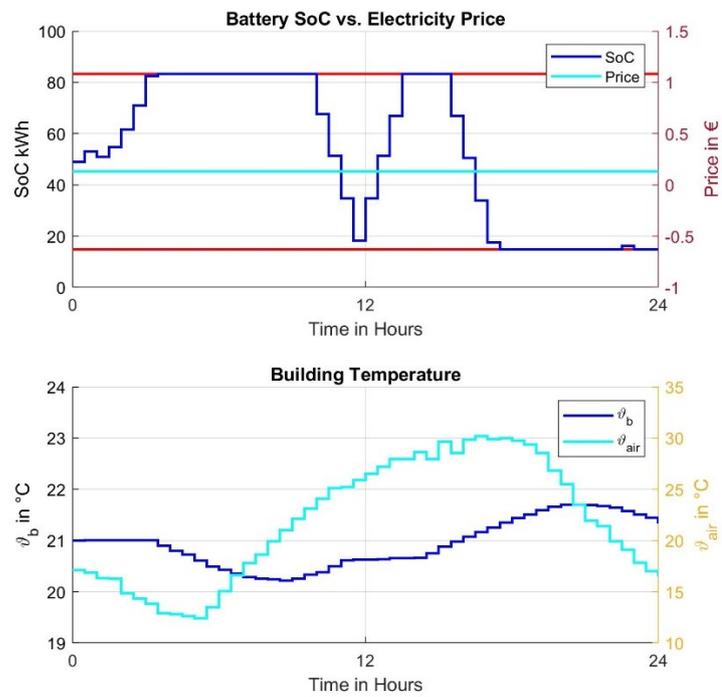
- Predictions of **z35** values from 2018-07-01 at 00:00:00 to 2018-07-30 at 23:30:00, so we will have a total of 1440 prediction data.
- Predictions of **z35_6hours_min** and **z35_daily_max** values from 2018-07-01 at 00:00:00 to 2018-07-31 at 23:30:00, so we will have a total of 1488 prediction data.

- Predictions of **True_Load** values from 2019-06-08 at 00:00:00 to 2019-06-30 at 23:30:00, so we will have a total of 1104 prediction data.
- Predictions of **True_Load_6hours_min** and **True_Load_daily_max** values from 2019-06-08 at 00:00:00 to 2019-07-01 at 23:30:00, so we will have a total of 1152 prediction data.

Now we report the plots results of each simulation executed.

1) Simulation using **real data**

In this case we use as prediction dataset directly the real data of electric load demand, so the results of the simulation will correspond to the real situation that will happen for the target day being predictions of electric load the only predicted dataset used in the simulation model. The target day of the simulation executed is 2018-07-02, so we report the output plots of the simulation for that day using real electric load demand data.



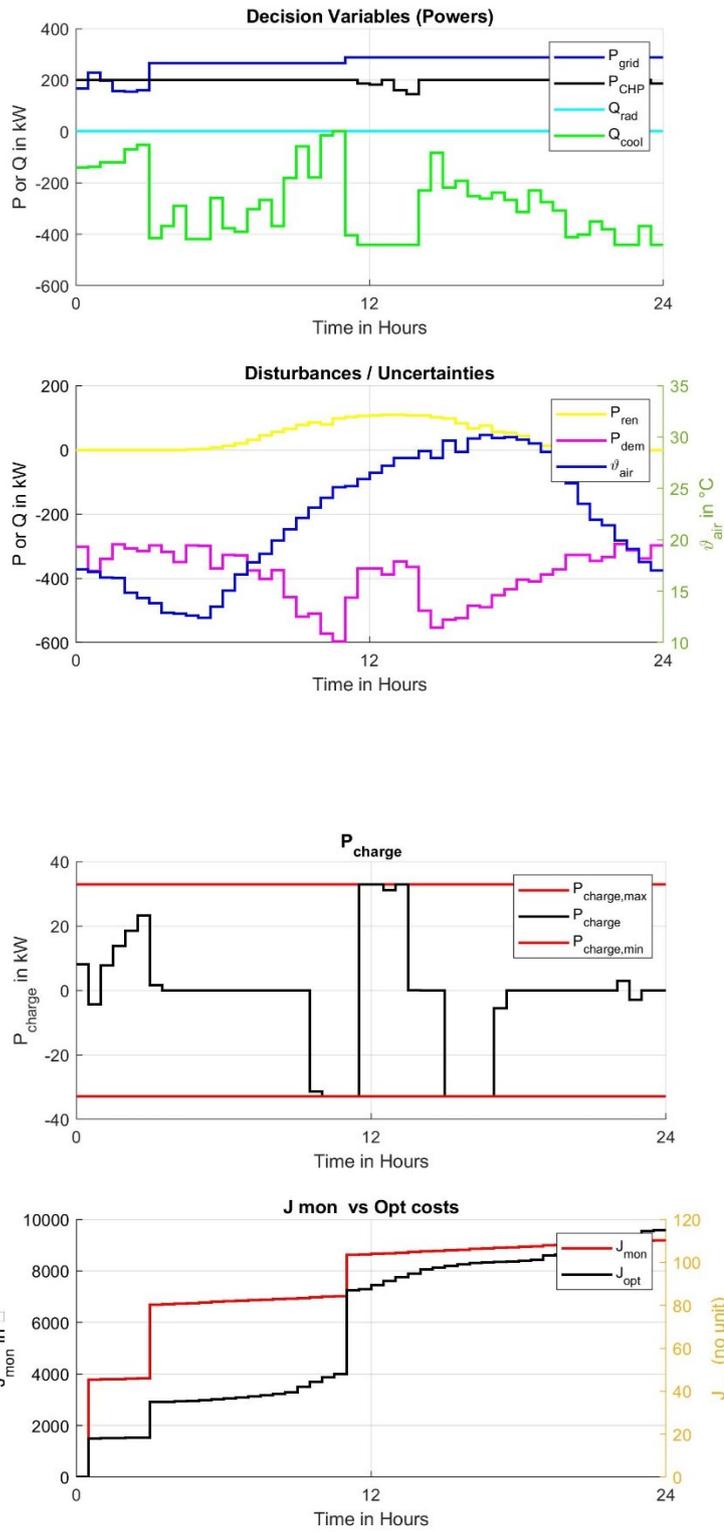


Figure 356. Simulation model result using real data

We have six plots in output as result of the simulation. In the first figure there is the plot of the Battery State of Charge (SoC) with its minimum and maximum values of charge (constraints) and the plot of the Electricity price, that is fixed to 0,131 €/kWh, so the cost of electric energy from the grid. The scenario in the EMS (valid also for the other simulation) is that we have a total of 7 batteries each one with a energy cap of 14 kWh that provides electric energy, the constraints on the Battery SoC are a minimum of total of SoC equal to the 15% and a maximum total SoC equal to the 85%.

In the second figure we have the plot of building temperature and air temperature, here we can observe that one of the objective of the optimization that is maintain the building temperature constant to 21°C has a deviation during the day less than $\pm 1^\circ\text{C}$.

In third figure, we have the plots of Power from the grid and Power from combined heat and power (chp) (these are positive quantities because provides energy to the system) and, in this case, the Power consumed for cooling (in the model the power consumed for heating is positive and the power consumed for cooling is negative and obviously we can't heating and cooling in the same time). If the power consumed for cooling or heating is null, the Power from the grid can be negative because we sell energy to the grid.

In fourth figure we have the plot of the Power Demand for that day (in this figure is always plot the real data of electric power demand, also if we used prediction data for electric power demand in the simulation) and the Power produced by Photovoltaic System that depend by solar radiation and therefore to air temperature.

In the fifth figure we have the plot of Power charge of the batteries during the day with its minimum and maximum values (constraints), it indicates when we are charging the batteries (positive values) and when we are consuming power from batteries (negative values). Max power charge for each battery is 4,7 kW,

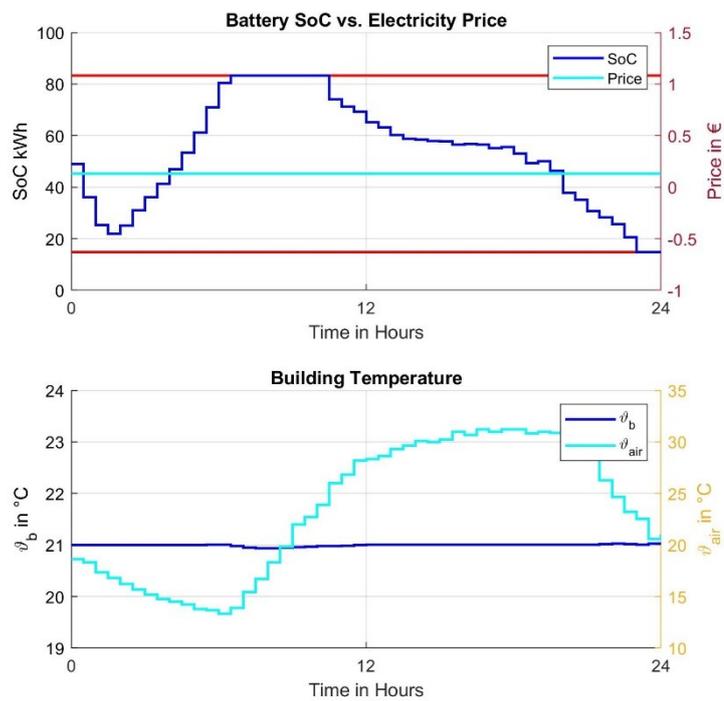
so the constraints are calculate as the number of batteries multiplied by max power charge for each battery.

In the last figure we have the plot of Monetary costs vs. Monetary costs in case of optimization. As we can observe, Pareto optimization drastically reduce Monetary costs while maintaining the building temperature in a deviation less than $\pm 1^{\circ}\text{C}$.

For simulations using prediction of $z35$, $z35_min$ and $z35_max$ we report the results of each one of them and at the end we describe the differences between them. We use the same procedure for simulations using prediction of $True_load$, $True_load_6h_min$ and $True_load_d_max$.

2) Simulation using **prediction data z35 values**

In this case we use as prediction dataset the prediction of z35 values. The target day of the simulation executed is always 2018-07-02, because in prediction dataset of z35 we have also the predictions of this day, so we can observe the differences from using real data of electric power demand. We report the output plots of the simulation.



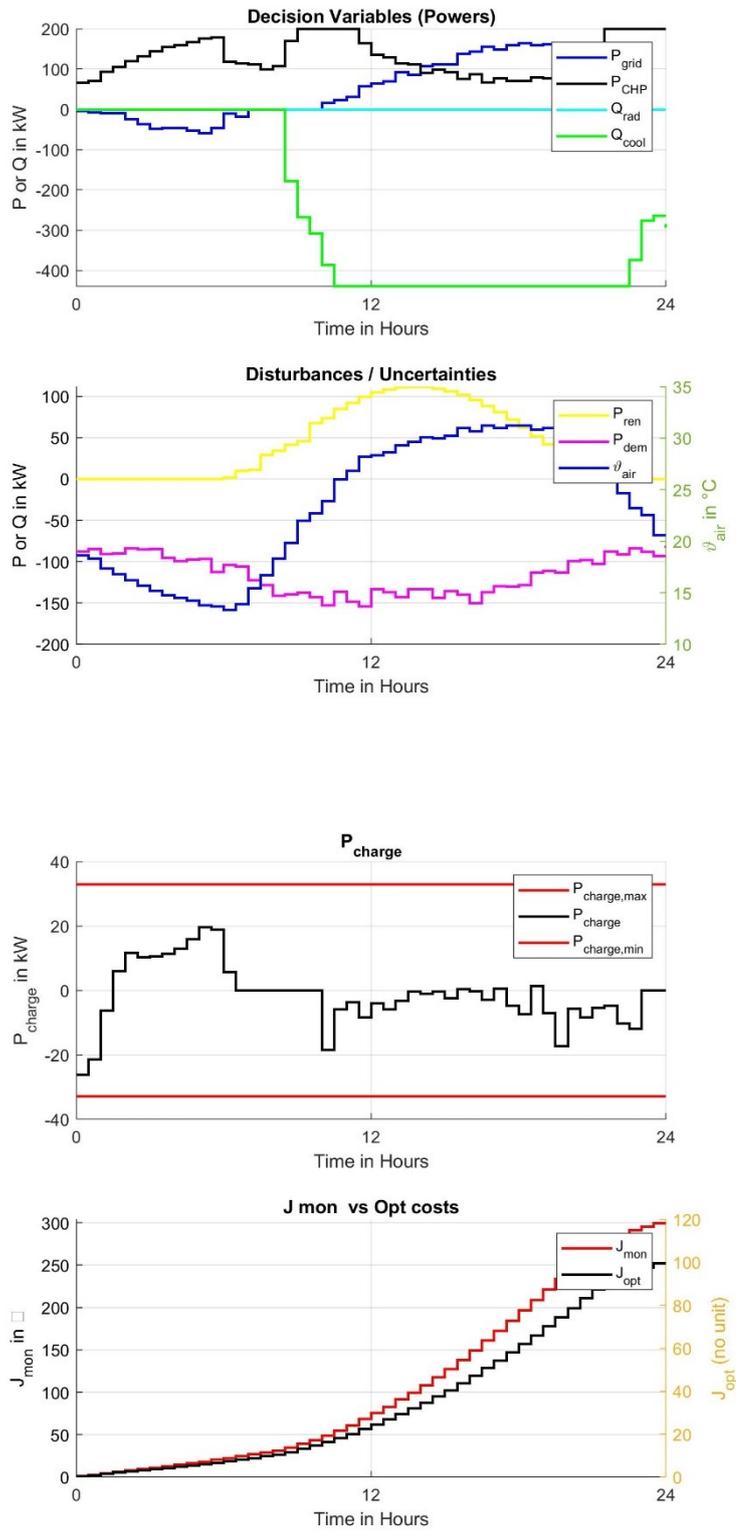
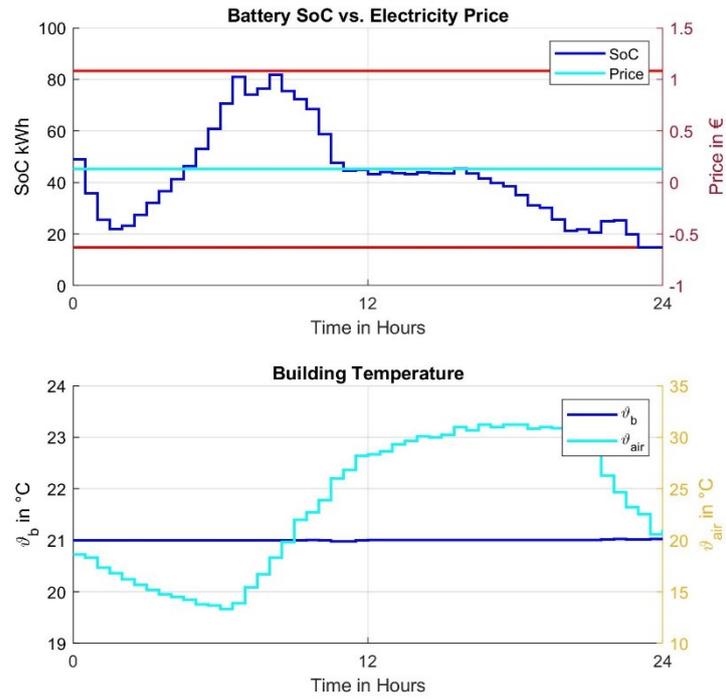


Figure 357. Simulation model result using z35 prediction values

3) Simulation using **prediction data z35_6h_min values**

In this case we use as prediction dataset the prediction of z35 minimal values. The target day of the simulation executed is always 2018-07-02.

We report the output plots of the simulation.



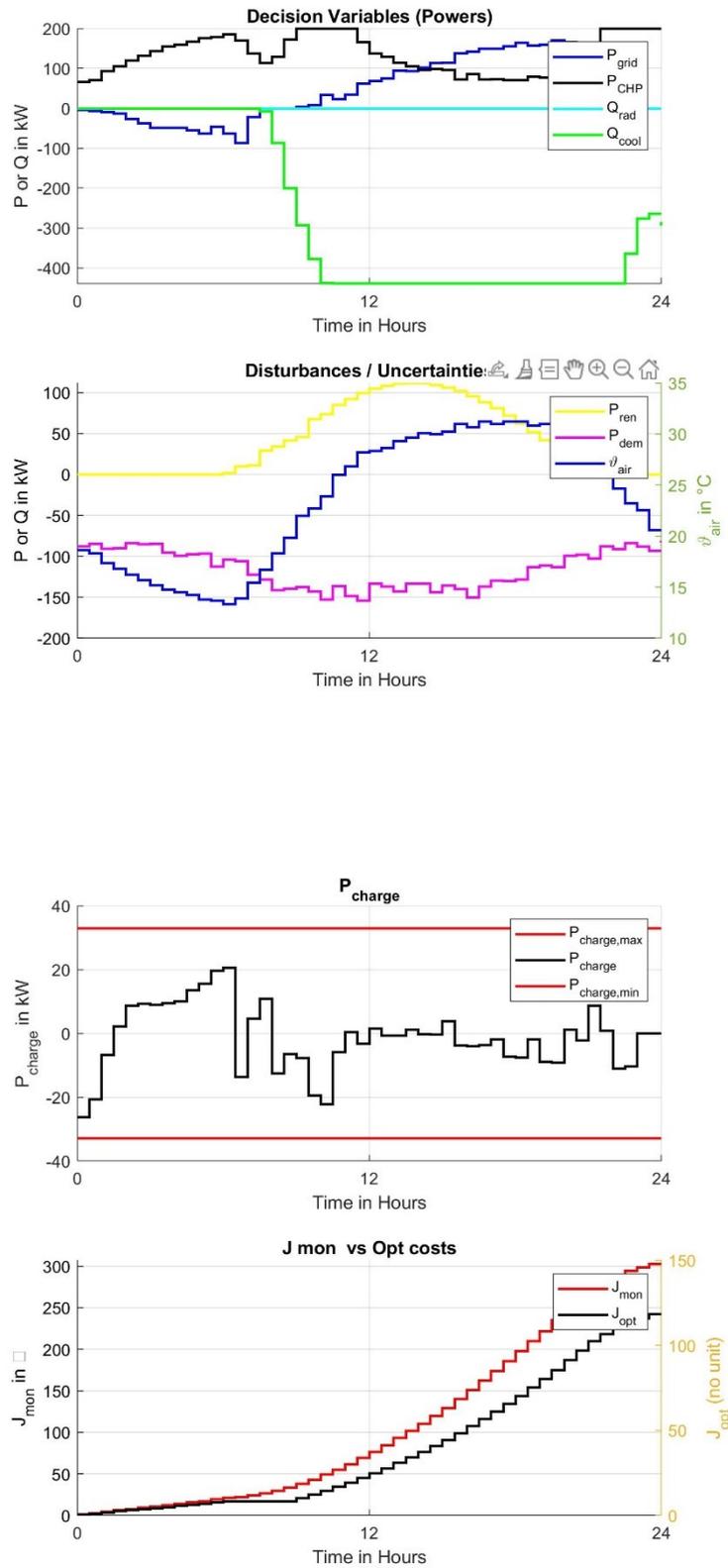
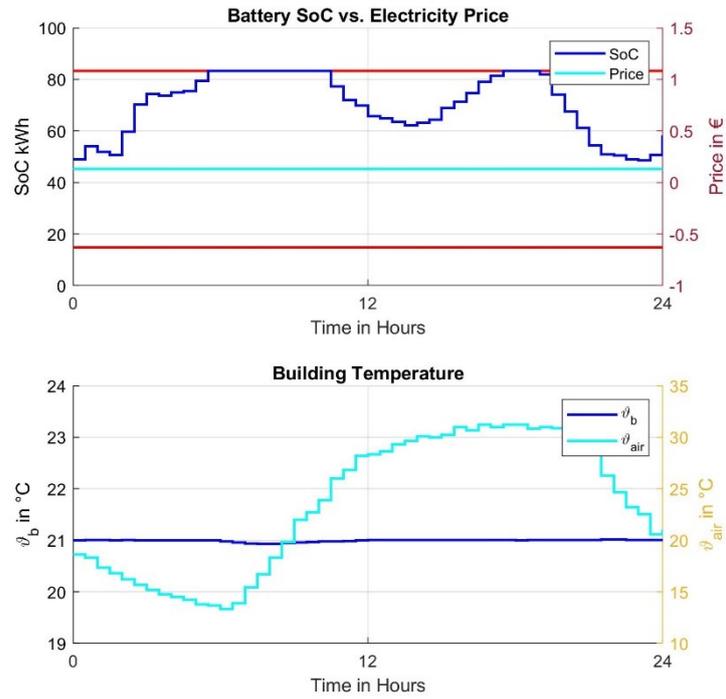


Figure 358. Simulation model result using z35_6h_min prediction values

4) Simulation using **prediction data z35_d_max values**

In this case we use as prediction dataset the prediction of z35 maximal values. The target day of the simulation executed is always 2018-07-02.

We report the output plots of the simulation.



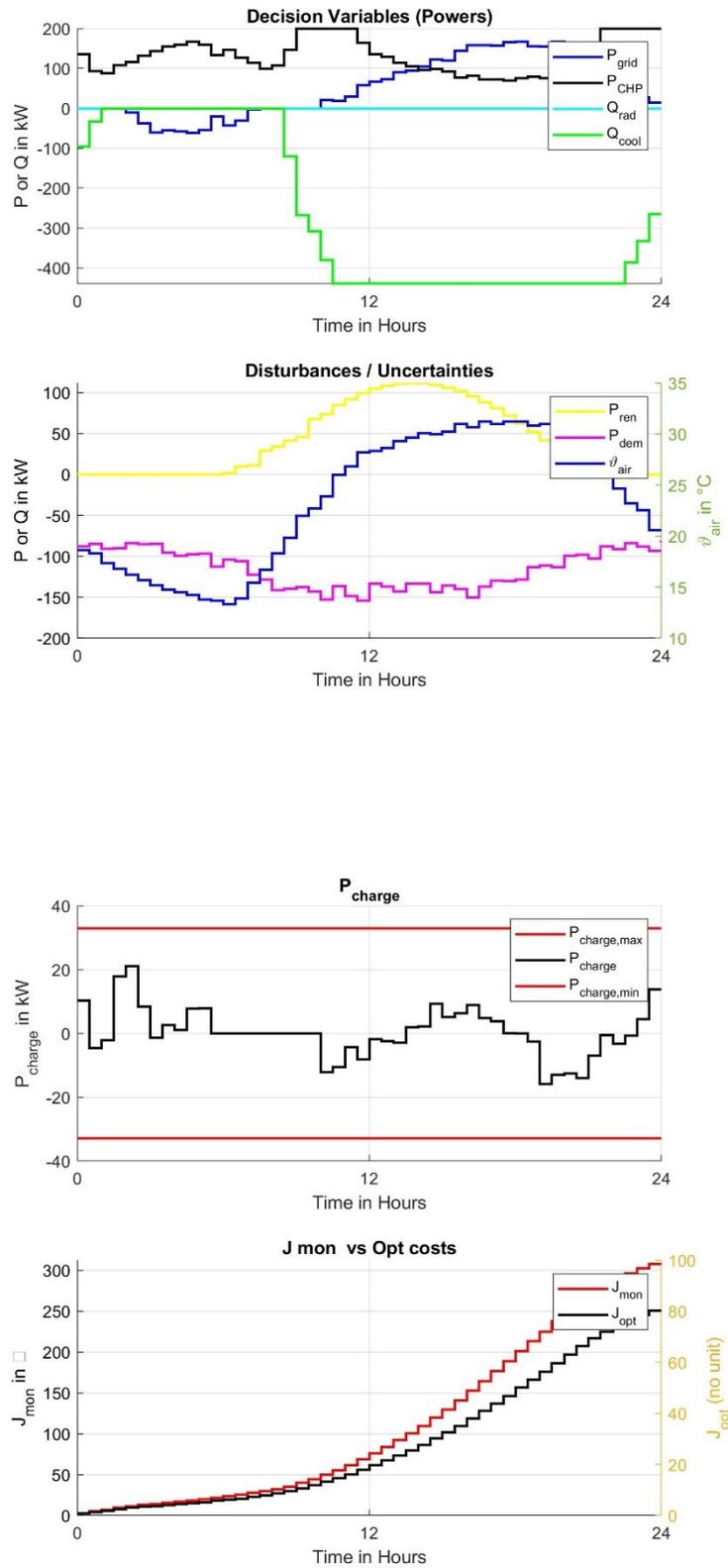
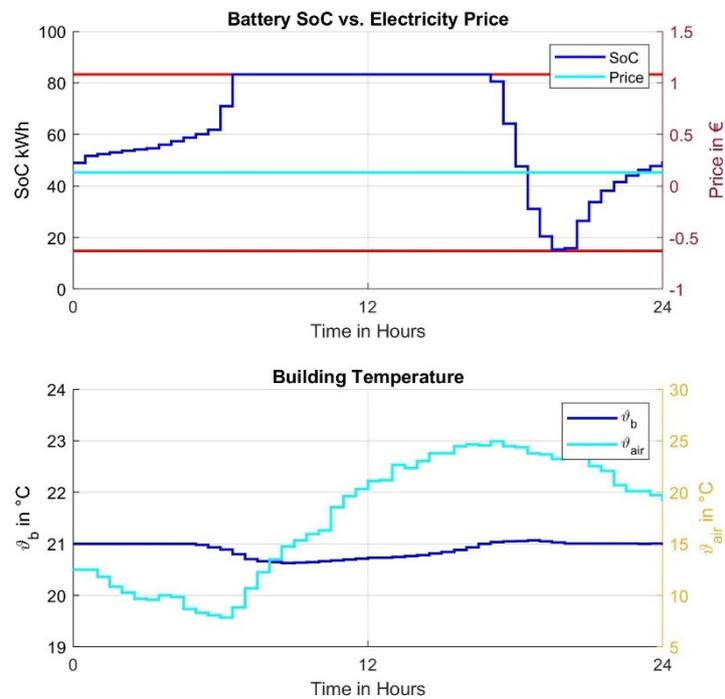


Figure 359. Simulation model result using z35_d_max prediction values

We want to investigate how the prediction data used influence the output of the simulation, so considering the three simulations results we focusing on tradeoff between building temperature and monetaty costs. In all the three cases, the building temperature is approximatevely constant at 21°C for all the day. This is probably due to the fact that the power demand considered is z35 and not True_load, as we observed during generalization of best models the range of values of True_load dataset is upper than z35 dataset, so there are no such problems for the system to maintain the comfort temperature, we can notice this also observing the different range of values of Monetary costs not optimized in the simulation 1 where we use real data. We can observe in each simulations that, when the air temperature goes up, for maintaining the comfort temperature we have an increment of the power consumed for cooling; we can also notice that the power demand is equal in the three simulations because it represent the real value of z35 electric power demand. In all the three simulations, Pareto optimization reduce Monetary cost function while maintaining the building temperature at the comfort value. What is important to notice is that if we use prediction of z35_max value in the simulation we obtain the best tradeoff between Temperature Deviation and Monetary costs, instead using predictions of z35_min value provide the worst tradeoff values.

5) Simulation using **prediction data True_load values**

In this case we use as prediction dataset the prediction of True_load values. The target day of the simulation executed is 2019-06-09. In this case we can't use the same target day of z35 simulations because the real True_load dataset has values for the month of June 2019 and the prediction dataset used have values from 2019-06-08 to 2019-06-30. We report the output plots of the simulation.



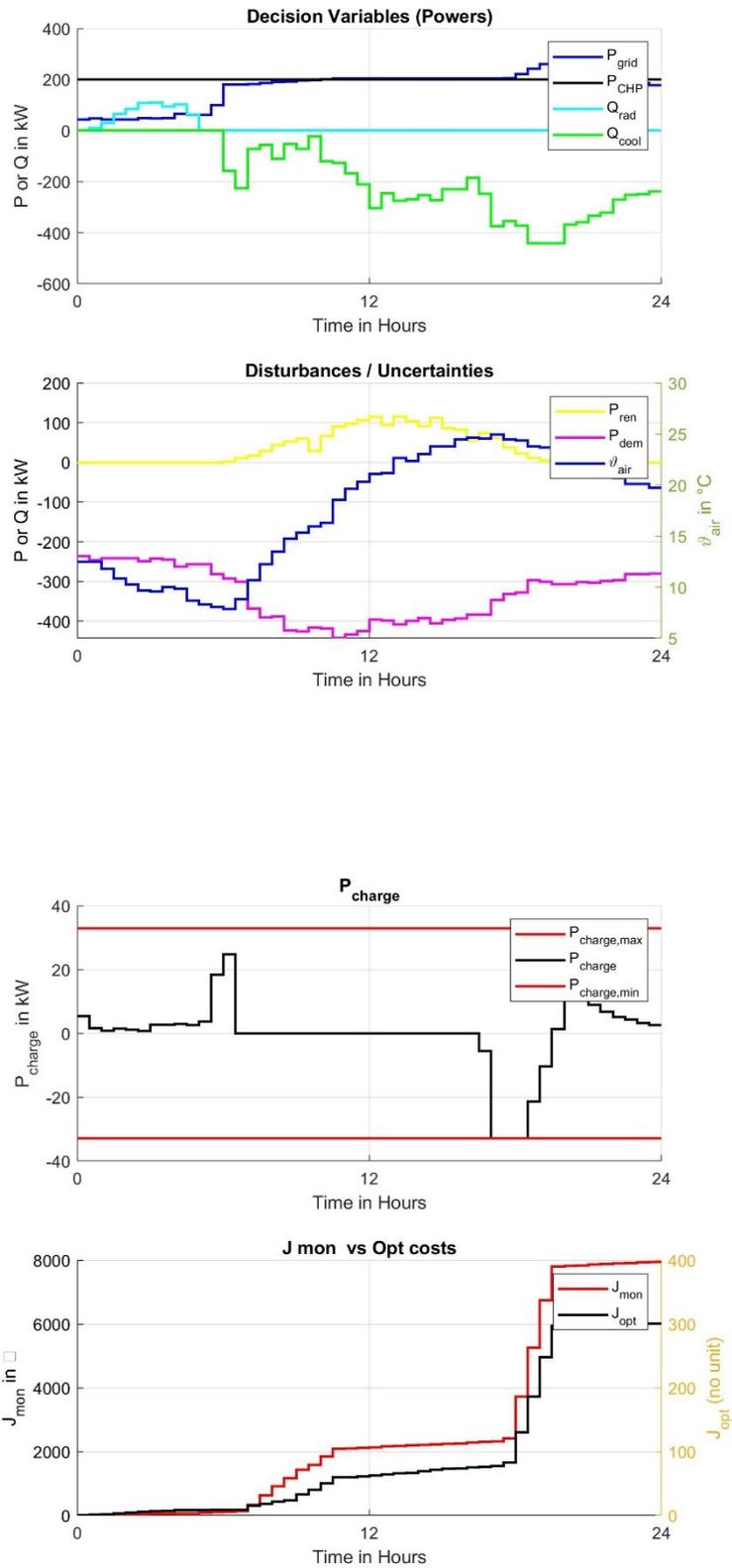
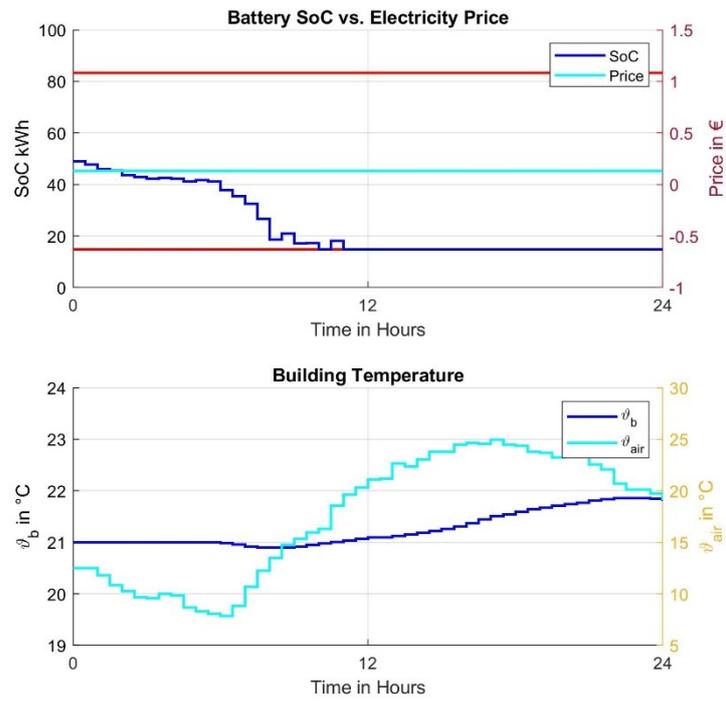


Figure 360. Simulation model result using True_load prediction values

6) Simulation using **prediction data True_load_6h_min values**

In this case we use as prediction dataset the prediction of True_load minimal values. The target day of the simulation executed is 2019-06-09.

We report the output plots of the simulation.



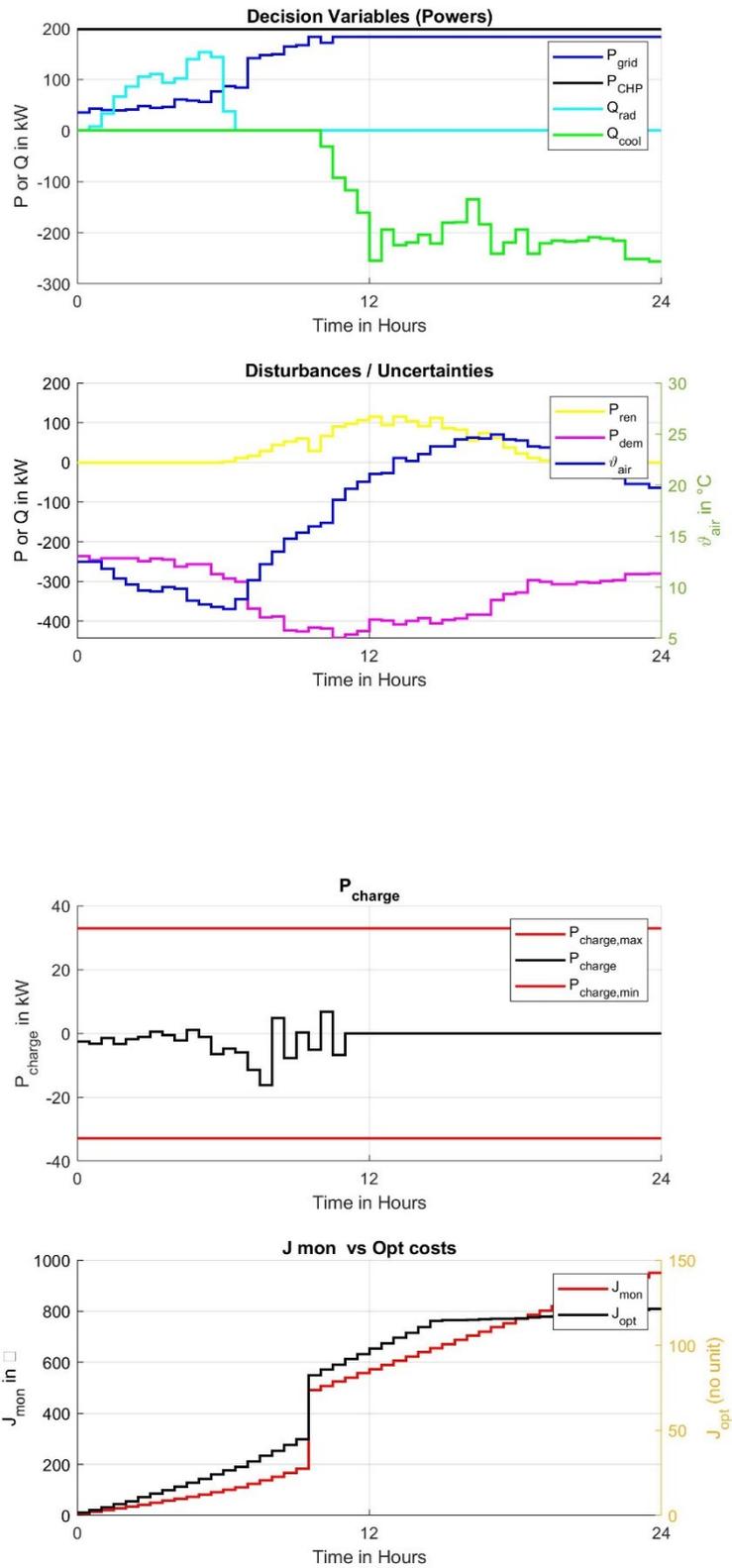
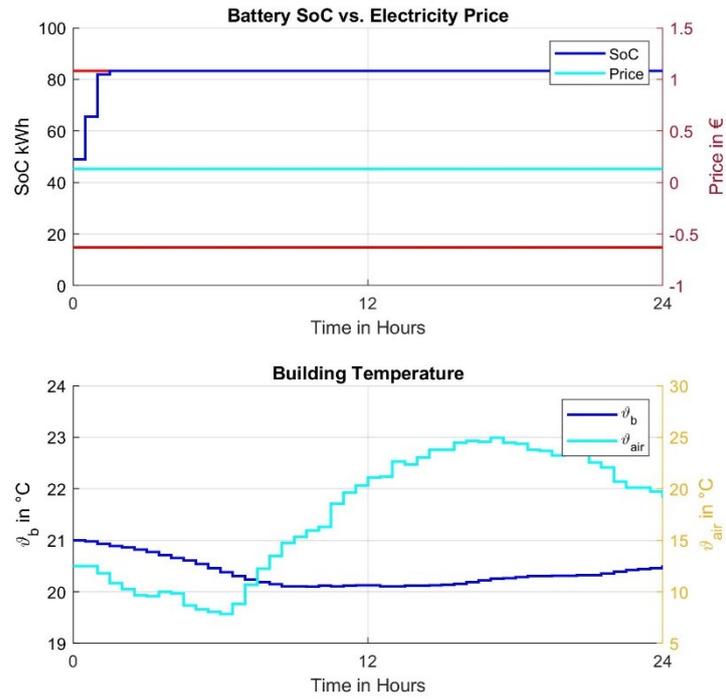


Figure 361. Simulation model result using True_load_6h_min prediction values

7) Simulation using **prediction data True_load_d_max values**

In this case we use as prediction dataset the prediction of True_load maximal values. The target day of the simulation executed is 2019-06-09.

We report the output plots of the simulation.



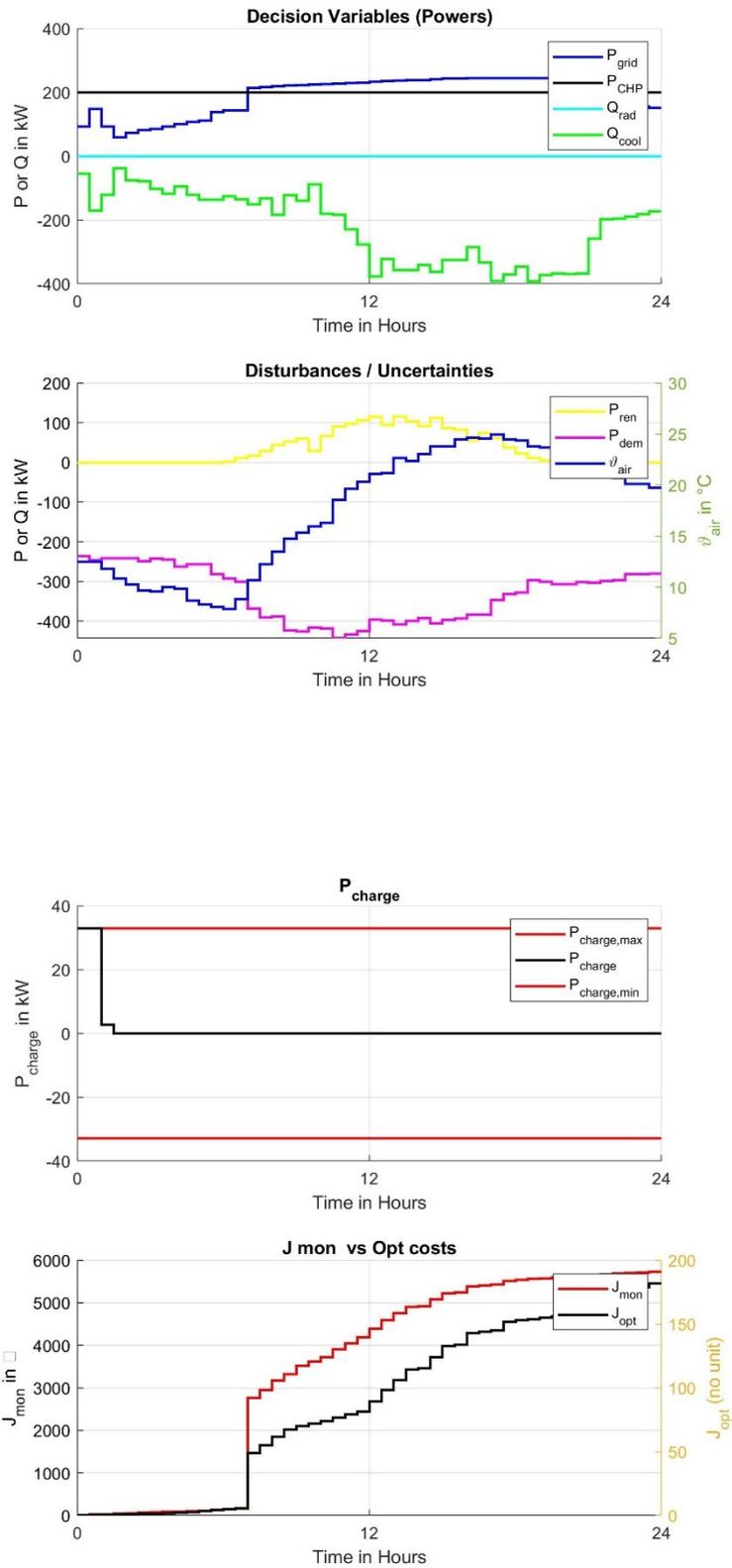


Figure 362. Simulation model result using True_load_d_max prediction values

From figures, we can observe that in simulation 5 using predictions of True_Load values we can maintain the building temperature at the comfort temperature approximatively for all the day, but the optimized Monetary costs are higher than simulations 6 and 7. Instead in simulation 6 and 7 where we use prediction of min and max values of True_Load respectively, the building temperature has a deviation from the target temperature, 1°C higher in simulation 6 and 1°C degree lower in simulation 7, that's because using min value predictions in simulation lead to reduce Monetary costs function but reduce also the power used for cooling, while using max value predictions in simulation lead to use more power for cooling than necessary, and the Monetary costs are higher than simulation 6. Otherwise in all the three simulations, Pareto optimization reduce Monetary cost function.

From the various simulations runned with prediction data, we want to focus on the results of the simulations using prediction of True_Load values, True_Load_6h_min values and True_Load_d_max values because the simulation model is designed to run with these data. The simulations run using z35 predictions are adjusted to see the results of the simulations, but they are using other dataset related to True_Load and not on z35.

We can conclude that the case of using prediction of True_Load max values in simulation (so overestimate Electricity Power Load) is the worst scenario because we can't maintain building temperature at the comfort value and Monetary costs are higher than in simulation 6 where we use prediction of True_Load min values. In simulation 6 we have the lowest value of Monetary costs but also here we can't maintain the building temperature for all the day. In simulation 5 where we use prediction of True_Load values we could have the best scenario because we can maintain the building temperature at comfort value approximatively for

all the day, but the Monetary costs in this case are very higher than in the other two simulation.

At the end, all depends on what is our objective, because if we absolutely want maintain the comfort temperature for all the day ignoring how much it costs in Monetary terms, the best choice is the scenario in simulation 5. Instead if a Temperature Deviation of 1°C is acceptable, the best choice is the scenario in simulation 6, where we have the minimum of Monetary costs function.

Obviously, what is very important to make this results significant is that the predictions used in simulations are enough accurate.

CONCLUSION

We can conclude that the objectives of the research project are achieved. We have studied different families of regression models for electric load time series forecasting, from Recurrent Neural Network to Linear and Ensemble models. For each kind of regression models, we have found the best models for predicting electric load and related features like standard deviation, variance, minimal and maximal values in order to have a confidence bound where the predictive values of the electrical load probably fall. In the searching of the best models, the focus was: on the analysis of the time series target and the correlation between other time series that can improve the prediction performance of the models; on the technique of scaling used also with the objective to generalize the models on other time series related to electric load power; on optimization of the configurations of the models and the hyperparameters and the use of same error metrics for final comparison. From the results of the best models found, considering RNN, the using of Multivariate models leads to an improvement in the prediction performance for each features of interest, as long as the extra time series used have some sort of correlation with the time series target. Linear models are not affected like RNN from using Univariate or Multivariate models, but are very adapt for fitting time series characterized by a periodicity, like the electric load time series and mostly the time series extracted for finding the confidence bound, which all have weekly seasonality. Ensemble models that uses only Bagging technique with decision tree as base models are not adapt for electric load time series forecasting, different is for models that uses also Boosting technique. At the end we have found as best model for predicting electric load time series the RNN model and for predicting the other features extracted the Linear models. We have also generalized the best models found and showed in which way we can apply them to different electric load time series. Then we applied the prediction dataset obtain from the models realized in the EMS simulation framework where the objective was to find the optimal

tradeoff between building temperature and monetary costs and we have investigated how the different prediction used influenced the output of the simulation.

APPENDIX: RIASSUNTO PROGETTO

L'argomento del progetto di tesi riguarda la predizione di serie temporali relative al consumo di energia elettrica. L'obiettivo è quello di realizzare e testare diversi modelli di regressione per la predizione di serie temporali relative al consumo di energia elettrica dello stabilimento aziendale dell'HRI (Honda Research Institute) e per la predizione di serie temporali relative alla deviazione standard, alla varianza, al valore minimo e al valore massimo di energia elettrica consumata, in maniera tale da ottenere un intervallo di confidenza entro il quale molto probabilmente ricadono le previsioni di consumo di energia elettrica.

I modelli di regressione testati appartengono a tre tipologie di modelli: modelli lineari, modelli ensemble e Recurrent Neural Network (RNN).

Per ogni tipo di modello, si sono ricavati i modelli di predizioni più accurati per ciascuna delle cinque serie temporali di interesse in base al punteggio ottenuto sulle metriche d'errore utilizzate per il test dei modelli. Infine, sempre sulla base delle metriche d'errore, si sono confrontati i migliori modelli precedentemente ricavati per eleggere i modelli globalmente più accurati per la predizione delle serie temporali relative appunto al consumo di energia elettrica, alla sua deviazione standard, alla sua varianza, al suo valore minimo ed al suo valore massimo.

Nell'analisi e nella ricerca dei modelli di previsione più performanti, si sono testati sia modelli di tipo Univariate (che utilizzano i dati di una serie temporale per realizzare le previsioni) che di tipo Multivariate (che utilizzano i dati di più serie temporali per realizzare le previsioni).

La combinazione finale di serie temporali utilizzata nei modelli Multivariate dopo diversi test effettuati ed in base anche alla correlazione osservata tra le grandezze utilizzate è costituita da sei serie temporali riguardanti: il consumo di energia elettrica, il suo valore medio giornaliero, la sua deviazione standard

giornaliera/ogni sei ore, la sua varianza giornaliera/ogni sei ore, il suo valore minimo giornaliero/ogni sei ore, il suo valore massimo giornaliero/ogni sei ore e la codifica dei giorni della settimana.

Queste scelte derivano dal fatto che dall'analisi della serie temporale di consumo di energia elettrica dello stabilimento dell'HRI, essa presenta una periodicità settimanale dei dati di consumo. La frequenza con la quale sono campionati questi valori è di 15 minuti, quindi si hanno 4 campioni ogni ora, 96 campioni al giorno e 672 campioni a settimana. La scelta di utilizzare serie temporali estratte dalla serie temporale del consumo di energia elettrica (quindi la media, la deviazione standard, la varianza, il valore minimo e massimo) calcolate su intervalli di tempo pari a un giorno o di sei ore dipende dall'informazione che in questo modo si può fornire al modello, cioè con quale delle due scelte il modello migliora l'apprendimento.

Per quanto riguarda i modelli lineari, l'approccio utilizzato è stato quello di realizzare e testare prima una serie di modelli per la predizione della serie temporale del consumo di energia elettrica nel caso Univariate, andando ad applicare ai modelli più promettenti un processo di ottimizzazione degli iperparametri in modo da ottenere delle predizioni migliori. I modelli lineari che sono risultati più promettenti sono: il modello Linear Regression, il modello Stochastic Gradient Descent Regressor ed il modello Passive Aggressive Regressor. Successivamente si sono utilizzate le configurazioni ricavate dai modelli, testandole sia nel caso Univariate che Multivariate (con le sei serie temporali precedentemente descritte) per predire le cinque serie temporali di interesse. Per ciascuna serie temporale, i modelli migliori sono risultati essere:

- Consumo di energia elettrica: modello Linear Regression, caso Univariate
- Deviazione standard giornaliera: modello Stochastic Gradient Descent, caso Multivariate

- Varianza ogni sei ore: modello Linear Regression, caso Multivariate
- Valore minimo ogni sei ore: modello Linear Regression, caso Multivariate
- Valore massimo giornaliero: modello Passive Aggressive Regressor, caso Univariate

Un'altra analisi che è stata effettuata riguarda l'influenza nei modelli Multivariate delle serie temporali aggiuntive che vengono utilizzate per realizzare le predizioni. Nei modelli lineari testati si è visto che l'utilizzo di modelli Multivariate può apportare miglioramenti al modello rispetto al caso Univariate, anche se ciò non è sempre vero dato che alcuni dei migliori modelli ricavati sono di tipo Univariate. In generale si è osservato che i modelli hanno buone performance di predizione per quanto riguarda la tipologia di serie temporali considerate, caratterizzate da una periodicità settimanale.

Nei modelli ensemble, l'approccio utilizzato è stato il medesimo utilizzato per i modelli lineari. I modelli ensemble che sono stati testati sono: Random Forest Regressor, Extreme Gradient Boosting Regressor e Bagging Regressor che utilizza come modello base il precedente modello Extreme Gradient Boosting Regressor. Quindi riportiamo direttamente i modelli che sono risultati più accurati nel predire valori futuri delle serie temporali di interesse.

- Consumo di energia elettrica: modello Bagging Regressor, caso Multivariate
- Deviazione standard ogni sei ore: modello Extreme Gradient Boosting Regressor, caso Univariate
- Varianza ogni sei ore: modello Extreme Gradient Boosting Regressor, caso Univariate
- Valore minimo ogni sei ore: modello Bagging Regressor, caso Univariate

- Valore massimo ogni sei ore: modello Bagging Regressor, caso Univariate

In questo caso l'utilizzo di modelli ensemble puramente di tipo Bagging (che allenano in parallelo i modelli base e la predizione finale è data dalla media delle singole predizioni) non hanno predizioni sufficientemente accurate, infatti nessuno tra i modelli migliori è di tipo Random Forest. Per i modelli ensemble l'utilizzo di modelli Multivariate non porta ad un miglioramento delle performance di predizione, con l'unica eccezione del modello che predice il consumo di energia elettrica. Ciò è probabilmente dovuto al tipo di serie temporali che il modello deve predire, infatti le serie estratte hanno valori che si ripetono per l'intervallo di tempo in cui sono state calcolate, invece la serie del consumo di energia elettrico assume valori sempre differenti essendo campionata direttamente con uno strumento di lettura. I modelli più promettenti tra quelli ensemble per le predizioni di serie temporali sono quelli che utilizzano la tecnica Boosting (i modelli base si allenano in maniera sequenziale, cioè la predizione del modello base successivo dipende da quella del modello base precedente).

Per quanto riguarda i modelli RNN, abbiamo utilizzato layers di tipo LSTM, più adatti ad essere allenati su lunghe sequenze temporali, in quanto dotati di una memoria a lungo termine. L'approccio utilizzato è leggermente diverso, poiché abbiamo utilizzato un modello base (quindi utilizziamo sempre la stessa struttura dei layers nel modello, con lo stesso numero di unità per ciascun layer) nel quale rispetto alla configurazione iniziale abbiamo solamente modificato il tipo di ottimizzatore e la funzione costo da minimizzare, senza però operare un'ottimizzazione degli iperparametri ma utilizzando per essi i valori di default. Si è operata questa scelta poiché il modello iniziale utilizzato forniva già buone prestazioni, inoltre si è voluto testare, utilizzando il modello base come riferimento, l'influenza delle serie temporali aggiuntive nel caso Multivariate. I

modelli migliori di tipo RNN ricavati per predire le cinque serie temporali di interesse sono:

- Consumo di energia elettrica: modello RNN, caso Multivariate
- Deviazione standard giornaliera: modello RNN, caso Multivariate
- Varianza ogni sei ore: modello RNN, caso Multivariate
- Valore minimo ogni sei ore: modello RNN, caso Multivariate
- Valore massimo ogni sei ore: modello RNN, caso Multivariate

Possiamo notare che nel caso RNN i modelli migliori ottenuti sono tutti di tipo Multivariate, si è infatti notato un incremento significativo delle performance di predizione utilizzando serie temporali aggiuntive, a patto però che tali serie siano correlate con la serie temporale che il modello deve predire, sia direttamente che inversamente.

Considerando i modelli migliori ricavati per ciascuna delle tre tipologie di modelli testati, i modelli risultati essere globalmente più accurati per predire le cinque serie di interesse sono:

- Consumo di energia elettrica: modello RNN, caso Multivariate
- Deviazione standard giornaliera: modello Stochastic Gradient Descent, caso Multivariate
- Varianza ogni sei ore: modello Linear Regression, caso Multivariate
- Valore minimo ogni sei ore: modello Linear Regression, caso Multivariate
- Valore massimo giornaliero: modello Passive Aggressive Regressor, caso Univariate

Dunque non abbiamo nessun modello ensemble tra quelli migliori, mentre come avevamo osservato precedentemente, i modelli lineari risultano adatti a predire le serie temporali estratte dalla serie del consumo di energia elettrica, anche per il fatto che sono serie periodiche, mentre il modello RNN nel caso Multivariate

mostra le performance migliori nella predizione del consumo di energia elettrica, un ottimo risultato considerando anche il fatto di avere utilizzato un modello base di riferimento, e quindi con margine di miglioramento nel caso di ottimizzazione degli iperparametri.

Successivamente si sono generalizzati i modelli migliori ricavati per la predizione di altre serie temporali sempre relative al consumo di energia elettrica. Queste altre serie temporali mostrano sempre una periodicità settimanale, però sono caratterizzate da un range di valori diverso rispetto alla serie temporale utilizzata per allenare i modelli ed in alcuni casi presentano valori anomali (outliers) che influiscono nelle predizioni dei modelli e vanno quindi individuati e sostituiti, nel nostro caso con il valore mediano. Abbiamo dunque analizzato e mostrato le procedure per applicare i modelli migliori ricavati su altre serie temporali nei vari casi che si possono presentare.

Infine abbiamo utilizzato le previsioni del consumo di energia elettrica, del suo valore minimo e del suo valore massimo ricavate utilizzando i modelli migliori realizzati su di un modello di simulazione di tipo predittivo di un sistema di controllo dell'energia, il cui obiettivo è quello di mantenere la temperatura di comfort all'interno dell'edificio minimizzando i costi, un modello che opera dunque un'ottimizzazione multiobiettivo cercando il giusto tradeoff tra le due funzioni costo. L'output della simulazione ha un orizzonte temporale di un giorno ed in base alle predizioni di energia elettrica consumata ottimizza le due funzioni costo. Quello che noi vogliamo fare è verificare come le predizioni influenzano l'output della simulazione. Considerando le predizioni ricavate utilizzando i modelli migliori generalizzati sulla serie di consumo di energia elettrica utilizzata nel modello di simulazione, si è ricavato che il peggior scenario si ottiene utilizzando le previsioni del valore massimo di energia elettrica, poiché la temperatura viene mantenuta al di sotto della temperatura di comfort con conseguente aumento dei costi dovuti all'eccessivo utilizzo di energia per il raffreddamento dell'aria. Utilizzando le previsioni del valore minimo di energia

elettrica non si riesce a mantenere per tutta la giornata la temperatura di comfort ma si hanno i costi minori. Infine utilizzando le previsioni dei valori di energia elettrica, la temperatura di comfort viene approssimativamente mantenuta per tutta la giornata, però i costi risultano essere più elevati rispetto alle altre due simulazioni. In tutti e tre i casi utilizzando l'ottimizzazione multiobiettivo si ottengono dei costi drasticamente minori rispetto al caso non ottimizzato. Ovviamente lo scenario migliore dipende sempre dalle esigenze del progettista poiché nell'ottimizzazione multiobiettivo bisogna cercare il giusto compromesso tra gli obiettivi da raggiungere, magari privilegiando un'aspetto rispetto all'altro. Quello che comunque è importante per rendere significativi i risultati delle simulazioni è che le previsioni utilizzate siano sufficientemente accurate.

BIBLIOGRAPHY

- [1] - Deep Learning For Time Series Forecasting: The Electric Load Case - Alberto Gasparin, Slobodan Lukovic, Cesare Alippi – 22 July 2019
- [2] - Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow - 2nd Edition - Aurélien Geron – O'REILLY
- [3] - Internship Report of Research on Forecasting Electrical Load Demand of Honda R&D Europe - HRI-EU Report 2018-11 Final - Ryunosuke Yamashita, Advisor: Dr. Steffen Limmer
- [4] - https://git.rwth-aachen.de/ems_work-master - Thomas Schmitt - Technische Universität Darmstadt
- [5] - <https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks>
- [6] - <http://colah.github.io/posts/2015-08-Understanding-LSTMs>
- [7] - <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
- [8] - <https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/>
- [9] - <https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b>
- [10] - <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning>
- [11] - <https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e>
- [12] - <https://medium.com/analytics-vidhya/ensemble-models-bagging-boosting-c33706db0b0b>

[13] - <https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f>

[14] - <https://towardsdatascience.com/from-linear-regression-to-ridge-regression-the-lasso-and-the-elastic-net-4eaecaf5f7e6>

[15] - <https://aaaanchakure.medium.com/linear-regression-and-its-mathematical-implementation-29d520a75ede>

[16] - https://www.tensorflow.org/tutorials/structured_data/time_series

[17] - https://scikit-learn.org/stable/modules/linear_model.html

[18] - <https://scikit-learn.org/stable/modules/ensemble.html>

[19] - <https://xgboost.readthedocs.io/en/latest/>

[20] - https://en.wikipedia.org/wiki/Multi-objective_optimization

[21] - https://en.wikipedia.org/wiki/Model_predictive_control