Calibration of sensors

Giacomini Nicolò

January 16, 2024

Abstract

In this paper I describe the procedure to obtain the best results by cheap sensors in order that they are as close as possible to the results of the O_3 in the air measured by the reference stations. The sensors measure the value of the the O_3 in the air but also the temperature and the humidity in the air. All this values can be used in combination in order to get the most accurate value of O_3 in the air.

Studies show that both temperature and humidity affect on the amount of ozone (O_3) is in the air, but they do it differently. High temperatures are correlated to an high value of ozone in the air, and with an high effect. On the other hand, humidity also influences the amount of O_3 in the air, although the impact is not relevant as much as the temperature effect.

In the following experiment I will use some Machine Learning technique in order to calibrate the data. The idea is to take the data of the sensors and try to get the reference station data. I will use two different approaches in this experience. First, I will implement a FNN (Feedforward Neural Network) and then I will use Bayesian multiple linear regression.

The dataset The dataset that I use in this experiment is composed by 1000 data. Each instance in the dataset contains:

- Timestamp of the data composed by day and time. The data are gathered every half an hour from the 21st June, 2017 at 7:00 to the 12th July, 2017 at 4:30.
- Reference station data: data of O3 in the air that I want to predict using the models, they are the reliable data, values measured $\mu gr/m^3$.
- Sensor O3: data of O3 in the air got from the sensors, low performance e less accurate than reference station, values measured in in $k\omega$.
- Temperature: data of temperature measured by the sensors, values measured in in °C.
- Humidity: data of humidity measured by the sensors, values measured in %.

1 Calibration using FNN

1.1 Introduction

The FNN (Feedforward Neural Network) is an artificial neural network. This kind of model is one of the simplest neural network. It is also known as a Multi Layer Perceptron (MPL). A neural network is a collection of connected nodes, called neurons or perceptrons. Every neuron is a scalar function $f_i^{(L)}$ that evaluates each component of the vectorial function $f_i^{(L)}$. Neurons are collected by layers and the results produced by the neurons of a layer are sent to the neurons of the next layer. The procedure until the final layer that can produce the final result. The final result can be a boolean value used for

the classification of the input, or a numerical results elaborated by the linear regression. The latter case is that one used in this experiment.

1.2 The neurons

The neuron functions are composed by the following vectors:

- Weights: control the signal (or the strength of the connection) between two neurons. More the weight is high, more its influence on the output is high.
- Biases: they are additional inputs into the next layer that will always have the value of 1. Bias units are not influenced by the previous layer (they do not have any incoming connections) but they do have outgoing connections with their own weights. The bias unit guarantees that even when all the inputs are zeros there will still be an activation in the neuron.
- Logits: they are the sum of the neurons output of the layer. The logit is a mathematical transformation applied to the output of a neuron before it is passed through the activation function. The formula is the following:

$$z_i^{(k)} = \sum_{j=1}^{Nk-1} \theta_{ji}^{(k)} a_j^{(k-1)} + b_i^{(k)} \tag{1}$$

  where z is the logit computed, k is the layer, θ is the weight matrix, a is the activation, and b is the bias associated to the layer.

- Activation: it is the function of used to activate neurons during the process. The activation function introduces non-linearity into the model, allowing the neural network to learn complex patterns in the data. In my model I will use ReLU activation function. This function is defined as:

$$ReLU(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

1.3 The model

The model used in this experiment is composed by 3 layers:

- Input layer: in this case we have always 3 input neurons, since we get 3 data from the sensors (o3, temperature, humidity).
- Hidden layer: in this case we have different numbers of neurons, based on the test. In this experiment I want to try to evaluate the different effects that I get based on the number of neurons.
- Output layer: in this case we have as a number of input the number of neurons in the input layer, but as an output we will eventually get one single output, that is the value of the regression.

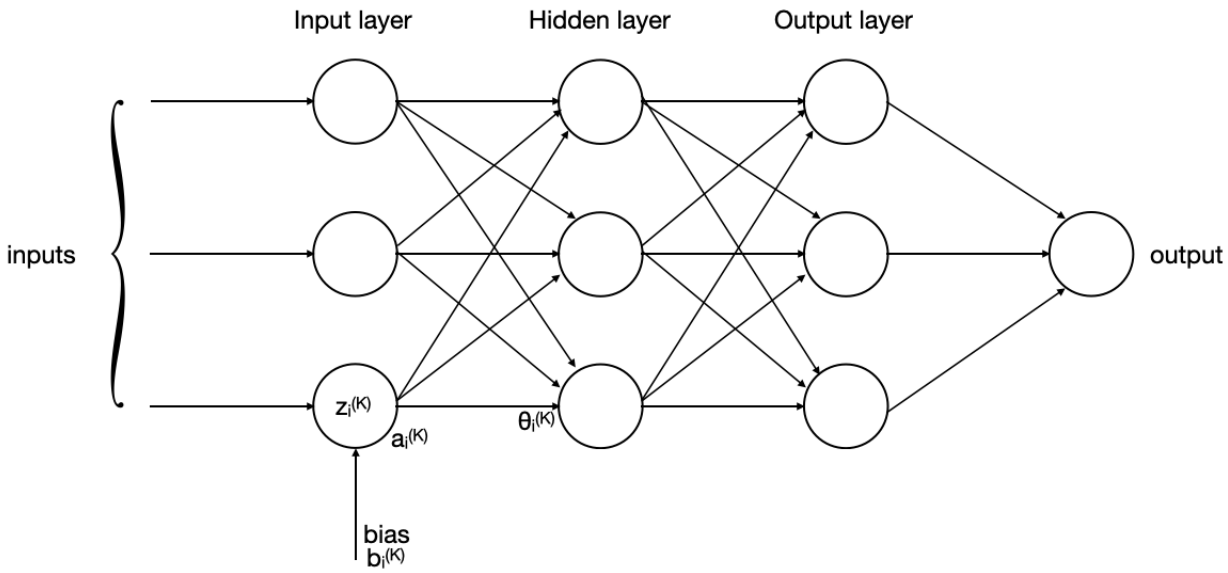

Figure 1: Neural Network model

For the experiment, first, I normalized data in order to use data with mean equal to zero and standard deviation equal to 1. In this way any problem that I get about the different unit of measure in the dataset are avoided.

The model is trained and testing at the end. I shuffled the dataset and I split it into 3 groups:

- Training set: this is the dataset used to training the model, it is the 80% of the total dataset. the model learns the patterns and relationships in the data by adjusting its parameters (weights and biases) based on the input-output pairs in the training set.
- Validation set: the validation set is used to set the model's hyperparameters and to provide an unbiased evaluation of the model fit during training.
- Testing set: this is the dataset used to test the performance of the model after it has been trained and fine-tuned using the training and validation sets and compare the results with the reference station data.

1.4 Experiment

The experiment consists in the following steps:

1. First I build the model structure and I defined the forward method. Then I defined the training function. For the training I used Adam optimizer in order to reach the zero gradient. In fact at every iteration the model follows the Stochastic Gradient Descent method in order to minimize the gradient and achieve the best approximation. The iteration of SGD proceeds as:

$$\Theta^{(t+1)} = \Theta^{(t)} - h\nabla_{\Theta} J(\Theta^{(t)}) \tag{3}$$

   where h is a constant known as the learning rate.

   As a loss function I used MSE loss in order to minimize the Mean Square Error.

2. Once I have the model I start to train it. For the experiment I wanted to try with 2, 3, 4, and 10 neurons for input and hidden layers. At the same time I tried to repeat the iteration for 1,

2, 3, 5, 10, 50, 100, 200, 300, 500, 1000, 2000, 3000, and 5000 epochs in order to check how the RMSE change based on those two parameters. An epoch consists to one complete pass through the training set in the training of a neural network. During each epoch, the model processes the entire training dataset, computes the loss, and updates its parameters (weights and biases) based on the optimization algorithm. At each iteration I evaluate the model with the validation set in order to adjust the parameters.

3. Finally I tested the model using the testing set. I plotted the real value of the testing set compared to the value of the sensor elaborated by the model. In this way I can evaluate the performance of the model.

1.5 Results

Now, I show the results that I obtained by the elaboration of the model. In the following plot I show how the RMSE change based on the number of neurons and the number of epochs.

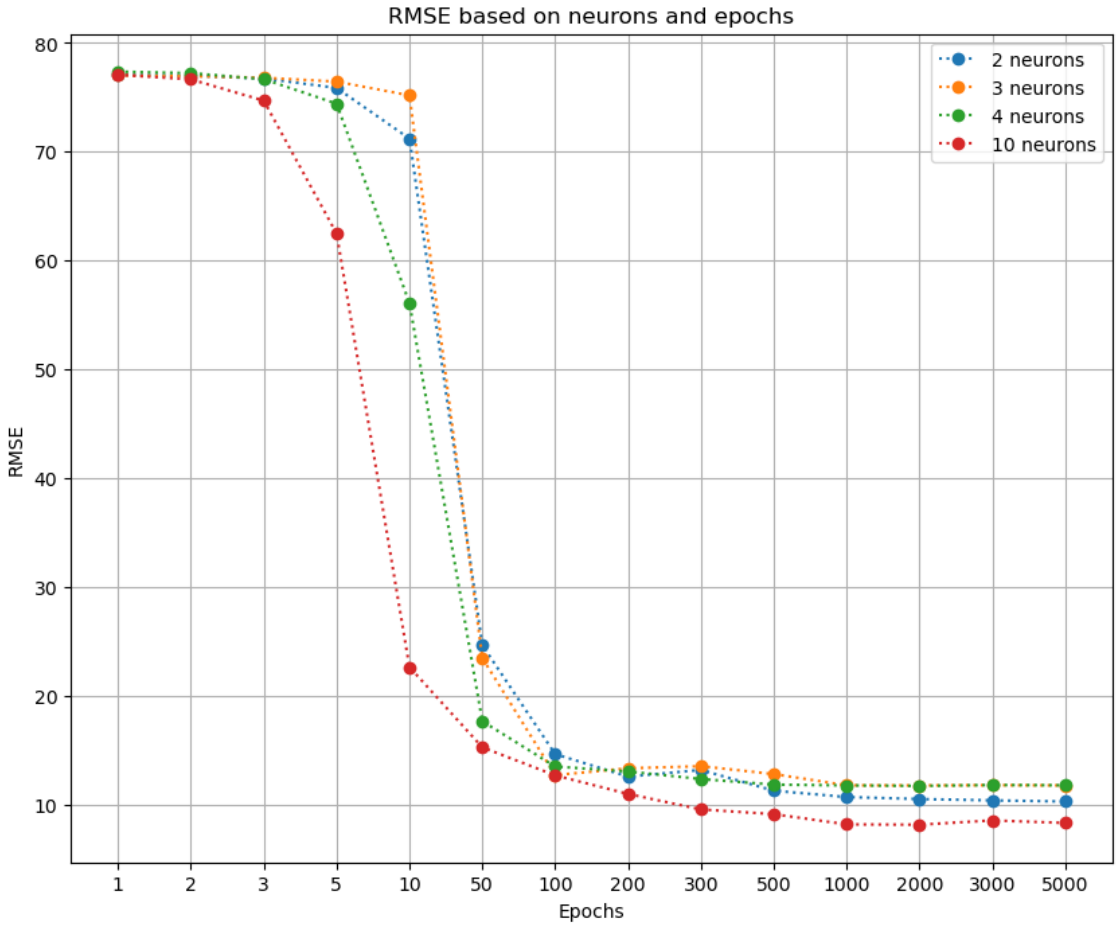

Figure 2: RMSE based on the number of neurons and epochs

The minimum value for the RMSE, in this case is with 10 neurons and with 1000 epochs, the result is RMSE = 8.1201.

By the result I can say that the number of epochs influences the result a lot, indeed more epochs I train the model, more the result is accurate. While the number of neurons, in this case, influences less the performance of the model. Although increasing the number of neurons, I get better results, the

trend of the RMSE is pretty the same during the epochs. Moreover I can claim that after 1000 epochs the results does not change so much and it is possible get very good result stopping to 1000 epochs. In addition I can say that with less neurons we have an underfitting case since with a small number of neurons we have an high RMSE. We can achieve a better RMSE increasing the number of neurons. However, with more complex model, we could find an overfitting case, where the model is too much accurate with original training data and we will interpolate the points in the TS.

Finally I plot the result of the testing. I compared the real value of the testing set with the results produced by the model (the model is the final model, with 10 neurons and after 5000 epochs) This is the plot.

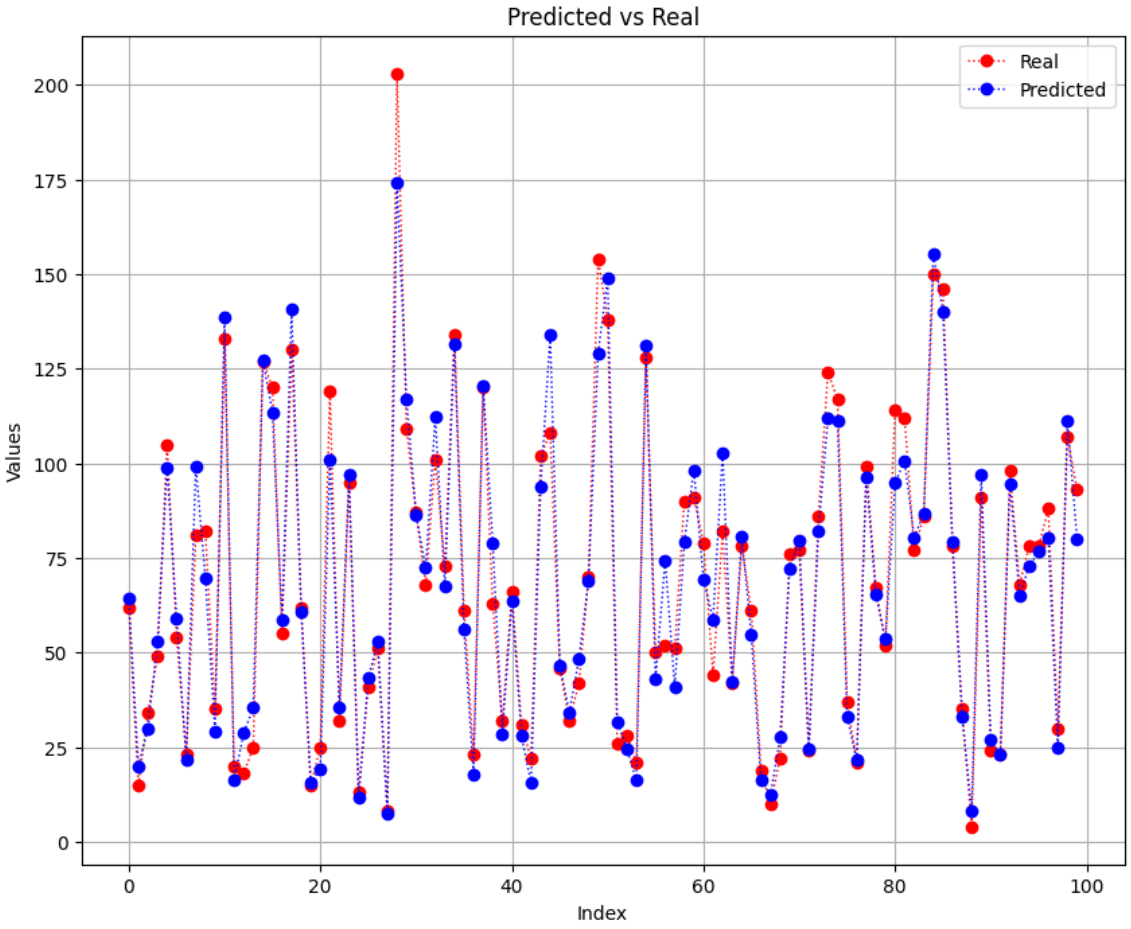

Figure 3: Testing of the model

By the result I can say that the model works very good. I am satisfied about the model since the results are very similar to the reference station data. The calibration worked.

2 Calibration using Bayesian multiple linear regression

2.1 Introduction

In this experience the main goal is to build a Bayesian multiple linear regression model. The dataset that I used is the same that I used in the previous experience. For the training of the model I used the first 943 data (until 10th July).

2.1.1 Bayesian estimation

The basic idea of the Bayesian estimation is to build the likelihood based on the past results. Theoretically it is possible to represent the probability of an event y based on the knowledge that certain events happened before and on the degree of belief on the parameter values. This approach is very different from the Frequentist estimation where the probabilities are fixed and unknown. In the Bayesian statistics, parameters are treated as random variables with probability distributions.

More formally it is possible express the probability of a certain events happen like a conditional probability $p(\theta|y)$ that we call *posterior probability*, a probability distribution. This is the formula:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{4}$$

where:

- $p(\theta)$ the *prior distribution*, i.e. the probability that we know before do the experiment.
- $p(y|\theta)$ is the *likelihood* that event y happens when θ occurs. This is not a probability distribution on θ.
- $p(y)$ is the *evidence*, i.e. the probability of event y. It can be calculated like a sum or integral of the product $p(y|\theta)p(\theta)$, but in many practice cases this formula is not feasible. In those cases to calculate the evidence it is possible use some algorithms like Markov Chain Monte Carlo method. In my model I used this latter method.

In order to build this model I need to define the multi linear regression model. This is as follows:

$$y = \theta_0 + \theta_1 x_{O_3} + \theta_2 x_{temp} + \theta_3 x_{hum} \tag{5}$$

where y is the prediction function, θ_0 is the interception, $theta_1$ corresponds to the coefficient of O3 values, $theta_2$ is the coefficient of temperature values, $theta_3$ is the coefficient of the humidity values.

The main objective of the model is to get the the posterior distribution that allows to describe the data of the reference station dataset.

2.2 Results of the model

After to build the model, I trained it and eventually I obtained the posterior trace. For the training, I performed 4 different chain. This allowed to me to get different way to find the solutions, so the results are more accurate.

In the following plots I show the distribution for each parameter and the plot of the signal:

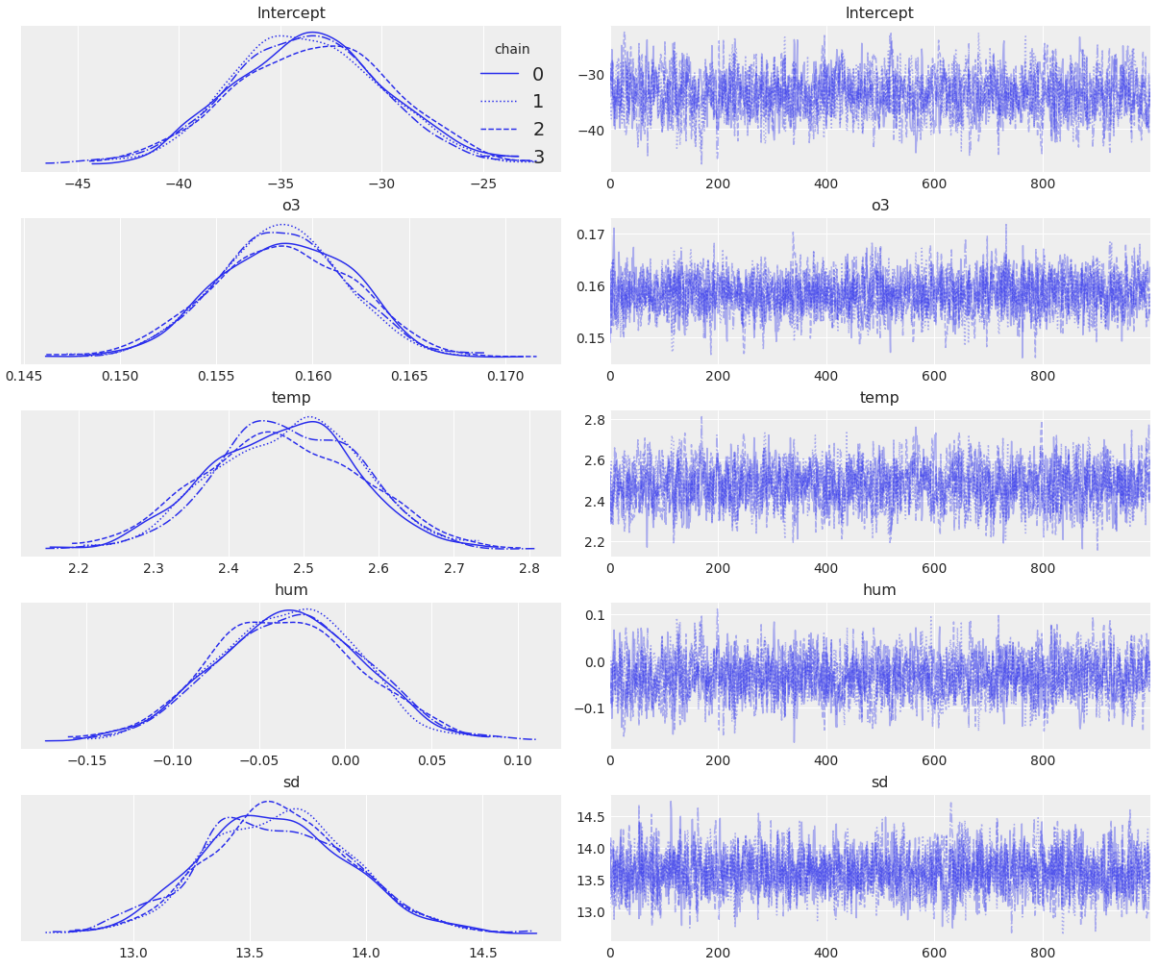

Figure 4: Prediction distribution of each parameter. The intercept corresponds to θ_0, o3 corresponds to θ_1, temp corresponds to θ_1, hum corresponds to θ_2 and sd corresponds to standard deviation σ

It is possible view better the distribution in the following graphs:

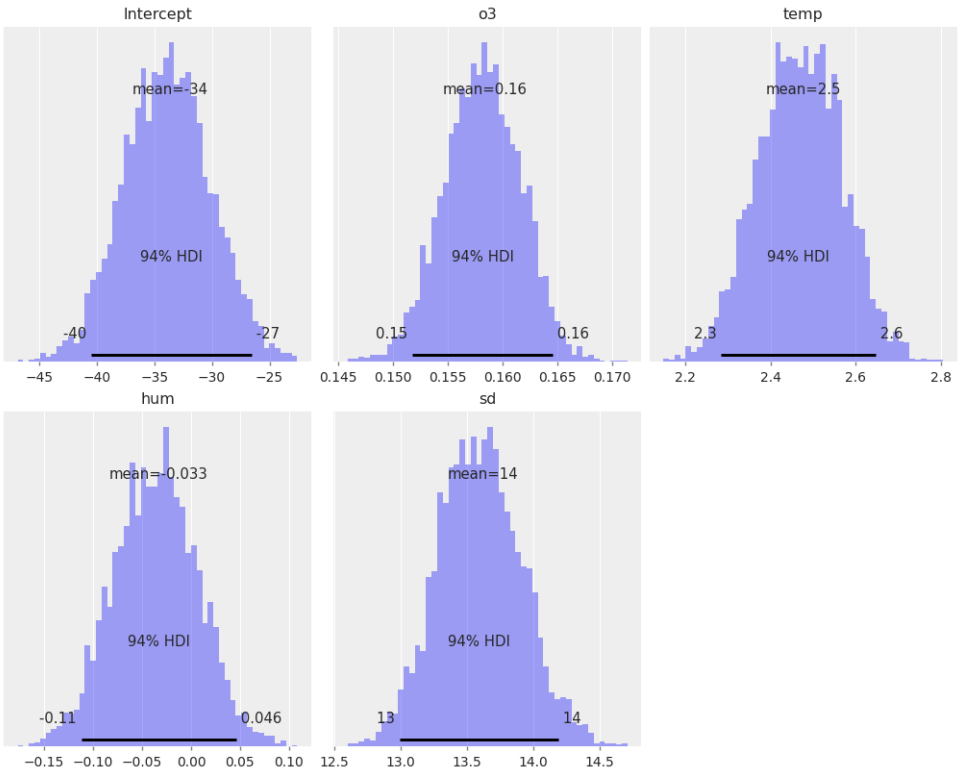

Figure 5: Credibility interval of each parameter

By the histograms I could calculate variance and mean of each parameter:

Parameter	Mean	Variance
θ_0	-34	13.3876
θ_1	0.16	1.15e-05
θ_2	2.5	0.0088
θ_3	-0.033	0.001829
σ	14	0.09204

Table 1: Mean and Variance of each parameter

To notice that the variance of o3 is very low, this means that the o3 parameter (θ_1) keeps constant during the training of the model. I can say also that the O3 sensor values are the most important and reliable values during all the training. This could represent that there is a sort of correlation between the reference station data and the O3 sensor data.

Finally I plot also the posterior distribution compared to the observed distribution, i.e. the function that shows how the reference dataset values are distributed. This is the result:

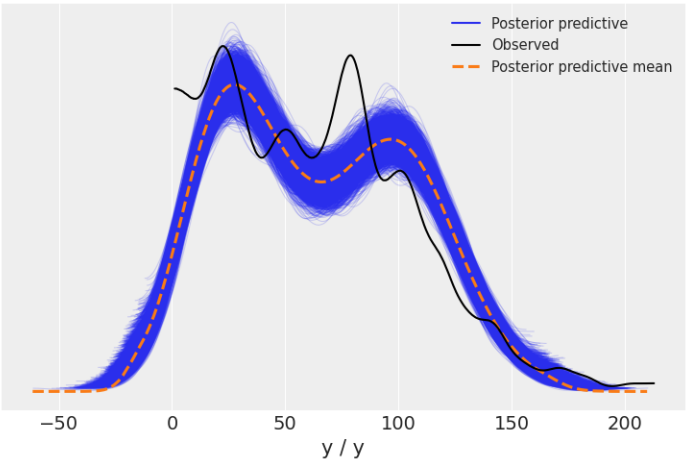

Figure 6: Posterior predictive and observed distribution

By this result it may seem that there is something wrong in the distribution. The fact is that in the model I defined as a family distribution the normal distribution. However the dataset that I had available does not follow a normal distribution. This is the reason why the distribution are quite different.

2.3 Testing of the model

Once to evaluate the model that I built, I tested the model on new values. In fact, at the beginning, I just used some of data to train the model. For testing the model I used the data that start on 11th July, until the end.

First I wanted to test if the model was able to predict a result just with some values. I tested the model passing the sensor data of the 11th July at 11:00. The following histogram shows the result that I obtained:

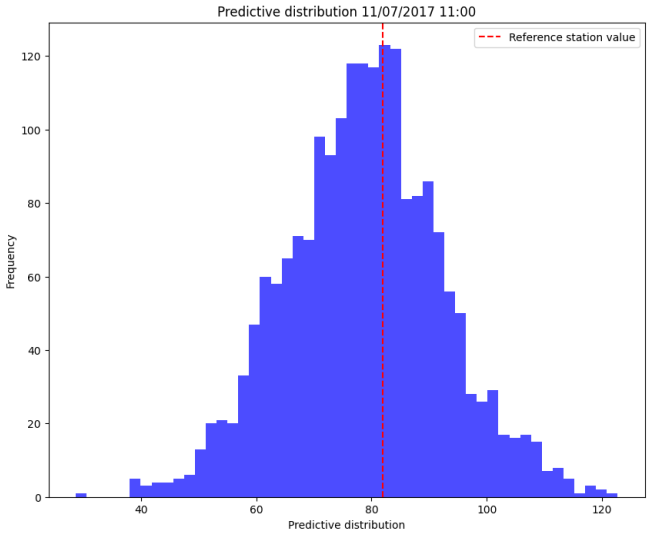

Figure 7: Predictive distribution of 11th July at 11:00

The mean and the variance of this sample are:

Mean	Variance
78.81	185.2746

Table 2: Mean and Variance of sample

The predictive distribution shows that, in this case, the prediction is very accurate, indeed the prediction value is very close to the expected value (mean) of the distribution.

I tried to repeat the same procedure with another value. I took the sensor data of the 11th July at 18:00. The following histogram show the distribution:

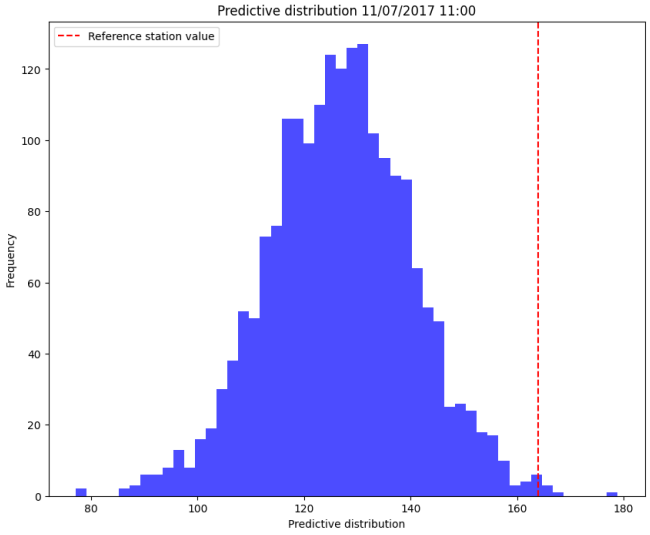

Figure 8: Predictive distribution of 11th July at 18:00

In this case the mean and variance of the sample are:

Mean	Variance
127.39	189.7750

Table 3: Mean and Variance of sample

Unfortunately, in this case the expected value of the distribution is very far from the real value of the reference dataset.

Finally, I plotted the distribution of the remaining dataset and the data of the reference dataset, in order to check how much the model is able to approximate and predict the real values. I plot also the line of the 2 previous day to understand better where they are collocated and how much the real values are far from the expected values.

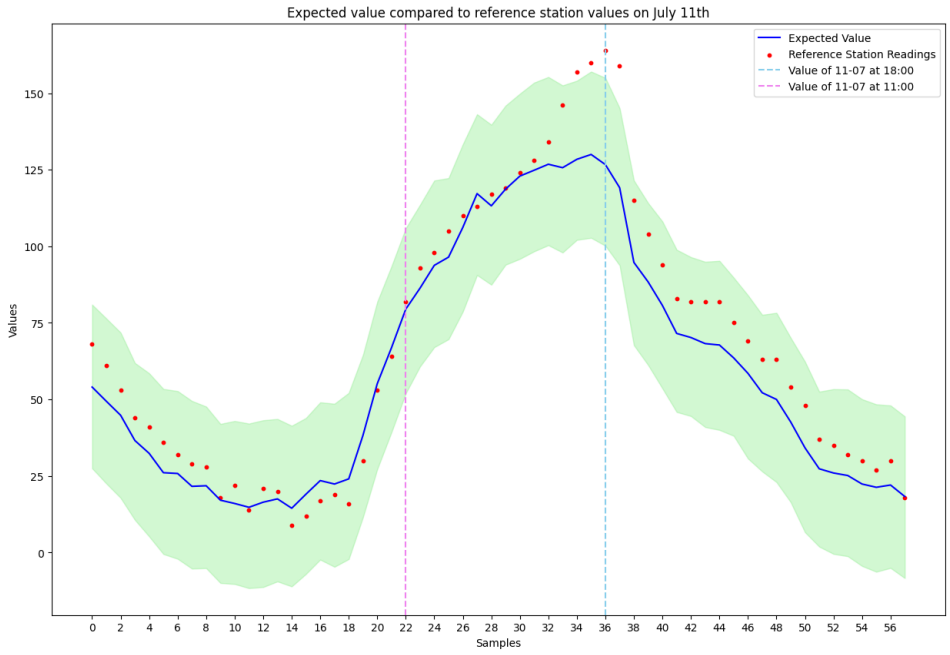

Figure 9: Predictive distribution of 11th July at 18:00

The result is very good, the expected values line is very close to the real values, just around 36 samples the data are outside from the credibility interval. However the distribution of the model is coherent with the real data.

This latter plot is very interesting because it helps to understand better the meaning of the results. The model provides a distribution based on the sensor data. In the graph those distribution are represented by the green area. The expected value identifies the mean of the distribution in a certain point. More the area is wide, more the degree of uncertainty increases. On the other hand, more the area is tight and close to the expected value, more the result is close to the real value.

3 Conclusion

By this experience I could understand two different approaches used in Machine Learning. I found very interesting both case and I think that they are applied in different contexts. In my opinion I found the second approach more suitable for this scenario, because I think that Bayesian linear regression fit better with those case where we have available historic data about a certain event. On the contrary, I think that Neural Network are very useful in those cases where we have several pattern to recognize and to predict. I think that this experiment helped me to learn how to implement different model of Machine Learning and. Overall, I feel satisfied about the experience.