Università degli Studi di Bergamo

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Corso di Laurea in Ingegneria informatica

Time-series forecasting and explainability with LSTM recurrent neural networks: Modeling air quality across Lombardy

Relatore: Alessandro Fassò

Candidati: David Guzman Piedrahita, Marco Vinciguerra

Matricole: 1068605, 1064889

ANNO ACCADEMICO 2021-2022

Abstract

The use of machine and deep learning models has introduced the world to highly performant models. These have fueled breakthroughs in different fields such as the generation and identification of images and text. The study of time series data, which has been mostly dominated by statistical learning and modeling, also stands to benefit, but progress has been slower. In this thesis, LSTM neural networks, commonly used for text classification and generation, are evaluated as a tool for time series forecasting as it applies to an air-quality case study in Lombardy, Italy. Several configurations of the model are presented and analyzed with regards to their performance, complexity and interpretability. The hyperparameter space is explored across different datasets coming from different geographic locations. Performance metrics thereof obtained are compared and weighed against the complexity of their respective configurations. The analysis of the different iterations of a given model also allows to determine which hyperparamaters perform best with respect to different metrics. In spite of the black-box nature of deep learning models, feature importance is evaluated locally through surrogate models, in particular using Lime (local interpretable model-agnostic explanations). The raw data, retrieved from a series of different weather and air-quality stations across Lombardy, was subject to an explorative analysis as it pertains to the discrepancies between the variables measured by different stations and the missing values that they may present. Weather and air-quality stations, often situated in different places, were tied together according to the distances between one another. The thesis begins by introducing the theoretical foundations and methods necessary for the case-study, to then present the different choices made during

the explorative analysis of the data and to present the modeling process and results, along with a study of their interpretability. Lastly, conclusions are drawn and further developments are proposed.

Contents

CHAPTER

1 Problem description

At the time of writing this thesis, there appears a link between the emission of ammonia into the air and the formation of atmospheric particulate matter [17]. Ammonia, along with nitrogen oxides, is a major contributor of the so-called secondary particulate matter in the atmosphere, particularly, of ammonium nitrate and ammonium sulfate, which can make up as much as 50 percent of the total particulate mass.

Ammonia (NH3), is a colorless gas with a very strong odor, it is irritating and toxic. Its main anthropogenic cause is due to management of animal manure (in animal shelters, through the use of slurry storage and spreading) and the use of nitrogen fertilizers. It is also naturally occurring in the air and is essential for the nitrogen cycle .

When dissolved in the blood of human beings, it raises the pH (makes it more basic) and makes it unable to release hemoglobin into the tissues. It also forms gas emboli. To date, there are no regulatory limits for this pollutant.

On the other hand, particulate matter is the set of liquid or solid substances suspended in air of very small dimensions: the ones that have a size up to 2.5 μm in size belong to the PM2.5 category and the ones with a size up to 10 μm fall in the PM10 category.

It has an environmental impact on climate, water and soil contamination and the health of living beings. Because of their small size, particles are able to enter the respiratory system and

alter the proper functioning of the human body.

The objective of this thesis is therefore to exploit the existing and presumed relationship between particulate matter, other pollutants, like NH3 and meteorological variables to evaluate the viability and performance of deep learning models, namely LSTM neural networks, as tools to forecast the concentration of PM10 in Lombardy, Italy.

1.1 Manuscript structure

The thesis is divided into two parts. The first part, Theory and Methods, describes, as the name suggests, the technical foundations upon which the predictive models will be constructed; the second part, Case Study, describes the application of these techniques to air quality and meteorological data gathered in Lombardy, Italy.

Part 1 contains two chapters. Chapter 2 provides an introduction to neural networks in general and to recurrent LSTM neural networks specifically, along with strategies to deal with missing observations, to add explainability to these otherwise opaque models and to optimize them by tuning certain parameters. Chapter 3 contains a brief overview of Random Forest models, which are inherently interpretable and very quick to build, making them the ideal tool to easily generate explanations for different types of model behaviors.

Part 2 contains three chapters. Chapter 4 describes the datasets to be used in the application of the theory described in Chapter 2, along with an exploratory analysis of its aptitude to be used for air-quality modeling. Chapter 5 describes the entire modeling phase of the case study, including the introduction of baseline models to be used as benchmarks, the optimization process of candidate models, and how their behavior can be explained through surrogate linear models. Chapter 6 contains the final conclusions of the case study and describes possible future developments.

Part I

Theory and Methods

CHAPTER

2 Neural networks

All the figures of this chapter are taken from [6], [7] and [8].

2.1 Introduction to neural network

The artificial neuron is a mathematical model that was originally proposed in 1943 in McCulloch and Pitts [20]. Starting in the 1980s, efficient and effective variations of the basic model were published, and around the 1990s these types of techniques became more popular due to the high computing power provided by new computers.

Originally, the proposal was a simplification of the biological neuron model, known as the artificial neuron. It is based on one or more binary inputs and a single binary output. The artificial neuron ”activates the output” when a certain number of inputs are active. The connection, or edge, between two neurons or between an input and a neuron is called a link. Each link is associated with a Boolean value that can be either 0 or 1. With this infrastructure, neurons can perform operations based on the traditdional logic operators AND, OR or NOT.

Figure 2.1 depicts the architecture consisting of a single neuron with a single output and one or more inputs.

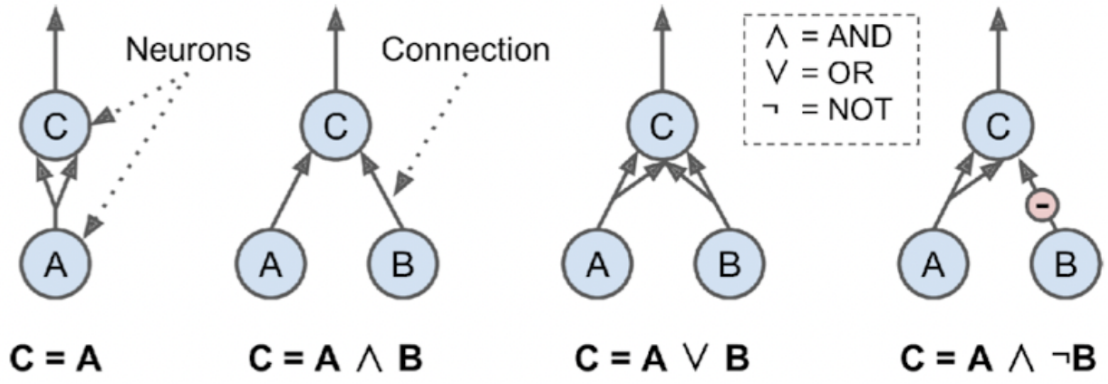

Figure 2.1: Scheme of a classical artificial neuron capable of implementing boolean algebra.

2.2 The perceptron and multilayer perceptron

The perceptron is one of the simplest ANN (artificial neural network) architectures. It was invented in 1957 by Frank Rosenblatt. It is based on a neuron called threshold logic unit (TLU) connected to inputs and with only one output link. The inputs and outputs, instead of binary values as in the classical model, are scalar numbers and each link is associated with a weight. The TLU computes a weighted linear combination of the inputs called z $\left(z = w_1x_1 + w_2x_2 + \ldots + w_nx_n = \mathbf{x^t w}\right)$.

After this input, the step function is applied to the summation represented by z. Formally, this procedure takes place as shown in Eq (2.1) and Figure 2.2.

$$h_w(\mathbf{x}) = step(z), where\, z = \mathbf{x^t w} \tag{2.1}$$

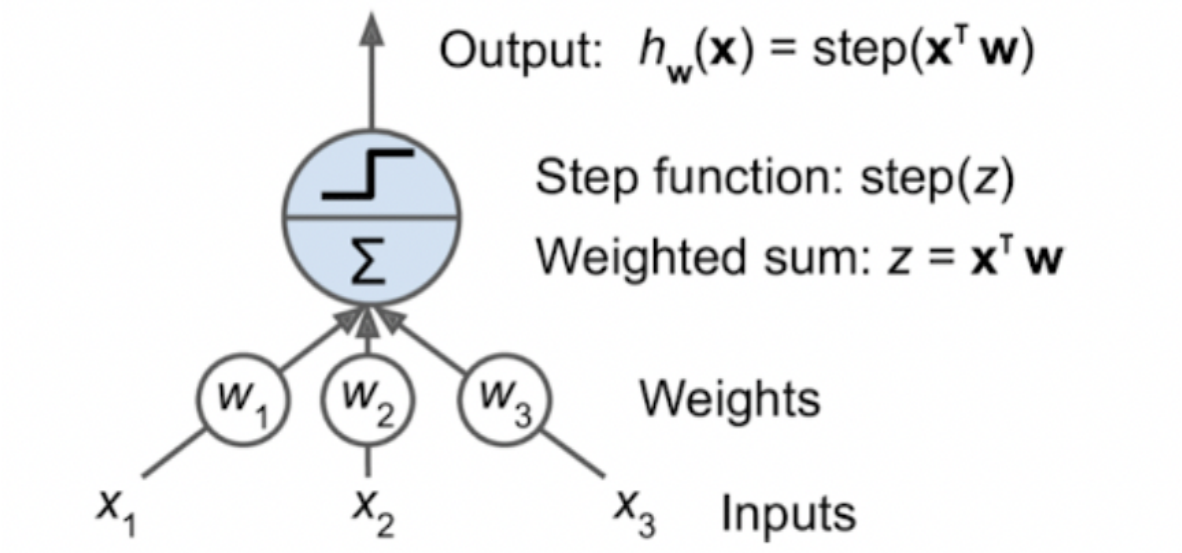

Figure 2.2: Scheme of a Perceptron cell.

The most common step function is the heaviside (or step function). In this case,it is used to determine the activation of the neuron based on the output of the linear combination, as

shown in Eq (2.2):

$$\text{heaviside}\,(z) = \begin{cases} 0 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{cases} \tag{2.2}$$

A single TLU can be used for simple binary classification. If the input exceeds a certain threshold, it is associated with one category, otherwise, if it is lower than said threshold, it is associated with the other category.

A so called perceptron simply consists of a single layer of TLUs, where each TLU is connected to all of the inputs.

It is possible to extend this architecture by stacking more layers, where each output of the neurons of a given layer is connected to each input of the neurons in the next layer.This new architecture is called a Multilayer Perceptron (MLP), where the layers between the input and output layers are called hidden layers. (Figure 2.3).

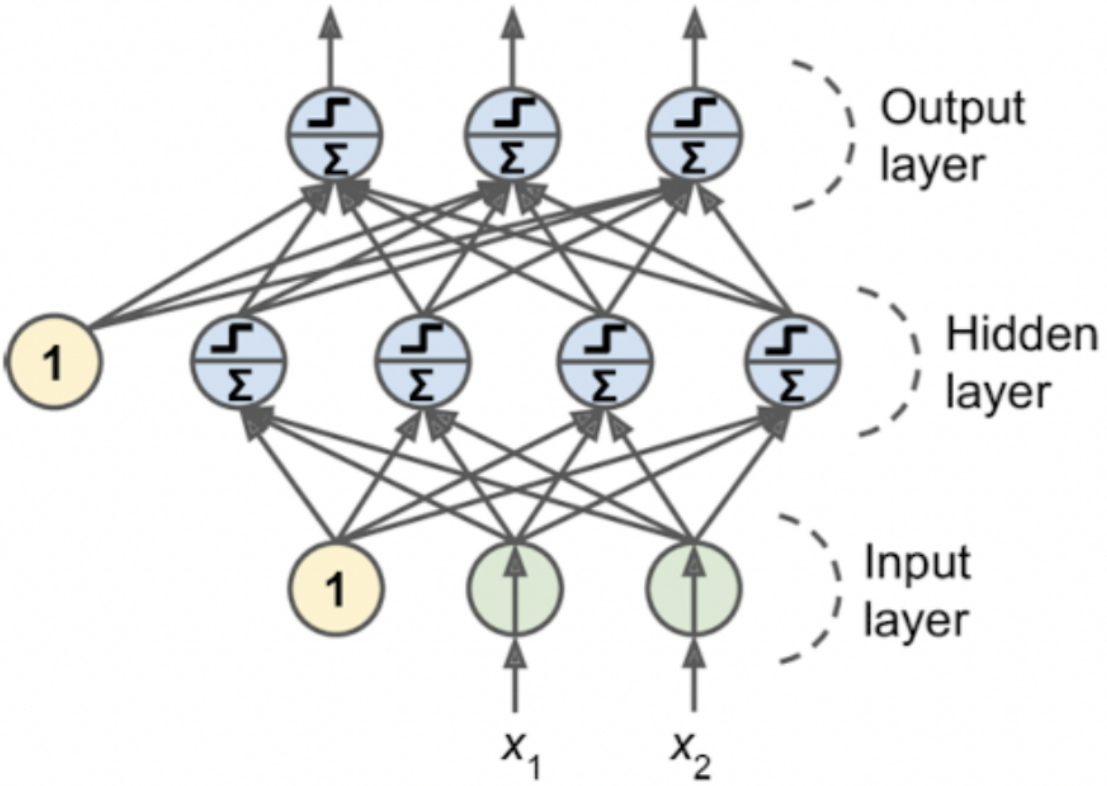

Figure 2.3: Figure 3: multi layer perceptron.

When all neurons in a layer are connected to every neuron in the previous layer, the layer is called a fully connected layer or dense layer. Eq (2.3). represents this type of model.

$$h_{W,b}(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b}) \tag{2.3}$$

In this equation:

- **X** represents the input feature matrix.
- **W** represents the matrix of weights. It contains all the weights except the bias neuron.
- **b** is a vector that contains all the weights between the bias neuron and the artificial neuron (it contains the distortions between the results and reality).
- ϕ is called an activation function. Which is the step function in the case of the Multilayer Perceptron.

Regarding the learning phase, a learning rule that was largely inspired by Hebb's rule, introduced in Morris [22], is used. In the text it is said that, in nature, when a biological neuron is activated, it consequently activates or triggers another neuron, the connection between the two becomes greater. The perceptron is trained with a variant of this concept that takes into account the errors made by the network when making predictions. The learning rule reinforces connections that help reduce error and reduces those that contribute to increased error. Eq (2.4) represents the the learning rule.

$$w_{i,j}^{(nextstep)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \tag{2.4}$$

- $w_{i,j}$ is the connection weight between to given neurons.
- x_i is the value of the input.
- $\hat{y}_j$ is the interpolated value of the output.
- y_j is the observed value of the output.
- η is the learning rate.

2.3 Backpropagation and parameter optimization

Error backpropagation is an algorithm used for training artificial neural networks, which is used as a method for stochastic gradient descent and thus enables error reduction in the learning process. The term backpropagation refers only to the algorithm for computing the gradient, not to describe how the gradient is used.

The learning phase therefore consists in finding the optimal parameters that create the best model, and an iterative approach is used to do this by gradually modifying the parameter values with respect to the previous iteration.

This training mechanism starts with random initialization of the parameters and is carried out using input-output pairs until a minimum of the cost function, which is responsible for calculating the error, is reached. Each iteration is called an epoch. The optimization procedure is as follows:

- The weights and biases of the network are initialized with random values.
- These parameters are iteratively optimized. This phase consists in 4 subphases:
  - The model prediction is calculated using the current parameters.
  - The cost function, that drives the optimization process, is applied to the obtained output with respect to the expected output.
  - The gradient, which will be used to determine how to tweak the parameters, is calculated with respect to the cost function through the process of backpropagation.
  - The parameters are updated.

These four steps can be repeated for an arbitrary number of iterations, and the process is usually stopped when a local minimum is reached.

2.4 Hyperparameters

While the values of a neural network's parameters, namely its weights and biases, are defined by its training process, there's a set of different parameters, called hyperparameters, that are not optimized by the learning process. The hyperparameters of a neural network define its architecture and the characteristics of its learning phase. They act as factors, external to the learning phase, that have an impact on the model's performance. The most important hyperparameters are the following:

- **Batch**

  When Stochastic Gradient Descent (SGD) is used, the parameters are updated after each sample of the dataset is evaluated in the training phase. This approach is considered significantly noisy since the gradient descent direction derived by one sample might differ from the direction derived from other samples, significantly increasing the variance across the different iterations of the learning process.

  Generally speaking, the model gets updated after a specific number of samples has been processed. This is known as the batch size of samples. SGD can therefore be seen as having a batch size of one. In addition to being less noisy, the use of batch sizes greater than one results in reduced training times if paired with the use of GPUs or other specialized hardware. Moreover, the optimization towards a minimum will likely be less erratic than in the case of SGD, increasing the chances of actually converging.

- **Epoch**

  An epoch in machine learning means one complete pass of the training set through the algorithm, that is, its propagation through the network's architecture. Normally an epoch is divided into one or more batches. The number of epochs and batches per epoch are both integers.

  There's no limit to the amount of epochs to use in the training process, which can thus take up an arbitrary amount of time. To stop the algorithm from running, a fixed number of epochs can be used. Alternatively, early-stopping can be used, which consists of stopping the training as soon as the validation error stops showing significant

improvements after a predetermined number of epochs, referred to as patience. As a consequence, the number of epochs can be set to a big integer to give ample room for the training to take place, while relying on early-stopping to stop when appropriate.

- **Number of hidden layers**

  As explained in chapter 2.2, the number of hidden layers is determined by the amount of layers between the input and output layers.

- **Number of neurons per layer**

  The number of neurons in input and output layers is determined by the type of the input and output necessary for the task at hand. For the hidden layers, on the other hand, it used to be common to structure them into a pyramid-shaped, with fewer and fewer neurons at each layer. For instance, a neural network could have 3 hidden layers, the first one with 300 neurons, the second with 200 neurons and the last with 100. This practice, however, is not used anymore because it's been empirically proven that using the same number of neurons in all hidden layers performs just as well in most cases.

- **Learning rate**

  It determines the step size to be used during the optimization process between a training iteration and the next. It can therefore result in the training process stalling, due to a very small learning rate that hinders progress towards a minimum, or it can result in a model that converges too quickly to a local minimum that yields suboptimal results, making it one of the most important hyperparameters.

  One way to find a good learning rate is to train the model for a few hundred iterations starting with a learning rate (for example 10-5) and gradually increasing it up to a large value (for example $10^{\frac{\log\left(10^6\right)}{500}}$). This is done multiplying the learning rate by a constant factor at each iteration (like). As a consequence, the loss should drop more and more quickly when comparing different runs of the training phase that use increasingly bigger learning rates. The ideal learning rate will be one that strikes a balance between the training time and the model's ability to actually converge to a minimum.

2.5 Training test, validation test and testing set

The training, validation and testing set are derived by splitting the dataset in three subsets. The first one (training dataset) is exclusively used in the training phase during which the model's parameters are defined. The validation set instead is used for evaluating a given model's evolving performance during the training phase, by using data that it hasn't used for the training itself.
Lastly, the testing set is used for evaluating the model once it has been completely trained. Technically speaking, the same set could be used for both testing and validation, but it's good practice to use two separate ones.

2.6 Metrics

2.6.1 Coefficient of determination (R2)

The coefficient of determination [25] is the proportion of the variation of the response variable that can be explained through the independent variables of the model. In order to define it mathematically, it is necessary to introduce a few concepts, like the definition residuals, residual sum of squares and the total sum of squares.
The residuals are the difference between the observed data of the target variable and the respective values predicted by the model. Therefore, the residual sum of squares (Eq. (2.5)) is the sum of all residuals in which each residual has been squared.

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2 \tag{2.5}$$

The total sum of squares (Eq. (2.6)) is the sum of the squared differences between the observed data of the target variable and their mean.

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2 \tag{2.6}$$

Knowing these concepts, the coefficient of determinations can be described as in Eq (2.7). Its value ranges between zero and one.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{2.7}$$

2.6.2 Root mean squared error (RMSE)

The root mean squared error [25] is one of the most commonly used metrics for evaluating the quality of predictions. It shows how far predictions fall, relative to observed values, using Euclidean distance. It is a measure of spread out the residuals are, in that it is determined by how concentrated the data is around the line of best fit.
The root mean square error can be calculated using Eq (2.8)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}}, \tag{2.8}$$

In this equation y is the real value and ŷ is the interpolated value.

2.6.3 Mean Absolute Error MAE

The MAE is one of the many metrics that allow for an assessment of the performance of a machine learning model. It's the average of the absolute error as shown in the Eq (2.9).

$$mae = \frac{\sum_{i=1}^{n} abs\left(y_i - \lambda\left(x_i\right)\right)}{n} \tag{2.9}$$

2.7 Recurrent neurons (RNN) and layers

So far the theoretical part of the thesis has been focused on artificial neural networks where the activation link is monodirectional and propagates its values to the next layer, or to the output of the model in the case of the output layer. This type of model is called feedforward neural network.

For the purposes of this thesis, however, a different type of mathematical model called recurrent neural network (RNN) was used. In it, the output of the neuron itself is used as an input for the next time step, thus creating a loop on the node (Figure 2.4). Generally speaking, RNNs use the logistic curve as their activation function, but the hyperbolic function and ReLU can still be used.

It has been found that the simple introduction of this mechanism increases the performance of t neural networks when they are used as regression models, especially when the data is correlated over time.

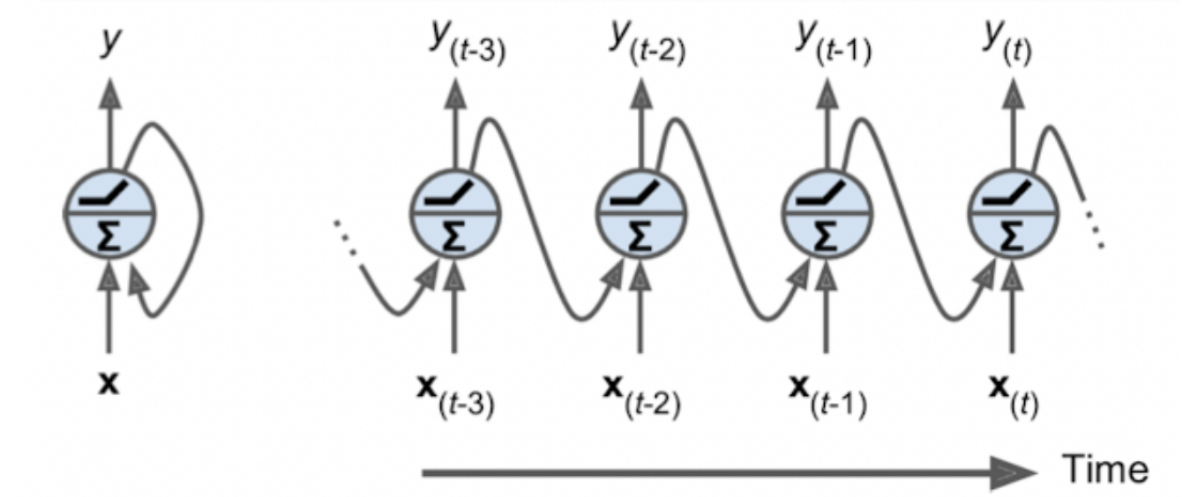

Figure 2.4: Recurrent neurons.

Starting from a single RNN neuron it is possible to easily create one or more layers (Figure 2.5). In more advanced models instead of using the output of a single neuron as an input for the next time step, the entire output of the layer can be used.

Each recurrent neuron has two types of weights: one for the input x(t) and the other for

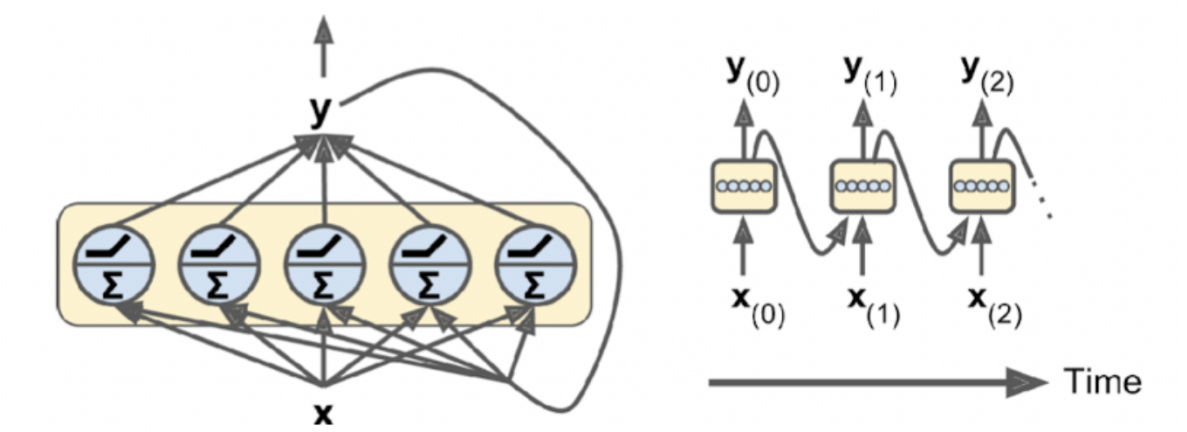

Figure 2.5: RNN architecture with multiple neurons.

the output at the previous step $y_{(t-1)}$. These two types of weights are called wx and wy, respectively. The output vector of the model is shown in Eq. (2.10):

$$\mathbf{y}(t) = \phi(\mathbf{W}_x^T x(t) + \mathbf{W}_y^T \mathbf{y}_{(t-1)} + \mathbf{b}) \tag{2.10}$$

where $\mathbf{W}_x^T$ s the transposed version of matrix $\mathbf{W}_x$.

Jus like with feedforward neural network, it is possible to compute a single-step output of any given layer for an entire batch of values by putting them all in the input matrix $X_{(t)}$ via the Eq. (2.11):

$$\mathbf{Y}_{(t)} = \phi(\mathbf{X}_{(t)}\mathbf{W}_x + \mathbf{Y}_{(t-1)}\mathbf{W}_y + \mathbf{b}) = \phi([\mathbf{X}_{(t)}, \mathbf{Y}_{(t-1)}]\mathbf{W} + \mathbf{b}) \; with \; \mathbf{W} = [\mathbf{W}_x, \mathbf{W}_y]^T \tag{2.11}$$

In this equation:

- $\mathbf{Y}_{(t)}$ is an m x $n_{neurons}$ matrix that contains the output of the layer at a given step t for each instance in the -batch (m is the number of instances, n neurons is the number of neurons).
- $X(t)$ is an m x n inputs matrix containing the inputs for all instances.
- $\mathbf{W}_x$ is a $n_{inputs}\, x\, n_{neurons}$ matrix containing the connection weights for the inputs at a

time t.

- $\mathbf{W}_y$ is a $n_{neurons} x n_{neurons}$ matrix containing the connection weights for the outputs at a previous step.

- **b** is a vector of size n, equal to the number of neurons, containing the bias of each neuron. Regarding the second part of the equation (compact notation):

- **W** is the vertical concatenation of the matrices $\mathbf{W}_x$ and $\mathbf{W}_y$.

- $[\mathbf{X}_{(t)}, \mathbf{Y}_{(t-1)}]$ represents the vertical concatenation of the matrices $\mathbf{X}_{(t)}$ and $\mathbf{Y}_{(t-1)}$.

2.8 Memory Cells and long short memory term network

2.8.1 Introduction to LSTM and limits of RNN

When using an RNN, some information is lost or forgotten at each step. After a certain number of iterations the state of the RNN no longer contains any trace of information from the first inputs. In fact, the neurons' memory typically does not go beyond 10 steps.

It is, however, possible to extend the model in order to add memory beyond 10 steps, which is useful for doing time series forecasting, of both univariate and multivariate in nature.

2.8.2 LSTM cell

The Long Short-Term Memory (LSTM) cell was introduced in 1997 in Hochreiter and Schmidhuber [9]. It offers improved performance and results compared to a other types of cells, such as RNN cells. In fact, LSTM neural networks have shorter convergence times and are also able to keep track of medium to long term dependencies in time series data such as trend and seasonality.

As can be see in figure 2.6, the cell works exactly like a classic cell, except that the state is split into 2 vectors: $\mathbf{h}_{(t)}$ and $\mathbf{c}_{(t)}$, which represent respectively the short term state and the long term state.

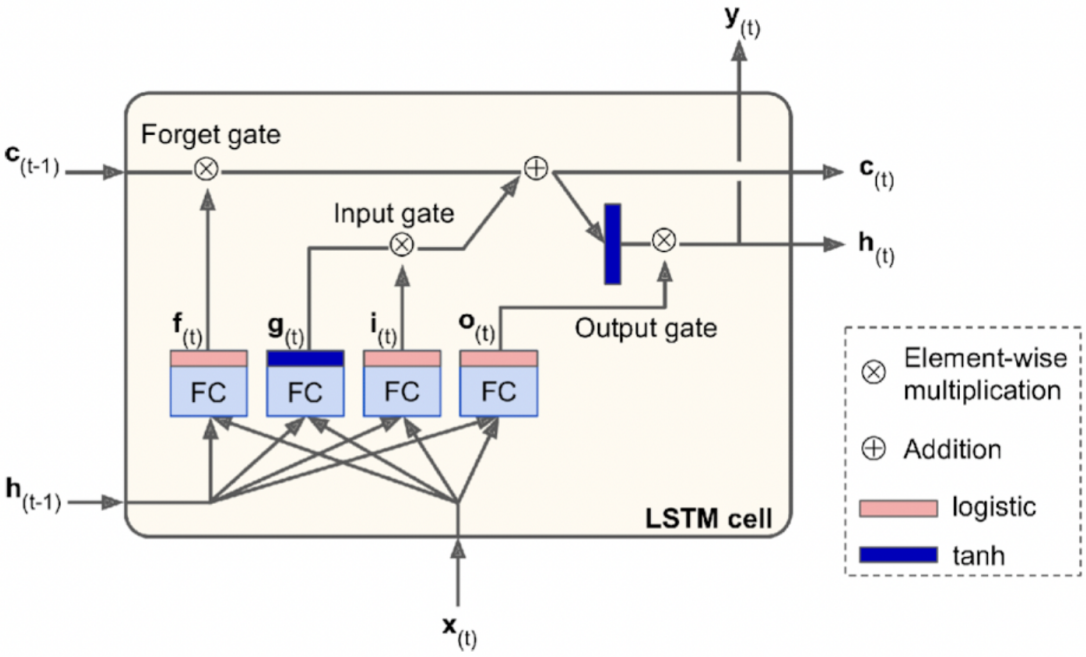

Figure 2.6: LSTM cell.

- The main component is the one that deals with the output $\mathbf{g}_{(t)}$. Its activation function is a hyperbolic tangent. It deals with analyzing the input $\mathbf{g}(t)$ and the previous state $\mathbf{h}_{(t-1)}$. In a basic cell this would be the only component and its output would go directly to $\mathbf{y}_{(t)}$ and $\mathbf{h}_{(t)}$. In contrast, in LSTM cells the output of this cell is processed such that the most important information in the long-term state is kept and the rest of the data that is not useful is dropped.

- The other three components are called gate controllers, because they use the logistic function as the activation function and thus their output range is between 0 and 1. Their roles are as follows:

  - $\mathbf{f}_{(t)}$ is called the forget gate, which defines which part of the long-term memory is to be erased.

  - $\mathbf{i}_{(t)}$ is the input gate, which determines which part of g (t) should be added to the long-term memory.

– $\mathbf{o}_{(t)}$ is called the output gate and controls which part of long-term memory should be read and added to the output, both for $\mathbf{h}_{(t)}$ and $\mathbf{y}_{(t)}$.

Therefore, an LSTM cell can learn to recognize particularly important inputs, add them the long-term state, and store them internally until it is needed. This explains why this model has been more successful in capturing long-term patterns when it comes to time series.

Eq. (2.12) contains the mathematical construction of an LSTM cell.

$$\begin{aligned}
\mathbf{i}_{(t)} &= \sigma\left(\mathbf{W}_{xi}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hi}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_i\right) \\
\mathbf{f}_{(t)} &= \sigma\left(\mathbf{W}_{xf}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hf}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_f\right) \\
\mathbf{o}_{(t)} &= \sigma\left(\mathbf{W}_{xo}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{ho}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_o\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hg}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_g\right) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\left(\mathbf{c}_{(t)}\right)
\end{aligned} \tag{2.12}$$

The meanings of the terms in the equations are as follows:

- $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{x0}$ and $\mathbf{W}_{xg}$ are the matrices that contain the weights for each of the four components with respect to their connection to the input vector $\mathbf{x}_{(t)}$.
- $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{h0}$ and $\mathbf{W}_{hg}$ are the matrices that contain the weights for each of the four components with respect to their connection to the short term memory state $\mathbf{h}_{(t-1)}$.
- $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_0$, and $\mathbf{b}_g$ are the bias values for each of the four components.

2.8.3 Other cells: Dense cells and GRU cells

The Gated Recurrent Unit (GRU) cells were first proposed in 2014 by Cho et al. [3]. GRU cells are a simplified version of LSTM cells, which seem to work more efficiently. The major simplifications are as follows:

- The state vectors (long and short term) are combined into a single vector $\mathbf{h}_{(t)}$.
- A single gate controller $\mathbf{z}_{(t)}$ controls both the forgotten gate and the input gate.
- There is no output gate. Instead, the entire state vector is used as the output.

The schematic of the GRU cell is available in figure. 2.7:

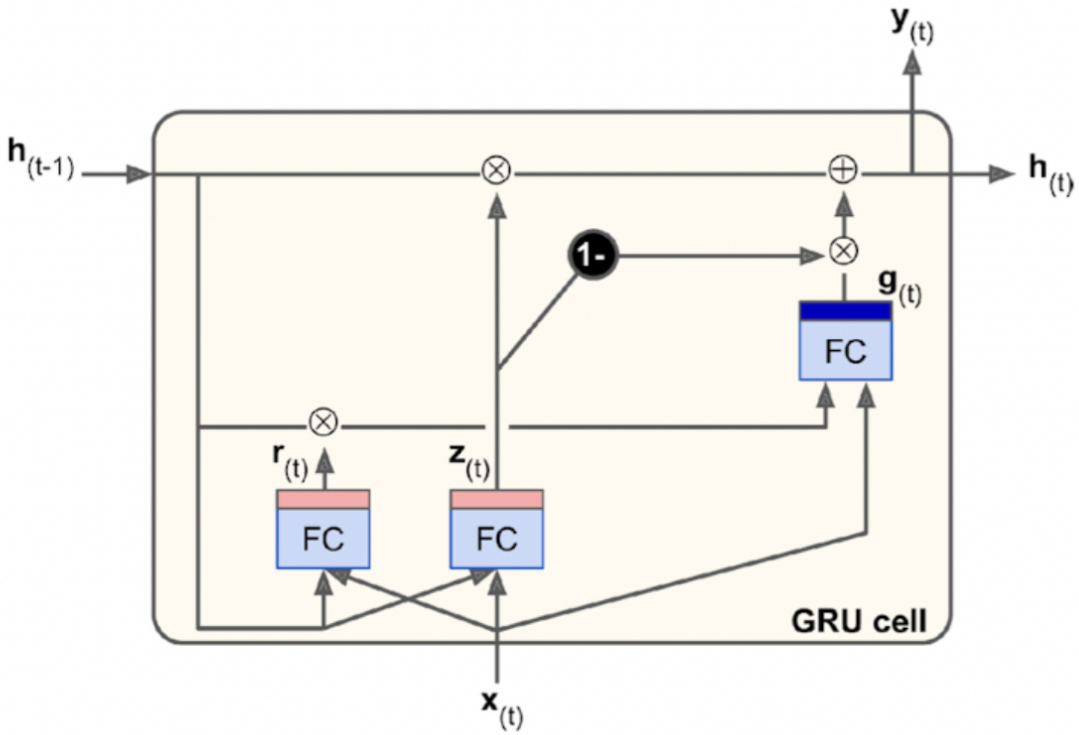

Figure 2.7: GRU cell.

2.9 Dealing with missing values in LSTM neural networks

Traditional recurrent neural networks, including LSTM and GRU models, do not offer any inherent mechanisms to handle and learn from missing data points in any special way. On the other hand, there have been many efforts in the research field to create specialized architectures that aim to consider, and not ignore, missing values and, in doing so these efforts also tend to improve on the quality of the predictions.

2.9.1 Main strategies to address the presence of missing values

As outlined in Chollet [4], there are three main techniques that can be used to extend the baseline capabilities of RNNs in the presence of missing values.

2.9.1.1 Using null values

The first and simplest approach consists of simply replacing these missing values with zeros, but this is only applicable if zero itself is not a meaningful value for the dataset. Needless to say, in the case of air quality and weather phenomena null values hold a different meaning, in that they represent instances in which data was successfully captured but either the substance is determined to be absent or the value of the weather phenomenon is zero in the given scale that it is measured with.

That being said, in cases in which zero is indeed not a relevant value, it is expected that the network will be capable of identifying the role of this special value and therefore it is also expected that it will learn to ignore them whenever identified in out-of-sample data.

2.9.1.2 Taking advantage of normalization bounds

Given that it is often necessary to normalize or scale the data in order to speed up convergence and get values that are less prone to cause exploding gradients, whenever zero is out of the question, missing values can instead be replaced by another value, often an integer with a small absolute value, can be chosen as long as it lies outside the bounds of the scaling that has been applied to the data.

The expected effect is the same as before: through the learning process, the special value is to be interpreted as a datapoint that must be ignored when forecasting. The fact that there is no overlap between it and the scaled data should increase the feasibility of this approach, increasing the chances that the network will learn the role of the value.

2.9.1.3 Masking

The third and most sophisticated solution consists of using masking. This consists of filling missing values with a predefined value often referred to as mask, just like before, this value benefits from being outside of the range of values of the dataset, and is meant to be seen by the model as a representation of a data point to ignore. The main difference in this case, however, is that the model is explicitly notified of the existence of this mask value and knows that they represent missing values from the outset, consequently circumventing the need to actually learn the role of the special value.

This can be easily integrated into previous models, in fact, the Keras implementation of TensorFlow can be extended to support the use of mask values in different types of supported types layers, like the LSTM and Dense layers that are of interest for the case study.

2.10 Exploration of the hyperparameter space

Neural networks are often likened to black boxes, in that, especially due to the presence of hidden layers, their outputs tend not to be explainable. Their seemingly fickle behavior also extends to the modeling and tuning phase: while the choice of the type of neurons or cells to use can often be based on what these were originally built to do, the more specific choice of the so called hyperparameters –batch size, learning rate, number of layers, number of neurons per layers and the like– is commonly made by mere trial and error.

Different combinations of these values, using the same type of cells and the same dataset, can perform in significantly different ways, which makes testing different hyperparameters a worthwhile endeavor. Due to its nature, this phase is often referred to as the exploration of the hyperparameter space or, more concisely, hyperparameter optimization or tuning.

This optimization process can take place following a series of different strategies that may be more or less appropriate depending on the data that they are presented with.

2.10.0.1 Grid search

The most traditional one is called Grid Search [10] which employs an exhaustive search approach of values that come from a previously specified subset. The use of criteria to guide which types of values may yield better results is notably absent, which means that the algorithm trains all possible combinations blindly, so to speak. However, that does not immediately mean that there are no opportunities to speed up the searching process: this this process is actually what is commonly referred to as embarrassingly or delightfully parallel: the effort needed to parallelize the computational load is practically negligible, because the choice of which hyperparameters to use is independent from training processes that came before and that will come after it.

2.10.0.2 Random Search

Even better results can be obtained when applying the Random Search algorithm [1]. This one, in fact, tackles one of the limitations of Grid Search by, as the name suggests, testing a random subset of the possible combinations, instead of systematically training all possible models in spite of their performance.

That means that in situations in which only some among all the possible hyperparameters have a significant impact on the results, this approach tends to yield better results within the same timeframe.

Random search, however, falls short from taking into account an actual choosing criterion for the hyperparameters themselves. Along with Grid Search, these algorithms select the next combination to try either by, in the case of the latter, following a previously specified order, or, in the case of the former, arbitrarily.

2.10.0.3 Optimization based approaches

This issue is addressed by a myriad of other algorithms that use specific strategies to choose the next combination to test based on the performance of the previous ones. Said strategies can be based on the same principles that underlie the forecasting mechanisms of statistical and machine learning models. Therefore, optimization techniques, like **bayesian** [27], **gradient-based** [19] and **evolutionary optimization** [21] are often used for this purpose.

In all such cases, the algorithm is not forced to test combinations upon combinations of hyperparameters that yield mediocre and even worsening results. Instead, substantial amounts of time can be saved by considering values that have the potential to improve on previous results. Just like with any approach, however, these ones are not without their limitations: there is always the possibility of converging to local optima and, while these often present very performant results, it means that these algorithms cannot conclusively find a set of values that is guaranteed to be the best among all.

2.10.0.4 Population based training

It is important to note that parameter optimization is also the core concept that powers the predictive prowess of deep learning models, hence, there have also been efforts to combine the parameters optimization of neural networks, which is tasked with finding appropriate weights and biases, and the optimization of the hyperparameters that define that very process.

Population based training [13] extends the evolutionary approach of evolutionary optimization to train both the hyperparameters and cell weights and biases in a process that, iteratively, tries new alternatives and discards the ones that cannot match better performing fitness functions (which are any given metric used to evaluate improvements). All of this while not having to make any a-priori assumptions.

2.10.0.5 Early-stopping based approaches

The introduction of different optimization algorithms for hyperparameter tend to yield more performant results in the same amount of time, giving credence to the broader claim that using criteria to choose the next combination of hyperparameters to test is a more efficient strategy

than simply going through all the different models either sequentially or at random. However, there are different predictive mechanisms that do not solely consist of finding local optima.

Early stopping algorithms, for instance, use a combination of statistical tests and random searches to determine what to evaluate next and, additionally, they owe their name to the fact that models that present early signs of low performance can be pruned, that is, removed from the pool of promising models. In doing so they spare computation time that would have been otherwise wasted, and can potentially allocate it to combinations that, according to the criteria of choice, seem more promising. Some implementations of this approach are **Successive Halving (SHA)** [12] and **Asynchronous Successive Halving (ASHA)** [14] that prune based on statistical test results and based on performance metrics respectively. One more recent approach (2016), dubbed **Hyperband** [15], combines the previous two approaches, which allows it to cater to a wider array of applications.

2.11 Deep Learning Explainability

As far as mathematical devices go, neural networks are neither straightforward nor exceedingly complex. The core concept that powers them is matrix and vector multiplication, which is a type of operation that has been thoroughly studied and is commonly used in most other fields that use math. In fact, while certain cells, like LSTM and GRU, introduce an additional level of complexity, even these types of widely used models do not need to resort to the bleeding edge of unexplored mathematics.

And yet, despite using fundamental, well-understood notions, their application both to research and commercial applications often relies heavily on heuristic rules and trial and error in order to both train the models and interpret their results.

This leads to a seemingly paradoxical double nature: while the weights and biases of input, output and even hidden layers are readily available for inspection, they offer little to no insight as to why the model may have converged to a specific state, what modifications could lead to better results, or what are the logical steps that make a neural network arrive to a certain conclusion.

While in certain, simpler cases it may be possible to identify which parts of the input data lead to a specific output, the effect of co-dependencies among the features, the non-linearity of the model and the stochastic nature the training process tend to inhibit a clear understanding of said connections in terms of human logic.

Statistical and other machine learning models outside the deep learning realm can clearly present information –in terms of coefficients, importance metrics, statistical significance and the like– that can potentially shine a light on the real-life motivations that tie the target phenomenon with its predictive features. Crucially, this information is often derived directly from the inner workings of the model, as is the case with regression coefficients in linear regression or splitting rules in decision trees or random forests, to name a few.

2.11.1 Surrogate models

If, on the other hand, one is to expect a certain degree of explainability coming from a neural network, more creative solutions need to be applied, that mostly rely on other models that, unlike neural networks themselves, are intrinsically explainable.

The quintessential example of an intrinsically explainable model is simple linear regression. In this case, the beta coefficients, that act as weights in a linear combination, can be used as a metric that reflects the importance of a given feature, especially if said coefficient is deemed to be statistically significant.

In such a model, these coefficients remain the same for any given input. Thus, the resulting feature importance results are valid for the entire model, in this case the model is said to have **global fidelity** [23] More complex models, on the other hand, do not necessarily use the same fixed weights for every prediction: neural networks have a series of connections that can trigger, or not trigger, under different circumstances. These circumstances are shaped by the parameter tuning that takes place during the training process and make the prospect of generating a feature importance list with global fidelity an unlikely one.

For this reason, many explainability strategies for black-box style models aim to provide instead so-called **local fidelity**, in which the validity of a feature importance value is exclusively associated to an individual prediction or output.

One way to achieve this is by using a simpler, usually linear model as a local surrogate for the black-box model. In fact, by assuming that the behavior of the black-box model can be accurately modeled with a linear function in the vicinity of a specific data point, the resulting coefficients can be taken as a humanly comprehensible explanation as to why the overarching model produced the output in question. Moreover, by not making any further assumptions other than the feasibility of using a local linear model as a surrogate, these strategies can be described as **model agnostic**.

Two specific implementations of explainability libraries, **LIME** [23] and **SHAP** [18], make use of this approach to provide a better degree of interpretability.

2.11.1.1 Lime

Lime is a software library that is available for different languages like R and Python. Its name is an acronym that stands for local interpretable model-agnostic explanations and it was proposed in Ribeiro, Singh, and Guestrin [23].

As the name implies, it aims to provide interpretability to a model's prediction, by using localized data and by not making a priori assumptions on the nature of the model itself, ergo model-agnostic.

Much like in the case of SHAP , the first step in achieving these claims is creating a local dataset around the chosen data point which is built from sampling values that surround it the global dataset.

This local dataset, however, isn't built starting directly from the original data, instead a so-called interpretable data representation is created in a series of different ways that depend on the type of data. Text, images and tabular information is supported.

In all cases, this representation is designed to be a version of the original data that is more easily comprehensible both by experts, but especially non-experts in the field. In the case of tabular data, this is achieved by generating a weighted combination of the columns, whereas in the case of text or images, the presence or absence of words or superpixels respectively is used instead.

Having done this, the actual training set for the linear, surrogate model is created by

uniformly and randomly sampling the interpretable representation and, armed with this data, the library tests the newly derived values so that it can gauge the consequent changes in the prediction. Altered features that fail to modify the result are attributed lower importance metrics compared to the ones that succeed.

When it comes to presenting the actual importance metrics, the algorithm has to strike a balance between how well the linear model emulates the behavior of the original model, and how easily interpretable the information can be.

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathscr{L}(f, g, \pi_x) + \Omega(g) \tag{2.13}$$

This is achieved by minimizing (2.13), which is composed of the sum of two terms.

The first one, called **locality-aware loss**, gives an indication as to how well the linear model adapts to the black-box one. It depends on the explanatory model g, a proximity measure that indicates the distance between the data point in question, x, and the perturbed values that are used to train the explanatory model, and the original predictor model f.

The second one, called **measure of model complexity**, indicates how complex the ensuing interpretations are from a human perspective. It therefore depends on the type of intrinsically explainable model of choice: for linear regression it could be the amount of statistically significant coefficients.

Minimizing the first term, therefore, increases local fidelity; minimizing the second one increases the perceived explainability of the model.

2.11.1.2 SHAP

SHAP is another software library that provides functionality similar to LIME's. It is model-agnostic and provides explanations with local fidelity. The novelty that SHAP brings to the table is the use of Shapley values, a game-theory concept that is the key to replacing the use of surrogate linear models, eliminating the possibility of having to compromise local fidelity in exchange for explainability.

Shapley values were originally proposed by mathematician and economist Lloyd Shapley [26]–who later went on to win the Nobel prize For Economic Sciences in 2012 due to this

very same contribution– as an indicator that can be calculated to determine the contribution of each player in a game, especially when these work together in coalitions.

In other words, a metric for the marginal contribution of each player is calculated by perturbing the input of each player and comparing the result to an unperturbed baseline. These marginal contributions are then presented as an average, which is calculated by attributing different weights to the difference between the contribution of a coalition without a given player and the contribution of the same coalition with said player as a part of it.

The fraction represents the weight of the respective marginal contribution, whereas the first term in the subtraction represents the contribution of the coalition with the player in it, and the second term the same contribution without it.

$$\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(N-|S|-1)!}{N!}(v(S \cup \{i\}) - v(S)) \tag{2.14}$$

When applied to machine learning, the players become the features or predictors, and the average marginal contribution measured by the Shapley values is associated with the output of the model, and how each one of these features contributes to it. Coalitions are therefore combinations of features with respect to which the contribution of a given predictor is measured. The implementation of said coalitions for machine learning models represents both an interesting opportunity for explainability, but also one of the biggest disadvantages of Shapley values.

Indeed, this strategy allows for the evaluation of every single predictor against its impact in the context of different groups of other predictors, which shines a line on how its contribution can change as a function of its interaction with other features.

Yet, the amount of subsets of features that need to be taken into account increases exponentially with each additional feature, making the computational cost of Shapley values all too high when no approximations are used.

In addition to the need for approximation to deal with high amounts of features, the use of Shapley values for machine learning requires one additional modification. While the theoretical definition of their calculation requires the exclusion of the feature that is being tested for the baseline prediction, it is not possible to simply exclude it when calculating the

predictions with neural networks or random forests and the like. The solution used in practice consists of sampling the distribution of the feature itself and then averaging its value across different samples. This can be done in different ways that are more or less computationally expensive and that differentiate all the different libraries that use Shapley values.

2.11.1.3 Local fidelity and global explainability

Both Lime and SHAP produce explanations that can be tied to the actual behavior of the black-box model only in the vicinity of the chosen data point. In fact, either by creating a local linear model or by calculating the Shapley values for a certain input, the indicators that are produced only consider the behavior of the original model for the chosen data point, and not its global behavior.

It may be, therefore, tempting to calculate these local explanations for multiple data points, so that all-encompassing conclusions can be drawn from them.

If a subset of inputs is chosen so that the data is representative of the entire domain of input values, the features that come up as having high importance more often than not may be considered to be more relevant to the predictions.

While identifying features that are consistently crucial is, without a doubt, useful to form a better understanding of the model's choices, this must not be confused with the concept of global fidelity.

An explanation that attains global fidelity is one that accurately explains the choices of the model in all circumstances, whereas the above proposal only constitutes a qualitative assessment, and cannot guarantee that a predictor will be more or less relevant than the others in all cases, unless all possible inputs are tested and this is shown to be the case. The domain of possible inputs, however, is, more often than not, infinite, hence true global fidelity is not feasible with these approaches.

CHAPTER

3 Other models

3.1 Random forest

Random forests are a famous type of machine learning model created by Breiman [2] that involves combining the output of multiple decision trees to obtain a single result. It is used for both classification and regression problems.

It is based on collections of decision trees (hence the name forest) that are split recursively in a binary fashion until the leaf is reached by simple comparison.

Each decision tree is constructed by taking data randomly from the source dataset; this step is called bootstrapping.

3.2 Multiple linear regression with ARIMA errors

One of the most commonly used models used for time-series forecasting are the multiple regression models with ARIMA errors [24]. Being statistical models, instead of machine learning models, they can serve as a benchmark to compare the performance of these two different types of models.

Part II

Case study

CHAPTER 4

Exploratory data analysis

4.1 Preliminary analysis and missing observations

The first phase of the project is to search for the air quality monitoring stations in Lombardy that measure NH3, PM10 and PM2.5. They must measure all three simultaneously or at least two of them, since it is not possible to study the relationships between these pollutants if only one of the three is available. The data sampling frequency is daily.

It is also important to take into account the frequency with which the data are not measured, namely the amount of missing observations, as this factor can erode any model's capability of understand the underlying behavior of the variables in question. If there are too many missing values, the dataset may stop being representative of the real world interactions between the pollutants considered.

In total, ARPA Lombardia provides 174 air quality stations and 279 meteorological stations, each one with a unique ID. In order to prioritize stations that might be more likely to continue gathering the same data in the coming years, assuming that stations that have consistently done so in recent years might continue to do so, the period between 2018 and 2020 was analyzed to determine which stations measure all 3 pollutants simultaneously. Of these, among the 26 stations that measure at least one of them, there and only the following 6:

- Cremona via Fatebenefratelli (ID station: 677, urban area)

- Schivenoglia (ID station: 703, rural area)
- Sannazzaro de Burgondi Agip (ID station: 693, urban area)
- Pavia via Folperti (ID station: 642, urban area)
- Milano Pascal Città Studi (ID station: 705, urban area)
- Moggio (ID station: 681, rural area)

The map of Lombardy in figure 10 shows all the stations that measure at least one of the pollutants being considered. The map also shows which stations measure which combinations of pollutants. The possible combinations are:

- Only NH_3
- Only PM10
- Only PM2.5
- PM10 and NH_3
- NH_3, PM10 and PM2.5 missing at the same moment

As can be seen in figure 4.1, the stations are not evenly distributed over the territory: in some cases the stations tend to be agglomerated together, and in others they remain isolated. For our case study, we are interested in the stations that simultaneously measure the 3 variables of interest.

Among the stations that measure all three pollutants, the ones with the least amount of missing observations were identified in order to create the most accurate mathematical model possible.

Additionally, weather variables that directly impact the distribution and interaction between the pollutants, can contribute to the predictive performance. For this reason they are also subject to analysis. The chosen weather variables are: the wind speed and direction, temperature and rainfall.

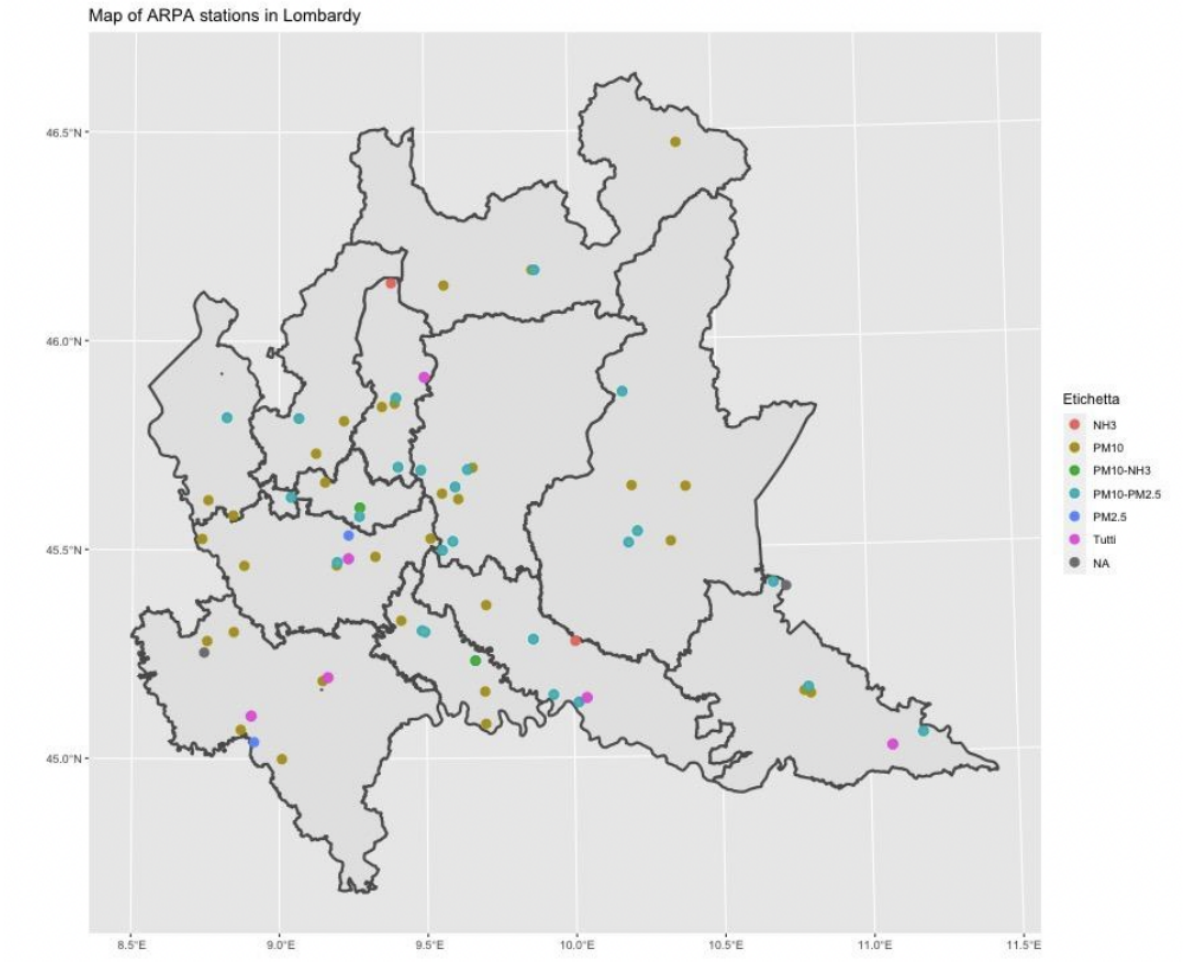

Figure 4.1: Lombardy map that highlights the geographic position of air quality stations that measure at least one pollutant among NH3, PM10 and PM2.5.

The three best results were chosen for the upcoming modeling phase. One of them is categorized as belonging to an urban area (U- urban) and two of them to rural areas (R- rural) respectively, in which the relationship between Ammonia and particulate matter is expected to be stronger.

In particular it was found that Cremona via Fatebenefratelli (U - urban) in the period of interest (2018-2020) has only 42 missing data on ammonia, 39 for PM10 and 7 for PM2.5, additionally there are only two days in which all three are missing.

Moggio (R - rural) has 112 days from 2018 to 2020 where NH3 was missing, 55 in the case of PM10 and 66 in the case of PM2.5. all three are missing simultaneously 22 times.

Schivenoglia (R - rural) has 380 missing observations from 2018 to 2020 in the case of ammonia, 76 in the case of PM10 and 84 in the case of PM2.5. All 3 were missing at the same time 15 times.

The results are presented in table 4.1.

2018-2020	Ammonia	PM10	PM2.5	Missing all 3 regressors
Cremona (U)	42	39	73	6
Moggio (R)	112	55	66	22
Schivenoglia (R)	380	76	84	38

Table 4.1: Missing data for the three stations from 2018 to 2020

Logically, extending the time period subject to analysis brings forth an increase of the absolute number of missing values across the board. Table 4.2 shows the results when analyzing the years between 2014 and 2020.

2014-2020	Ammonia	PM10	PM2.5	Missing all 3 regressors
Cremona (U)	74	119	169	25
Moggio (R)	311	191	237	53
Schivenoglia (R)	675	153	150	60

Table 4.2: Missing data for the three stations from 2014 to 2020

In order to visualize the distribution of the missing observations for each station, custom time series plots were created (figure 4.2, figure 4.3 and figure 4.4) in which every day that presents no observation for a given pollutant is highlighted by a blue, vertical line.

Among the chosen ones, Schivenoglia is perhaps the most problematic when it comes to the concentration of ammonia, as evidenced by Figure 4.2. In fact, out of the 1096 days between the beginning of 2018 and the end of 2020 there are 380 missing observations, which equates to a missing value rate of 34.67 %, which far exceeds the 3.83 % of Cremona and the 10.22% of Moggio. This problem is particularly pronounced in the year 2019.

Cremona and Moggio, on the other hand, have a much more consistent track record in measuring the concentration of ammonia, as can be seen in figure 4.3 and figure 4.4 respectively.

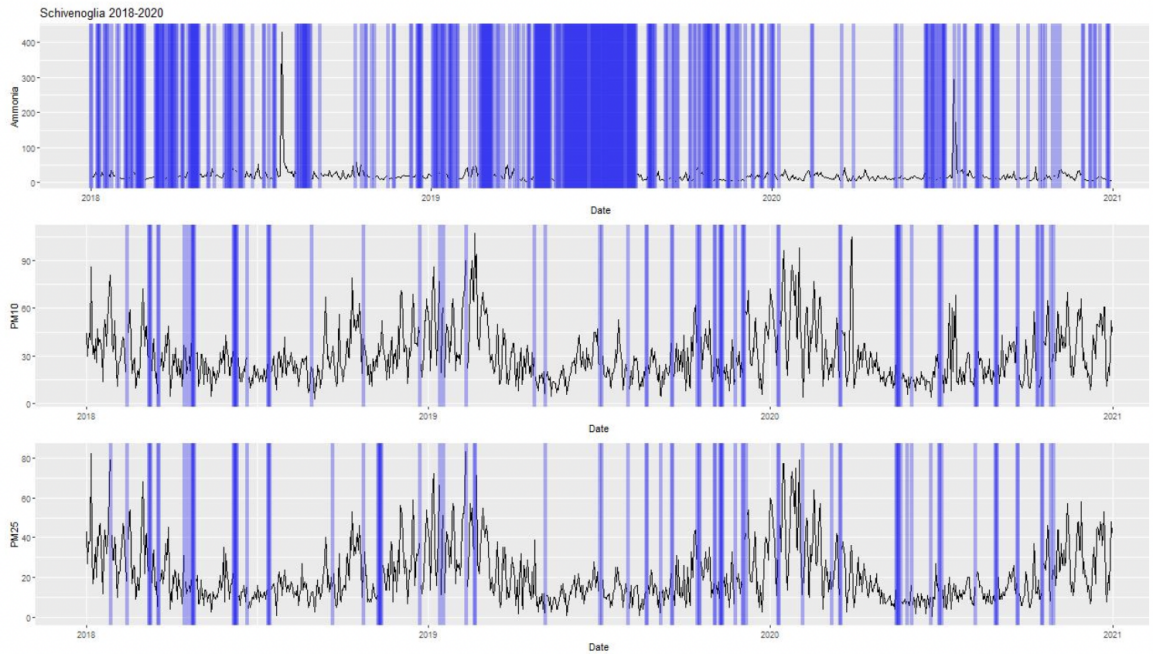

Figure 4.2: Visual representation of the missing observations in Schivenoglia from 2018 to 2020

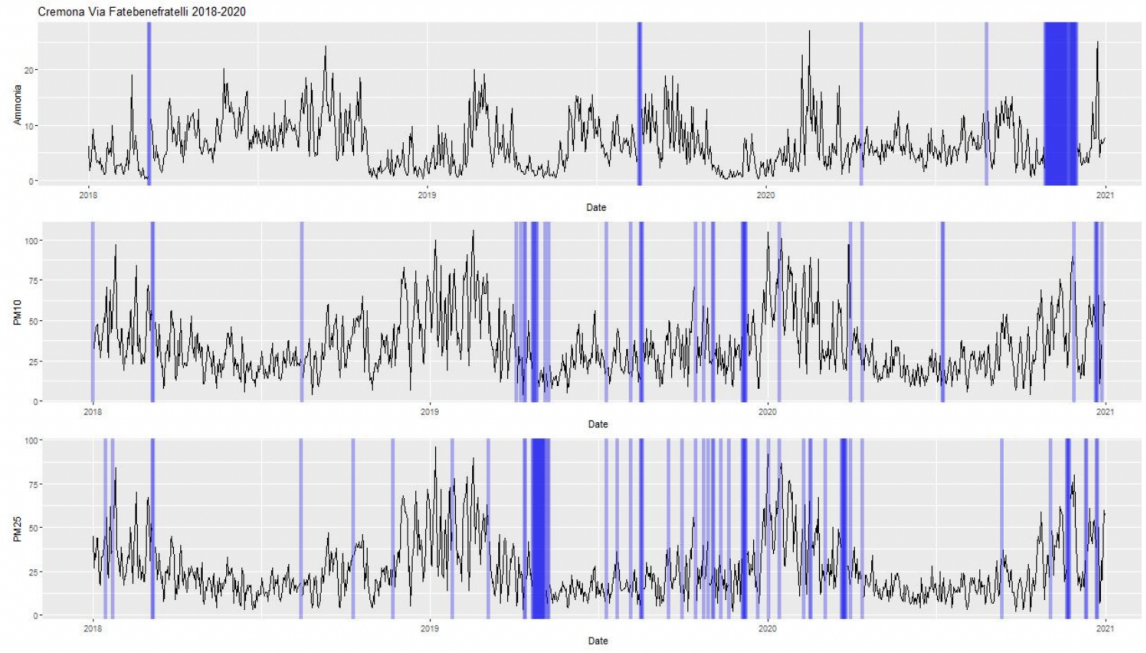

Figure 4.3: Visual representation of the missing observations in Cremona via Fatebenefratelli from 2018 to 2020

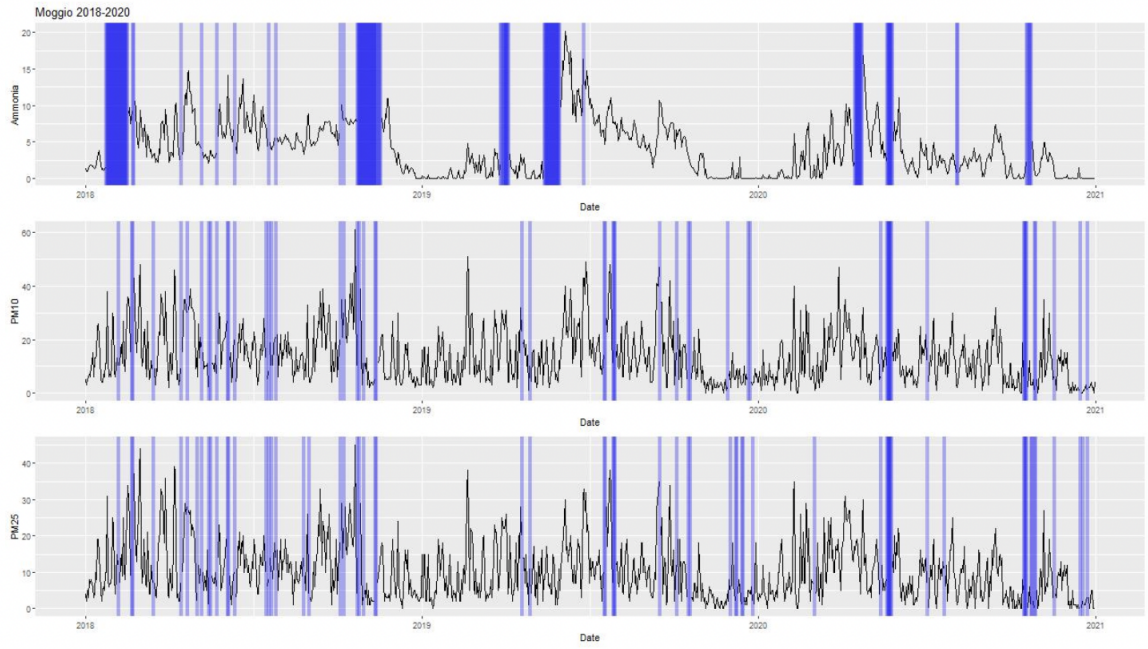

Figure 4.4: Visual representation of the missing observations in Moggio from 2018 to 2020

4.1.1 Improvement of data for training

In order to improve the quality of the input data to the neural network –and consequently improve the model– the input dataset was modified. In fact, the angle of the direction of the wind was replaced by a number that represents one of four categories, each of which corresponds to one of the four quadrants of the coordinate plane. The substitution takes place by using the number representing the quadrant that matches the general direction of the angle that it replaces. More specifically, it works as follows:

- Wind direction angles between 0° and 90° are replaced with the number 1.
- Wind direction angles greater than 90° and less than equal to 180° are replaced with the number 2.
- Wind direction angles greater than 180° and less than equal to 270° are replaced with the number 3.
- Wind direction angles greater than 270° and less than equal to 360° are replaced with the number 4.

By doing this, the model is unlikely to consider adjacent or nearly adjacent degrees in the unit circle, such as 359° and 0°, as erroneously representing completely different directions. In this way it is possible to make a discretization of the model.

4.2 Descriptive data analysis

	Ammonia	PM10	PM25	Wind_speed	Temperature	Rainfall
count	1822	2339	2337	2477	2476	2477
mean	16,714	32,645	23,497	1,2	14,829	1,53
std	16,868	18,634	15,415	0,912	8,421	4,642
min	0	3	0	0,016	-30	0
25%	10	20	13	0,549	8,139	0
50%	14,3	29	19	0,98	14,809	0
75%	20,2	41	31	1,603	22,007	0,2
max	430,6	255	123	8,353	31,844	54,4

Table 4.3: Descriptive data analysis of the different features from Schivenoglia's stations.

	Ammonia	PM10	PM25	Wind_speed	Temperature	Rainfall
count	1822	2339	2337	2477	2476	2477
mean	16,714	32,645	23,497	1,2	14,829	1,53
std	16,868	18,634	15,415	0,912	8,421	4,642
min	0	3	0	0,016	-30	0
25%	10	20	13	0,549	8,139	0
50%	14,3	29	19	0,98	14,809	0
75%	20,2	41	31	1,603	22,007	0,2
max	430,6	255	123	8,353	31,844	54,4

Table 4.4: Descriptive data analysis of the different features from Moggio's stations.

	Ammonia	PM10	PM25	Wind_speed	Temperature	Rainfall
count	2407	2363	2314	2470	2470	2470
mean	7,662	36,767	27,403	0,632	15,118	2,089
std	5,251	21,124	18,411	0,463	8,359	6,88
min	0,3	4	0	0,005	-2,742	0
25%	4	22	15	0,322	7,974	0
50%	6,5	32	23	0,522	15,232	0
75%	10,1	47	35	0,803	22,338	0,2
max	40,2	175	159	3,864	32,967	82,8

Table 4.5: Descriptive data analysis of the different features from Cremona's stations.

4.3 Addition of meteorological data to the dataset

Having analyzed the air quality data, it is now necessary to study the weather data that, by virtue of their direct interaction with the pollutants, can offer additional information to improve the predictive capabilities of any potential model.
Air quality and weather data stored separately, and a station of the former does not necessarily measure the latter as well.
Assuming that the weather data logged by a weather station close to an air quality station will have information that is representative of the meteorological conditions around it, it is then possible to aggregate the two datasets into one that can be directly used to train the models.
It was therefore necessary to create an algorithm that would find the weather station with the shortest Euclidean distance from a given air quality station. Moreover, in order to guarantee that the necessary data would be available, the weather stations needed to meet certain criteria: they needed to measure wind speed and direction, temperature and precipitation (irrespective of the amount of missing observations, which will be analyzed afterwards) and they needed to be active since at least 2018, so that there would be enough data to work with. Figure 4.5 and table 4.6 show the results of this algorithm.

Air quality station	Closest	Second closest	Third closest	Fourth closest
Moggio Loc Penscei	Cassina Valsassina Moggio 34.16	5200.87	9315.78	11959.18
Cremona Via Fatebenefratelli	Cremona Via Fatebenefratelli 0	2622.66	2622.66	8924.53
Schivenoglia Via Malpasso	Mantova Tridolino 22802.93	24168.28	24955.88	25260.47

Table 4.6: Distances of the four weather station candidates with respect to each of the air quality stations.These stations measure wind speed and direction, temperature and precipitation, and they are active at least since 2018.

Executing the algorithm once again, and not imposing the aforementioned constraints, namely the requirement to measure wind speed and direction, temperature and precipitation results in the selection of stations from figure 4.6 and table 4.7 . When it comes to the weather stations closest to the air quality station, the only difference is for Schivenoglia. In fact there's a weather station at roughly 16km, instead of approximately 23 km, that would be more appropriate if it weren't for the fact that the wind speed and direction data in 2018 is unavailable for roughly 75% of the year.

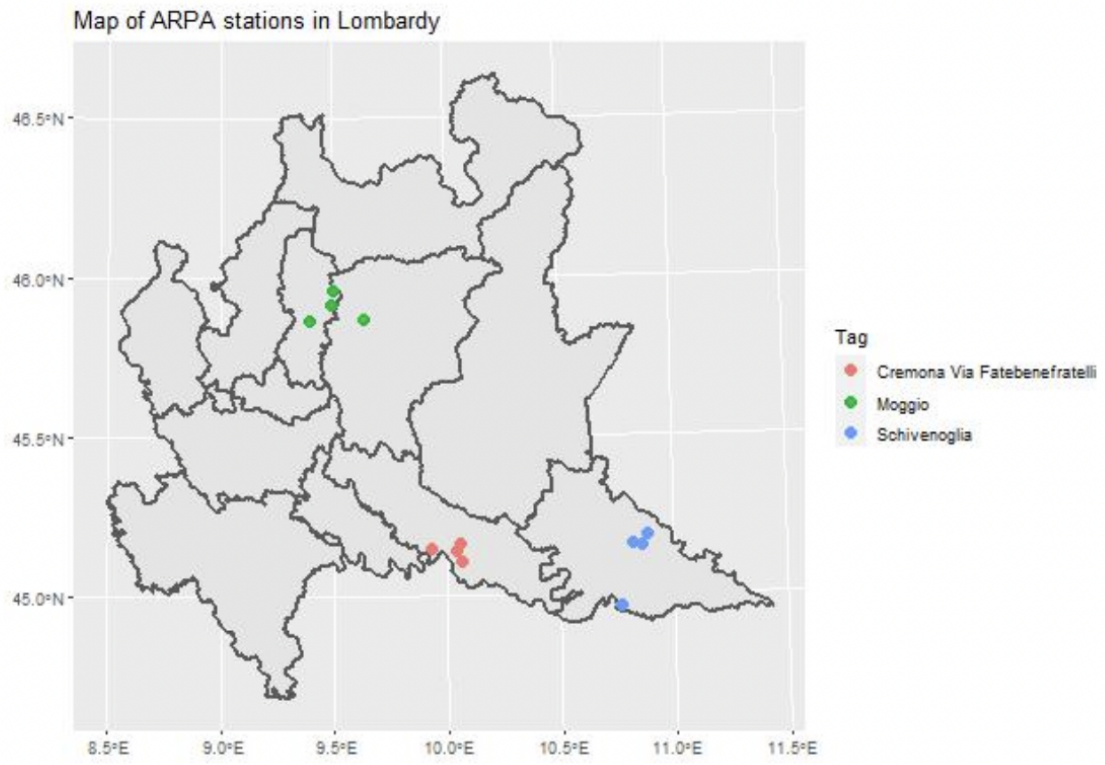

Figure 4.5: Map of Lombardy that reflects the positions of the four weather station candidates for each of the air quality stations. These stations measure wind speed and direction, temperature and precipitation, and they are active at least since 2018.

Air quality station	Closest	Second closest	Third closest	Fourth closest
Moggio Loc Penscei	Cassina Valsassina Moggio 34.16	5200.87	7480.26	8181.56
Cremona Via Fatebenefratelli	Cremona Via Fatebenefratelli 0	2622.66	4175.53	8924.53
Schivenoglia Via Malpasso	Sermide E Felonica Sp 6808.98	22802.93	24168.28	24955.88

Table 4.7: Distances of the four weather station candidates with respect to each of the air quality stations. These stations do not necessarily measure wind speed and direction, temperature and precipitation, and they may not be active since 2018.

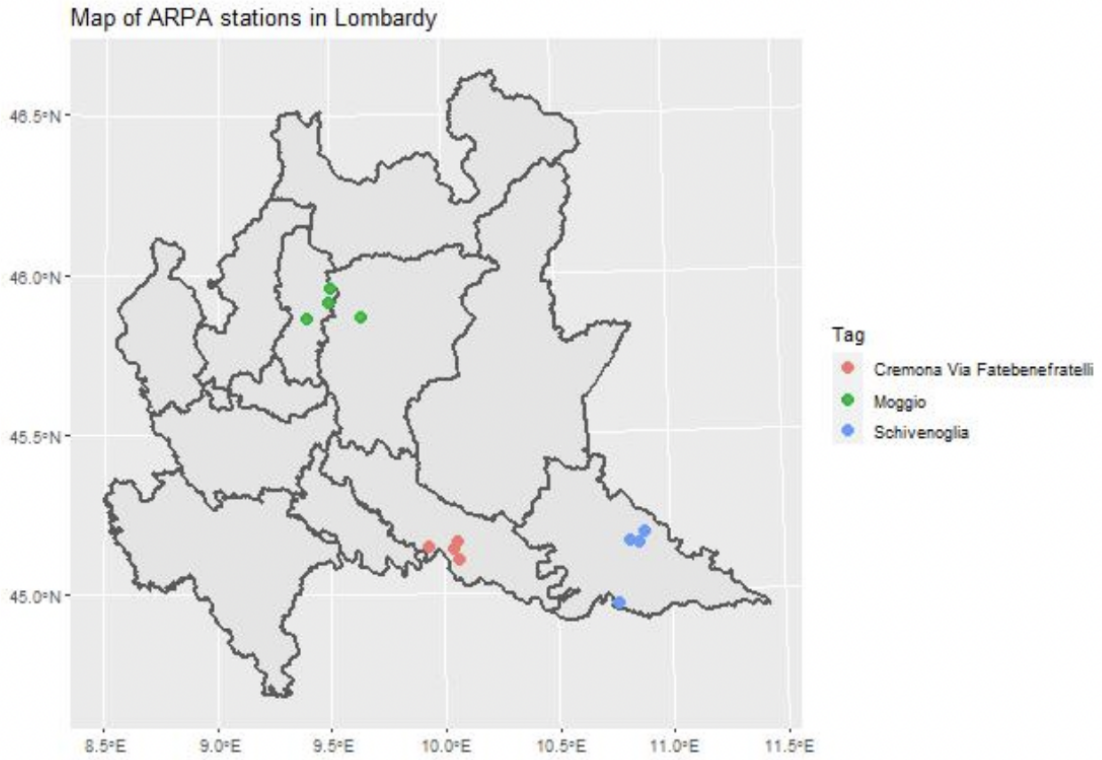

Figure 4.6: Map of Lombardy that reflects the positions of the four weather station candidates for each of the air quality stations. These stations do not necessarily measure wind speed and direction, temperature and precipitation, and they may not be active since 2018.

Therefore, the tradeoff proposed by the other stations that do meet the criteria is worthwhile, given that they are guaranteed to provide weather variables that are expected to be informative for a forecasting model.

4.4 Missing data analysis for meteorological data

In figure 4.7, figure 4.8 and figure 4.9 any missing observations for the chosen variables are highlighted in orange for the selected weather stations for Moggio, Cremona and Schivenoglia respectively. Conveniently, there are virtually no missing observations, with the exception of some isolated instances in Cremona's, and especially in Schivenoglia's associated weather station.

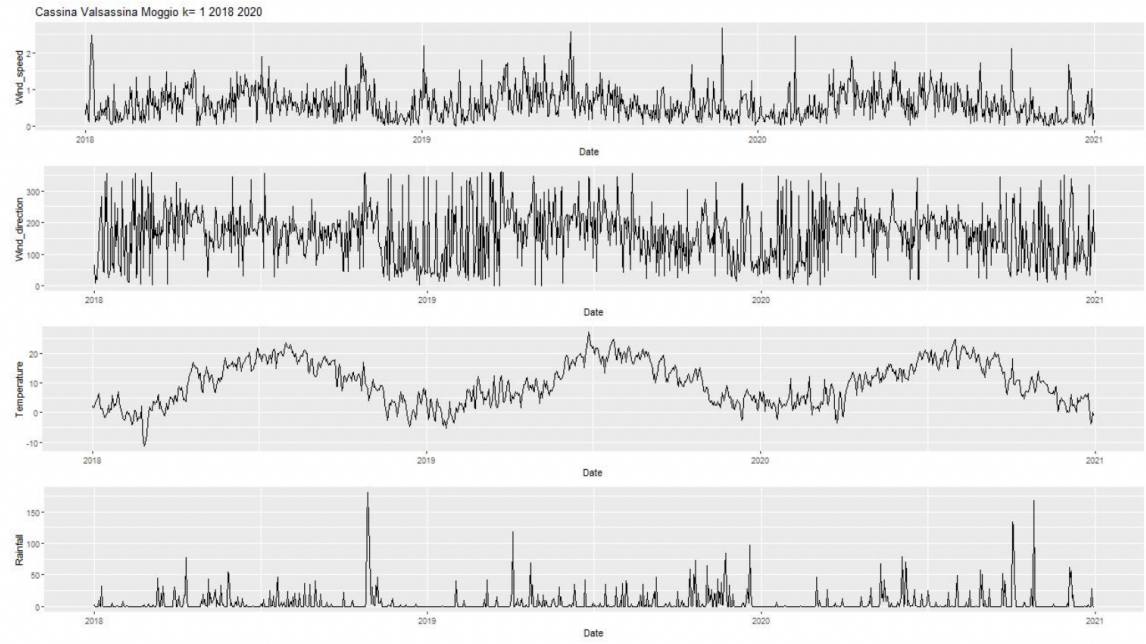

Figure 4.7: visual representation of the missing observations for weather variables in Cassina Valsassina Moggio

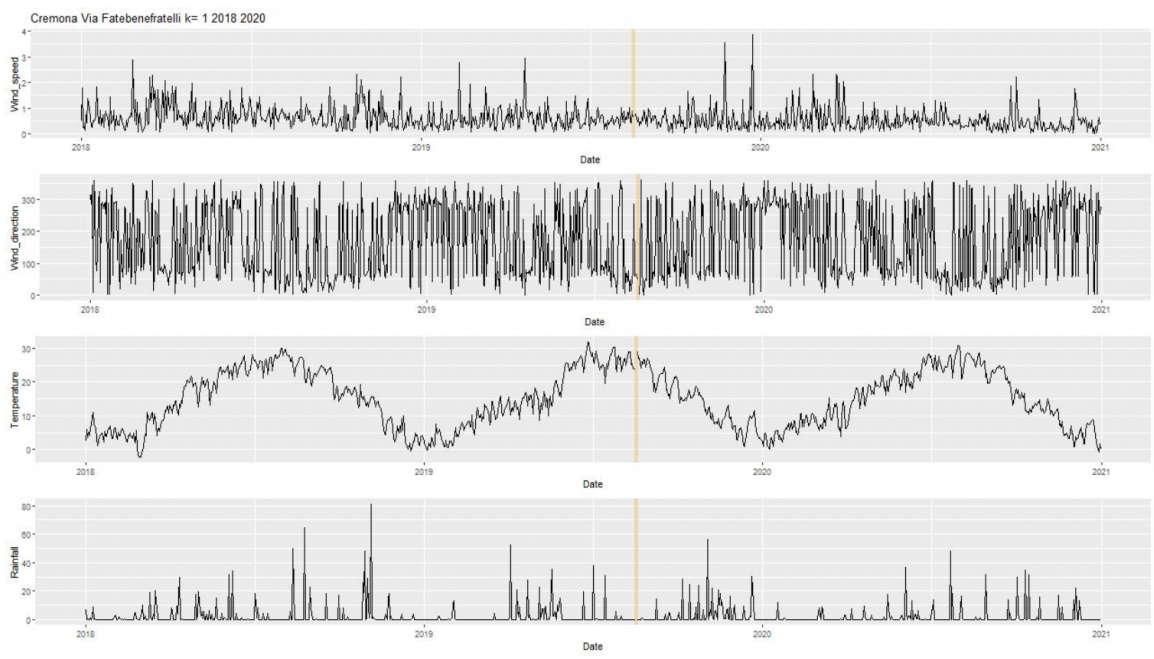

Figure 4.8: visual representation of the missing observations for weather variables in Cremona via Fatebenefratelli

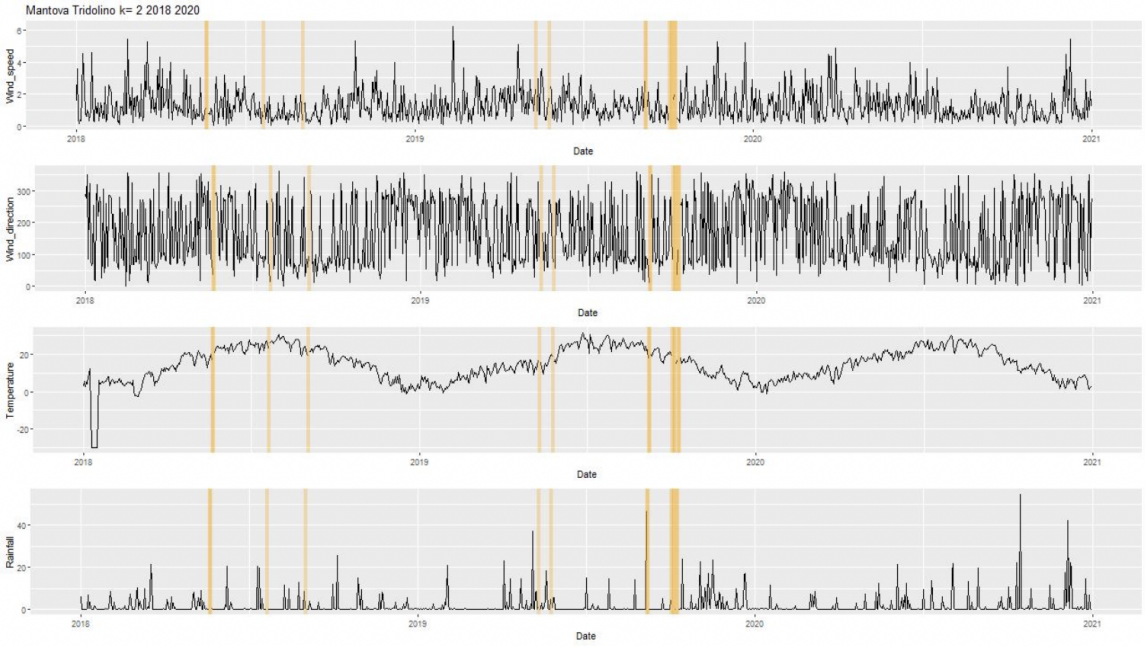

Figure 4.9: visual representation of the missing observations for weather variables in Sermide and Felonica

4.5 Linear relationships in the data

While neural networks are not limited to linear relationships between variables, determining which variables are significantly linearly correlated to the response variable –often referred to as target feature in the context of machine and deep learning– can help form a preliminary understanding of how informative these can potentially be.

Figure 4.10, figure 4.11 and figure 4.12 contain the scatterplots for the chosen air quality and meteorological variables in the Moggio, Cremona and Schivenoglia stations respectively in the time period from the beginning of 2014 to the end of 2020. Looking at the first row of the scatterplots, which shows how the different variables correlate to PM10, the target feature, it is clear that PM2.5, in all cases, presents a strong linear correlation with it that ranges from a Pearson coefficient of 0.92 in Schivenoglia to 0.97 in Moggio. It is therefore expected to be the variable that contains the most information with respect to PM10, which comes as no surprise given that one is a subset of the other.

All other variables present significantly lower coefficients, with the temperature being the second strongest for Cremona and Schivenoglia, where it happens to be negatively correlated to PM10, and Ammonia which is strongest in the case of Moggio.

All of these observations, however, are limited in scope, not only due to possible nonlinear relationships but also due to the fact that it ignores possible correlations through time are ignored, both of which can be exploited by an LSTM neural network due to its own design (Chapter 2.8.2).

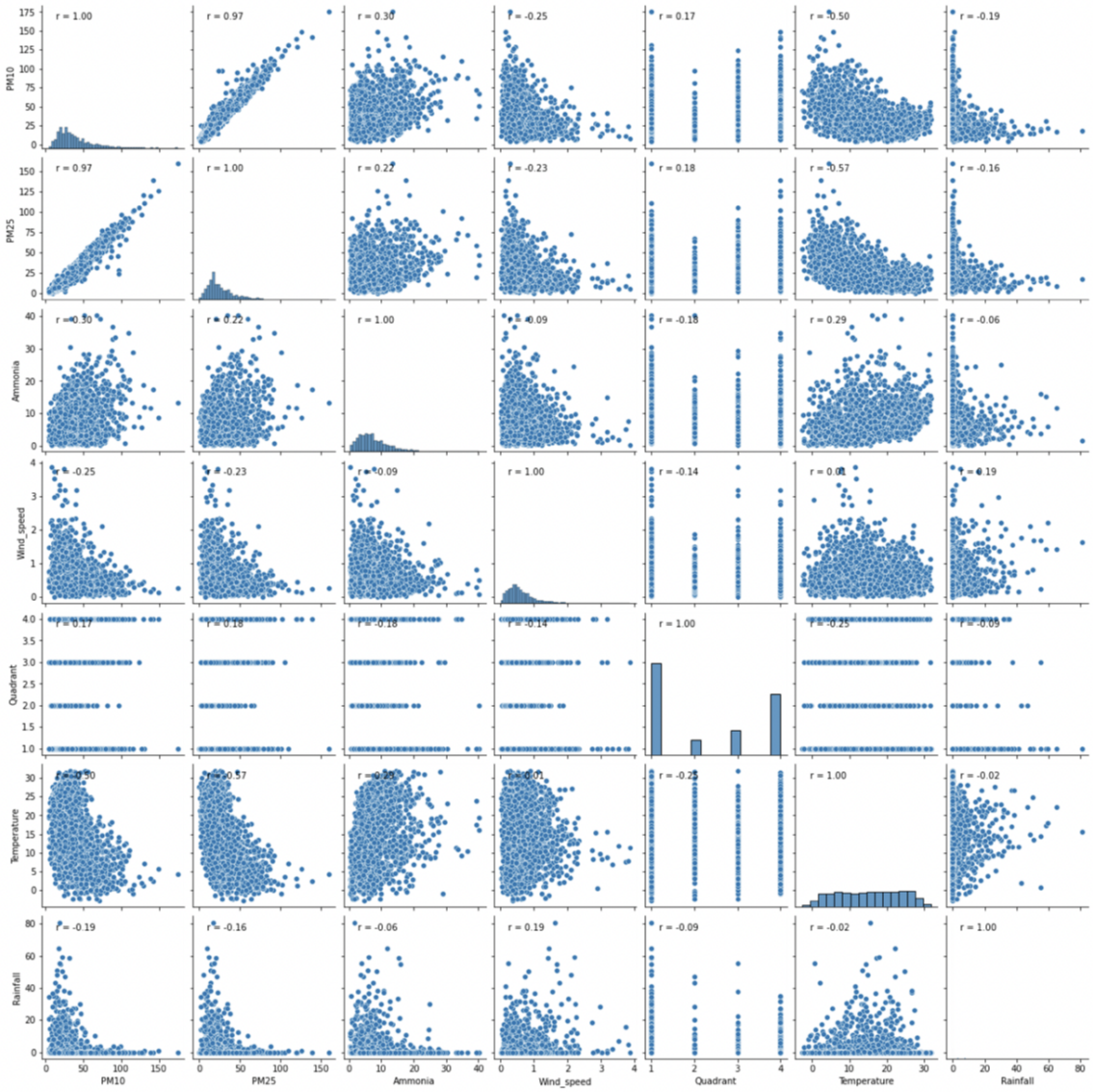

Figure 4.10: Scatterplot and Pearson coefficients for all variable pairs in Cremona

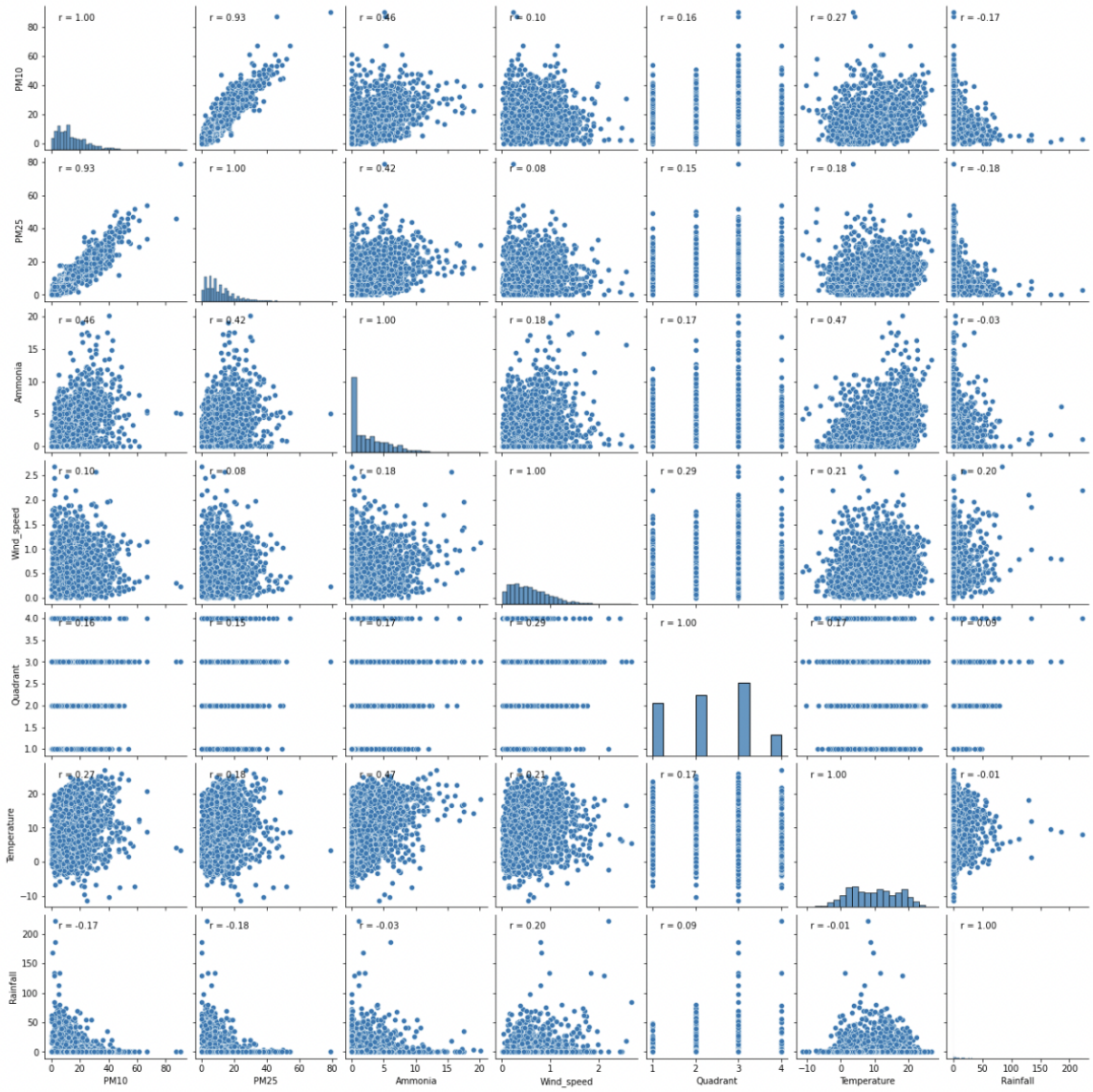

Figure 4.11: Scatterplot and Pearson coefficients for all variable pairs in Moggio

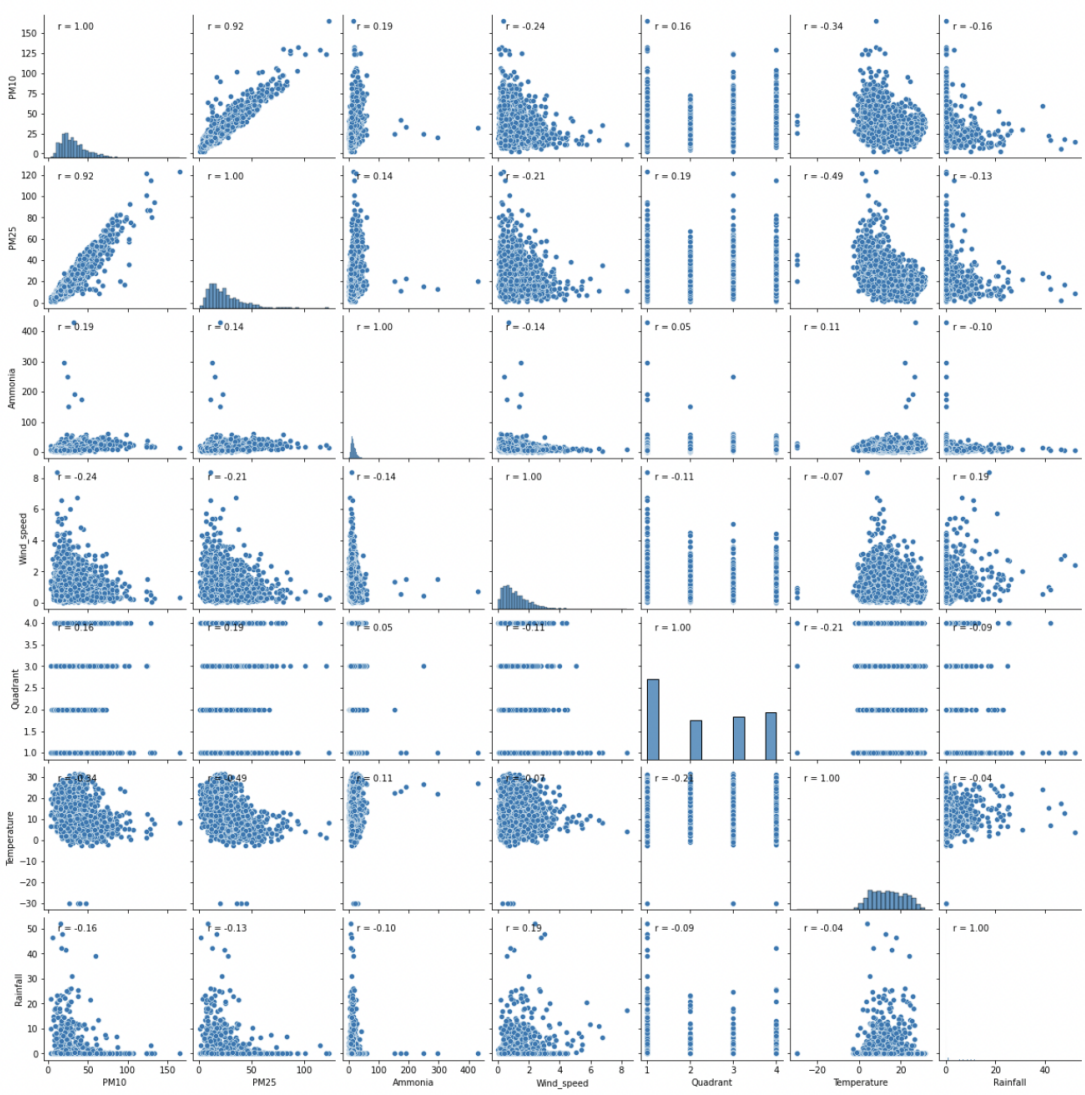

Figure 4.12: Scatterplot and Pearson coefficients for all variable pairs in Schivenoglia

CHAPTER

5 Modeling

In this chapter, the datasets are used to build models capable of forecasting the concentration of PM10, using the results of the exploratory data analysis from Chapter 4 as a starting point. These models will be optimized by means of hyperparameter optimization and efforts will be made to inspect their reasoning by using surrogate linear models.

5.1 Description of the general structure

The different solutions proposed to devise a neural network capable of achieving the task at hand all use the same base structure. In fact, it is possible to give a generalized description of the structure of all the different scripts that are used throughout the entire case study.

At a high level of abstraction, the strategy is composed of three core parts: preprocessing of the tabular data, tailored to the specific subtask; construction of the network's architecture, which can include the use of a RandomSearch approach to explore the hyperparameter space; and finally the execution of different predictions that put the model to the test.

5.1.1 Tailored, in-script, pre-processing

The bulk of the data engineering operations that were applied to the raw data coming from ARPA Lombardia takes place in an entirely different phase that precedes the modeling phase itself, which is described in detail in Chapter 4.

Despite being essential to the construction of all the models derived thereof, such an extensive preprocessing phase often needs to be followed by a series of finishing touches that prepare the dataset to be used.

Chief among these alterations is the removal or substitution of missing values: as discussed in chapter 2.9, dealing with the absence of certain data points is central to optimize and perfect the quality of the data and, in turn, that of the predictions that are constructed based on it.

Moreover, missing values are not automatically handled by regular LSTM networks. Consequently, and depending on the approach that is to be followed, they either need to be removed indiscriminately –which represents the simplest of the possible solutions– or assigned special values that can be consecutively learned, so to speak, by the network itself.

The data is further trimmed to a more manageable state by removing all of the columns, that is, all of the features that are not to be taken into account during the modeling phase. In fact, the core datasets are built to include the widest array possible of variables that are measured by their respective sensors, but these are rarely used all together.

These steps are followed by another crucial modification, which can be briefly described as the transformation of the tabular set, into a three-dimensional set that can be directly used by the LSTM model.

To this end, a custom Python function is used that describes the same data, but this time with a specific number of lags that is equivalent to the number of days of information that will eventually be necessary to produce an output of the model. Additionally, it is possible to specify the number of future lags that can be used to train the network in such a way that it can predict a number of days equivalent to the future lags.

The second step in this three-dimensional reconstruction of the information consists of taking the output of this function and reshaping it into a 3D matrix whose dimensions are, namely (number of observations) x (number of lags) x (number of features). This is done right

after having divided the entire dataset into a training set, testing set, and validation set.

5.1.2 Construction of the network's architecture

Once the data is fully ready to be fed into the model, the network itself must be built. While this can be done using specific and constant hyperparameters, more advanced approaches can also include testing different combinations of value sets so that, subsequently, the best-performing model can be chosen. In other words, the hyperparameter space is partially explored to better tune the model.
The first order of business is, therefore, to define a flexible code structure for the model that can be repeatedly instantiated to test all of the different combinations. Its architecture must therefore be parametric such that it is flexible enough to account for different numbers of LSTM layers, neurons per layer, different learning rates, batch sizes, and so on.

When using the Random Search algorithm (chapter 2.10.0.2), different hyperparameters, chosen from a previously specified set, are paired randomly to evaluate their impact on performance. To do so, a maximum number of trials needs to be chosen, which represents the maximum number of combinations that can be evaluated. Additionally, the number of executions per trial can be explicitly set to a value different from its default, which is one: this can be useful in order to assess the different possible results that can come from one single network, given the stochastic nature of the training process.

Once the training process is completed –which can take a variable amount of time depending on the complexity of the model and the nature of the dataset– there will be as many models as there were iterations of the hyperparameter tuning algorithm to choose from. At this point, the best model, in terms of validation loss or any other metric, is chosen to proceed with the third and last phase.

5.1.3 Predictions and evaluations

Once the model has been trained, it can be immediately used to make predictions. Generally speaking, these predictions are made using data that wasn't previously presented to the network, and that can also be used to calculate different metrics to evaluate performance.

The resulting plots and metrics are stored so that they can be used to compare the results of different approaches or datasets coming from different places across Lombardy.
The extent to which this phase goes into detail depends on different factors, like the intended use of the model and what it needs to be compared to.
Of notable interest in this phase is the degree to which the model's predictions can be explained. Understanding why a choice has been made often helps measure the trustworthiness of the results, if only qualitatively. Yet, neural networks are inherently obscure in their inner workings, which means that, if a certain degree of explainability is expected, there needs to be an additional effort that goes beyond the mere construction of the network.
While other types of models, outside of the deep-learning category, offer in-built explainability, for neural networks external techniques need to be used and may not be applicable in all cases.
One such technique is Lime, which stands for Local Interpretable Model-Agnostic Explanations and is explained in more detail in chapter 2.11.1.1 This explainability toolkit can be accessed through a Python library, and it can easily offer insight into the importance of the different features of the dataset on a specific prediction, using a highly-accurate local linear model as a surrogate for the behavior of the neural network.
Lime explanations can therefore be a crucial aspect of this last phase, whenever applicable and relevant to the use case at hand.

5.2 Baseline models

5.2.1 Baseline: persistence model

In machine and deep learning it is common practice to use one or several so-called naive models that, as the name suggests, use very simple algorithms that are expected to provide reasonable predictions given the nature of the problem.
In classification problems, the predicted class is determined by either choosing a class at random: Random Prediciton algorithm, or by always choosing the most common class in the dataset: the Zero Rule algorithm.
Neither of these techniques, however, are appropriate for the problem at hand. When dealing with time series, the data can be continuous in nature instead of being restricted to a limited amount of classes and, perhaps more importantly, data-points can be autocorrelated through time and space.
Therefore, despite the required simplicity of a baseline model, there still is a need to implement one that takes into account the nature of the dataset. The
Persistence algorithm is one such strategy; it, quite simply, predicts the value of a target feature at a given time step as the value of that same feature at the previous time step.
It is possible to test the performance of said model for any feature in the different datasets. For instance, when applied at the Moggio Loc Penscei air quality station for the concentration of PM10 in the year 2019, a MAE of 6.232, RMSE of 8.324 and R2 of 0.328 are achieved.

In figure 5.1,where missing values are set to the mean of the entire distribution of the feature across the entire year, it is quite clear that the predicted value is exactly the same as the previous datapoint in the observation set. As a consequence, phenomena that depend heavily on the previous observation's value perform well with this type of naive model.
When applying the same algorithm to the observations of PM10 during 2020 in Moggio Loc Penscei (Figure 5.2), the model achieves a MAE of 5.456, a RMSE of 7.662 and a R2 of 0.156.

Table 5.1 reflects the performance of this naive model across a series of different pollutants during 2020 in Moggio Loc Penscei.

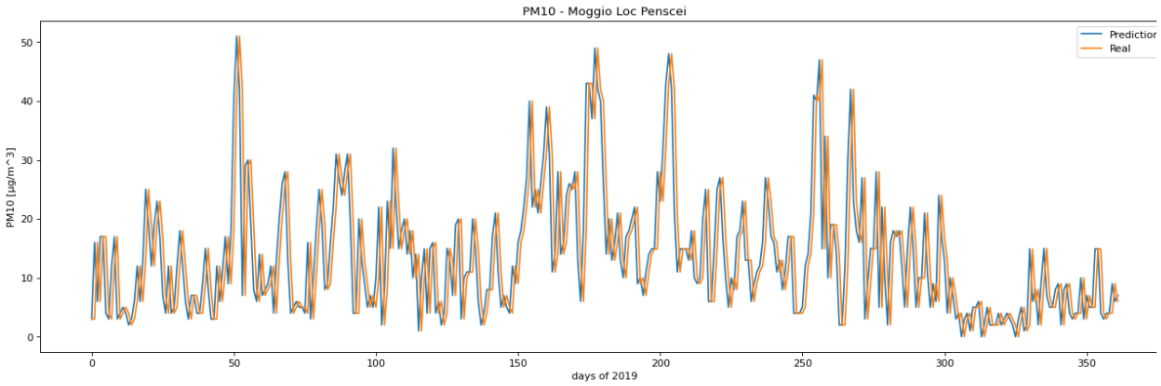

Figure 5.1: Forecast for 2019 made by the Moggio Loc Penscei's naive baseline model based on the persistence algorithm, which predicts the value of a target feature at a given time step as the value of that same feature at the previous time step.

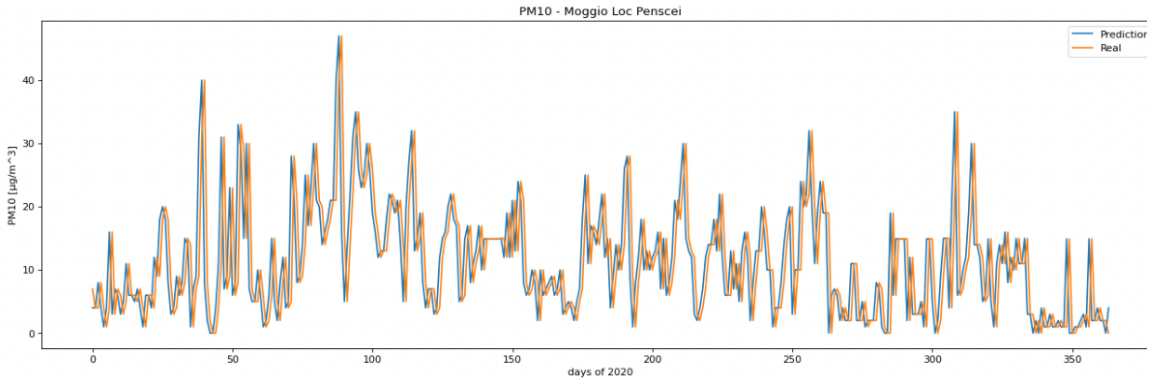

Figure 5.2: Forecast for 2020 made by the Moggio Loc Penscei's naive baseline model based on the persistence algorithm, which predicts the value of a target feature at a given time step as the value of that same feature at the previous time step.

While other pollutants, such as ozone (see table 5.1) achieve substantially high metrics, when it comes to PM10, as seen in table 5.3, the predictive performance of the persistence model in any of the stations leaves ample room for improvement, with R2 metrics that do not surpass 0.559.

Crucially, these performance metrics for PM10 are not consistent year over year, demonstrating that the model does not generalize well.

Pollutant	RMSE	MAE	R2
Ammonia	1.418	0.811	0.720
Ozone	22.339	16.050	0.465
PM25	6.127	4.380	0.208
PM10	7.662	5.456	0.156
NOx*	4.388	2.651	-0.381
NO2*	4.331	2.646	-0.516

Table 5.1: Performance metrics for the persistence model baseline across a series of different pollutants during 2020 in Moggio Loc Penscei.
$*$ NOx and NO2 readings were enabled during early 2020, so there's less data to work with.

Pollutant	RMSE	MAE	R2
Ozone	10.223	7.594	0.893
CO	0.106	0.075	0.762
NOx	21.742	13.740	0.700
NO2	6.661	4.953	0.677
PM10	13.752	9.862	0.559
PM25	12.032	8.118	0.534
Ammonia	3.136	2.031	0.274
Sulfur_dioxide	0.689	0.445	0.177

Table 5.2: Performance metrics for the persistence model baseline across a series of different pollutants during 2020 in Cremona Via Fatebenefratelli

Station	Pollutant	Year	RMSE	MAE	R2
Cremona Via Fatebenefratelli	PM10	2019	13.574	10.288	0.480
Moggio Loc Penscei	PM10	2019	8.324	6.232	0.328
Schivenoglia Via Malpasso	PM10	2019	12.103	9.001	0.486
Cremona Via Fatebenefratelli	PM10	2020	13.752	9.862	0.559
Moggio Loc Penscei	PM10	2020	7.662	5.456	0.156
Schivenoglia Via Malpasso	PM10	2020	13.749	9.606	0.430

Table 5.3: Performance metrics for the persistence model baseline for the forecast of PM10 during 2020 in Cremona Via Fatebenefratelli, Moggio Loc Penscei and Schivenoglia Via Malpasso.

5.2.2 Neural networks baseline

Once the data has been processed as described in Chapter 4 and Chapter 5.1.1, the next step is the creation of a model that can accurately emulate its behavior. Neural networks are quite flexible with the type of data they can be fed and expected to produce, but as a first attempt, it was chosen to simply forecast the values of PM10 one day into the future, using different air quality and weather variables as input features for the model, including past values of PM10 itself.

As mentioned in Chapter 5.1.1, the dataset needs to be modified so that it can be better suited to do multi-step regression with Keras TensorFlow. Thus, instead of using the strictly tabular data directly, it is transformed into a three-dimensional matrix where the dimensions are (number of observations) x (number of lags) x (number of features).

For this first experiment, the aforementioned Moggio Loc Penscei air quality station is paired with its neighboring Cassina Valsassina Moggio weather station, given that they provide the training with the least missing values among the three proposed locations.

The variables that will be used for the model, among all made available by both stations, are:

- **PM10**: which is also the target feature, and was chosen as such given to the suspected relationship between ammonia and particulate matter.
- Ammonia: It is expected to be one of the more relevant features due to its suspected relationship with the target feature.
- **PM2.5**: which, as seen in the scatterplots of Chapter 4.5, has a strong linear relationship with PM10, potentially caused by the fact that PM2.5 is a subset of PM10.
- **Wind speed**: as the name implies, it reflects the speed of the wind that surrounds the weather station of origin.
- **Wind direction**: originally expressed in degrees, but, as seen in Chapter 4.1.1, it has been transformed into quadrants. It, therefore, acts as categorical data.
- **Temperature**: which, again, as seen in Chapter 4.5, has a non negligible degree of correlation to the target feature.

- **Rainfall**: Which, as the other meteorological variables, can have a direct impact in the distribution of and the interaction between pollutants.

The dataset spans from January 1, 2014, to December 31, 2020. Due to the possible impact on the rate of emissions caused by the first COVID-19 lockdowns in Italy, the entirety of 2020 is set aside to test the generalization capabilities of the model, it hence constitutes the test set. This leaves the period between 2014 and 2019 as the actual data that will be divided into a training and a validation set.

All the data up to December 31, 2018, constitutes the training set that will therefore be used to tune the model's parameters (weights and biases). The remaining data, the year 2019, is instead the validation set, which is, by definition, not used for parameter optimization. Instead, it will be a benchmark of how the model's performance on out-of-sample data gradually changes.

Since this model's role is to be a baseline, the most basic strategy for dealing with missing values is used: every single day with at least one of the features missing is discarded, which entails also discarding the observation of the other variables even if their value was available. When accounting for all the missing values, the dataset has a total of 2032 days in which none of the features had any missing observations. These observations are then scaled between zero and one with a MinMax Scaling strategy in which the smallest value is set to zero and the biggest value is set to one.

Since the model will use 5 consecutive days to determine the values of the target feature for the 6th day, the table to be given as an input for the model provides 5 consecutive observations of all of the features in a single row, which means that the very first 5 days of the dataset are all expressed in the first row of the table. With this in mind there's a total of 1391 input rows that constitute the training set, out of which the last row will, thus, include the last 5 days of 2018. Following the same logic, the validation set, when accounting for missing values, contains 314 days that come from the year 2019. After being transformed, the table that includes the 5-day lag still has 314 rows, because the reduction of rows was already accounted for in the testing set.

In much the same way, the testing set, that contains the observations of the year 2020, has 322

input rows.

Having said that, the shape of the three-dimensional matrices that can be built by reshaping the aforementioned sets are 1391 x 5 x 7 , 314 x 5 x 7 and 322 x 5 x 7 for the training, testing and validation set respectively, where the first dimension is the number of days, the second one is the number of lags or observations per feature used to calculate a prediction and the third one is the number of features.

These matrices can now be used for training, but to do that, it is necessary to first define the architecture of the neural network that will be used.

The baseline model has an input layer whose dimensions are number of (lags) x (number of features), namely 5 x 7; followed by one hidden layer with LSTM cells with input shape (lags) x (numbers of neurons), that is 5 x 10 and output shape 1 x 10, which is, in turn, connected to the output layer which is instead a Dense layer whose shape is 1 x 1 because it provides one single output relative to a single time instant, i.e. the predicted value of PM10 at the time instant that comes after the five time instants given in input.

At this point, there's no information available to determine the choice of hyperparameters to use. Hence the only guiding principle is to build a simple model with a standard configuration. The choice of having 10 neurons was arbitrary: its impact, as well as the impact of other hyperparameters, will be studied later during the exploration of the hyperparameter space (Chapter 5.4). Table (5.4) shows which hyperparameters were used.

Hidden layers	Neurons per hidden layer	batch size	learning rate	Maximum number of epochs
1	10	32	0.001	500

Table 5.4: Hyperparamenters used for the construction of the baseline neural network models for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso.

Once the architecture is ready, the training ensues. The steps followed are the ones described in the algorithm in Chapter 2.3, which are repeated iteratively epoch after epoch. The training finally stops either when a specific hard limit is reached –which is set to 500 epochs– or when the early-stopping mechanism determines that there are no significant improvements in 25 successive epochs. Regardless, the same early stopping mechanism retrieves the weights and biases of the best training iteration in terms of training loss, which in

this case is measured by the MAE, mean absolute error.

Once the training phase ends, the model is ready to be used, after having achieved a MAE of 0.0702 on the testing set and a MAE of 0.0608 on the validation set and after having stopped after 82 epochs of training.

To put it to the test, it is necessary to give it an input that respects the same specification used to restructure the training data: it needs to be a three-dimensional matrix that is scaled between zero and one. The output is expected to be within the same zero-to-one range and therefore needs to be scaled back for it to be measured against observed values.

When used on the validation set, which contains the data of 2019, the model achieves an RMSE of 7.222, a MAE of 5.451 and a R2 of 0.519, rounding off to three decimal places. The prediction's time series is depicted in Figure 5.3.

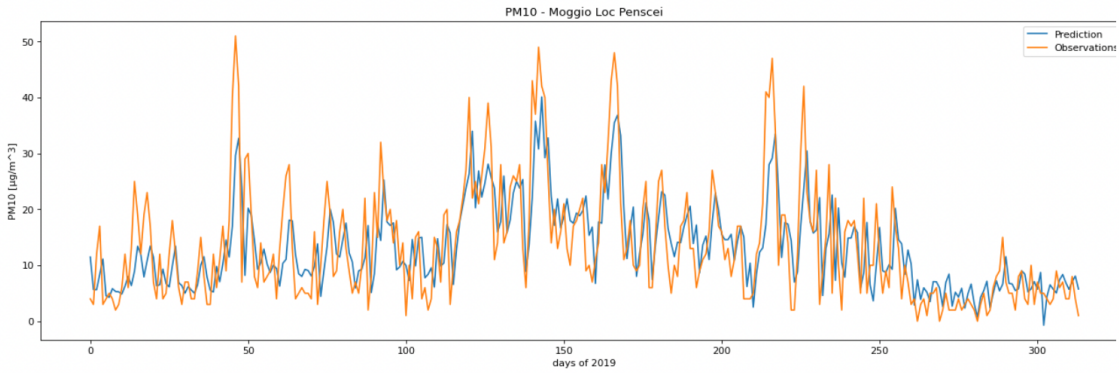

Figure 5.3: Forecast for 2019 made by the Moggio Loc Penscei's neural network baseline model

On the testing set, with the data of 2020, the results are less promising, with an RMSE of 6.874, a MAE of 5.191 and a R2 of 0.349. The prediction's time series is once again depicted, this time in Figure 5.4.

As was pointed out before, 2019 and 2020 are the validation and testing set respectively, neither of which are used to tune the parameters of the network. The forecasts presented above are, hence, out-of-sample predictions, in that the data upon which they were made wasn't used for training purposes.

he drop in performance between 2019 and 2020 could be attributed to the fact that the model was trained exclusively on pre-COVID-19 data. The model may, thus, be better suited to

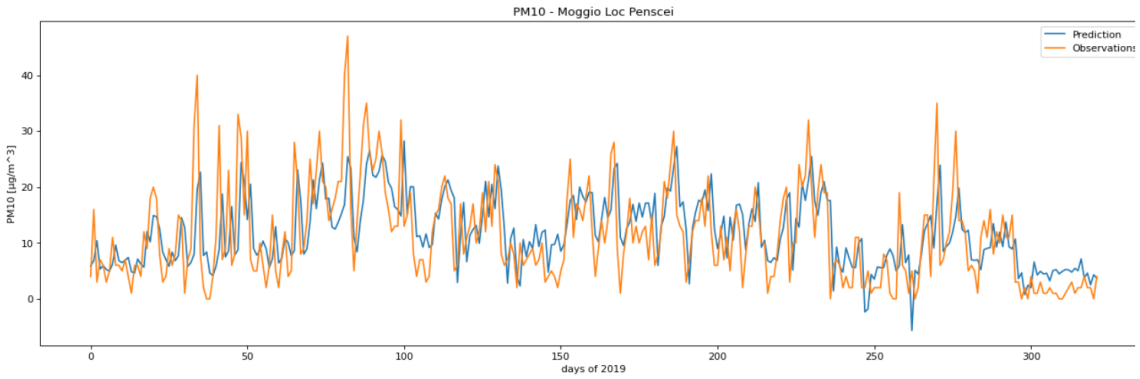

Figure 5.4: Forecast for 2020 made by the Moggio Loc Penscei's neural network baseline model

generalize on data that follows pre-pandemic dynamics, rather than data that may present newly introduced behaviors [5].

What is more, upon inspection of Figure 5.3 and Figure 5.4, the model seems to be relying heavily on the previous timestep of the target feature to inform the prediction. In fact, while its predictions do present slightly different value oscillations instead of being an exact copy, the prediction seems to trail behind the observed data exactly by one timestep, much like in the baseline persistence model. A more quantitative assessment of this phenomenon is presented in Chapter 5.5.

Table 5.5 contains the results of this very same architecture in all three different locations.

Location	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Moggio Loc Penscei	7.222	5.451	0.519	6.874	5.191	0.349
Cremona Via Fatebenefratelli	12.405	9.299	0.612	11.102	7.929	0.674
Schivenoglia Via Malpasso	12.834	9.847	0.431	13.043	9.528	0.510

Table 5.5: Performance metrics for the forecast of the baseline neural network models for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso.

Drawing from the results of these models, a large number of different optimizations and changes in approach can be conceived. Exploring these can prove to be worthwhile owing to their potential to improve performance. Chapter 5.3 explains some of these different proposals.

5.2.3 Linear model baseline

The use of non-naive models as baselines also has the potential of being very informative as a benchmark for the neural network models. If a linear model is used as a baseline, it can for instance serve as a tool to determine whether the non-linearity of neural networks is actually contributing to the predictive performance of the model, or whether a simpler linear model could suffice.

For this very purpose, regression models with ARIMA errors were built for all three stations starting from the same set of regressors used in the neural network baseline models presented in Chapter 5.2.2.

In order to avoid multicollinearity problems, the feature set, which originally contained the values of PM10, PM2.5, NH3, wind speed, wind direction, temperature, rainfall for the five days prior to each prediction, was trimmed down using stepwise variable selection. This allowed for the removal of all non-statistically-significant regressors, for which a p-value threshold of 5% was imposed.

The parameters of the ARIMA part of the models were determined using a variation of the Hyndman-Khandakar algorithm [11], which combines unit root tests, minimisation of the AICc and MLE to obtain an ARIMA model, which is made available as a library for statistics programming language R.

The resulting models in Eq (5.1) Eq (5.2) and Eq (5.3) for Moggio, Cremona and Schivenoglia respectively vary slightly in terms of ARIMA parameters and regressors. n(t) represents the ARIMA part of the equation that models the errors and z(t) is the white noise process.

$$
\begin{aligned}
\text{PM10}(t) = 0.7758^{*}\text{PM10}(t-1) - 0.0963^{*}\text{Rainfall}(t-1) + \\
0.3333 * \text{Temperature}(t-1) + \\
-0.2965^{*}\text{Temperature}(t-4) + 0.8453^{*}\text{WindDirection} \\
(t-1) + 0.1276^{*}\text{PM10}(t-3) - 0.1088^{*}\text{PM10}(t-2) \\
+0.6359^{*}\text{WindDirection}(t-2) + n(t) \\
n(t) = 0.5733^{*}n(t-1) - 0.8181^{*}z(t-1) + z(t)
\end{aligned}
\tag{5.1}
$$

Moggio	PM10(t-1)	Rainfall(t-1)	Temperature(t-1)	Temperature(t-4)	WindDirection(t-1)	PM10(t-3)
Standard error	0.0586	0.0156	0.0654	0.0632	0.2480	0.0321

Table 5.6: Standard errors for the coefficients of the regressors of Moggio's baseline linear model.

$$
\begin{aligned}
\text{PM10}(t) = 25.8472 + 0.6955 * \text{PM10}(t-1) + -0.6470 * \text{Temperature}(t-2) - \\
2.4167^{*}\text{WindSpeed(t-1)} \ -0.1881^{*}\text{Rainfall(t-1)} \ - 0.1181^{*}\text{PM2.5}(t-2) + 0.1965^{*}\text{PM2.5}(t-3) \\
-1.5642^{*}\text{WindSpeed}(t-3) + z(t)
\end{aligned}
\tag{5.2}
$$

Cremona	Intercept	PM10(t-1)	Temperature(t-2)	WindSpeed(t–1)	Rainfall(t-1)	PM2.5(t-2)	PM2.5(t-3)	WindSpeed(t–3)
Standard error	1.6057	0.0248	0.0511	0.7216	0.0487	0.0290	0.0621	0.7053

Table 5.7: Standard errors for the coefficients of the regressors of Cremona's baseline linear model. The estimated parameters for the ARIMA part of Cremona's model are all null, therefore it has no ARIMA errors part.

$$
\begin{aligned}
\text{PM10}(t) = 21.1378 + 0.6013^{*}\text{PM10}(t-1) - 0.3739^{*}\text{Temperature}(t-2) \\
-1.4857^{*}\text{WindSpeed}(t-1) - 0.2732 * \text{Rainfall}(t-1) + n(t) \\
n(t) = -0.8232^{*}n(t-1) + 0.0024^{*}n(t-2) + 0.8084^{*}z(t-1) + z(t)
\end{aligned}
\tag{5.3}
$$

Schivenolgia	Intercept	PM10(t-1)	Temperature(t-2)	WindSpeed(t–1)	Rainfall(t-1)	n(t-1)	n(t-2)	z(t-1)
Standard error	19.072	0.0352	0.0519	0.4337	0.0855	0.3139	0.0499	0.3019

Table 5.8: Standard errors for the coefficients of the regressors of Schivenoglia's baseline linear model.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Moggio	7.530.473	5.779.235	0.4769976	7.246.967	5.459.332	0.2764452
Cremona	12.8	9.870.085	0.5872086	1.138.417	8.390.027	0.6575667
Schivenoglia	1.325.567	1.005.153	0.3929744	1.270.738	9.378.591	0.5348965

Table 5.9: Performance metrics for the forecast of the baseline linear regression models with ARIMA errors for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso.

As can be seen in table 5.9, in most cases the performance is on par with that of the baseline neural network and it tends to surpass that of the baseline persistence model.

5.3 Systematic construction of the deep learning models

Having made a naive baseline model and a neural network without any tuning, the next step is to proceed by optimizing the latter with the objective of achieving better predictions. There's a series of different modeling choices that can have a profound impact on performance: in this chapter, the focus will be on modifying the input data and tuning the hyperparameters.

5.3.1 Dealing with missing values in the dataset

As seen in Chapter 4, there are different periods across the years covered by the datasets, during which no information was measured for some of the variables. This information gap can potentially disturb the autocorrelation through time presented by different pollutants, which –depending on their length and the nature of the variable– can limit or contaminate the learning process of LSTM models.

While there's no in-built mechanism in traditional LSTM cells to natively approach this problem, the different alternatives presented in Chapter 2.9 can be evaluated. The first of which, simply removing the gaps and accepting the resulting disruption to the autocorrelations through time, was already used in the baseline model.

For all of the following model proposals, the architecture and data fed to the network will remain the same as the one described in the base neural network model.

5.3.1.1 Using out-of-bounds values and masking

By replacing all of the missing values in the Moggio Loc Penscei dataset with a number that falls outside of the scaling bounds applied on the dataset, the network could learn to interpret it as a marker for unreliable or non-usable data.

Considering that the scaled data spans from zero to one, a good candidate could be the value minus one. However, the network fails to learn its role, and instead continues to rely mostly on the previous timestep of the target feature to inform its prediction. For instance, in the prediction for the year 2019 (figure 5.5), whenever there are missing values and, therefore, minus one values replacing them, it interprets them as actual observations. Hence, in those

cases the model outputs values close to minus one that get scaled back –due to the MinMax scaling strategy– to negative concentrations of PM10, which are both wrong as predictions and physically impossible.

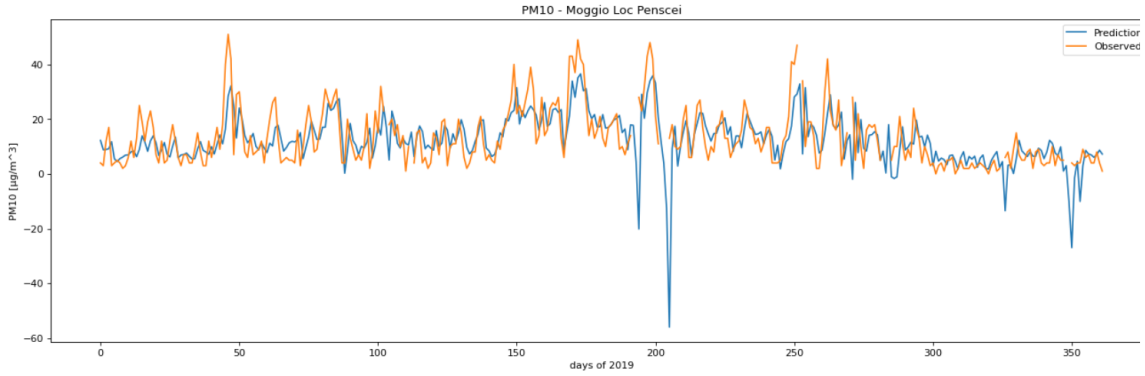

Figure 5.5: Moggio Loc Penscei's forecast for the year 2019 when using minus one as an out-of-bounds value fill in for missing observations.

Nearly identical results are obtained when using a masking layer (see chapter 2.9.1.3) as the input layer to the network: the model fails to interpret the special value as a token that represents missing information (Figure 5.6).

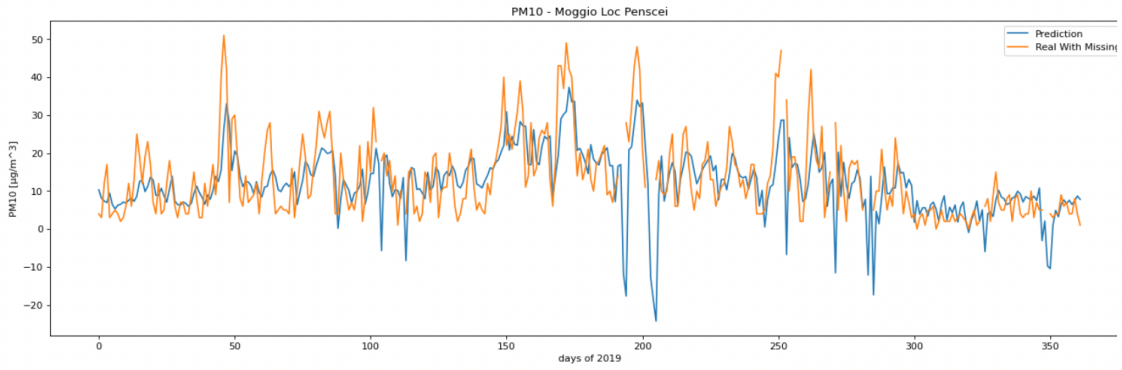

Figure 5.6: Moggio Loc Penscei's forecast for the year 2019 when using minus one as a out-of-bounds value fill in for missing observations along with a masking layer in the neural network.

5.3.1.2 Using informative values

When, instead of using an out-of-bounds values, a more informative value is used to replace missing data-points, the behavior is decidedly more realistic. This more informative value

can be, for instance, the mean of the distribution of each respective feature. Alternatively, interpolation methods can be used in order to preserve the local trend of the data. Figure 5.7, Figure 5.8 and Figure 5.9 show the prediction plots when using the mean, linear interpolation and third order spline interpolation respectively.

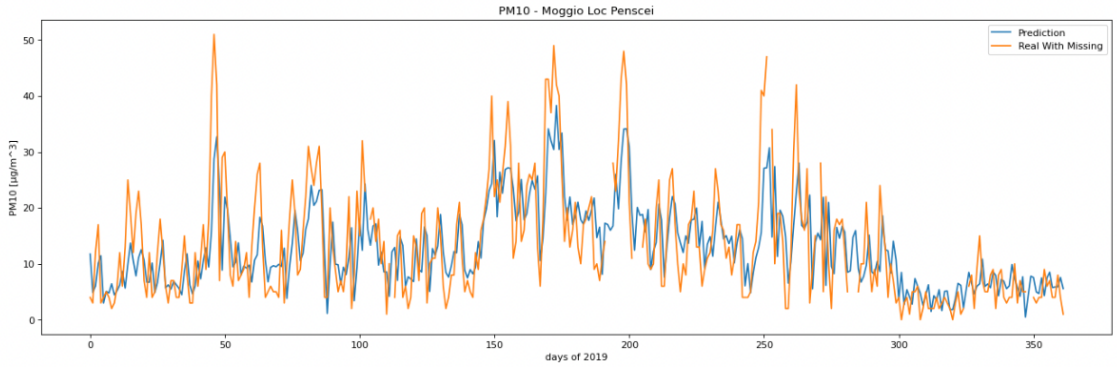

Figure 5.7: Moggio Loc Penscei's forecast for the year 2019 when using the mean of the distribution of each feature to fill in for their respective missing observations.

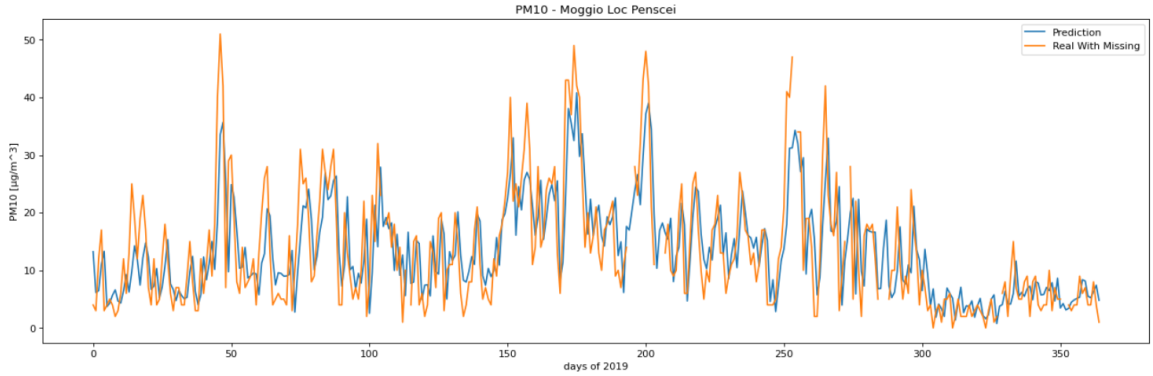

Figure 5.8: Moggio Loc Penscei's forecast for the year 2019 when using linear interpolation to fill in for their respective missing observations.

In all three cases, the predictions when there are missing values in the target feature rely on the data provided by the interpolation method. This behavior isn't fundamentally different from the one presented when using minus one as the value to replace the missing information: in all four cases the model blindly trusts the fabricated values by applying the same prediction mechanism in which it mostly uses the previous timestep. This, however, positions the use of interpolation as the more appropriate solution. Provided that the gaps caused by missing

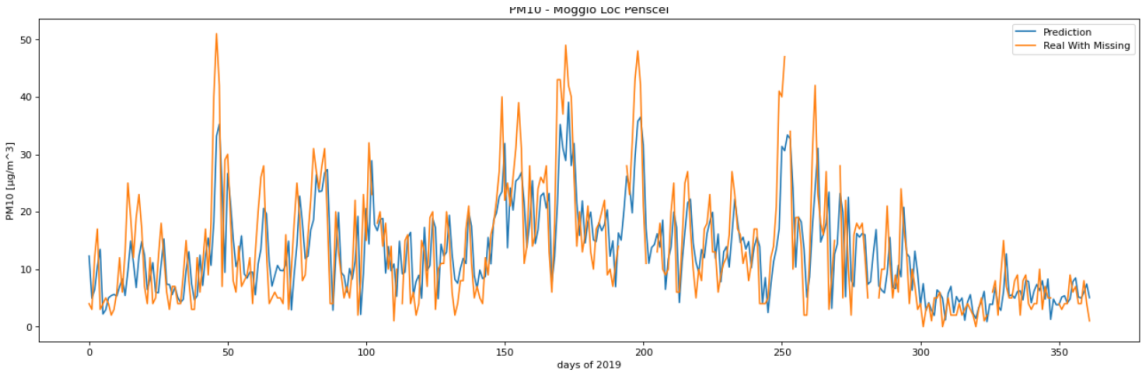

Figure 5.9: Moggio Loc Penscei's forecast for the year 2019 when using third-order spline interpolation to fill in for their respective missing observations.

values aren't too long, using interpolation can give an inkling as to what the time series's trend is at a given point it time, both for the target feature and all of its covariates.

While the neural network does seem to use information provided by other features other than the target feature itself in the timesteps that precede the forecast, their role is mostly secondary, as will be further discussed in chapter 5.5.

This behavior, while more sophisticated, is not too different from the one presented by the baseline persistence model. A reasonable next step is the revaluation of the covariates used in the model, especially the timespan of information associated to each of them, that is provided for each prediction.

5.3.2 Covariate modification

The baseline neural network model uses the information of five consecutive days of all features, including the target feature, to inform the prediction of the sixth day for the target feature itself. However, given that the goal is to forecast the value of just one of the features, instead of all of them with the same model and at the same time, it is also possible to use the values of all the features except the target feature itself on the day in which it is being forecast.

By modifying the model such that, for the forecast of a given time step t, the information of all covariates since time step t-5 up to time step t itself, for a total of six time steps, is used –with the exception of the target feature, for which the data at time step t isn't used given that it is the time step to be forecast– it is possible to provide more information to the model that has the potential of being much more relevant to the prediction at hand.

Using the data coming from at the Moggio Loc Penscei air quality station, the new training, validation and testing set dimensions are 1391 x 6 x 7 , 314 x 6 x 7 and 322 x 6 x 7, in which the only difference is that they now reflect a total of 6 time steps per input to the neural network instead of five and the. To respect the matrix dimensions, there's a column that would technically represent the values of the target feature at time step t, which is exactly what the model is built to predict. However, these values are all set to zero, with the expectation that, much like in the case of missing values, the model will learn to ignore them given that they provide no predictive insights.

The only changes made to the network's architecture are the ones relevant to the new input dimensions. Consequently the input shape of the networks goes from being 5 x 7, meaning five consecutive observations for seven different features, to being 6 x 7, given that the number of observations in input increases by one.

With all other hyperparameters being equal (see table 5.4), once the model finishes training, it scores a training MAE of 0.0321 and a validation MAE of 0.0243 after a total of 463 epochs. Its forecast for the validation set –the year 2019– is presented in figure 5.10, with an RMSE of 3.150, a MAE of 2.182 and a R2 of 0.909.

Whereas for the year 2020 (figure 5.11), which constitutes the testing set, it scores an RMSE 3.344, a MAE of 2.085 and a R2 of 0.846.

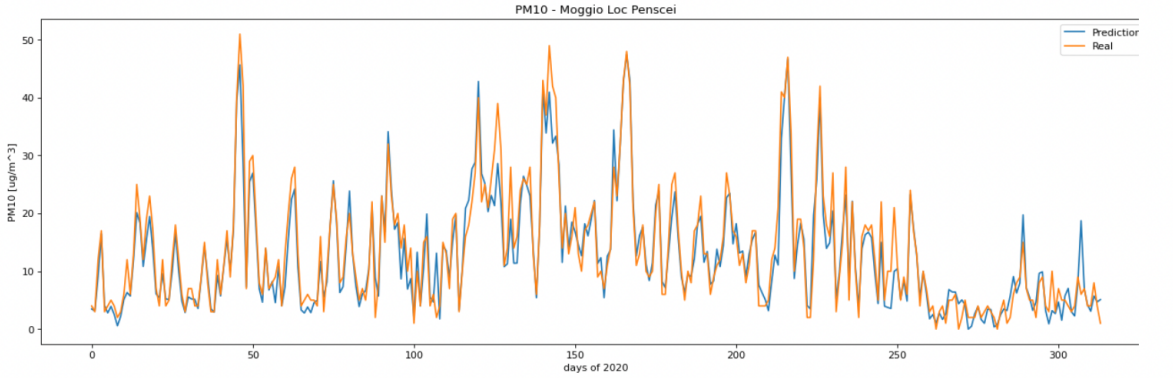

Figure 5.10: Forecast for 2019 for the Moggio Loc Penscei's station after updating the features that inform the forecast to contain information up to the very instant that is being predicted.

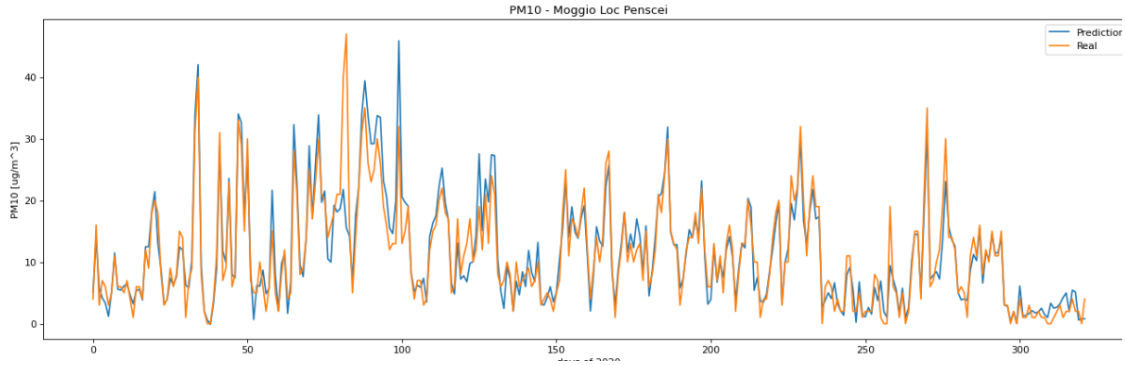

Figure 5.11: Forecast for 2020 for the Moggio Loc Penscei's station after updating the features that inform the forecast to contain information up to the very instant that is being predicted.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Neural network baseline	7.222	5.451	0.519	6.874	5.191	0.349
Linear model baseline	7.530	5.779	0.4769	7.246	5.459	0.2764
Modified covariates	3.150	2.182	0.909	3.344	2.085	0.846

Table 5.10: Performance metrics comparison between the Moggio Loc Penscei 2020 forecasts made by the baseline neural network model, which uses the data of the prior five days for each prediction, and the updated model that also uses the covariates' information of the very day of the prediction.

When compared to the performance of the baseline neural network, the metrics improve across the board. Moreover, upon inspection of the prediction plots, it would be reasonable to assume that the new model does not rely quite as heavily on the value of the target feature at the previous time step: when compared to the observed values, unlike in the baseline neural network and the baseline persistence model, the predicted values do not seem to simply output

the observed trajectory shifted one step into the future. Instead, the predictions match the trend and punctual values of the original data at the same instant much more frequently, presumably due to the new information that is now available to the model.

By using a local surrogate model, as shown in Chapter 5.5 using LIME, it is possible to compare the choices of this model with respect to the baseline neural network more precisely and quantitatively. The same modifications can be applied to all other stations. Table 5.11 lists the results.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Neural network baseline Moggio	7.222	5.451	0.519	6.874	5.191	0.349
Linear model baseline Moggio	7.530	5.779	0.4769	7.246	5.459	0.2764
Modified covariatesMoggio	3.150	2.182	0.909	3.344	2.085	0.846
Neural network baseline Cremona	12.405	9.299	0.612	11.102	7.929	0.674
Linear model baseline Cremona	12.800	9.870	0.587	11.384	8.390	0.657
Modified covariates Cremona	5.041	3.723	0.936	6.377	3.513	0.893
Neural network baseline Schivenoglia	12.834	9.847	0.431	13.043	9.528	0.510
Linear model baseline Schivenoglia	13.255	10.051	0.392	12.707	9.378	0.534
Modified covariates Schivenoglia	4.671	3.161	0.925	7.145	3.759	0.853

Table 5.11: Performance metrics comparison between the 2020 forecasts made by the baseline neural network model, which uses the data of the prior five days for each prediction, and the updated model that also uses the covariates' information of the very day of the prediction.

When applying the three different missing values strategies to this type of model, it behaves in much the same way: the use of out-of-bounds values to replace missing values fails to be applicable in this scenario, in fact, with or without masking, the networks does not learn to identify the missing data points and instead treats them in the same it would treat normal observation. The use of interpolation, on the other hand, remains the more robust solution, improving the performance of the forecast when compared to the results presented above, in which days with missing values are simply removed.

Technique	Validation RMSE	Validation MAE	Validation R2	Test RMSE	Test MAE	Test R2
Missing values removed	3.150	2.182	0.909	3.344	2.085	0.846
Replacing missing values with the mean	3.662	2.645	0.871	3.538	2.347	0.821
Linear interpolation	3.385	2.486	0.893	3.339	2.314	0.843
Third order spline interpolation	3.177	2.308	0.908	3.652	2.364	0.826

Table 5.12: Performance metrics comparison for the Moggio Loc Penscei 2020 forecasts made using different techniques to fill in missing observations in the dataset.

All of the alternatives yield similar results, which seem to perform ever so slightly worse than the original, simpler approach (table 5.12).

It is, however, important to note that, when calculating any of the metrics in table 5.12, the predicted value in any given time step with a missing observation is compared to the value that replaces the latter. For instance, when the mean of a feature is used to replace its missing values, the predicted value in those time steps will always be compared to the mean, which contains no information about the local trend of the data and also explains why interpolation techniques, that are better suited to simulate the trend of the missing values, tend to perform better.

In any case, the impossibility of comparing any prediction with the observed data, whenever that data is missing, means that the evaluation of the model's performance in those specific instances cannot be, by any means, as precise as it would be otherwise. The modest drop in performance with any of the replacement techniques in table 5.12 could therefore be considered to not be enough of a reason to completely disregard them.

On the other hand, these techniques create the possibility of evaluating the addition of other features to the model that may present longer periods of time with no observations, but that, whenever the data is actually available, may better inform the prediction made by the model.

5.4 Optimization of the model through hyperparameter exploration

Having a more resilient way to deal with missing data points, that does not imply discarding an entire day's worth of observations whenever there's at least one feature missing, means that other weather and air quality variables –that had been originally dismissed due to significant amounts of missing values– can now be reintroduced in hopes of further improving the performance and accuracy of the predictions.

Additionally, all previous models use hyperparameters that have not been optimized and optimizing them with any of the techniques mentioned in chapter 2.10 can lead to better forecasts as well as being helpful to find the simpler and quicker-to-train models amongst the more performant ones.

To do so, it is therefore appropriate to use a hyperparameter optimization technique, such as Random Search (see Chapter 2.10.0.2), that chooses random combinations of hyperparameters from the set that is being explored and evaluates the performance of the resulting models. The hyperparameters used in all previous models are listed in table 8. These are expanded upon by defining new sets to be explored:

- **Hidden layers**: The amount of layers between the input and output layers is not fixed to one anymore, it instead can range from one to three, the set is therefore 3,4,5.

- **Neurons per hidden layer**: The amount of cells in each hidden layer, for each iteration of the Random Search algorithms, is chosen from the set 10, 20, 30, 40, 50, whereas it was previously statically set to 10.

- **Batch size**: The batch size used during the training of the model, which determines the amount of observations to be propagated through the neural network at each iteration and based on which its weights and biases are tuned, is chosen from the set 4 ,8 ,16, 32, 64, 128, 256.

- **Learning rate**: As the name suggests, it determines the step size used by the optimizer

of choice, the Adam optimizer (see chapter 2.4), to use in an attempt to get closer to a minimum in each successive iteration. It is chosen from the set 1e-1, 1e-2, 1e-3, 1e-4.

- **Maximum number of epochs**: It is doubled to 1000 epochs. The same Early stopping mechanism is in place that stops the training after 25 epochs with little to no improvement.

With the objective of further improving predictive performance, the Random Search algorithm was used on all three datasets coming from the three stations, therefore exploring the different possible models that come as a result of combinations of hyperparameters from the sets mentioned above.

A total of 50 randomly chosen combinations were trained for each station. All features remain the same and the objective remains to forecast the concentration of PM10. As for the missing observations present in the dataset, linear interpolation was used: as seen in chapter 5.3.1, its performance is among the best among all other techniques, and it also has the advantage of maintaining the trend of the data, while being significantly less computationally expensive compared to other interpolation methods.

Among all of these, the results are mixed, with certains combinations that perform much worse than the previous model for each station, and others that manage to achieve even better results. The full list of results is available in the appendix from table 6.1 to table 6.12.

The three best performing models in terms of testing R2 that were trained on the data from the Moggio Loc Penscei air quality station use the hyperparameters in table 5.13. For the forecast of the testing set, the absolute best model achieves an RMSE of 2.955, a MAE of 1.899 and a R2 of 0.877, after 268 epochs of training as seen in table 5.14 .

The performance of these models are presented in table 5.14. All three manage to provide improvements with respect to previous models.

There are no specific rules to determine exactly what combination of hyperparameters will work best in a particular scenario or for a particular dataset. Case in point, all three models in table X and table Y use the same data and their objective is the same: to forecast the concentration of PM10, and yet they use very disparate sets of hyperparameters. Similar results can be achieved with the data of the Schivenoglia Via Malpasso and Cremona Via

Model	Hidden layers	Neurons per hidden layer	batch size	learning rate	Effective number of epochs
Baseline	1	10	32	0.001	82
Modified covariates	1	10	32	0.001	463
Optimized model 1	3	50	8	0.0001	268
Optimized model 2	4	20	128	0.01	81
Optimized model 3	3	40	32	0.0001	473

Table 5.13: Performance metrics comparison between different competing Moggio Loc Penscei models. The optimized models use three best performing hyperparameter combinations among the 50 that were evaluated through the Random Search algorithm.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Baseline	7.222	5.451	0.519	6.874	5.191	0.349
Modified covariates	3.150	2.182	0.909	3.344	2.085	0.846
Optimized model 1	3.041	2.294	0.914	2.955	1.899	0.877
Optimized model 2	3.082	2.221	0.911	2.972	2.036	0.875
Optimized model 3	2.975	2.204	0.917	3.047	2.009	0.868

Table 5.14: Performance metrics comparison between different competing Moggio Loc Penscei models. The optimized models use three best performing hyperparameter combinations among the 50 that were evaluated through the Random Search algorithm.

Fatebenefratelli stations n the appendix from table 6.1 to table 6.12.

Moreover, it is possible to model the importance of each hyperparameter when it comes to optimizing a given metric. The Weights and Biases service, which provides a framework to compare and gather the data of competing models, automatically trains a Random Forest model (see chapter 3.1) that determines the role that each hyperparameter plays. In fact Random Forest models are appropriate for the task because, unlike neural networks, they are both intrinsically explainable, because the reasoning behind any prediction is transparent to the user, and much faster to train.

In the case of Moggio Loc Penscei, when optimizing for the highest possible testing R2, which means the best possible R2 performance using the validation set which contains the data gathered in 2019, the hyperparameter importance is presented in figure 5.12.

The most relevant hyperparameter is the learning rate. This comes as no surprise, because it tends to be a very impactful one in all sorts of neural networks. In fact, since it determines the step size to be used during the optimization process in the transition between a training iteration and the next, it can determine whether the learning process might stall, due to a very small learning rate that hinders progress towards a minimum, or whether it might cause the model to converge too quickly to a local minimum that yields suboptimal results.

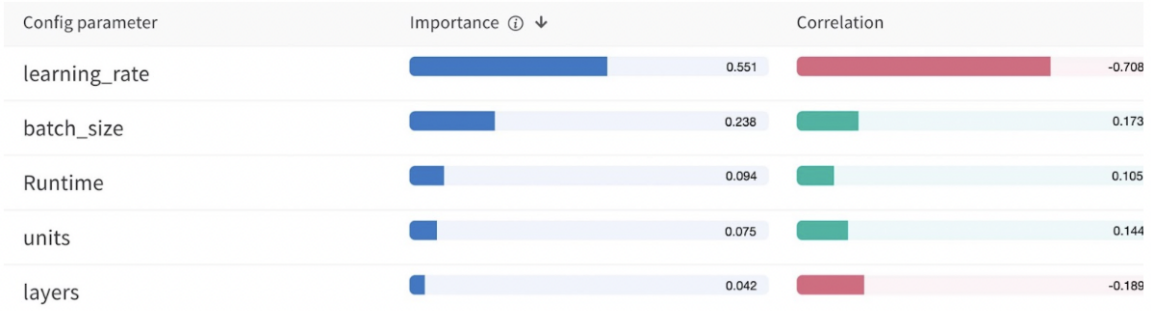

Figure 5.12: Ranking that represents the importance of the different hyperparameters towards the optimization of the testing R2 with respect to its correlation to hyperparameters' values and the feature importance results of a Random Forest model.

What is more, the correlation between the value of the testing R2 and the learning rate is -0.706 (figure 5.12), which suggests that, among the 50 different models, the ones that tend to have smaller –or rather, slower– learning rates tend to have a higher R2. Therefore, it could be argued that a more parsimonious approach to the learning process is more likely to yield better results in this scenario.

The second most important feature, as determined by the feature importance of the Random Forest model, is the batch size, which, having a weak, yet positive correlation with the testing R2, suggests that bigger batch sizes are more conducive to higher R2 metrics.

The runtime, which indicates the amount of time it takes to train the model, is in part determined by the learning rate, which in turn dictates the rate at which the training takes place. It presents a positive correlation with regards to the testing R2, which is consistent with the negative correlation between the learning rate and the R2: a smaller learning rate can lead to longer training runtimes and vice versa.

Finally, the units, or number of neurons per layer, and the amount of hidden layers itself are the two least impactful hyperparameters. The correlations shown in figure 5.12) suggest that an approach with fewer layers and more neurons tends to work best, but, according to the feature importance results of the Random Forest model, their impact is smaller than the one associated with the batch size and especially the learning rate.

With this in mind, looking back to the three best models for Moggio Loc Penscei in table 5.13, it is clear that the correlation mentioned above are not unbendable rules: for instance while

model 1 presents, as would be expected, a number of layers and a learning rate on the lower end of their respective exploration sets, its batch size is also on the lower end and not the other way around; similarly, model 2 runs counter to the norm by using a learning rate on the higher end and a number of neurons on the lower end.

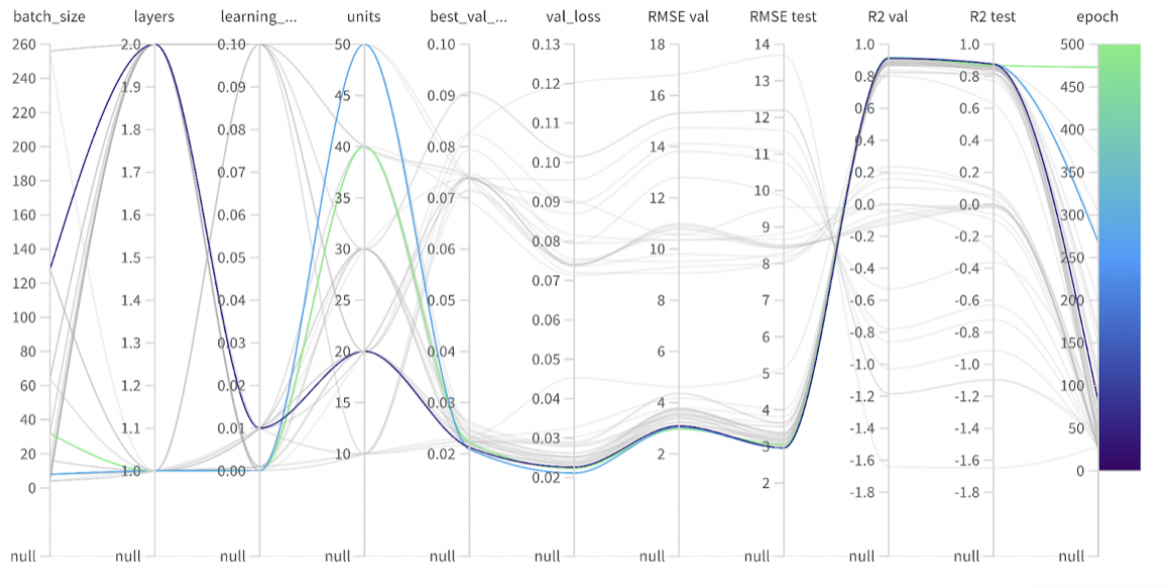

Figure 5.13: Parallel coordinate plot that shows all the 50 combinations of hyperparameters associated to the resulting RMSE and R2 metrics. The best performing model in terms of testing R2 for each station is highlighted, the rest are grayed out.

Figure 5.13 highlights the three different models presented in table 11 in three different colors, whereas the remaining 47 competing models are grayed out in the background. It shows the heterogeneity of the different combinations considered by the Random Search algorithm and helps illustrate the fact that, even knowing the results of the hyperparameter importance analysis, different combinations of hyperparameters can behave in highly unexpected ways.

5.4.1 Performance analysis with an extended feature set

All three stations gather information on air quality pollutants that go beyond the protagonists of this thesis: particulate matter and ammonia. These variables, however, are not always the same across the different stations and, by increasing the amount of features to pre-process before the modeling phase, the problems associated with the absence of observations can become more severe.

With that being said, just like before, linear interpolation can be used to address concerns related to missing observations and, in doing so, the potential contribution of these variables can be explored.

When it comes to the Cremona Via Fatebenefratelli in addition to the NH3, PM2.5 and PM10, the station also logs concentrations of Ozone, NO2, NOx CO and Sulfur Dioxide. The Schivenoglia Via Malpasso extends this set of pollutants by also measuring Arsenic, Benzene, Benzopyrene, Cadmium, Lead and Nickel, and, finally, the Schivenoglia Via Malpasso provides even more additional information, by also tracking the concentrations of NO. Furthermore, their closest weather stations are not limited to wind speed, wind direction, temperature and rainfall, because they also measure relative humidity and global radiation.

Executing the Random Search algorithm in much the same way as before, while, of course, using the new, different features available for the different stations, it is possible to determine whether this new information can actually improve the forecasting capabilities of the models. In fact, by taking into account dozens of different hyperparameter combinations, the likelihood of any differences in performance between the previous models and any new ones being simply attributed to inappropriate hyperparameter tuning is greatly reduced.

Despite the seemingly high potential of improvement, the actual performance metrics (table 5.15) of the best performance models across the different stations show no significant benefit associated with the introduction of these new features.

The feature set limited to PM2.5, NH3, wind speed, wind direction, temperature and rainfall is informative enough to predict PM10 to the point that adding the remaining variables made available by the stations as features for the model adds virtually no extra performance. This, however, may not come as a surprise: after all, PM2.5, which, by definition, is a subset

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Limited Moggio	3.041	2.294	0.914	2.955	1.899	0.877
Extended Moggio	3.056	2.232	0.912	3.014	2.051	0.872
Limited Cremona	5.052	3.730	0.933	6.524	3.966	0.899
Extended Cremona	4.904	4.013	0.937	6.555	4.213	0.898
Limited Schivenoglia	5.297	3.637	0.905	6.633	4.221	0.868

Table 5.15: Performance metrics comparison between the PM10 forecasts for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso when using a limited set of features composed of PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself, compared to when all the variables available to each station are used as features.

of PM10, is used for the prediction and the correlation between the two, as seen in chapter 4.5, is very strong: the model could be basing most of its prediction of PM10 on the value of PM2.5 at the same instant.

This begs the question: how important is PM2.5 to the prediction of PM10? The relevance of this and other features is discussed in detail in chapter 5.5, in which a surrogate linear model is used to help shed light on the issue. On the other hand, hyperparameter optimization can also help compare the performance when PM2.5 is used and not used as a feature.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Extended Moggio	3.056	2.232	0.912	3.014	2.051	0.872
Extended Moggio, no PM2.5	5.652	4.241	0.701	5.254	3.921	0.611
Extended Cremona	4.904	4.013	0.937	6.555	4.213	0.898
Extended Cremona, no PM2.5	7.801	6.159	0.841	8.031	5.908	0.848
Extended Schivenoglia	5.441	4.181	0.899	7.319	4.348	0.839
Extended Schivenoglia, no PM2.5	9.392	6.690	0.701	9.856	6.838	0.708

Table 5.16: Performance metrics comparison between the PM10 forecasts for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso when using and not using PM2.5 among the covariates, considering that all the other variables available to each station are used as features.

In table 5.16, the metrics of models that use all variables made available by the station are compared to models that use all variables but PM2.5. The results suggest that, in all three stations, removing PM2.5 results in a loss in predictive capabilities.

An even bigger loss of performance is measured when comparing the best models with the original set of features – PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself– and the ones in which PM2.5 is removed (Table 5.17). Therefore, while the addition of the extra features provides little to no benefit when the concentration of PM2.5 is

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Limited Moggio	3.041	2.294	0.914	2.955	1.899	0.877
Limited Moggio, no PM2.5	6.259	4.665	0.634	4.939	3.726	0.655
Limited Cremona	5.052	3.730	0.933	6.524	3.966	0.899
Limited Cremona, no PM2.5	9.689	7.276	0.755	9.227	6.741	0.799
Limited Schivenoglia	5.297	3.637	0.905	6.633	4.221	0.868
Limited Schivenoglia, no PM2.5	9.479	7.044	0.696	10.076	7.108	0.697

Table 5.17: Performance metrics comparison between the PM10 forecasts for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso when using and not using PM2.5 among the covariates, when the rest of the feature seti is composed of NH3, wind speed, wind direction, temperature, rainfall and PM10 itself.

available, the information they convey can still be beneficial in its absence: when comparing any two models in table 5.16 and table 5.17 among the ones that do not use PM2.5, the extended versions that use all features available tend to achieve better results.

What is perhaps more noteworthy is the fact that, as show in table 5.18, the introduction of a linear regression models with ARIMA errors that starts from the same limited feature set, built in the same way as the models in Chapter 5.2.3, performs in much the same way as the neural network models, regardless of whether they use the limited or extended set. This result suggests that the non-linearity capabilities of LSTM neural networks fail to be of any use for PM10 forecasts that use these feature sets as regressors.

Model	validation RMSE	validation MAE	validation R2	test RMSE	test MAE	test R2
Limited Moggio	3.041	2.294	0.914	2.955	1.899	0.877
Linear model Moggio	2.879	2.089	0.922	2.947	1.988	0.877
Extended Moggio	3.056	2.232	0.912	3.014	2.051	0.872
Limited Cremona	5.052	3.730	0.933	6.524	3.966	0.899
Linear model Cremona	5.087	3.777	0.932	6.680	4.097	0.894
Extended Cremona	4.904	4.013	0.937	6.555	4.213	0.898
Limited Schivenoglia	5.297	3.637	0.905	6.633	4.221	0.868
Linear model Schivenoglia	491.931	3.184.683	0.9181646	6.823.322	3.895.711	0.8611572
Extended Schivenoglia	5.441	4.181	0.899	7.319	4.348	0.839

Table 5.18: Performance metrics comparison between the PM10 forecasts for the air quality stations located in Moggio Loc Penscei , Cremona Via Fatebenefratelli and Schivenoglia Via Malpasso when different types of models. Namely, the linear models use linear regression with ARIMA errors, the limited models use an LSTM neural network with the limited feature set from chapter 4, and the extended models use all features available to the station in question.

For context, Eq (5.4), Eq (5.5) and Eq (5.6) are the mathematical formulations of the above mentioned linear models.

$$
\begin{aligned}
\mathrm{PM10}(t) = 1.0463 * \mathrm{PM25}(t) + \\
0.1975 * \mathrm{Temperature}(t-1) + 0.2526 * \mathrm{PM10}(t-1) - \\
0.1908 * \mathrm{PM2.5}(t-1) - 0.1451 * \mathrm{PM2.5}(t-2) + 0.0407 * \mathrm{PM10}(t-3) + 0.0661 * \mathrm{PM10}(t-2) - \\
0.1084 * \mathrm{Temperature}(t-2) + 0.8773 * \mathrm{Ammonia}(t) - \\
0.7208 * \mathrm{Ammonia}(t-1) - 0.0147 * \mathrm{Rainfall}(t) + \\
0.0181 * \mathrm{PM10}(t-5) + n(t) \\
n(t) = -0.9887 * z(t-1) + z(t)
\end{aligned}
\tag{5.4}
$$

Moggio	PM25(t)	Temperature(t-1)	PM10(t-1)	PM2.5(t-1)	PM2.5(t-2)	PM10(t-3)	PM10(t-2)	Temperature(t-2)	Ammonia(t)	Ammonia(t-1)	Rainfall(t)	PM10(t-5)	z(t-1)
Standard error	0.0154	0.0495	0.0243	0.0301	0.0299	0.0116	0.0243	0.0496	0.1372	0.1380	0.0061	0.0091	0.0054

Table 5.19: Standard errors for the coefficients of the regressors of Moggio's linear model

$$
\begin{aligned}
\mathrm{PM10}(t) = 2.9963 + 0.2263 * \mathrm{PM10}(t-1) - \\
0.1978 * \mathrm{PM2.5}(t-1) + 1.0432 * \mathrm{PM2.5}(t) + \\
0.2888 * \mathrm{Ammonia}(t) - 0.9168 * \mathrm{WindSpeed}(t) - 0.0650 * \\
\mathrm{Rainfall}(t) + 0.2915 * \mathrm{WindDirection}(t-1) - \\
0.1205 * \mathrm{Ammonia}(t-1) + 0.0690 * \mathrm{Temperature}(t-1) + \\
0.0392 * \mathrm{Ammonia}(t-4) - 0.5087 * \mathrm{WindSpeed}(t-4) \\
+ n(t) \\
n(t) = 0.9281 * n(t-1) - 0.8592 * z(t-1) - 0.0105 * z(t-2) = z(t)
\end{aligned}
\tag{5.5}
$$

Cremona	intercept	PM10(t-1)	PM2.5(t-1)	PM2.5(t)	Ammonia(t)	WindSpeed(t)	Rainfall(t)	WindDirection(t-1)	Ammonia(t-1)	Temperature(t-1)	Ammonia(t-4)	WindSpeed(t-4)	n(t-1)	z(t-1)	z(t-2)
Standard error	0.0462	0.1290	0.0777	0.7955	0.1011	0.1112	0.0121	0.0369	0.2568	0.0161	0.0911	0.0455	0.0304	0.0261	0.2499

Table 5.20: Standard errors for the coefficients of the regressors of Cremona's linear model

$$\begin{aligned}
\mathrm{PM10}(t) = 2.3372 + 1.0470 * \mathrm{PM2.5}(t) + \\
0.5693 * \mathrm{PM10}(t-1) - 0.5485 * \mathrm{PM2.5}(t-1) + \\
0.2096 * \mathrm{Temperature}(t) + \\
0.0806 * \mathrm{PM10}(t-5) - \\
0.0760 * \mathrm{PM2.5}(t-5) - 0.7234 * \mathrm{WindSpeed}(t) - \\
0.1220 * \mathrm{Temperature}(t-5) - 0.1078 * \mathrm{Rainfall}(t-5) - \\
0.0499 * \mathrm{PM2.5}(t-4) + n(t) \\
n(t) = 1.1808 * n(t-1) - 0.3462 * n(t-2) - 1.4426 * z(t-1) + 0.5762 * z(t-2) + z(t)
\end{aligned} \tag{5.6}$$

Schivenolgia	intercept	PM2.5(t)	PM10(t-1)	PM2.5(t-1)	Temperature(t)	PM10(t-5)	PM2.5(t-5)	WindSpeed(t)	Temperature(t-5)	Rainfall(t-5)	PM2.5(t-4)	n(t-1)	n(t-2)	z(t-1)	z(t-2)
Standard error	0.8347	0.0226	0.1600	0.1857	0.0637	0.0345	0.0424	0.2489	0.0425	0.0424	0.0234	0.2741	0.1756	0.1431	0.1204

Table 5.21: Standard errors for the coefficients of the regressors of Schivenoglia's linear model

5.5 Interpretation of results through LIME surrogate linear models

While comparing models that use different feature sets offers an opportunity to hypothesize as to why certain ones perform better or make different predictions, it would be necessary to build a big amount of different versions of the model in order to derive unambiguous explanations from them.

On the other hand, the use of explainability techniques, such as LIME (see chapter 2.11.1.1), that are applicable to black-box models, such as the LSTM neural networks used for all models, can more elegantly provide clearer insight

as to which features are the most important in different circumstances.

These explainability techniques, however, only provide explanations with local-fidelity: that is, the information they provide is only valid for one predetermined prediction, and it alone cannot be used to make assumptions that are valid for the entire model. After all, these techniques use linear models as surrogates to explain the behavior of the black-box model for a specific prediction, and considering that neural networks are nonlinear in nature, the linear model applied to one specific prediction is unable to capture all the different facets of the nonlinear behavior or the neural network.

LIME, in particular, uses the coefficients of a linear model to determine how relevant every feature is when determining a specific output of the model. For example, the prediction for august 10th, 2020 made by the Cremona's baseline neural network model is explained by LIME as shown in figure 5.14.

Based on the values of the coefficients of the linear model, it determines that the most datapoint for this specific prediction is the value of PM2.5 the day prior, the second most important would be Ammonia the day prior as well, and so on. The sign of the coefficients defines whether the data point contributed positively or negatively to the outcome: in this specific case the surrogate model suggests that while the values of PM2.5 and Ammonia the day before contribute positively, their values two days prior contribute negatively.

As has already been established, this explanation has local –not global– fidelity, therefore this

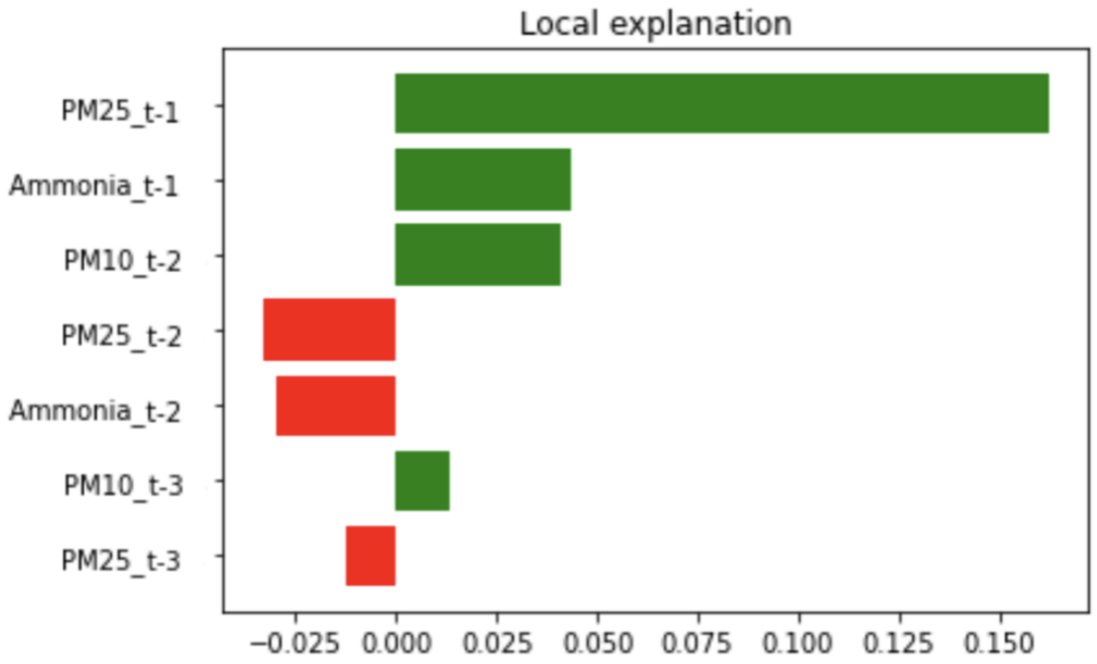

Figure 5.14: Feature importance chart for the prediction for august 10th, 2020 made by the Cremona's baseline neural network model is explained by LIME through a surrogate linear model.

result does not reflect the global behavior of the model. While determining which features are more important than others and under which conditions is not possible through a direct inspection of the model, counting the instances in which they occupy a specific position on the importance ranking created by LIME during a period of time can help determine which variables are the most important most often as described by a new ranking.

Specifically, the idea is to associate weights to each single feature equal to their place in the ranking, when the ranking itself is sorted from lowest to highest importance. This way the most important feature for the prediction of any given day gets the highest weight. The sum of these weights across the explanations associated with the predictions for every single day of a specific time period determines the new ranking.

The results for the baseline neural network models of all the predictions possible in 2020 are depicted in table 5.22.

For instance, looking at the results for Cremona Via Fatebenefratelli, graphically depicted

Variable	Moggio Loc Penscei	Cremona Via Fatebenefratelli	Schivenoglia Via Malpasso	test RMSE	test MAE	test R2
PM10	2049	2224	1899	2.955	1.899	0.877
Rainfall	2035	1516	1084	4.939	3.726	0.655
PM2.5	1622	1616	1343	6.524	3.966	0.899
Temperature	1365	1832	911	9.227	6.741	0.799
Wind direction	1102	114	1170	6.633	4.221	0.868
Ammonia	609	553	297	10.076	7.108	0.697

Table 5.22: Feature importance weights associated with each feature for each station, where the sum of weights is determined by adding up their place in the ranking for each prediction in the year, when the ranking itself is sorted from lowest to highest importance.

in figure 5.15, the most important feature across all predictions in 2020 turns out to be, unsurprisingly, PM10 itself. In fact, since the model uses the previous 5 days' worth of observations of each feature for every single prediction, it is reasonable that it would rely on the previous data of the target feature to approximate its future value. The third most relevant feature is PM2.5, which is also reasonable considering the strong correlation between it and PM10. The meteorological variables, such as the temperature and rainfall that occupy the second and fourth place in the ranking respectively, also play a big role, whereas, by comparison, the remaining features –Wind speed, Ammonia and especially the quadrant associated with direction of the wind– are deemed less useful for the prediction, as determined by the more pronounced drop in weights associated with these three features with respect to the other four.

It is however important to note that the nature of the model remains nonlinear, and therefore these results, unlike the coefficients of a linear model, cannot be interpreted as a representation of the choices of the model for any given input. As a matter of fact, this is further reinforced by the fact that these results are determined using surrogate models as proxies for explainability that, as explained in chapter 2.11.1.1, are built to compromise a high degree of precision in exchange for easier interpretability, which further increases the disconnect between the real LSTM model and this interpretation of its behavior.

These results are therefore to be considered as a reflection of the time-invariant aspects of the neural network's reasoning when it comes to forecasting, as opposed to a traditional model feature importance ranking with global fidelity.

Given that the model uses the observations of the previous five days with respect to the day of the prediction, it is also possible to increase the level of granularity by ranking each lag of

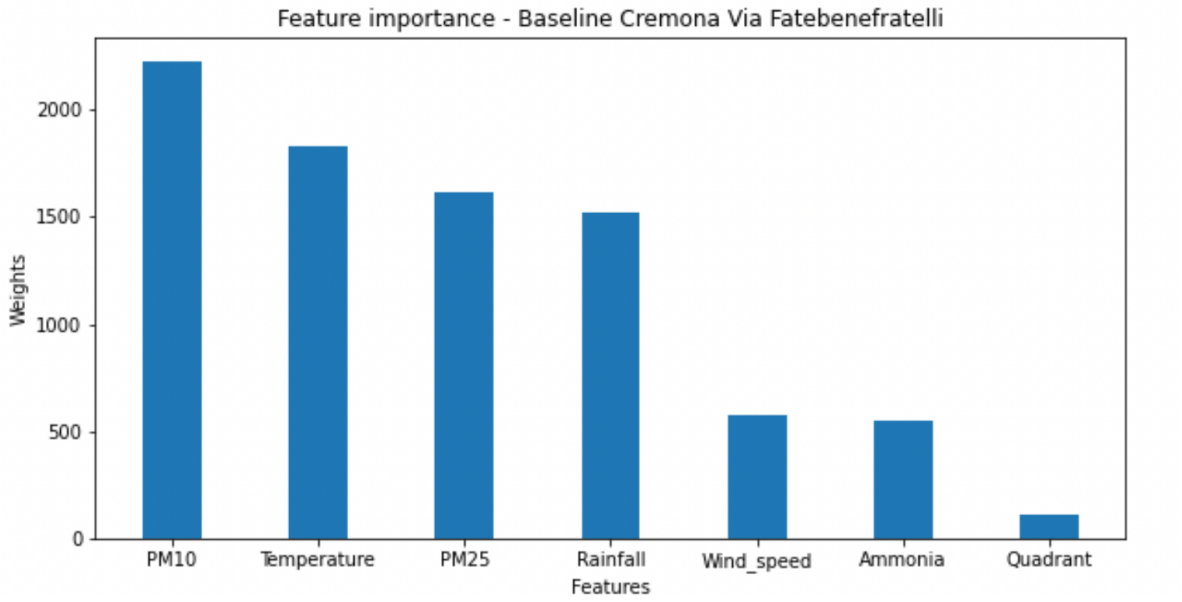

Figure 5.15: Feature importance ranking based on the predictions made by the Cremona's baseline neural network model throughout 2020. Each feature is assigned a sum of weights which is directly proportional to its importance. The sum of weights is determined by adding up their place in the ranking for each prediction in the year, when the ranking itself is sorted from lowest to highest importance.

each feature as an independent feature, instead of aggregating all of the weights of all lags of each feature. In figure 5.16, the twelve most relevant features-lag pairs are listed for the year 2020 in Cremona Via Fatebenefratelli. PM10, Rainfall, PM2.5 and Temperature remain the most relevant and, what is more, the data that is temporally closer to the prediction is deemed to be more informative, with the first five places all being from the day prior to the prediction. This supports the hypothesis that, as seen in Chapter 5.2.2, the baseline models tend to rely heavily on the previous day of observations.

Broadly speaking, when it comes to the Moggio Loc Penscei and Schivenoglia Via Malpasso, the behavior is similar. While the actual rankings are different (see table 6.13 in the appendix), PM10 and PM2.5 remain the most relevant pollutants and Ammonia remains among the least useful. Rainfall and Temperature still play an important role for Moggio's baseline model, while wind speed and direction gain more relevance, being the third and fourth most relevant feature respectively for Schivenoglia's model. Additionally, they both still favor data from the previous day of observation more heavily than the other data points.

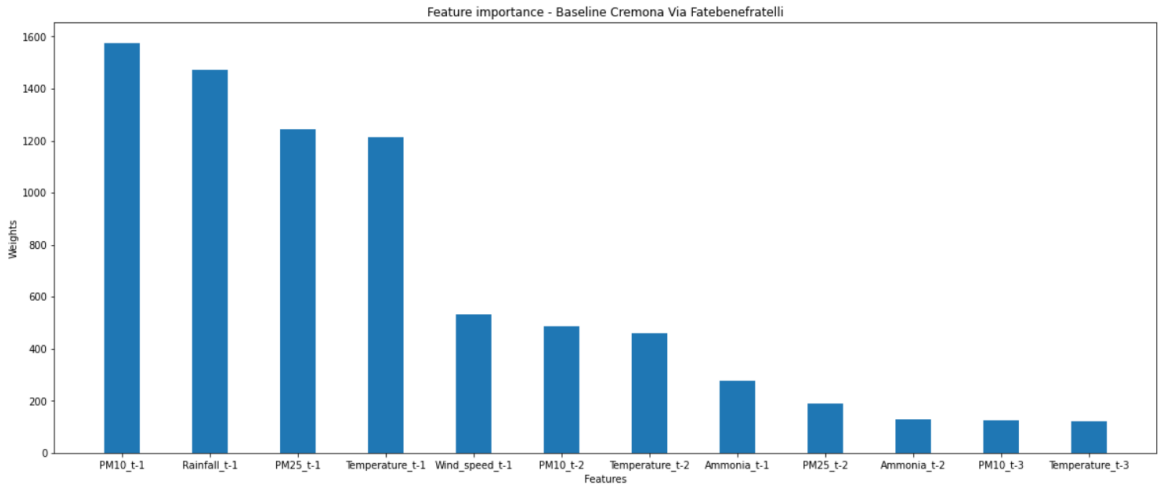

Figure 5.16: Feature importance ranking based on the predictions made by the Cremona's baseline neural network model throughout 2020. Each feature, at each time step, is assigned a sum of weights which is directly proportional to its importance. The sum of weights is determined by adding up their place in the ranking for each prediction in the year, when the ranking itself is sorted from lowest to highest importance.

The explainability results presented so far offer even more insight when compared to other model proposals from chapter 5.3 In fact, given that there is a noticeable difference in performance between the baseline and any of the optimized models, feature importance analysis can help construct more nuanced explanations as to why this is the case.

Figure fig:dang shows the feature importance results for Cremona Via Fatebenefratelli's best performing model using the limited feature set containing PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself. This model has access to the data of not only the 5 days prior to the prediction, but also of the day of the prediction itself with the exception of PM10, the target feature. By also taking into consideration Figure 5.18, that shows the most relevant features-lag pairs for the same station, it is possible to determine that the most informative piece of information is the concentration of PM2.5 on the very same day of the prediction, followed by the concentrations of PM2.5 and PM10 in the previous three days,alongside certain meteorological variable such as the amount of rainfall the for the day of the prediction and the temperature of the day prior.

Interestingly, according to the model, the contribution of the concentration of Ammonia to the prediction is roughly tantamount to that of the amount of rainfall and temperature.

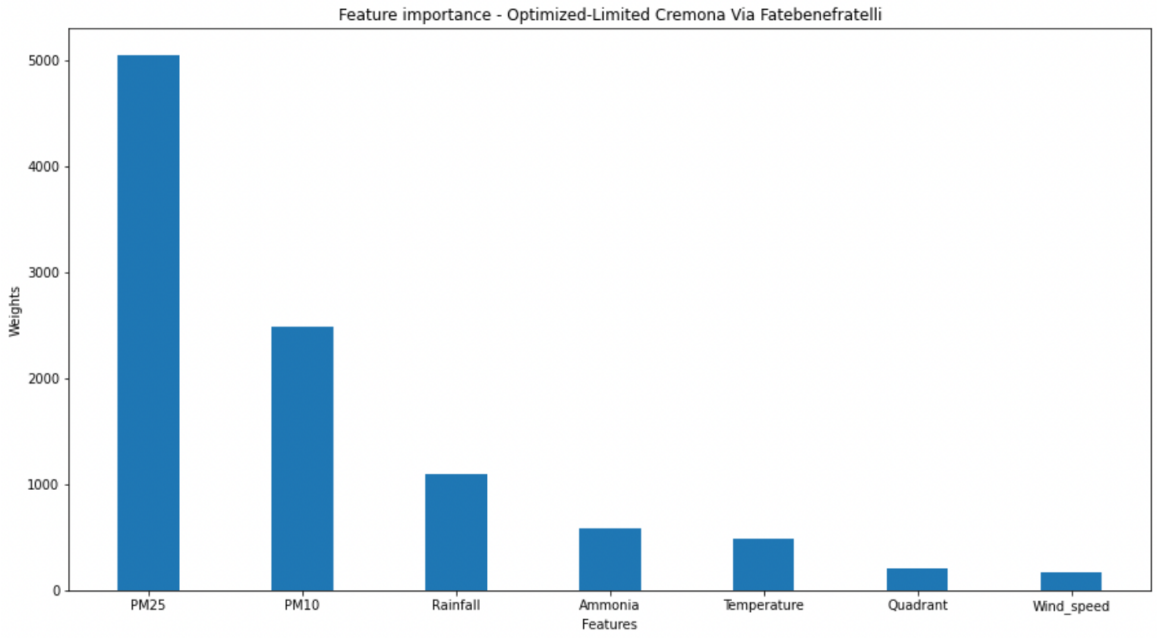

Figure 5.17: Feature importance ranking based on the predictions made by the Cremona's best model, using only the features analyzed in chapter 4, throughout 2020. Each feature is assigned a sum of weights which is directly proportional to its importance.

Similarly, in the best performing model using the data of Schivenoglia Via Malpasso limited to the feature set mentioned above (See table 6.14 in the Appendix) the concentration of PM10, PM2.5 and the temperature are the most relevant, whereas the amount of rainfall and the concentration of Ammonia have approximately the same, much more more limited impact. And yet, in the case of Moggio Loc Penscei, as presented in figure 5.19, the role Ammonia plays is much more important, managing to be the second most relevant feature, relegating PM10 to third place. In fact, as can be seen in figure 5.20, Ammonia turns out to be among the most informative features of both for the day of the prediction and the previous day.

This could be attributed to differences in the environments that surround the stations, such the fact that, among the three, Moggio's station is located in mountainous terrain whereas both Cremona and Schivenoglia are located in flat terrain. In order to truly determine whether there is a bigger truth behind this discrepancy or whether it is fluke, it would be necessary to analyze a much bigger number of stations.

The fact remains that Moggio's models tend to attribute more importance to Ammonia than as

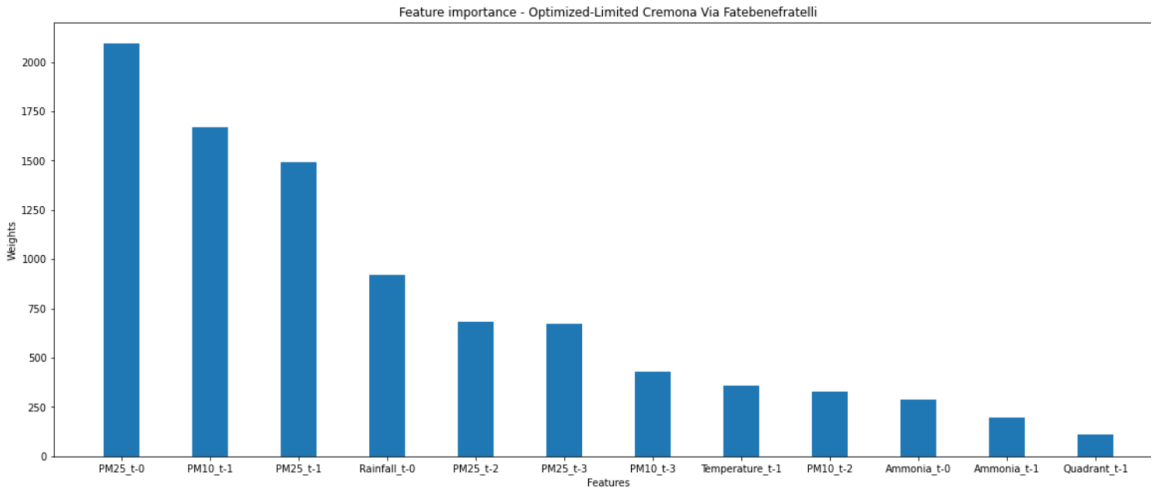

Figure 5.18: Feature importance ranking based on the predictions made by the Cremona's best model, using only the features analyzed in chapter 4, throughout 2020. Each feature, at each time step, is assigned a sum of weights which is directly proportional to its importance.

a feature. As a matter of fact, whenever PM2.5 is removed, Ammonia becomes the highest ranking variable for the model. When the feature set is extended to include all variables measured by the station it remains among the most relevant, taking the fourth place after Arsenic, PM10 and Cadmium (see table 6.16 in the appendix).

On the other hand, Cremona's models, tend not to give much importance to Ammonia unless PM2.5 is not available: with the reduced feature set it becomes the second most important feature, after PM10 itself, and with the extended feature set it goes from being the 9th most important feature to the 5th most important (see table 6.16 and 6.17 in the appendix).

Finally, in the case of Schivenoglia, even after the removal of PM2.5 as a feature, with or without an extended feature set that includes all variables offered by the station, Ammonia continues to be among the ones that contribute the least to the predictions (see table 6.14, 6.16 and 6.17 in the appendix).

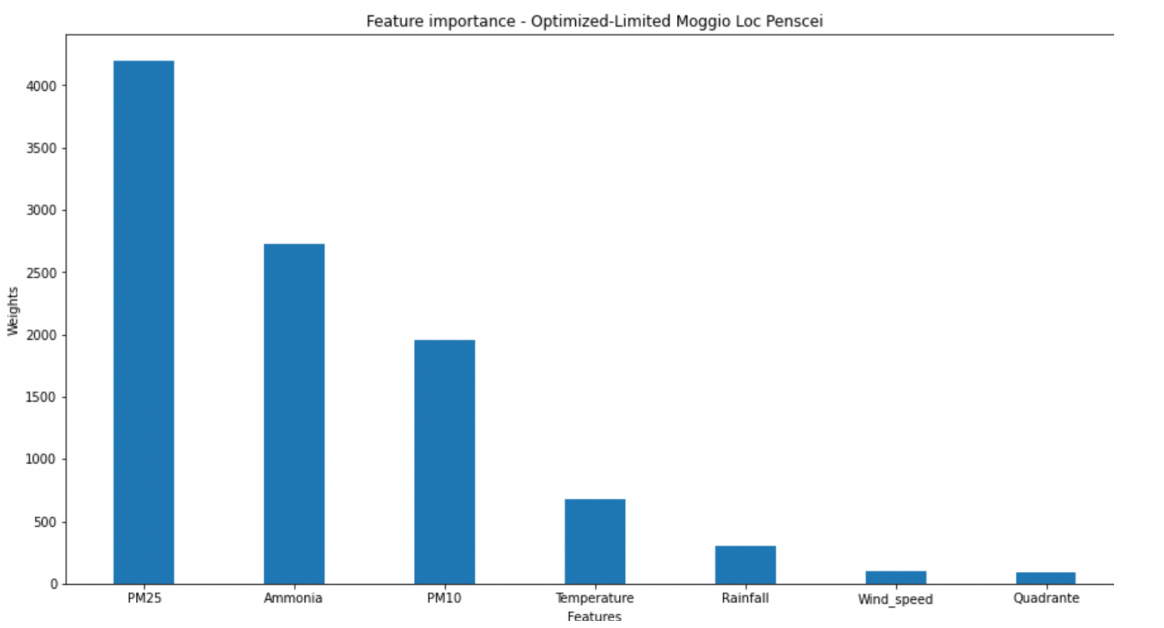

Figure 5.19: Feature importance ranking based on the predictions made by the Cremona's best model, using only the features analyzed in chapter 4, throughout 2020. Each feature is assigned a sum of weights which is directly proportional to its importance.

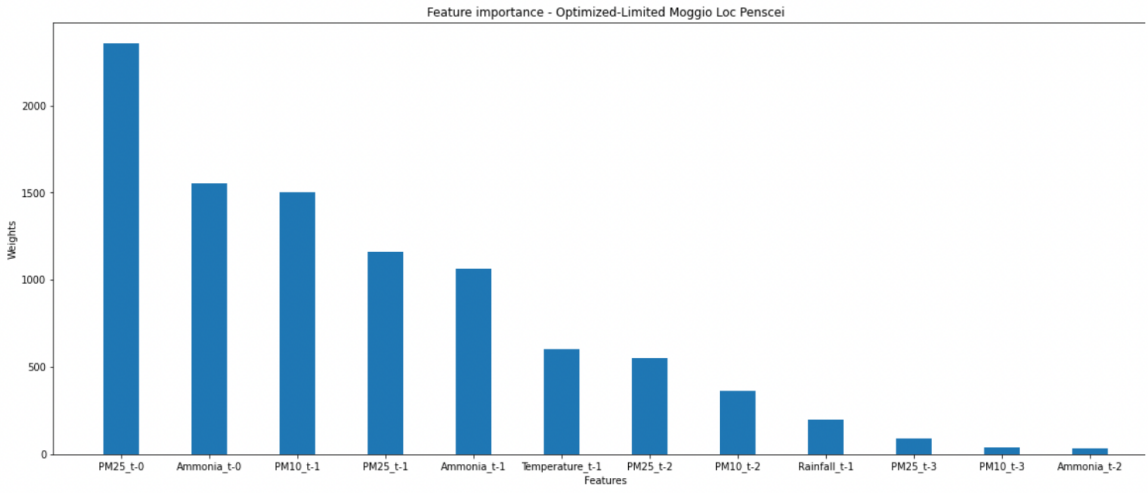

Figure 5.20: Feature importance ranking based on the predictions made by the Moggio's best model, using only the features analyzed in chapter 4, throughout 2020. Each feature, at each time step, is assigned a sum of weights which is directly proportional to its importance.

CHAPTER 6

Conclusions and further developments

Throughout this thesis, the raw data was used as a starting point to form an assessment of its potential to be modeled, which was initially based on the presumed relationship between the pollutants being discussed. These results were later used to construct and fine tune the architecture of a series of LSTM recurrent neural networks that capture the data's behavior.

Different obstacles were tackled, such as the problem posed by the presence of missing observations, which was addressed exploring different alternatives and evaluating their performance; or the difficulty of determining what type of data should be used to inform the predictions made by the models, which more concretely consisted of analyzing which time steps, relative to the instant of the prediction, were necessary to obtain the best results possible. The impact of the use of different hyperparameters was explored, which resulted both in better models and the possibility to deduce, from the results of the hyperparameter tuning process, which of them tend to have a stronger influence on performance for these specific datasets. The neural network's learning rate and batch size came forth as the most impactful ones.

It was also possible to glean a more concrete understanding of the otherwise nebulous black-box nature of neural network predictions through the use of surrogate linear models built to emulate its behavior with local fidelity, from which was thereby reasonable to conclude that, in combination with the analysis of models with different feature sets, the originally proposed selection of variables, which consists of PM2.5, NH3, wind speed, wind direction,

temperature, rainfall and PM10 itself, was informative enough to produce strong forecasts.

It was also found that the most relevant information towards the forecast of PM10 is the data of the other regressors on the very same day of the prediction. While the information of the days preceding the prediction was also found to be informative, the values at the same time instant as the forecast were always ranked the highest in the feature importance results of all models. Their removal results in regression R2 metricsthat drop from the 70% to 80% range to the 30% to 60% range.

In the best performing models, regression R2 metrics for the prediction of PM10 surpass 80% were achieved even in the absence of its closest cousin, PM2.5, among the regressors. What is more, this level of performance was sustained when the models were tested on data from 2020 –subject to different dynamics due to the COVID-19 lockdowns in Italy– despite the fact that the models were trained on pre-pandemic data, which reaffirms the efficacy of the proposed neural networks.

However, the same level of performance can be obtained with appropriately tuned, multiple linear regression models with ARIMA errors. This highlights the fact that the non-linear modeling capabilities of LSTM neural networks offer little to no improvements in performance for the forecasting of PM10 at different locations across Lombardy, when using the feature sets offered by the datasets that were used. As a consequence, the biggest differences between the two types of models are the less restrictive assumptions imposed on the data in the case of neural networks, and the inherent inferencing capabilities of the statistical linear models, which are not shared by LSTM neural networks.

The proposal could be further improved by adding geographical information so that the different networks can be retrained on the data of more than one location at time, improving the chance of capturing overarching trends that may be present in different regions of the Lombard territory. Additionally, the use of bleeding edge neural networks for time-series forecasting could be evaluated, such as Google's Temporal Fusion Transformer models [16] that have inbuilt explainability and offer uncertainty information through quantile regression as well as multi-step forecasting.

Bibliography

[1] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: *Journal of Machine Learning Research* 13.10 (2012), pp. 281–305. URL: http://jmlr.org/papers/v13/bergstra12a.html.

[2] Leo Breiman. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/a:1010933404324. URL: https://doi.org/10.1023/a:1010933404324.

[3] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[4] Francois Chollet. *Deep learning with python*. en. New York, NY: Manning Publications, Oct. 2017.

[5] Alessandro Fassò, Paolo Maranzano, and Philipp Otto. “Spatiotemporal variable selection and air quality impact assessment of COVID-19 lockdown”. In: *Spatial Statistics* 49 (2022). Spatio-temporal spread of Covid patterns: its spread, causes and scale, p. 100549. ISSN: 2211-6753. DOI: https://doi.org/10.1016/j.spasta.2021.100549. URL: https://www.sciencedirect.com/science/article/pii/S2211675321000592.

[6] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems”. In: Addison-Wesley, 2015. Chap. 7.

[7] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems”. In: Addison-Wesley, 2015. Chap. 10.

[8] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems”. In: Addison-Wesley, 2015. Chap. 15.

[9] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[10] Chih-wei Hsu, Chih-chung Chang, and Chih-Jen Lin. “A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin”. In: (Nov. 2003).

[11] Rob J. Hyndman and Yeasmin Khandakar. “Automatic Time Series Forecasting: The forecast Package for R”. In: *Journal of Statistical Software* 27.3 (2008), pp. 1–22. DOI: 10.18637/jss.v027.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v027i03.

[12] Kevin G. Jamieson and Ameet Talwalkar. “Non-stochastic Best Arm Identification and Hyperparameter Optimization”. In: *CoRR* abs/1502.07943 (2015). arXiv: 1502.07943. URL: http://arxiv.org/abs/1502.07943.

[13] Ang Li et al. “A Generalized Framework for Population Based Training”. In: *CoRR* abs/1902.01894 (2019). arXiv: 1902.01894. URL: http://arxiv.org/abs/1902.01894.

[14] Liam Li et al. “Massively Parallel Hyperparameter Tuning”. In: *CoRR* abs/1810.05934 (2018). arXiv: 1810.05934. URL: http://arxiv.org/abs/1810.05934.

[15] Lisha Li et al. “Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits”. In: *CoRR* abs/1603.06560 (2016). arXiv: 1603.06560. URL: http://arxiv.org/abs/1603.06560.

[16] Bryan Lim et al. “Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting”. In: 2021.

[17] ARPA Lombardia. *Inquinamento atmosferico: gli effetti potenziali sulla salute*. 2020. URL: https://www.regione.lombardia.it/wps/portal/istituzionale/HP/DettaglioRedazionale/servizi-e-informazioni/cittadini/salute-e-prevenzione/Sicurezza-negli-ambienti-di-vita-e-di-lavoro/inquinamento-atmosferico/inquinamento-atmosferico/.

[18] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *CoRR* abs/1705.07874 (2017). arXiv: 1705.07874. URL: http://arxiv.org/abs/1705.07874.

[19] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. *Gradient-based Hyperparameter Optimization through Reversible Learning*. 2015. DOI: 10.48550/ARXIV.1502.03492. URL: https://arxiv.org/abs/1502.03492.

[20] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/bf02478259. URL: https://doi.org/10.1007/bf02478259.

[21] Risto Miikkulainen et al. "Evolving Deep Neural Networks". In: *CoRR* abs/1703.00548 (2017). arXiv: 1703.00548. URL: http://arxiv.org/abs/1703.00548.

[22] R.G.M Morris. "D.O. Hebb: The Organization of Behavior, Wiley: New York 1949". In: *Brain Research Bulletin* 50.5-6 (Nov. 1999), p. 437. DOI: 10.1016/s0361-9230(99)00182-3. URL: https://doi.org/10.1016/s0361-9230(99)00182-3.

[23] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[24] David S. Stoffer Robert H. Shumway. "Time Series Analysis and Its Applications With R Examples". In: Springer Nature, 2017. Chap. 6.6.1.

[25] Sheldon M. Ross. "Introduction to Probability and Statistics for Engineers". In: Academic Press, 2016. Chap. 9.

[26] Lloyd S. Shapley. *Notes on the N-Person Game mdash; II: The Value of an N-Person Game*. Santa Monica, CA: RAND Corporation, 1951. DOI: 10.7249/RM0670.

[27] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. DOI: 10.48550/ARXIV.1206.2944. URL: https://arxiv.org/abs/1206.2944.

Appendix

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
38	256	1	0,01	50	0,900	0,934	6,524	5,052	56,000	0,023	81,000	0,019	0,023
39	64	1	0,01	40	0,897	0,932	6,621	5,105	29,000	0,023	54,000	0,024	0,023
900	8	2	0,0001	50	0,893	0,934	6,751	5,016	189,000	0,023	214,000	0,019	0,024
36	128	1	0,01	20	0,892	0,923	6,773	5,430	61,000	0,023	86,000	0,019	0,024
49	128	2	0,01	50	0,891	0,929	6,797	5,204	42,000	0,023	67,000	0,018	0,023
184	16	2	0,01	40	0,890	0,929	6,829	5,206	55,000	0,024	80,000	0,022	0,025
42	128	2	0,01	20	0,890	0,923	6,834	5,432	54,000	0,023	79,000	0,018	0,024
152	64	2	1	30	0,890	0,933	6,841	5,069	197,000	0,023	222,000	0,018	0,025
54	128	2	0,01	30	0,889	0,927	6,856	5,285	63,000	0,023	88,000	0,018	0,024
155	32	2	1	30	0,889	0,920	6,871	5,550	102,000	0,023	127,000	0,020	0,024
274	8	2	1	40	0,888	0,934	6,886	5,050	33,000	0,024	58,000	0,020	0,025
94	256	2	1	30	0,885	0,932	6,991	5,099	228,000	0,024	253,000	0,019	0,025
273	64	1	0,0001	50	0,884	0,919	7,009	5,572	324,000	0,024	349,000	0,020	0,024
190	16	2	1	50	0,882	0,932	7,071	5,103	54,000	0,024	79,000	0,020	0,026
909	16	2	0,0001	10	0,880	0,925	7,127	5,361	451,000	0,024	476,000	0,021	0,024
440	32	2	0,0001	20	0,880	0,924	7,156	5,385	403,000	0,025	428,000	0,021	0,025
273	8	1	1	40	0,879	0,917	7,170	5,662	48,000	0,024	73,000	0,020	0,025
746	4	1	1	40	0,879	0,916	7,182	5,673	94,000	0,026	119,000	0,017	0,027
43	256	1	0,01	20	0,877	0,920	7,236	5,546	130,000	0,023	155,000	0,020	0,029
96	32	2	0,01	10	0,876	0,927	7,246	5,302	62,000	0,025	87,000	0,022	0,028
212	256	1	0,0001	30	0,876	0,926	7,260	5,332	583,000	0,025	608,000	0,020	0,026
65	256	2	1	10	0,873	0,922	7,337	5,457	186,000	0,025	211,000	0,021	0,026
48	256	1	1	40	0,872	0,912	7,369	5,824	108,000	0,024	133,000	0,020	0,025
271	8	1	1	30	0,872	0,927	7,373	5,309	49,000	0,024	74,000	0,021	0,027

Table 6.1: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Cremona when the feature set is limited to PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
52	128	2	0,01	40	0,800	0,755	9,228	9,690	28,000	0,039	53,000	0,039	0,039
53	256	2	0,01	20	0,792	0,758	9,410	9,634	75,000	0,039	100,000	0,039	0,041
29	256	1	0,01	10	0,789	0,779	9,464	9,219	90,000	0,039	115,000	0,041	0,040
164	16	1	1	20	0,788	0,759	9,499	9,614	93,000	0,040	118,000	0,040	0,040
94	256	2	1	20	0,786	0,762	9,535	9,568	224,000	0,040	249,000	0,043	0,041
95	128	2	1	20	0,776	0,757	9,748	9,655	206,000	0,040	231,000	0,042	0,041
72	64	1	1	30	0,774	0,755	9,808	9,701	102,000	0,040	127,000	0,042	0,040
91	256	2	1	50	0,773	0,752	9,823	9,762	165,000	0,041	190,000	0,044	0,042
527	32	2	0,0001	20	0,771	0,747	9,855	9,864	481,000	0,042	506,000	0,046	0,043
50	64	1	0,01	50	0,771	0,745	9,858	9,889	34,000	0,039	59,000	0,040	0,043
215	16	2	1	30	0,769	0,757	9,917	9,669	59,000	0,040	84,000	0,039	0,044
93	256	2	1	40	0,768	0,763	9,938	9,547	163,000	0,041	188,000	0,043	0,042
31	128	1	0,01	30	0,767	0,745	9,959	9,888	46,000	0,039	71,000	0,039	0,043
333	256	2	0,0001	30	0,765	0,730	9,997	10,179	956,000	0,043	981,000	0,046	0,044
272	256	1	0,0001	20	0,761	0,736	10,084	10,077	988,000	0,044	999,000	0,046	0,044
274	16	1	1	10	0,757	0,737	10,165	10,054	179,000	0,039	204,000	0,042	0,045
49	64	1	0,01	40	0,756	0,722	10,178	10,335	50,000	0,038	75,000	0,038	0,044
193	8	1	0,01	30	0,754	0,687	10,221	10,969	42,000	0,043	67,000	0,044	0,044
30	128	2	0,1	10	0,746	0,682	10,383	11,051	22,000	0,039	47,000	0,043	0,043
205	256	2	0,0001	40	0,746	0,714	10,387	10,481	508,000	0,044	533,000	0,048	0,045
27	128	1	0,1	10	0,745	0,680	10,414	11,084	41,000	0,040	66,000	0,043	0,043
92	256	1	1	40	0,740	0,747	10,514	9,865	229,000	0,040	254,000	0,042	0,046
573	4	2	1	20	0,740	0,736	10,515	10,077	60,000	0,044	85,000	0,037	0,048
165	128	1	0,0001	30	0,733	0,708	10,644	10,593	460,000	0,046	485,000	0,048	0,046

Table 6.2: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Cremona when the feature set is limited to NH3, wind speed, wind direction, temperature, rainfall and PM10 itself. Notably, PM2.5 is not in the feature set.

Runtime	batch_size	layers	learning_rate	units	R2 test	R2 val	RMSE test	RMSE val	best_epoch	best_val_loss	epoch	loss	val_loss
51	256	1	0,01	30	0,899	0,937	6,555	4,905	87,000	0,022	112,000	0,017	0,023
31	128	1	0,01	10	0,895	0,929	6,691	5,239	72,000	0,023	97,000	0,018	0,024
53	128	2	0,01	30	0,893	0,931	6,731	5,137	34,000	0,023	59,000	0,021	0,024
154	64	2	1	40	0,893	0,923	6,753	5,429	161,000	0,023	186,000	0,017	0,024
513	4	2	1	30	0,891	0,933	6,798	5,057	45,000	0,022	70,000	0,017	0,025
158	32	1	1	30	0,891	0,934	6,813	5,018	147,000	0,023	172,000	0,018	0,024
92	128	1	1	50	0,889	0,927	6,863	5,283	107,000	0,023	132,000	0,018	0,024
25	256	1	0,1	10	0,889	0,914	6,869	5,732	51,000	0,023	76,000	0,023	0,025
1136	16	2	0,0001	10	0,888	0,934	6,901	5,024	582,000	0,023	607,000	0,018	0,023
231	256	1	0,0001	40	0,888	0,931	6,911	5,165	665,000	0,024	690,000	0,019	0,024
754	4	2	1	10	0,887	0,925	6,918	5,376	89,000	0,023	114,000	0,017	0,026
105	64	1	1	10	0,887	0,931	6,936	5,147	186,000	0,024	211,000	0,018	0,024
634	16	1	0,0001	20	0,884	0,929	7,020	5,236	359,000	0,023	384,000	0,018	0,024
192	128	1	0,0001	40	0,884	0,923	7,020	5,436	449,000	0,024	474,000	0,019	0,024
574	16	2	0,0001	20	0,883	0,926	7,048	5,325	249,000	0,023	274,000	0,019	0,024
874	16	2	0,0001	30	0,882	0,925	7,080	5,380	387,000	0,023	412,000	0,018	0,025
333	128	2	0,0001	20	0,881	0,919	7,105	5,575	746,000	0,024	771,000	0,020	0,025
146	32	1	1	40	0,881	0,922	7,113	5,468	128,000	0,023	153,000	0,020	0,027
52	64	1	0,01	20	0,880	0,906	7,149	6,024	64,000	0,023	89,000	0,023	0,026
151	32	1	0,01	50	0,879	0,932	7,177	5,103	92,000	0,023	117,000	0,017	0,025
318	64	1	0,0001	10	0,878	0,925	7,194	5,369	613,000	0,025	638,000	0,020	0,025
213	32	1	0,0001	40	0,878	0,929	7,196	5,224	179,000	0,025	204,000	0,020	0,025
273	128	1	0,0001	30	0,876	0,918	7,254	5,616	628,000	0,024	653,000	0,018	0,024

Table 6.3: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Cremona when the feature set is composed of all of the features available to the station.

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
277	4	1	1	40	0,848	0,841	8,032	7,802	24,000	0,034	49,000	0,030	0,035
50	128	1	0,01	20	0,839	0,837	8,283	7,909	61,000	0,033	86,000	0,030	0,034
31	256	1	0,01	10	0,831	0,861	8,474	7,313	69,000	0,035	94,000	0,033	0,036
27	128	1	0,01	30	0,830	0,858	8,497	7,384	31,000	0,034	56,000	0,030	0,036
24	256	1	0,01	30	0,821	0,858	8,712	7,394	44,000	0,035	69,000	0,033	0,037
92	64	1	1	20	0,821	0,845	8,720	7,717	116,000	0,035	141,000	0,033	0,037
52	128	2	0,01	50	0,818	0,838	8,792	7,879	58,000	0,034	83,000	0,029	0,037
77	64	2	1	40	0,816	0,846	8,835	7,701	82,000	0,036	107,000	0,031	0,038
880	4	2	0,0001	50	0,814	0,844	8,890	7,733	111,000	0,036	136,000	0,032	0,038
92	128	1	1	30	0,813	0,839	8,917	7,864	160,000	0,037	185,000	0,032	0,038
99	64	2	1	30	0,811	0,839	8,965	7,857	131,000	0,036	156,000	0,031	0,038
92	32	1	1	20	0,808	0,848	9,026	7,640	67,000	0,035	92,000	0,033	0,039
570	4	1	0,0001	40	0,808	0,834	9,042	7,995	84,000	0,037	109,000	0,034	0,039
54	256	1	1	20	0,804	0,831	9,115	8,058	220,000	0,038	245,000	0,034	0,039
151	16	1	0,01	20	0,794	0,824	9,347	8,229	44,000	0,036	69,000	0,046	0,039
154	32	2	0,01	40	0,793	0,784	9,372	9,106	88,000	0,036	113,000	0,028	0,039
573	64	2	0,0001	20	0,792	0,825	9,391	8,201	945,000	0,039	970,000	0,037	0,040
92	64	1	1	10	0,791	0,839	9,434	8,107	173,000	0,038	198,000	0,035	0,041
155	16	2	1	50	0,790	0,845	9,455	7,715	37,000	0,037	62,000	0,033	0,042
273	128	1	0,0001	40	0,784	0,820	9,571	8,314	562,000	0,040	587,000	0,036	0,040
60	32	1	0,01	10	0,773	0,835	9,826	7,963	34,000	0,034	59,000	0,037	0,046
80	16	1	1	30	0,771	0,839	9,858	7,862	30,000	0,037	55,000	0,037	0,045
559	4	1	0,0001	30	0,767	0,815	9,952	8,437	89,000	0,038	114,000	0,037	0,044
333	32	2	0,0001	40	0,767	0,809	9,953	8,563	297,000	0,041	322,000	0,039	0,043

Table 6.4: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Cremona when the feature set is composed of all of the features available to the station with the exception of PM2.5.

Runtime	batch_size	layers	learning_rate	units	R2 test	R2 val	RMSE test	RMSE val	best_epoch	best_val_loss	epoch	loss	val_loss
811	8	1	0,0001	50	0,877	0,914	2,955	3,041	243,000	0,021	268,000	0,031	0,021
52	128	2	0,01	20	0,875	0,911	2,972	3,082	56,000	0,021	81,000	0,032	0,023
391	32	1	0,0001	40	0,869	0,917	3,048	2,976	448,000	0,022	473,000	0,031	0,022
390	16	1	0,0001	30	0,868	0,912	3,056	3,070	251,000	0,022	276,000	0,032	0,022
37	64	2	0,01	10	0,866	0,889	3,078	3,446	34,000	0,022	59,000	0,031	0,023
23	64	1	0,01	20	0,865	0,902	3,095	3,238	8,000	0,022	33,000	0,034	0,023
90	128	1	1	30	0,863	0,894	3,116	3,370	136,000	0,021	161,000	0,032	0,022
30	64	1	0,01	40	0,861	0,882	3,138	3,556	19,000	0,021	44,000	0,032	0,023
314	8	2	1	10	0,859	0,908	3,163	3,139	95,000	0,022	120,000	0,030	0,024
62	128	2	0,01	30	0,858	0,867	3,167	3,776	50,000	0,022	75,000	0,032	0,024
67	64	2	1	40	0,857	0,894	3,181	3,370	77,000	0,022	102,000	0,032	0,024
332	8	1	1	20	0,855	0,903	3,201	3,218	83,000	0,023	108,000	0,028	0,025
68	32	2	1	40	0,855	0,879	3,210	3,598	22,000	0,022	47,000	0,032	0,023
211	4	1	1	10	0,852	0,918	3,242	2,961	15,000	0,023	40,000	0,031	0,025
213	128	2	0,0001	50	0,849	0,883	3,276	3,538	345,000	0,023	370,000	0,034	0,024
492	8	2	0,0001	40	0,848	0,910	3,280	3,115	140,000	0,024	165,000	0,031	0,025
270	8	1	1	30	0,846	0,886	3,306	3,492	52,000	0,025	77,000	0,029	0,026
49	256	1	0,1	50	0,843	0,869	3,339	3,751	87,000	0,025	112,000	0,042	0,026
212	32	2	0,0001	40	0,841	0,876	3,353	3,645	173,000	0,024	198,000	0,034	0,024
102	64	2	1	20	0,841	0,869	3,358	3,749	124,000	0,024	149,000	0,035	0,025
39	32	1	1	20	0,840	0,872	3,368	3,711	23,000	0,023	48,000	0,033	0,025
214	128	2	0,0001	40	0,837	0,872	3,395	3,701	279,000	0,025	304,000	0,035	0,025
179	4	1	0,01	30	0,823	0,892	3,539	3,395	5,000	0,026	30,000	0,034	0,028
211,000	4,000	1,000	0,010	20,000	0,822	0,902	3,553	3,234	13,000	0,025	38,000	0,034	0,028

Table 6.5: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Moggio when the feature set is limited to PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
30	128	1	0,01	40	0,656	0,634	4,940	6,260	55,000	0,040	80,000	0,050	0,041
24	128	1	0,01	30	0,626	0,689	5,147	5,772	26,000	0,042	51,000	0,058	0,042
137	32	1	1	10	0,621	0,715	5,182	5,528	174,000	0,043	199,000	0,056	0,043
95	16	2	1	40	0,621	0,699	5,183	5,680	31,000	0,040	56,000	0,056	0,042
90	32	1	1	20	0,616	0,736	5,214	5,316	103,000	0,042	128,000	0,055	0,043
237	4	1	1	30	0,612	0,720	5,246	5,476	31,000	0,041	56,000	0,051	0,043
90	16	1	1	30	0,605	0,689	5,290	5,771	44,000	0,042	69,000	0,057	0,044
331	4	2	1	10	0,597	0,671	5,346	5,941	41,000	0,042	66,000	0,054	0,044
72	16	2	0,01	30	0,596	0,619	5,351	6,388	19,000	0,045	44,000	0,055	0,045
38	64	2	0,01	10	0,589	0,663	5,394	6,013	33,000	0,040	58,000	0,053	0,044
1204	8	2	0,0001	20	0,584	0,669	5,426	5,956	417,000	0,045	442,000	0,060	0,045
814	4	1	0,0001	30	0,584	0,684	5,428	5,818	175,000	0,044	200,000	0,059	0,045
92	64	2	1	20	0,577	0,594	5,476	6,601	128,000	0,044	133,000	0,061	0,045
49	64	1	0,01	20	0,572	0,612	5,509	6,448	55,000	0,041	80,000	0,049	0,047
303	4	1	1	10	0,563	0,669	5,566	5,952	44,000	0,040	69,000	0,053	0,046
388	32	1	0,0001	40	0,561	0,659	5,578	6,049	481,000	0,046	506,000	0,061	0,046
1772	4	1	0,0001	10	0,556	0,642	5,609	6,191	435,000	0,045	460,000	0,060	0,046
47	256	1	1	40	0,555	0,686	5,615	5,802	164,000	0,043	189,000	0,059	0,046
299	4	1	1	40	0,549	0,706	5,650	5,617	36,000	0,040	61,000	0,049	0,046
873	8	1	0,0001	20	0,542	0,666	5,697	5,987	334,000	0,046	359,000	0,061	0,047
150	8	1	0,01	40	0,541	0,623	5,705	6,356	26,000	0,041	51,000	0,049	0,046
77	128	2	1	20	0,531	0,695	5,768	5,714	207,000	0,042	232,000	0,055	0,046
50	128	1	1	50	0,530	0,690	5,769	5,767	109,000	0,041	134,000	0,056	0,047
151	16	1	1	20	0,530	0,707	5,770	5,601	77,000	0,043	102,000	0,053	0,046

Table 6.6: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Moggio when the feature set is limited to NH3, wind speed, wind direction, temperature, rainfall and PM10 itself. Notably, PM2.5 is not in the feature set.

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
27	256	1	0,01	40	0,872	0,913	3,015	3,057	19,000	0,022	44,000	0,034	0,023
42	128	1	1	50	0,846	0,869	3,303	3,741	68,000	0,023	93,000	0,033	0,023
30	128	1	0,01	50	0,824	0,756	3,540	5,103	23,000	0,025	48,000	0,039	0,027
93	128	2	1	40	0,823	0,922	3,543	2,883	115,000	0,021	140,000	0,031	0,027
154	16	2	1	20	0,800	0,893	3,770	3,384	38,000	0,025	63,000	0,030	0,030
50	256	1	0,1	40	0,792	0,841	3,840	4,119	98,000	0,024	123,000	0,045	0,031
69	16	2	0,01	40	0,790	0,700	3,865	5,667	8,000	0,028	33,000	0,035	0,032
50	32	1	0,01	10	0,787	0,868	3,893	3,763	26,000	0,024	51,000	0,031	0,031
29	256	1	0,01	30	0,779	0,910	3,964	3,108	36,000	0,021	61,000	0,033	0,034
51	32	1	0,01	30	0,778	0,916	3,976	2,994	10,000	0,024	35,000	0,032	0,032
58	32	2	1	20	0,777	0,875	3,979	3,663	27,000	0,028	52,000	0,031	0,031
52	64	2	0,01	10	0,759	0,899	4,138	3,283	40,000	0,025	65,000	0,037	0,032
314	8	1	0,0001	50	0,757	0,874	4,151	3,675	97,000	0,026	122,000	0,030	0,035
211	4	1	1	20	0,738	0,870	4,318	3,731	8,000	0,029	33,000	0,029	0,036
441	4	1	0,0001	10	0,733	0,836	4,356	4,192	78,000	0,035	103,000	0,033	0,037
113	8	2	1	30	0,713	0,906	4,515	3,168	10,000	0,023	35,000	0,031	0,038
702	4	2	0,0001	20	0,700	0,876	4,619	3,638	91,000	0,037	116,000	0,031	0,042
214	8	2	1	20	0,696	0,901	4,646	3,249	45,000	0,028	70,000	0,028	0,039
291	4	1	0,0001	40	0,691	0,874	4,686	3,676	37,000	0,030	62,000	0,031	0,040
33	128	1	1	10	0,680	0,887	4,767	3,473	72,000	0,026	97,000	0,033	0,039
153	8	2	1	40	0,646	0,885	5,017	3,506	12,000	0,029	37,000	0,030	0,044
453	4	1	0,0001	20	0,641	0,859	5,052	3,877	69,000	0,040	94,000	0,031	0,046
52	32	2	0,01	20	0,615	0,905	5,233	3,179	15,000	0,025	40,000	0,030	0,039
105	64	1	0,0001	40	0,603	0,876	5,310	3,637	197,000	0,027	222,000	0,034	0,043

Table 6.7: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Moggio when the feature set is composed of all of the features available to the station.

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss
50	64	1	0,01	20	0,642	0,625	5,038	6,340	52,000	0,039	77,000	0,048
58	128	2	0,01	50	0,634	0,571	5,091	6,778	49,000	0,040	74,000	0,058
53	64	2	0,01	20	0,616	0,616	5,217	6,419	40,000	0,040	65,000	0,046
56	64	1	0,01	40	0,609	0,572	5,265	6,773	52,000	0,041	77,000	0,044
101	32	1	1	40	0,607	0,653	5,277	6,098	70,000	0,040	95,000	0,050
82	128	1	1	50	0,600	0,646	5,324	6,162	137,000	0,043	162,000	0,051
42	256	1	0,01	30	0,577	0,698	5,472	5,688	109,000	0,040	134,000	0,046
934	8	2	0,0001	40	0,562	0,634	5,574	6,268	190,000	0,045	215,000	0,058
507	8	1	0,0001	50	0,549	0,594	5,651	6,593	131,000	0,046	156,000	0,059
1111	8	1	0,0001	20	0,545	0,663	5,680	6,009	359,000	0,047	384,000	0,056
51	128	1	0,01	10	0,543	0,695	5,688	5,720	91,000	0,039	116,000	0,048
64	64	2	0,01	50	0,527	0,622	5,788	6,366	45,000	0,038	70,000	0,048
151	32	1	1	30	0,505	0,633	5,920	6,270	89,000	0,040	114,000	0,050
272	32	1	0,0001	40	0,505	0,648	5,921	6,143	246,000	0,049	271,000	0,063
93	64	1	1	10	0,467	0,580	6,144	6,711	71,000	0,050	96,000	0,064
29	256	1	0,1	10	0,394	0,456	6,552	7,639	96,000	0,047	121,000	0,063
50	32	1	0,01	50	0,384	0,614	6,608	6,430	14,000	0,048	39,000	0,055
54	256	1	0,01	50	0,355	0,585	6,761	6,666	98,000	0,039	123,000	0,053
212	32	1	0,0001	30	0,337	0,531	6,855	7,093	166,000	0,057	191,000	0,071
50	32	1	0,01	20	0,292	0,567	7,085	6,810	19,000	0,044	44,000	0,051
96	64	2	1	40	0,253	0,524	7,276	7,142	75,000	0,048	100,000	0,057
269	16	2	0,0001	30	0,226	0,465	7,406	7,574	102,000	0,063	127,000	0,073
415	16	2	0,0001	10	0,223	0,461	7,422	7,603	197,000	0,065	222,000	0,072
202	4	2	1	10	0,200	0,508	7,529	7,262	2,000	0,065	27,000	0,066

Table 6.8: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Moggio when the feature set is composed of all of the features available to the station with the exception of PM2.5.

Runtime	Sweep	batch_size	dropout	layers	learning_rate	units	R2 test	R2 val	RMSE test	RMSE val	best_epoch	best_val_loss	epoch	loss	val_loss
50		32	0	1	0,01	40	0,869	0,905	6,633	5,298	24,000	0,016	49,000	0,020	0,017
121		16	0	2	1	40	0,865	0,912	6,717	5,099	40,000	0,015	65,000	0,018	0,016
45		64	0	2	0,01	20	0,861	0,905	6,835	5,310	46,000	0,015	71,000	0,019	0,016
109		16	0	1	1	20	0,856	0,905	6,941	5,296	52,000	0,015	77,000	0,018	0,016
166		8	0	2	1	40	0,856	0,902	6,949	5,390	22,000	0,015	47,000	0,019	0,019
50		64	0	1	0,01	10	0,856	0,906	6,951	5,266	81,000	0,016	106,000	0,018	0,017
213		8	0	2	1	20	0,853	0,903	7,022	5,360	30,000	0,015	55,000	0,018	0,019
52		128	0	1	1	20	0,852	0,893	7,051	5,615	156,000	0,015	181,000	0,018	0,016
152		32	0	2	1	20	0,849	0,893	7,110	5,614	105,000	0,015	130,000	0,018	0,016
449		16	0	2	0,0001	30	0,847	0,905	7,162	5,288	253,000	0,016	278,000	0,020	0,017
92		256	0	1	1	30	0,847	0,900	7,166	5,440	189,000	0,015	214,000	0,018	0,016
51		64	0	1	0,01	40	0,842	0,889	7,275	5,718	48,000	0,015	73,000	0,019	0,017
92		128	0	1	1	10	0,841	0,890	7,307	5,699	166,000	0,015	191,000	0,019	0,016
212		32	0	1	0,0001	20	0,839	0,897	7,344	5,507	217,000	0,016	242,000	0,020	0,017
310		32	0	2	0,0001	20	0,838	0,899	7,374	5,465	325,000	0,017	350,000	0,020	0,017
92		16	0	2	0,01	40	0,835	0,901	7,429	5,401	27,000	0,017	52,000	0,024	0,019
213		256	0	2	0,0001	40	0,834	0,897	7,457	5,528	415,000	0,016	440,000	0,021	0,017
26		128	0	1	0,01	40	0,832	0,881	7,513	5,929	24,000	0,016	49,000	0,018	0,017
27		128	0	2	0,01	20	0,828	0,870	7,585	6,207	29,000	0,017	54,000	0,018	0,019
212		128	0	2	0,0001	40	0,815	0,869	7,876	6,218	360,000	0,017	385,000	0,020	0,017
56		64	0	2	1	20	0,815	0,856	7,886	6,528	32,000	0,017	77,000	0,020	0,018
498		4	0	1	0,0001	10	0,806	0,869	8,074	6,227	90,000	0,020	115,000	0,020	0,022
54		128	0	2	0,1	30	0,804	0,851	8,113	6,643	105,000	0,016	130,000	0,022	0,019
100		64	0	1	0,0001	50	0,802	0,844	8,157	6,799	119,000	0,017	144,000	0,021	0,019

Table 6.9: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Schivenoglia when the feature set is limited to PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself

Runtime	batch_size	layers	learning_rate	units	MAE test	MAE val	R2 test	R2 val	RMSE test	RMSE val	best_epoch	best_val_loss	epoch	loss	val_loss
91	64	1	1	20	7,108	7,044	0,697	0,696	10,076	9,480	106,000	0,028	131,000	0,031	0,028
92	256	2	1	10	6,983	7,231	0,697	0,674	10,085	9,815	403,000	0,028	428,000	0,032	0,028
52	128	2	0,01	30	7,327	7,292	0,686	0,679	10,263	9,742	40,000	0,028	65,000	0,029	0,029
52	128	2	1	20	7,220	7,381	0,686	0,663	10,267	9,986	135,000	0,029	160,000	0,033	0,029
630	32	1	0,0001	10	7,016	7,238	0,682	0,670	10,325	9,874	945,000	0,028	970,000	0,031	0,028
51	256	1	1	20	7,289	7,400	0,680	0,655	10,355	10,097	219,000	0,029	244,000	0,032	0,029
332	64	2	0,0001	40	7,033	7,244	0,680	0,662	10,363	9,998	603,000	0,028	628,000	0,032	0,028
49	128	1	1	10	7,271	7,512	0,678	0,650	10,393	10,176	165,000	0,029	190,000	0,032	0,029
26	64	1	0,01	40	7,173	7,157	0,678	0,678	10,395	9,764	19,000	0,028	44,000	0,030	0,028
49	128	1	1	20	7,233	7,294	0,678	0,671	10,396	9,868	154,000	0,028	179,000	0,032	0,029
213	8	2	1	40	7,440	6,991	0,677	0,709	10,402	9,276	36,000	0,028	61,000	0,028	0,030
1593	4	2	0,0001	20	7,242	7,173	0,675	0,685	10,446	9,655	221,000	0,028	246,000	0,031	0,029
270	256	1	0,0001	40	7,399	7,350	0,673	0,667	10,468	9,926	988,000	0,029	999,000	0,032	0,029
29	128	1	0,01	30	7,473	7,578	0,672	0,663	10,488	9,982	38,000	0,028	63,000	0,028	0,030
492	4	1	0,0001	40	7,289	7,248	0,670	0,672	10,523	9,855	92,000	0,029	117,000	0,032	0,029
294	32	2	0,0001	40	7,233	7,699	0,670	0,626	10,525	10,509	310,000	0,028	335,000	0,033	0,029
331	128	2	0,0001	20	7,301	7,384	0,669	0,658	10,533	10,062	999,000	0,029	999,000	0,033	0,029
1058	4	2	0,0001	20	7,301	7,354	0,668	0,666	10,544	9,938	187,000	0,028	212,000	0,032	0,029
211	8	1	1	40	7,685	7,286	0,667	0,697	10,565	9,469	47,000	0,029	72,000	0,029	0,030
525	4	1	0,0001	30	7,347	7,381	0,666	0,661	10,582	10,014	111,000	0,029	136,000	0,032	0,029
83	16	1	0,01	10	7,367	7,373	0,665	0,682	10,592	9,704	45,000	0,028	70,000	0,029	0,029
451	8	1	0,0001	50	7,211	7,228	0,663	0,674	10,599	9,811	145,000	0,029	170,000	0,032	0,029
92	128	2	1	30	7,305	7,367	0,664	0,642	10,612	10,295	112,000	0,029	137,000	0,032	0,030
391	4	1	1	30	7,866	7,525	0,664	0,652	10,621	10,146	56,000	0,030	81,000	0,027	0,031

Table 6.10: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Schivenoglia when the feature set is limited to NH3, wind speed, wind direction, temperature, rainfall and PM10 itself. Notably, PM2.5 is not in the feature set.

Runtime	batch_size	layers	learning_rate	units	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
214	8	2	1	30	0,900	7,319	5,442	27,000	0,016	52,000	0,017	0,017
740	8	1	0,0001	40	0,899	7,435	5,462	231,000	0,015	256,000	0,017	0,016
50	128	1	0,01	50	0,871	7,655	6,173	80,000	0,016	105,000	0,019	0,018
754	4	2	1	10	0,860	7,658	6,430	96,000	0,016	121,000	0,017	0,019
243	32	1	0,0001	20	0,897	7,916	5,510	274,000	0,018	299,000	0,020	0,018
159	8	1	1	30	0,880	7,940	5,962	29,000	0,016	54,000	0,017	0,021
93	32	1	1	30	0,903	7,963	5,349	30,000	0,016	55,000	0,018	0,018
154	32	2	1	20	0,905	8,011	5,309	81,000	0,019	106,000	0,017	0,019
440	4	1	1	40	0,898	8,039	5,498	55,000	0,018	80,000	0,015	0,020
51	128	1	1	30	0,857	8,087	6,508	48,000	0,019	73,000	0,025	0,021
51	64	1	0,01	40	0,890	8,135	5,704	19,000	0,018	44,000	0,021	0,021
32	128	1	1	40	0,869	8,139	6,221	34,000	0,018	59,000	0,024	0,022
693	4	2	1	20	0,887	8,276	5,765	85,000	0,018	110,000	0,014	0,020
196	8	2	1	20	0,889	8,287	5,721	29,000	0,018	54,000	0,018	0,023
145	32	2	1	40	0,879	8,325	5,987	100,000	0,018	125,000	0,017	0,021
46	256	1	1	30	0,868	8,459	6,238	135,000	0,017	160,000	0,018	0,019
51	128	1	0,1	30	0,880	8,570	5,958	52,000	0,015	77,000	0,024	0,022
51	128	1	1	20	0,784	8,706	7,991	63,000	0,019	88,000	0,022	0,020
394	8	2	0,0001	40	0,837	8,755	6,936	71,000	0,021	96,000	0,021	0,024
216	4	1	1	40	0,779	9,076	8,075	5,000	0,022	30,000	0,018	0,029
52	128	2	1	10	0,753	9,174	8,532	86,000	0,023	111,000	0,024	0,025
53	128	2	0,01	30	0,795	9,368	7,781	38,000	0,017	63,000	0,021	0,026
95	32	2	1	30	0,904	9,563	5,321	41,000	0,017	66,000	0,018	0,027
87	16	1	0,01	10	0,787	9,678	7,926	32,000	0,017	57,000	0,022	0,029

Table 6.11: SThese are the 24 best results of the hyperparameter tuning process, using Random Search, for Schivenoglia when the feature set is composed of all of the features available to the station.

Runtime	batch_size	layers	learning_rate	units	R2_test	R2_val	RMSE_test	RMSE_val	best_epoch	best_val_loss	epoch	loss	val_loss
27	128	1	0,01	30	0,708	0,701	9,857	9,392	32,000	0,025	57,000	0,025	0,027
30	128	1	0,01	20	0,694	0,695	10,095	9,484	45,000	0,026	70,000	0,026	0,027
91	32	1	0,01	40	0,683	0,647	10,268	10,207	48,000	0,025	73,000	0,025	0,028
887	4	2	0,0001	40	0,672	0,639	10,442	10,322	137,000	0,027	162,000	0,026	0,028
49	32	1	0,01	20	0,672	0,719	10,442	9,116	22,000	0,025	47,000	0,026	0,028
72	128	2	1	30	0,671	0,645	10,456	10,240	179,000	0,027	204,000	0,024	0,029
270	16	1	0,0001	20	0,669	0,662	10,492	9,987	192,000	0,027	217,000	0,029	0,028
189	16	1	0,0001	40	0,659	0,656	10,651	10,076	117,000	0,026	142,000	0,030	0,028
23	128	1	0,01	40	0,645	0,707	10,871	9,297	22,000	0,028	47,000	0,026	0,029
40	128	1	0,1	30	0,644	0,554	10,888	11,471	110,000	0,024	135,000	0,032	0,028
396	8	2	0,0001	40	0,610	0,676	11,398	9,774	109,000	0,030	134,000	0,029	0,031
992	4	2	0,0001	10	0,607	0,642	11,432	10,275	169,000	0,030	194,000	0,029	0,031
50	32	2	0,01	20	0,603	0,575	11,496	11,209	25,000	0,026	50,000	0,028	0,034
52	64	2	0,01	50	0,594	0,572	11,629	11,236	22,000	0,025	47,000	0,029	0,031
273	4	2	1	50	0,583	0,655	11,775	10,101	12,000	0,027	37,000	0,024	0,037
24	128	2	0,01	10	0,575	0,708	11,899	9,284	22,000	0,034	47,000	0,029	0,036
49	64	1	0,01	50	0,549	0,673	12,250	9,831	55,000	0,023	80,000	0,030	0,035
49	256	1	1	10	0,517	0,680	12,674	9,726	112,000	0,032	137,000	0,031	0,039
30	128	1	1	20	0,513	0,473	12,724	12,476	63,000	0,031	88,000	0,031	0,034
90	64	1	0,0001	40	0,383	0,487	14,331	12,304	70,000	0,038	95,000	0,037	0,041
18	256	1	1	20	0,335	0,360	14,881	13,743	23,000	0,043	48,000	0,039	0,043
28	128	1	0,0001	20	0,278	0,257	15,502	14,810	45,000	0,044	70,000	0,044	0,045
96	32	2	0,0001	10	0,246	0,337	15,841	13,989	81,000	0,045	106,000	0,044	0,045
46	128	2	0,0001	30	0,241	0,295	15,893	14,433	85,000	0,044	110,000	0,043	0,045

Table 6.12: These are the 24 best results of the hyperparameter tuning process, using Random Search, for Schivenoglia when the feature set is composed of all of the features available to the station with the exception of PM2.5.

Baseline	Moggio	Cremona	Schivenoglia
Ammonia	599	553	297
PM10	2031	2224	1899
PM25	1635	1616	1343
Quadrante	1093	114	1170
Rainfall	2023	1516	1084
Temperature	1370	1832	911
Wind_speed	265	573	1220
Ammonia_t-1	585	276	287
Ammonia_t-2	5	128	0
Ammonia_t-3	4	99	8
Ammonia_t-4	2	36	2
Ammonia_t-5	3	14	0
Ammonia_t-6	0	0	0
PM10_t-1	1875	1576	1489
PM10_t-2	4	487	388
PM10_t-3	124	126	10
PM10_t-4	22	21	4
PM10_t-5	6	14	8
PM10_t-6	0	0	0
PM25_t-1	1182	1245	1331
PM25_t-2	426	191	7
PM25_t-3	12	83	2
PM25_t-4	14	69	3
PM25_t-5	1	28	0
PM25_t-6	0	0	0
Quadrante_t-1	943	87	515
Quadrante_t-2	104	10	641
Quadrante_t-3	33	3	11
Quadrante_t-4	7	4	0
Quadrante_t-5	6	10	3
Quadrante_t-6	0	0	0
Rainfall_t-1	1984	1471	1071
Rainfall_t-2	22	16	6
Rainfall_t-3	14	7	3
Rainfall_t-4	2	14	3
Rainfall_t-5	1	8	1
Rainfall_t-6	0	0	0
Temperature_t-1	1357	1213	899
Temperature_t-2	0	461	9
Temperature_t-3	3	121	2
Temperature_t-4	8	23	0
Temperature_t-5	2	14	1
Temperature_t-6	0	0	0
Wind_speed_t-1	224	531	854
Wind_speed_t-2	1	29	169
Wind_speed_t-3	3	11	135
Wind_speed_t-4	13	0	59
Wind_speed_t-5	24	2	3
Wind_speed_t-6	0	0	0

Table 6.13: These are the weights that represent the feature importance of all the different features during 2020. The feature set is limited to PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself up to the day prior to the prediction

Limited	Moggio	Cremona	Schivenoglia
Ammonia	2722	590	476
PM10	1954	2487	2211
PM25	4200	5054	4553
Quadrante	91	206	142
Rainfall	306	1091	458
Temperature	677	484	2024
Wind_speed	102	168	244
Ammonia_t-0	1556	288	239
Ammonia_t-1	1064	197	138
Ammonia_t-2	32	30	56
Ammonia_t-3	18	20	9
Ammonia_t-4	32	35	23
Ammonia_t-5	20	20	11
PM10_t-0	0	0	0
PM10_t-2	366	326	441
PM10_t-3	40	430	5
PM10_t-4	23	39	89
PM10_t-5	25	25	40
PM25_t-0	2358	2096	1952
PM25_t-1	1163	1492	1257
PM25_t-2	549	683	1215
PM25_t-3	91	674	20
PM25_t-4	32	44	52
PM25_t-5	7	65	57
Quadrante_t-0	13	11	107
Quadrante_t-1	10	113	7
Quadrante_t-2	16	38	7
Quadrante_t-3	14	16	6
Quadrante_t-4	21	11	5
Quadrante_t-5	17	17	10
Rainfall_t-0	22	921	293
Rainfall_t-1	201	54	122
Rainfall_t-2	16	53	6
Rainfall_t-3	16	17	5
Rainfall_t-4	27	19	19
Rainfall_t-5	24	27	13
Temperature_t-0	22	24	1412
Temperature_t-1	601	359	507
Temperature_t-2	17	6	66
Temperature_t-3	5	39	13
Temperature_t-4	16	44	8
Temperature_t-5	16	12	18
Wind_speed_t-0	12	52	141
Wind_speed_t-1	15	22	54
Wind_speed_t-2	18	41	7
Wind_speed_t-3	29	24	5
Wind_speed_t-4	22	16	25
Wind_speed_t-5	6	13	12

Table 6.14: These are the weights that represent the feature importance of all the different features during 2020. The feature set is limited to PM2.5, NH3, wind speed, wind direction, temperature, rainfall and PM10 itself, including the values of the very same day of the prediction with the exception of the response variable, PM10

Limited no pm2.5	Moggio	Cremona	Schivenoglia
Ammonia	4247	1976	572
PM10	971	2048	2313
Quadrante	49	150	295
Rainfall	1331	1380	2567
Temperature	939	1756	517
Wind_speed	2	250	1317
Ammonia_t-0	1610	1169	496
Ammonia_t-1	1634	773	75
Ammonia_t-2	998	34	0
Ammonia_t-3	1	0	1
Ammonia_t-4	4	0	0
Ammonia_t-5	0	0	0
PM10_t-0	0	0	0
PM10_t-1	795	1597	1625
PM10_t-2	154	436	568
PM10_t-3	2	3	21
PM10_t-4	10	11	0
PM10_t-5	10	1	99
Quadrante_t-0	36	137	143
Quadrante_t-1	8	7	2
Quadrante_t-2	0	4	122
Quadrante_t-3	0	0	19
Quadrante_t-4	1	2	1
Quadrante_t-5	4	0	8
Rainfall_t-0	608	908	1354
Rainfall_t-1	696	469	1160
Rainfall_t-2	12	0	52
Rainfall_t-3	7	1	1
Rainfall_t-4	5	1	0
Rainfall_t-5	3	1	0
Temperature_t-0	123	442	2
Temperature_t-1	796	798	51
Temperature_t-2	0	470	344
Temperature_t-3	17	27	114
Temperature_t-4	2	19	6
Temperature_t-5	1	0	0
Wind_speed_t-0	1	249	1076
Wind_speed_t-1	0	0	222
Wind_speed_t-2	0	0	5
Wind_speed_t-3	0	0	3
Wind_speed_t-4	1	1	0
Wind_speed_t-5	0	0	11

Table 6.15: These are the weights that represent the feature importance of all the different features during 2020. The feature set is limited to NH3, wind speed, wind direction, temperature, rainfall and PM10 itself, including the values of the very same day of the prediction with the exception of the response variable, PM10. Notably, PM2.5 is not in the feature set.

Extended	Moggio	Cremona	Schivenoglia
Ammonia	5030	1207	337
Arsenic	15117		8033
Benzene			8743
Benzo_a_pyrene	1278		774
CO		4233	1761
Cadmium	6048		16192
Global_radiation	1942	1580	320
Lead	1794		2728
NO2	3269	1849	2576
NOx	601	1284	332
Nikel	781		651
Ozone	2922	387	1286
PM10	7045	7826	5688
PM25	10842	12152	12241
Quadrante	407	445	2925
Relative_humidity	378	728	285
Temperature	927	3256	7275
Wind_direction			
Wind_speed	440	404	500
Wind_speed_max			
Sulfur_dioxide		452	568
NO	4277		
Rainfall	1819	1997	2385
Ammonia_t-0	3280	473	111
Ammonia_t-1	1481	556	41
Ammonia_t-2	55	44	44
Ammonia_t-3	76	58	36
Ammonia_t-4	65	43	58
Ammonia_t-5	73	33	47
Arsenic_t-0	6235		4037
Arsenic_t-1	5780		3266
Arsenic_t-2	2820		623
Arsenic_t-3	207		15
Arsenic_t-4	40		41
Arsenic_t-5	35		51
Benzene_t-0			4402
Benzene_t-1			4047
Benzene_t-2			186
Benzene_t-3			10
Benzene_t-4			37
Benzene_t-5			61
Benzo_a_pyrene_t-0	579		149
Benzo_a_pyrene_t-1	127		219
Benzo_a_pyrene_t-2	150		94
Benzo_a_pyrene_t-3	161		113
Benzo_a_pyrene_t-4	150		91
Benzo_a_pyrene_t-5	111		108
CO_t-0		1938	62
CO_t-1		2059	1514
CO_t-2		63	60
CO_t-3		55	54
CO_t-4		44	19
CO_t-5		74	52
Cadmium_t-0	2645		6031
Cadmium_t-1	3023		4716
Cadmium_t-2	142		4844
Cadmium_t-3	71		330
Cadmium_t-4	73		214
Cadmium_t-5	94		57
Global_radiation_t-0	917	1274	82
Global_radiation_t-1	286	81	66
Global_radiation_t-2	199	40	91
Global_radiation_t-3	230	58	43
Global_radiation_t-4	151	48	23
Global_radiation_t-5	159	79	15
Lead_t-0	295		720
Lead_t-1	1152		1905
Lead_t-2	94		16
Lead_t-3	111		25

Table 6.16: These are the weights that represent the feature importance of all the different features during 2020. The feature set is composed of all of the features available to the station.

Lead_t-4	72		38
Lead_t-5	70		24
NO2_t-0	2863	1366	1953
NO2_t-1	192	159	364
NO2_t-2	44	144	86
NO2_t-3	86	66	83
NO2_t-4	28	56	43
NO2_t-5	56	58	47
NOx_t-0	127	900	65
NOx_t-1	193	159	77
NOx_t-2	79	46	36
NOx_t-3	95	50	46
NOx_t-4	42	34	61
NOx_t-5	65	95	47
Nikel_t-0	327		232
Nikel_t-1	231		307
Nikel_t-2	64		49
Nikel_t-3	52		17
Nikel_t-4	46		29
Nikel_t-5	61		17
Ozone_t-0	1787	68	675
Ozone_t-1	849	47	235
Ozone_t-2	45	69	148
Ozone_t-3	92	59	75
Ozone_t-4	71	83	78
Ozone_t-5	78	61	75
PM10_t-0	0	0	0
PM10_t-1	4345	3709	4942
PM10_t-2	1264	2037	78
PM10_t-3	1133	1464	541
PM10_t-4	151	438	40
PM10_t-5	152	178	87
PM25_t-0	5742	4273	5294
PM25_t-1	3284	3231	4472
PM25_t-2	511	1624	1401
PM25_t-3	943	1887	911
PM25_t-4	265	845	122
PM25_t-5	97	292	41
Quadrante_t-0	125	175	900
Quadrante_t-1	69	118	1915
Quadrante_t-2	57	24	31
Quadrante_t-3	51	35	19
Quadrante_t-4	62	21	41
Quadrante_t-5	43	72	19
Relative_humidity_t-0	69	218	36
Relative_humidity_t-1	36	244	40
Relative_humidity_t-2	73	99	126
Relative_humidity_t-3	79	63	33

Relative_humidity_t-4	65	64	14
Relative_humidity_t-5	56	40	36
Temperature_t-0	67	2626	4741
Temperature_t-1	570	114	2320
Temperature_t-2	69	170	92
Temperature_t-3	71	139	40
Temperature_t-4	90	117	47
Temperature_t-5	60	90	35
Wind_direction_t-0		144	
Wind_direction_t-1		109	
Wind_direction_t-2		50	
Wind_direction_t-3		48	
Wind_direction_t-4		36	
Wind_direction_t-5		65	
Wind_speed_max_t-0			
Wind_speed_max_t-1			
Wind_speed_max_t-2			
Wind_speed_max_t-3			
Wind_speed_max_t-4			
Wind_speed_max_t-5			
Wind_speed_t-0	97	65	47
Wind_speed_t-1	76	53	303
Wind_speed_t-2	55	83	15
Wind_speed_t-3	74	81	71
Wind_speed_t-4	22	77	35
Wind_speed_t-5	116	45	29
Sulfur_dioxide_t-0		144	205
Sulfur_dioxide_t-1		109	135
Sulfur_dioxide_t-2		50	66
Sulfur_dioxide_t-3		48	71
Sulfur_dioxide_t-4		36	45
Sulfur_dioxide_t-5		65	46
NO_t-0	94		
NO_t-1	57		
NO_t-2	53		
NO_t-3	53		
NO_t-4	86		
NO_t-5	64		
Rainfall_t-0	1166	1411	2176
Rainfall_t-1	162	278	40
Rainfall_t-2	118	119	33
Rainfall_t-3	172	81	55
Rainfall_t-4	142	55	50
Rainfall_t-5	59	53	31

Extended no pm2.5	Moggio	Cremona	Schivenoglia
Ammonia	6286	3216	1904
Arsenic	14396		6370
Benzene			9223
Benzo_a_pyrene	1042		121
CO		5770	8447
Cadmium	15279		11264
Global_radiation	2972	481	19
Lead	3252		6559
NO2	2030	3389	4615
NOx	229	848	360
Nikel	127		1084
Ozone	404	1862	1122
PM10	3424	3580	4859
Quadrante	234	335	304
Relative_humidity	1244	2590	2691
Temperature	2730	8627	4107
Wind_direction			
Wind_speed	641	63	757
Wind_speed_max			
Sulfur_dioxide		743	311
NO	2295		
Rainfall	295	1256	4283
Ammonia_t-0	4138	1908	1753
Ammonia_t-1	1845	1291	140
Ammonia_t-2	206	8	4
Ammonia_t-3	89	0	1
Ammonia_t-4	2	5	4
Ammonia_t-5	6	4	2
Arsenic_t-0	5207		2336
Arsenic_t-1	5373		4015
Arsenic_t-2	3724		0
Arsenic_t-3	1		12
Arsenic_t-4	53		1
Arsenic_t-5	38		6
Benzene_t-0			5244
Benzene_t-1			3943
Benzene_t-2			3
Benzene_t-3			16
Benzene_t-4			16
Benzene_t-5			1
Benzo_a_pyrene_t-0	837		37
Benzo_a_pyrene_t-1	35		22
Benzo_a_pyrene_t-2	34		19
Benzo_a_pyrene_t-3	45		13
Benzo_a_pyrene_t-4	71		19
Benzo_a_pyrene_t-5	20		11
CO_t-0			3906
CO_t-1		3239	4481
CO_t-2		2504	51
CO_t-3		13	1
CO_t-4		5	8
CO_t-5		8	0
Cadmium_t-0	5242	1	5739
Cadmium_t-1	5147		88
Cadmium_t-2	4420		1715
Cadmium_t-3	72		2892
Cadmium_t-4	356		532
Cadmium_t-5	42		298
Global_radiation_t-0	1236	242	6
Global_radiation_t-1	839	197	6
Global_radiation_t-2	771	8	1
Global_radiation_t-3	53	16	1
Global_radiation_t-4	43	9	0
Global_radiation_t-5	30	9	5

Table 6.17: These are the weights that represent the feature importance of all the different features during 2020. The feature set is composed of all of the features available to the station with the exception of PM2.5.

Lead_t-0	2452		3022
Lead_t-1	32		3165
Lead_t-2	725		347
Lead_t-3	17		11
Lead_t-4	6		13
Lead_t-5	20		1
NO2_t-0	1900	2597	3499
NO2_t-1	16	610	1083
NO2_t-2	67	123	21
NO2_t-3	22	45	0
NO2_t-4	8	7	9
NO2_t-5	17	7	3
NOx_t-0	216	540	50
NOx_t-1	3	271	279
NOx_t-2	2	6	21
NOx_t-3	1	10	1
NOx_t-4	1	11	0
NOx_t-5	6	10	9
Nikel_t-0	29		453
Nikel_t-1	19		579
Nikel_t-2	17		2
Nikel_t-3	18		22
Nikel_t-4	24		8
Nikel_t-5	20		20
Ozone_t-0	312	864	33
Ozone_t-1	9	987	877
Ozone_t-2	36	5	175
Ozone_t-3	4	1	7
Ozone_t-4	24	5	13
Ozone_t-5	19	0	17
PM10_t-0	0	0	0
PM10_t-1	2415	3205	4735
PM10_t-2	848	59	7
PM10_t-3	36	277	80
PM10_t-4	118	5	36
PM10_t-5	7	34	1
Quadrante_t-0	204	58	36
Quadrante_t-1	6	52	8
Quadrante_t-2	20	218	152
Quadrante_t-3	2	6	76
Quadrante_t-4	1	1	28
Quadrante_t-5	1	0	4
Relative_humidity_t-0	908	2115	1761
Relative_humidity_t-1	10	354	911
Relative_humidity_t-2	288	15	0

Relative_humidity_t-3	18	73	5
Relative_humidity_t-4	10	20	4
Relative_humidity_t-5	10	13	10
Temperature_t-0	1897	4048	2102
Temperature_t-1	700	3587	1980
Temperature_t-2	114	12	0
Temperature_t-3	1	698	4
Temperature_t-4	6	143	11
Temperature_t-5	12	139	10
Wind_direction_t-0			718
Wind_direction_t-1			27
Wind_direction_t-2			10
Wind_direction_t-3			0
Wind_direction_t-4			2
Wind_direction_t-5			0
Wind_speed_max_t-0			
Wind_speed_max_t-1			
Wind_speed_max_t-2			
Wind_speed_max_t-3			
Wind_speed_max_t-4			
Wind_speed_max_t-5			
Wind_speed_t-0	199	15	
Wind_speed_t-1	217	20	
Wind_speed_t-2	208	14	
Wind_speed_t-3	13	7	
Wind_speed_t-4	4	0	
Wind_speed_t-5	0	7	
Sulfur_dioxide_t-0		709	114
Sulfur_dioxide_t-1		17	162
Sulfur_dioxide_t-2		7	17
Sulfur_dioxide_t-3		8	1
Sulfur_dioxide_t-4		2	8
Sulfur_dioxide_t-5		0	9
NO_t-0	8		
NO_t-1	3		
NO_t-2	5		
NO_t-3	5		
NO_t-4	5		
NO_t-5	10		
Rainfall_t-0	133	430	2188
Rainfall_t-1	82	773	1667
Rainfall_t-2	23	13	81
Rainfall_t-3	39	16	272
Rainfall_t-4	6	14	53
Rainfall_t-5	12	10	22