UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Master's Thesis of

Máster Universitario en Ingeniería y Ciencia de Datos

Prediction of the noise pollution in Barcelona and model explainability using SHAP values

Marc Jordà Mascaró

Directed by: José Luis Aznarte Mellado

Academic Year: 2021-2022: September call

Acknowledgements

To my thesis director Dr. José Luis Aznarte Mellado, for having accepted to supervise a work about a topic from my choice, for having suggested ideas about the content, methods and planning, and for having shared a lot of insightful bibliographic references.

To the Environmental Assessment and Management Department of the Environmental Quality and Energy Services Management of the City Council of Barcelona, for having provided the necessary data that made possible this research project, and also for having kindly answered my multiple questions about it.

To my parents, for the unconditional support, for making my life so much easier and for their huge understanding when difficulties showed up.

To my sister Sara, because she is the brightest light when I need to clear my mind.

To my closest co-workers, who have shown a very positive attitude about my project and have given me very good ideas to carry through this research.

To my friends, who constantly cheered me up during this whole process and are eager for me to take a break nearly as much as I am.

Thank you very much.

Abstract

Noise pollution is the second most important environmental risk factor for health in Western Europe. It affects a large amount of people, it can cause a wide range of serious illnesses, and it is estimated to be the reason for 12000 premature deaths in Europe every year.

Barcelona is above the 75th percentile of European cities exposed to harmful road traffic noise levels, and it is one of the most affected by nightly leisure noise. Several initiatives have been recently developed to address this problem, following the European regulations on this matter. The city provides a network of sensors to collect noise data at every minute all over the territory.

We use noise data from 2017 to 2021 from a significant point of Barcelona. We process this information to transform it into an appropriate input for machine learning models, handling the missing values with the Prophet algorithm. Our multivariate time series problem is the following one: predicting the hourly noise values of the following 10 hours based on the previous 48 hourly values of noise and the values of weather and seasonal variables from the last hour.

We compare different modelling approaches, all of them introduced with a theoretical framework. On the one hand, we use AutoML tools, such as TPOT and Keras, to determine optimal models for our problem. On the other hand, we manually tune Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), designed to perform well on long sequences of data. A manually tuned neural network combining RNN, LSTM and GRU layers outperformed all the other approaches with an average test RMSE of 3.412 dB(A) over all prediction horizons.

Neural networks, though, are often considered black boxes, because they are so complex that it is very hard for the developers to justify the decisions they make. Therefore, in this work there is a theoretical introduction about the explainability of machine learning and deep learning models, focused on SHAP (Shapley Additive explanation) values.

The Deep SHAP method is used to calculate the importance of the features on the predictions of the RNN-LSTM-GRU model. The feature with the highest contribution to the output is a seasonal variable informing the hour range of the day, followed by the noise in the three most recent hours.

Keywords: Noise pollution, Time Series, Barcelona, AutoML, LSTM, GRU, SHAP

Contents

Nomenclature

L_{Xeq} Equivalent continuous weighted sound pressure level, page 10

L_{eq} Equivalent continuous sound pressure level, page 9

BPTT Backpropagation Through Time, page 30

DALY Disabled Adjusted Life Years, page 2

DeepLIFT Deep Learning Important Features, page 43

END Environmental Noise Directive, page 4

GDPR General Data Protection Regulation, page 11

GRU Gated Recurrent Unit, page 32

LSTM Long Short-Term Memory network, page 30

MSE Mean squared error, page 38

NAS Neural Architecture Search, page 29

RMSE Root mean squared error, page 38

RNN Recuurent Neural Network, page 29

SHAP Shapley Additive explanation, page 42

SPL Sound pressure level, page 9

TPOT Tree-based Pipeline Optimization Tool, page 28

List of Figures

List of tables

Chapter 1

Introduction

1.1 Context

1.1.1 Noise pollution

Environmental noise is a pollutant which highly affects the health and well-being of many human beings and wildlife in general. It is the second most important risk factor for health in Western Europe, right after air pollution. Therefore, in the last years noise pollution has become a huge social concern, specially in urban areas, for both citizens and policymakers.

To give some perspective about the effect of noise pollution, at least 20% of the European Union population live in areas in which traffic noise levels are considered harmful to health. One out of three people in Europe are annoyed by noise during the day, and one out of five suffer from sleep disturbance due to traffic noise. There is evidence that long-term exposure to high noise levels increases the risk of developing cardiovascular diseases. These reasons explain why noise pollution is currently not only treated as an environmental nuisance, but also as a threat to public health.

Namely, it is estimated that 113 million European individuals are affected by long-term traffic noise levels above 55 dB(A). Besides, there are 22 million people who suffer chronic high annoyance and 6.5 million people with chronic high sleep disturbance because of nightly noise levels above 50 dB(A). For example, in Spain, 27.7% of the population suffer noise levels higher than 65 dB(A) and 30.5% of Spanish homes reported to be annoyed by sounds coming from outside their housings.

Since 2002, the *Environmental Noise Directive* (END) considers noise pollution as a general problem affecting most of the European citizens. Reducing environmental noise is one of the key goals under the *Seventh Environment Action Programme* (7th EAP) developed in 2013. However, unlike other stressors such as air pollutants, which are declining, the number of people exposed to high levels of noise has remained stable since 2012 and is projected to increase due to the urban growth and an increasing demand for mobility.

1.1.2 Health risks

In the past, noise pollution was related to health issues just for people exposed to high noise levels in their work environment, and the focus was just on hearing impairment. Afterwards, those people were found out to be affected by sleep disturbance, stress, headaches, cardiovascular diseases and, in consequence, a higher mortality risk. Later studies showed that people with a long exposure to lower noise levels could have similar health issues. Thus, environmental noise is not just a work problem affecting certain reduced groups, but a public health concern affecting millions of people.

In Europe, long-term exposure to environmental noise causes a chronic high annoyance to 22 million people and a chronic high sleep disturbance to 6.5 million people. Every year, noise pollution is estimated to trigger 48000 new cases of ischaemic heart disease and to be the main reason of 12000 premature deaths. Besides, aircraft noise impacts on school learning impairment of 12500 children.

The *World Health Organisation* (WHO) employs a measure called *Disabled Adjusted Life Years* (DALYs) to quantify the burden of disease from environmental noise. It combines the years of life lost due to premature mortality with the ones lived in states of poor health or disability. It is estimated that DALYs lost by noise pollution are 437000 years for sleep disturbance, 453000 years for annoyance, 156000 years for heart disease and 75 years for cognitive impairment of children. Trying to avoid some double counting, the results tentatively show that 1 million years of healthy life are lost every year due to health issues related to environmental noise.

As the auditory system is analysing acoustic information all the time, even sleeping, arousal of the autonomic nervous and the endocrine system can repeatedly generate temporal changes and biological responses, causing health effects in the long run.

1.1.2.1 Cardiovascular diseases

Epidemiological studies illustrate a higher risk of ischaemic heart disease (related to artery obstruction), hypertension, high blood pressure and myocardial infarction for people exposed to long-term high road or air traffic noise levels. Some studies set different thresholds of the noise level from which the health risk is increased. For example, 65 dB(A) for ischaemic heart diseases, 50 dB(A) for myocardial infarction and 55 dB(A) for high blood pressure. All these chronic effects, of course, can lead to premature mortality.

Research suggests that the risk increase for a noise-induced cardiovascular disease is generally moderated. However, it becomes quite important because of the large number of people experiencing that risk and because the noise levels of exposure keep growing.

1.1.2.2 Cognitive impairment in children

More than 20 studies have proved children to show difficulties on reading comprehension, memory and attention when they suffer from high levels of noise either at school or home.

One of these studies took place in Munich when an airport was relocated in 1992. Before the relocation, 9- to 10-year-old children with schools next to the airport presented deficits in long-term memory and reading comprehension. Those effects disappeared, though, two years after the closure of that airport, suggesting that cognition issues might be reversible if noise exposure ceases. Those same effects, though, were developed by the children close to the new airport location after two years.

Another study, in Barcelona, showed that the exposure to traffic noise inside the classrooms can be related to the increase of hyperactivity symptoms and attention deficit disorders.

1.1.2.3 Sleep disturbance

Noise from the environment can affect sleep by means of repeated arousal responses, sleep stage changes or awakenings, causing sleep fragmentation. This implies cognitive function deterioration, daytime performance and a higher risk of accidents. At long-term, it can end up with chronic sleep disturbance.

Under nocturnal noise exposure between 30 dB(A) and 40 dB(A), modest effects are observed to increase: body movements, awakenings, self-reported sleep disturbance and arousals. Between 40 dB(A) and 55 dB(A), adverse health effects are reported, with people having to adapt their lives to cope with noise at night. Over 55 dB(A), the situation is considered dangerous for public health, since annoyance and sleep disturbance become frequent, and the evidence of cardiovascular diseases increases.

1.1.2.4 Tinnitus

Tinnitus is defined as the sensation of sound in absence of external sound sources. It can be caused by an excessive noise exposure. It can cause sleep disturbance, cognitive effects (worse attention and concentration), anxiety, psychological distress, depression (even case reports of suicide), communication problems, frustration, irritability, tension, inability to work, reduced efficiency and restricted participation in social life.

Hearing impairment is expected to happen above 75 dB(A), with leisure noise as the main source, since it typically exceeds this threshold.

1.1.2.5 Annoyance

Noise annoyance is a term used to define the multifaceted answer of stress and discomfort caused by the noise pollution. People annoyed by noise can experience a wide range of negative effects such as anger,

disappointment, dissatisfaction, withdrawal, helplessness, irritation, depression, anxiety, distraction, insecurity, agitation, exhaustion, personality disorder, mental disorder or suicide.

Exposure category $L_{den}(dB(A))$	Percentage of population exposed	Percentage of population highly annoyed	Number of cases per million
< 55	50	2.77	13835
55-59	17	8.16	13868
60-64	19	12.96	24621
65-69	9	20.08	18068
70-74	4	30.25	12100
> 75	1	30.25	3025
Total	**100**		**85517**

Table 1.1: Highly annoyed people due to road traffic noise in the European Union

Finally, there is also scientific evidence regarding the harmful effects of noise pollution on terrestrial and marine wildlife by reducing reproductive success and increasing mortality and emigration, resulting in lower population densities.

1.1.3 Environmental noise policy context in Europe

In the European Union, the *Environmental Noise Directive* (END) is the legislative framework created to achieve noise reduction. This law presents a common approach to prevent exposure to environmental noise by requiring the member states to:

- Produce strategic noise maps every 5 years for all major roads, railways and airports, using common noise indicators.
- Determine the number of people exposed to every noise source above the thresholds considered to be harmful to health, which are 55 dB(A) during the day and 50 dB(A) during the night[1].
- Prevent and reduce environmental noise in places where the exposure levels can be harmful to human health.
- Preserve quiet areas by keeping a good acoustic environmental quality.

Nowadays, action plans generally include noise reduction targets -both the number of decibels and the number of people exposed-, description of measures to achieve these goals, schedule of priorities, expected costs of the measures, number of people expected to get benefit from them and roles in charge to implement

[1]It is worth to mention that the *World Health Organization* (WHO) establishes these harmful thresholds at 53 dB(A) during the day and 45 dB(A) during the night for traffic noise and does not provide a common threshold for all noise sources.

these measures.

Although some progress has been made on the data management, the implementation of these measures has been too delayed. Besides, countries should use the same assessment methods. Actually, the number of people exposed to high levels of noise (65 dB(A) during the day and 60 dB(A) at night) remained stable between 2012 and 2017. However, in the places where measures were applied, the benefits outweighed the costs.

1.1.4 Barcelona

In comparison to other European cities, Barcelona is found above the 75^{th} percentile of cities with most exposed population to harmful noise levels. The context is even worse for noise exposure at night, since it is in the highest tiers of cities with population exposed to excessive nightly noise.

Road traffic is by far the main source of noise in Barcelona. 57% of the population are exposed to a traffic noise level higher than the threshold that the *World Health Organization* (WHO) considers harmful to health, both during the day (53 dB(A)) and the night (45 dB(A)). Namely, a 27% of the population suffer noise much higher than these thresholds (more than 65 dB(A)).

Leisure noise is mainly concentrated on Friday and Saturday nights, affecting 3% of the population, who is exposed to noise levels above 50 dB(A). There are some areas, though, such as *Ciutat Vella*, which register annual averages higher than 60 dB(A) or even 70 dB(A) during these weekly periods. At minor rates, noise from rail traffic only affects the 0.65% of the citizens and industrial or aircraft noise can be negligible inside the city.

The *Health Survey of Barcelona* in 2016 reported that 47% of the respondents agreed that their neighbourhood was very noisy, and 26% of them claimed their homes to be noisy from the outside. It is estimated that in Barcelona 217000 people (16% of adults) suffer from severe noise annoyance and 60800 people (4% of adults) suffer from severe sleep disturbance because of noise. The long-term exposure to road traffic noise would cause 300 new cases of ischaemic heart diseases and 30 deaths per year could be attributed to noise pollution.

1.1.4.1 System for surveillance of the health impact of environmental noise

In this context, the *Barcelona Public Health Agency* created the *System for surveillance of the health impact of environmental noise* in Barcelona in order to assess, control and communicate the exposure, perception and health risk of noise pollution. These are their main goals:

- Raising awareness about this environmental factor

- Comparing the impact of noise pollution to other pollutants and to the noise pollution in other territories
- Monitoring the health impact of the noise pollution in the city[2]
- Identifying vulnerable groups
- Designing measures to control the environmental noise and assessing the policies to control the traffic in the city

It is highly recommended to drastically decrease the road traffic in Barcelona, and also try to minimise the noise generated by the vehicles, either by using electric cars or covering the roads appropriately. The most affected areas, such as *Eixample*, and the surroundings of schools should be prioritised. Leisure nightly noise should be reduced as well, specially in *Ciutat Vella*. In addition, epidemiological studies related to noise pollution should be promoted in order to improve the knowledge about health risks and to find the appropriate noise thresholds for every situation.

1.1.4.2 New measures in 2022

In 2022, several new measures are being applied in order to reduce the noise pollution in Barcelona, focused on prevention, mitigation and protection:

- Deployment of the *Urban Mobility Plan* (PMU) 2024, whose goal is that 81.5% of the journeys in the city are made on foot, in public transport or by bike
- Reduction of the vehicle velocities from 50 km/h to 30 km/h
- Implementation of the Barcelona *superblocks*, which prioritise pedestrians
- Awareness-raising campaigns about the noise impact of motorbikes and also making compatible leisure and sportive activities with the rest of the neighbours
- Control of the noise levels of loading and unloading activities and works
- Noise-radar system to detect vehicles exceeding the allowed noise levels
- Acoustic limitation to live music concerts in the street
- Assessment in certain *special-regulation noise zones* (ZARE), in which night noises are too high at night. If any area exceeds 3 decibels the allowed noise levels, there will be grants to improve the quality of its housings
- Pacification of the surroundings of schools
- Use of construction techniques to mitigate the sound levels of the roads

[2]Barcelona strategic noise map is created every 5 years by the City Council to follow European regulations. Most recent edition is from 2017 (Ajuntament de Barcelona (2017))

1.1.5 *Sentilo*

In Barcelona, data about noise pollution is gathered all over the city by means of a platform called *Sentilo* which works with a network of sensors distributed at tenths of points of the territory, measuring the sound levels registered every minute.

Sentilo is a platform based on sensors designed to build the *Smart City* architecture for any city or organization based on openness and easy interoperability. It was sponsored by the *Barcelona City Council* in 2011 to define a strategy to become a reference in the field of *Smart Cities*. Currently, Barcelona holds the *Smart Cities Expo World Congress*, a global event for smart cities stakeholders, and it also leads the *City Protocol* initiative to connect global cities in different projects and to face common challenges. Barcelona has used different kinds of sensors to evaluate noise pollution, air contamination, traffic congestion and even waste management.

Smart City solutions generally let their data and logics flow across different domains. However, many sensor data solutions depend excessively on specific technologies, products or providers, making it impossible to properly share the data. They can also cause duplicity and multiplicity of data and infrastructure, which can increase investment and maintenance costs.

Thus, *Sentilo* is a cross-platform oriented infrastructure and data management service aimed for municipalities or organizations who need to process lots of data, share information among heterogeneous systems and centralise a way to distribute homogeneous data across their systems. The platform *Sentilo* is released as *open-source* and it is nowadays used by more than 10 cities and 15 companies, some of them leaders in the *Smart Cities* field.

1.1.5.1 Technical details

Sentilo is a software architecture that isolates applications using the sensor data from the layer which provides the data. It is based on *open-source* elements such as *Redis*, *MySQL*, *Mongo*, *Hibernate*, *JSON* and *JQuery*. Its architecture is based on the following components:

- A front-end for message processing
- An administration console
- A memory database with high performance rates
- A *non-SQL* database to get a more flexible and scalable system
- A universal viewer
- A statistics module with basic performance indicators
- An extensible component architecture to enlarge the platform functionality

The main features of this technology system are:

- High performance to deal with thousands of messages
- Modularity and extensibility to add functionalities
- Horizontal scalability
- Cross platform (*Java*, *Redis* and *MongoDB*)
- Simple *REST* interface to send and receive data, orders and alarms
- Triggers to generate alerts, calculations, stats or messages
- Frontend app
- Open source

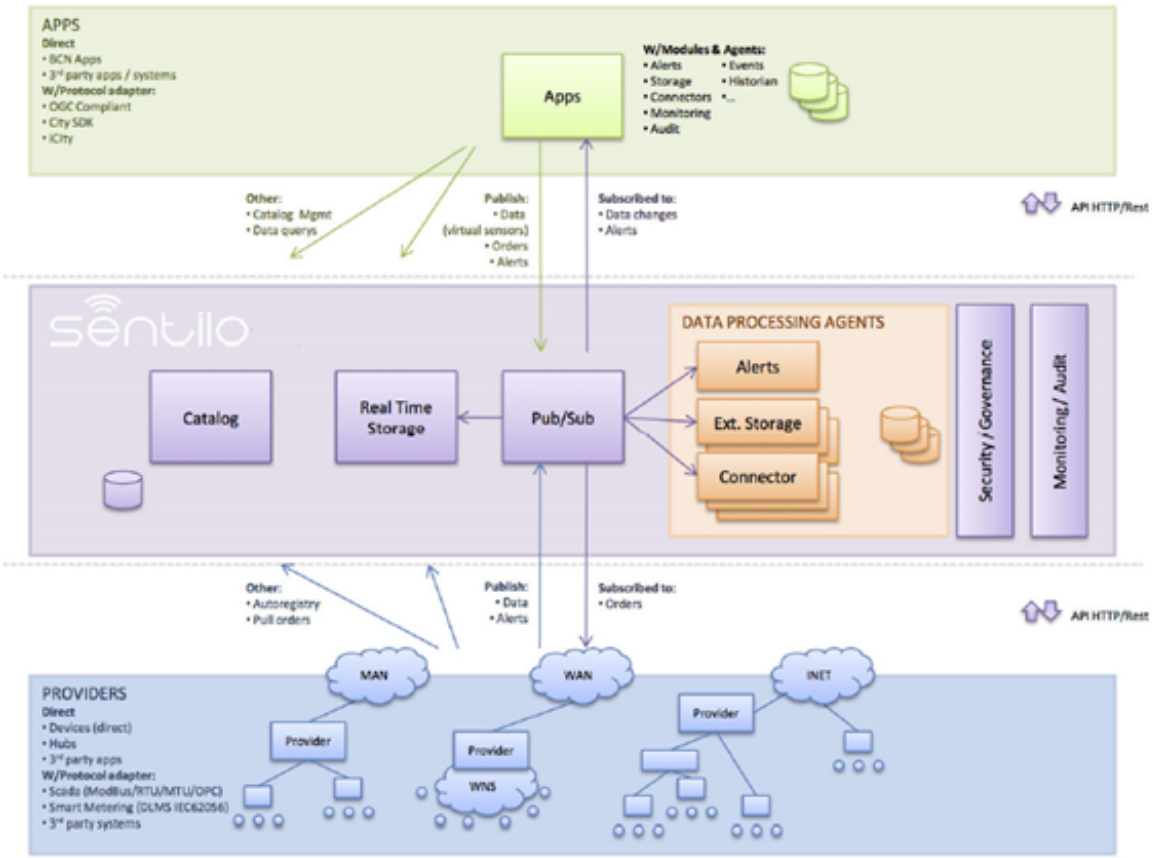

Figure 1.1: Summary of the *Sentilo* platform architecture

1.1.6 Measurements

This section describes the sound and its measurements from a physical perspective. It will be useful to understand later what is the meaning of the noise data gathered in this work.

Noise is an acoustic signal made up of sound signals with different frequencies that vary their energy according to time. It is formed by mechanical waves travelling through the air and causing slight pressure changes in it. Since the auditory system works in a logarithmic way, sound pressure levels are quantified by means of a logarithmic relation between an average value and a benchmark value. This relation is called *decibel* (dB). Human beings can hear noises from 0 dB to approximately 140 dB.

Since not every frequency is perceived with the same intensity, we use equal-loudness contours to represent the sound pressure levels of equal loudness for every frequency in the audible range. The *Fletcher-Munson curves* were the first ones to be calculated, in this case experimentally.

Noise measurements are usually weighted in terms of frequency to adapt the environmental data to what is experienced by the auditory system. The most common weighting filter is the A filter, resulting from the *Fletcher-Munson curve* of 40 phons. Namely, the expression to weight the different frequency components of an acoustic signal is the following one:

$$A(f) = 10\log\left[\frac{1.562339^2 \cdot f^4}{(f^2+107.65265^2)\,(f^2+737.86223^2)}\right] + 10\log\left[\frac{2.242881^{16} \cdot f^4}{(f^2+20.598997^2)\,(f^2+12194.22^2)}\right], \tag{1.1}$$

where f is the frequency in Hz.

Given this context, we can present now the most used noise indicators.

1.1.6.1 Sound pressure level (SPL)

The *sound pressure level* (SPL) obtains the intensity level generated by an instantaneous pressure sound and it is defined as:

$$NPS = 10\log_{10}\frac{p^2}{p_0^2}(dB), \tag{1.2}$$

where p is the sound pressure and p_0 is the minimal pressure variation that human beings can detect at 1 kHz, equal to 20 μPa.

1.1.6.2 Equivalent continuous sound pressure level (L_{eq})

Noise is usually assessed by averaging pressure levels during certain time intervals. The *equivalent continuous sound pressure level* (L_{eq}) is mathematically defined as:

$$L_{eq,T} = 10\log_{10}\left(\frac{1}{T}\int_0^T \frac{p(t)^2}{p_0^2}\mathrm{d}t\right), \tag{1.3}$$

where $p(t)$ means the square root of the instantaneous sound pressure level and T is the time interval. This indicator expresses the average noise energy during a certain time interval.

1.1.6.3 Equivalent continuous weighted sound pressure level (L_{Xeq})

It is based on the equivalent continuous sound pressure level, but it also uses a weighting filter. It is usually used to create noise maps. Mathematically, it is defined as:

$$L_{Xeq,T} = 10\log_{10}\left(\frac{1}{T}\int_0^T \frac{p_X(t)^2}{p_0^2}\mathrm{d}t\right), \tag{1.4}$$

where $p_X(t)$ is the square root of the instantaneous weighted sound pressure level. It is measured in dB(X), where X is the type of filter.

Hence, $L_{Aeq,T}$ is the equivalent level of the average A-weighted sound energy within a time interval T. This is the recommended indicator to measure continuous road traffic or industrial noises, since it allows to quantify the acoustic energy during certain time periods.

1.1.6.4 Maximal noise index

To evaluate contexts with a few discrete events, it is recommended to calculate the *maximal A-weighted noise index* ($L_{A,max}$), which is the highest A-weighted sound pressure level registered during a certain period. For instance, it is an appropriate indicator to assess the noise-induced sleep disturbance.

1.1.6.5 Long-term indicators

In order to assess the environmental noise, European laws usually recommend to average sound level pressure indicators at long-term. The most common indicators are L_d, L_e and L_n, defined as the average sound level pressure calculated during all the morning periods (7h-19h), afternoon periods (19h-23h) and night periods (23h-7h) of a whole year, respectively.

As a metric combining the former three, L_{den} is a noise indicator that works out L_{Aeq} in a 24-hour period, penalizing 5 dB(A) the evening noises and 10 dB(A) the nightly ones. Its expression is the following one:

$$L_{den} = 10\log_{10}\left[\frac{1}{24}\left(12\cdot 10^{\frac{L_d}{10}} + 4\cdot 10^{\frac{L_e+5}{10}} + 8\cdot 10^{\frac{L_n+10}{10}}\right)\right] \tag{1.5}$$

1.2 Motivation and goals

Noise pollution is an environmental risk factor for the health and well-being of the population, increasing the chance to develop several serious illnesses that even cause premature deaths every year. These health issues and discomfort affect mostly people living in urban areas. Environmental noise has not been studied as in depth as air pollution, for instance, but recent studies are increasing the concern from citizens and policymakers.

Barcelona is found in the highest quartile of European cities with most population affected by harmful noise levels, and it is one of the most impacted by nightly exposure to noise pollution. Recently, several ini-

tiatives have been developed to address this issue in the city, following European Union general regulations. This new strategy includes data gathering, monitoring and descriptive analysis through noise sensors, public health surveys about this topic and an action plan in 2022 focused on prevention, mitigation and protection.

Time series problems have been traditionally dealt with classical statistical methods such as ARIMA. In the latest years, machine learning and deep learning techniques have started to be used as well, because they make it easier to provide additional information to train the models beyond the target, achieving more accurate predictions. These models can even be automatically selected by AutoML tools recently created to optimise and automate the search for the optimal method for the data that is being used. In addition, concrete deep learning techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs) have been specifically designed to work with very long sequences of values, managing to keep the track of the past information without letting it vanish from the optimisation process.

Although deep learning models provide results which usually outperform simpler models, they are considered as black boxes. This means that they are so complex that it is very hard for the developers to justify the decisions they make. This issue is very problematic because users are normally not keen on employing methods they do not understand or trust. This is also raising awareness among policymakers, especially with the EU General Data Protection Regulation (GDPR) legislated in 2018, which makes compulsory to explain automated decisions to any interested third parties affected by them. There is a broad debate within the Artificial Intelligence (AI) field about the proper way to address these problems.

A better understanding of a model can help us to comprehend the reason behind its predictions and prevent it from failing in the future. However, at a technical level, model explainability is still a recent research field with huge margin for improvement. Most used techniques involve approximating the models at a local level with other simpler -and so, more interpretable- models or assigning values to all the features depending on their importance for every prediction. Among all of them, SHAP values stand out as a very common approach to give insights about model predictions.

Having stated the main motivations for this research, its main goals are the following ones:

- Process the data provided by noise sensors in Barcelona properly, combined with some other explicative variables, so that it can be used as input for machine learning and deep learning models.

- Choose the best modelling approach by comparing the optimal choices from AutoML tools with other specific neural networks -RNNs, LSTMs and GRUs- that, theoretically, are specially designed to perform well in multivariate time series problems.

- Provide an explanation for the predictions of the best model by using SHAP values and give insights to understand what are the features that have the highest influence on them.

1.3 Literature review

The field of research about noise pollution has usually been related to the use of traditional time-series methods, despite in the latest years several models of machine learning started to deal with this problem as well. Even more recently, complex deep learning models such as *Recurrent Neural Network*s (RNNs), *Long Short-Term Memory networks* (LSTMs) and *Gated Recurrent Units* (GRUs) have arisen as the most accurate models to study very long sequences of data. Papers about this topic are generally considered from two perspectives: either they try to correlate the levels of noise pollution with adverse health effects over the population or they try to build predictive models to rise alarms about high levels of noise to rise alarms in certain industrial or urban areas.

It is quite remarkable the substantial number of studies carried out by Julio Díaz and Cristina Linares in the city of Madrid, using time-series methods to link traffic noise with daily mortality and cardiovascular, respiratory and diabetes-related diseases in the elderly. We mention some of their investigations:

- Using time-series analysis methods like ARIMA and Poisson regression, they showed that between 1995 and 1997 increases of road traffic noise levels by 1 dB(A) during the day were related to a rise of admissions in the *Hospital Gregorio Marañón of Madrid* by 5.1%, excluding accidents and labours, regardless the amount of air pollutants.

- Using Case-Crossover time series analysis from 2003 to 2005, they showed that the rise of the road traffic noise levels by 1dB(A) was related to the increase of the relative risk (RR) of mortality due to cardiovascular diseases by 6.6% for people over age of sixty-five. A similar study conducted from 2001 to 2009 proved that increasing daily noise levels by 0.5 dB(A) could increment myocardial infarctions by 7% and mortality due to cardiovascular diseases by 3% for this group of people.

- Using Case-Crossover time series analysis from 2003 to 2005, they also showed that an increment of 1 dB(A) in road traffic noise levels could rise the risk of mortality by respiratory causes by 4% for people older than sixty-five, regardless the chemical pollution. Similarly, data collected from 2001 to 2009 allowed to prove that this same group of people suffered an 8% more of pneumonia and 3.5% more of mortality due to respiratory issues when noise levels were 0.5 dB(A) higher.

- From 2001 to 2009, nightly noise levels increments of 0.5 dB(A) were statistically significant to explain a rise of 4% of mortality due to diabetes for the whole population.

- From 2003 to 2005, they researched about the environmental causes of the mortality for people over sixty-five years old. Results indicated that every year 312 defunctions could be avoided if the average noise levels were lowered by 0.5 dB(A), objective that could be accomplished by having a 12% of electric cars. These conclusions were regardless the effects by air pollutants.

Other researchers have developed complex machine learning and deep learning models to predict noise levels in the recent years. For example, S.K. Tiwari, L.A. Kumaraswamidhas, C. Gautam and N. Garg presented in 2022 an ensembled model based on an auto-encoder concatenated to a LSTM to solve the univariate time-series environmental noise prediction in 7 Indian cities. They transformed the time-series data

into supervised learning format (input-output pair), also min-max normalizing all variables in the interval [0,1]. The proposed model could extract correlations from past day and night noise data and outperformed other techniques like support vector machines (SVM), RNNs or a single LSTM, with a root-mean-square error (RMSE) of 0.563 dB(A).

On the other hand, Q. Zeng, Y. Liang, G. Chen, H. Duan and C. Li predicted in 2021 the noise levels in a chemical industry park in Shadong Province, China. They considered a multivariate problem involving wind speed, vehicle flow and close noise data based on wind direction at every instant. The proposed model combined LSTM and Prophet, using multi-station noise as regression variables for the Prophet model. They resampled the data from 30-second to 10-minute intervals and used the 3σ criterion to deal with data close to 0dB, by assuming that any point outside of the (u – 3σ, u + 3σ) interval -being u the mean value of noise and σ its standard deviation- is not a random error but a gross error, and in the paper it is replaced by the mean value. The RMSE and MAE was reduced by 32.2% and 23.3%, respectively, thanks to this technique, obtaining 0.53 dB and 0.46 dB, respectively, for the test data and outperforming traditional prediction methods.

In 2020, P.J. Wen and C. Huang analysed the noise equivalent level (L_{eq}) at the *National Synchrotron Radiation Research Center* (NSRRC) of Taiwan using a gradient boosting model (GBM). They included the previous two minutes of noise data for every sequence, and also additional features like the day of the week, the hour of the day and whether the day was a holiday, a Saturday or a Sunday. The results obtained a RMSE of predicted harmful noise minor to 1 dB(A) and a coefficient of determination (R2) greater than 0.7.

Also in 2020, J.M. Navarro, R. Martínez-España, A. Bueno-Crespo, R. Martínez and J.M. Cecilia used a LSTM to predict near-time future noise values, in terms of both pressure and loudness, for different time period ranges from 1 min to 60 min. The proposed model outperformed classical statistical models, achieving a mean squared error (MSE) lower than 4.3 dB for sound pressure and below 2 phons for loudness.

Finally, in 2014, N. Genaro exhaustively collected data in Granada and developed an artificial neural network model to predict the noise in that city, getting no instances with more than a 5% of error and outperforming classical mathematical models. The defined training process is claimed to be fast enough to be suitable for real-time uses. Besides, a principal component analysis (PCA) was carried out to select the most relevant features among 25 variables.

1.4 Structure of the thesis

In this section, we will briefly describe the most relevant parts of this work. This document is structured as follows.

Chapter 1 gives the whole background behind the problem we want to solve. We present the noise

pollution and its related health risks and the European environmental noise policy. We then focus on the case of Barcelona, presenting its particularities and how the noise data is gathered there. Besides, we also introduce the physical ways to measure the sound. Finally, we review the existing literature to give an overview of similar studies performed in the past.

Chapter 2 describes all the methods performed to gather and process noise and meteorological information to transform it in a set of data useful to become the input of machine learning models or neural networks. This includes the imputation of missing values, the aggregation and scaling of the original values, the creation of new features, the reshaping of the data to be used in a supervised machine learning framework and the management of train, validation and test sets.

Chapter 3 gives a theoretical context about the predictive approaches that will be used in this work. It introduces AutoML techniques, such as TPOT and Autokeras, and also some neural networks designed to perform well on time series problems, like Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). Afterwards, all these approaches are presented in the context of this specific case of study, describing how they are used with our data. Finally, the error metric choice is discussed, and all the performances are compared to select the optimal model.

Chapter 4 provides a theoretical framework to discuss model explainability and specifically SHAP values are introduced. Then, these are applied to the best detected model to provide insights about the way predictions are made.

Finally, chapter 5 discusses the entire work, providing conclusions as a review. It also describes next steps that could be performed to improve and extend the scope of this study.

Chapter 2

Data sources and processes

2.1 Data sources and data gathering

Any project in the field of data science mainly relies on the possibility to get appropriate data sources with good data quality. The success of machine learning or time series models mainly depend on the availability of suitable data rather than the choice of the algorithm or the parameters. Thus, to develop a model capable of predicting the noise pollution in Barcelona, it became crucial to find the appropriate data series to work with.

2.1.1 Noise Data

Information gathered by Sentilo platform described in the section 1.1.5 can be consulted at real time in a webpage (Ajuntament de Barcelona (2013)) based on an interactive map that shows all the available sensors. Once a sensor is selected, the latest measures are graphically shown, among some other general statistics. This interface, despite being quite user-friendly, is not useful in terms of academic research because any model dealing with this prediction problem needs a long sequence of data to be trained on.

With the purpose of obtaining a long enough sequence of environmental noise data, we contacted the Environmental Assessment and Management Department of the Environmental Quality and Energy Services Management of the City Council of Barcelona to request data from four different sound stations of the city during five consecutive years, namely from 2017 to 2021, for academic research purposes. The four sensors had the same brand (CESVA TA120) and the selection process took into account three main criteria:

- The initial date of the data series informed on the website had to be prior 2017
- The sound sensor had to be relatively close to a weather station to have the possibility to develop a multivariate time series problem involving meteorological features
- The location of the noise station had to be meaningful for some reason

Taking all these considerations into account, the four requested data series were in the locations shown in Table 2.1.

Location	Sensor code	Singularity
Lleó Street, corner with Isaac Newton Street	TA120-T241007	Next to a motorway
Maquinista Street with Joan de Borbó Avenue	TA120-T241005	Close to Camp Nou (F.C. Barcelona stadium)
Passeig de Gràcia streetlight with AP22	TA120-T240426	City Centre + road traffic
Passeig de Sant Joan streetlight with AP-PSJ-01 T2, Llobregat side	TA120-T240992	City Centre + road traffic

Table 2.1: Noise sensors where we requested data from.

In fact, the main goal is to develop a model just for one of those locations. The reason to ask for more choices was to have some backup in case one of them turned out to be infeasible due to lack of information or poor-quality data.

After iterating communications during a month to wait for certain administrative issues and extraction processes, the data was finally shared. The submission of the data was done in a compressed folder, with a directory for every sensor containing one csv-format file per day of collected data. Every file contained noise level entries for every minute within that 24-hour period. Noise was measured using the equivalent continuous A-weighted sound pressure level ($L_{Aeq,T}$) indicator, which measures the equivalent level of the average A-weighted sound energy within a minute and it is recommended to measure continuous environmental noise, as explained in section 1.1.6. Barcelona City Council also confirmed then that the first two stations (TA120-T241007 and TA120-T241005) did not have data available for 2017 and most of 2018, so they had to be discarded from the beginning.

It is worth to mention that the public administration discards any data entry when it is raining, without replacing it by anything else. The reason to do so, according to their answer, is that they are usually interested in the noise generated by a certain source, like the road traffic or the leisure events, so the rain supposes an alteration to this analysis, for example by modifying the noise of the tyres in the road. They claimed, though, that the measures from the sensors when it rains were equally valid. As in this project we want to study the noise pollution from a complete point of view, considering all the sources, we decided to keep the measures within rain periods because they were simply another factor of noise.

2.1.2 Meteorological variables

To potentiate the power of the model and the completeness of this study, we decide to use a multivariate approach, which means that some other features beyond the target are added to improve the capabilities of the model and to make the interpretability of the model easier to understand.

When dealing with environmental prediction problems, one of the most common choices is to use meteorological variables, since, despite not being especially noisy by themselves, they highly affect all the activities in the city, including the noisiest factors of Barcelona like the road traffic and the leisure activities.

Despite the existing weather stations of Barcelona are integrated within the *Sentilo* network, there was no need to request this data to the public administration because the meteorological data can be found near real-time updated in the Open Data Catalogue from the Catalan government (Portal de dades obertes de la Generalitat de Catalunya (2019)). This data, at National level, is part of the *Automatic Weather Stations Network* (XEMA) from the *Meteorological Service of Catalonia* (Meteocat). There was a single available file which was quite heavy -more than 30 GBs- because it contained data from all the 233 weather stations of Catalunya from 2009 to 2022. In section 2.2, we will explain how we treated such a big file.

As mentioned before, all the noise sensors were chosen to be found relatively close to some weather station in Barcelona. As we already discarded two out of the four sensors due to lack of data, the weather stations close to the remaining noise sensors were the ones described in Table 2.2:

Location	Weather Station Code	Close noise sensors	Number of available variables
Zoo	X2	TA120-T240426 TA120-T240992	6
El Raval	X4	TA120-T240426 TA120-T240992	16

Table 2.2: Weather stations close to the requested non-discarded noise sensors.

Considering that both potential noise sensors were relatively close to either weather station, the big difference in the number of available variables and the fact that the six variables included for the X2 station were also included in the X4 one, there was no doubt in using the variables coming from the weather station located in *El Raval* (X4). The variables are registered in 30-min time intervals, so they need to be aligned with sensor data at the preprocessing stage. The variables informed for X4 weather sensor are listed in the Table 2.3.

Having described the data sources that were used to develop this work, in section 2.2 we will describe the methods used to clean and transform them in order to get a suitable dataset which can be used by prediction models.

2.2 Data processing

In projects involving the use of data, it is not only important to count with the appropriate sources, but also to transform their data into a proper format to provide the models information they can understand. Tasks such as cleaning, transforming and complementing the raw data are performed during the preprocessing stage. In this case, all the procedures were performed using Python 3.8 in JupyterLab.

Variable name	Original variable code	Description	Units
temperature	32	Average temperature	ºC
temperature_min	42	Minimum temperature	ºC
temperature_max	40	Maximum temperature	ºC
rain	35	Precipitation	mm
rain_1min_max	72	Maximum precipitation in 1 minute	mm
relative_humidity	33	Relative humidity	%
relative_humidity_min	44	Minimum relative humidity	%
relative_humidity_max	3	Maximum relative humidity	%
atm_pressure	34	Atmospheric pressure	hPa
atm_pressure_min	2	Minimum atmospheric pressure	hPa
atm_pressure_max	1	Maximum atmospheric pressure	hPa
irradiance_global	36	Solar irradiance	W/m^2
wind_speed_10m	30	Wind speed at 10m height	m/s
wind_gust_10m_max	50	Maximum wind gust at 10m height	m/s
wind_10m_direction	31	Wind direction at 10m height	º
wind_gust_10m_max_direction	51	Direction of the maximum wind gust at 10m height	º

Table 2.3: Meteorological variables included in the X4 weather station.

2.2.1 Noise variables

As explained in section 2.1.1, noise data was delivered in a compressed folder, with a directory for each requested sensor and a file for every day of collected data. In those files, there were some rows with metadata information and then one sound measure per minute. At first, some adjustments to the structure of the delivery needed to be done. Some concrete files were renamed due to wrong naming structure. Some other files were placed in the wrong folders and were deleted. The information provided related to the rain registers was discarded as well because, as explained in the section 2.1.1, we decided to keep all the measures and we also had the rain information from another source. Finally, some adjustments were considered since some files contained additional empty columns.

The files, besides the time and the noise measure, also contained two more columns called overload and underange. The former identified a wrong register due to the presence of a noise peak and the latter meant that the noise was under the threshold of the device capabilities, so it was unfeasible to determine it. Nevertheless, since those two columns were set to False in the whole set of available data, they were just discarded.

The mechanism to join all the data coming from different files within a certain folder was based on creating a base structure indexed with all the minutes from 01/01/2017 to 31/12/2021 and then iteratively reading the useful content of the files and keep attaching the measurements to the structure. The procedure also removed duplicated data because there was some overlapping content in certain files.

2.2.1.1 Aggregation

One of the most important decisions made during the preprocessing stage was aggregating the measures, originally in 1-minute time period, to a 1-hour time interval. This logic relied on the fact that, functionally, it seems not useful to have a model that can predict the noise levels of the next minutes in a city, whereas forecasting for the next hours allows the possibility to the local administration to apply certain temporary policies to mitigate this effect or to protect the well-being of the people living in those locations. In addition, having a sequence of 1-minute data for 5 years could have been too heavy in terms of storage and computational time.

Another key decision was the way of aggregating the data. Our two main options were computing either the maximum or the average of all the minute-by-minute measures. Despite one might expect the averaged data be much smoother, maybe being able to achieve better prediction results, globally it showed similar standard deviation parameters than the maximized data, although it is true that the latter presented more peaks at very concrete instants of time. However, the maximum aggregation was much more understandable in our business use-case. If we are trying to detect noise levels that can be upsetting to the population within the next hours, it is sensible to focus on the noise peaks we can detect in that period. Using the average aggregation could certainly mitigate situations in which we have just very concrete undesirable peaks of noise surrounded by more normal sound levels, so we would be missing valuable information. This reasoning led us to aggregate the data using the maximum sound level detected in the minute-by-minute measures during that hour.

2.2.1.2 Missing values

Data collecting processes can be affected by specific technical malfunctions which temporarily prevent them from generating new entries. Devices can even be momentarily unavailable due to software updates. In the case of our noise data, we found some gaps in the time sequence, sometimes even affecting a whole day. Namely, the percentage of informed minutes was 91.97% for the TA120-T240426 device and 94.45% for the TA120- 240992 device. This significant difference was the reason to decide to use for our model the sensor TA120- 240992, located in Passeig de Sant Joan streetlight with AP-PSJ-01 T2, Llobregat side.

The problem of handling the missing values in a sequence of time series data has no obvious solution. The seasonal pattern of the entries, together with the fact that sometimes we had a whole day without any information, makes the most common methods practically useless, because we couldn't just inform the null values with some linear interpolation, forward fill or backward fill, because we wouldn't be respecting the properties of the data. Non-linear interpolations were not working either.

A good alternative we found to deal with missing values is Prophet, the time-series forecasting module for Python and R created by Facebook, which is very robust to missing data, shifts in trend and outliers. One good thing about the models returned by Prophet algorithms is that they do not only generate fore-

Figure 2.1: Location of the data sources in the city of Barcelona, labelled accordingly.

casting values in the future, but they also apply the model learned from the data to the past, so it is capable to plot values in instants that were not originally informed in the input data. To exemplify how this method works, we show in Figure 2.2, in blue, the sequence of data from 15/01/2021 to 01/03/2021. We see that there were three gaps of information within that period. In orange, we can check how Prophet handled these missing values. It is true that it is not feasible for the Prophet model, without any regressors. to generate noise peaks by itself, but it successfully respects the seasonal pattern of the data, outperforming other techniques mentioned before.

2.2.1.3 Lagged variables

If we want to use machine learning or deep learning models to predict the noise pollution in Barcelona, the time series problem must be interpreted as a supervised learning problem. To do so, we need to reshape the available sequence of time series data into pairs of input and output shorter sequences. On the one hand, time series problems are based on a long list of values ordered by a time index. On the other hand, a supervised learning problem uses input and output patterns, such that a model can learn how to predict the output patterns from the input ones.

For instance, as a very basic example, we could use the observation from the last time step, $y(t-1)$, as the input to predict the observation at the current time step, the output $y(t)$. In practical terms, we use the *shift()* function in *pandas* to push the values some steps backwards to generate the new inputs.

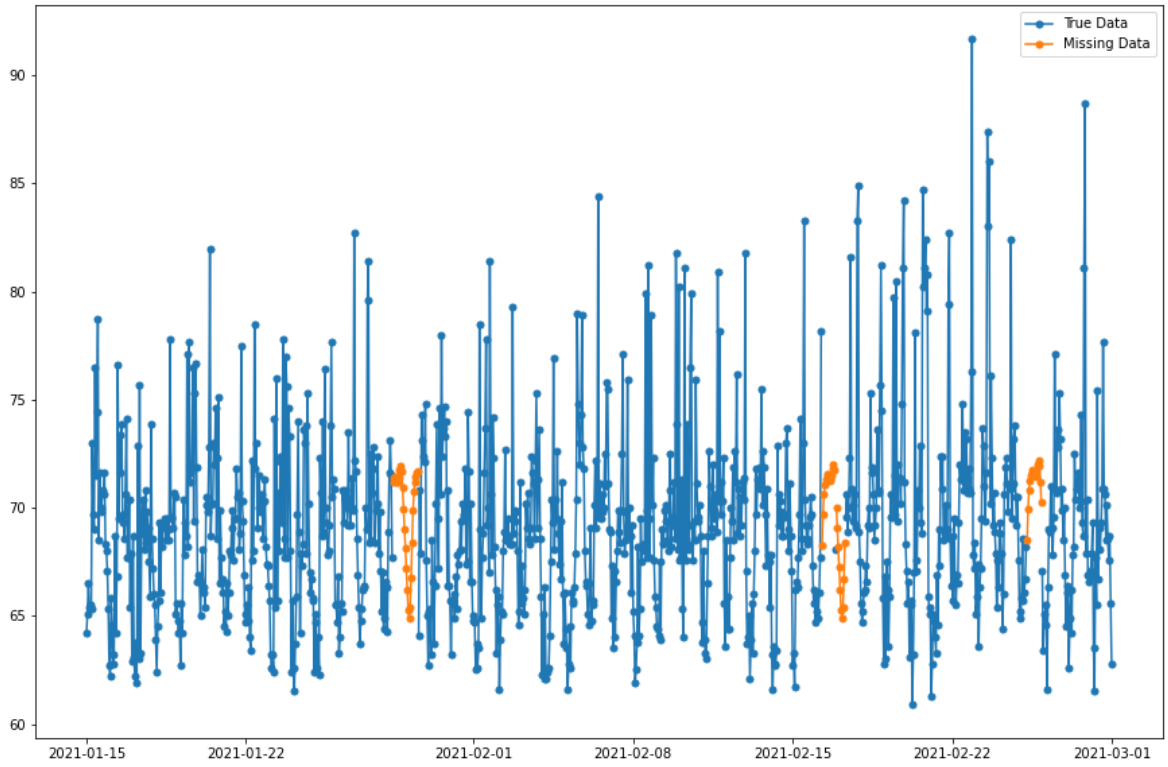

Figure 2.2: Imputation of missing noise values from 15/01/2021 to 01/03/2021. In blue, the values taken from the data source. In orange, the missing values computed by the Prophet algorithm.

For every instant, these former values that happened immediately before and that can be used as input are normally called lagged variables. In our case, we decide to use the past two days of noise data, plus some other features at present, to predict the noise level in the future. Therefore, we will use 48 lagged variables in total, from $y(t-48)$ until $y(t-1)$.

The same procedure used to generate the past lagged values of noise data can also be used to generate future values. Hence, the problem can predict not only the noise at the current time, but at a certain longer future horizon. The output doesn't need to necessarily be a single target value, but a target sequence of noise data. For example, in our case the model is intended to predict the noise pollution for the next 10 hours, with a target from $y(t)$ until $y(t+9)$.

2.2.1.4 Additional note about noise above thresholds

To understand how necessary it is to limit the exposure of the population to the noise pollution, we can check the ratio of values in our data that are above the noise thresholds considered harmful to health. According to the END, these reference values are 55 dB(A) during the day (from 8h to 22h) and 50 dB(A) at night (from 23h to 7h). If we look at the original measures, taken every minute, a 94.1% of them are over these thresholds. If we do the same exercise but with the data aggregated at hourly level, the 100% of the values are considered harmful to health.

2.2.2 Meteorological variables

As we stated in section 2.1.2, the weather data source was obtained in a single file with more than 30 GBs that contained information about 233 weather stations in Catalunya from 2009 to 2022, with variables informed on a 30-minute basis. Besides the difficulty of storage of such a big file, we needed an approach to efficiently read from this source and selecting just the entries from the X4 weather station from 2017 to 2021.

In order to deal with such problem, we use *dask*, a Python's library based on *pandas* which is designed to work with very big datasets by performing computations in parallel before finally transforming the objects into a *pandas* dataframe. Using *dask* functionalities, we read the original file and we applied a pipeline of transformations involving filtering the weather station, date range and variables we needed, dropping unnecessary columns and renaming some others. Similarly as how *PySpark* works, these instructions are performed with lazy evaluation, since we finally transform the dataset into a *pandas* dataframe and the whole pipeline is computed.

Later, we pivoted the object to transform a row-based dataset -with an already acceptable size- to a column-based one, with one variable per column. The original codes of the variables were renamed to more understandable names, as shown in Table 2.3. Next, an aggregation was performed to transform the 30-minute time intervals into 1-hour intervals, to align these explicative variables with our target. The type of aggregation depended on each variable, and it is summarized in Table 2.4.

Type of aggregation	Affected variables
Average	temperature, rain, relative_humidity, atm_pressure, irradiance_global, wind_speed_10m
Maximum	temperature_max, rain_1min_max, relative_humidity_max, atm_pressure_max, wind_gust_10m_max
Minimum	temperature_min, relative_humidity_min, atm_pressure_min
First	wind_10m_direction, wind_gust_10m_max_direction

Table 2.4: Type of aggregation performed in every meteorological variable.

Finally, we also checked the percentage of informed entries for every variable. There were just a 0.24% of missing values, so we handled missing values by applying the Prophet algorithm separately for each one, like we did for the noise data.

2.2.3 Seasonal variables

The activity in a city like Barcelona can significantly vary depending on the moment of the week. Road traffic is usually higher in working days, while there's often much more leisure activities in weekends or

holidays. Thus, it makes sense to provide this information to the model, so it can learn from it.

First, we manually elaborated a list with all the Bank holidays in Barcelona from 2017 to 2021 in order to build a variable called *ind_holidays* to identify all the hours belonging to a Bank holiday. Similarly, we created a variable called *ind_weekend* to be able to distinguish Saturdays and Sundays from the rest of the days of the week. What is more, as we wanted this information not only to be useful to describe those days, but also the afternoons, evenings and nights before them, we altered both variables to also be active after 15h of the day before. The way to understand this is that, from the moment most people finish working, their behaviour is different since they do not need to work the following day.

Another important factor to consider is that urban activity is completely different based on the hour of the day. To discover the seasonal pattern within a day, we decided to analyse the sequence of data compared to the minimum noise level registered in that day. This is useful because we are avoiding that trend components affect our study of the seasonality. Then, we averaged this difference grouping by the hour of the day. In Figure 2.3 we can see this analysis applied just to the training data. We found out that there were three very different ranges in daily seasonality terms. Hence, we created a variable called *hour_range* that accounts for this factor: it is equal to 0 from 2h to 6h, it is equal to 1 at 0h, 1h, 7h or 23h and it is equal to 2 otherwise.

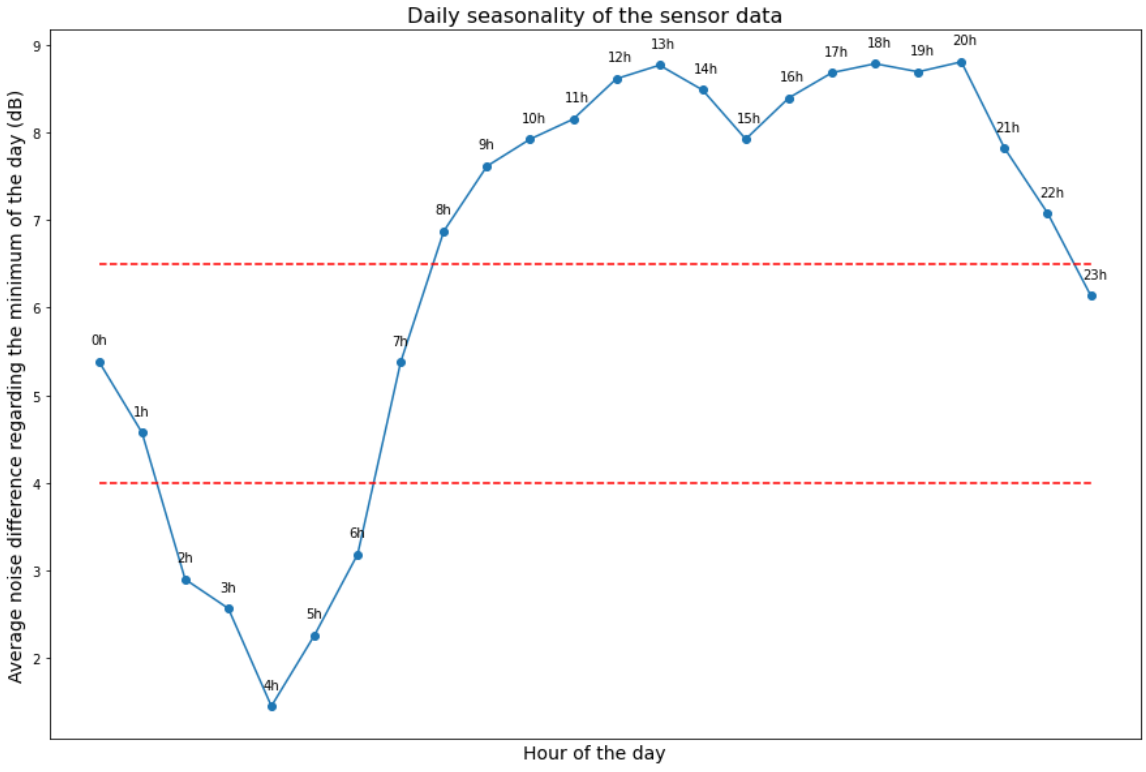

Figure 2.3: Daily seasonality of the training data. Namely, it is the average noise difference regarding the minimum of the day per each hour of the day.

2.2.4 Train, validation and test split

In every machine learning model, we need to split the dataset into a training part, where the model can learn from the data in order to choose the best algorithm and the best parameters to face the problem, and a testing part to evaluate the performance of the chosen model. In time series problems, the split of the data is normally performed sequentially, in the sense that the values of the past are the ones that should help to predict the future, not all the way round. It is usually a good practice to consider an additional dataset, the validation dataset. In the sages where we decide the best model and tune the hyperparameters, we will be using the train set to train and the validation set to evaluate the results. Once the final model is decided, we will train it in the whole train and validation datasets to finally evaluate it in the test set.

We decide to use a 70% of the dataset for the train set (hourly noise data from 01/01/2017 at 0h to 28/06/2020 at 18h), a 20% for the validation set (from 28/06/2020 at 19h to 27/06/2021 at 10h) and a 10% for the test set (from 27/06/2021 at 11h to 31/12/2021 at 23h).

2.2.5 Feature scaling

Since all the features that we use to feed the model can take very different scales of values, the algorithm could misunderstand that the variables with larger numbers are more important than the ones with smaller ones. In order to provide a fair framework among all variables, numerical features are usually scaled to the same range.

We scaled all features and the target to the range [-1, 1] by using the *MinMaxScaler* provided by the *scikit-learn* Python's library. This class transforms the values of every column proportionally within the desired range. It is the most common choice because it preserves the shape of the dataset without any distortion. Other options like *StandardScaler* or *RobustScaler* are preferred in case we know the data distribution is normal or there are lots of outliers, respectively. The range is chosen to be [-1, 1] because LSTM networks -the most common deep learning models for time series problems- expect data to be within the scale of its activation function, which is usually the hyperbolic tangent, whose output values range from -1 to 1.

It is important to fit the scaler class just in the train dataset and then using it to transform the validation or test dataset, because otherwise we could contaminate the experiment with knowledge from the test dataset, giving unfair advantage to the model and increasing the risk of overfitting. So, the scaling stage will be performed after splitting the dataset in train, validation and test sets. It is also recommended to use two different scaler objects, one for the input and another one for the output.

Finally, we will have to use an inverse method to transform the predictions back to their original scale. Error metrics, actually, are always calculated in the original scale to give a better idea of the usefulness of the model.

To sum up, we have built a final dataset to use it as input of our prediction models that contains 67 features: 48 lagged noise variables $(y(t-h), h = 1, \ldots, 48)$, 16 meteorological variables and 3 seasonal variables. On the other hand, there is a target that contains 10 columns to take into account the predictions in the following 10 hours: $y(t+h), h = 0, \ldots, 9$.

Chapter 3

Predictive models

In the chapter 2, we described the information sources we gathered and we detailed all the procedures applied to the available data to generate a suitable dataset to feed machine learning models. Now, in this chapter, we will introduce the methods we will use to predict the noise in the city of Barcelona.

3.1 Modelling theoretical context

In order to solve this time series problem, we will generate different models through the following approaches and compare their performance on the data:

- An optimal model generated by TPOT, an AutoML tool based on *scikit-learn*
- An optimal model generated by AutoKeras, an AutoML tool based on *Keras* and therefore specialized in neural networks
- Two manually tuned models based on Long Short-Term Memory networks (LSTMs), Recurrent Neural Networks (RNNs) and/or Gated Recurrent Units (GRUs), which are deep learning models that are expected to work well on long sequences of data

In this section, we will first describe from a theoretical point of view these AutoML tools and specialized models by giving the proper context. Later on, in section 3.2, we will explain how we have used them all to train and select models capable of predicting the noise pollution in Barcelona.

3.1.1 AutoML tools

The term AutoML refers to tools designed to automate the several stages of a machine learning problem, including data processing, feature engineering, model selection and hyperparameter tuning. There are some AutoML tools which focus on concrete parts of this pipeline and some others that cover the whole process end to end.

These tools are becoming more and more popular because they are supposed to allow to train models with good performance to users with limited coding knowledge or data science experience to satisfy their business or academic needs. Of course, it is true that it takes very long to manually search through all possible algorithms and hyperparameters, evaluate them on the training and validation sets and think about useful additional criteria to select the best model for an application. Nevertheless, it is worth to mention that these tools cannot solve business or academic problems on their own, because professional expertise is still needed to gather the appropriate data sources, design the experiment, process the data and take into account any insightful reason to select or discard a model suggested by the tool.

3.1.1.1 TPOT

Tree-based Pipeline Optimization Tool (TPOT) is an open-source Python library which performs AutoML processes optimising machine learning pipelines by means of genetic programming. TPOT is designed to explore thousands of possible pipelines smartly to find the one that fits best the data of the problem. In addition, it is capable to generate the Python code from the optimal choice so as the developer can use it later. The library is based on *scikit-learn* functionalities, both for data transformations and machine learning algorithms.

TPOT is based on a tree-shaped structure which represents most parts of the general pipeline for machine learning problems. As we can see in Figure 3.1, it includes feature selection, feature processing, feature construction, model selection and hyperparameter tuning. A stochastic global optimisation process based on a genetic programming algorithm finds the tree structure that performs best in the given dataset.

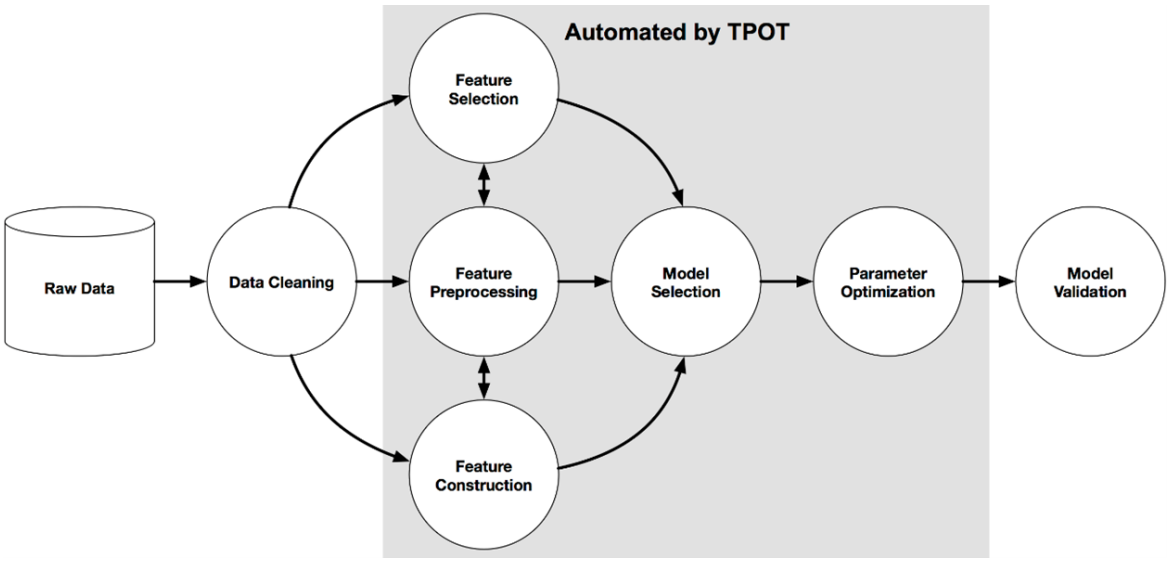

Figure 3.1: Representation of a typical machine learning pipeline and the stages of the pipeline which are automated by TPOT.

Genetic programming is a type of evolutionary algorithm (a subset of machine learning) that is used to

find solutions to problems that cannot be directly solved by humans. It is inspired by the *Natural Selection*, and it generates solutions to optimisation problems in computer science. In the case of Machine Learning, it selects the best algorithm with its optimal hyperparameters.

It is worth to mention that TPOT has several built-in configurations which are handful because they are supposed to perform well for certain types of machine learning problems. One of these configurations is *TPOT NN*, based on neural networks written in *PyTorch*. Unfortunately, *TPOT NN* currently only supports classification problems, so we will not be able to use it in our regression problem.

3.1.1.2 AutoKeras

AutoKeras is an open-source Python library used to perform AutoML processes based on neural network architectures. It is based on the framework Keras -specifically, the *tf.keras* API provided by *TensorFlow 2*- and automatically searches the optimal layers and hyperparameters of deep learning models.

In the case of neural networks, AutoML involves *Neural Architecture Search* (NAS) algorithms, which means that it looks for the optimal architecture and hyperparameters to train the model that achieves the best performance, under the necessary constraints, fitting the data of the problem.

The API is based on the *scikit-learn* design, so it is easy to use, since the user only needs to specify the location of the data and the number of models to try. The approach is simple and effective, and it can be used for a wide range of machine learning problems, including structured classification and regression datasets.

In our case, we will use the built-in configuration *StructuredDataRegressor*, which is flexible in terms of data format. This operator can be applied to our dataset without much effort. The only drawback is that AutoKeras does not include recurrent, LSTM or GRU cells, which are the structures expected to perform best when dealing with prediction of multivariate long time series, as we have in our problem. This is why we will also manually tune some models involving these types of layers.

3.1.2 Models for time series problems

In this section, the most suitable machine learning methods to predict long time series are described. In all cases, they are neural networks within the field of deep learning.

3.1.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a kind of neural network architecture especially useful to deal with sequential data such as texts or numerical time series. Whereas traditional neural networks struggle to

persist the information at long term, RNNs contain recurrent units, which are cycles or loops that allow to feed the network with both the output of the previous time step and the input of the current one. If we unroll the hidden layers of the network, as in Figure 3.2, we can easily see that the hidden state is recurrently passed to the next step together with the input of the current one.

RNNs have been successfully applied to problems related with speech recognition, language modelling, translation, image captioning and time series.

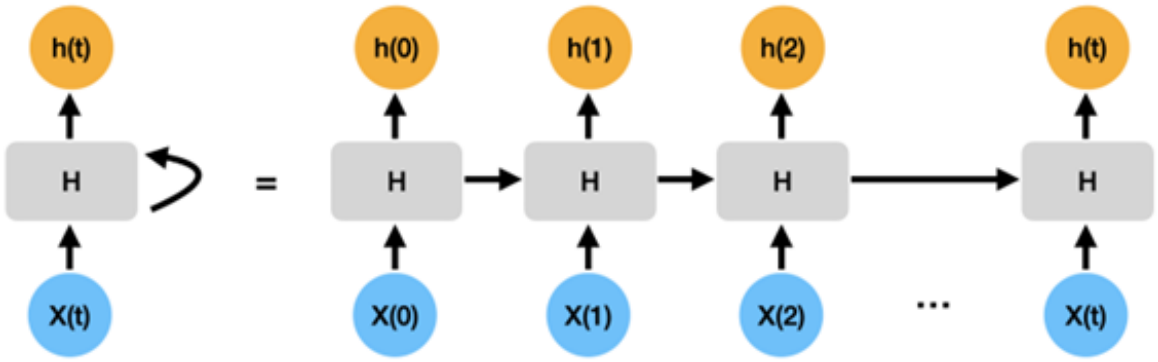

Figure 3.2: Unrolled RNN with $X(t)$ as input at time t, H as the hidden layer and $h(t)$ as its output at time t.

Nevertheless, for very long sequences, it is difficult that the hidden state can retain information from much prior stages. At every step, the *backpropagation through time* (BPTT[1]) process updates the gradient, which can potentially vanish. When this happens, the contributions of the far earliest states are lost in the process.

RNNs, though, are still very useful for tasks where only recent information is required to compute the present value. However, when the gap between the relevant values and the places where they are needed become large, RNNs cannot connect those elements. In order to solve the vanishing gradient problem, other networks such are LSTMs and GRUs have been created.

3.1.2.2 Long Short-Term Memory Networks

Long Short-Term Memory networks (LSTMs) are an adapted version of the RNNs which are designed to be able to deal with long range dependencies by retaining useful information about previous data to improve the generation of new points. Like RNNs, LSTMs are based on a chain-like structure, but every repeating block consists of three additional layers, called gates, which act as filters to decide the contributions to keep

[1] *Backpropagation Through Time* (BPTT) is a specific training algorithm that minimise the loss function of recurrent neural networks by iteratively unrolling the network, calculating and accumulating errors along all time steps and rolling-up the network and updating its weights.

and the ones to discard.

We will describe the architecture of these networks to understand why they can keep the information at long term. First, the horizontal line shown at the top of the diagram of Figure 3.3 is called cell state and it is thought to run through the entire chain with few modifications. In addition, the LSTM has three gates that can remove or add information to the cell state. These gates are neural networks including sigmoid activation and a pointwise multiplication operator. The three gates included in any LSTM represent three steps of the training process and are called forget gate, input gate and output gate.

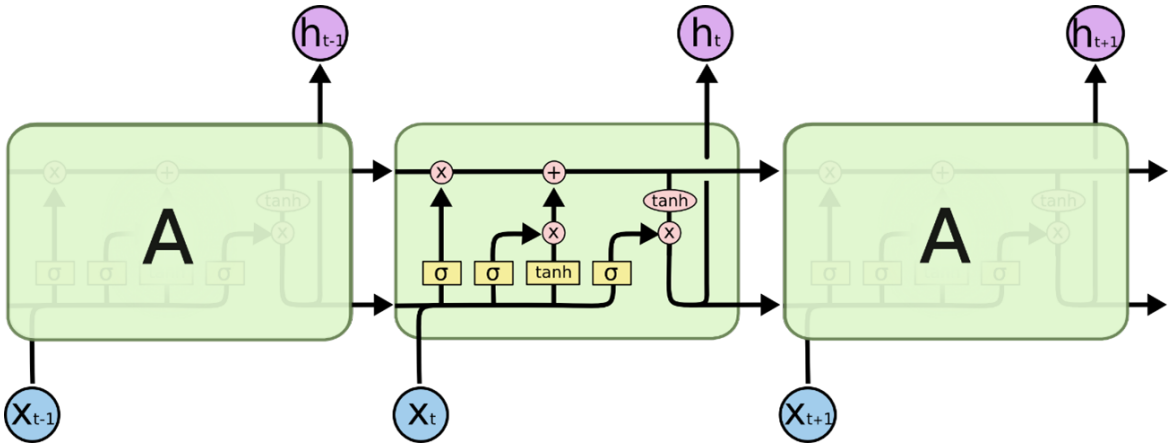

Figure 3.3: LSTM structure showing the four interacting layers in every repeating cell of the network.

In every cell, the first step involves the use of the forget gate, which determines -as its name states- the information that should be forgotten from the cell state. The sigmoid activation function gives values between 0 and 1. An output close to 0 means that some component is considered irrelevant, and one close to 1 is related to information important enough to keep. This behaviour is applied to the cell state by means of a pointwise multiplication.

Next, the network decides what information to include in the cell state. There is a double component to do so, the new memory network and the input gate. On the one hand, the new memory network establishes the update degree of every component of the long-term memory (cell state) given the new data. On the other hand, the input gate identifies the elements of the new memory vector which are important enough to be added, acting as a filter, like the forget gate, by means of the sigmoid function and the pointwise multiplication.

Finally, the output gate is in charge to be sure that only relevant information is added to the output, by determining the new hidden state based on the newly updated cell state, the previous hidden state and the new input data.

All the steps explained above are repeated many times, one per time step. Since the output of this

procedure is still a hidden state, at the end LSTM networks use a linear layer to transform it into the real output of the process.

In the case of our problem, we will be designing LSTMs with a *many-to-many* configuration, which means that they use a sequence of values (the lagged noise variables and the weather and seasonal variables) to predict another sequence of values (the next 10 noise values).

3.1.2.3 Gated Recurrent Units

Gated Recurrent Units (GRUs) are another adaptation of the RNN architecture in order to overcome the vanishing gradient problem. The design of their network is very similar to the LSTM, but since they are more recent, they can be considered as a simplified version of the LSTMs and thus are faster to train. GRUs also control the flow of data through gates, but instead of three gates, they only contain two: the reset gate and the update gate. These gates are two vectors that define what information to keep in the output and what information to exclude. Another remarkable difference is that it does not use any cell state, but it only presents the hidden state. Hence, like RNNs, at every timestamp, it takes an input and the previous hidden state and it generates a new hidden state for the current timestamp.

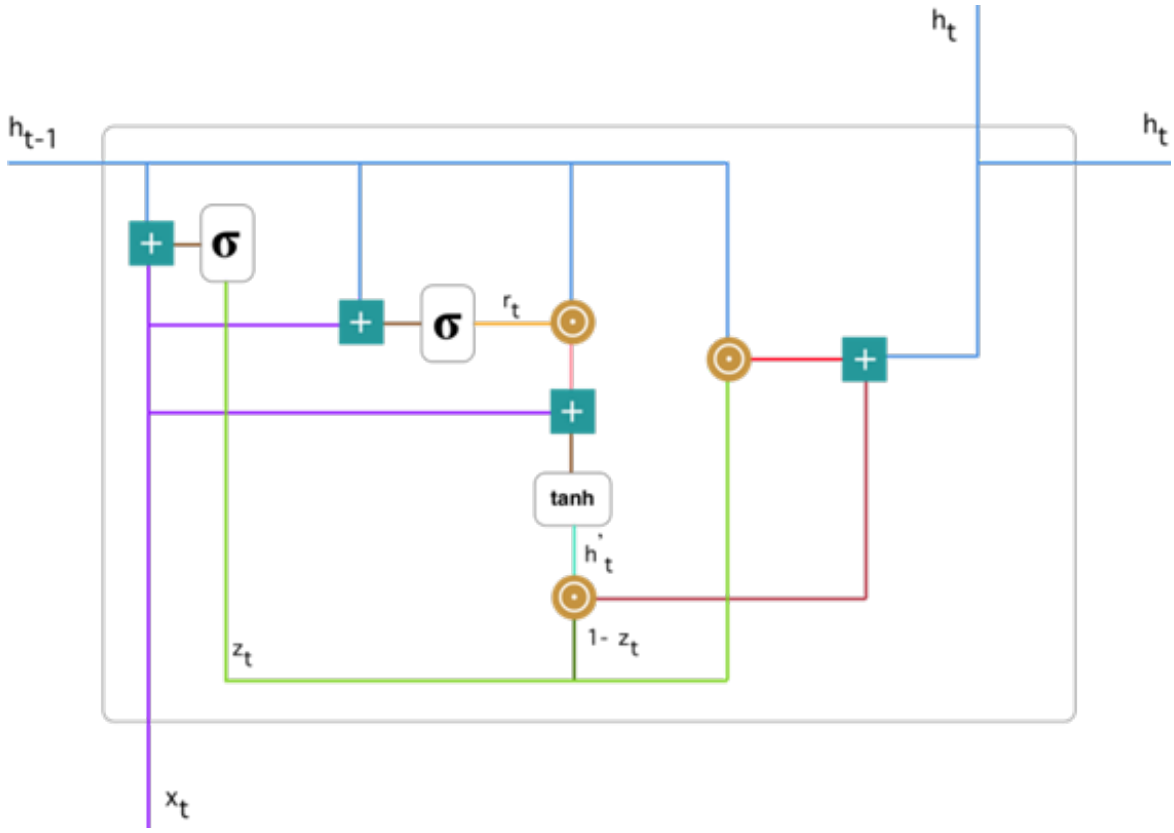

Figure 3.4: Detailed version of a Gated Recurrent Unit cell.

The mechanism behind the GRU is a two-step process. First, the update gate is responsible for long-

term memory, which means that it determines what are the contributions from previous steps that are worth to pass along to the future. It is equivalent to the output gate in LSTM units. Secondly, the reset gate accounts for short-term memory of the network (the hidden state) by deciding how much past information will be forgotten. Hence, it can mitigate the effect the previous data has on the current values of the sequence. It is analogous to the combination of the input gate and the forget gate in LSTM units.

With this mechanism, GRUs are capable to reduce the effect of the vanishing gradient problem, because they don't reset the information every time they get a new input, so they can keep the relevant information from the past and join it to the current data of the network. Like LSTMs, when properly trained, they perform excellently even in complicated problems.

3.2 Training and model selection

In this section, we will describe how we used the different approaches described in the section 3.1 to generate machine learning models capable to deal with the noise prediction problem in Barcelona.

The input of any possible model will be the dataset we have previously generated. As a brief reminder, this dataset contains 67 features: 48 lagged noise variables, 16 meteorological variables and 3 seasonal ones. The target consists of a sequence of 10 values containing the following 10 hours of noise to predict. Both the features and the target have been normalised to share the same scale, which is $[-1, 1]$. The 70% of the available data is used for training purposes (hourly noise data from 01/01/2017 at 0h to 28/06/2020 at 18h), the 20% is used to validate the training process (from 28/06/2020 at 19h to 27/06/2021 at 10h) and the remaining 10% is used to evaluate the models (from 27/06/2021 at 11h to 31/12/2021 at 23h).

We remind that when selecting the best models for every approach, we will be using the train set to train the models and the validation set to evaluate their performance. Once the optimal models are selected for every approach, we will train them again using both the train and validation sets altogether and we will evaluate them using the test set to get their final performance and to be able to compare their results. This comparison will be presented in section 3.2.7.

3.2.1 Baseline

In general, it is not trivial to know at first sight whether a performance obtained by one model is good or not. In machine learning projects, it is very common to use a baseline model, which is a very simple approximation that can give us a reference to the worst performance we should get. The baseline, actually, sets the threshold to be improved by any model we would like to use.

In time series problems, a very common baseline is to consider that our predictions will be equal to the

latest available training value of the series. In our case, mathematically, that will mean the following:

$$y(t-1) = y(t+h), \forall h = 0, \ldots, 9 \tag{3.1}$$

This rather simple approach, though, is sometimes surprisingly difficult to overcome, specially for the closest prediction horizons. We will check its performance in section 3.2.7.

3.2.2 TPOT

The AutoML tool TPOT has a very simple usage as long as you have generated a proper dataset to feed it. It is used by means of a Python library called *tpot*. In our case, we employed the *TPOTRegressor* class, which according to the official documentation, performs an intelligent search over machine learning pipelines that can contain supervised regression models, pre-processors, feature selection techniques and any other estimator or transformer within the *scikit-learn* API, and their hyperparameters. In addition, the user can even guide this search by tweaking some parameters. It is worth to mention that in our case we will carry out the optimisation process using the negative mean squared error as scoring function.

Although it is true that we could have trusted AutoML tools to perform the pre-processing steps of our problem, we decided to take care of these stages of the process manually because the data generally needs to be interpreted in the right way by the data scientists according to their business or academic needs. However, AutoML tools can be very useful to try the data in a wide range of models and hyperparameters on its own, a task that would be very time-consuming for a human.

Supervised machine learning methods like the ones used by TPOT needed a single target to work with, so we needed to adapt the methodology to be able to generate predictions for the next 10 values of the sequences. As we considered it would not make sense for TPOT to make 10 different searches to generate 10 different models, one per prediction horizon, we decided to use the closest target, $y(t)$, to choose the optimal model, and then train this same optimal model, separately, 10 different times iterating on the target column, to be able to generate proper predictions for every prediction horizon. Fortunately, one of the great advantages of TPOT is that, after having selected the optimal model, it provides the Python code so as the developer can run it at later stages. This was very handy in our case because we just needed to run the same code 10 different times.

After several tests and iterations, the final optimal pipeline chosen by TPOT was very simple, containing the following steps:

- Reading the input data
- Applying a *scikit-learn* class called *SimpleImputer*, which just imputes missing values with the median strategy

- Defining a *XGBoost* machine learning model, namely the *XGBRegressor*, with the parameters specified in table 3.1
- Training the model
- Generating the predictions, based on the validation or testing data

In section 3.2.7, we will compare the performance of such boosting model with other deep learning methods we will introduce next.

***XGBRegressor* parameter**	**Value**
learning_rate	0.1
max_depth	3
min_child_weight	7
n_estimators	100
n_jobs	1
objective	*reg:squarederror*
subsample	0.55
verbosity	0

Table 3.1: Optimal values for the *XGBRegressor* parameters, chosen by TPOT.

3.2.3 AutoKeras

As it happens with TPOT, AutoKeras is also an AutoML quite easy to use. Since AutoKeras is based on deep learning models that can be applied to completely different situations, it contains classes designed to specifically work on tasks such as image classification/regression, text classification/regression, structured data classification/regression or time series forecasting. Contrary to what one could think at first sight, the class *TimeSeriesForecaster* is designed to be used with a sequence of time series data, with no regressors and in the original array format. Since we transformed our time series problem into a supervised learning problem to be able to use the richer multivariate approach, the class that suits our needs is the *StructuredDataRegressor* instead.

The main difference compared to TPOT is that we will not obtain Python code lines now, but a model object that we will save and load if we want to use it again. One huge advantage of using neural networks in our problem is that they can directly manage the fact of having a sequence of values as a target, following the many-to-many approach. We only require that the last dense layer of the network uses 10 neurons instead of 1. This strategy will be used here as well as in the next two networks we will manually tune.

All in all, the optimal neural network selected by AutoKeras contains 9 layers, mostly alternating dense and *ReLU* layers. Besides, there is a multicategory encoding in the beginning, followed by a normalisation

layer, and at the end a dropout followed by a dense layer. The main aspects of this model are summarised in the Table 3.2. It contains 24337 parameters in total, 24202 of which trainable and 135 non-trainable. The metric used during the *backpropagation through time* (BPTT) optimisation process was the mean squared error.

Layer Type	Output Shape	Number of parameters
InputLayer	$[(None, 67)]$	0
MultiCategoryEncoding	$(None, 67)$	0
Normalization	$(None, 67)$	135
Dense	$(None, 32)$	2176
ReLU	$(None, 32)$	0
Dense	$(None, 512)$	16896
ReLU	$(None, 512)$	0
Dropout	$(None, 512)$	0
Dense	$(None, 10)$	5130

Table 3.2: Details of the optimal model selected by AutoKeras.

3.2.4 LSTM

It is important to remind that AutoKeras selects an optimal neural network based on different kinds of layers, but among those candidates there are not recurrent layers, LSTMs or GRUs, which are the ones expected to work best from a theoretical point of view. This is the reason we wanted to manually tune some models based on these structures to compare it to the algorithms supposed to be optimal among all the other options.

The next model we propose is a basic LSTM model, composed of two LSTM layers of 50 neurons, followed by a dropout and a final dense layer with 10 neurons to allow the model to forecast values for the 10 horizon targets. Details of the model are shared in Table 3.3. It contains 44310 parameters in total, all of them trainable. The loss function used in the optimisation process was the mean squared error.

Layer Type	Output Shape	Number of parameters
LSTM	$(None, 1, 50)$	23600
LSTM	$(None, 50)$	20200
Dropout	$(None, 50)$	0
Dense	$(None, 10)$	510

Table 3.3: Details of the LSTM model.

It is worth to mention that neither the increase of the number of LSTM layers nor the number of neurons could significantly improve the performance of this model. The use of the dropout layer indeed improved the obtained metrics, whereas more dense layers made no difference and *ReLU* layers made the training much more unstable. Using bigger batch sizes or number of epochs was not useful as well. It is important to mention that the parameter *return_sequences* must be set to True in all the LSTM layers of the model except the immediately previous to the final dense layer. Otherwise, dimensions do not properly fit, causing the model to fail.

3.2.5 RNN, LSTM and GRU combined

In the model selection stage, we also tried to use GRUs and RNNs instead of LSTMs, as well as combining all of them. Performance of the model rarely improved the LSTM model of the section 3.2.4, but we found a configuration that could compete quite well against it.

The last model we suggest is a combination of the three kinds of layers supposed to work well for long sequences of data. It contains 6 layers of 50 neurons each: two simple RNN layers, two GRUs and two LSTMs -in that order-, followed by a dropout and a 10-neuron dense layer. Details of the model can be found in Table 3.4. It contains 82460 parameters in total, all of them trainable. Again, the chosen loss function was the mean squared error.

3.2.6 Error metric

In any project involving predictive models, the choice of the metric error is not a minor issue at all. First, it is worth to mention that although we trained the models over scaled features and targets, it would not give us any valuable information to study the errors in the normalised scale, since they would be meaningless to understand how good or bad is our model in relation to the purpose of the problem. Instead, once the models are trained, we apply the inverse information with the scaler object we used to normalise the data. So, it is very important to store the *scikit-learn* scaler objects when they are created to be able to retrieve them afterwards.

Layer Type	Output Shape	Number of parameters
SimpleRNN	$(None, 1, 50)$	5900
SimpleRNN	$(None, 1, 50)$	5050
GRU	$(None, 1, 50)$	15300
GRU	$(None, 1, 50)$	15300
LSTM	$(None, 1, 50)$	20200
LSTM	$(None, 50)$	20200
Dropout	$(None, 50)$	0
Dense	$(None, 10)$	510

Table 3.4: Details of the RNN-GRU-LSTM model.

Once we have decided to represent the performance metric in the original scale, it is also good to point out that all our training processes have used the *mean squared error* in the optimisation stage. This gives a good clue that we should check this metric to evaluate the selected models. The *mean squared error* (MSE) averages the squares of the differences between the predictions and the actual values. The only drawback of the MSE is that it is expressed in square units, so it might be not very intuitive to interpret when related to our data. Hence, finally, we decided to use the *root mean squared error* (RMSE) to be able to express it in the same units as the data, which are dB(A). Mathematically:

$$RMSE = \sqrt{\frac{\sum_{t-1}^{T} \left(\hat{y(t)} - y(t)\right)^2}{T}}, \tag{3.2}$$

where t is the time step, T the total number of time steps, $y(t)$ the actual value of the target at time step t and $\hat{y(t)}$ is the prediction of the model at time step t.

3.2.7 Model comparison

We have selected four candidate optimal models to solve the noise prediction problem in Barcelona. As a brief recap:

- A *XGBRegressor*, chosen by TPOT
- A neural network based on dense and *ReLU* layers, chosen by AutoKeras
- A LSTM neural network, manually tuned

- A neural network concatenating RNNs, LSTMs and GRUs, manually tuned

As we said in section 3.2.6, the chosen error metric is the root mean squared error (RMSE). We expect the performance of the models to drop as the horizon prediction becomes large. We can first see the comparison between the chosen optimal models and the baseline model in Figure 3.5. It is clear that any of the candidate models is much better than the baseline, not only because the first prediction error is approximately a 26% lower than the one obtained by the baseline, but also because the baseline model drops its performance over the prediction horizons much more than any other candidate model. The baseline average RMSE over all horizons is 5.598 dB(A).

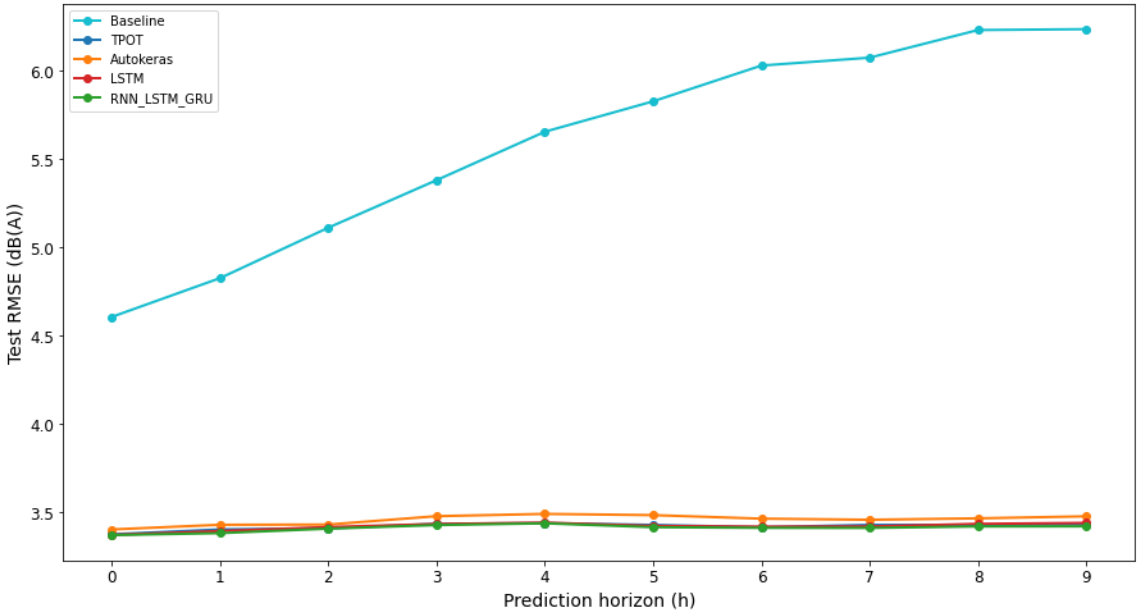

Figure 3.5: Performance of the candidate models over the prediction horizons, with the main goal to compare them to the baseline model. The error metric used is the *root mean squared error* (RMSE), evaluated in the test set. The unit of this metric is dB(A).

If we forget the baseline now and we zoom the plot a little bit, we can see the comparison between candidate models summarised in a very clear way in Figure 3.6. All models have an acceptable performance, but the best of them is the RNN-LSTM-GRU neural network, which has the minimal error in all the horizon windows. Its average RMSE over all horizons is 3.412 dB(A). Slightly above, we can find the LSTM neural network and the *XGBRegressor* chosen by TPOT, which have very similar performances, both with an average RMSE of 3.421 dB(A) over all horizons. The worst performance is offered by the neural network selected by AutoKeras, which has an average RMSE of 3.460 dB(A) over all horizons.

It is also noticeable in the pattern of the plotted lines, that there is an increasing error for prediction horizons from 0 to 4, then the error decreases a little bit until 7 and then it gets bigger again. This might be related to some effect of the seasonality of the data.

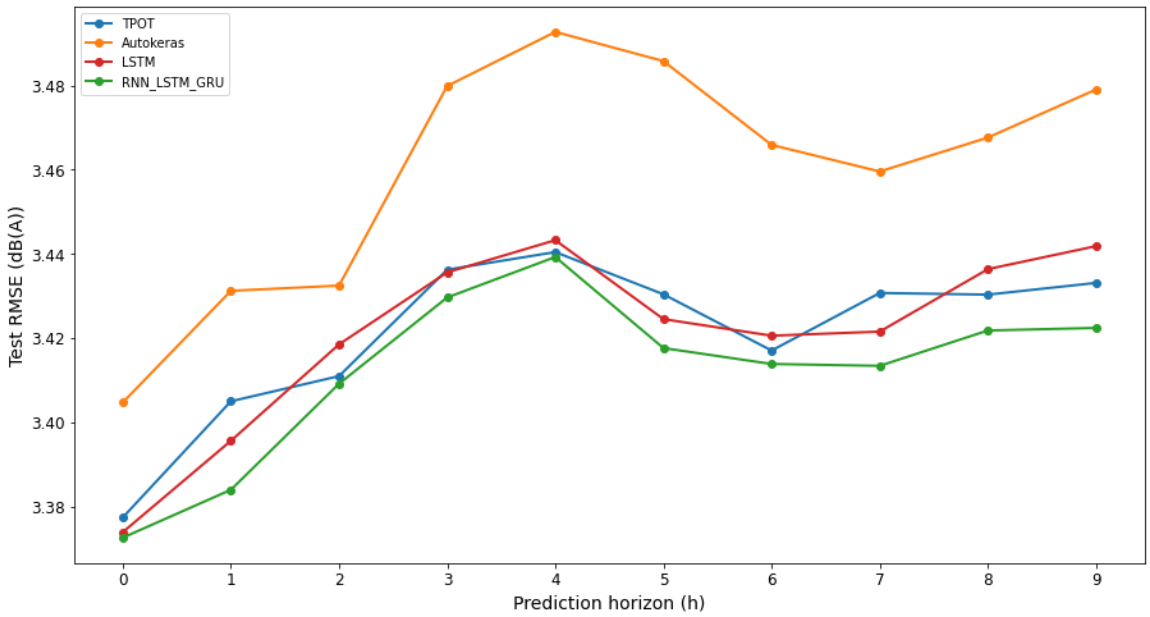

Figure 3.6: Performance of the candidate models over the prediction horizons. The error metric used is the *root mean squared error* (RMSE), evaluated in the test set. The unit of this metric is dB(A).

After having compared the error metrics, it seems that the best model to choose to deal with the noise prediction problem in Barcelona is our manually tuned RNN-GRU-LSTM model.

Model	**Average RMSE over all prediction horizons (dB(A))**
RNN-LSTM-GRU	3.412
LSTM	3.421
XGBRegressor (TPOT)	3.421
Neural network (AutoKeras)	3.460
Baseline	5.598

Table 3.5: Average RMSE of the candidate models over all prediction horizons, in ascending order.

Chapter 4

Model explainability

4.1 Theoretical context

Predictive models which solve real-world problems cannot be fully understood by just giving a performance metric on a test dataset. In many cases, the task to analyse is not well-known enough or there could be potential risks if significant mistakes are made by the model. What is more, there might be people affected by the decisions of a machine learning model, therefore having the right to ask why the output of the model is one or another. In fact, several regulations and laws have been recently approved to start to control these issues, like the EU *General Data Protection Regulation* (GDPR). Thus, data scientists don't only have to design models with high accuracy, but also have to be able to justify the decisions these models make.

Simpler models are much easier to explain and this is why they are sometimes chosen, even at the expense of losing accuracy. However, current applications require a lot of data and need to be dealt with more complex models, such as machine learning or deep learning methods. These models can achieve very good results, but they are normally considered black boxes, because the developers can rarely explain the reasons behind the predictions they make.

The term interpretability is usually defined as the degree to which a human can understand the cause of a decision that a model makes. Generally, a simple model can be explained by itself, because it follows clear rules, or it is based on clear relations between variables. However, frameworks beyond three dimensions are difficult to understand by humans. Therefore, complex models need help from other specific methods to be properly understood.

Global interpretation of a machine learning model gives insights about the more important features that affect on the prediction output and about the existing interactions among them. There are some methods that work by changing the input of any supervised learning model and measure the changes produced in the prediction output. Nevertheless, a global understanding is hard to achieve in practice, and we usually try to explain just certain parts of the model. Other approaches work by returning data instances as explanations, for example trying to justify what the model predicts for a particular input. At a local level, predictions may

only depend linearly or monotonously on some features. Therefore, local approaches are often considered to be more interpretable than the global ones.

4.1.1 Additive feature attribution method

Nowadays, many of the tools used for model explainability are based on the additive feature attribution method. We will explain it mathematically. Let f be a prediction model and let x be a single input, so $f(x)$ is a prediction we want to locally explain with another model g. Local explanation methods assign simplified entries x' to the original input through a mapping function $x = h_x(x')$, trying to ensure that $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$. These models follow the additive attribution method, which is described as a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{4.1}$$

where M is the number of features, $z' \in \{0,1\}^M$ and $\phi_i \in R$. The explanation model assigns an effect ϕ_i to every feature and it sums all these contributions to approximate the output $f(x)$ of the original model.

4.1.2 SHAP values

SHAP (Shapley Additive explanation) is likely the most common approach to deal with model explainability. It is a unified approach to explain the output of any machine learning or deep learning model by connecting game theory with local explanations based on the additive feature attribution method. SHAP describes the influence of every input variable in the final predictions by comparing what the model returns with or without that feature. Besides, this method is applied in every possible order, since the order in which we take the features could affect the outcome of the model.

SHAP values can be considered a unified measure of feature importance. Following the mathematical notation we introduced before, they are calculated by defining $f_x(S) = f(h_x(z')) = E\left[f(x)|x_s\right]$, where S is the set of non-zero indexes in z' and $E[f(x)|x_s]$ is the expected value of the function conditioned on a subset S of the input features. SHAP are a combination of game theory, classic Shapley values and these conditional expectations to assign weights ϕ_i to every feature with the following combinatorial equation:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)], \tag{4.2}$$

where N is the set of all input features.

4.1.3 Deep SHAP

One of the approximation methods inside SHAP is Deep SHAP, which is specifically designed to perform fast for deep learning algorithms, supporting TensorFlow and Keras models. Thus, it is the one we will use in our problem.

Deep SHAP is based on Deep Learning Important FeaTures (DeepLIFT), which is an explainability method used in Deep Learning that assigns an importance score to the feature inputs for a given output. It decomposes the predictions into specific inputs by propagating backwards the contribution of all the neurons in the network and comparing the activation of each neuron with its activation in a reference state. It approximates Shapley values assuming that input features are independent and that the predictive model is linear.

Deep SHAP uses a distribution of background samples instead of a single reference value. It uses Shapley equations to linearize separate components of the network and it combines them altogether to work out SHAP values for the whole network. These SHAP values sum up to the difference between the expected model output on the background samples and the current model output ($f(x) - E[f(x)]$).

4.2 SHAP values to explain RNN-GRU-LSTM predictions

In this section, we will use SHAP values to provide a better understanding of the model we selected as optimal for this problem, which is the neural network combining recurrent layers, GRUs and LSTMs we presented in the previous chapter. Namely, we will follow the approach Deep SHAP, appropriate for deep learning models. The main goal of this study is to understand the most relevant features for the model and under what conditions they make the prediction to increase or decrease. A better understanding of the model can help us to detect any inconsistency and to prevent it from failing in the future. In order to simplify the analysis, it has been decided to just study the explainability of the model for the target in the first prediction horizon of the target, $y(t)$.

SHAP can be applied to our problem using a *Python* library called *shap*. The class we use for this task, the *DeepExplainer*, is fed with the already trained model and the train and validation sets altogether, so it learns from a dataset of 39267 observations and 67 features. Next, it calculates the SHAP values for all the instances of the test set, which consists of 4363 observations and 67 features.

It is important to remind that our model was trained over scaled features and targets in the interval $[-1, 1]$, so resulting SHAP values are also scaled. Since SHAP values are the contributions that we assign to the features to modify the target, they will normally have values close to 0, either positive or negative. We could not simply use the inverse process of the scaler object we used to normalise the data, because it would just detect values within the interval $[-1, 1]$ and would transform them into values in the scale of

target, that is, the noise pollution. Therefore, we use the following method to solve the scaling issues:

- The scaled expected value[1] is added to the also scaled SHAP values. Thus, the SHAP values have now an appropriate magnitude to be used for the scaler.
- The scaler applies the inverse transformation to both the expected value and the modified SHAP values.
- The expected value in its natural scale is subtracted from the SHAP values which were also in the target scale, finally obtaining the contribution of every feature to the target in every observation.

One of the best advantages of the *shap* library is that it provides a wide variety of graphs to explain the model, both globally and locally. This allows us to get a complete view of how our model works.

4.2.1 Local insights

We will start by looking at some individual predictions, arbitrarily chosen, to check how the RNN-GRU-LSTM model generates them. For example, we start by looking at the Figure 4.1. This kind of graph is called *force plot* and it shows how the different features push the output of the model from the base value of the model, which is 69.62 dB(A), to the actual prediction for this observation. Features in red contribute increasing the target value and the ones in blue contribute decreasing it.

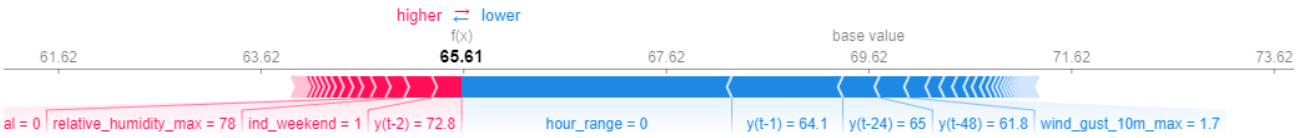

Figure 4.1: Individual prediction (27th of June of 2021 at 3h) generated by the RNN-GRU-LSTM model explained using Deep SHAP. Red features contribute the target to increase and blue ones make it decrease.

The first observation (Figure 4.1) is the first instance of the test set, and it corresponds to the 27th of June of 2021 at 3h. We can see that since it is a nightly value, the feature *hour_range*, which takes the value 0, has a big negative contribution to the outcome, making it decrease almost three points, with an associated SHAP value of -2.67 dB(A). We can see that other relevant contributions are the immediate previous noise value ($y(t-1)$), and also the noise values at the same time the day before ($y(t-24)$) and the day before that ($y(t-48)$), which were all smaller than the base value and caused the target to decrease. On the other hand, features like the noise two hours before ($y(t-2)$), which was higher than the base value, and the flag that informs the timestamp to be part of the weekend (*ind_weekend* $= 1$) pushed the outcome to increase.

[1]The expected value or base value is provided by the *DeepExplainer* and represents the average prediction of our model on train data. As all the other information provided by our model, it is originally normalised in the interval $[-1, 1]$.

The second observation (Figure 4.2) corresponds to the 16th of November of 2021 at 18h. In this case, *hour_range* takes the value 2 since it is a daily value and it is again the most important feature, making the output to increase by 1.58 dB(A). Other positive contributions are the two immediate prior noise values ($y(t-1)$ and $y(t-2)$). On the other side, variables referring to the wind are the ones which try to make the prediction lower.

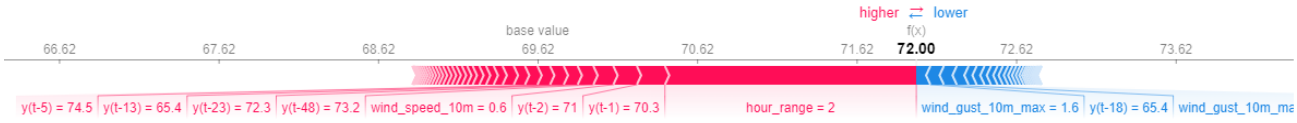

Figure 4.2: Individual prediction (16th of November of 2021 at 18h) generated by the RNN-GRU-LSTM model explained using Deep SHAP. Red features contribute the target to increase and blue ones make it decrease.

4.2.2 Global insights

However, it is not enough to check random individual predictions which, despite having reasonable behaviours, cannot be directly extrapolated to the whole test set. Fortunately, SHAP provides some graphics to easily explain the impact of the features over the target at a global level.

Figure 4.3 is a summary plot in its bar version. There, we can easily see the features with higher contribution to the predictions, but with SHAP coefficients expressed with absolute values. By far, the most relevant feature is *hour_range*, which accounts for the daily seasonality of the data. After that, it makes sense to find the three immediate noise values before the target ($y(t-1)$, $y(t-2)$ and $y(t-3)$). It is worth to mention that very high on the list we can find $y(t-24)$ and $y(t-48)$, which are the noise values at the same time of the target but the day before and the day before that, respectively. It is a very good signal that these two features are quite relevant to the model, among all other lagged noise variables. On the other hand, it is noticeable that, among all weather variables, the most relevant one is *relative_humidity_max*. Finally, the feature related to the moment of the week of the measure, *ind_weekend*, is also important. Since there are just few holidays on the calendar every year, it makes sense we don't find the seasonal variable *ind_holidays* on the top of the list.

Figure 4.4 is analogous to the previous one, but now the summary plot is found in its dots version. It also shows the most important features of the model, ordered by their impact on the output. However, this representation gives some additional clues. Since, for every feature, the SHAP value of every observation is plotted, we can check the proportion of instances that make every feature to increase or decrease the output. Besides, there is an extra dimension to analyse, because the colour red indicates a high value of the feature for the observation represented by that point, whereas the blue represents a low value instead. Based on that, we can clearly see that when *hour_range* equals 2 (from 8h to 22h), its contribution makes the output to increase. However, when it equals 1 (0h, 1h, 7h or 23h), it makes the output lower and, when

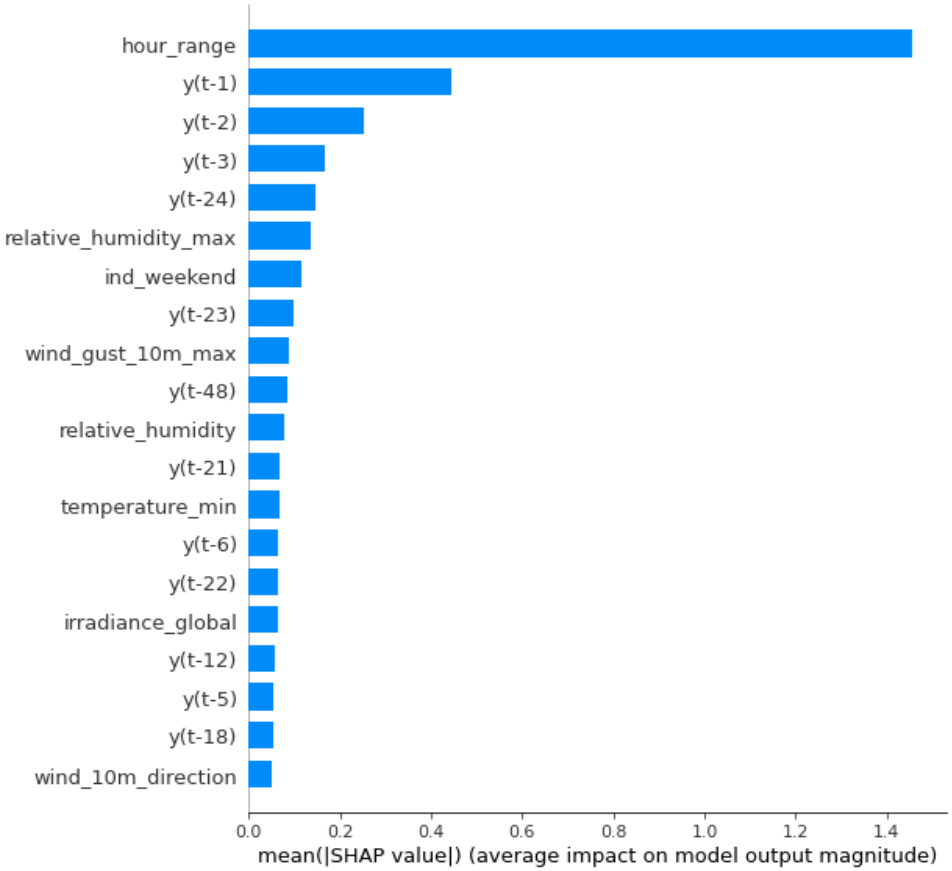

Figure 4.3: Bar graph showing the features of the model RNN-GRU-LSTM ordered by the absolute SHAP values averaged over all the instances of the test set.

it equals 0 (from 2h to 6h), even lower. On the other hand, all the lagged noise variables work as expected, making the target increase when they are high and making the opposite when they are low. It is curious that when the feature *ind_weekend* equals 0 (Monday to Friday), it barely affects the prediction, but when equals 1 (Saturday and Sunday) it can either increase or decrease the target. Finally, it is also noticeable that *temperature_min* works differently from the rest of the features: when the minimal temperature is higher, the target noise is lower, and vice versa.

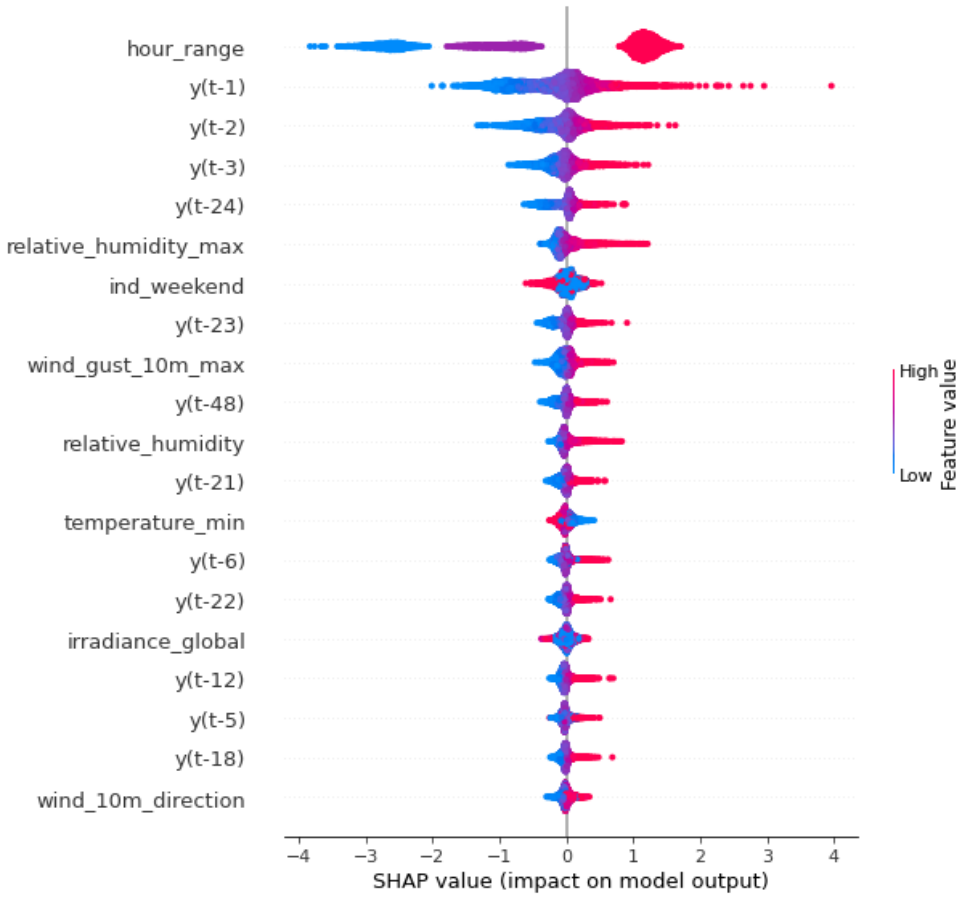

Figure 4.4: Summary plot of the most important features of the RNN-GRU-LSTM model, showing them ordered by their absolute impact in the predictions. For every feature, its associated SHAP value of every instance is plotted as a dot in the graph, sometimes accumulated vertically to show a higher density of points. The observations on the right of the vertical axis have contributions for the corresponding feature that make the output increase. The ones on the left behave in the opposite way. Red colours imply high values of the features, and blue colours imply low values for them.

Chapter 5

Conclusions and future work

5.1 Conclusions

Although noise pollution is the second most important risk factor for health in Western Europe, it has not been generally given the importance it deserves. Unlike air pollution, whose risks for the human health and environment are generally well-known, general public have normally been unaware of the dangers to be exposed at long-term to high noise levels.

After some recent studies, noise pollution has shown to be a bigger problem than one might expect due to the large amount of people affected, the wide range of illnesses it can cause and the gravity of them, being even responsible for premature deaths. In Europe, long-term exposure to environmental noise causes a chronic high annoyance to 22 million people and a chronic high sleep disturbance to 6.5 million people. Every year, it is estimated to cause 48000 new cases of ischaemic heart disease and to be the main reason of 12000 premature deaths. In addition, aircraft noise impacts on school learning impairment to 12500 children. Therefore, noise pollution is starting to become a serious public concern for both citizens and policymakers.

Most of the work available in the literature studies the environmental noise with classical time series descriptive analysis, calculating the affectation it can have to develop certain illnesses in specific places. Some predictive models have been developed as well, mainly in industrial environments.

Barcelona, one of the European cities most affected by road traffic and nightly leisure noise, provides an ideal framework for predicting the noise pollution. First, it owns a network of sensors spread all over the city called *Sentilo*, which allows to exploit this very valuable information with a lot of applications to be used for. What is more, due to recent policies in the European Union, a new action plan to decrease noise levels is being applied in 2022 in Barcelona and noise pollution is a trending topic at a local level.

With the help of the Barcelona City Council, we were able to gather noise measures every minute during 5 years in a significant point of the city. However, it might not make sense to build a model to determine

the environmental noise of the next minute in a certain urban area. Therefore, we opted to aggregate the data at hourly level. It is also worth to mention that Prophet algorithm turned out to be very useful to handle missing values for seasonal data.

Besides, we also gathered meteorological information and we considered variables related to the weather, as well as some extra variables with seasonal information, such as the time of the day, the fact of being at the weekend or the bank holidays. The inclusion of these features transformed our univariate problem into a multivariate problem. We performed several tests on neural networks and the training errors were lower when considering the additional features.

One of the trickiest parts when dealing with time series data is that if you want to use machine learning or deep learning models, you need to reshape the data into a supervised learning paradigm. This is usually done by creating lagged variables based on the target, but you could use lagged features as well. This means that, unlike ARIMA or Prophet which are trained based on a long sequence of data, machine learning models are trained with thousands of instances of shorter sequences. Besides, you can also create horizon variables to predict more than just the next value at every time. Namely, in our problem we were predicting the hourly maximum noise within the next 10 hours, based on the hourly noise values from the last 48 hours and the meteorological and seasonal features from the last hour.

In terms of modelling, it has been very interesting to compare different approaches and check the best one. On the one hand, we used neural networks which are specifically designed to perform well on long sequences of data: Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). On the other hand, we used two AutoML tools that were able to choose optimal models, but they were not considering the three types of networks we just mentioned: TPOT included a wide range of machine learning models and Autokeras chose between certain kinds of neural networks. Beforehand, it should not be that clear whether a specific network manually tuned would outperform other theoretically worse options but optimally selected by automatic tools.

All the options had a very good performance when compared to the baseline model, which had an average RMSE -the error metric we decided to use in this problem- of 5.598 dB(A) over all prediction horizons. The optimal choice by Autokeras was a neural network based on dense and ReLU layers, with an average RMSE of 3.460 dB(A) over all prediction horizons. From TPOT side, a boosting model was selected, specifically an XGBRegressor, with an average RMSE of 3.421 dB(A) over all prediction horizons. This same performance was shown by a manually tuned LSTM, which was the third model we took under consideration. Finally, a manually tuned neural network that combined RNN, LSTM and GRU layers was able to outperform all the other options with an average RMSE of 3.412 dB(A) over all prediction horizons. The RNN-LSTM-GRU network was chosen as our optimal model.

These error metrics have a magnitude which is acceptable when compared to other works in the literature and it is compatible with the purpose of the model, which intends to predict if certain noise thresholds

are overcome or potentially compare several points of the city.

However, a single test metric is rarely enough to explain the good or bad behaviour of a model. One of the main parts of this work explained the importance of providing additional tools to better understand the predictions of machine learning and deep learning models. Users will only use applications if they understand and trust them. People has the right to know, as well, the reason behind some automated decisions that can affect their lives. We have focused on SHAP (Shapley Additive explanation) and its Deep Shap method.

Thus, SHAP values have been calculated to explain the feature importance for the predictions made by our RNN-LSTM-GRU model. Some additional steps were needed in order to revert the fact that the model worked with scaled features and target. We have provided some meaningful insights that show that the feature with most influence in the model is, by far, a seasonal variable informing the hour range of the day, which means that it was a good decision to include it[1]. Other features with high contributions to the model output were the three latest hours of data, which makes total sense, but it is worth to mention that the noise measure from 24 and 48 hours before were also very important, which can be understood as the model reflecting well the seasonality of the data. The variable informing the presence of the weekend was also considerable. Among all the meteorological variables, the maximum relative humidity had the greatest contribution.

5.2 Future work

Noise pollution is starting to be considered as a public concern and European cities have improved their data management about it. However, there is a clear delay on the implementation of laws to accomplish the goals of the Environmental Noise Directive (END) of the European Union. Unlike other stressors such as air pollutants, which are declining, the number of people exposed to high levels of noise has remained stable since 2012 and is expected to increase in the future. Thus, one of the goals of this research is raising awareness about the importance of mitigating the noise exposure to the population by explaining the health risks related to it.

At a technical level, it is remarkable that all the optimal models had very similar error metrics. Therefore, under the actual conditions, we might have reached a milestone very difficult to improve just by selecting another model or further retuning its hyperparameters. Instead, it is more likely to upgrade performance by adding other significant features. For instance, we could add data related to the COVID pandemic and the several lockdowns which took place since March 2020, so the model could better understand some significant changes that happened in several short periods of time. It could be also very beneficial to include data from road traffic, which is the most important noise factor in Barcelona, but we would need further steps to properly gather, manage and process it. It could also be interesting to add information about tourism,

[1]There might be a feature with a very high importance causing the model to make something not intended, but in this case we consider it as a good decision accounting that we stated before that the multivariate approach outperformed the univariate one

which is essential to reflect the activity in a city like Barcelona. There could be, as well, other encoded seasonal variables, such as the month of the year, the day of the week or the hour of the day.

Either by adding more variables or not, the model should be more sensible and easier to explain by applying, before training, some prior process related to the selection of variables, such as *Principal Component Analysis* (PCA). If there are features that are too correlated, it is more difficult for the model to find out the most meaningful relations to generate the target.

Another interesting point to test could be increasing the number of lagged noise variables and the number of the horizon to predict and check how it affects to the performance of our model. For example, if we were using a whole week of lagged hourly noise variables and a whole day as the hourly horizon to predict, could we improve our model in the closest horizons? And what about the furthest ones?

The last technical point in which this research could be widened is the consideration of other explainability techniques beyond SHAP values to understand how our model makes the predictions. Getting more insights on this matter could be very valuable for public institutions and the research field.

At a business level, there exist multiple advantages of knowing in advance which will be the noisiest areas in a city. As we explain in this work, the *Sentilo* framework provides a huge amount of valuable noise data all over the city. With proper access to all of it, we could extend this study to other points of the city, and that would be even more insightful. The predictive model could generate alerts when certain noise thresholds were expected to be overcome. Then, the public institutions could apply short-term actions to mitigate these effects, like decreasing the traffic speed limits in certain streets, redirecting the traffic to other areas or sending police officers to control specific activities.

Actually, with all the *Sentilo* data available, there exists a wide range of applications both in descriptive and predictive analysis, mainly to detect the most endangered areas to take further action on them.

Bibliography

(2011). Burden of disease from environmental noise. Technical report, World Health Organization, Regional Office for Europe; JRC European Comission.

(2020). Environmental noise in europe. Technical report, European Environment Agency.

(2022). Soroll ambiental i salut a la ciutat de barcelona. Technical report, Agència de Salut Pública de Barcelona.

Ajuntament de Barcelona (2013). Sentilo BCN. http://connecta.bcn.cat/connecta-catalog-web/component/map.

Ajuntament de Barcelona (2017). Mapes de dades ambientals. https://ajuntament.barcelona.cat/mapes-dades-ambientals/soroll/ca/.

Ajuntament de Barcelona (2022). Noves mesures per reduir la contaminació acústica. https://www.energia.barcelona/ca/noticia/noves-mesures-per-reduir-la-contaminacio-acustica_1171015.

Awan, F. M., Minerva, R., and Crespi, N. (2021). Using noise pollution data for traffic prediction in smart cities: Experiments based on lstm recurrent neural networks. *IEEE Sensors Council*, 21.

Bain, M. (2011). *Sentilo Case Study*.

Brownlee, J. (2017a). How to convert a time series to a supervised learning problem in python. https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/.

Brownlee, J. (2017b). Time series forecasting with the long short-term memory network in python. https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/.

Brownlee, J. (2020). Multivariate time series forecasting with lstms in keras. https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/.

Chandra, B., Middya, A. I., and Roy, S. (2020). Spatio-temporal prediction of noise pollution using participatory sensing. *Advances in Intelligent Systems and Computing*, 1286:597.

Doe, R. (2009). This is a test entry of type @ONLINE. http://www.test.org/doe/.

Díaz, J. and Linares, C. (2015). Health effects of noise traffic: Beyond 'discomfort'. *Rev. salud ambient*, 15(2):121–131.

Facebook Open Source (2017). Prophet - quick start. https://facebook.github.io/prophet/docs/quick_start.html.

García, M. V. (2019). Interpretable forecasts of NO_2 concentrations through deep shap. Master's thesis, UNED, Madrid, Spain.

García, M. V. and Aznarte, J. L. (2020). Shapley additive explanations for no2 forecasting. *Ecological Informatics*, 56.

García, N. G. (2014). *Sistema de predicción de ruido urbano mediante redes neuronales*. PhD thesis, Universidad de Granada, Granada, Spain.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*. O'Reilly.

Herkert, D. (2022). Multivariate time series forecasting with deep learning. https://towardsdatascience.com/multivariate-time-series-forecasting-with-deep-learning-3e7b3e2d2bcf.

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM.

Kostadinov, S. (2017). Understanding gru networks. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

Kumar, E., Venkatasubramanian, S., Scheidegger, C., and Friedler, S. A. (2021). Problems with shapley-value-based explanations as feature importance measures. *cs.AI*.

Le, T. T., Fu, W., and Moore, J. H. (2020). Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, 36(1):250–256.

Lendawe, V. (2021). How to do multivariate time series forecasting using lstm. https://analyticsindiamag.com/how-to-do-multivariate-time-series-forecasting-using-lstm/.

Lundberg, S. (2018). Welcome to the shap documentation. https://shap.readthedocs.io/en/latest/index.html#.

Montantes, J. (2021). Introduction to automl. https://becominghuman.ai/introduction-to-automl-d498bff8ad65.

Navarro, J. M., Martínez-España, R., Bueno-Crespo, A., Martínez, R., and Cecilia, J. M. (2020). Sound levels forecasting in an acoustic sensor network using a deep neural network. *Sensors*, 20(3):903.

Nielsen, A. (2019). *Practical Time Series Analysis*. O'Reilly.

Olah, C. (2015). Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Portal de dades obertes de la Generalitat de Catalunya (2019). Dades meteorològiques de la xema. https://analisi.transparenciacatalunya.cat/Medi-Ambient/Dades-meteorol-giques-de-la-XEMA/nzvn-apee.

Ramírez, J. A. M. (2017). *Integración de indicadores borrosos en redes de sensores inalámbricos. Aplicación en la monitorización de contaminación acústica*. PhD thesis, Universidad de Jaén, Jaén, Spain.

Saxena, S. (2021). Introduction to long short term memory (lstm)s. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

Tiwari, S., Kumaraswamidhas, L., Gautam, C., and Garg, N. (2022). An auto-encoder based lstm model for prediction of ambient noise levels. *Applied Acoustics*, 195.

Vishwas, B. V. and Patel, A. (2020). *Hands-On Time Series Analysis with Python*. Apress.

Wen, P.-J. and Huang, C. (2020). Noise prediction using machine learning with measurements analysis. *Applied Sciences*, 10.

Yi, V. M., Caballero, J., and Cavas, L. (2006). Análisis predictivo de contaminación acústica aplicados al tráfico vehicular, relación entre un modelo teórico y uno computacional. TecniAcústica.

Zeng1, Q., Liang, Y., Chen, G., Duan, H., and Li, C. (2021). Noise prediction of chemical industry park based on multi-station prophet and multivariate lstm fitting model. *EURASIP Journal on Advances in Signal Processing*, 106.