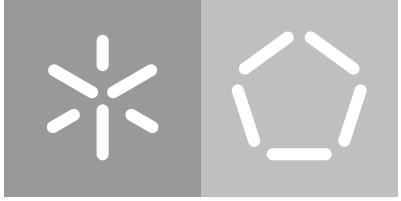
Universidade do Minho
Escola de Engenharia

Carolina Oliveira da Silva

A Machine Learning Approach to Environmental Sustainability

December, 2020



Universidade do Minho
Escola de Engenharia

Carolina Oliveira da Silva

A Machine Learning Approach to Environmental Sustainability

Master's Dissertation
Master's in Systems Engineering

Work supervised by
Professor Paulo Novais
Professor Bruno Fernandes

December, 2020

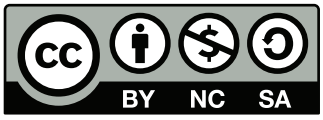
COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositoriUM of Universidade do Minho.

License granted to the users of this work



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
CC BY-NC-SA 4.0

https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Acknowledgements

Firstly, I would like to thanks my supervisor, professor Paulo Novais, for having given me the opportunity to do this dissertation.

To professor Bruno Fernandes a special thanks for all the commitment, patience, and availability to always help me and provide the main guidance to conclude this dissertation.

Third, I would like to thanks Pedro Oliveira for all the availability to help me when I needed, during the dissertation process.

Then, I would like to thanks my friends for all the support.

For last, but not least, a special thanks to my family, special to my parents and sister, who always supported me on this journey.

This work was partially supported by National Funds through the Portuguese funding agency, FCT - Foundation for Science and Technology, within the project DSAIPA/AI/0099/2019.

Abstract

Environmental sustainability is one of the biggest concerns nowadays. With environmental increasingly latent negative impacts, it is substantiated that future generations may be compromised. Thus, this research addresses this topic, in particular, the air quality and atmospheric pollution, as well as water issues regarding a wastewater treatment plant.

This study comes from a combination of Machine Learning supervised models to predict multiple parameters regarding environmental sustainability. Through the application of regression and classification models, the study target involves the air and the water quality in Guimarães city. Therefore, the key research goals are to predict attributes such as the Ultraviolet index, Carbon Monoxide air concentration, and water pH. Using Decision Trees, Random forest, Multilayer Perceptron, and Long Short-Term Memory, these parameters were forecasted. In this way, this study describes these models' implementation and optimization processes, as well as the results generated.

Predicting parameters of this nature will allow the anticipation of problematic situations, enabling preventive actions. Further, it grants the optimization and reallocation of resources, promoting the best for the population and the common good.

After the entire implementation process, several conclusions arose from this research. First, from the Ultraviolet index levels (defined by the World Health Organization) prediction, was achieved a maximum accuracy of approximately 93%. Moreover, regarding this parameter prediction using regression models, the best result showed a Mean Absolute Error of 0.36. Besides, this index was further predicted based on a time series, resulting in a Mean Absolute Error of about 0.15. Additionally, also using a time series approach, the Carbon Monoxide air concentration was forecasted, achieving a Mean Absolute Error of 1.345×10^{-7}. Finally, considering the water pH problem was reached, as the lowest Mean Absolute Error, a value equal to 0.11.

Keywords: Deep Learning , Environmental sustainability, Machine learning, Supervised Learning

Resumo

A sustentabilidade ambiental é uma das maiores preocupações da atualidade. Com impactos negativos ambientais cada vez mais latentes, está provado que as gerações futuras podem estar comprometidas. Assim, esta pesquisa vem abordar este tópico, em particular, a qualidade do ar e a poluição atmosférica, bem como as questões hídricas no contexto de uma estação de tratamento de águas residuais.

Este estudo advém de uma combinação de modelos de aprendizagem supervisionada com o objetivo de prever vários parâmetros referentes à sustentabilidade ambiental. Através da aplicação de modelos de regressão e classificação, o alvo da investigação envolve a qualidade do ar e da água na cidade de Guimarães. Por conseguinte, os principais objetivos da pesquisa passam por prever atributos como o índice de radiação ultravioleta, a concentração de monóxido de carbono no ar e o pH da água. Usando Árvores de Decisão, *Random Forest*, *Perceptron* Multicamadas e *Long Short-Term Memory*, esses parâmetros foram alvo de previsão. Deste modo, este estudo descreve os processos de implementação e otimização desses modelos, bem como os resultados gerados.

A previsão de parâmetros desta natureza permitirá a antecipação de situações problemáticas, possibilitando ações preventivas. Ademais, permite a otimização e realocação de recursos, promovendo o melhor para a população e o bem comum.

Após todo o processo de implementação, várias conclusões surgiram desta pesquisa. Em primeiro lugar, da previsão dos níveis do índice ultravioleta (definidos pela Organização Mundial da Saúde), foi alcançada uma precisão máxima de, aproximadamente, 93 %. Além disso, em relação à previsão deste parâmetro por meio de modelos de regressão, o melhor resultado apresentou um Erro Médio Absoluto de 0,36. Além do mais, esse índice foi alvo de previsão com base em uma série temporal, resultando em um Erro Médio Absoluto de cerca de 0,15. Ainda, também utilizando uma abordagem de série temporal, a concentração de monóxido de carbono no ar foi prevista, atingindo um Erro Médio Absoluto de 1,345 $\times 10^{-7}$. Por fim, considerando o problema do pH da água foi atingido, como o menor Erro Médio Absoluto, um valor igual a 0,11.

Palavras-chave: Aprendizagem Supervisionada, *Deep Learning*, *Machine learning*, Sustentabilidade ambiental

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

CART Classification And Regression Tree

CO Carbon Monoxide

CRISP-DM Cross Industry Standard Process for Data Mining

DO Dissolved Oxygen

H_2SO_4 Sulfuric acid

HNO_3 Nitric acid

ID3 Iterative Dichotomiser 3

IDE Integrated development environment

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MDL Minimal Description Length

ML Machine Learning

MLP Multilayer Preceptron

MSE Mean Absolute Error

N Nitrogen

N_2 Nitrogen gas

NO_x Nitrogen Oxides

O_3 Ozone

PAE Portuguese Agency for the Environment

PAN Peroxyacetyl nitrates

Pb Lead

PM Suspended particulate matter

ppb Parts per Billion

ppm Parts per Million

RELU Rectified Linear Unit

RMSE Root Mean Square Error

Rn Radon

RNN Recurrent Neural Networks

SO_2 Sulfur Dioxide

SO_3 Sulfur Trioxide

Tanh Hyperbolic Tangent

TDS Total Dissolved Solids

TSS Total Suspended Solids

UV Ultraviolet

VOC Volatile organic compounds

WHO World Health Organization

WWTP Wastewater treatment plant

1. Introduction

1.1 Contextualization

The world is living in a data age, illustrated by the fact that a vast amount of data are being produced continuously, like never before. Given this growing emergence of data, it is essential to note that this may be a valuable asset for many organizations, whether business or public entities. Thus, this large-scale data may be collected and study to extract meaningful information from it and helps entities to act in accordance (Sandryhaila and Moura, 2014; Lima, Novais, Costa, Bulas Cruz, and Neves, 2010).

Machine Learning (ML) is a field from Artificial Intelligence (AI) that has been growing in the last decades (Qiu, Wu, Ding, Xu, and Feng, 2016). This field concerns data collection, analysis and then predict several important parameters, or find important patterns from them. Further, ML is based on learning from data without being expressly programmed for that.

ML is used in many areas nowadays, once it is recognized as a field that can provide some solutions, allowing the extraction of information from the available data. The various entities that operate in the most diverse areas can, through ML, acquire the ability to act before something occur, avoiding negative impacts (Qiu, Wu, Ding, Xu, and Feng, 2016).

Since ML can be implemented in a vast range of areas, it is interesting to apply it in a way that can make a difference, aiming for the common good. From this emerges an area that requires so many concerns today: environmental sustainability.

Sustainable development, from an environmental point of view, has been a major problem for countries for decades since its negative impacts are increasingly perceived and verified (Isabel Molina-Gómez, Rodríguez-Rojas, Calderón-Rivera, Luis Díaz-Arévalo, and López-Jiménez, 2020). This topic is one of the main concerns nowadays. Due to the registered population growth, the demand for natural resources has been consequently increased. Further, the industrial activities are increasingly pronounced in developed countries, which is a key for its economic sustainability, arising the need to find a balance between environmental and economic sustainability. All of this generates anthropogenic emissions, which damage the environment, causing visible impacts such as public health problems, climatic changes, and so on. Therefore, all of this can compromise future generations. To avoid this risk, the responsible entities need to make decisions in order to reduce its negative impacts, and consequent risk for the population, now and in the future (Sarkodie, 2021).

In sum, ML techniques are highly useful for entities once they can look to data and to ML models to

guide the decision-making process (Hino, Benami, and Brooks, 2018). This research can help overcome some challenges once, by forecasting several future values of relevant parameters (in the environmental sustainability scope), makes it possible to anticipate problematic situations. Thus, it allows one to act in a preventive way to provide the best for the population and the future generations, allocating resources and consequently, reduce negative effects and maximize the potential benefits.

1.2 Motivation

The motivation for carrying out this dissertation comes from the high interest in the ML field, acquired in the course unit Similarity-Based Systems, inserted in the Master of Systems Engineering. There, I had a closer contact with AI, more specifically with ML. Realizing that it is an area with application in the most assorted areas, and after having contact with a practical approach, the idea of doing a dissertation in this field has grown.

After concluding my bachelor in business management, I felt the need to enroll for something different that, in the future, would work as a complement to my base area. Thus, the opportunity to enter in the Master of Systems Engineering arose. After experiencing many new and highly interesting areas, it was ML that stood out. I perceived it to be a very relevant field, with applications in several areas, and that would be, of course, a complement to my base area. Consequently came the opportunity to carry out this dissertation, with the possibility of learning more in this field.

Since it is a field that can be applied in a wide range of areas, environmental sustainability emerged as an interesting one. Combining the suggestion of the supervising professor, the availability of data, and, as previously mentioned, the constant concerns about this issue nowadays (once there is a continuing perception that this sustainability can be being compromised, causing a risk to life on Earth), this topic arose. So, in addition to a field of high interest for me, there was the possibility of adapting this to an area that deserves special attention nowadays, being able to contribute for the common good.

1.3 Research Methodology

The methodology used in this research is Cross Industry Standard Process for Data Mining (CRISP-DM). This methodology is the most used methodology for data mining projects (Mariscal, Marbán, and Fernández, 2010).

This consists of six distinct phases, as figure 1.1 shows.

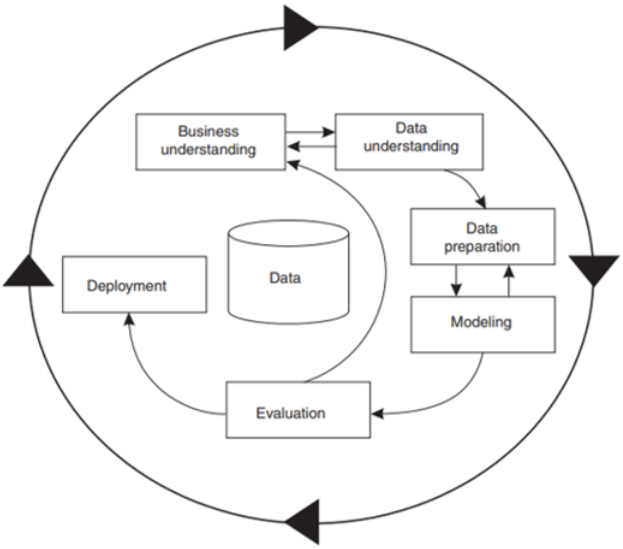


Figure 1.1: Cross Industry Standard Process for Data Mining methodology (Chapman et al., 2000).

These phases are:

- **Business understanding:** This initial step focuses on identifying the project goals and specifications (such as costs, resources available, among others) and, from this information, define a data mining problem, understand, and plan how to achieve these goals;
- **Data understanding:** This phase starts with an initial collection of data. After that, the data is explored, to be familiar with it, identifying some data issues, and understand the data behave, aiming to get some information from it;
- **Data preparation:** This phase regards the process that covers all activities required to transform the initial raw data into the final data set on which the project will be based. These tasks are about fix data issues, cleaning, construct, integrate, and reformat data;
- **Modeling:** Different modeling techniques are chosen and implemented in this step. Further, their parameters are tuned to find the optimal values. Some techniques have particular requirements for the data type. So, it is always appropriate to return to the data preparation stage;
- **Evaluation:** Was developed a model (or models) at this point of the project that appears to be of quality, from a data analysis perspective (a model that achieves good results). It is important to

analyze the model more closely and review the steps taken to create the model before moving to the final implementation to ensure that it achieves the business goals correctly;

- **Deployment:** Organize the knowledge extracted, as well as to present it in a way that makes sense to the project. Thus, this is the step in which is perceived what actions would need to be carried out to use the models generated and the knowledge and information extracted from it.

1.4 Objectives

This research will make use of supervised ML models in an environmental context, with a view to the common good: the best for current and future generations. Thus, this is a contribution to understand, forecast, and help the decision-making process concerning environmental sustainability.

The main goals of this project are:

- Understand and forecast air characteristics and quality. This goal is based on an exhaustive analysis of the available data set concerning air pollutants, weather, and Ultraviolet (UV) index, in Guimarães city. The forecast involves the UV index, using regression and classification models, as well as the Carbon Monoxide (CO);
- Understand and forecast water characteristics and quality. This goal is based on an exhaustive analysis of the available data set concerning a WWTP, also in Guimarães, to understand the wastewater quality in the several WWTP sections. The forecast concerns the water pH at the moment that it is discharged to the hydric sources;
- Understand what are the relevant attributes in the research, for each data set, concerning the forecasts quality:
- Understand how each model (Decision Trees, Random Forest, Multilayer Preceptron (MLP), and LSTM) behaves for each approach and dataset and tune it;

1.5 Document Structure

Besides this chapter (Introduction), this dissertation has more five chapters.

Chapter number two concerns the state of art. There it is explored the environmental sustainability issues relevant to the research. Accordingly, it is covered the atmospheric pollution, the water quality issues, and the treatment processes (in specific, the wastewater treatment), regarding its role and impact on environmental sustainability. Besides that, this chapter shows also the ML state of art, exploring the models used in this research, and the metrics to evaluate them.

Chapter number three is the materials and methods chapter. Here, both datasets used are explored, describing each attribute meticulously. Further, all the transformations made in the raw data are described, along with the description of the technologies used.

Chapter number four is the chapter that exposes the research experiments. Here are shown the built data scenarios and the models' conception, as well as its tuning process.

Chapter number five shows the results obtained in this research, for all the models and datasets, concerning classification and regression problems.

Finally, chapter number six shows the conclusions and future work of this study.

2. State of art

2.1 Environmental sustainability

Environmental problems have been increasing in recent years, mostly caused by the demographic expansion and the higher development of the industrial activity. Today's society produces a large amount of daily waste (from solid, liquid, and gaseous form), promoted by the growing consumption of natural resources, used to meet current needs. All of which leads to environment disturbances, fostered by anthropogenic action, causing changes in soil, water, and air quality (Wright, Richard T., 2019; Abdel-Shafy and Mansour, 2018).

Sustainable businesses are a keyword in the twenty-first century world. The long time survival of a company is crucial, but think about other kinds of sustainability is also important. Once industries are the largest responsible for several environmental negative impacts, there's a need for these to invest in actions that promote the environmental sustainability. Thus, they may change their processes to put into practice green management to create sustainable products and services (Schoenherr, 2012).

2.1.1 The atmospheric pollution and the environmental sustainability

Atmospheric pollution, more specifically, air pollution is an important environmental concern, primarily linked to urban conditions and industrial emissions. There are many challenges in trying to manage the air quality according to what is considered ideal for the population, allowing its well-being and health, bearing in mind current and future generations is the main .

Air pollution contributes to the emerging of several diseases. This issue has growth, resulting from the increase of polluting industries. In addition, there is an air pollution index rise on countries with low and medium/low incomes (Cohen et al., 2017). As previously mentioned, air pollution has its main source of human activity. Thus, the major anthropogenic air pollutants are divided into two categories, the primary and the secondary pollutants (table 2.1):

Table 2.1: Main anthropogenic air pollutants, adapted from Wright, Richard T., 2019.

Primary pollutants	
Suspended particulate matter (PM):	A complex combination of solid particles and smoke, metals, dust, and salts. Decreases lung function and cardiovascular function (Wright, Richard T., 2019).
Volatile organic compounds (VOC):	As its main source in incomplete combustion of fossil fuels and plants, become carcinogenic components (Wright, Richard T., 2019).
CO:	Has its main source in the incomplete combustion of fuels, also. This gas is invisible, odorless, and tasteless. This pollutant is poisonous because of the ability to bind to hemoglobin and block oxygen delivery to tissues and aggravates cardiovascular diseases (Wright, Richard T., 2019).
Nitrogen Oxides (NO_x):	From fuel and wood burning. This gas is lung irritant, and aggravate respiratory diseases. Contribute to acid rains and ozone formation (Wright, Richard T., 2019).
Sulfur Dioxide (SO_2):	Has its main source in the combustion of coal. It is a poisonous gas that impairs breathing. Contributes, also, to acid rain (Wright, Richard T., 2019).
Lead (Pb):	Its main sources are the batteries manufacture, combustion of leaded fuels, and solid wastes. It is toxic at low concentrations and damages the children's brain (Wright, Richard T., 2019).
Air toxics:	The fuel combustion in vehicles and industrial processes are the principal sources. Some examples are benzene, asbestos, vinyl chloride, known human carcinogens (Wright, Richard T., 2019).
Radon (Rn):	Comes from rocks and soil. It is a radioactive gas that can compromise humans' lungs. (Wright, Richard T., 2019).
Secondary pollutants	
Ozone (O_3):	Photochemical reactions between VOCs and NO_X. It is toxic to animals and plants and highly reactive for the human lungs (Wright, Richard T., 2019).
Peroxyacetyl nitrates (PAN):	Results, also, from a photochemical reaction between VOCs and NO_X. Damages plants and forests, and, in humans, cause irritations of mucous membranes of eyes and lungs (Wright, Richard T., 2019).
Sulfuric acid (H_2SO_4):	Produces acid deposition and damages lakes, soils, artifacts (Wright, Richard T., 2019).
Nitric acid (HNO_3):	Oxidation of NO_X, and produces, also, acid deposition, damaging lakes, soils, and artifacts (Wright, Richard T., 2019).

2.1.1.1 Carbon Monoxide

The CO is a colorless and odorless gas with natural and anthropogenic sources. Causes such as forest fires and volcanic activity are the main natural source of this atmospheric pollutant. Although, the main cause is not natural, and it is due to the combustion processes, from industries and fuel combustion products such as gas, coal, or wood (Fenger, 1999). This atmospheric pollutant needs special attention regarding non-natural causes, once the natural causes emit less than 1 p.p.m (a small slice). Thus, the main percentage of CO in the air has anthropogenic sources (J.Clifford, 2008).

The growing need to control the level of this pollutant is because, at high levels, it causes several problems for the population, fauna, and flora. The biggest issue and the most well recognized pathophysiological effect of CO is tissue hypoxia (oxygen deficiency in tissues), due to its capability to bind the hemoglobin, blocking the tissues to enough oxygen (K. K. Lee, Spath, Miller, Mills, and Shah, 2020). Besides that, studies show that exposure to CO can cause deleterious neurologic sequels in humans in general, and neurocognitive impairment and behavioral abnormalities in children (Block et al., 2012). This pollutant, in high concentrations, also shows some other effects, such as oxidative stress, inflammation, and endothelial dysfunction (Thom, Xu, and Ischiropoulos, 1997; Thom, Fisher, Xu, Garner, and Ischiropoulos, 1999; Lo Iacono et al., 2011).

The most practical way to perceive and act accordingly to reduce risks is to constantly measure the levels of CO and check if they comply with the reference values. According to Portuguese Agency for the Environment (PAE) (PAE, 2013), the reference values to CO air concentration are the ones presented in the table 2.2.

Table 2.2: Carbon Monoxide reference values.

Classification	**μg/m^3**		**ppm**	
	Min	**Max**	**Min**	**Max**
Very bad	400	–	40.4	–
Bad	200	399	15.4	40.3
Medium	140	199	11.8	15.3
Good	100	139	9.4	11.7
Very Good	0	99	0	9.3

These values represent the pollutant concentration in the air. Usually these concentrations are measured in units of the mass of chemical (milligrams, micrograms, nanograms, or picograms) or in Parts per Million (ppm) (10^{-6}) or in Parts per Billion (ppb) (10^{-9}).

2.1.1.2 Sulfur Dioxide

With a lifetime of one week on average (Seinfeld, Pandis, and Noone, 1998), SO_2 might be found in the atmosphere caused by natural and anthropogenic sources. It is estimated that 35-65% of total SO_2 emissions (Cheremisinof, 2002) stand out from the volcanic eruptions and organic matter decomposition

(Liu, Pellikka, Li, and Fang, 2019). However, the main source of SO_2 resulting in human activities is the combustion of fossil fuels, once this generates small proportions of other sulfur components, such as Sulfur Trioxide (SO_3), H_2SO_4, and sulfates (Gimeno, Marín, Del Teso, and Bourhim, 2001). Besides that, thermal power plants burning high-sulfur coal or heating oil, industrial boilers, nonferrous metal smelters, domestic coal burning, and even emissions from vehicles are other anthropogenic sources of SO_2 (Cheremisinof, 2002). Further, SO_2 is a big precursor of sulfate, which is one of the principal causes of the formation of fog-haze (A. K. Lee, 2015).

Ambient SO_2 may directly impact human health since it is a precursor of sulfate (Knibbs et al., 2018). Beyond this, studies show that this pollutant might cause cardiovascular, ischaemic heart, and respiratory diseases once the it dissolves into the watery fluids of the upper respiratory system and is absorbed into the bloodstream (Sunyer et al., 2003; Wang et al., 2018; Zhang, Di, Liu, Li, and Zhan, 2019). The principal symptoms are irritation of the eyes, nose, and throat. To risk groups, such as children, the elderly, and those already suffering from respiratory ailments (e.g asthmatics) are especially at risk. Health impacts can occur even to brief exposures (concentrations above 1,000 pg/m3).

Studies have shown that plants also suffer from expose to this pollutant once they may lose their foliage, become less productive, or die prematurely. Some species are so sensitive that they manifest negative impacts, even with small concentration exposure. Thus, it presents a risk to the normal ecosystem behavior (Cheremisinof, 2002).

According to PAE (PAE, 2013), the reference values to SO_2 air concentration are the ones presented in the table 2.3.

Table 2.3: Sulfur Dioxide reference values.

Classification	**μg/m^3**		**ppm**	
	Min	**Max**	**Min**	**Max**
Very bad	500	–	1004	–
Bad	350	499	703	1001
Medium	210	349	210	349
Good	140	209	162	329
Very Good	0	139	0	160

These concentration values are also usually measured in units of the mass of chemical (milligrams, micrograms, nanograms, or picograms), ppm, or ppb.

2.1.1.3 Ultraviolet radiation

The UV index was created to inform the population of the UV radiation index risk and are needed to better inform populations-at-risk and cultivate changes in mindset and attitudes against exposure to the sun (Igoe, Parisi, and Carter, 2013).

Several factors can be appointed as aspects that affect the solar UV index:

- Atmospheric attenuation of incident solar radiation. This angle (between the sun's rays and the normal on a surface), when minimum, corresponds to the minimum index. This angle is influenced by time of day, latitude and season (Allaart, van Weele, Fortuin, and Kelder, 2004; McKenzie, Aucamp, Bais, Björn, and Ilyas, 2007);

- An Ozone and atmospheric aerosol concentrations, along with cloud cover, affect the clear-sky influence of the incident angle by increasing solar UV attenuation through scattering and absorption processes (Klumpp et al., 2006). Ozone strongly attenuates UV radiation. The depletion of the ozone layer is likely to aggravate existing health effects caused by exposure to this radiation, onve stratospheric ozone is an effective UV radiation absorber. As the ozone layer becomes thinner, the protective filter provided by the atmosphere is progressively reduced. Consequently, human beings and the environment are exposed to higher UV radiation level, having a dangerous impact on humans, animals, and marine organism's health, as well as in the plant's life (Wright, Richard T., 2019);

- The atmospheric absorption by aerosols (e.g., dust, smoke, and anthropogenic pollutants such as vehicle exhausts) tends to reduce incident radiation at the earth's surface, absorb the radiation, and, consequently, the UV index (Madronich, McKenzie, Björn, and Caldwell, 2003; Kirchstetter, Novakov, and Hobbs, 2004; Román et al., 2013; Downs, Butler, and Parisi, 2016);

- Clouds cover might reduce the UV Index when compared to a cloud-free situation (Kuchinke and Nunez, 1999), but, in contrast, a day with clouds can present a higher UV index when compared with a cloud-free condition, depending on the surface UV radiation (Mayer and Kylling, 2005).

As mentioned earlier, high UV radiation causes impacts on human health. This impacts might be negative or positive. Since part of the UV radiation comes from the sun, it has benefits for human health: it suppresses negative thinking and stress, improves sleep, prevents some illness, increases production of vitamin D in the human body (which is necessary for keeping muscles and bones healthy). On the other hand this may cause eye problems such as ocular melanoma, skin problems, such as skin cancer (melanoma and non-melanoma skin cancers), besides the premature skin aging (Norval et al., 2011).

According to World Health Organization (WHO) (World Health Organization, 2017), the reference values, to UV index levels are defined between 1 and 11+, as presented in the table 2.4.

Table 2.4: Ultraviolet Indexes reference values.

Level	Value
Low	1-2
Medium	3-5
High	6-7
Very high	8-10
Extreme	11+

2.1.2 The role of water in Environmental Sustainability

The sustainability of life in today's society is largely based on an essential element, the water. Therefore, the quality of this resource is very important once, when it is compromised, it becomes unfeasible to serve the population and aquatic living beings. Thus, an entire ecosystem can be compromised.

It is essential to understand that one of the main causes of the aquatic ecosystem contamination are the anthropogenic activities, responsible to release waste into the aquatic environment, every day. The main waste accountable for water contamination accrue from domestic and industrial water utilization. Water from those activities is called wastewater. This wastewater characterizes itself for the presence of substances like solids, oils, fats, organic material, nutrients, and toxic substances (Spellman, 2013). If this wastewater returned to water resources again (without any treatment) would be harmful to the environment. Due to these consequences, adequate treatment is necessary so that environmental aggression, arising from the discharges of such residues, shall be reduced. Therefore, it is clear that wastewater management has a central role in sustainable development, and, in this context, WWTP emerges, responsible for such management (Mendonça et al., 2013).

2.1.2.1 Operations of a Wastewater Treatment Plant

Water is used in many contexts: at home, industries, agriculture, among others. But, it is important understand what happens to the residual water that results from that use. First, this water is collected and transported to a plant in water pipes (a network that connects the water sources to the plant). In this plant, the residual water, commonly called wastewater, is treated and returned to water sources, in environmentally safe conditions. This plant is called a WWTP.

Further, there are two main lines in a WWTP: a wastewater line and a sludge line. The first one, a liquid line, regards the wastewater treatment to fulfill the conditions required in the discharge license. On the other hand, the sludge line concerns the solids treatment, the ones removed from the wastewater.

Figure 2.1 shows a schematic representation of how a WWTP works.

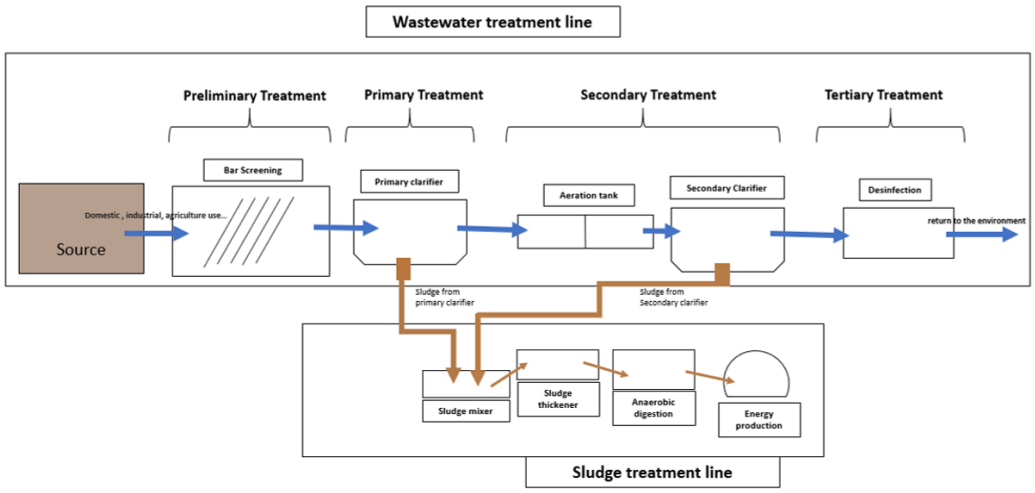


Figure 2.1: Scheme of a WWTP operation.

As previously mentioned, a WWTP works fundamentally through two lines: the line accountable to the wastewater treatment itself, and the line that handles the sludge. The wastewater treatment line follows several phases, described in detail below (Metcalf and Eddy, 2003):

1. **Pre-treatment**

   The pre-treatment is the first wastewater treatment process phase. Here occurs the removal of coarse solids and floating materials. This first step is crucial once remove some components that may damage the WWTP equipment.

   - Bar Screening: In this stage, the solids pass through a bar screening that works as a large filter to remove large residues.

2. **Primary Treatment**

   The second phase is the primary treatment. Here, part of the suspended solids are removed (smaller than the previous ones) as well as organic matter from the effluent wastewater.

   - Primary Clarifier: Using a clarifier some remaining solids are removed by the action of the gravity. The solids resulting from this process are called primary sludge.

3. **Secondary Treatment**

   In the secondary treatment occurs the biodegradable organic matter removal (suspended or in solution), along with suspended solids and nutrients (such as nitrogen and phosphorus). The biological process of this phase, may happens into two fundamental kind of conditions: aerobic and anaerobic conditions. The produced biomass is removed afterwards from the system by decanting.

   - Aeration tank: Here occurs a biological process, in which the organic matter that is considered a pollutant is removed from the water, consumed by microorganisms present in this tank. This tank has two distinct zones: an aerated zone and an anoxic zone. The first one represents an oxygen-rich zone, while the second presents an oxygen deficient environment. Thus, the microorganisms and processes that require a high oxygen presence act in the aerated zone while those that don't need act in the anoxic zone.
   - Secondary Clarifier: In this clarifier, the remaining suspended solids (many of these resulting from the previous biological process) are separated from water, by the action of the gravity.

4. **Tertiary Treatment**

   In this phase, the remaining solids (not eliminated in the secondary treatment) are removed, together with organic matter and toxic compounds. The disinfection step also occurs here, by removing the remaining nutrients (several times using UV radiation to disinfect the water).

5. **Discharge to surface water**

   The last phase is the process in which water is returned to the hydric resources, safely, without compromising the ecosystems.

After understanding how a wastewater treatment line works, it is essential to mention how the solid line works as well. It constitutes the solids treatment (the ones removed from residual water in the liquid phase) called sludge. The principal treatment in this line is, therefore, the sludge treatment. This sludge is treated to inactivate pathogenic organisms and reduce the sludge's volume. Besides, they may produce bio-gas, which may be used to produce energy, used in the WWTP process, promoting the economic sustainability of the plant.

2.1.2.2 Wastewater characteristics and quality

There are some wastewater characteristics that may be explored to determine its quality. Further, there are psychical, chemical, and biological parameters that may be studied and evaluated in a WWTP. As physical wastewater parameters, it is possible to highlight some, such as:

1. **Solids**

   Depending on the size and source, solids are the main attribute that works as water contaminants. There are dissolved and suspended forms. Depending on size, even the smallest are considered a problem, once provides the accumulation of harmful agents (chemical and biological), being a risk for human beings. In the wastewater treatment process, there are several phases to erase these solids (Spellman, 2013; B.Baird, D.Eaton, and W.Rice, 2017). In the wastewater, the total solids concentration is usually between 350 and 1200 mg/L (Spellman, 2013). Table 2.5 present the range of solids that can be evaluated in a WWTP system.

Table 2.5: Type of existing solids in a Wastewater Treatment Plant (B.Baird, D.Eaton, and W.Rice, 2017).

Solids designation	**Description**
Total Solids	In a water sample, the total solids are the left ones after evaporation and subsequent oven drying at a defined temperature (usually 150°C).
Total Suspended Solids (TSS)	The quantity of solids in a water sample that passes through a filter with a pore size equal to 1.2 μm.
Total Dissolved Solids (TDS)	The quantity of solids that keep retained in the filter, this is, bigger than 1.2 μm.
Fixed solids	The total amount of solids, suspended or dissolved, remaining in a sample after a calcination process, in a temperature rounding 550°C.
Volatile solids	The total amount of solids, suspended or dissolved, lost from a sample after the calcination process mentioned in the preceding point.

2. **Turbidity**

   This is related with the water's clarity. This is measured considering how absorbed or diffused is the light by the suspended material. The principal source contributing to water turbidity are detergents, soaps, and general emulsifying agents. This parameter has special attention in drinking water. As previously mentioned, the wastewater disinfection process may be done using UV radiation. For this process works accurately it is vital the UV radiation penetrates the water. Turbid water difficult this action and, consequently, the disinfection process (Spellman, 2013).

3. **Color**

   Water considered pure is colorless. The color comes from substances such as vegetation, mineral, and aquatic organisms. In the water treatment process, the color is generated by dissolved solids that remain after the removal of suspended matter. In wastewater treatment, color is just an indicator of the wastewater state. In a primary stage, the wastewater is light brownish-gray color. Furthermore, the water containing Dissolved Oxygen (DO) (another important water parameter) is usually gray. At the end of the process, the water should be colorless (Spellman, 2013).

4. **Odor**

   Odor is an important characteristic of wastewater treatment. The odor of the wastewater comes from industrial waste, anaerobic conditions, and operational problems. In wastewater, this characteristic requires special attention, once people are residing nearby. Usually, these odors are generated by gases from organic matter or substances that are added to this wastewater (Spellman, 2013).

5. **Temperature**

   In a WWTP, the wastewater temperature varies according to the stage at which the water is along the system. Wastewater is generally warmer when compared with the regular water. In the wastewater treatment process, the temperature is a very important parameter that once influences the role of the microbial population in the gas transfer rates and the sedimentation process of the biological solids (Spellman, 2013). Besides that, it is important to refer that when the DO in the wastewater reduces, the temperature of the wastewater increases (Colacicco and Zacchei, 2020). The limit temperature value for superficial water is around 28° (PAE, 2013).

6. **Conductivity**

   This parameter represents the concentration of ions with a capacity to conduct electric current. Therefore, the presence of a large amount of these ions in a wastewater sample implies a higher electrical conductivity value (B.Baird, D.Eaton, and W.Rice, 2017).

As mentioned before, there are also chemical wastewater characteristics:

1. **Organic parameters**

   Proteins, proteinaceous materials, oils, greases, and carbohydrates are the main wastewater organic components that come mostly from food. Oils and greases may come also from spills, or other small discharges containing petrol. Besides that, activities such as agriculture are the main users of pesticides and herbicides, that are harmful to water quality. Additionally, detergents are also organic molecules that are presented in wastewater. This kind of components, when not controlled, may cause problems in the WWTP (Spellman, 2013):

   - High levels of grease can produce clogging of filters, nozzles, and sand beds. Additionally, if the grease is not removed before the WWTP discharge , this can damage the normal water biological processes and contribute to the appearance of floating matter;
   - The presence of detergents, pesticides, and herbicides may influence the biological processes in the WWTP, once reduce the oxygen uptake in this procedures. Further, may kill useful living organisms with a significant role in the WWTP processes.

2. **Inorganic Parameters**

   - Alkalinity: Alkalinity is the capacity of the water to neutralize the acid. High values of alkalinity make water unpleasant, with some impact on human and aquatic health. Alkalinity in wastewater treatment has a crucial role once holds its neutral pH during the biological treatment processes (Spellman, 2013);
   - pH: pH refers to the hydrogen ions concentration in water. This concentration is responsible for many reactions in living organisms. A pH of less than 5 indicates acid water, greater than 9, alkaline. Identify the pH value in a WWTP is essential since this is a controlling agent of the biological and physical-chemical wastewater functions. The pH must be maintained between certain appropriate levels (6-9) for the maintenance of microbiological community, and its development in the treatment process. Domestic wastewater has a pH very close to neutrality, nevertheless, when it comes from industrial sources, it can assume very different values (alkaline water). This parameter deserves special attention, mainly in the aeration thank. Here, occurs a biological treatment, with the action of several microorganisms (that need the right conditions to develop its role). In the secondary clarifier, the pH values should be monitored since it allows spotting some dysfunctions. At the exit of a WWTP (the discharge process), the pH value must comply with the emission limit values considered by law. According to PAE (PAE, 2013), these values must be included between 6.5 and 8.5 to be classified as “excellent” and between 5.5 and 9 to be considered “good” (Spellman, 2013);
   - Nitrogen (N): In a WWTP, N can be found in several forms: nitrate (NO_3-), nitrite (NO_2-), ammonia (NH_3), and organic nitrogen (in order of decreasing oxidation) (B.Baird, D.Eaton, and W.Rice, 2017). This is an important component in wastewater treatment once take part of

the nitrification and denitrification processes. The nitrification process concerns the oxidation of ammonia to nitrite and then to nitrate, performed by bacteria, in order to obtain energy. This process occurs in an aerated environment, which means, a high presence of oxygen (once ammonia is unstable in this context). At the denitrification process occurs the nitrate reduction to nitrites and subsequently to Nitrogen gas (N_2), released after to the atmosphere (3.2–10% of the total anthropogenic emission of nitrogen) (Law, Ye, Pan, and Yuan, 2012). This activity take place in the anoxic zone, an environment with low presence of oxygen (Sun, Cheng, Sha Li, Liu, and Sun, 2013). Nitrogen data are essential for evaluating the treatment of wastewater by biological processes (Spellman, 2013);

- Phosphorus: Phosphorus is a macronutrient that is necessary for all living cells. Municipal wastewater may contain 10 to 20 mg/L phosphorus, much of which comes from detergents (Spellman, 2013). Further, this parameter is related to the water conductivity, once a higher conductivity implies a higher phosphorus amount in a water sample (Bayo, López-Castellanos, and Puerta, 2016);
- Heavy metals: Heavy metals are toxicants and are found, mainly, in industrial wastewater, especially the automotive industries. This component affects the biological treatment of wastewater once is toxic, even to the microorganisms (Spellman, 2013);
- DO: It is possible to find DO in water. The main impact of this parameter is in the the aeration tank microorganisms' functionality, once they need oxygen to perform several processes, because of the role it plays in pH and alkalinity (Spellman, 2013);

3. **Bacterias** and **Viruses**

In wastewater, it is possible to find some bacteria and viruses. Bacteria are found, as previously mentioned, in the biological processes, such as nitrification and denitrification. Besides, they are also found in the degradation of organic matter processes. Further, other bacteria are presented in the wastewater, in which human and animal wastes are the primary source. These are a problem and have to be eliminated (Spellman, 2013). It is possible, also, to detect viruses in wastewater, mainly excreted by humans. This constitutes a problem, once are difficult to detect. (Spellman, 2013).

2.2 Machine Learning

A significant difference between humans and computers is how humans can automatically change their behavior through learning from previous mistakes. Humans try to solve problems, correcting them, or finding a new way to deal with them (J. Carneiro, Saraiva, Martinho, Marreiros, and Novais, 2018). Besides, they rely on common sense to draw their conclusions. Analyzing traditional computer programs, they do not look at the results of their tasks and, therefore, are unable to improve their behavior. Thus, the concept of ML appears, addressing this problem. Its essence is the creation of models and tools that can learn and consequently improve their performance, through continuous data collection, resulting in experience and expertise (Global, Llp, Corporation, Llp, and Global, 2017; Analide, Novais, Machado, and Neves, 2006).

Being a field of computer science, more specifically a field of AI, ML shares knowledge with mathematics, statistics, game theory, and optimization, among others. It does not seek to automatically imitate intelligent human behavior, but rather to use it as special strengths and abilities of computers to complement human intelligence, allowing the development of tasks that go beyond the human capacities D. Carneiro, Novais, Miguel Pêgo, Sousa, and Neves, 2015. For example, the ability to examine and process large amounts of data, once ML models can detect patterns that are unattainable for human capacity (Mitchell, 1997). Thus, this field focuses on building algorithms to detect patterns and predict classes from data. Its task is identifying a function $f : x- > y$, where x represents the input , and y the output, this is, the predictions or patterns (Bekkerman, Bilenko, and Langford, 2011; Zhou, Tang, and Zheng, 2015).

To be aligned with the ML concept it is important to understand the notion of learning. Learning is the process to convert experience into expertise or knowledge (Bekkerman, Bilenko, and Langford, 2011). A learner model input is expressed by training datasets, that represent the experience. On the other hand, the output is the expertise, which, usually, are models that can develop tasks, find meaningful patterns, and predict with good precision (Shalev-Shwartz and Ben-David, 2013).

2.2.1 Learning paradigms

There are three main learning approaches (Libbrecht and Noble, 2015).

2.2.1.1 Supervised Learning

In the supervised learning the data used by the learner model has information about the result. This type of learning uses labeled examples, using inputs with known and corresponding outputs to train the models. It is used when historical data is available, and the goal is predicting future outcomes. For this reason, it is possible to compare the values created by the system with the expected ones. There are many examples of models that use supervised learning: Decision Trees, Random Forest, Support Vector Machine, Naive Bayes Classifier, Bayesian Networks, Neural Networks, among others (Qiu, Wu, Ding, Xu, and Feng, 2016).

2.2.1.2 Unsupervised Learning

Models based on unsupervised learning use unlabeled data to the training process, emerging expertise as an output. The system does not know the answer, just find out the data patterns, using only inputs for that. In this type of models, is available the input data, but the outcomes are unknown before the models being implemented and performed (Qiu, Wu, Ding, Xu, and Feng, 2016). K-means and K- Nearest neighbors are examples of unsupervised learning models.

2.2.1.3 Reinforcement Learning

Reinforcement Learning is the training of ML models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain complex environment. Facing several situations, the solutions are find using trial and error. For the machine understand what are the goals, receives rewards or penalties for the actions it performs. Thus, the goal is to maximize the total reward (Song, Li, Quan, Yang, and Zhao, 2021).

2.2.2 Machine Learning Models

Aforementioned, there are several ML models. Following will be addressed four supervised learning models, with an increasing degree of complexity.

2.2.2.1 Decision Trees

The most popular algorithms for the construction of Decision Trees are Iterative Dichotomiser 3 (ID3), C4.5, and Classification And Regression Tree (CART). ID3 was developed by J. Ross Quinlan, during the 70s, which created a simple tree construction from the roots to the leaves (top-down construction). Despite this, this algorithm presented a clear limitation, it was only able to deal with nominal variables and, consequently, be used in classification problems. To solve this problem, Quinlan created a new algorithm, the C4.5, considered an improvement of ID3. Thus, it allows continuous numerical variables, being able to deal with regression problems (in addition to the classification problems). Besides, other improvements emerged, such as the trees pruning possibility. This last is considered an important addition since it allows removing branches that cause “noise” in the data and, consequently, an improvement in the prevision’s accuracy. Another algorithm developed by a group of statistics (L. Breiman, J. Friedman, R. Olshen, and C. Stone), in parallel with the mentioned before, is CART. This is an algorithm with a very similar approach to C4.5, used for classification and regression models, also (Han, Kamber, and Pei, 2012).

Decision Trees are generated from a set of training data, resulting in the creation of a tree-shaped structure used to classify new cases. Each case is described as a set of features or attributes associated with each case of the training data. The output is the value/name of the class, that is, the target feature, the one that is predicted (Quinlan, 1993). Usually, a Decision Tree adopts the structure represented in figure 2.2.

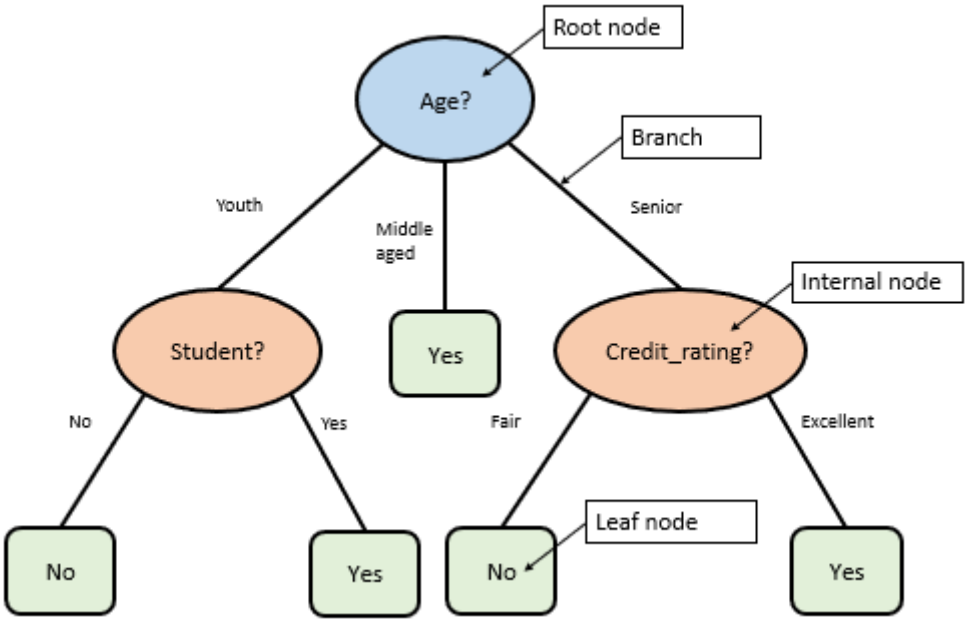


Figure 2.2: Example of a Decision Tree regarding a credit assignment, adapted from Han, Kamber, and Pei, 2012.

As the main constituents of a tree, arises:

- **Root node:** Represents a variable, and it is the first Decision Tree node;
- **Branch:** Represents the outcomes of each test executed in a node. Each Decision Tree test is based on conditions if-then and the result of each test is used to decide which branch is followed next;
- **Internal Node:** Represents also a variable and involves all the tree nodes other than the root node and the terminal nodes;
- **Leaf Nodes:** These are the Decision Tree terminal nodes. When a test feature reaches a leaf node it contains class labels instead of the tests from the other tree nodes. In this way, as soon as the feature reaches this terminal node, a class is assigned.

As mentioned before, the Decision Tree model adopts a top-down construction, following some important steps and parameters (Han, Kamber, and Pei, 2012).

As input the model uses:

- A training dataset, a set of tuples and the corresponding classes, represented by F ;
- An attribute list, a list of all the features that belong to the training dataset;
- The chosen attribute selection method, the procedure responsible to establish the partition method that best split training data tuples into individual classes.

As an output the model creates a Decision Tree, capable to classify each case. The Decision Tree building process follows some steps (Han, Kamber, and Pei, 2012):

1. The tree begins with a single node, D, that, at this instance, represents all the training dataset (the root node);

2. If all tuples from F belongs to the same class C, the node D becomes a leaf with that class assigned;

3. Otherwise, the algorithm defines the splitting criterion according to the attribute selection method chosen. The splitting criterion define:

    - Which attribute test at node D, which define the best way to partition the training dataset into individual classes;
    - Which branches to grow from node D, according to the outcomes of the tests carried out in the previous node.

4. The node D is labeled with the splitting criterion. Each branch that grows from this node is one of the splitting criterion outcomes. Let X be the splitting attribute, with v distinct values, $x_1, x_2, x_3 x_v$ and three possible splitting scenarios:

    a) **X as a discrete value:** The test outcomes at node D are all the different possible values from X, represented by each branch that grows from the node, x_v. The partition F_j is the subset of F that contains all the tuples with the x_v value, from X. Once all the tuples in each partition F_j have the same value for X, there's no need to be partitioned anymore. For this reason, X is removed from the attribute list. The figure 2.3 shows an example of this.

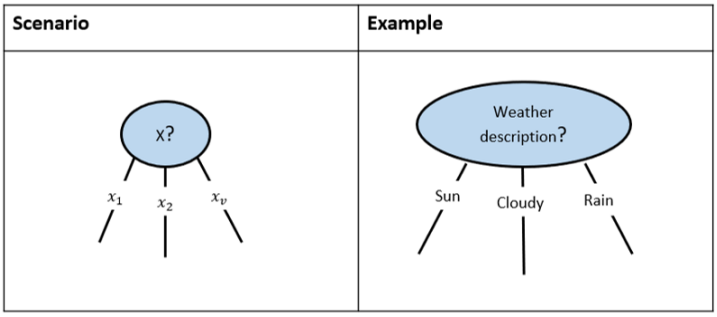


Figure 2.3: Example of a discrete attribute, adapted from Han, Kamber, and Pei, 2012.

    b) **X as a continuous value:** In this case, there are two possibilities as test outcomes: $X \leq split\ point$ and $X > split\ point$. The split point is defined by the attribute selection method chosen. Two branches result from node D and each one represents the possible outcome. The tuples are split such that F_1 is a subset where all the tuples respect the condition $X \leq split\ points$ and the subset F_2 the remaining ones, as figure 2.4 shows.

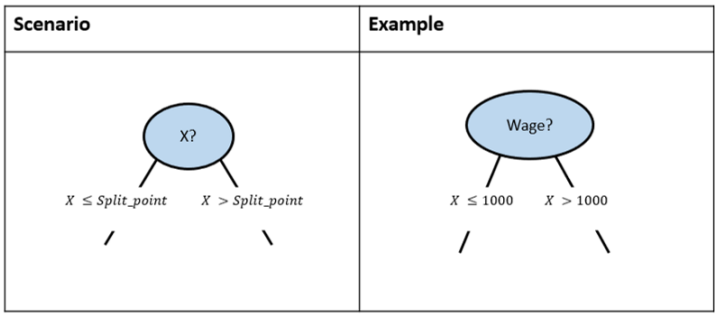


Figure 2.4: Example of a continuous attribute, adapted from Han, Kamber, and Pei, 2012.

c) **X as a discrete value that can generate a binary tree:** This specific case depends on the used algorithm and attribute selection measure chosen. The test performed at node D follow the condition "$X \in S_X$?", where S_X represents a subset of tuples from X, which results from attribute selection measure chosen as a splitting criterion. A value from X, x_v, is tested in the following away: if $x_v \in S_x$, then the test at the node D is positive. Two branches are grown from the node,and, by convention, the left branch is labeled "yes" and the right one labeled "no". As a result, there are two subsets, F_1, corresponding to the tuples that satisfy the test, (labeled "yes") and F_2, the remaining ones (labeled "no") as figure 2.5 shows.

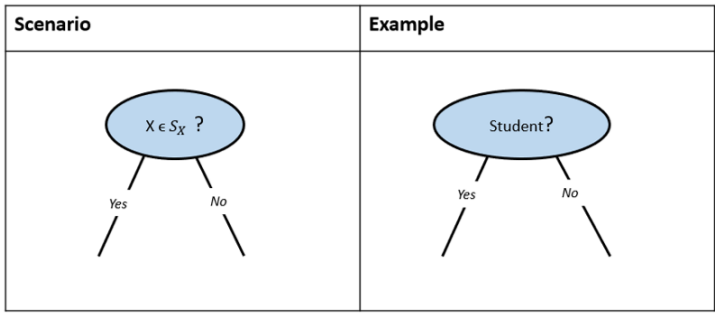


Figure 2.5: Example of a discrete attribute that can generate a binary tree, adapted from Han, Kamber, and Pei, 2012.

5. The process stops and the classes are assigned when:

   - All the tuples from F belongs to the same class C;
   - There are no left attributes on which the tuples may be partitioned. In this specific case, the majority voting is used, the node becomes a leaf and is labeled with the most frequent class;
   - There are no remaining tuples, that is, the partition F_j is empty. A leaf is created and the most common class in F is assigned.

Decision Trees are considered a good approach once they allow us to know the importance and impact of each variable in the result. That is, a variable located closer to the root of the tree produces a better effect on the result than a variable closer to the leaf nodes of the tree (Raglio et al., 2020).

An attribute selection measure is a heuristic to appoint the splitting criterion that separates a given training data partition, F, classified individually. If F is divided into smaller partitions according to the division results, ideally, each partition would be pure, that is, all instances that end up in each partition would belong to the same class. For this reason, the best measure chosen is the one that considers this "pure" concepts and makes the best split according to that. Besides that, attribute selection measures determine how the instances at a given node are to be split and, due to that, are also denominated splitting rules (Han, Kamber, and Pei, 2012). Thus, there are three attribute selection measures, described below:

- Information Gain

  Information Gain is an attribute selection measure based on entropy in its reckoning. It is the measure used in the ID3 algorithm. Entropy is an uncertainty measure that allows figure out which attribute should take place in each tree position. It is associated with a set of objects that allows identifying the degree of data disorder (low purity), calculated for each one of the attributes under study (Han, Kamber, and Pei, 2012). The entropy calculation is based on the formula (equation 2.1) (Yin, Langenheldt, Harlev, Mukkamala, and Vatrapu, 2019):

$$E(F) = -\sum_{i=1}^{m} pilog_2pi \tag{2.1}$$

  Where pi presents the probability of each class i appears in a dataset, computed over a total of m classes.

  Entropy varies in a range between 0 and 1. Values closer to 1 meaning a high level of data disorder and a low level of purity. On the other hand, values closer to 0 meaning the opposite. If values are closer to 0 this represents, also, that the data is distributed in a homogeneous way, there is no predominance of classes in that dataset. The graph from figure 2.6 shows how entropy varies according to the probability (p) of each class i appears in the dataset.

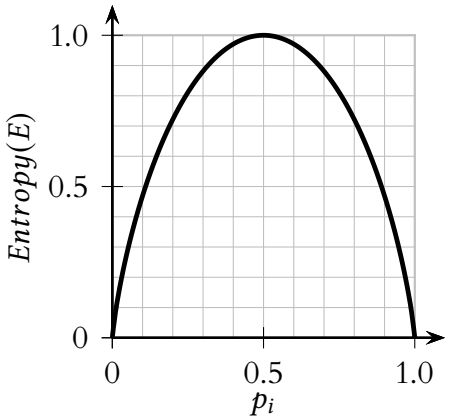


Figure 2.6: A graphical view of entropy.

  The information gain measures the expected reduction in entropy as the dataset is partitioned. The attribute with the greatest reduction in entropy and, consequently, the greatest gain in information

is considered the one that must take up the tree root (since it produces a better effect). Occupying the root position allows reducing the tree depth (Mingers, 1989).

The calculation of the information gain is given by the expression 2.2 (Yin, Langenheldt, Harlev, Mukkamala, and Vatrapu, 2019):

$$Information\ gain(X) = E(F) - \sum_{j=1}^{v} \frac{|F_j|}{|F|} \times E(F_j) \tag{2.2}$$

The expression represents:

- $E(F)$: Entropy before the partitioning of dataset F;
- $\sum_{j=1}^{v} \frac{|F_j|}{|F|} \times E(F_j)$: Quantifies how much more information do we still need (after the partitioning) to the partition be pure, this is, all the tuples from F_j subset belong to the same class:
  - $\frac{|F_j|}{|F|}$: It represents the weight of jth partition, in the total dataset;
  - $E(F_j)$: The entropy of the subset F_j that results from the partitioning at the attribute X. The lower this value, the purer the partition.

So that, by the formula analysis, the information gain results from the difference between the original requirement of information (before the partitioning) and the new current requirement (after partitioning). So, tells how much information gained when dividing the attribute X into several branches. The process is repeated for all the constituent attributes of the training dataset. These would occupy the tree from the top to the bottom according to information gain (the higher the closer to the root).

The information gain measure tends to be skewed against tests with numerous outcomes (e.g. attributes that acts as a unique identifier, such as Id's). In this case, each partition is pure, once each one contains only one tuple of data and, consequently, the information gain with being maximal. Certainly, this situation would be useless to classification problems. To solve this problem, another measure is used, the gain ratio, addressed below.

- Gain Ratio

  As previously mentioned, this measure of attribute selection has been solving some of the limitations and specificities of information gain. Thus, this measure is used in the algorithm C4.5 (ID3 successor) and is considered an extension of the information gain. The following equation (2.3), shows the potential information that is created through the partitioning of the training data set, D, in a total of v outcomes, resulting from the test at the attribute X (v partitions).

$$Information\ split\ in\ X\ gain(F) = -\sum_{j=1}^{v} \frac{|F_j|}{|F|} \times log_2(\frac{|F_j|}{|F|}) \tag{2.3}$$

  Here, to each outcome, it considers the number of tuples that have that possibility, in the total number of tuples in F. The difference between this measure and information gain,is that this last just considers the classification that is obtained based on the same partitioning. Finally, the gain ratio is calculated as the following equation, 2.4, shows (Han, Kamber, and Pei, 2012):

$$Gain\ Ratio(X) = \frac{Information\ Gain(X)}{Information\ split\ in\ X\ gain(F)} \tag{2.4}$$

- Gini index

  This measure is applied in the CART algorithm. Is used to measure the impurity of a dataset or subset (in this specific case, the dataset D or a data partitioning resulting from it). Gini index is computed as equation 2.5 shows (Han, Kamber, and Pei, 2012):

$$Gini(F) = 1 - \sum_{i=1}^{m} {p_i}^2 \tag{2.5}$$

  Where p_i represents the probability that a tuple in D belongs to class C_j, computed over a total of m classes. This measure contemplates a binary split for each attribute. There are two possibilities to consider: the attribute as a discrete value or as a continuous value (Han, Kamber, and Pei, 2012).

  Considering the case where X is a discrete-value, and there is v distinct values to that attribute, in D. In this case, to evaluate the best binary split on X, we analyze all the potential subsets which can be generated using the values of X. Each subset, S_X, can be considered as a binary test for the attribute X of the form "X $\in$ S_X?". Given a tuple, this test is satisfied if the value of X for the tuple is among the values listed in S_X. If X has v possible values, then there are 2^v possible subsets. Is excluded the set with all the possible outcomes, a_1, a_2, a_v, and the empty set, { }. For this reason, there are 2^{v-2} possibilities to form two partitions resulting from D, form by a binary split on X. When considering a binary split, it is calculated a weighted sum of the impurity of each resulting partition, the Gini Index of D, giving a binary partitioning, represented by 2.6 (Han, Kamber, and Pei, 2012):

$$Gini\ in\ X(D) = \frac{|F_1|}{|F|} \times Gini(D_1) + \frac{|F_2|}{|F|} \times Gini(D_2) \tag{2.6}$$

The expression represents a split on partitions D into two, D_1 and D_2. For every one attribute, each of the possible binary splits is considered. In the specific case of discrete values, the subset with a minimum Gini index for that attribute is selected as its slipping subset (Han, Kamber, and Pei, 2012).

On the other and, for continuous-valued attributes, split-points must be considered. The strategy of the midpoint (between each pair of adjacent values), is used as a possible split-point. For a possible split-point of X, D_1 is the set of tuples in D satisfying $X \leq split\ point$, and D_2 is the set of tuples in D satisfying $X > split\ point$. The reduction in impurity that would be incurred by a binary split on a continuous attribute X, is presented in equation 2.7 (Han, Kamber, and Pei, 2012):

$$\Delta Gini(X) = Gini(D) - Gini\ in\ X(D) \tag{2.7}$$

The attribute that maximizes the minimum Gini index value, that is, the one that maximizes the reduction in impurity, is selected as the splitting attribute (Han, Kamber, and Pei, 2012).

Further, another important issue about Decision Trees is the tree pruning. The tree pruning is the process of deleting internal-nodes from a Decision Tree (Ayyadevara, 2018). Pruning methods have been developed to allow a Decision Tree model to stop until it is no longer possible to split it (Mingers Bsrcd, 1989). This happens because can occur overfitting, and the tree pruning may help to solve this. Overfitting occurs when the data fits so well the training dataset, that when the model is confronted with the real "world", that is, a new input (different from the training dataset) the model turns out to be poor (Shalev-Shwartz and Ben-David, 2013). Thus, the tree pruning allows noisy data or redundancies to be eliminated from the model. The tree pruning process may use two different techniques: The pre-pruning and the post-pruning. The first one concerns that the tree construction process stops when it is considered that there are not advantages to have more splits in the tree. However, the post-pruning is based on let the tree grow as far as it can, and after prune the unneeded nodes (Mingers Bsrcd, 1989).

Last, Decision Trees model has some advantages but also some limitations. When compared with other models, Decision Trees are considered simple and good models once achieve good predictions with reduced endeavors. For this reason, there are advantages to using Decision Trees (Kashyap, 2017;Han, Kamber, and Pei, 2012)). Decision Trees are a fast, simple, intuitive, and inexpensive model. Besides this, this model provides a good view of which attributes and fields are the most significant for classification and prediction, and can deal with categorical and numerical data. Even so, present some limitations, that can be pointed out. This is a model that does not produce good results if there are many uncorrelated variables in the data set. Further, for this model, data sets more complex would generate tress also more complexes. So, although simple to implement, this huge tree has numerous divisions and many sub-trees, which takes a lot of time and can become computationally expensive.

2.2.2.2 Random Forest

Another ML model considered is Random Forest. This model has Decision Trees in its constitution and it is considered a model with good results and high precision predictions (Kumar, 2017; Qi, Jin, Li, and Qian, 2020).

In Random Forest, each Decision Tree is constructed using a subset of data chosen randomly from the training data. Each subset will create a tree, and so, when testing the model with an input, each tree will classify with the respective class. Thus, the output that will result from a this model will be, for example, the mode of the classes generated by the individual trees (Pal, 2005).

Figure 2.7 shows an example of how the Random Forest model works.

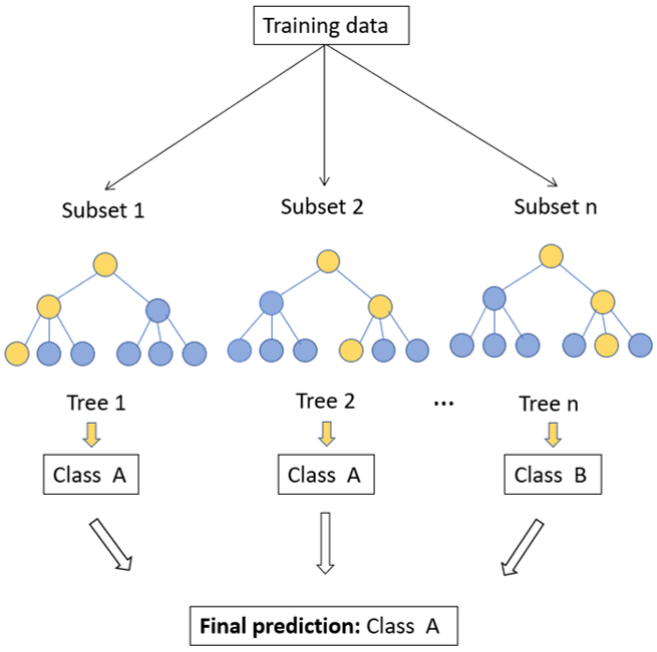


Figure 2.7: Scheme of a Random Forest model operation.

This model also has its advantages and limitations. It operates successfully on large datasets and handles fairly well with outliers and models "noise". Further, requires less computational efforts, when compared to other tree ensemble models. Furthermore, just like the Decision Trees, this model enables us to understand which variables are important. Although this model usually obtains better results when compared to Decision Trees, it requires higher computational efforts (Kumar, 2017).

2.2.2.3 Artificial Neural Networks

Artificial Neural Network (ANN) are a deep learning model, with a higher degree of complexity, when compared with the models previously mentioned. ANN is a deep learning model that performs based on human brain functioning. The basic unit of an ANN is the artificial neuron (units, cells, or nodes), which also represents the human neuron. Equally to the human brain, these neurons are connected between them, creating a network (Fausett, 1994).

In ANN models, neurons are responsible to process information. Thus, the signs generated from each neuron are transmitted to the next neurons, between a communication link. These links have a weight, which is responsible to multiply the transmitted signals. To create an output, each neuron applies an activation function for every input that reaches it. The ANN training process consists of that weights updated over iterations until produce meaningful results. These iterations are called epochs and are an important parameter in ANN models (Heaton, 2012). Thus, an ANN is constituted by layers of neurons, that are connected between them.

A neural network with more than two layers is considered deep, as figure 2.8 shows. The first research about deep neural networks was developed by Pitts and McCulloch, in 1943 (McCulloch and Pitts, 1943).

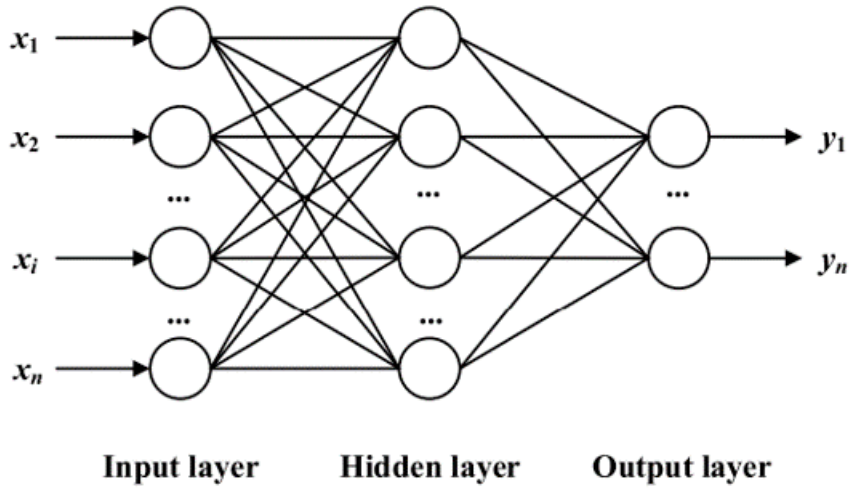


Figure 2.8: Example of an artificial neural network (Gao and Su, 2020).

Two main processes happen in a ANN, the feedforward process, and the backpropagation.

The feedforward process involves the flow of information from the input layer to the output layer, while the backpropagation occur from the output to the input layer. This last process is responsible for the learning process, at the same time that corrects the network weights.

The feedforward process follows the maths represented above.

$$\underset{Input}{\begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}} \times \underset{Weights}{\begin{bmatrix} w11, w12, ..., w1i, ..., w1n \\ w21, w22, ..., w2i, ..., w2n \\ ..... \\ wj1, wj2, ..., wji, ..., wjn \end{bmatrix}} + \underset{Bias}{\begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_j \end{bmatrix}} = \begin{bmatrix} x_1 \times w_1 1 + x_2 \times w_1 2 + ... + x_n * w_j 1 + b_1 \\ x_1 \times w_1 2 + x_2 \times w_2 2 + ... + x_n * w_j 2 + b_2 \\ ..... \\ x_1 \times w_1 n + x_2 \times w_2 n + ... + x_n * w_j n + b_j \end{bmatrix}$$

That said, for each j neurons, in the hidden layer (equation 2.8):

$$O_j = \sum_{i=1}^{n} w_j i x_i + b_j, \quad \forall\ j \tag{2.8}$$

Where b_j is the bias (a number added responsible to help and speed the learning process), $w_i j$ the link weights (from the input node i to hidden layer node j), and the x_i represents the input data. Bias works as an addition, allowing the output (that comes from the activation function) to move its values to the left or right, in the axis of abscissas (Heaton, 2012).

The most simple ANN is the MLP in which a neuron in a single layer receives input and transforms it into output as figure 2.9 shows.

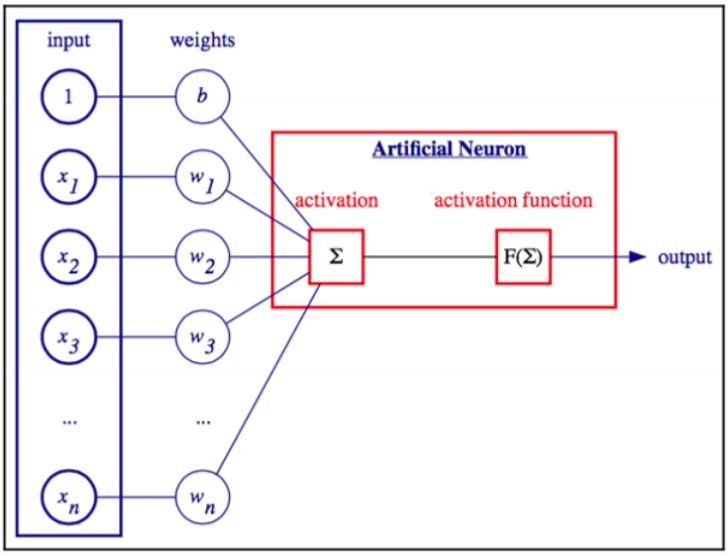


Figure 2.9: Schematic representation of a ANN neuron operation (Vasilev, Slater, Spacagna, Roelants, and Zocca, 2019).

From each neuron results an output that works as input for the next layer neurons. Thus, after the process before described (equation 2.8), for each neuron, is applied an activation function. From here results the final neuron output. This function application follows the following equation, 2.9:

$$y_j = f(O_j) = f\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right) \tag{2.9}$$

There are many known activation functions. Among the most popular are three highlighted in this research: the sigmoid, the Hyperbolic Tangent (Tanh) and the Rectified Linear Unit (RELU), next presented in tables 2.6, 2.7, and 2.8, respectively (Vasilev, Slater, Spacagna, Roelants, and Zocca, 2019):

Table 2.6: Logistic function (or sigmoid).

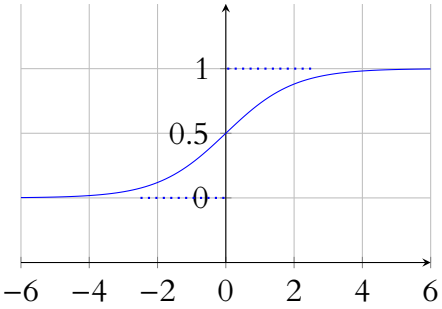
Logistic function (or sigmoid)
$f(a) = \frac{1}{1+exp^{-a}}$ Its output is between 0 and 1 

Table 2.7: Tanh function.

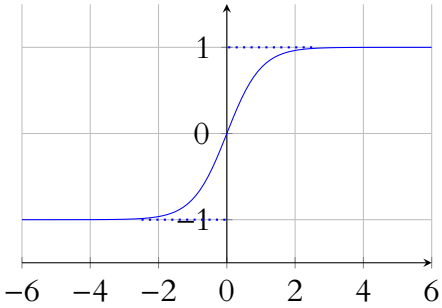
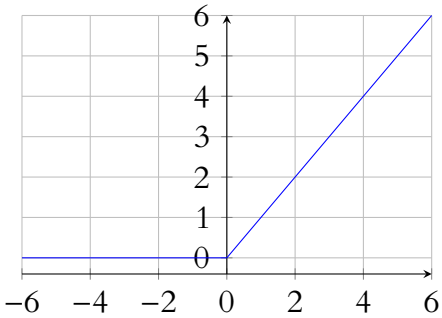
Tanh function
$f(a) = \frac{1-exp(-2a)}{1+exp(-2a)}$ Its output is between -1 and 1 

Table 2.8: Rectified Linear Unit function.

Rectified Linear Unit function
$f(a) = \begin{cases} a \; if \; a \geq 0 \\ 0 \; if \; a < 0 \end{cases}$ 

The weight matrix is the only parameter that varies during the ANN training process. As the weights are corrected, the neural network efficiency improves, converging to the smallest possible error. These update process is the backpropagation, in which the information flows in the opposite feedforward flow (output layer -> hidden layer(s) -> input layer).

In ANN models, the only parameter that varies among the training process is the weights matrix. The neural network performance improves as the weights are corrected, converging to the smallest possible error. Thus, come out the backpropagation concept. The output flows in the opposite way when compared with the feedforward. To correct the weights (output layer -> hidden layer(s) -> input layer). This process occurs through several steps, following mentioned (Heaton, 2012).

1. From the feedforward process, an output value results (a). To understand how well the ANN works, and how far these output results are closer to the expected ones, is used a Loss function (C_o). There are several loss functions, among them:

   a) **Absolute error** Is the most simple metric. Is the difference between the predicted value (p) and the actual value (a) (equation 2.10).

   $$E = p - a \tag{2.10}$$

   b) **Mean Absolute Error (MAE)**

   This metric results from the absolute error, squared and sum, and then divided by the total number of cases (equation 2.11). This error, since it is squared, deletes negative error numbers.

   $$MAE = \frac{1}{n}\sum_{i=1}^{m}(p - a)^2 \tag{2.11}$$

   c) **Root Mean Square Error (RMSE)**

   The RMSE method is identical to the Mean Absolute Error (MSE), except for the reason that the square root is added to the sum (equation 2.12).

   $$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{m}(p - a)^2} \tag{2.12}$$

2. The second key concept in ANN is the gradient descent. As previously mentioned, train ANN consist of a search for the best set of weights that generates the outputs with the lowest error. As a first approach, the proposal of finding the best set of weights, experimenting all the possible combinations comes up. This is, test the model with all the possible weights and find which returns a small error. Obviously, it turns out an disapproved possibility once it is almost impossible, because takes a lot of time and almost infinity computer capacity (Heaton, 2012). So, the train process use iterations to find the best set of weights, called epochs.

Here, the only set of weights that are known are the current ones of each iteration. That's not possible to find an error curve to all the weights but, it is feasible to know the instantaneous slope of the error function at a specific weight, which means, the derivative (Heaton, 2012).

It is essential understand how to calculate the partial derivative (concerning the weight) for the output of each neuron. In this process, the chain rule is applied. The chain rule deals with composite functions, which happens when a function is used as input of another function. To calculate the derivative across the ANN this technique is used since a neural network is, also a complex composite function.

But how to calculate the gradient? The gradient is the slope of a straight line, that is, the tangent to the error function. Using notation: $y = mx + b$, the gradient is represented by the m, and it is the significant part of this process.

To understand the gradient descent and the chain rule it is imperative to give special attention to figure 2.10 and the equations below (equation 2.13, 2.14, 2.15). In the figure it is represented a link weight (w) connecting two neurons, between the layer $L-1$ and L.

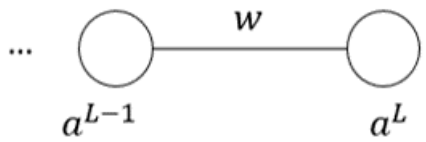


Figure 2.10: Link between two neurons, layer $L-1$ and L.

Representing the cost function as C_o, the equations above allow us to understand better how the chain rule works.

$$Z^{(L)} = (w^{(L)}.a^{(L-1)} + b^{(L)}) \tag{2.13}$$

$$a^{(L)} = f(w^{(L)}.a^{(L-1)} + b^{(L)}) \tag{2.14}$$

$$a^{(L)} = f(Z^{(L)}) \tag{2.15}$$

Be of importance to understand how sensitive the cost function is for small changes in the weight, W^L. For this, it is computed the $\frac{\partial C_o}{\partial w^L}$, this is, the gradient. Based on the chain rule technique, this gradient is calculated based on the following equation, 2.16:

$$\frac{\partial C_o}{\partial w^L} = \frac{\partial Z^L}{\partial w^L} \times \frac{\partial a^L}{\partial Z^L} \times \frac{\partial C_o}{\partial a^L} \tag{2.16}$$

3. Now that all the values are calculated, it is viable to apply the backpropagation method. This method is used to train the neural network and use the gradient values to update the weights. As the gradient values go down (the error varies less with the variation in the weights), the neural network error goes down, also (Heaton, 2012).

   In this process, there are two crucial concepts: the batch and the epochs. Each batch is a set of training elements that are used to train the model. Here, the weights are not updated for every single training set element. Instead, the gradients for each training set element are summed. Once every training set element has been used, the neural network weights may be updated. At this point, the iteration is considered complete. Here enter the concept of epoch, which are the iterations responsible for updating weights in the neural network (Heaton, 2012).

   The main goal is to minimize the loss function and, to do so, it is necessary a method to update the internal weights of neural networks, responsible for this reduction. Using the gradient descent to train the model the weights are updated through the following equation (2.17):

$$\triangle W_{(t)} = \triangle W_{(t-1)} + \alpha \frac{\partial C_o}{\partial W} \tag{2.17}$$

   Where α represents the learning rate. This rate is responsible for scale the gradient and slow down or speed up the learning process (Heaton, 2012).

2.2.2.4 Recurrent Neural Networks

Traditional neural networks can not use previous information to help understand the later one. Under those circumstances, when an input arrives in a neuron, an activation function is applied, giving rise to an output. Then, this output will be transmitted to the next layer neuron, without storing any information in the current neuron. Recurrent Neural Networks (RNN) have a particularity, using loops, they can keep that information. Further, in this type of neural networks sequential data of different length may be processed. Here, instead of generating output and just transmitting it to the next neuron, the RNN neuron will transmit this information to the neuron of the next layer, but also to itself again. In this way, the neuron is able to store that information, as figure 2.11 shows.

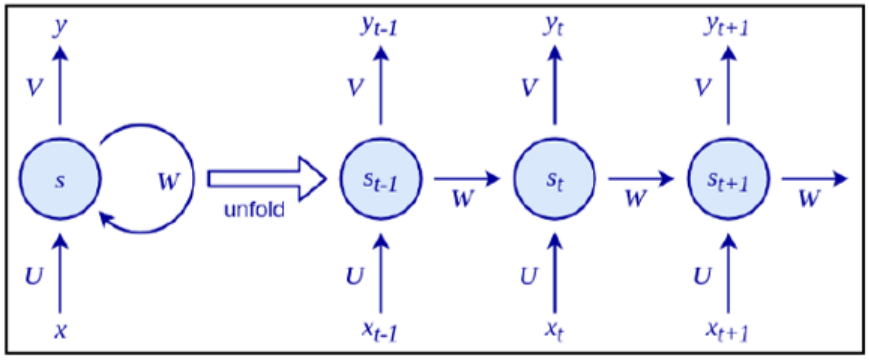


Figure 2.11: RNN diagram (Vasilev, Slater, Spacagna, Roelants, and Zocca, 2019).

In the figure above, the x is the neuron input, whereas the y represents the output. There is well represented how an RNN works. The input x reaches the neuron and result and output (y) from that. This output is transmitted to the next neuron but also re-transmitted to the current neuron again. Thereby, this neuron can store this information, and, in the future, will use this to create the next output. Considering that s_t is the actual state, to create the y_t will use the information store in the neuron resulting from the s_{t-1} output (y_{t-1}), and so on, as shown in the figure.

There are three weights responsible in this process: the U, which transform the input to the current state s_t, the W, that transform the previous state (s_{t-1}) to the current one (s_t) and the V responsible to map the s_t to the output, y_t.

The process takes place following the subsequent equations, in which f represents the activation function (Vasilev, Slater, Spacagna, Roelants, and Zocca, 2019):

$$s_t = f(s_{t-1} * W, x_t * U) \tag{2.18}$$

$$y_t = s_t * V \tag{2.19}$$

RNN are used for various purposes, one of which is the time series. When compared with the regression and classification problems, the time series add the temporal dependency between observations (Brownlee, 2018). Besides that, it is important to refer that RNN models can give good answers when there is a small gap between the relevant information and what is needed at the moment. On the contrary, when the gap between the relevant information and the moment when it is needed is too wide, the RNN becomes an inefficient model (Yoshua Bengio, Patrice Simard, and Paolo Frasconi, 1994). To fix this, in 1997, Hochreiter and Schmidhuber introduced the LSTM model, capable to handle "Long-term dependencies" (Hochreiter and Schmidhuber, 1997; Oliveira et al., 2020). A LSTM model operates as figure 2.12 shows.

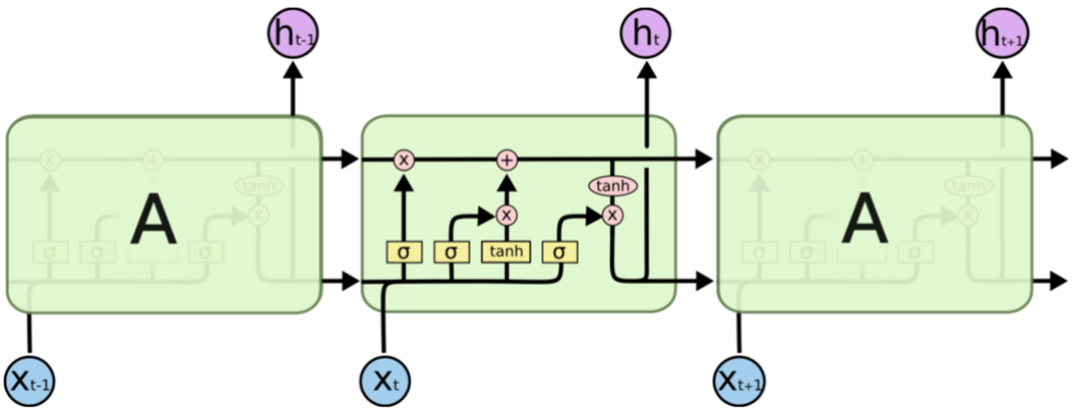


Figure 2.12: LSTM model operating process (Olah, 2015).

The LSTM cell represented in the figure above shows a horizontal line on the top of the diagram. This line is responsible for some small transformations and allows the information to be transported through

the cell.

In LSTM cells, there are structures called gates. These are the structures that allow remove or add information to the cell. In an LSTM cell, there are three gates, as represented in the figure. These gates use a sigmoid function, and consequently, from its resulting values between 0 and 1. A value equals to 1 represents the information that can pass and it is important, on the other hand, 0 represents the information that cannot be stored at all (because it is not useful). Hereupon, a LSTM cell works following four fundamental steps (Vasilev, Slater, Spacagna, Roelants, and Zocca, 2019):

1. The first is the first cell gate, a forget gate. Here, is determined which information keeps in the cell and the one that is eliminated, following the equation 2.20:

$$f_t = \sigma(W_t * x_t + U_t * h_t - 1) \tag{2.20}$$

2. The next step regards another cell gate, the input gate. First, it is used $h_t - 1$ and x as inputs, and then a sigmoid is applied, creating, also, a value between 0 and 1. Here, 0 represents that no information is added to that cell block's memory. This process follows the equation 2.21:

$$i_t = \sigma(W_t * x_t + U_t * h_t - 1) \tag{2.21}$$

   After this, using h_{t-1} (previous output) and the x_t, the input of this state, is applied a Tanh function, creating a candidate input to be added, c'_t, as the following equation, 2.22, shows:

$$c'_t = \sigma(W_c * x_t + U_c * h_t - 1) \tag{2.22}$$

3. Once the forget gates and input gates chose which new and old information should be included (creating the total amount of information considered important), they are related to the transformations that occur in the top horizontal line of the cell (equation 2.23).

$$c_t = f_t \otimes c_{t-1} \oplus i_t \otimes c'_t \tag{2.23}$$

4. Finally, the LSTM model needs to define what is the output. First using h_{t-1} and the x_t, a Tanh function is applied, in the last gate, the output gate, following the equation 2.24:

$$o_t = \sigma(w_o * x_t + U_o * h_{t-1}) \tag{2.24}$$

   After this, and as the final step to create the output, is used the c_t (the actual cell state) through a Tanh function, to give an output between -1 and 1, and multiply this to the output gate (equation 2.25), resulting:

$$h_t = o_t * tanh(c_t) \tag{2.25}$$

2.2.3 Evaluation Metrics

To evaluate the ML models' performance there are a several metrics that can be used, to understand the quality of the predictions generated.

2.2.3.1 Classification Accuracy

The accuracy is the simplest metric and commonly used in classification problems. This metric results from the number of correct predictions divided by the total number of examples, as the following equation shows (equation 2.26).

$$Accuracy = \frac{Number\ of\ correctpredictions}{total\ number\ of\ examples} \tag{2.26}$$

2.2.3.2 MSE

This function calculates the average of the squared model errors (equation 2.27). That is, smaller differences are less important, while larger differences are given more weight.

$$MSE = \frac{1}{n}\sum_{i=1}^{m}(\hat{y} - y)^2 \tag{2.27}$$

2.2.3.3 RMSE

There is a variation of MAE, which facilitates interpretation: the RMSE. It is simply the square root of the MAE (equation 2.28). In this case, the error returns to the original units of measure of the dependent variable.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{m}(\hat{y} - y)^2} \tag{2.28}$$

2.2.3.4 MAE

In this case, instead of assigning a weight according to the magnitude of the difference, it assigns the same weight to all differences, in a linear way (equation 2.27).

$$MAE = \frac{1}{n}\sum_{i=1}^{m}|\hat{y} - y|^2 \tag{2.29}$$

3. Materials and methods

3.1 Data Exploration

Data exploration is a crucial phase in data mining projects. Visualize and understand data is a fundamental step to allow its correct preparation and achieve the best results. Thence, this chapter presents the data sets that were used in this research, respecting a case study in the city of Guimarães.

3.1.1 Air quality dataset

In the first instance, the data collected relates to the UV index, CO and SO_2 air concentration, as well as some features concerning the weather, in the city of Guimarães. This data represents several important parameters to this research, especially to extract some conclusions in the air quality and atmosphere problems domain.

3.1.1.1 Ultraviolet Index

The first dataset concerns the UV Index in the city of Guimarães. Table 3.1 depicts the dataset features.

Table 3.1: Ultraviolet Index dataset constitution.

Feature	Description	Data type
UV Value	The UV index.	Number(Double)
Date	Register time stamp.	Local date time

This dataset has in its constitution 5776 rows and a time range defined from 24-07-2018 (14:00:00) to 23-03-2020 (14:00:00).

On average, this dataset presents by day, 12 registers (every two hours). However, some days don't exhibit any observation, due to a registration failure, a total of 102 missing timesteps:

- From 13-12-2018 to 14-01-2019
- From 07-03-2019 to 28-03-2019
- From 27-07-2019 to 07-08-2019
- From 20-11-2019 to 05-12-2019

- From 04-04-2019 to 09-04-2019
- From 30-01-2020 to 10-02-2020
- Day: 02-05-2019

Besides, do a statistical analysis of the dataset allow a deeper data exploration. Thus, table 3.2 shows statistics measures, relating to the UV index dataset. Metrics such as minimum, maximum, mean, standard deviation, and variance are displayed, granting a better data understand.

Table 3.2: Ultraviolet index dataset statistical analysis.

Metrics	UV index
Minimum	1.110
Maximum	10.290
Mean	5.077
Standard Deviation	2.775
Variance	7.700

In the following figures, 3.1 and 3.2 , are represented the UV Index mean by year and month, in the dataset time range.

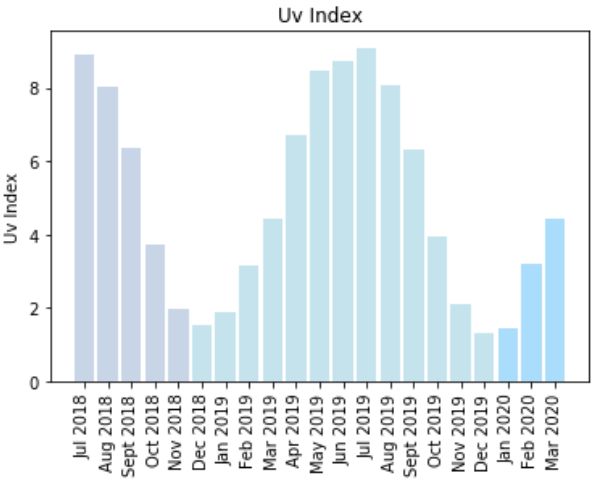


Figure 3.1: Ultraviolet index mean by month and year.

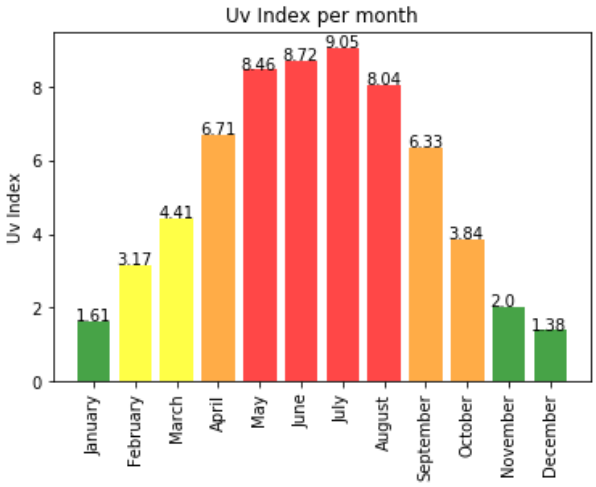


Figure 3.2: Ultraviolet index mean by month (2018-2020)

Analyzing the graphs above it is possible to realize that the months with lower values of the UV index are November, December, and January, reaching the higher values between April and August. For this reason, it is perceptible that the UV Index varies according to the month, showing very similar values for the same months, in different years.

After explore the dataset it was noticed that UV Index does not presents a significant variation in a day. Figure 3.3 shows the variation of the UV Index in a day, displaying a random sample of days.

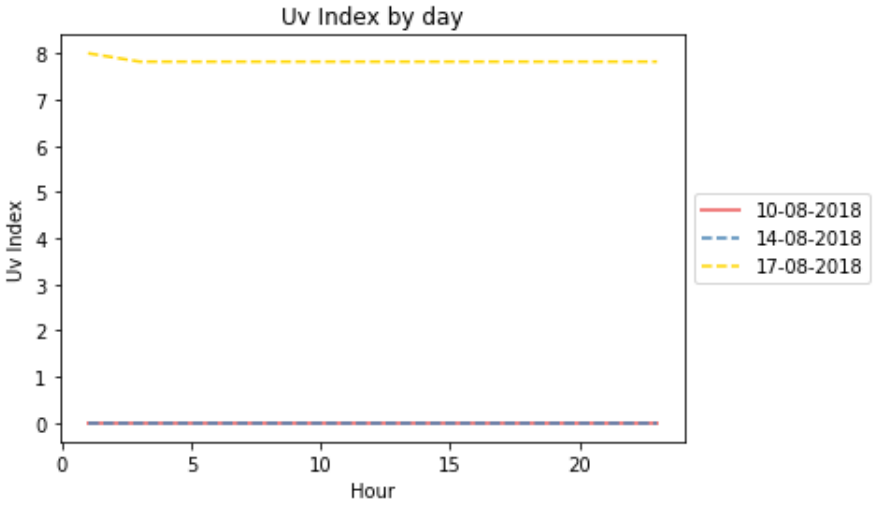


Figure 3.3: Example of Ultraviolet index variation by day.

3.1.1.2 Atmospheric Pollutants (CO and SO_2)

This dataset regards the atmospheric pollutants, in particular CO and SO_2, also in the city of Guimarães. In table 3.3 are represented the dataset features.

Table 3.3: Atmospheric pollutants dataset constitution.

Feature	Description	Data type
CO Value	Carbon Monoxide concentration, in ppm.	Number (double)
SO_2 Value	Sulfur Dioxide concentration, in ppm.	Number (double)
Date	Register time stamp.	Local date time

This dataset reveal 5776 rows and a time range from 24-07-2018 (14:00:00) to 23-03-2020 (14:00:00). Further, it has the same time registers as the UV Index dataset, once were recorded at the same time intervals.

Table 3.4 shows the statistical information about the the features that constitute this dataset.

Table 3.4: Atmospheric Pollutants dataset statistical analysis.

Metrics	CO	SO_2
Minimum	2.70×10^{-6}	1.36×10^{-9}
Maximum	6.25×10^{-6}	1.07×10^{-6}
Mean	4.23×10^{-6}	-4.06×10^{-11}
Standard Deviation	8.12×10^{-7}	4.43×10^{-10}
Variance	6.59×10^{-13}	1.97×10^{-19}

The graph from figure 3.4 depicts the CO evolution over time, whereas the graph from figure 3.5 represents the mean “CO value” by month.

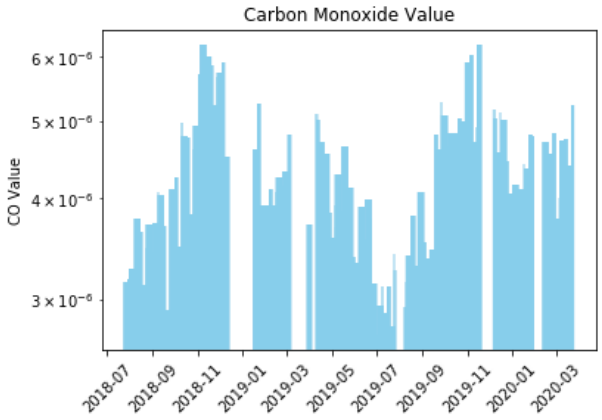


Figure 3.4: CO air concentration over time.

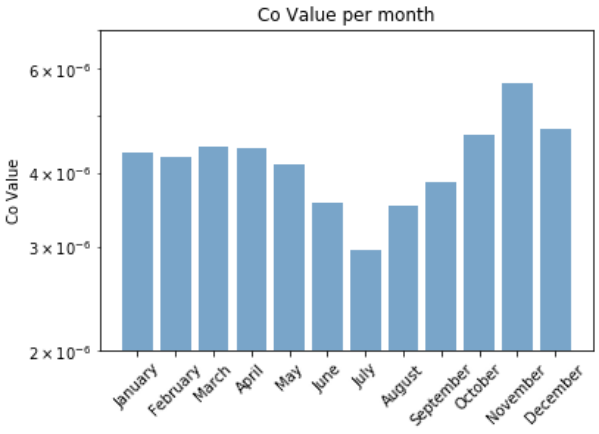


Figure 3.5: Mean monthly CO concentration (2018-2020).

When evaluated, the “CO value” evolution over time shows some blank spaces. It is possible to deduce that these represent the previously mentioned missing timesteps. Both figures, 3.4 and 3.5, allow assumes that CO values are higher in November and December, and reach the lowest values in July.

As the UV index, the CO air concentration does not show a significantly variation in a day, figure 3.6. This graph shows a sample of days, to display this insignificant variation.

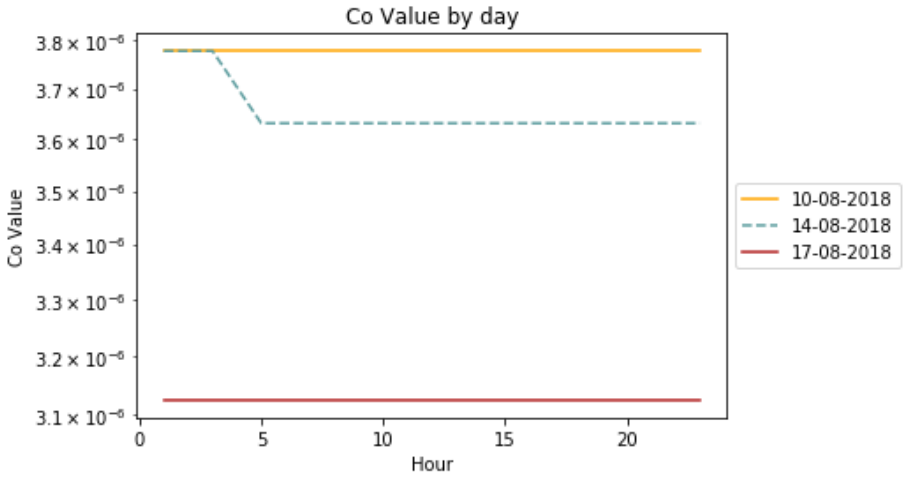


Figure 3.6: Example of Carbon Monoxide air concentration variation by day.

The graph from 3.7 shows the evolution of SO_2 concentration over time.

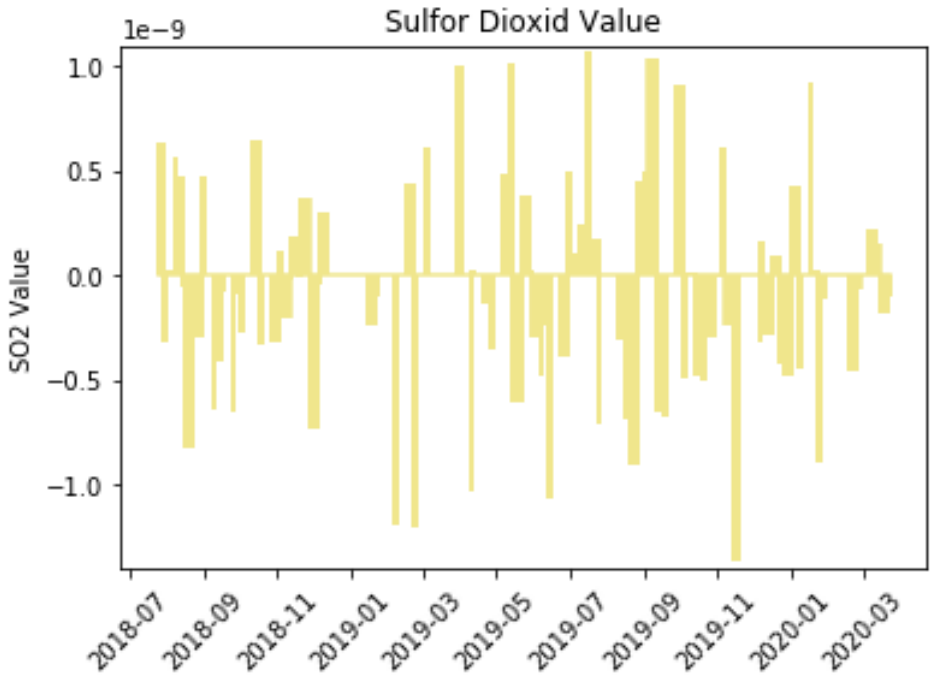


Figure 3.7: Sulfur Dioxide air concentration over time.

As the graph shows, SO_2 concentration reaches negative values, a fact that does not exist from a scientific point of view. These negative values may occur, possibly, due to the sensor exposed to nitrogen dioxide, which causes a negative interference in this sensor. Once there are no registers in the dataset of nitrogen dioxide, the negative values are considered important, since, even negative, it represents the real interference of another pollutant.

3.1.1.3 Weather Indicators

This dataset includes data related to the weather conditions in the city of Guimarães. Table 3.5 represent some dataset characteristics.

Table 3.5: Weather dataset constitution.

Feature	Description	Data type
Weather description	Weather description at a given day and hour.	String
Temperature	Temperature recorded at a given date and hour, in degrees Celsius (°C).	Number (integer)
Atmospheric pressure	Atmospheric pressure at a given day and hour, in millibars.	Number (integer)
Humidity	Percentage of humidity at a given day and hour.	Number (integer)
Clouds	Percentage of clouds at a given day and hour.	Number (integer)
Precipitation	Precipitation level at a given day and hour.	Number (integer)
Date	Register time stamp.	Local Date Time

The dataset has 11817 rows in its constitution in a time range defined from 24-07-2018 (14:00:00) to 23-03-2020 (14:00:00).

Table 3.6 shows the dataset statistical metrics.

Table 3.6: Weather dataset statistical analysis.

Metrics	Temperature	Atm pressure	Humidity	Clouds	Precipitation
Minimum	0	985	14	0	0
Maximum	35	1035	100	100	2
Mean	14.80	1018.31	81.58	35.42	5.9×10^{-4}
Standard Deviation	5.32	6.50	17.18	34.85	0.03
Variance	28.27	42.26	294.99	1214.47	9.31×10^{-4}

This dataset present more observations per day when compared with the two other mentioned before. On average, has 24 records in a day, which indicates an hourly record (with exceptions). Just like the others, this dataset has some days that do not present any observation (a total of 102 days). The missing timesteps are:

- From 13-12-2018 to 14-01-2019
- From 06-03-2019 to 27-03-2019
- From 04-04-2019 to 09-04-2019
- Day: 02-05-2019
- From 27-07-2019 to 07-08-2019
- From 20-11-2019 to 05-12-2019
- From 30-01-2020 to 10-02-2020

Then, an overview of each of this dataset's constituent features is shown in the following items.

- **Weather description**

  Represented as a string, the "Weather description" feature shows thirty-one different values among which "trovoada", "nuvens dispersas"and "aguaceiros".

  There are some unfamiliar characters at the "Weather description" values. This may constitute an error, requiring special attention in the data preparation process.

- **Temperature**

One of the features from this dataset is the registered temperature. Its variation over time and the monthly mean are depict in the following figures, 3.8 and 3.9, respectively.

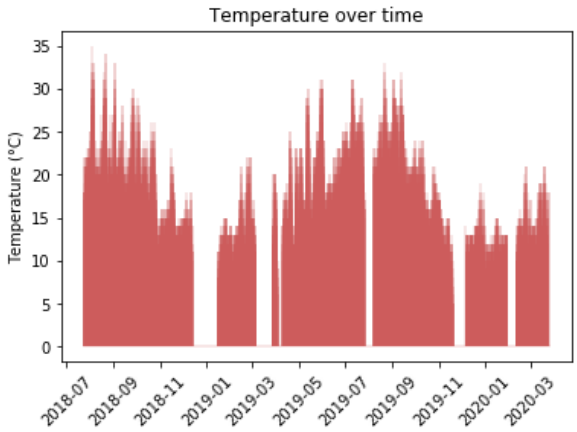


Figure 3.8: Temperature over time, in °C.

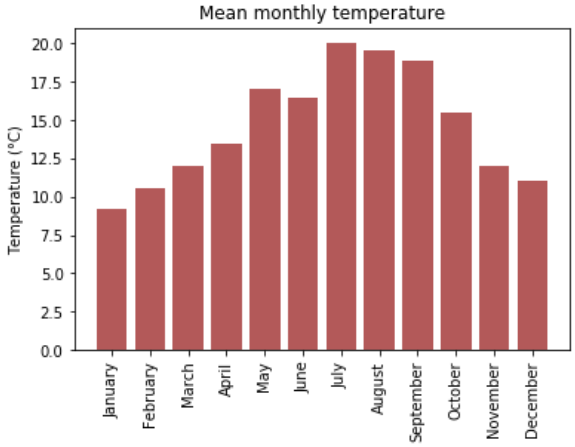


Figure 3.9: Mean monthly temperature, in °C (2018-2020).

After evaluate the graphs from the figures above, it is possible to apprehend that the higher temperature values are recorded from July to October. On the other hand, the smallest value is verified in January. Figure 3.8 shows some blank spaces due to the missing dates steps referred.

- **Atmospheric pressure**

The third feature is the “Atmospheric Pressure”. The graph from the figure 3.10 shows its variation over time.

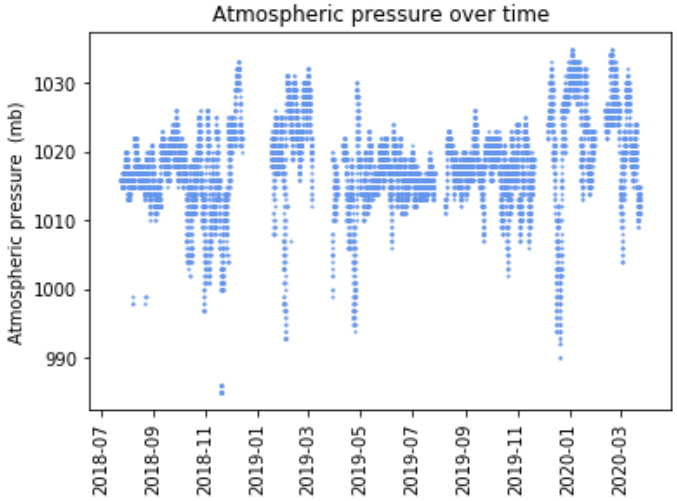


Figure 3.10: Atmosphere pressure over time, in millibar.

It is possible to realize that the atmospheric pressure does not vary significantly. Besides, does not present a particular behavior, that points to an upward or downward trend, according to the time of the year.

- **Humidity**

The following graphs, figure 3.11 and 3.12, present the humidity variation, considering the year and the month.

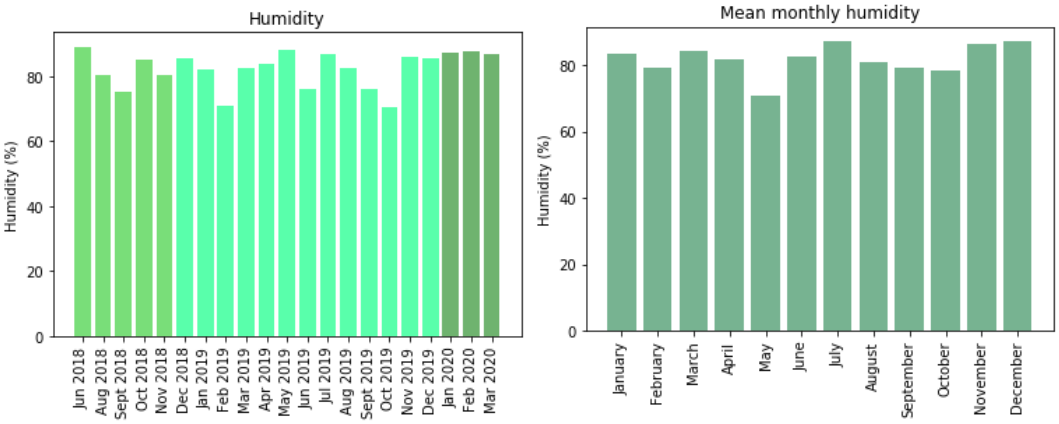


Figure 3.11: Humidity (%) by month and year. Figure 3.12: Mean monthly humidity (%) (2018-2020).

From the graphs above, it is possible to conclude that the months with the highest percentage are July, November, and December. The smallest humidity percentage registered is in May, regarding the data available. However, the humidity percentage does not presents significant different values between months.

- **Clouds**

The “Clouds” behavior is delineated in the figure 3.13 and 3.14.

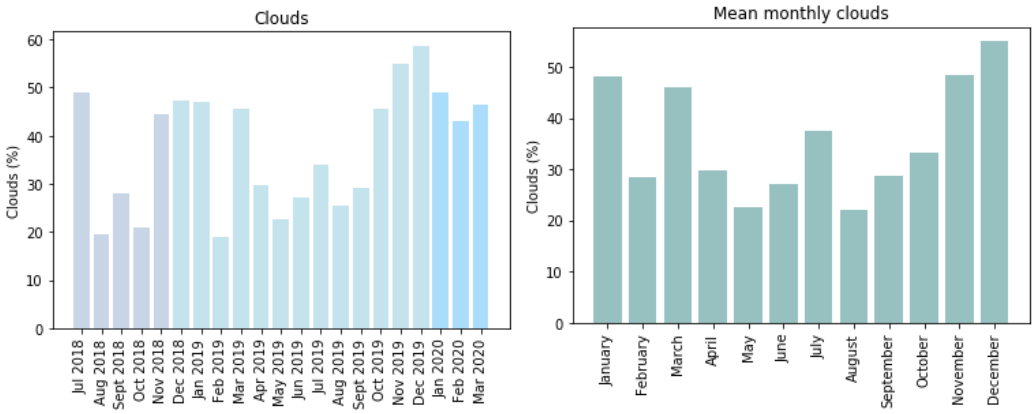


Figure 3.13: Clouds (%) mean by month and year. Figure 3.14: Mean monthly clouds (%) (2018-2020).

As the figures show, November, December, January and March are the months with higher clouds percentage registered.

- **Precipitation**

  In a total of 11817 records, the "Precipitation" feature presents 11817 records equals to zero. The remaining five observations are presenting in the table 3.7.

Table 3.7: Precipitation values.

Precipitation	Date
2	2020-01-17 00:00:00
2	2020-01-17 01:00:00
1	2020-01-17 02:00:00
1	2020-01-17 03:00:00
1	2020-03-19 16:00:00

After an evaluation of this feature, it was concluded that it has inconclusive values. Thence, it might be removed in the future.

3.1.1.4 Missing values

The blank spaces aforementioned do not constitute missing values, but missing timesteps. That said, it is of relevant importance to mention that, in the three datasets, missing values were not found.

3.1.1.5 Correlation between features

After analyzing all the three datasets were investigated the correlation between its features, putting them all together. This analysis made use of a correlation matrix, showing the Spearman rank correlation. This correlation measure was used once the "UV value" (target feature) has some ordinal nature in it, once can be converted into levels (as mentioned before). Further, this correlation does not take into account the data distribution.

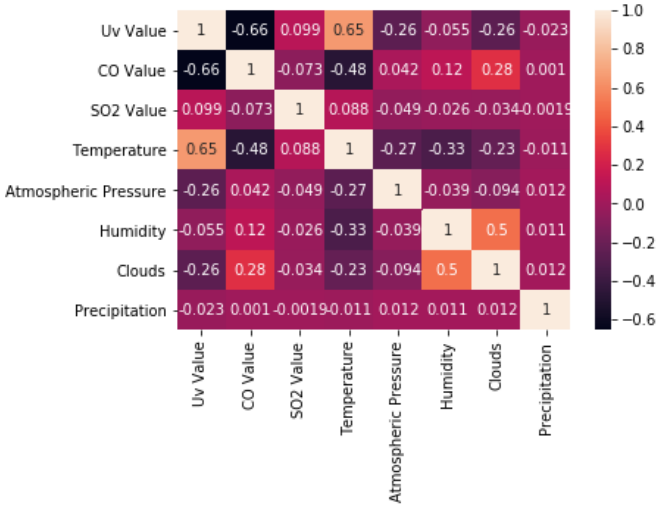


Figure 3.15: Correlation matrix concerning the air quality dataset.

The matrix shows that there are not a significant correlation between the features that are not used as a target. Further, the only considerable correlation showed is a positive correlation between the "UV Value" and the "Temperature", as well as a negative correlation between "CO Value" and the "UV Value" features. This may indicate that these features must be used to train the model.

3.1.2 Water quality dataset

The second dataset explored in this research concerns the water quality. This dataset regards a WWTP located, also, in Guimarães. It presents three distinct datasets, according to the WWTP unit, from where results information about relevant parameters.

Figure 3.16 depicts a schema about the data available in the WWTP under study. This schema allows a better understanding of the available features.

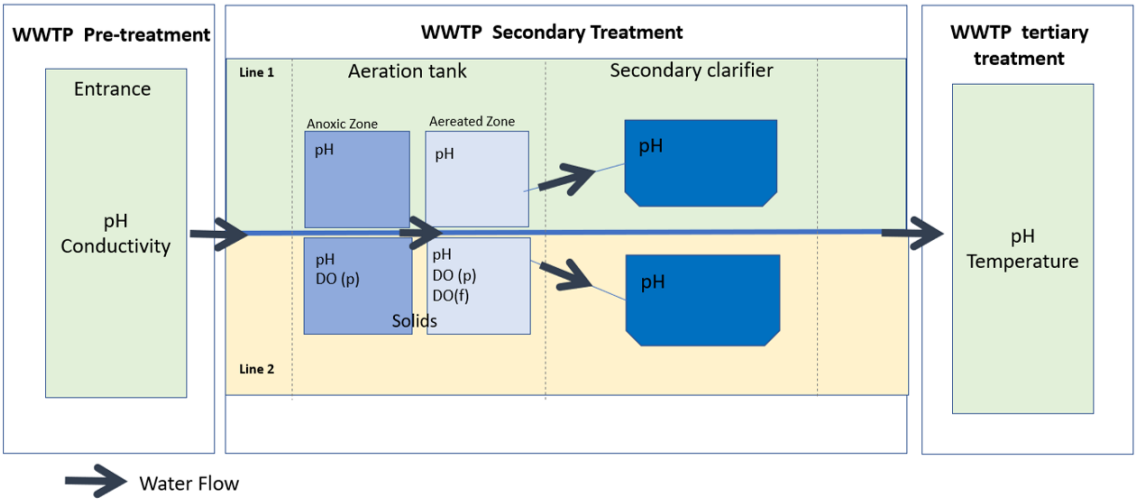


Figure 3.16: Schema showing the available features in the dataset concerning a WWTP.

The data set presents fourteen features, among three units, as the figure above shows. The pre-treatment unit is the first one, representing the wastewater at the moment that arrives in the WWTP. These features concern the water that comes from the domestic, industrial, agriculture, among other sources, without any treatment yet. From this WWTP unit, there are two features available in the dataset. The secondary treatment is divided into two sub-units, the aeration tank (that has an anoxic zone and an aerated Zone) and the secondary clarifier. Each one of these sub-units (the aeration thank and the secondary clarifier) is, further, divided into two lines (line 1 and line 2). From the secondary treatment unit, the dataset present has a total of 10 features. The last unit is the tertiary treatment, which represents the last phase. Here it is presented the water before being returned to water natural sources, showing two available features. Even being a single dataset, it will be analyzed by unit to facilitate the data exploration process.

The first step is to analyze the "Date" feature, separately, once is common to all the units. This feature is the register time stamp, using a "Local date time" format. In general, the dataset presents a total of 2379 rows and presents a time range from 2016-01-01 (08:00:00) to 2020-05-28 (08:00:00). Analyzing

the dataset, it is possible to conclude that there are daily registers, at 08:00:00 and 16:00:00, with some exceptions. Despite this, there are days without any observation (as table A.1 from appendix A shows). Further, from the beginning of 2018, the registers are not so frequently, with a single registration by day (on most days, also with some exceptions).

3.1.2.1 Pre-treatment

The first unit to be explored is the pre-treatment unit. The table 3.8 shows the features available, and the respective designation, for this unit.

Table 3.8: Pre-treatment data set features.

Feature	Description	Data type
pH	pH value, measured with a fixed meter.	Number (double)
Conductivity	Conductivity value, measured in microsiemens (μS).	Number (double)

Missing values

This dataset presents several missing values, as table 3.9 shows.

Table 3.9: Pre-treatment - Missing values

Feature	Missing Values
pH	729
Conductivity	730

Numbers considered not meaningful

From a chemical point of view, the "pH" presents three numbers that do not make sense, once pH takes values between 0 and 14 only. The registers containing these values are thus listed in table 3.10 and are likely to result from incorrect readings/insertion errors.

Table 3.10: Non meaningful pre-treatment pH observations.

Date	pH
2017-02-14 16:00:00	5015.0
2016-03-03 08:00	69.47
2017-10-04 16:00	20.0

After removing these observations, since they difficult the analysis, the statistics from this data set were computed, as table 3.11 shows.

Table 3.11: Pre-treatment features statistic analyzes.

Metrics	pH	Conductivity
Minimum	0.0	0.0
Maximum	9.43	6392.0
Mean	4.430	441.351
Standard Deviation	2.373	839.079
Variance	5.633	704054.1

The minimum values, for both features, are zero. Analyzing the high numbers of standard deviation and variance for the "Conductivity" feature, it is possible to conclude several items: the data points are not close to the mean values and the feature presents a high range of values, as well as a big spread between them. Besides, it is possible to mention that these features present a high number of observations equal to zero. Consequently, it results in a very high spread between the values, higher in the "Conductivity" feature, once it presents higher values.

Figures 3.17 and 3.18 allow perceive the "pH" and "Conductivity" behavior, as well as a visual perception of the missing data.

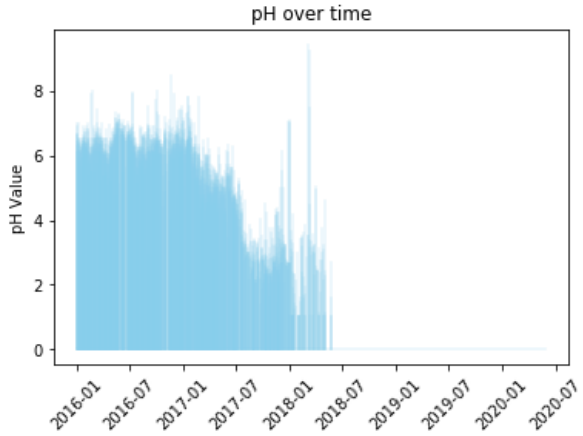


Figure 3.17: Pre-treatment wastewater pH over time.

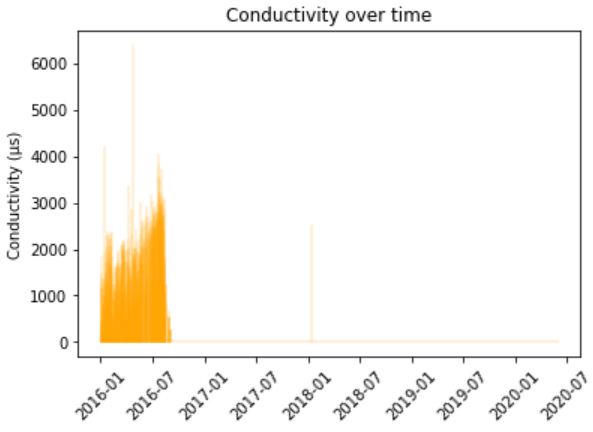


Figure 3.18: Pre-treatment conductivity over time.

After analyzing the figure 3.18, it is possible to understand that there are only data available until 2018-05-25. After this date, until day 23-09-2018, there are only observations equal to zero. After that, all are missing values. This interval leads to the conclusion that the values equal to zero may indicate values not recorded, and, instead of that, replaced by 0.

The figure 3.17 shows a similar circumstances. There are only meaningful data available until 2016-09-06. After this, until 2018-09-23, there are only values equal to zero (except 2018-01-12). Hereafter all the registrations are missing values.

3.1.2.2 Secondary Treatment

This data set is related to the secondary treatment phase. Here there are two units available, the aeration thank and the secondary clarifier. Also, there are two lines through which wastewater crosses. The table 3.12 shows the features of this data set.

Table 3.12: Secondary treatment data set features.

Line 1		
Feature	**Description**	**Data type**
pH-Anoxic Zone	pH value, measured with a fixed meter, in the aeration thank anoxic zone (line 1).	Number (double)
pH-Aerated Zone	pH value, measured with a fixed meter, in the aeration thank aerated zone (line 1).	Number (double)
pH-Secondary clarifier	pH value, measured with a fixed meter, in the secondary clarifier (line 1).	Number (double)
Line 2		
Feature	**Description**	**Data type**
pH-Anoxic Zone	pH value, measured with a fixed meter, in the aeration thank anoxic zone (Line 2).	Number (double)
DO-Anoxic Zone (p)	Dissolved oxygen (mg/L) measured with a portable meter (p), in the aeration thank anoxic zone (Line 2).	Number (double)
pH-Aerated Zone	pH value, measured with a fixed meter, in the aeration thank aerated zone (line 2).	Number (double)
DO-Aerated Zone (p)	Dissolved oxygen (mg/L) measured with a portable meter (p), in the aeration thank aerated zone (Line 2).	Number (double)
DO-Aerated Zone (f)	Dissolved oxygen (mg/L) measured with a fixed meter (f), in the aeration thank aerated zone (Line 2).	Number (double)
pH-Secondary clarifier	pH value, measured with a fixed meter, in the secondary clarifier (Line 2).	Number (double)
Solids	Amount of solids (mg/L) that result from the aeration thank , in the aeration thank (Line 2).	Number (double)

Line 1

For a more meticulous data exploration, the lines from the secondary treatment will be analyzed separately, starting with the line 1.

Missing values

During the exploration process, from the line 1 data set, were found a total of 1149 missing values in a total of 7137 records. Table 3.13 shows the number of missing values by feature, for line 1.

Table 3.13: Secondary treatment (Line 1) - Missing values.

Feature	Missing Values
pH- Anoxic Zone	455
pH- Aerated Zone	523
pH- Secondary clarifier	171

Table 3.14 indicates the statistical parameters of this dataset.

Table 3.14: Secondary treatment (Line 1) features statistical analysis.

Metrics	pH- Anoxic Zone	pH- Aerated Zone	pH- Secondary clarifier
Minimum	1.0	1.0	1.0
Maximum	11.0	11.0	9.95
Mean	2.712	1.650	7.0298
Standard Deviation	2.870	1.924	1.290
Variance	8.238	3.702	1.664

From the table above arise several conclusions. Concerning the standard deviation and the variance present values that may indicate that: the data, for each feature, are not spread out over a large range of values and that these values tend to be reasonably close to its mean value. This conclusions may be supported by the graphs in the figure 3.19, 3.20, and 3.21, showing the features behavior over time.

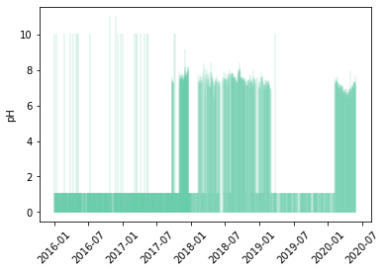


Figure 3.19: pH-Anoxic Zone.

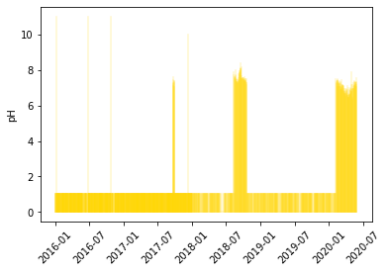


Figure 3.20: pH-Aerated Zone (p).

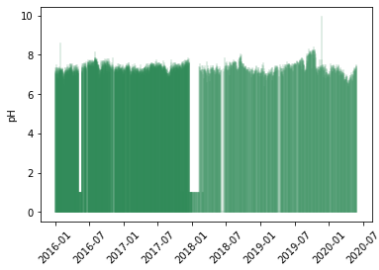


Figure 3.21: pH- Sec. clarifier (p).

Graph 3.19 shows that the pH from the anoxic zone presents most values equals to 1. It is estimated that these values represent the default of registration once exist in large portions and at well-defined intervals. Regarding this, there are three main time intervals to focus: 2016-01-01 to 2017-10-27, 2017-12-15 to 2018-02-07, and 2019-03-02 to 2020-02-06. Thus, this feature shows a few relevant information.

In addition, the graph 3.20 manifests, for pH at the aerated zone values, a similar data behavior (when compared with the pH from the anoxic zone). Here, there are even less meaningful values. The main lack of information is between 2016-01-01 and 2018-08-10 and between 2018-10-19 and 2020-02-06.

The last graph (figure 3.21) shows that the pH value in the secondary clarifier is, until this point, the most complete feature. It shows a small lack of meaningful information. Thus, the highlight intervals (the only ones with observations equal to 1) are 2016-05-04 to 2016-05-20 and 2017-11-14 to 2018-02-28.

Line 2

Line 2 has more available features than the previous one. However, present more missing values, as table 3.15 shows.

Table 3.15: Secondary treatment (Line 2) - Missing values.

Feature	Missing Values
pH- Anoxic Zone	174
DO-Anoxic Zone (p)	152
pH- Aerated Zone	174
DO- Aerated Zone (p)	153
DO- Aerated Zone (f)	153
pH- Secondary clarifier	170
Solids	732

Like the previous dataset, here were found some values considered wrong. Thus, table 3.16 shows this values, and consequently, the registers deleted from the dataset to facilitate the data analysis.

Table 3.16: Non meaningful secondary treatment (Line 2) registers.

Date	DO Anoxic Zone (p)	DO Aerated Zone (p)	DO Aerated Zone (f)
2018-08-29 08:00:00	72.4	-	-
2018-02-28 08:00:00	-	-	20.3
2018-02-17 08:00:00	-	-	19.6
2018-02-15 08:00:00	-	-	17.78
2018-02-03 16:00:00	-	-	20.2
2017-04-23 16:00:00	-	-	21.41
2017-04-16 16:00:00	-	19.4	-
2017-03-14 08:00:00	-	-	52.43
2017-02-15 16:00:00	-	232.0	-
2016-10-08 16:00:00	-	168.0	183.0
2016-03-09 16:00:00	-	-	240.0
2016-02-05 08:00:00	-	18.2	20.0

After delete the wrong registers, the statistics were computed. So, the table 3.17 shows the different metrics for each feature.

Table 3.17: Secondary treatment (Line 2) feature statistical analysis.

Metrics	pH An. Zone	DO An. Zone (p)	pH Aer. Zone	DO Aer. Zone (p)	DO Aer. Zone (f)	Solids
Minimum	1.0	0.0	1.0	0.0	0.0	0.0
Maximum	9.86	2.5	8.92	7.84	4.86	8101.0
Mean	6.869	0.059	6.864	1.8448	1.847	3018.267
Standard Deviation	1.535	0.101	1.529	0.649	0.650	2293.572
Variance	2.356	0.010	2.340	0.422	0.422	5260475.1777

It is possible to conclude that the “Solids” shows a high value of standard deviation and variance. It happens because this feature show a very high amount of registers equals to zero. So, there are a high range of values in the dataset, with high dispersion between them.

The below items provide a thorough exploration of the features available from the line 2 .

- **pH**

  The graphs in the figure 3.22, 3.23 and 3.24, shows the pH behaviour over time in the aerated zone, in the anoxic zone and in secondary clarifier, respectively (in line 2).

Figure 3.22: pH-Aerated Zone.

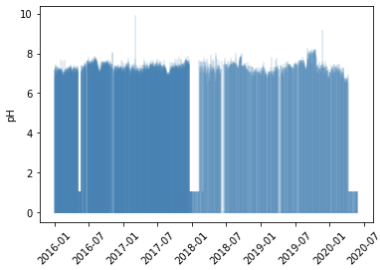


Figure 3.23: pH-Anoxic Zone.

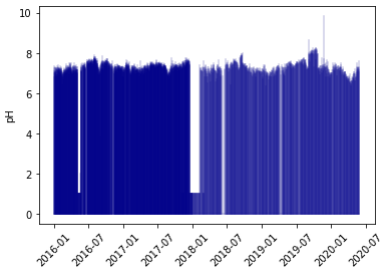


Figure 3.24: pH- Sec. clarifier.

After analyzing the graphs, it can be conclude that these features have a similar behavior as the pH values represented in the graph 3.21.

- **DO**

  The amount of DO in the water, over time, is represented in the graphs from the figure 3.25, 3.26, and 3.27.

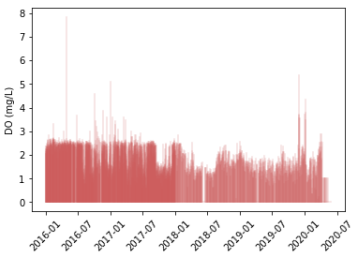


Figure 3.25: DO-Aerated Zone (p).

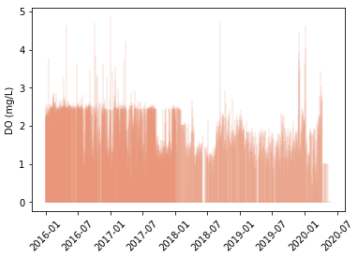


Figure 3.26: DO-Aerated Zone (f).

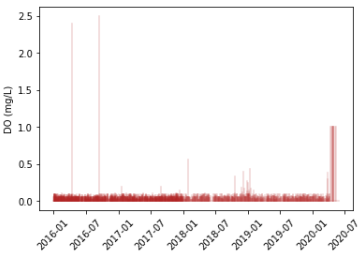


Figure 3.27: DO-Anoxic Zone (p).

The variation over time is very similar when DO is measured with a portable and a fixed meter (figure 3.25 and 3.26, respectively).

The aerated zone has higher values of DO when compared with the anoxic zone. This is expected once, as previously mentioned, the aerated zone is characterized by the presence of oxygen, while the anoxic zone reveals the contrary. These discrepancies of DO values (between zones) are due to the nitrification and denitrification processes. In the anoxic zone, there are some oxygen registered once, there are a small amount of oxygen "bound" in some molecules, such as nitrates and nitrates (due to denitrification process).

- **Solids**

The variation over time of solids presented in the wastewater, in the aeration thank, are presented in figure 3.28.

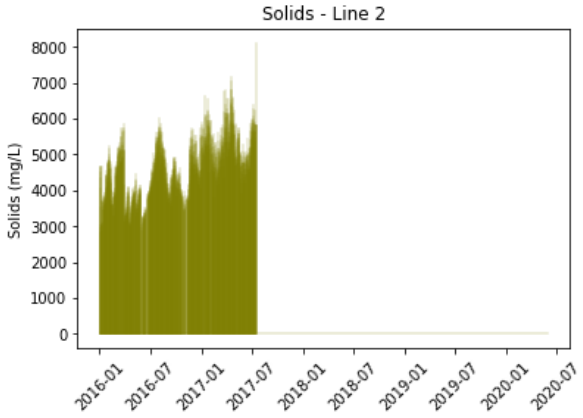


Figure 3.28: Amount of solids presented in the wastewater, in the aeration thank.

Analyzing the graph it is possible to conclude that the amount of solids in the aeration thank presents meaningful values until 2017-07-18. After that, until 23-09-2018, present values equal to zero. Henceforth only exhibit missing values. It is the main reason for the high variance recorded for this feature (table 3.17).

3.1.2.3 Tertiary treatment

The tertiary treatment requires special attention once depicts the last wastewater phase. For that reason, this phase determine the water condition before being returned to the water environment.

The data available concerns the pH and the temperature in this WWTP unit, represented in the table 3.18.

Table 3.18: Tertiary treatment data set constitution features.

Feature	Description	Data type
pH-Tertiary treatment	pH value in the tertiary treatment unit.	Number (double)
Temperature	Temperature value, measured in degrees Celsius (°C).	Local date time

After, the missing values were found as figure 3.19 shows.

Table 3.19: Tertiary treatment - Missing values.

Features	Missing Values
pH-Tertiary treatment	169
Temperature	147

After analyzing the dataset, were recognized, also, values outside the normal range. This values are represented in the table 3.20.

Table 3.20: Non meaningful tertiary treatment observations.

Date	pH-Tertiary treatment	Temperature
2018-08-02 08:00:00	757.0	-
2018-02-07 08:00:00	-	-17.5
2017-03-11 16:00:00	722.0	-

After remove the observations from the table above, the statistics were calculated, as table 3.21 shows.

Table 3.21: Tertiary treatment features statistical analysis.

Metrics	pH-Tertiary treatment	Temperature
Minimum	0.0	0.0
Maximum	9.56	31.9
Mean	7.076	21.384
Standard Deviation	11.329	4.747
Variance	1.766	22.537

The “Temperature” does not vary significantly, the reason why presents a small standard deviation. The graphs from the figures 3.29 and 3.30 shows the pH and the temperature behaviour over time, in the tertiary treatment unit.

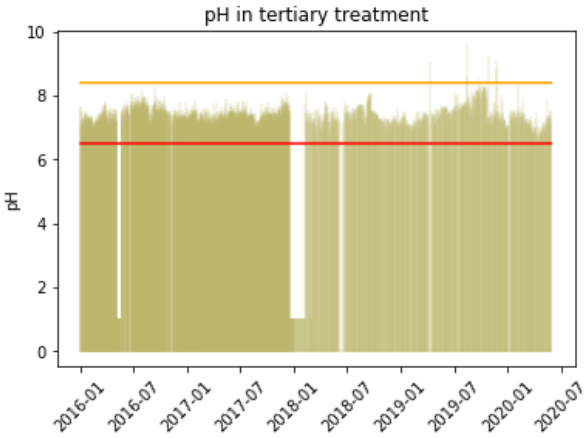


Figure 3.29: pH in tertiary treatment.

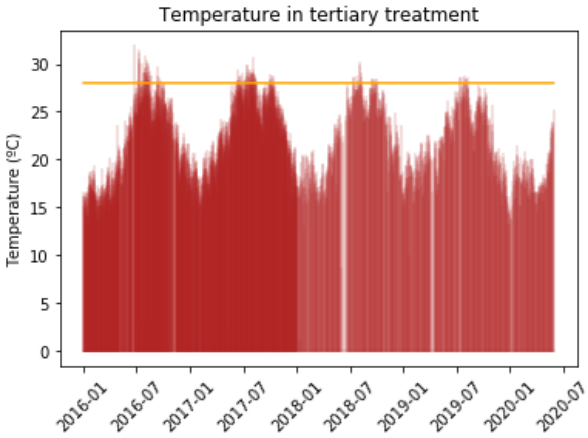


Figure 3.30: Temperature in tertiary treatment.

Analyzing the graphs, may be concluded that pH value has a similar behavior when compared with the pH values from the secondary treatment. Also, both features ("pH-tertiary treatment" and "Temperature") present a considerable number of registers when compared with other feature previously explored. It is substantiated by table 3.19, which presents a small number of missing values.

Besides, both features mostly comply with the defined limits, demonstrated in the chapter 2, as the horizontal lines in the graphs show.

3.1.2.4 Correlation between features

After analyzing all the features, some revealed themselves with scarcity of meaningful observations. Thus, to explore the features correlation were only used the ones that support more information. Figure 3.31 shows a correlation matrix, using, also, the spearman's rank correlation coefficient.

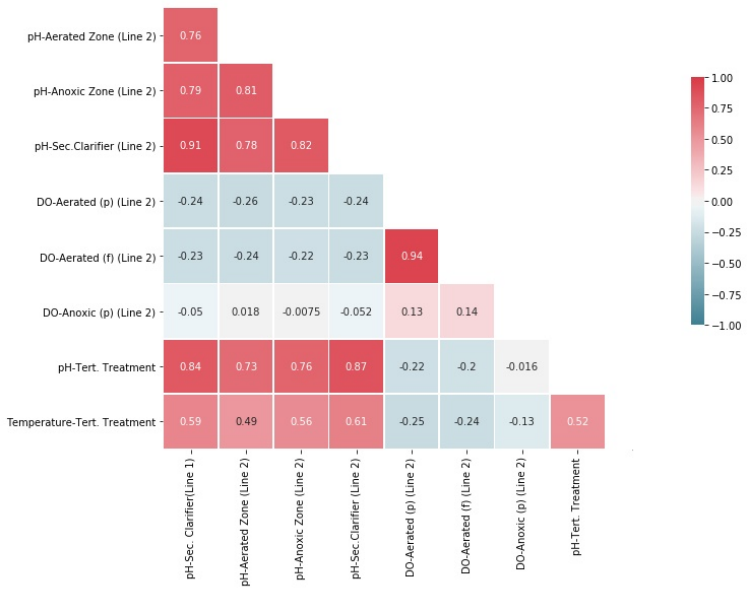


Figure 3.31: Correlation matrix concerning the water quality dataset.

Analyzing the matrix above it is possible to conclude that all the pH values available are highly correlated. Further, the DO measured with a fixed and portable meter, in the aerated zone, shows a high correlation, also. It was expected once, as mentioned before, some pH values features have similar behavior, as well as the DO in the aerated zone (meter with different approaches, fixed and portable meter, but resulting in similar observations).

3.2 Data preparation

Data preparation subsection addresses both dataset, previously explored. Thus, the main changes are next presented.

3.2.1 Air quality dataset

3.2.1.1 String Manipulation

In a first analysis, some observations presented some unexpected characters as well as not meaningful words, according to the known weather descriptions, in Portuguese. To leave no doubts, it was necessary to manipulate some strings, to make the data readable and understandable. The manipulations performed were:

(a) Some words exhibited defective characters ("Ã©"). Words such as "cÃ©u claro", "nÃ©voa" and "cÃ©u limpo" were found and considered wrong. So, the correct words should be: "céu claro", "névoa" and "céu limpo", respectively. For this reason, the characters "Ã©" were replaced for "é", using a KNIME node "String Manipulation";

(b) Using the same node, the "nuvens quebrados" was replaced for "nuvens quebradas", in the overall dataset;

(c) The third, and last incongruity related to strings in this data set, was the string "chuva de intensidade pesado". For the same reasons presented in the previous item, the string "chuva de intensidade pesado" was replaced by the string "chuva de intensidade pesada".

This process were implemented in KNIME and is shown in the figure B.1 of appendix B.

3.2.1.2 Join Data

After the data analysis, and realizing that the recorded dates were almost the same, among the three datasets (UV, atmospheric pollutants, and weather), these were joined, resulting in a single dataset (the air quality dataset).

To join the data, and match it, was used the attribute "Date". As consequence, it was lost a day (06-03-2019), once was the only date not common to the three datasets. From the transformation resulted

a single dataset, with 506 observations. This process was also implemented in KNIME, and is shown in figure B.2 from appendix B.

3.2.1.3 Remove "Precipitation" feature

As mentioned in a previous subsection, 3.1, the feature "Precipitation" has values that are inconclusive and do not allow to extract meaning and, further, present an extremely small standard deviation. For this reason, this feature was removed from the dataset.

Using the "Column Filter" node, in KNIME, the attribute "precipitation" was removed. Also, the same process was executed in python, by using the pandas' function "drop". The figure B.3 and B.7 from the appendix B, show this process, respectively.

3.2.1.4 Group data by date (day, month and year)

Once the UV index and the "CO Value" (target features) do not vary significantly in a day, the data were grouped by day. In a first instance was necessary to manipulate the "Date" attribute, to remove the hour, from it. To do this was used a "String Manipulation" node. After this, the date presented a string format. Thus, was necessary to convert the string to a correct date form, using the "String to Date&Time" node. Finally, the data were ready to be grouped by day. The metrics used to group the data were:

- Mean of "UV Value", "CO Value", "SO_2 Value", "Temperature", "Atmospheric pressure", "Humidity" and "Clouds" features;
- Mode of "Weather description", the most frequent description in a day;
- A list of "Weather description", an array representing all the descriptions in a day. This attribute was aggregated twice, using different metrics, in view of the different scenarios tested, explained further below, in Chapter 4.

These transformations are exhibited in the figure B.5, B.6 and B.7 from appendix B, respectively.

3.2.1.5 One-Hot Encoding

The "Weather description" attribute was converted through one-hot encoding. This process was made using the "Java snipped" KNIME node, used to prepare the data to implement in Decision Trees and Random Forest. Once the data was grouped by day, and a "Weather description" list created (for each day), is easier for the models to read each value, using the One-Hot encoding technique.

By creating a column for each "Weather description" value, and using a Java switch statement, each value from the list is verified. Then, is assign "1" if the "Weather description" match with the header, and "0" in the remaining ones. This process were implemented in KNIME, and demonstrated in the figure B.8 from the appendix B.

3.2.1.6 Label encoding

"Weather description", a categorical feature, shows twenty-seven different values (after data join). Once neural networks just accept and read numerical values, there was a need to convert these features into numerical data. Using a simple block of code, was possible to assign a number to each different value of the attribute "Weather description", resulting in twenty-seven different numbers, from 0 to 26.

Figure B.9, from appendix B, shows the process deployed in python.

3.2.1.7 Data binning

As previously mentioned, this research also addresses classification models. To enable this, the target value, UV Index, was binned, according to the WHO reference levels (table 2.4). Using the KNIME node "Numeric Binner", the data was processed (figure B.10 from appendix B).

Using the reference table, the process resulted in four different values: "Low", "Moderate", "High", and "Very High" (there were not verified UV indexes higher than 11). Thereby, a new column was created, the "UV Value binned" column.

3.2.1.8 Extract data fields

A column with each observed month was added to the model. Using the KNIME node "Extract data field", the month was extracted and an additional column was created, the "Month(name)" feature (figure B.11 from appendix B). Further, regarding the ANN model, the "Date" feature become infeasible to use, since this model only accept numerical values. Thus, the day, month and year were extracted from the date, in python, as figure B.12 from appendix B shows.

3.2.1.9 Data normalization

Once the data has several features with different ranges and units of measure, it was necessary to normalize it. This data transformation grant that all the numeric dataset features use a common scale after its implementation. Figures B.13 and B.14 (from appendix B) shows the normalization data process, using KNIME and python, respectively. Using $MinMaxScaler$ (in python), each attribute is scaling to a range, between zero and one, except the LSTM prediction. Here, this normalization range takes values between -1 and 1.

3.2.1.10 Missing timesteps

In a time series forecast, the previous information is crucial to understand the later one. Thus, as mentioned before, the datasets show some missing timesteps. Once were used an LSTM to predict the "UV Value" and the "CO Value", it is essential not to have missing timesteps. Consequently, were created the observations to cover these missing timesteps and then were filled with a value equal to 1.1. This process, implemented in python, is represented in the figure B.16, from appendix B.

3.2.1.11 Missing values

As previously mentioned, missing values were not find in the entire datasets. Therefore, no adjustments were made in the dataset.

3.2.2 Water quality dataset

3.2.2.1 Features removal

In this dataset were removed several features:

(a) From the pre-treatment unit: "'pH" and "Conductivity" since they present a high number of missing values;

(b) From the secondary treatment unit: all the pH features from line 1 were removed. These features are the "pH- Anoxic Zone" and "pH- Aerated Zone", because are features with incomplete registers, and consequently considered incomplete information. "pH-Secondary clarifier" were also removed once presents a behavior similar to the remaining pH features from the dataset, and would make the training dataset redundant.

Concerning the line 2 were removed the "Solids", once present also very incomplete information. Further, some pH features from this line were likewise removed ("pH-Anoxic Zone" and "pH-Aerated Zone") due to the redundancy issue. Due to this problem, the attribute "DO-Aerated Zone (f)" was dropped, once has similar registers when compared with the DO measured with the portable meter.

After this removal, the remaining features are:

- The "Date" attribute;
- A set of features concerning the line 2: "pH- Secondary Decanter", "DO- Anoxic Zone (p)", "DO-Aerated Zone (p)" features;
- Two features from the tertiary treatment, the "Temperature" and the "pH-tertiary treatment" (target feature).

To remove this set of features was used the "Column filter" KNIME node (also used in the air quality dataset).

3.2.2.2 Missing values

To deal with the missing values in the dataset, some criteria were taken:

(a) The "pH-tertiary treatment" has a special attention in the preparation step since it is the target feature. As mentioned before, this attribute (among others) presents some values equal to 1 and 0

that may represent missing values. It is may be conclude because, after a detailed analysis, it was possible to realize that they are presented in well defined time intervals and do not make sense when compared with close values, that is, the days immediately after and before this intervals. Thus, these values (equal to 0 and 1) were converted into missing values and then removed from the dataset.

(b) After this removal, the attribute "DO-Aerated Zone (p)" also presents values equal to 1 and 0, with the same pattern mentioned in the previous item. In this way, these values were converted into missing values and the corresponding rows subsequently removed from the dataset.

After these transformations, the dataset, previously with 2363 rows, was left with 2149 rows.

Finally, after these adjustments, the other remaining features ("Temperature", "DO-Anoxic Zone", and "pH-secondary decanter") still presented some punctual missing values. Thus, these missing values were replaced by meaningful values through linear interpolation. It is a process that adapts a function to the data available and extrapolates the missing values using this function. This process was made in python and it is demonstrated in figure B.15 from appendix B.

3.3 Technologies

This research was mainly supported by the python programming language and KNIME analytic software. These two technologies were crucial throughout all the process since the raw data analyzes to the exploration of the results. The python programming language was used to explore and prepare part of the available data. Further, this was mainly used to construct and train the MLP and LSTM models, using TensorFlow and Keras, as main libraries. Part of the work was developed in the spyder Integrated development environment (IDE), mostly the model's construction. To tune and train these models the Google Colaboratory cloud service were used. This service uses a Jupyter notebook environment and runs totally in the cloud, using computing resources, such as GPUs. Besides, other important libraries such as pandas, matplotlib, NumPy, and scikit-learn were also used.

The KNIME software was used also to explore and prepare data. Besides, the Decision Trees and Random Forest models were constructed and trained using this software. This process occurred in an ACER computer with an Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz processor, RAM of 8,00 GB, and an NVIDIA Geforce 920M graphic card.

4. Experiments

4.1 Experimental Setup

To find the best solutions and the most beneficial data preparation techniques, several scenarios were constructed and tested in this research.

4.1.1 Ultraviolet Index prediction data scenarios

Considering the air quality dataset a set of scenarios were built. Among the created scenarios, the features used are different, as well as the preparation/transformation process. Moreover, these scenarios differ according to the used models. Table 4.1 shows the scenarios used to train the Decision Tree and Random Forest models.

Table 4.1: Scenarios construction regarding UV index prediction (Decision Tree and Random Forest).

Tree UV Scenario 1	• Date • CO Value • SO_2 Value
Tree UV Scenario 2	• Date • CO Value • SO_2 Value • Temperature
Tree UV Scenario 3	• Date • CO Value • SO_2 Value • Temperature • Clouds
Tree UV Scenario 4	• Date • CO Value • SO_2 Value • Temperature • Clouds • Month
Tree UV Scenario 5	• Date • CO Value • SO_2 Value • Temperature • Clouds • Month • Weather description (Mode)
Tree UV Scenario 6	• Date • CO Value • SO_2 Value • Temperature • Clouds • Month • Weather description (One-hot encoded)

The setup presents these configurations mainly to understand the impact of the features in the model results. The first scenario, uses only the atmospheric pollutants ("CO value" and "SO_2 value") and the "Date" as features. Then, the "Temperature" is added, next the "Clouds", subsequently the "Month", and finally, the "Weather description". This last feature is differently configured, as long as presents a vast range of different values. It may result in an attempt to understand the data preparation impact in the models results.

To train the MLP model were used the scenarios outlined in the table 4.2.

Table 4.2: Scenarios construction regarding UV index prediction (Multilayer Perceptron).

MLP UV Scenario 1	• Day • Month • Year • CO Value • SO_2 Value
MLP UV Scenario 2	• Day • Month • Year • CO Value • SO_2 Value • Temperature
MLP UV Scenario 3	• Day • Month • Year • CO Value • SO_2 Value • Temperature • Clouds
MLP UV Scenario 4	• Day • Month • Year • CO Value • SO_2 Value • Temperature • Clouds • Weather description (Label encoded)

Here, there was a need to create new features such as "Day", "Month", and "Year", since the ANN only accept numeric values as input (which make the "Date" an infeasible feature). Besides, the "Weather description" was label encoded, instead of the one-hot encode process. So, the configuration is very similar when compared to the scenarios created to the tree-based models. Further, also enables us to understand the impact of the same features in the model.

Since the model's goal is to predict the UV index, this features does not configure in the tables above. Therefore, these are the "UV Value" and "UV Value binned" features, regarding regression and classification approaches, respectively.

Besides these models, the UV index was also predicted using an LSTM model, based on a time series approach. On this wise, the model was deployed using only the "UV Value" as a feature.

4.1.2 Carbon Monoxide value prediction data scenarios

Considering that for CO forecasting only a time series approach was followed, the "CO Value" was the only feature used, similarly to the LSTM UV index forecast.

4.1.3 Water pH prediction data scenarios

Concerning the water quality dataset, were created four different data scenarios. Further, these were applied to Decision Trees, Random Forest, and MLP models. For the reasons previously mentioned, it is not feasible for MLP models to use the "Date" as input. Consequently, this feature was replaced by new four: "Day", "Month", "Year", and "Hour".

Table 4.3 shows the features used to train the MLP model. Once again, the only difference between these data scenarios and the ones used in the tree-based models is the "Date" feature.

Table 4.3: Scenarios construction regarding water pH prediction (Multilayer Perceptron).

Scenario 1	• Day • Month • Year • Hour • pH-Secondary clarifier • DO-Aereated Zone (p) • DO-Anoxic Zone (p) • Temperature
Scenario 2	• Day • Month • Year • Hour • pH-Secondary clarifier • DO-Aereated Zone (p) • DO-Anoxic Zone (p)
Scenario 3	• Day • Month • Year • Hour • DO-Aerated Zone (p) • DO-Anoxic Zone (p) • Temperature
Scenario 4	• Day • Month • Year • Hour • pH-Secondary clarifier • Temperature

These scenarios were created to understand the DO, the temperature, and the pH (from another zone of the WWTP) impacts in the models results (concerning the water pH prediction in the tertiary treatment unit).

4.2 Tree-based models

This subsection shows the conception and tuning process of the simplest models explored, the tree-based models (Decision Trees and Random Forest). These exhibit some simplicity once they are easy to construct and achieve good results without requiring high computational efforts.

4.2.1 Decision Trees

The first tree-based model explored are the Decision Trees. This model, as mentioned before, was constructed, tuned, trained, and deployed using the KNIME software. Further, were used regression and classification models to forecast some parameters according to the nature of the target features.

4.2.1.1 Model conception

Classification

Figure 4.1 shows the classification Decision Tree construction and deployment process, in KNIME.

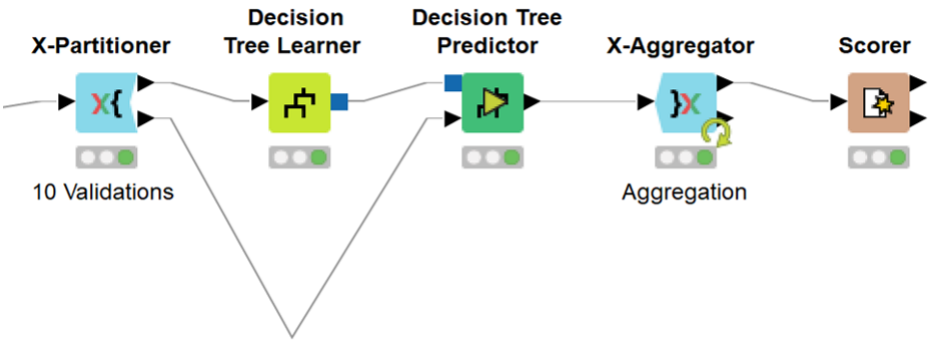


Figure 4.1: Classification Decision Tree model conception, in KNIME.

The model construction exhibit a node *"X-Partitioner"* that allows the model cross-validation. Here, the technique used is K-fold cross-validation. This technique consists of split all the dataset in K subsets (mutually exclusive) and, from there, a subset is used to test the model, while the remaining ones ($K-1$) are used to train the model and validate it. Cross-validation is a commonly used technique since it allows the model generalization, and consequently, decreases the occurrence of overfitting. In this tree construction process ten iterations are used, which means, a K value equal to 10. Thus, all the trees implemented in this research use ten validations at the cross-validation process.

Further, the model construction uses a *"Decision Tree Tree Learner"*, the node that allows the model train process, and the target attribute choice (must be a nominal value). In this study, the only target feature used, regarding a classification problem, is the "UV Value binned". Besides, using the C4.5 algorithm, this node also allows to define the regression tree parameters, which are:

- Quality measure: To select the quality measure. The Available options are the "Gini Index" and the "Gain Ratio";
- Pruning method: The available options are the Minimal Description Length (MDL) or none;
- Reduced Error Pruning: If checked (default), a simple pruning method is used to cut the tree in a post-processing step: starting by the leaves, each node is replaced with its most popular class, but only if the prediction accuracy doesn't decrease. Reduced error pruning has the advantage of simplicity and speed;
- Min number records per node: To select the minimum number of records at least required in each node. If the number of records is smaller or equal to this value the tree is not grown any further. It corresponds to a stopping criterion (pre pruning).

After this, a *"Decision Tree Predictor"* node is used for the prediction process, as well as a *X-Aggregator*. This node is the end of a cross-validation loop. It extracts the results from the predictor node, collecting all the ten iterations results. Finally, to perceive the model performance, and ideal to evaluate classification models, a *"Scorer"* node is used. The output of this node is a set of metrics, such as the accuracy (%), the error(%), the number of correctly classified, the number of wrongs classified, and the Cohen's Kappa (k). Besides, this node generates a confusion matrix, used to compare which classes are correctly and wrong predicted.

Regression

As mentioned before, regression Decision Trees were also implemented. This model construction was made using KNIME, as figure 4.2 shows.

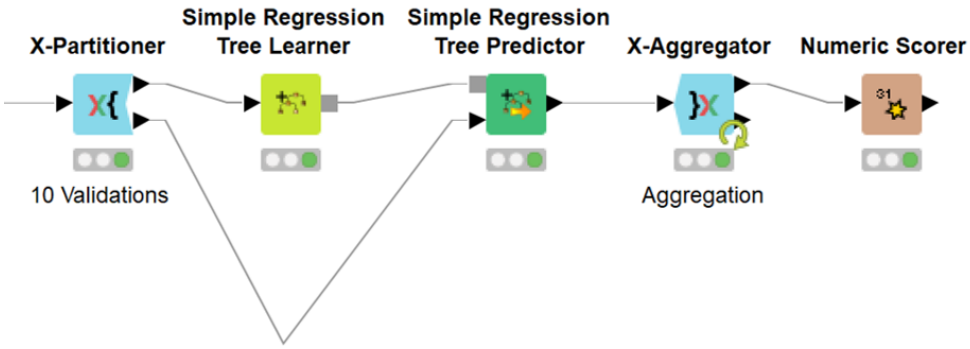


Figure 4.2: Regression Decision Tree model conception, in KNIME.

"Simple Regression Tree Learner" is the node that enables the learning process. Here, it is applied the CART algorithm with some simplifications. Further, it enables us to define the target feature, which must be a numerical value. Varying according to the dataset, the target features in the present research are the "UV Value"and the "Water pH".

The regression Decision Tree parameters are:

- Missing value handling: There are two available options : XGBoost and Surrogate. The first one acts as the default value. In this case, the tree learner must determine which direction is appropriate for missing values, by sending these in each direction of a split. The direction from which the biggest gain results is used as a default direction for missing values. The other option, Surrogate, compute for each split alternative splits, approximating the optimal one;
- Limit number of levels (tree depth): The number of tree levels in which the model will learn;
- Minimum split node size: The minimum number of records in a Decision Tree node to attempt the next split. For this reason, this parameter does not make implications in a terminal node, once there is no next split. This number needs to be at least twice as large as the minimum child node size;

– Minimum node size : The minimum number of records in child nodes (can have at most half of the minimum split node size). This parameter is not used for nominal splits;

The *"Simple Regression Tree Predictor"*, as the name implies, is the node responsible for the predictions. The *"Numeric Scorer"* allows understanding the model performance once it calculates individual statistics, computed between the effective values and the predicted ones. So, it uses statistics, such as: R^2-coefficient of determination, MSE, MAE, RMSE, Mean signed difference, and Mean absolute percentage error.

4.2.1.2 Model tuning

To tune the ML models several parameters were used. According to the model, it was established a range of values for each parameter, to further find its best combination. Using loops to vary these values (numeric or nominal), several iterations resulted from the process, finding the optimal combination for each model and dataset.

Classification

For the classification Decision Tree model, the parameters used to tune the model are presented in table 4.4.

Table 4.4: Set of parameters used to tune the classification Decision Trees.

Nominal parameters			
Quality measure	['Gini Index', 'Gain ratio']		
Pruning method	['No pruning', 'MDL']		
Reduced error pruning	['True', 'False']		
Numerical parameters			
	Start value	**Stop Value**	**Step size**
Minimun records per node	1	15	1

Using these parameters the model was tuned, using all the possible combinations. The classification Decision Tree tuning process was implemented using the workflow presented in the following figure, figure 4.3.

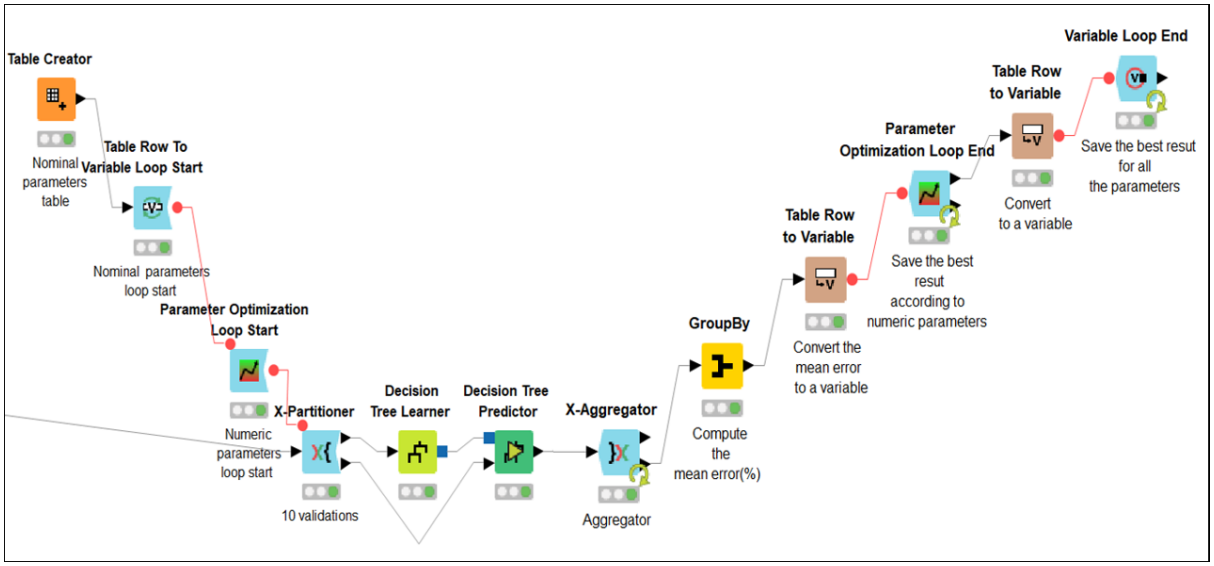


Figure 4.3: KNIME Workflow to carry out the classification Decision Tree tuning process.

Using a *"Table creator"* node were defined the different possibilities of nominal parameters, the ones presented in the table 4.4. After, using the node *"Table Row to Variable Loop Start"*, the model can run using each one of the nominal parameters, defined in the previous step. Next, using the *"Parameter Optimization Loop Start"*, the numerical parameters (from table 4.4) can be defined, allowing the model execution using each one of this numerical options. As previously mentioned, the *"X-Partitioner"* node is responsible for the model cross-validation. Consequently, the model will execute ten times for each parameter combination. Thereafter, the learning process occurs in the node *"Decision Tree Learner"* (ten times for each combination), for later, in the *"Decision Tree Predictor"* node, to occur the prediction process. The *"X-Aggregator"* node collects all the results from the 10 validations, to then, compute the mean error (%), using the *"Group by"* node. Since the model performance is evaluated based on this metric, the mean error resulting from this step needs to be converted to a variable, using the *"Table Row to Variable"* node. Afterward, to have the desired results, the *"Parameter Optimization Loop End"* save the best combination of numerical parameters (the combination with the reduced error(%)), for then convert this into a variable (using the *"Table Row to Variable"* node). The last step is the *"Variable Loop End"* that saves the best combinations for each nominal parameter.

Regression

Using the parameters from table 4.5, the Decision Tree regression model was tuned.

Table 4.5: Set of parameters used to tune the regression Decision Trees.

Nominal parameters			
Missing value handling	['XGBoost','Surrogate']		
Numerical parameters			
	Start value	**Stop Value**	**Step size**
Limit number of levels (tree depth)	1	20	1
Minimum node size	1	15	1

Using each combination of nominal and numerical parameters the model was evaluated. The deployment process to tune this model (regression Decision Tree) is presented in figure C.1 from appendix C. This tune workflow is similar to the one from the figure 4.3, being the model used (a regression tree) the only difference pointed out.

4.2.2 Random Forest

The other tree-based model used was a Random Forest. Alike, this model was implemented making use of the software KNIME, as a classification and regression model.

4.2.2.1 Model conception

Classification

The classification Random Forest conception is represented in the figure 4.4.

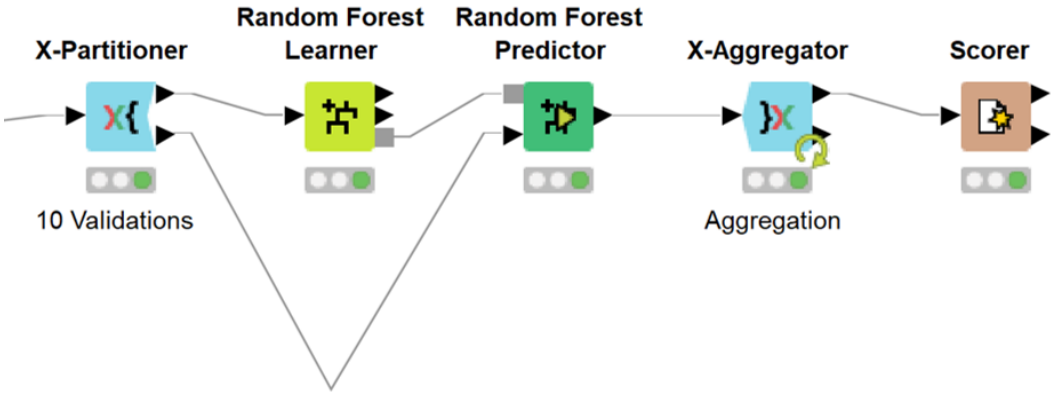


Figure 4.4: Classification Random Forest model conception, in KNIME.

This model is implemented similarly to the regression Decision Tree model. The main differences are the "*Random Forest Learner*" and "*Random Forest Predictor*" nodes. The first, as known, supports

the Random Forest train process, allowing the assigning of the target attributes, in this research, the "UV value binned". The second provides the prediction process.

Besides, the parameters assigned in the *"Random Forest Learner"* are different, these being:

- Split Criterion: Here is choosen the split criterion. Gini index, information gain and gain ratio are the possibilities;
- Limit number of levels (tree depth): Number of tree levels in which trees will learn;
- Minimum child node size: The minimum number of records in child nodes. If each tree fits perfectly in the training dataset (the training dataset does not have equivalent rows with different labels), this number is equal to 1;
- Number of models: The number of Decision Trees to be learned.

Regression

Equally to the Decision Tree model, the Random Forest was implemented using regression and classification models (figure 4.5).

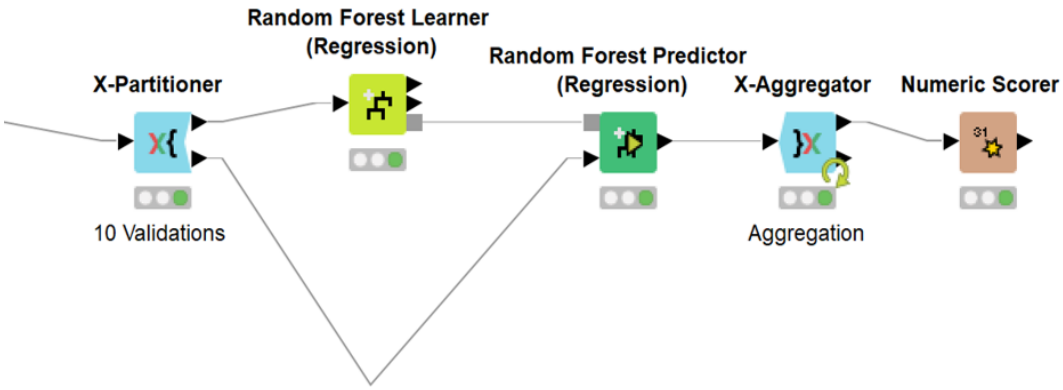


Figure 4.5: Regression Random Forest model conception, in KNIME.

This model construction shows many similarities with the Decision Trees conception process. Thereby, use also a *"X-Partitioner"* to perform the model cross-validation (ten validations). The main differences are in the following two nodes: the *"Random Forest Learner (Regression)"* and the *"Random Forest Predictor (Regression)"* (the predictor node). The first one is the node responsible for the learning process. This node allows the model to learn using an ensemble of regression Decision Trees. Here, the target attribute is chosen. According to the problem over study, these target features are the "UV Value"and the "Water pH". Additionally, this *"Random Forest Learner (Regression)"* is also where the Random Forest parameters are assigned. These parameters are:

- Limit number of levels (tree depth): Number of tree levels to be learned;
- Minimum child node size : The Minimum number of records in child nodes;
- Number of models: The number of regression trees to be learned.

4.2.2.2 Model tuning

Classification

The model was tuned using the parameters from the table 4.6. Further, the model was evaluated using every combination of nominal and numerical parameters, regarding the classification Random Forest.

Table 4.6: Set of parameters used to tune the classification Random Forest.

Nominal parameters			
Split criterion	['InformationGain', 'InformationGainRatio','Gini']		
Numerical parameters			
	Start value	**Stop Value**	**Step size**
Limit number of levels (tree depth)	1	25	1
Minimum child node	1	15	1
Number of Models	100	1200	100

The workflow to implement this tuning process is presented in the figure C.2, from the appendix C. This workflow has similarities with the one explained above (from figure 4.3), except for the model used (now a Random Forest classifier).

Regression

In the regression Random Forest model, only numerical parameters were tuned, as table 4.7 shows.

Table 4.7: Set of parameters used to tune the regression Random Forest.

Numerical parameters			
	Start value	**Stop Value**	**Step size**
Limit number of levels(tree depth)	1	15	1
Minimum child node size	1	15	1
Number of Models	100	1200	100

The resources used to tune these model are in figure C.3 from the appendix C. This process is similar to the tuning workflows mentioned before but simplest, once only use loops to the numerical parameters.

4.3 Deep learning models

Deep learning models are more complex and require more computing efforts when compared with tree-based models. Thus, the MLP and the LSTM are the models used in this context. Following, the models' conception and tuning processes are presented. Here, it is used the python programming language, supported by Keras and Tensorflow.

4.3.1 Multilayer Perceptron

4.3.1.1 Model conception

The first step of the model conception was build the ANN (using python) responsible to the MLP implementation, as figure 4.6 shows. This model was implemented for regression and classification problems.

```
def build_model(h_layers = 2, neurons = 128,
                activation = 'relu',
                learn_rate = 0.001):
    model = keras.Sequential ()

    model.add(layers.Dense(neurons, activation = activation, input_shape = [(len(data.keys()))]))

    if h_layers==1:
        model.add(layers.Dense(neurons, activation = activation))

    elif h_layers==2:
        model.add(layers.Dense(neurons, activation = activation))
        model.add(layers.Dense(neurons, activation = activation))
    else:
        model.add(layers.Dense(neurons, activation = activation))
        model.add(layers.Dense(neurons, activation = activation))
        model.add(layers.Dense(neurons, activation = activation))

    model.add(layers.Dense(1))

    optimizer = tf.keras.optimizers.RMSprop(learning_rate=learn_rate)

    model.compile(loss = rmse,
                  optimizer = optimizer,
                  metrics = ['mae' , rmse])
    return model
```

Figure 4.6: Multilayer Percepetron model, in python.

The MLP model was constructed using a function, called “build_model”. In its constitution, this function has several particularities that deserve special attention. First, the MLP has one input layer. It is a dense layer that implements the operation: output = activation(dot(input, kernel) + bias), where “activation” refers to the activation function and the “neurons” to the layer number of neurons. These parameters will be tuned (4.3.1.2) to understand which ones fit better and, consequently, produce greater results. Besides, this model presents one, two, or three hidden layers, with a respective activation function. The hidden layers’ optimal number will be also determined, further ahead. Additionally, the model has one output layer that is defined also, as a dense layer, with a single neuron.

Further, the model has an *RMSprop optimizer Keras class* which is an optimizer that implements the RMSprop algorithm. This algorithm maintains a moving average of the square of gradients, and divide the gradient by the root of this average. Moreover, is here where the learning rate is defined.

Furthermore, the *model compile* works as a training Application Programming Interface (API) and enables us to define the evaluation metrics list, used during the training and test process (in this particular example, the RMSE and MAE). Besides, here is assigned the optimizer used (*RMSprop*), and defined the loss, this is, the metric to be minimized by the model.

The model from the figure concerns the regression prediction. Additionally, as previously mentioned, the MLP model was also implemented as a classification problem (presenting few differences when compared with the regression one). The main differences between these two are the evaluation metrics, once is used the accuracy for the classification problem. Further, the output layer has also a different configuration, using four neurons (regarding the "UV value binned" prediction, the four classes available: "Low", "Moderate", "High", and "Very High").

The split into train and test dataset was made using cross validation, using three iterations (figure 4.7).

```
seed = 7
numpy.random.seed(seed)
kfold = KFold(n_splits=3, shuffle=True, random_state=seed)

for train, test in kfold.split(dataset):
    #Build data
    x_train = dataset.iloc[train]
    y_train = labels[train]
    x_test = dataset.iloc[test]
    y_test = labels[test]

    #Build model
    model = build_model()

    #Fit the model
    model.fit(x_train,y_train, epochs=200, batch_size=16, verbose=0)

    #Evaluate the model, using predictions

    #Predict using the x_test
    predictions = model.predict(x_test)

    #Invert the normalization of predicted y values
    real_predictions = normalizers['Uv Value'].inverse_transform(predictions)

    #Invert the normalization of real y values
    y_test = y_test.values.reshape(-1,1)
    real_y_test = normalizers['Uv Value'].inverse_transform(y_test)

    #Variables to save the predicted and true values (denormalized), by iteration
    y_true_summary.append(real_y_test)
    pred_true_summary.append(real_predictions)

    #Compute the MAE and save it, by iteration
    real_mae = mean_absolute_error(real_y_test,real_predictions)
    mean_ae.append(real_mae)
    print(real_mae)

    # Compute the RMSE and save it, by iteration
    real_rmse = sqrt(mean_squared_error(real_y_test,real_predictions ))
    root_ms.append(real_rmse)
    print(real_rmse)
    print("------")
```

Figure 4.7: Multilayer Percepetron deplyoment, using cross-validation, in python.

The code from the figure above shows the model's train and test process. First, it is applied the neural network previously defined (figure 4.6). Further, it is used the *dataset*, which represents each scenario (a set of features). From that, for each iteration, is defined the *x_train*, representing the train dataset, and the *x_test* (representing the test dataset), using also *y_train* and *y_test* as labels, for each one, respectively. Moreover, to train the model (for each iteration) is used the *model.fit*. Here, is assigned

the training dataset and the respective labels. Also, is here defined the *batch_size* and the number of epochs. Finally, after the training process, the model is tested, to posteriorly calculate the MAE and the RMSE. To create meaningful results the labels were denormalized.

4.3.1.2 Model tuning

Using MLP model, the tuning process was implemented using the parameters following exposed, table 4.8.

Table 4.8: Set of parameters used to tune the MLP model.

Activation Function	['relu', 'tanh', 'sigmoid']
Hidden Layers	[1,2,3]
Learn Rate	[0.001, 0.005, 0.01]
Neurons	[16, 32, 64, 128]
Batch Size	[16, 23, 30]

To find the best parameters for the model (the parameters combination that results in the smaller error/best performance) the model tuning was made.

The first step consists of finding the best number of epochs. To determine this number the model was deployed using the values of the smallest and highest parameters (from the defined ranges presented in the table 4.8) for each activation function.

Accordingly, as the values of the lowest and highest parameters, were used:

Lowest	**Highest**
Hidden layers: 1	Hidden layers: 3
Learn rate: 0.001	Learn rate: 0.01
Neurons: 16	Neurons: 128
Batch Size: 16	Batch Size: 30

Further, using these parameter combinations, the evaluation metrics were analyzed to understand which would be the ideal epochs number. Consequently, it was created a learning curve (for each combination), showing the evaluated metrics (RMSE, MAE, and accuracy) by epoch. This iterative process consists in perceiving, through the analysis of the graphs, around which epoch these metrics stop vary significantly. After recognizing this, for each activation function, the maximum number obtained among these tests is chosen to ensure that the model will train for sufficient epochs.

Moreover, to tune the remaining parameters was used the Grid Search hyper-parameters optimization technique. This technique allows us to establish a grid of parameter values, train the model, and give an evaluation metric for each combination (with all combinations being tested). The Grid Search implementation was made in python, as figure 4.8 shows.

```
model = KerasClassifier(build_fn = build_model, epochs = 200 , verbose = 0, batch_size = 16)

activation = ['relu', 'tanh', 'sigmoid']
h_layers_v = [1, 2, 3]
learn_rate = [0.001, 0.005, 0.01]
neurons = [16, 32, 64, 128]
batch_size = [16,23,30]

param_grid = dict(h_layers = h_layers_v,
                  activation = activation,
                  learn_rate =learn_rate,
                  neurons = neurons,
                  batch_size = batch_size)

grid = GridSearchCV(estimator = model, param_grid = param_grid, cv=3,
scoring = "neg_mean_absolute_error")

grid_result = grid.fit(dataset_train, train_labels)

print(grid_result.cv_results_)
print ("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']


for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Figure 4.8: Multilayer Perceptron tuning process using Grid Search.

The *dict* function made the MLP model parameters vary, using all the parameters combinations, from the table 4.8. The *GridSearchCV* enable to define the scoring metric, using the "neg_mean_absolute_error", in this case. The designation "neg" is because the algorithm chose the higher value. Once it is an error metric, the idea is to choose the smallest error value from the best combination, among all. Here was used three iterations to split the training dataset (here, defined as 80% of the dataset). So, is used the training dataset to train the model, and consequently, to evaluate the combinations.

4.3.2 Long Short-Term Memory

4.3.2.1 Model conception

The LSTM model deployment was done using python. Figure 4.9 shows the script that generated the recurrent network, responsible for the process.

```
def build_model(timesteps, features,
                h_layers = 1,
                neurons = 16,
                activation = 'relu',
                learn_rate = 0.001):
    model = keras.Sequential ()

    model.add(layers.Masking(mask_value=1.1VER,input_shape=(timesteps, features)))

    if h_layers==1:
        model.add(layers.LSTM(neurons, return_sequences=False, activation = activation))

    elif h_layers==2:
        model.add(layers.LSTM(neurons, return_sequences=True, activation = activation))
        model.add(layers.LSTM(neurons, return_sequences=False, activation = activation))

    else:
        model.add(layers.LSTM(neurons,  return_sequences=True, activation = activation))
        model.add(layers.LSTM(neurons,  return_sequences=True, activation = activation))
        model.add(layers.LSTM(neurons, return_sequences=False,  activation = activation))

    model.add(layers.Dense(1))

    optimizer = tf.keras.optimizers.RMSprop(learning_rate=learn_rate)

    model.compile(loss = rmse,
                  optimizer = optimizer,
                  metrics = ['mae' , rmse])
    return model
```

Figure 4.9: LSTM model, in python.

This construction is similar to the MLP model conception, represented in the figure 4.6. Here, the main differences are latent right in the input layer, once is used a *Masking layer*. This layer is used to mask the missing timesteps in the sequence. Each timestep with a missing observation will gain a value equal to 1.1, and, due to this layer, this value will be masked, skipping to the next timestep. 1.1 was chosen to not interfere with the known values, which, once normalized, these take values between -1 and 1. The second difference to point out is the layer applied. Here is used *LSTM layer* instead of the *Dense layer* used in the MLP model.

Once the LSTM model was implemented as a time series forecast, there was the need to put the data into the right shape. The logic behind this is to create a sequence of values (corresponding to consecutive days) with a timestep size. It is the parameter that will define how many days the forecast will be based on. The idea is forecast the day after the last day of that sequence, using, next, the real value to evaluate the prediction quality. For this, the function from figure 4.10 was used.

```
def to_supervised(df, timesteps, uv_value_col=0):
    data = df.values
    X, y = list(), list()
    #iterate over the training set to create X and y
    dataset_size = len(data)
    for curr_pos in range(dataset_size-timesteps):
        #end of the input sequence is the current position + the number of timesteps of the sequence
        input_index = curr_pos + timesteps
        #end of the labels corresponds to the end of the input sequence + 1
        label_index = input_index + 1
        #if we have enough data for this sequence
        if label_index < dataset_size:
            X.append(data[curr_pos:input_index, :])
            y.append(data[input_index:label_index, uv_value_col])
    return np.array(X).astype('float32'), np.array(y).astype('float32')
```

Figure 4.10: Function to reshape the data to LSTM time series forecast.

Using the complete sequence of data ("UV Value" and "CO Value") and the number of timesteps, the data is splitting using the deduction mentioned above. First, the iteration process is doing through the entire sequence minus the timesteps value. This happens to ensure that the entire sequence of values has always an available value for evaluating the forecast (a real label at the end of the sequence), and further, to evaluate the model. After this, from that data (used as input), are created two variables, the X and the y. The X is the sequence of values, with the size equal to $timesteps$ and the y is the value after the last day of that sequence, the value that will also be forecast. Besides, y includes only one value, since we are making recursive multi-step forecast. In a dataset, the number of sequences created is equal to $dataset\_size\text{-}timesteps$, as well as the number of y labels.

To forecast the values is used a function (figure 4.11), allowing the blind and known forecast. The blind forecast uses the prediction values to forecast. As the values are predicted, the actual values are replaced by these forecasts, allowing further predictions. For example, use the prediction values of the next day to forecast the day after. On the other hand, the known forecast use only the known values to predict the values, without replacing it.

```
def forecast(model, X_test, y_test, timesteps, multisteps, features, scaler, uv_value_col=0, blind=True):
    #using three dys
    days = 3
    input_seq = X_test[-days:, :, :]
    labels_seq = y_test[-days:, :]
    #iterate over the day
    results = list()
    for i in range(len(input_seq)-(multisteps-1)):
        inp = input_seq[i].copy()
        lab = labels_seq[i].copy()
        #for each step of the multistep
        labels = list()
        predictions = list()
        rmse_scores = list()
        mae_scores = list()
        for step in range(1, multisteps+1):
            #reshape
            inp = inp.reshape(1, timesteps, features)
            lab = lab.reshape(1, 1)
            #predict the value for the next timestep
            yhat = model.predict(inp, verbose=0)
            #invert normalized values
            lab_inversed = scaler.inverse_transform(lab)
            yhat_inversed = scaler.inverse_transform(yhat)
            #compute rmse and mae between true value and prediction
            rmse_val = mean_squared_error(lab_inversed, yhat_inversed)
            mae_val = mean_absolute_error(lab_inversed, yhat_inversed)
            #store results
            labels.append(lab_inversed[0][0])
            predictions.append(yhat_inversed[0][0])
            rmse_scores.append(rmse_val)
            mae_scores.append(mae_val)
            #insert a new value into the input sequence to predict the next timestep.
            #if blind we will use our prediction
            #If not blind we will use the real, known, value
            if blind:
                if step != multisteps:
                    #add yhat(the forecasted value)  to input sequence
                    new_line = input_seq[i+step][-1].copy()
                    np.put(new_line, uv_value_col, yhat)
                    new_line = new_line.reshape(1, features)
                    inp = np.concatenate((inp[0], new_line))
                    inp = inp[-timesteps:]
                    #update label to the next timestep
                    lab = labels_seq[i+step]
            else:
                if step != multisteps:
                    #add the real value to input sequence
                    inp = input_seq[i+step].copy()

                    #update label to the next timestep
                    lab = labels_seq[i+step]
        results.append((np.array(labels), np.array(predictions), np.mean(rmse_scores), np.mean(mae_scores)))
    return np.array(results)
```

Figure 4.11: LSTM forecast function.

This function allows the predictions based on the multistep values, the number of the days we want to forecast. First, the sequences are defining (*input_seq*), based on the *x_test* (the sequences used to test the model). The *x_test* stores individual sequences, an amount equal to the *days* chosen. These sequences have the size of the timestep value. Then, the labels sequence (*labels_seq*) is also defined. This number of labels is equal, also, to the *days* variable number. After this, the function computes the prediction, using *model.predict*. Since the predictions (*yhat*) and the corresponding labels (*lab*) are normalized, it is necessary to invert this. Subsequently, the RMSE and MAE are calculate, using the

denormalised data. These results are stored in a variable in each iteration. Then, if the blind approach is chosen, the *yhat* value, previously computed, will replace the real values in the sequence. In contrast, if the known values approach is chosen, the values are forecasted, using just the real values of the sequence. Finally, the function returns an array, with the labels (the effective values), the predictions (depending on the approach chosen), and the RMSE and MAE values.

After the model construction and the data reshape, the model is ready to be trained. To train the model there are some important parameters (tuned ahead) to assing. The figure 4.12 shows the parameters used in the LSTM model deployment.

```
timesteps =  21                          #the sequence size
multisteps = 3                           #how much to predict
features = 1                             #number of attributes used
batch_size =30

h_layer =  1                             #number of hidden layers
h_neuron =  32                           #number of neurons in each layer
activation = 'sigmoid'                   #LSTM layers activation function
learn_rate = 0.001                       # Learn Rate
```

Figure 4.12: Example of LSTM parameters values, in python.

After all, the model can be trained, as figure shows 4.13.

```
def split_dataset(training, perc=10):
    train_idx = np.arange(0, int(len(training)*(100-perc)/100))
    val_idx = np.arange(int(len(training)*(100-perc)/100+1), len(training))
    return train_idx, val_idx

X, y = to_supervised(dataset, timesteps)

tscv = TimeSeriesSplit(n_splits=3)
hist_list=[]
loss_list=[]
backtesting_loss = defaultdict(list)
evaluate_loss = defaultdict(list)

for train_index, test_index in tscv.split(X):
    #further split into training and validation sets
    train_idx, val_idx = split_dataset(train_index, perc=10)
    #build data
    X_train, y_train = X[train_idx], y[train_idx]
    X_val, y_val = X[val_idx], y[val_idx]
    X_test, y_test = X[test_index], y[test_index]

    #build model
    model = build_model(timesteps, features, h_layers=h_layer, neurons=h_neuron, activation=activation,
    learn_rate=learn_rate)

    #fit the model
    history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=150,
    batch_size=batch_size, shuffle=False, verbose=0)
    hist_list.append(history)

    #evaluate model
    metrics = model.evaluate(X_test, y_test, verbose=0)
    loss_list.append(metrics[2])
    evaluate_loss['RMSE'].append(metrics[2])
    evaluate_loss['MAE'].append(metrics[1])
```

Figure 4.13: LSTM deplyoment, using cross-validation, in python.

In this process was used cross-validation, using three iterations. The train and test indexes are defining by the cross-validation and, consequently, the train and test dataset. From the training dataset, 10% are removed to form the validation dataset, using the *split_dataset* function. Further, the parameters are assigned, building the model. Moreover, the training process occurs (due to *model.fit*). Here, the number of epochs (in which the model will be trained), the batch size, and the validation data (the validation dataset) are assigned. Besides, the model is evaluated (*model.evaluate*), using the test data. Finally, the forecast function was applied to predict the desired days. In this specific case, were used the last three days of the dataset, using a blind and known forecast (as previously mentioned). The figure 4.14 shows the employment of this function.

```
blind_results = forecast(model,
                         X,
                         y,
                         timesteps,
                         multisteps,
                         features,
                         scaler['CO Value'],
                         uv_value_col=uv_value_col,
                         blind=True)

known_results = forecast(model,
                         X,
                         y,
                         timesteps,
                         multisteps,
                         features,
                         scaler['CO Value'],
                         uv_value_col=uv_value_col,
                         blind=False)
```

Figure 4.14: LSTM forecast (blind and known predictions), in python.

4.3.2.2 Model tuning

The LSTM model parameter optimization was implemented using the values shown in the following table 4.9.

Table 4.9: Set of parameters used to tune the LSTM model.

Activation Function	['relu', 'tanh', 'sigmoid']
Hidden Layers	[1,2,3]
Learn Rate	[0.001, 0.005, 0.01]
Neurons	[16, 32, 64, 128]
Batch Size	[16, 23, 30]
Timesteps	[7, 14, 21]

When compared with the MLP model, the parameter values are the same, except for the "Timesteps". This parameter is crucial once determine the sequence of days used to predict.

The first step consists in find the best number of epochs. To decide this number is used the iterative process explained in 4.3.1.2. Using the values of the smallest and highest parameters from the defined ranges (shown in the table above), for each activation function, are created learning curves regarding the RMSE. Accordingly, as values of the lowest and highest parameters were used:

Lowest	**Highest**
Hidden layers: 1	Hidden layers: 3
Learn rate: 0.001	Learn rate: 0.01
Neurons: 16	Neurons: 128
Batch Size: 16	Batch Size: 30
Timesteps: 7	Timesteps: 7

Here, the parameter combinations are among the variables in the figure 4.15.

```
uv_value_col = 0

#main variables
timesteps_list = [7,14,21]
multisteps = 3
features = 1
batch_size_list = [30]
cv_splits = 3

#model hyperparameters
h_layers = [1,2,3]
h_neurons = [16,32, 64, 128]
activations = ['relu','tanh','sigmoid']
learn_rates = [0.001,0.005,0.01]
```

Figure 4.15: LSTM parameters used to tune the model.

Using a *for* loop, all the parameters value were combined, giving a result for each iteration. In this way, all the parameter combinations previously defined are executed. For each combination, the process explained in 4.3.2.1 is deployed. First, the data is split to achieve the ideal format for the time series forecast. Second, the cross-validation (using tree iterations) occurs. Then, the train, test, and validation index are defined, and consequently, the train, test, and validation datasets. Further, assigning each current parameter to the model it is fitted (train process) and evaluated, using the test data (*X_test* and *y_test*). Furthermore, the values are predicted based on a blind and known (real) forecast, using the forecast function before defined (figure 4.11). Finally, a set of metric results from this process. It is important to give particular attention to the blind forecast since it is the correct method of evaluating a prediction (even if it results in, often, worse results). Thus, the evaluation metrics from the blind forecast will allow defining the optimal set of parameters, the combination that produces the small error when used the blind forecast. Thw following figure, 4.16, depicts this process.

```
for batch_size in batch_size_list:
    for timesteps in timesteps_list:
        for h_layer in h_layers:
            for h_neuron in h_neurons:
                for activation in activations:
                    for learn_rate in learn_rates:
                            #create supervised problem with a one-timestep shift
                            X, y = to_supervised(df_road, timesteps, uv_value_col=0)

                            #Timeseries split for model validation
                            tscv = TimeSeriesSplit(n_splits=cv_splits)

                            for train_index, test_index in tscv.split(X):
                                #further split into training and validation sets
                                train_idx, val_idx = split_dataset(train_index, perc=10)
                                #build data
                                X_train, y_train = X[train_idx], y[train_idx]
                                X_val, y_val = X[val_idx], y[val_idx]
                                X_test, y_test = X[test_index], y[test_index]

                                #build model
                                model = build_model(timesteps, features, h_layers=h_layer, neurons=h_neuron,
                                activation=activation, learn_rate=learn_rate)

                                #fit the model
                                history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
                                epochs=150, batch_size=batch_size, shuffle=False, verbose=0)
                                hist_list.append(history)

                                #evaluate model
                                metrics = model.evaluate(X_test, y_test, verbose=0)
                                loss_list.append(metrics[2])
                                evaluate_loss['RMSE'].append(metrics[2])
                                evaluate_loss['MAE'].append(metrics[1])

                                #multistep forecast
                                blind_results = forecast(model, X_test.copy(), y_test, timesteps,
                                multisteps, features, scaler['Uv Value'], uv_value_col=uv_value_col, blind=True)
                                real_results = forecast(model, X_test.copy(), y_test, timesteps,
                                multisteps, features, scaler['Uv Value'], uv_value_col=uv_value_col, blind=False)
```

Figure 4.16: LSTM tuning process.

LSTM model conception code was produced based in existed researches (F. Fernandes et al., 2020)(B. Fernandes et al., (To appear))

5. Results and discussion

5.1 Ultraviolet Index Prediction as a Classification Problem

The first approach concerns the prediction of the UV index levels, treating it as a classification problem. Further, WHO reference indices were used. Accordingly, the target classes are the designation of each level: “Low”, “Medium”, “High”, and “Very High”.

5.1.1 Decision Trees

The first results registered are the ones from the classification Decision Trees. The results obtained, by scenario, are presented in table 5.1.

Table 5.1: Summary of classification Decision Tree tuning results, regarding UV index prediction.

	Quality measure	Pruning method	Reduced error pruning	Min no. rec. p/ node	E (%)	ACC (%)
Scenario 1	Gini index	No pruning	true	1	12.831	87.169
Scenario 2	Gain Ratio	No pruning	false	2	17.212	82.788
Scenario 3	Gain ratio	No pruning	false	4	20.745	79.255
Scenario 4	**Gini index**	**No pruning**	**true**	**2**	**7.910**	**92.090**
Scenario 5	Gain ratio	No pruning	false	1	9.094	90.094
Scenario 6	Gini index	No pruning	true	2	8.710	91.290

Scenario number one is the one with fewer features. This scenario uses only the “Date”, the “CO Value” and the “SO_2 Value” to train the model. From the parameter tuning process, and after analyzing all the parameter combinations, the best result presents a mean error of 12.831 %.

The second scenario adds the “Temperature” to train the model (making a set of features corresponding to “Date”, “CO Value”, “SO_2 Value”, and the “Temperature”). Further, it is possible to realize that the best combination produces a higher error, when compared with the previous scenario, of 17.212 %.

Scenario number three shows an error of 20.745 %, adding the “Clouds” feature, to train the model. Showing an increase of more than 3% it is concluded that, for this model, the clouds percentage feature (together with the four remaining features) is not a good addition, for the UV index levels prediction.

Scenario four shows a significant error decrease when compared with the previous one. Adding a column to the training process containing the month name (“Month (name)” feature) the error reduced to

7.910 %. So far, this is the best error produced. It is because, as previously analyzed, the UV index has a well-defined monthly behavior, promoting the predictions quality.

The fifth scenario presents an error of 9.094 %. Here was added to the training process the mode of the "Weather description" feature. The error obtained was higher compared with scenario number four (an increase of 1,18 %).

Scenario number six shows a smaller error, a value of 8.710 %, representing a small improvement. Here (when compared with the previous scenario) the mode of the "Weather description" attribute was replaced by this feature one-hot encoded. It shows that, for this model, the preparation of the "Weather description" attribute may result in a smaller error, but not considered a significant improvement.

Analyzing the whole table, it is possible to conclude that the scenario that shows, at this stage, the best performance is scenario number four. Further, note that the optimal pruning method for all the scenarios, is the "No pruning". Here, don't prone the tree produces better results, probably due to the low complexity of the trees that are generated, from each scenario. So, all the branches are considered important and its removal leads to a decrease in the model performance. Therefore, this allows us to assume that there are not redundant splits and, consequently, unnecessary branches in the tree.

5.1.2 Random Forest

The results from the classification Random Forest tuning are presented in the table 5.2.

Table 5.2: Summary of classification Random Forest tuning results, regarding UV index prediction.

	Split criterion	**Tree depth**	**Minimum node**	**Number of Models**	**E (%)**	**ACC (%)**
Scenario 1	InformationGain	13	6	100	20.356	79.644
Scenario 2	InformationGain	10	2	200	16.206	83.794
Scenario 3	InformationGain	24	2	900	17.787	82.213
Scenario 4	**InformationGainRatio**	**14**	**9**	**500**	**8.498**	**91.502**
Scenario 5	InformationGain	24	1	800	8.696	91.304
Scenario 6	InformationGainRatio	20	1	600	11.660	88.340

The first analysis shows that scenario number four and five have very similar results, presenting the best accuracy values registered (higher than 90 %).

Further, is shown that the scenarios with the smallest accuracy values and, consequently, a higher error are scenario number one and number three.

Doing an overall analysis, it can be concluded that using a larger number of features can bring better results. Besides, compared to scenario number five, scenario number six shows a higher error. It helps us to find out that the model produces better results when implemented using only the most frequent weather description in a day, instead of all one-hot encoded descriptions.

5.1.3 Multilayer Perceptron

For MLP prediciton the number of epochs considered ideal was 600 (process explained in appendix D). By evaluating the outcomes for all combinations (through 600 epochs), the remaining optimal parameters were found. The table 5.3 shows the results from the tuning process, by scenario.

Table 5.3: Summary of classification MLP tuning results, regarding UV index prediction.

	Activation function	Hidden Layer	Learn Rate	Neurons	Batch size	E (%)	ACC (%)
Scenario 1	**'relu'**	**3**	**0.001**	**64**	**23**	**6.548**	**93.452**
Scenario 2	'relu'	3	0.005	16	16	8.929	91.071
Scenario 3	'sigmoid'	1	0.005	128	16	9.524	90.476
Scenario 4	'sigmoid'	1	0.005	64	16	10.714	89.286

The results from the table above able us to conclude that the scenario that exhibit the best result is scenario number one. This scenario is the one that reflects the dataset with the fewest features used to train the model.

In addition, it is also concluded that the number of hidden layers is smaller for scenarios with more features (scenario numbers tree and four), this is, a single layer. However, it indicates a higher per-layer number of neurons.

5.1.4 Comparative analysis

After known all the results from the classification models, in the UV index prediction, it is possible to do a comparative analysis of the tree models deployed (Decision Trees and Random Forest), and MLP. Figure 5.1 shows the performance of each classification model regarding each scenario (using as evaluate performance the accuracy).

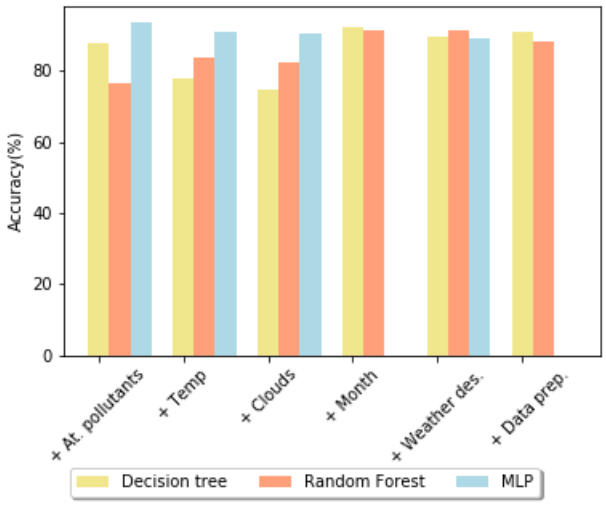


Figure 5.1: Comparative graph, concerning the accuracy for the UV index prediction.

It should be noted that the current graph analysis gathers the results from the above three models (Decision Tree, Random Forest, and MLP mode). The first conclusion is that the model that generally produces better accuracy values is MLP. Nevertheless, the addition of the "Weather description" feature constitutes an exception. Here, this attribute is expressed by the most frequent weather description in a day (the mode), for all the three models over study. It must be noted that the only difference between models is the label encoding process, deployed in the "Weather description" for the MLP model, since this does not accept nominal features as input.

For scenarios constituted by a higher number of features, Random Forest and the Decision Tree models produce similar result. Further, for the available scenario with the fewest features (scenario number one, constituted by the "Date", the "CO value", and the "SO_2 Value") Decision trees show a better outcome (when compared with Random Forest). It may allow us to conclude that, in this context, the Decision Tree models show better results for a small number of features, when compared with Random Forest. Besides, it is important to highlight the scenario that represents the addition of the "Month" feature (scenario number four). Here is shown a marked accuracy improvement, when compared with the previous ones. Finally, regarding this prediction, the best accuracy reached is equal to 93.452 %, using the MLP model and the features from scenario number one to train the model.

5.2 Ultraviolet Index Prediction as a Regression Problem

The second approach was to predict the UV index, treating it as a regression problem. Here, the target attribute is the numeric UV Index, the "UV Value" feature.

5.2.1 Decision Tree

Table 5.4 shows the best results obtained for each scenario, resulting from the tuning process.

Table 5.4: Summary of regression Decision Trees tuning results, regarding UV index prediction.

	Missing value handling	**Tree depth**	**Minimum node size**	**RMSE**
Scenario 1	XGBoost	18	4	0.09960
Scenario 2	XGBoost	11	4	0.12724
Scenario 3	Surrogate	8	6	0.13645
Scenario 4	**XGBoost**	**6**	**5**	**0.05167**
Scenario 5	XGBoost	10	14	0.05450
Scenario 6	XGBoost	6	4	0.05292

Concerning the model prediction error, in units of the target feature, MAE and RMSE were used as evaluation metrics. As a regression problem, the error above was computed using normalized values. Consequently, these outcomes do not represent the reality (once are not denormalized) and only enable a comparative analysis. Even so, the smallest values still represent the best results.

Considering the results, and as the main conclusion, scenario four is the one that present the smaller RMSE. However, scenario number five and six presents very similar error values when compared to this last. Further, the scenarios with higher error values are the number two an number three.

Deploying a Decision Tree regression model for the best combination of parameters (for each scenario), the results from table 5.5 arose. This table shows the RMSE and MAE resultant from this process. Here, the values were denormalized, which means that each value represents the correct error, in the same unit as the target value, the UV index.

Table 5.5: Regression Decision Tree results, regarding UV index prediction.

	MAE	**RMSE**
Scenario 1	0.58864	1.07349
Scenario 2	0.78879	1.24379
Scenario 3	0.90122	1.30233
Scenario 4	0.37320	0.51738
Scenario 5	0.43311	0.58212
Scenario 6	**0.36350**	**0.51335**

As previously mentioned, the scenarios number four, five, and six present very similar error values. Therefore, when the model is deployed with the best parameters (for each scenario) it shows better results for scenario number six. This small variation is due to the randomness of the model, mainly due to the cross-validation (creating small differences once there are small variations in the train and test datasets split). Figure 5.2 shows the graph representation, for each scenario, of the predicted and the actual values.

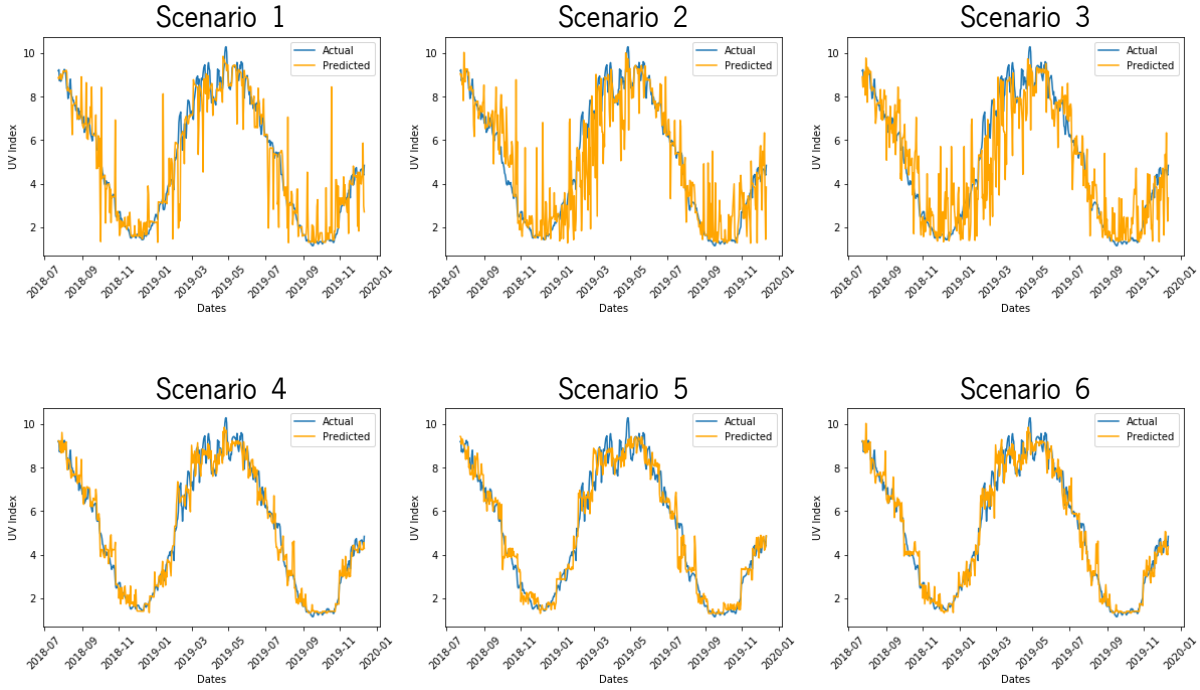


Figure 5.2: Graphs of regression Decision Tree predictions, concerning the UV index prediction.

Analyzing the graphs it is possible to reinforce the conclusions previously drawn. Scenario number one, two, and three represent the higher difference between the actual and the predicted value. However, scenario four, five, and six present predicted values more fitted to the actual ones.

The main conclusions go within the fact that the introduction of the variable "Month (name)" in the dataset causes a significant improvement in the model. Thus, is understood that, for this type of model and problem, the month name (extracted from the date), used as a feature, improves the model, since all the scenarios in which it is present (scenario number four, five, and six), show a better performance.

5.2.2 Random Forest

As well as the regression Decision Trees, the regression Random Forest models were tuned. The table 5.6 shows the results from this process.

Table 5.6: Summary of regression Random Forest tuning results, regarding UV index prediction.

	Tree depth	**Minimum child node size**	**Number of Models**	**MSE**
Scenario 1	3	1	1000	0.00956
Scenario 2	8	1	200	0.01000
Scenario 3	3	1	500	0.01135
Scenario 4	**11**	**1**	**1000**	**0.00311**
Scenario 5	2	1	1100	0.05209
Scenario 6	13	1	900	0.00910

Analyzing the table above it can be concluded that the scenario presenting the best results, without ambiguity, is scenario number four. After knowing the best parameters for each scenario, the model was deployed using them. Thus, allowing us to understand the error values in the correct units of measurement, the results are shown in table 5.7.

Table 5.7: Regression Random Forest results, regarding UV index prediction.

	MAE	**RMSE**
Scenario 1	1.438182	1.758826
Scenario 2	0.739012	1.044964
Scenario 3	1.128045	1.457048
Scenario 4	**0.410235**	**0.5384606**
Scenario 5	1.597893	1.839926
Scenario 6	0.703682	0.885547

Scenario number four shows the best result for both error metrics. Concerning the MAE and the RMSE, the scenario with the features, "Date", "CO Value", "SO_2 Value", "Temperature", "Clouds", and the "Month" is the one showing the best result (secnario number four).

The graphs in figure 5.3 allow emphasize the conclusions obtained and depicts the difference between the predicted and actual UV Index values.

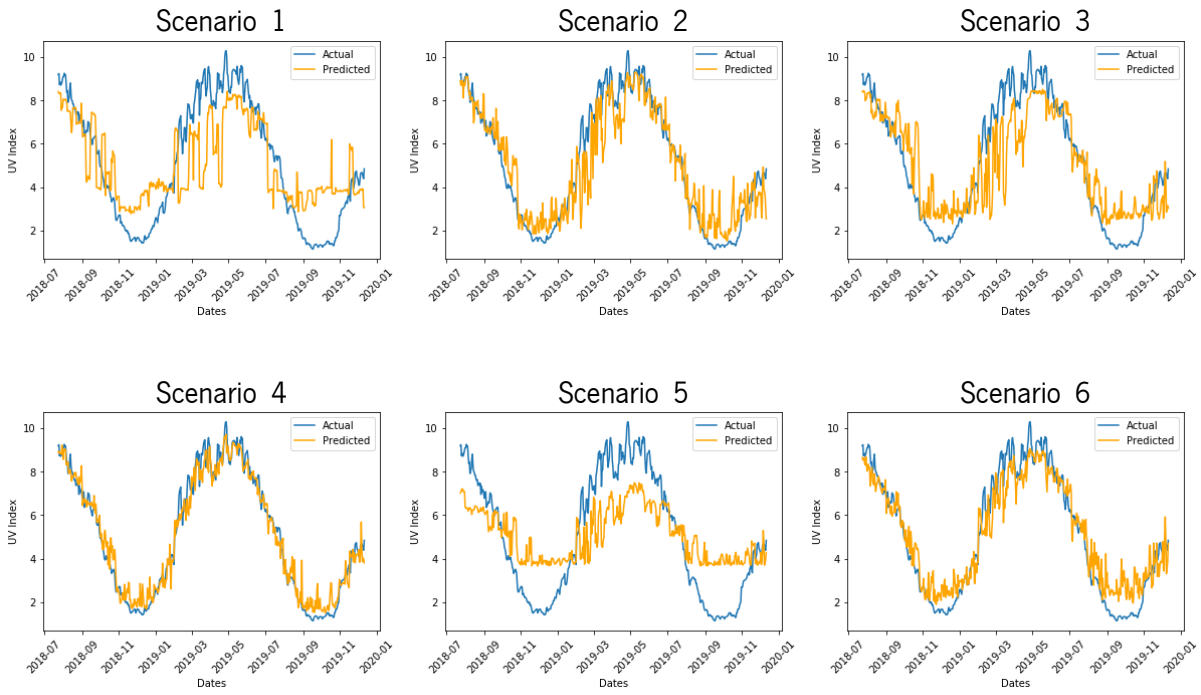


Figure 5.3: Graphs of regression Random Forest predictions, concerning the UV index prediction.

.

Analyzing the graphs, scenario number four presents the most similar values between the actual and predicted values. The scenario number five presents the scenario with the worst results, showing, consequently, the higher gap between the actual UV index values and the predicted ones. Note that, Random Forest model, in this particular instance, predicts above for higher index values, and below for lower index values.

5.2.3 Multilayer Perceptron

In this instance, the model used is the MLP. The first parameter found was the ideal number of epochs, to train the model. After the iterative process explained in subsection 4.3.1.2, the number of epochs considered ideal (regarding the model and this particular problem) was 200 (explained in the appendix E). Additionally, the remaining parameters were found through the analysis of the tuning process results. Thus, the best results, for each scenario, are presented in table 5.8.

Table 5.8: Summary of regression MLP tuning results, regarding UV index prediction.

	Activation function	Hidden Layer	Learn Rate	Neurons	Batch size	MAE
Scenario 1	'relu'	1	0.001	16	16	-0.417235
Scenario 2	'relu'	1	0.01	16	16	-0.417235
Scenario 3	'relu'	1	0.001	128	23	-0.417235
Scenario 4	'relu'	2	0.001	32	16	-0.417235

Presenting equal MAE values for each scenario, the best parameters combination were found. Deploying the model using the optimal parameters the results from table 5.9 have been achieved.

Table 5.9: Regression MLP results, regarding UV index prediction.

	MAE	**RMSE**
Scenario 1	0.490354	0.600022
Scenario 2	0.514139	0.630862
Scenario 3	0.520870	0.558326
Scenario 4	0.452455	0.578217

As previously demonstrated, the results obtained are very similar. It is not correct to say that a specific scenario shows better results than others. Thereby, use a dataset with more or fewer features will generate similar results, in this particular prediction. The graphs from the figure 5.4 allow understand these conclusions.

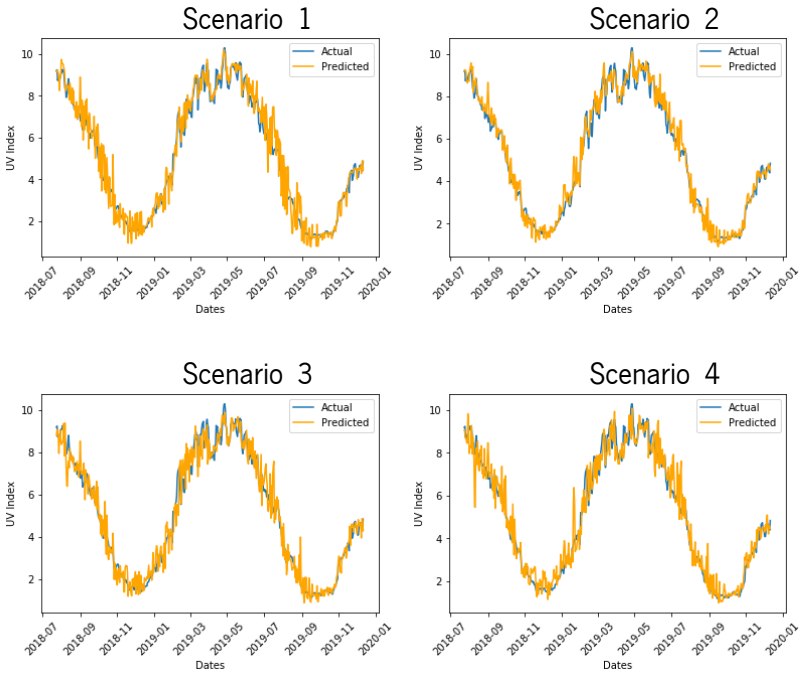


Figure 5.4: Graphs of MLP predictions, concerning the UV index prediction.

All the graphs, representing each scenario, show both lines (blue and orange) quite fitted, which means that the model predicts with a considered small error when compared with other models.

5.2.4 Comparative analysis

Once all the results of the UV forecast were assembled using regression models (Decision Tree, Random Forest, and MLP) it is crucial to do a comparative analysis to understand which model has a better

performance for a given dataset/scenario. The graphs from the figure 5.5 and 5.6 shows the MAE and RMSE for each model, regarding the scenarios under study.

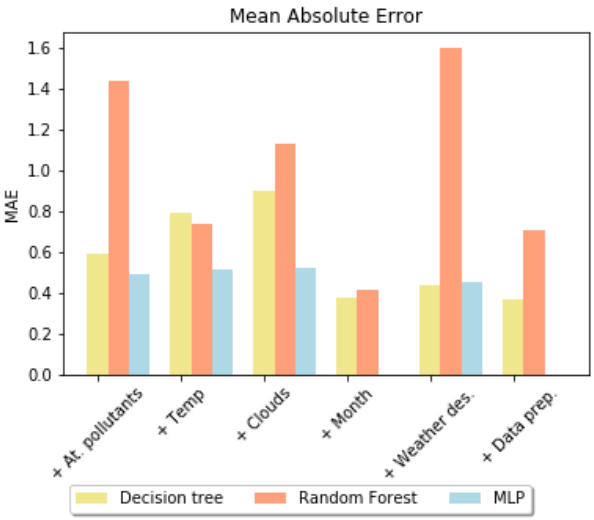


Figure 5.5: MAE comparative graph (UV prediction).

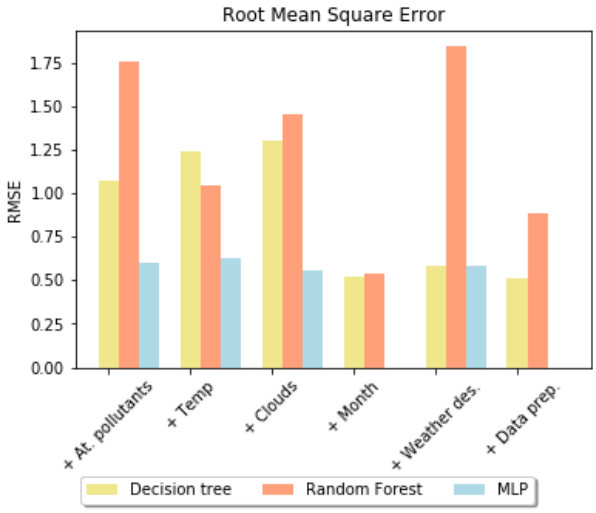


Figure 5.6: RMSE comparative graph (UV prediction).

Doing an overall analysis, it is clear that the model that generally presents the worst results is the Random Forest. It is followed by the Decision Trees, which presents better results, mainly (and with a noticeable difference) in the first set of data (using the "Date", the "CO value", and the "SO_2 value") and in the fifth one (using the "Date", the "CO value", the "SO_2 value', the "Temperature", the "Clouds" and the "Weather description" as main features to train the model). Further, for the tree-based models, the month name adding shows an increase in these models' performance (mainly for the Random Forest). Furthermore, the MLP is the model that, in a general way, produces better results.

Concluding, the best result obtained was an MAE of approximately 0.36, resulting from the Decision Tree prediction (scenario number six).

5.3 Ultraviolet Index prediction as a Time Series problem

Using the LSTM model, based on a time series forecast, the UV index was predicted. At this research, the number of days to forecast the UV Index was three days, which means, a multistep value equal to 3.

The first parameter tuned was the number of epochs. This value was found using the iterative process, resulting in 150 epochs (this process is presented in the appendix F, more detailed). Thus, the table 5.10 shows the remaining tuning parameters result, using blind MAE as the evaluated metric. This metric is used once, as mentioned before, the blind forecast is the one that should be used to evaluate the model, once allow further predictions and, uses the predicted values to predict the next ones. So, there are indicated the three combinations that produced the better results, and the three that generated the worst, respectively.

Table 5.10: Summary of LSTM tuning regarding UV index forecast.

Batch size	Timesteps	Hidden Layers	Neurons	Activation	Learn Rate	Blind MAE	Blind RMSE
16	21	1	16	tanh	0.01	0.150523	0.045132
30	21	1	16	relu	0.001	0.155112	0.079369
16	21	2	32	tanh	0.005	0.158052	0.045252
23	21	3	128	tanh	0.005	2.913609	11.051768
23	21	2	128	sigmoid	0.01	2.858105	11.265930
16	21	1	128	tanh	0.01	2.559773	14.616806

The following table, 5.11, shows the results from the known forecast (the forecast process using only the known values to forecast the next ones, instead of the predictions) for the optimal parameter combination.

Table 5.11: Know results from the LSTM tuning best parameters combination (UV index prediction).

MAE known result	RMSE known result
0.133698	0.039279

Using the first combination of parameters from the table 5.10 (the optimal one), was predicted two sets of three days, randomly chosen in the entire dataset. The days chosen were three in 2019 (2019-09-11, 2019-09-12, and 2019-09-13) and the remaining ones in 2020 (2019-09-11, 2019-09-12, and 2019-09-13) . Table 5.12 shows, for that dates, the actual values, and the respective blind and known forecast.

Table 5.12: Blind and known predictions examples, using the LSTM model (UV index prediction).

Date	UV index	Blind Forecast	Known Forecast
2019-09-11	6.883636	6.549842	6.549842
2019-09-12	6.4733334	6.2305017	6.4334965
2019-09-13	6.2633333	5.9936047	6.200255
2020-03-20	4.605	4.7148314	4.7148314
2020-03-21	4.465	4.737669	4.6896653
2020-03-22	4.386667	4.756311	4.6133194

Besides, each set of data predictions produced a mean MAE and RMSE values presented in the table 5.13.

Table 5.13: Evaluation of blind and known predictions example (LSTM UV index prediction).

Date	Blind MAE	Blind RMSE	Known MAE	Known RMSE
2019-09-11 2019-09-12 2019-09-13	0.28211817	0.08104643	0.1455698	0.038994793
2020-03-20 2020-03-21 2020-03-22	0.25071478	0.074349344	0.1870497	0.03796959

Further, the figure 5.7 depicts the resulted predictions, using this model.

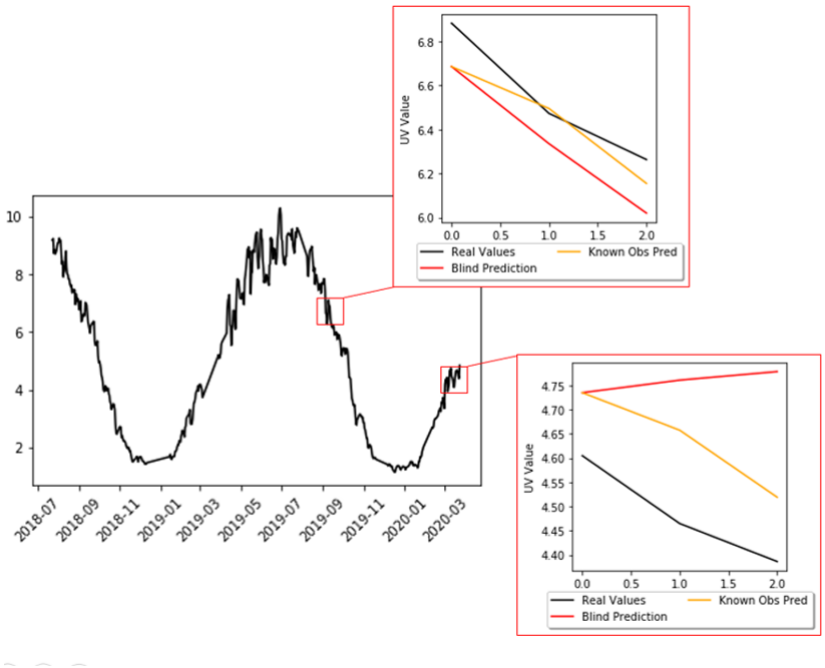


Figure 5.7: LSTM prediction examples, regarding blind and known UV index forecast.

The predictions presented above show always better results for the known forecast when compared with the blind results. Even so, it is essential to look at these last results as being correct in the evaluation of the model, for the reasons previously expressed.

5.4 Carbon Monoxide air concentration prediction as a Time Series problem

As previously referred, one of the main air pollutants is CO. Thus, this pollutant was predicted using a time series forecast model, the LSTM model. This single prediction, concerning CO, occurs due to a lack of data in the available datasets, that would allow a logical prediction about air pollutants That is, features referring to the city's industries, as well as other pollutants. Consequently, the model was tuned using the

CO data, as a uni-variate problem (using only the "CO Value" feature). In table 5.14 are presented the results obtained from this process, showing the three best and worst parameters combinations.

Table 5.14: Summary of LSTM tuning regarding CO forecast.

Batch size	Timesteps	Hidden Layers	Neurons	Activation	Learn Rate	Blind MAE	Blind RMSE
16	21	1	16	relu	0.005	1.345×10^{-7}	6.871×10^{-8}
23	21	2	128	tanh	0.005	1.348×10^{-7}	6.872×10^{-8}
23	21	2	128	sigmoid	0.005	1.570×10^{-7}	4.188×10^{-8}
23	21	3	16	relu	0.005	1.380×10^{-6}	3.212×10^{-6}
30	21	2	128	tanh	0.01	1.380×10^{-6}	2.953×10^{-6}
23	14	3	64	tanh	0.01	1.316×10^{-6}	1.934×10^{-6}

Besides, the metrics from the known forecast were also computed, showing, for the optima parameters combination, the values from the table 5.15.

Table 5.15: Know results from the LSTM tuning (CO prediction).

MAE known result	RMSE known result
1.16074×10^{-7}	5.94929×10^{-8}

Using these tuning optimal parameter, the model was deployed. For that, were used the same days as the previous LSTM prediction, the UV index prediction. Thus, table 5.16 shows the actual values, and the respective forecasts.

Table 5.16: Blind and known predictions examples, using the LSTM model (CO prediction).

Date	CO Value	Blind Forecast	Known Forecast
2019-09-11	3.1364316×10^{-6}	3.2964565×10^{-6}	3.2964565×10^{-6}
2019-09-12	3.4606148×10^{-6}	3.2088833×10^{-6}	3.0602096×10^{-6}
2019-09-13	3.4606148×10^{-6}	3.1101320×10^{-6}	3.3413589×10^{-6}
2020-03-20	4.3996315×10^{-6}	4.0611071×10^{-6}	4.0611071×10^{-6}
2020-03-21	4.6099562×10^{-6}	3.8132159×10^{-6}	4.1562903×10^{-6}
2020-03-22	5.2366045×10^{-6}	3.6389376×10^{-6}	4.4162643×10^{-6}

Further, this set of predictions result in a mean MAE and RMSE value presented in table 5.17.

Table 5.17: Evaluation of blind and known predictions example (LSTM CO prediction).

Date	Blind MAE	Blind RMSE	Known MAE	Known RMSE
2019-09-11 2019-09-12 2019-09-13	2.5407977×10^{-7}	7.0605×10^{-14}	2.2656202×10^{-7}	6.671811×10^{-14}
2020-03-20 2020-03-21 2020-03-22	9.109772×10^{-7}	$1.1006444 \times 10^{-12}$	5.375102×10^{-7}	$3.3112315 \times 10^{-13}$

Additionally, the figure 5.8 depicts the the predictions made, using this model, regarding the CO prediction.

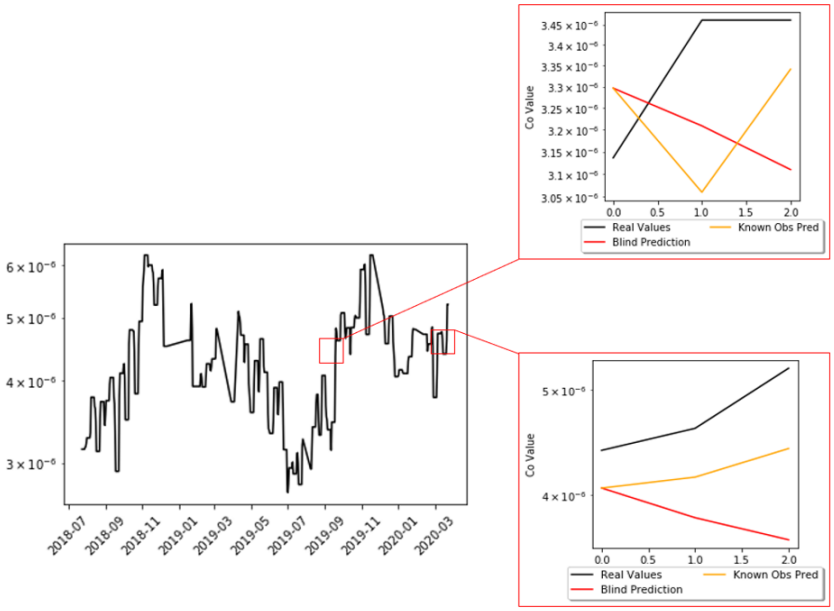


Figure 5.8: LSTM prediction examples, regarding blind and known CO forecast.

We can conclude, from these practical examples, that a small RMSE and also a higher MAE value are generated, when compared to the tuning error. Even so, the MAE value is still maintained, at around $\times 10^{-7}$. On the other hand, the RMSE value shows smaller values, since the RMSE, by nature, help to highlight higher error values.

5.5 Water pH prediction

Aiming for meaningful results, concerning the WWTP water problem, the water pH prediction was made. The following tables show the tuning and final results, using Decision Trees, Random Forest, and MLP models. Here, and since pH is a numeric target variable, only regression models were used.

5.5.1 Decision Trees

The first model deployed, in respect of this problem, was the Decision Tree model. Table 5.18 shows the tuning process results , for each scenario.

Table 5.18: Summary of regression Decision Tree tuning results, regarding water pH prediction.

	Missing value handling	**Tree depth**	**Minimum node size**	**RMSE**
Scenario 1	XGBoost	5	9	0.04962
Scenario 2	Surrogate	5	10	0.07465
Scenario 3	XGBoost	2	4	0.08023
Scenario 4	**Surrogate**	**5**	**5**	**0.04887**

After the tuning process, the real results (computed with the denormalized data) are presented in the table 5.19.

Table 5.19: Regression Decision Trees results, regarding water pH prediction.

	MAE	**RMSE**
Scenario 1	0.10646	0.16259
Scenario 2	0.18416	0.24765
Scenario 3	0.19487	0.26201
Scenario 4	**0.10586**	**0.16155**

The table shows that the scenario number four presents the best results, exhibiting similar results for scenario number one. Besides, the worst scenario is scenario number three. Along with the table above, the figure 5.9 graphically represents the conclusions drawn.

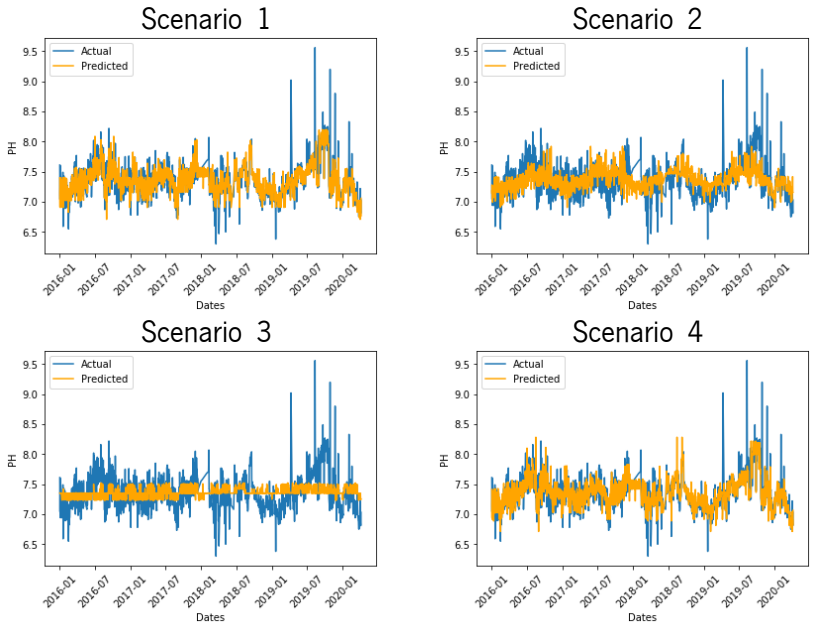


Figure 5.9: Regression Decision Tree predictions, concerning the water pH prediction.

First, by examining the graphs for each scenario, it is possible to visualize that, for scenario number three, the model produce prediction values very close to each other. It reflects the higher error, since the actual pH values have more pronounced variations. Moreover, as the scenario that produce worst outcomes, this make only use of the DO features (at the anoxic and aerated zone) to train the model (beyond the year, month, day, and hour, common feature to all the scenarios).

The best results come from scenario number four and number one. These scenarios are the ones with the "Temperature" (in tertiary treatment) and the "pH-secondary clarifier" (line 2) in its constitution. Thus, these features are demonstrated as relevant to predict the pH value, once all the scenarios of which they are part together (one and four) show better results.

5.5.2 Random Forest

As mentioned before, the second model implemented, for water pH problem, was the Random Forest. The tuning process results presented in the table 5.20 shows an early notion of the best scenarios, and consequently, which features to use to train the model.

Table 5.20: Summary of regression Random Forest tuning results, regarding water pH prediction.

	Tree depth	Minimum child node size	Number of Models	RMSE
Scenario 1	**4**	**8**	**500**	**0.047459**
Scenario 2	3	4	500	0.073146
Scenario 3	7	5	1100	0.080355
Scenario 4	3	5	100	0.048461

The results of the deployment process are shown in the table 5.21, enabling the denormalized errors to be understood.

Table 5.21: Regression Random Forest results, regarding water pH prediction.

	MAE	RMSE
Scenario 1	**0.11375**	**0.169247**
Scenario 2	0.18043	0.245012
Scenario 3	0.19329	0.26060
Scenario 4	0.12052	0.17760

After interpreting the table, it is possible to assume that the scenario with the best results is the number one, also, with similar ones for scenario number four. Using the graphs from figure 5.10, it is possible to reinforce the analysis.

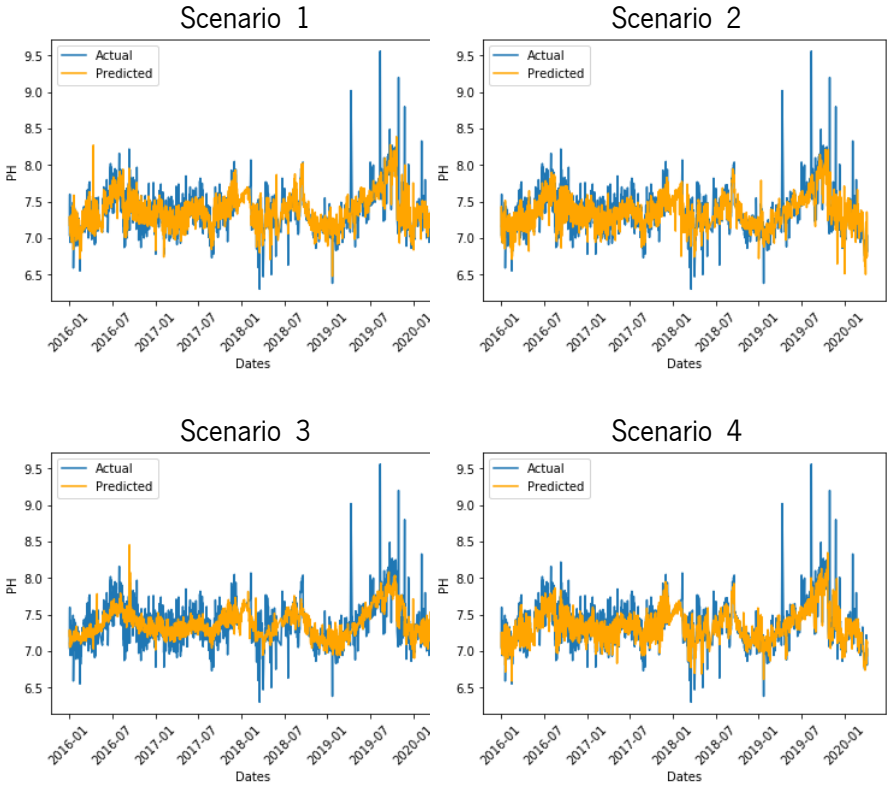


Figure 5.10: Regression Random Forest predictions, concerning the water pH prediction.

Compared to the Decision Tree results (before exhibited), the general conclusions regarding this model's predictions (regarding this waste water pH problem) are quite similar. So, the scenarios showing the best results are the ones with the "Temperature" (from tertiary treatment) and the "pH" (from the secondary clarifier) in its constitution (the scenario one and three).

Further, the worst results come from scenario number two, which use the DO features (at the anoxic and aerated zone), plus the pH in the secondary clarifier. Furthermore, the scenario that includes only DO values in its constitution (scenario number three) does not present the worst outcomes, however presents a bad performance.

5.5.3 Multilayer Perceptron

The last model implemented, concerning the water pH prediction was the MLP. Here, the tuning was made to find the optimal combination of parameters. As previously mentioned, the first step for this model was to find the best number of epochs to train the model. Using the iterative process presented in subsection 4.3.2.1, the number of epochs established was 200 (described in the appendix H). Besides that, the remaining parameters were tuned, resulting in the combinations presented in the table 5.22.

Table 5.22: Summary of regression MLP tuning results, regarding water pH prediction.

	Activation function	Hidden Layer	Learn Rate	Neurons	Batch size	MAE
Scenario 1	'relu'	3	0.001	16	16	-0.269315
Scenario 2	'relu'	2	0.001	64	16	-0.269315
Scenario 3	'relu'	1	0.001	16	16	-0.269320
Scenario 4	**'relu'**	**3**	**0.005**	**64**	**16**	**-0.269315**

Further, the model was deployed originating denormalized results, presented in the table 5.23.

Table 5.23: Regression MLP results, regarding water pH prediction.

	MAE	RMSE
Scenario 1	0.11859	0.17325
Scenario 2	0.14560	0.20719
Scenario 3	0.15402	0.21356
Scenario 4	**0.11834**	**0.17148**

The table shows that scenario number four present the better results, but with a high similarity when compared with MAE and RMSE values for scenario number one. However, the graphs from figure 5.11 allow a more careful analysis.

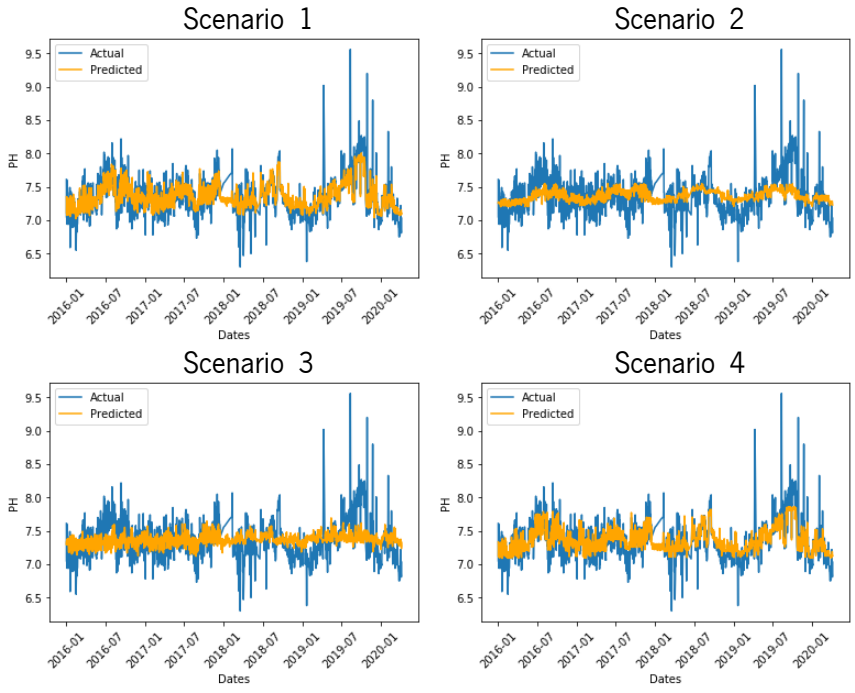


Figure 5.11: Regression MLP predictions, concerning the water pH prediction.

When compared with the two previous models, this one shows better results. Moreover, the main conclusions concerning a scenarios comparative analysis are very similar, when compared with these two models (Decision Tree and Random Forest), this is, scenario number one and four show the best results.

5.5.4 Comparative analysis

After analyzing the results, is now possible to do a comparative analysis for the pH water prediction. Here, the scenarios used to train the model are the same among models (except for the date attribute, explained before). Figure 5.12 and 5.13 depict the graphs comparing the models' predictions, evaluated by the MAE and RMSE, respectively.

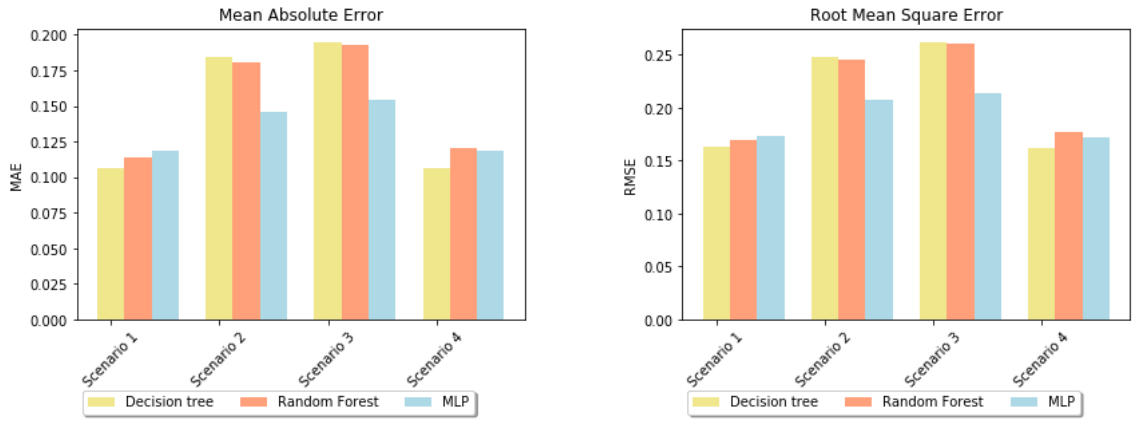


Figure 5.12: MAE comparative graph (pH prediction).Figure 5.13: RMSE comparative graph (pH prediction).

The first conclusion arising from the graph analysis is that, for this instance, Decision Trees and Random Forest produce very similar results. Further, for scenario number one, the prediction error is smaller, and similar between models. However, it is possible to figure that MLP slightly presents the worst predictive model for this set of features (DO features, the "pH-Secondary clarifier" and the "Temperature"). Moreover, the scenario number two show similar results for the Decision Tree and Random Forest prediction. At the same time, contrary to what was verified in the previous scenario, the MLP model presents the best result (the smallest error values) concerning this scenario (using the DO features and the pH value (from the secondary clarifier)). Scenario number three shows also the worst results for the tree-based models and the smallest error to the MLP predictions. Scenario number four shows a similar error between the Random Forest and MLP predictions. Here is pointed the Decision Trees as the model that produces more quality predictions (containing the pH value and the temperature, from the tertiary treatment, as features). In addition, it is possible to conclude that, using all the data available (the pH value, the DO features, and the temperature, scenario number one), the tree models predict quite well. The same happens in scenario number 4. However, using only the DO combined with pH value (scenario two), or with temperature (scenario number three), all the three models predictions reach the higher values. Finally, an approximate MAE value of 0.11 is achieved, as the best prediction error. This error can be reached using the three applied

models (the Decision Trees, the Random Forest, and the MLP) once they are all able to predict with this approximate MAE values.

6. Conclusion and Future Work

One of the key challenges nowadays is environmental sustainability. With an growing population, many of the planet's negative impacts have been verified. With a high rate of anthropogenic emissions, it is essential to give special attention to this topic once future generations may be compromised.

This research addresses a supervised learning approach to predict several parameters concerning environmental sustainability. Regarding the air and water quality, these parameters were predicted using four different models.

Initially, a primary investigation has been done, exposing some crucial concepts to this research. First, it was understood which may impact the target features. Then, concerning the air quality, was grasped information about the main factors that influence the concentration of the atmospheric pollutant, as well as the UV index. Further, the reference values were found, according to the established by specialized agencies, allowing a better data analysis. Concerning the water quality issues, some studies about WWTP were conducted. Here, was displayed how a WWTP works and some important parameters that, after perceived, may induce a better data analysis. Moreover, in this initial phase, all implemented supervised models are explained and explored, showing how they operate as well as some advantages and limitations that these may present.

The second part of this work contemplates a more practical approach. This part starts with an exhaustive analysis of both datasets. Here, some inferences are made to understand the best data preparation to achieve, as well as which are the best features to use to train the model. Besides that, the models were constructed and tuned to achieve the best predictions, regarding the available data sets.

The first models constructed and implemented were the tree-based models, the Decision Trees, and Random Forest. Then, more complex models were applied, using deep learning models, the MLP, and the LSTM.

Concerning the air quality and atmospheric pollution, the parameters predicted were the "UV Value" (using classification and regression models) and the CO air concentration ("CO Value" attribute). From the data analysis respecting this problem, the main conclusions denote that the pollutants analyzed show small values according to the reference values, and regarding the CO and the SO_2, the classification of the air quality is, according to PAE defined levels, "very good". Further, looking at the UV index, the available data does not show "extreme" values registered (+11), although, present "very high" levels, this ones requiring special attention. Further, it was possible to conclude that, by analyzing the correlation between features, there is a notorious negative correlation between the UV index and the CO value. Besides, it

shows a similar but positive correlation between the UV index and the temperature.

Hence, in respect of the classification UV index prediction, the main conclusions are several. First, the model that presents better predictions is the MLP, showing a higher accuracy for all the scenarios. Second, using tree-based models, introducing the month name into the training dataset (as a new feature), results in a significant improvement of the model performance. Further, were found the set of features that generate the best accuracy for each model. Accordingly, considering the Decision Trees, the scenario that shows the better performance is the one that contemplates the features: "Date", "CO Value", "SO_2 Value", "Temperature","Clouds", and "Month (name)" (scenario number four). Moreover, the MLP model shows the best results for scenario number one, the scenario with the fewest features. This scenario holds the "Day", the "Month", and "Year" besides the "CO Value", and the "SO_2 Value". Last, the result with the smallest error is produced by the MLP model, achieving an error of approximately 93%.

With regard to the UV index prediction as a regression problem there are several conclusions. First, the Random Forest, in general, is the model that shows the worst results, with the MLP model showing the best ones. Second, for the Decision Trees, introducing the month name produces also a significant improvement. Accordingly, to predict using the Decision Trees, the better set of features to train the model are: "Date", "CO value", "SO_2 Value", "Temperature", "clouds", "Month(name)", and the "weather description" one-hot encoded (Scenario 6). Further, using the Random Forest, the scenario that presents the smaller errors is scenario number four (using "Date", "CO Value", "SO_2 Value", "Temperature", "Clouds", and "Month (name)" as features). Finally, the MLP predictions do not show quite a difference error between scenarios. It may allow us to conclude that the set of features is not so crucial to predict the UV index by a regression MLP model, in this context. Finally, regarding this parameter prediction, is achieving, as the best MAE, an approximately of 0.36 (produced by the Decision Tree model).

About the UV prediction as a time series forecast, using the LSTM model, there are some significant conclusions. This model requires higher computational efforts (when compared with others). Moreover, the tuning process takes time, requiring time for the construction and implementation of this model. However, this model predicts, on average, with good results regarding this problem, reaching small errors (on average an MAE value of 0.150523).

Concerning the prediction of CO using the LSTM model (as a time series forecast), the conclusions are quite similar, when compared with the UV prediction using this model. Thus, it predicts with small errors, and presents, therefore, a good performance. In this specific prediction, achieves an error of approximately 1.3×10^{-7}.

In respect of the water pH problem in a WWTP context, there are numerous conclusions. Concerning the data exploration, it can be concluded, after this research, that this process is crucial in the WWTP processes. Accordingly, there are legally stipulated values to fulfill, and some issues to the population and ecosystems that can be avoided beforehand, without even predict the parameters, just analyzing them. Besides, this research shows a high correlation between the pH values among the WWTP units. So, according to the above-mentioned concerns, the prediction target parameter is the water pH, in the moment before it returns to the hydric sources. Hence, were used three models also: the Decision Tree, Random

Forest, and MLP. From the water pH prediction aroused some conclusions. First, the model that shows better predictions, in general, is the MLP. In respect of Decision Trees predictions, scenario number one and scenario number four show the smallest error registered (very similar results). The scenario number one is the one that assembles as features the "Date", "pH- secondary clarifier", "DO- Aerated zone (p)", "DO- Anoxic zone (p)", and the "Temperature" (from tertiary treatment). Besides, the scenario number four gather the "Date", "pH- secondary clarifier", and the "Temperature". Further, Random Forest predicts with the smaller error using also the features from scenario number one. Furthermore, the MLP model also showed a high results similarity between scenario number one and number four, these being the ones that register the lowest errors. This prediction shows (for all the models) that using only the DO and the date (to train the model) the resulted predictions are the worst registered. Last, this water pH prediction achieves a better error of approximately 0.11, with all three models showing the ability to achieve it.

The biggest challenges of this research are mainly based on the models' tuning process. This process requires a lot of time and a high computational capacity (mainly deep learning models). However, some of these limitations were overcome by the use of google colab, which allowed to run models much faster. Still, since there are several parameters and scenarios, it takes considerable time to obtain reliable results.

Finally, it is possible to conclude that this research may help to anticipate problematic situations, not only through the forecasting process, but also through data analysis. Besides, it is possible to optimize the resources used, react faster, and, consequently, provide the best to the population and responsible entities. This research allows creating a bridge between ML field and environmental sustainability, to take advantage of the data available (there are several sustainability parameters easy to obtain) and the existing ML models.

Thus, as future work emerge:

- Regarding the air quality and the forecast process, use more data to train the models, in particular, to forecast the CO air concentration. Taking advantage of more meaningful data, would be possible to predict using the Decision Tree, Random Forest, and MLP for this attribute and test the model's performance;
- Concerning the WWTP, understand the impact of the weather on the wastewater characteristics. Besides, regarding this, predict more meaningful parameters;
- Test more supervised models, and understand the results, making the trade-off between the difficulty of application and results obtained;
- Create dashboards to display the forecast results, allowing an easier visualization and consequent decision-making process.

Bibliography

Abdel-Shafy, H. I., & Mansour, M. S. (2018). Solid waste issue: Sources, composition, disposal, recycling, and valorization. *Egyptian Journal of Petroleum*, *27*(4), 1275–1290. https://doi.org/https://doi.org/10.1016/j.ejpe.2018.07.003

Allaart, M., van Weele, M., Fortuin, P., & Kelder, H. (2004). An empirical model to predict the UV-index based on solar zenith angles and total ozone. *Meteorological Applications*, *11*, 59–65.

Analide, C., Novais, P., Machado, J., & Neves, J. (2006). Quality of Knowledge in Virtual Entities. *Encyclopedia of Communities of Practice in Information and Knowledge Management, Elayne Coakes and Steve Clarke (Eds), Idea Group Reference*, 436–442. https://doi.org/10.1093/jigpal/jzq029

Ayyadevara, V. K. (2018). *Pro Machine Learning Algorithms*. https://doi.org/10.1007/978-1-4842-3564-5

Bayo, J., López-Castellanos, J., & Puerta, J. (2016). Operational and environmental conditions for efficient biological nutrient removal in an urban wastewater treatment plant [cited By 5]. *Clean - Soil, Air, Water*, *44*(9), 1123–1130. https://doi.org/10.1002/clen.201500972

B.Baird, R., D.Eaton, A., & W.Rice, E. (Eds.). (2017). *Standard Methods for the Examination of Water and Wastewater* (23rd). Amer Public Health Assn. https://doi.org/10.2105/SMWW.2882.193

Bekkerman, R., Bilenko, M., & Langford, J. (2011). *Scaling up machine learning*. https://doi.org/10.1145/2107736.2107740

Block, M. L., Elder, A., Auten, R. L., Bilbo, S. D., Chen, H., Chen, J. C., Cory-Slechta, D. A., Costa, D., Diaz-Sanchez, D., Dorman, D. C., Gold, D. R., Gray, K., Jeng, H. A., Kaufman, J. D., Kleinman, M. T., Kirshner, A., Lawler, C., Miller, D. S., Nadadur, S. S., ... Wright, R. J. (2012). The outdoor air pollution and brain health workshop. *NeuroToxicology*, *33*(5), 972–984. https://doi.org/10.1016/j.neuro.2012.08.014

Brownlee, J. (2018). Deep Learning for Time Series. *Machine Learning Mastery*.

Carneiro, D., Novais, P., Miguel Pêgo, J., Sousa, N., & Neves, J. (2015). Using mouse dynamics to assess stress during online exams. *Hybrid Artificial Intelligent Systems, Henrique Onieva et al. (eds), Springer - Lecture Notes in Computer Science*, *9121*, 345–356. https://doi.org/http://dx.doi.org/10.1007/978-3-319-19644-2_29

Carneiro, J., Saraiva, P., Martinho, D., Marreiros, G., & Novais, P. (2018). Representing decision-makers using styles of behavior: An approach designed for group decision support systems. *Cognitive*

Systems Research, *47*, 109–132. https://doi.org/https://doi.org/10.1016/j.cogsys.2017.09.002

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Rudiger, W. (2000). *Crisp-Dm 1.0*.

Cheremisinof, N. P. (2002). *Handbook of Air Pollution Prevention and Control*.

Cohen, A. J., Brauer, M., Burnett, R., Anderson, H. R., Frostad, J., Estep, K., Balakrishnan, K., Brunekreef, B., Dandona, L., Dandona, R., Feigin, V., Freedman, G., Hubbell, B., Jobling, A., Kan, H., Knibbs, L., Liu, Y., Martin, R., Morawska, L., ... Forouzanfar, M. H. (2017). Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. *The Lancet*, *389*(10082), 1907–1918. https://doi.org/10.1016/S0140-6736(17)30505-6

Colacicco, A., & Zacchei, E. (2020). Optimization of energy consumptions of oxidation tanks in urban wastewater treatment plants with solar photovoltaic systems. *Journal of Environmental Management*, *276*(August), 111353. https://doi.org/10.1016/j.jenvman.2020.111353

Downs, N., Butler, H., & Parisi, A. (2016). Solar ultraviolet attenuation during the australian (red dawn) dust event of 23 September 2009. *Bulletin of the American Meteorological Society*, *97*(11), 2039–2050. https://doi.org/10.1175/BAMS-D-15-00053.1

Fausett, L. V. (1994). *Fundamentals of Neural Networks Architectures, Algorithms and Applications*. https://doi.org/10.1017/CBO9781107415324.004

Fenger, J. (1999). Urban air quality. *Atmospheric Environment*, *33*, 4877–4900. https://doi.org/10.1016/j.atmosenv.2007.09.035

Fernandes, B., Silva, F., Alaiz-Moretón, H., Novais, P., Neves, J., & Analide, C. ((To appear)). Long short-term memory networks for traffic flow forecasting: Exploring input variables, time frames and multi-step approaches. *INFORMATICA*.

Fernandes, F., Silva, F., Alaiz-Moretón, H., Novais, P., Analide, C., & Neves, J. (2020). Traffic flow forecasting on data-scarce environments using arima and lstm networks. *WorldCIST, Advances in Intelligent Systems and Computing*, *930*, 273–282. https://doi.org/10.1007/978-3-030-16181-1_26

Gao, W., & Su, C. (2020). Analysis on block chain financial transaction under artificial neural network of deep learning. *Journal of Computational and Applied Mathematics*, *380*, 112991. https://doi.org/10.1016/j.cam.2020.112991

Gimeno, L., Marín, E., Del Teso, T., & Bourhim, S. (2001). How effective has been the reduction of SO2 emissions on the effect of acid rain on ecosystems? *Science of the Total Environment*, *275*(1-3), 63–70. https://doi.org/10.1016/S0048-9697(00)00854-8

Global, D., Llp, D. C., Corporation, I. D., Llp, D. C., & Global, D. (2017). *Machine learning : things are getting intense* (tech. rep. July 2017).

Han, J., Kamber, M., & Pei, J. (2012). *Data mining concepts and techniques* (Third). Morgan Kaufmann. https://doi.org/10.1109/ICMIRA.2013.45

Heaton, J. (2012). *Introduction to the Math of Neural Networks*. https://doi.org/10.1016/j.neunet.2004.10.001

Hino, M., Benami, E., & Brooks, N. (2018). Machine learning for environmental monitoring. *Nature Sustainability*, *1*(10), 583–588. https://doi.org/10.1038/s41893-018-0142-9

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Igoe, D., Parisi, A., & Carter, B. (2013). Characterization of a Smartphone Camera's Response to Ultraviolet A Radiation. *International Journal of Remote Sensing*, 215–218. https://doi.org/10.1111/j.1751-1097.2012.01216

Isabel Molina-Gómez, N., Rodríguez-Rojas, K., Calderón-Rivera, D., Luis Díaz-Arévalo, J., & López-Jiménez, P. A. (2020). Using machine learning tools to classify sustainability levels in the development of urban ecosystems. *Sustainability (Switzerland)*, *12*(8). https://doi.org/10.3390/SU12083326

J.Clifford, J. (2008). *Atmospheric pollution* (Vol. 42).

Kashyap, P. (2017). *Machine Learning for Decision Makers*. https://doi.org/10.1007/978-1-4842-2988-0

Kirchstetter, T. W., Novakov, T., & Hobbs, P. V. (2004). Evidence that the spectral dependence of light absorption by aerosols is affected by organic carbon. *Journal of Geophysical Research D: Atmospheres*, *109*(21), 1–12. https://doi.org/10.1029/2004JD004999

Klumpp, A., Ansel, W., Klumpp, G., Vergne, P., Sifakis, N., Sanz, M. J., Rasmussen, S., Ro-Poulsen, H., Ribas, À., Peñuelas, J., Kambezidis, H., He, S., Garrec, J. P., & Calatayud, V. (2006). Ozone pollution and ozone biomonitoring in European cities Part II. Ozone-induced plant injury and its relationship with descriptors of ozone pollution. *Atmospheric Environment*, *40*(38), 7437–7448. https://doi.org/10.1016/j.atmosenv.2006.07.001

Knibbs, L. D., Van Donkelaar, A., Martin, R. V., Bechle, M. J., Brauer, M., Cohen, D. D., Cowie, C. T., Dirgawati, M., Guo, Y., Hanigan, I. C., Johnston, F. H., Marks, G. B., Marshall, J. D., Pereira, G., Jalaludin, B., Heyworth, J. S., Morgan, G. G., & Barnett, A. G. (2018). Satellite-Based Land-Use Regression for Continental-Scale Long-Term Ambient PM 2.5 Exposure Assessment in Australia. *Environmental Science and Technology*, *52*(21), 12445–12455. https://doi.org/10.1021/acs.est.8b02328

Kuchinke, C. P., & Nunez, M. (1999). Cloud transmission estimates of UV-B erythermal irradiance. *Theoretical and Applied Climatology*, *63*(3-4), 149–161. https://doi.org/10.1007/s007040050100

Kumar, U. D. (2017). *Business Analytics Business: The Science of Data-Driven Decision Making* (Wiley, Ed.).

Law, Y., Ye, L., Pan, Y., & Yuan, Z. (2012). Nitrous oxide emissions from wastewater treatment processes. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 1265–1277. https://doi.org/10.1098/rstb.2011.0317

Lee, A. K. (2015). Haze formation in China: Importance of secondary aerosol. *Journal of Environmental Sciences (China)*, *33*, 261–262. https://doi.org/10.1016/j.jes.2015.06.002

Lee, K. K., Spath, N., Miller, M. R., Mills, N. L., & Shah, A. S. (2020). Short-term exposure to carbon monoxide and myocardial infarction: A systematic review and meta-analysis. *Environment International*, *143*(June), 105901. https://doi.org/10.1016/j.envint.2020.105901

Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, *16*(6), 321–332. https://doi.org/10.1038/nrg3920

Lima, L., Novais, P., Costa, R., Bulas Cruz, J., & Neves, J. (2010). Group decision making and Quality-of-Information in e-Health systems. *Logic Journal of the IGPL*, *19*(2), https://academic.oup.com/jigpal/article-pdf/19/2/315/9406757/jzq029.pdf, 315–332. https://doi.org/10.1093/jigpal/jzq029

Liu, Y., Pellikka, P. K., Li, H., & Fang, X. (2019). Detection of the dispersion and residence of volcanic SO2 and sulfate aerosol from Nabro in 2011. *Atmospheric Environment*, *197*(January 2018), 36–44. https://doi.org/10.1016/j.atmosenv.2018.10.022

Lo Iacono, L., Boczkowski, J., Zini, R., Salouage, I., Berdeaux, A., Motterlini, R., & Morin, D. (2011). A carbon monoxide-releasing molecule (CORM-3) uncouples mitochondrial respiration and modulates the production of reactive oxygen species. *Free Radical Biology and Medicine*, *50*(11), 1556–1564. https://doi.org/10.1016/j.freeradbiomed.2011.02.033

Madronich, S., McKenzie, R. L., Björn, L. O., & Caldwell, M. (2003). Changes in biologically active ultraviolet radiation reaching the Earth's surface. *Photochemical and Photobiological Sciences*, *2*(1), 5–15. https://doi.org/10.1039/b211155c

Mariscal, G., Marbán, Ó., & Fernández, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, *25*(2), 137–166. https://doi.org/10.1017/S0269888910000032

Mayer, B., & Kylling, A. (2005). Technical note: The libRadtran software package for radiative transfer calculations - Description and examples of use. *Atmospheric Chemistry and Physics*, *5*(7), 1855–1877. https://doi.org/10.5194/acp-5-1855-2005

McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Byophysis*, *5*, 115–133.

McKenzie, R. L., Aucamp, P. J., Bais, A. F., Björn, L. O., & Ilyas, M. (2007). Changes in biologically-active ultraviolet radiation reaching the Earth's surface. *Photochemical and Photobiological Sciences*, *6*(3), 218–231. https://doi.org/10.1039/b700017k

Mendonça, E., Picado, A., Paixão, S. M., Silva, L., Barbosa, M., & Cunha, M. A. (2013). Ecotoxicological evaluation of wastewater in a municipal WWTP in Lisbon area (Portugal). *Desalination and Water Treatment*, *51*(19-21), 4162–4170. https://doi.org/10.1080/19443994.2013.768021

Metcalf, & Eddy. (2003). Metcalf & Eddy, Inc. Wastewater Engineering Teatment and Reuse.

Mingers Bsrcd, J. (1989). An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning*, *4*, 227–243. https://link.springer.com/content/pdf/10.1023%7B%5C%%7D2FA%7B%5C%%7D3A1022604100933.pdf

Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, *3*(4), 319–342. https://doi.org/10.1007/bf00116837

Mitchell, T. M. (1997). *Machine Learning*. https://doi.org/10.1007/978-3-642-21004-4_10

Norval, M., Lucas, R. M., Cullen, A., de Gruijl, F., Longstreth, J., Takizawa f, Y., & van der Leung, J. (2011). The human health effects of ozone depletion and interactions with climate change. *Photochemical and Photobiological Sciences*, *10*(2), 173. https://doi.org/10.1039/c0pp90040k

Olah, C. (2015). Understanding LSTM Networks. Retrieved May 14, 2020, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Oliveira, P., Fernandes, B., Aguiar, F., Pereira, M., Analide, C., & Novais, P. (2020). A deep learning approach to forecast the influent flow in wastewater treatment plants. *In: Analide C., Novais P., Camacho D., Yin H. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2020. IDEAL 2020. Lecture Notes in Computer Science*, *12489*. https://doi.org/https://doi.org/10.1007/978-3-030-62362-3_32

PAE. (2013). https://www.apambiente.pt/

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222. https://doi.org/10.1080/01431160412331269698

Qi, S., Jin, K., Li, B., & Qian, Y. (2020). The exploration of internet finance by using neural network. *Journal of Computational and Applied Mathematics*, *369*, 112630. https://doi.org/10.1016/j.cam.2019.112630

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *Eurasip Journal on Advances in Signal Processing*, *2016*(1). https://doi.org/10.1186/s13634-016-0355-x

Quinlan, R. (1993). *C4.5 Programs for Machine Learning*. Morgan Kaufmann.

Raglio, A., Imbriani, M., Imbriani, C., Baiardi, P., Manzoni, S., Gianotti, M., Castelli, M., Vanneschi, L., Vico, F., & Manzoni, L. (2020). Machine learning techniques to predict the effectiveness of music therapy: A randomized controlled trial. *Computer Methods and Programs in Biomedicine*, *185*, 105160. https://doi.org/10.1016/j.cmpb.2019.105160

Román, R., Antón, M., Valenzuela, A., Gil, J. E., Lyamani, H., De Miguel, A., Olmo, J., Bilbao, J., & Alados-Arboledas, L. (2013). Evaluation of the desert dust effects on global, direct and diffuse spectral ultraviolet irradiance. *Tellus, Series B: Chemical and Physical Meteorology*, *65*(1). https://doi.org/10.3402/tellusb.v65i0.19578

Sandryhaila, A., & Moura, J. M. (2014). Representation and processing of massive data sets with irregular structure. *IEEE SIGNAL PROCESSING MAGAZINE*, *5*(31), 80–90.

Sarkodie, S. A. (2021). Environmental performance, biocapacity, carbon & ecological footprint of nations: Drivers, trends and mitigation options. *Science of the Total Environment*, *751*, 141912. https://doi.org/10.1016/j.scitotenv.2020.141912

Schoenherr, T. (2012). The role of environmental management in sustainable business development: A multi-country investigation. *International Journal of Production Economics*, *140*(1), 116–128. https://doi.org/10.1016/j.ijpe.2011.04.009

Seinfeld, J. H., Pandis, S. N., & Noone, K. (1998). *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change* (Vol. 51). https://doi.org/10.1063/1.882420

Shalev-Shwartz, S., & Ben-David, S. (2013). *Understanding machine learning: From theory to algorithms* (Vol. 9781107057). https://doi.org/10.1017/CBO9781107298019

Song, R., Li, F., Quan, W., Yang, X., & Zhao, J. (2021). Skill learning for robotic assembly based on visual perspectives and force sensing [cited By 0]. *Robotics and Autonomous Systems*, *135*. https://doi.org/10.1016/j.robot.2020.103651

Spellman, F. R. (2013). *Handbook of Water and Wastewater Treatment Plant Operations*. https://doi.org/10.1201/b15579

Sun, S., Cheng, X., Sha Li, F. Q., Liu, Y., & Sun, D. (2013). N 2 O emission from full-scale urban wastewater treatment plants : a comparison between A 2 O and SBR Shichang Sun , Xiang Cheng , Sha Li , Fei Qi , Yan Liu and Dezhi Sun. *Water Science & Technology*, *67.9*, 1887–1893. https://doi.org/10.2166/wst.2013.066

Sunyer, J., Ballester, F., Le Tertre, A., Atkinson, R., Ayres, J. G., Forastiere, F., Forsbergg, B., Vonk, J. M., Bisanti, L., Tenías, J. M., Medina, S., Schwartz, J., & Katsouyanni, K. (2003). The association of daily sulfur dioxide air pollution levels with hospital admissions for cardiovascular diseases in Europe (the Aphea-II study). *European Heart Journal*, *24*(8), 752–760. https://doi.org/10.1016/S0195-668X(02)00808-4

Thom, S. R., Fisher, D., Xu, Y. A., Garner, S., & Ischiropoulos, H. (1999). Role of nitric oxide-derived oxidants in vascular injury from carbon monoxide in the rat. *American Journal of Physiology*, *276*(3 PART 2), H984–H992. https://doi.org/10.1152/ajpheart.1999.276.3.h984

Thom, S. R., Xu, Y. A., & Ischiropoulos, H. (1997). Vascular endothelial cells generate peroxynitrite in response to carbon monoxide exposure. *Chemical Research in Toxicology*, *10*(9), 1023–1031. https://doi.org/10.1021/tx970041h

Vasilev, I., Slater, D., Spacagna, G., Roelants, P., & Zocca, V. (2019). *Python Deep Learning*. Packt. https://doi.org/10.7210/jrsj.33.92

Wang, L., Liu, C., Meng, X., Niu, Y., Lin, Z., Liu, Y., Liu, J., Qi, J., You, J., Tse, L. A., Chen, J., Zhou, M., Chen, R., Yin, P., & Kan, H. (2018). Associations between short-term exposure to ambient sulfur dioxide and increased cause-specific mortality in 272 Chinese cities. *Environment International*, *117*(April), 33–39. https://doi.org/10.1016/j.envint.2018.04.019

World Health Organization. (2017). No Title. Retrieved, from https://www.who.int/news-room/q-a-detail/ultraviolet-(uv)-index

Wright, Richard T., D. F. B. (2019). *Environmental Science Toward A Sustainable Future* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Yin, H. H. S., Langenheldt, K., Harlev, M., Mukkamala, R. R., & Vatrapu, R. (2019). Regulating Cryptocurrencies: A Supervised Machine Learning Approach to De-Anonymizing the Bitcoin Blockchain. *Journal of Management Information Systems*, *36*(1), 37–73. https://doi.org/10.1080/07421222.2018.1550550

Yoshua Bengio, Patrice Simard, & Paolo Frasconi. (1994). Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Network*, *5*(2), 157–166. http://www.dsi.unifi.it/%7B~%7Dpaolo/ps/tnn-94-gradient.pdf

Zhang, H., Di, B., Liu, D., Li, J., & Zhan, Y. (2019). Spatiotemporal distributions of ambient SO2 across China based on satellite retrievals and ground observations: Substantial decrease in human exposure during 2013–2016. *Environmental Research*, *179*(October), 108795. https://doi.org/10.1016/j.envres.2019.108795

Zhou, H., Tang, J., & Zheng, H. (2015). Machine Learning for Medical Applications. *Scientific World Journal*, *2015*. https://doi.org/10.1155/2015/825267

A. Data exploration appendix

Missing days		
2016-05-28 to 2016-05-31	2016-06-17 to 2016-06-20	2016-10-24 to 2016-10-26
2016-11-05 to 2016-11-09	2017-01-08	2017-02-02
2018-01-04	2018-02-09	2018-02-16
2018-03-15	2018-03-26	2018-04-04
2018-04-24	2018-06-01	2018-06-03
2018-06-06 to 2018-06-17	2018-06-20	2018-06-25
2018-07-11	2018-08-31	2018-10-25
2018-10-29	2018-11-09	2018-11-12
2018-12-09	2018-12-15	2019-02-01
2019-02-05	2019-04-02 to 2019-04-04	2019-04-07
2019-04-09	2019-04-10 to 2019-04-15	2019-05-18
2019-05-21	2019-06-20	2019-06-25
2019-07-13	2019-07-14	2019-07-18
2020-01-10	2020-01-28	2020-02-12

Table A.1: Missing observations concerning the water quality data.

B. Data preparation appendix

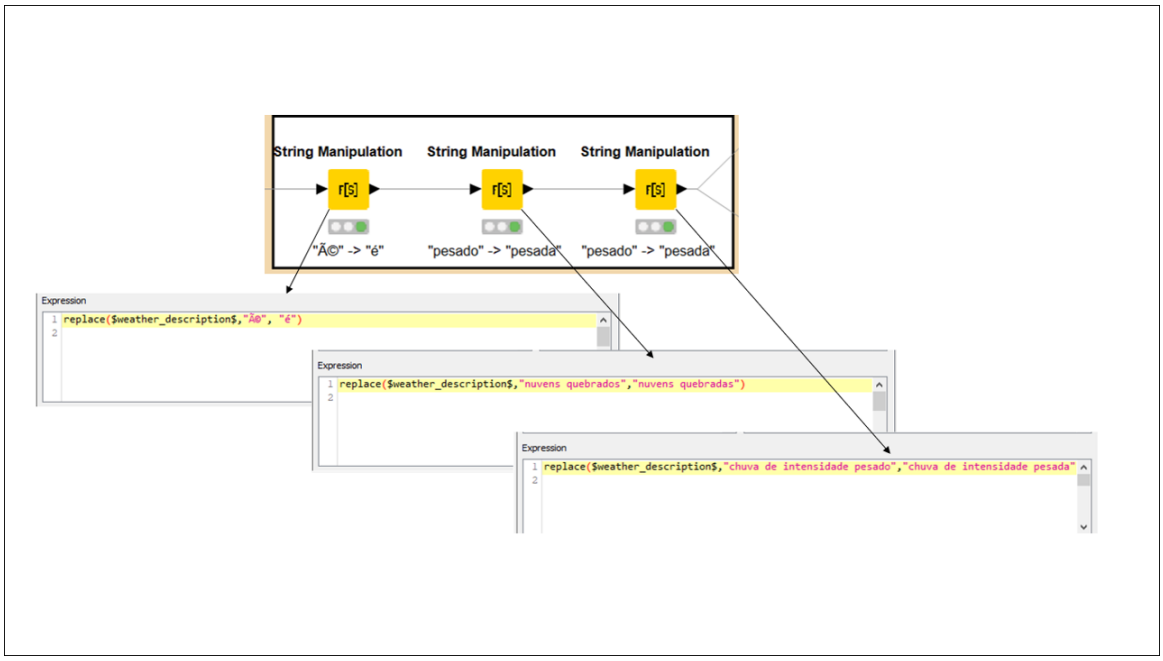


Figure B.1: String manipulation KNIME nodes and its configuration, regarding string manipulation process.

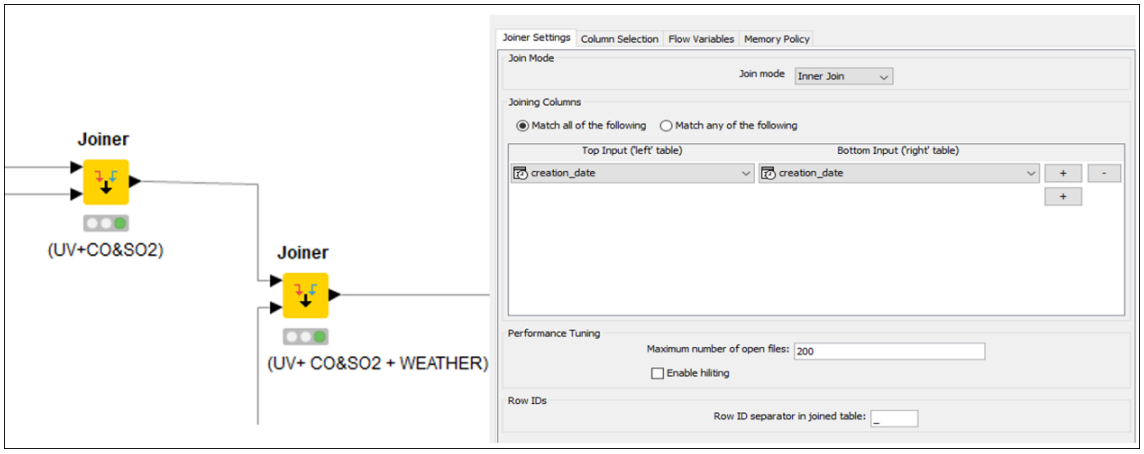


Figure B.2: Joiner KNIME node and its configuration, regarding data joining.

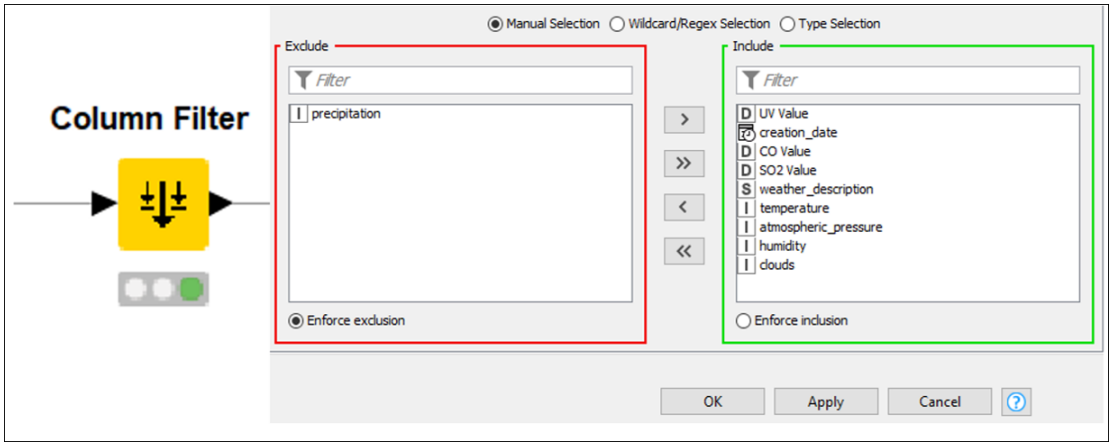


Figure B.3: Column Filter KNIME node and its configuration, to remove "Precipitation"attribute.

```
dataset = dataset.drop(columns = 'Precipitation')
```

Figure B.4: Precipitation' column drop, in python.

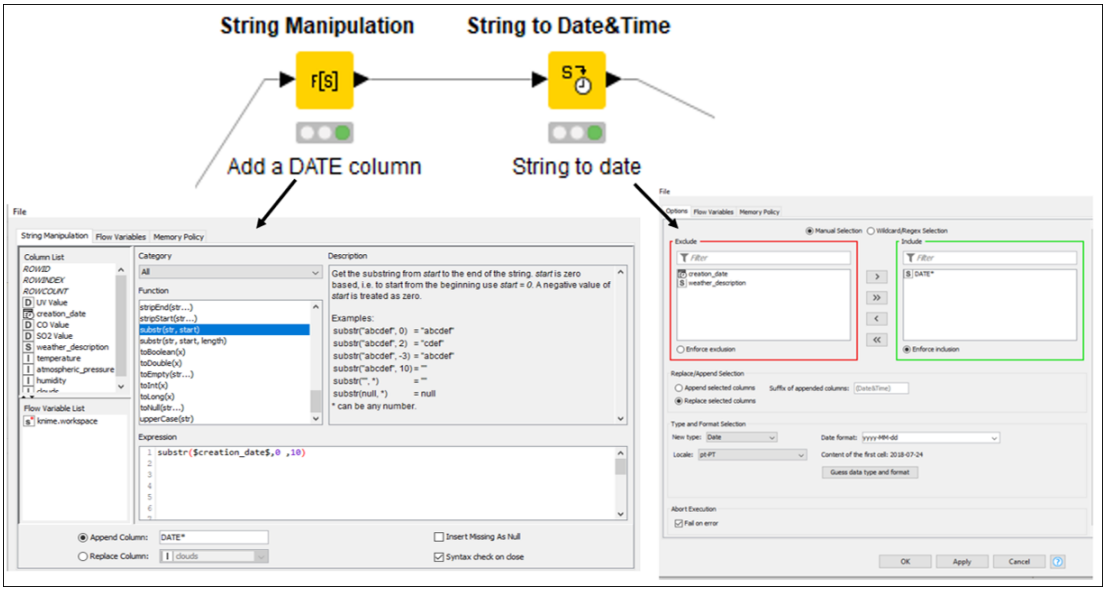


Figure B.5: KNIME nodes and its configurations to date manipulation.

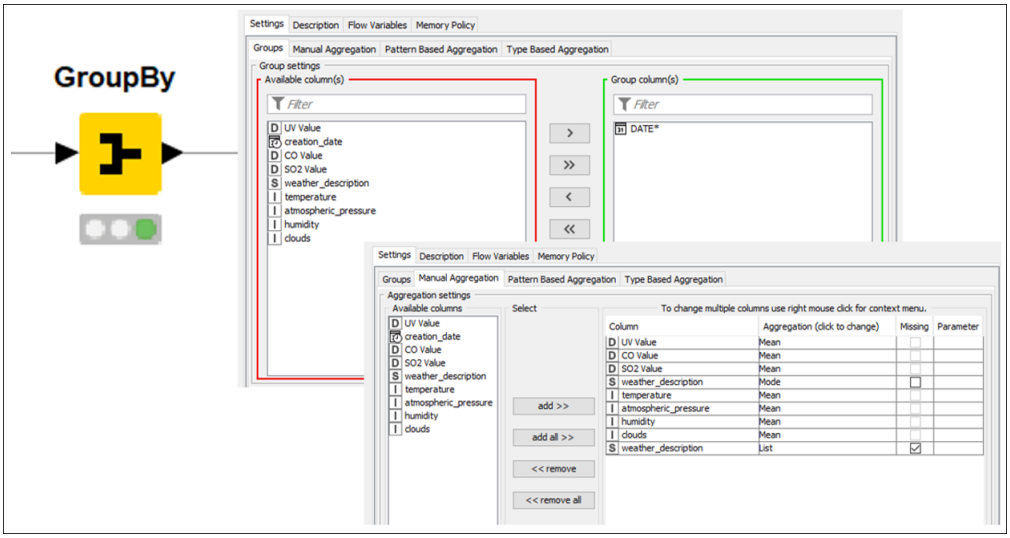


Figure B.6: Group by KNIME node and its configuration.

```
dataset = dataset.groupby(["Day","Month", "Year"]).agg({'Uv Value' : 'mean',
                                    'CO Value' : 'mean',
                                    'SO2 Value' : 'mean',
                                    'Temperature' : 'mean',
                                    'Atmospheric Pressure' : 'mean',
                                    'Humidity' : 'mean',
                                    'Clouds':'mean',
                                    'Weather description' : 'max'}).reset_index()
```

Figure B.7: Function to reshape the data to LSTM time series forecast.

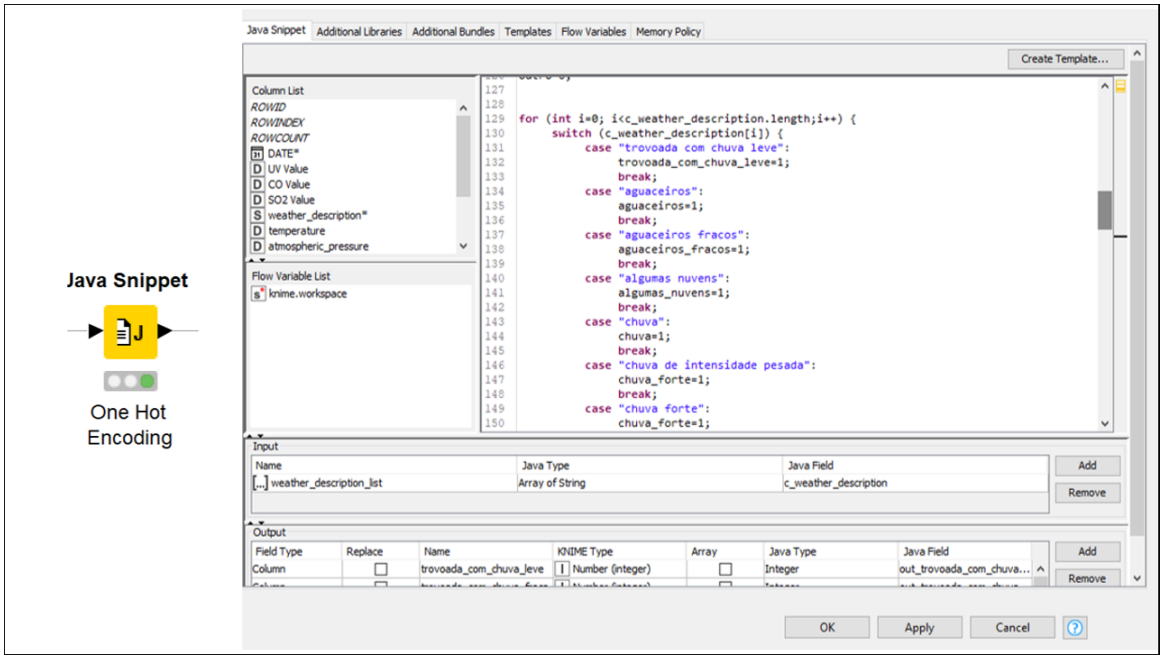


Figure B.8: Java snippet KNIME node and its configuration.

```
Weather = dataset.pop('Weather description')

w_list = list(set(Weather))

# >label encoding

le = preprocessing.LabelEncoder()

le.fit(w_list)

dataset['Weather description'] = le.transform(Weather)
```

Figure B.9: Label encoding of ”weather description”attribute, in python.

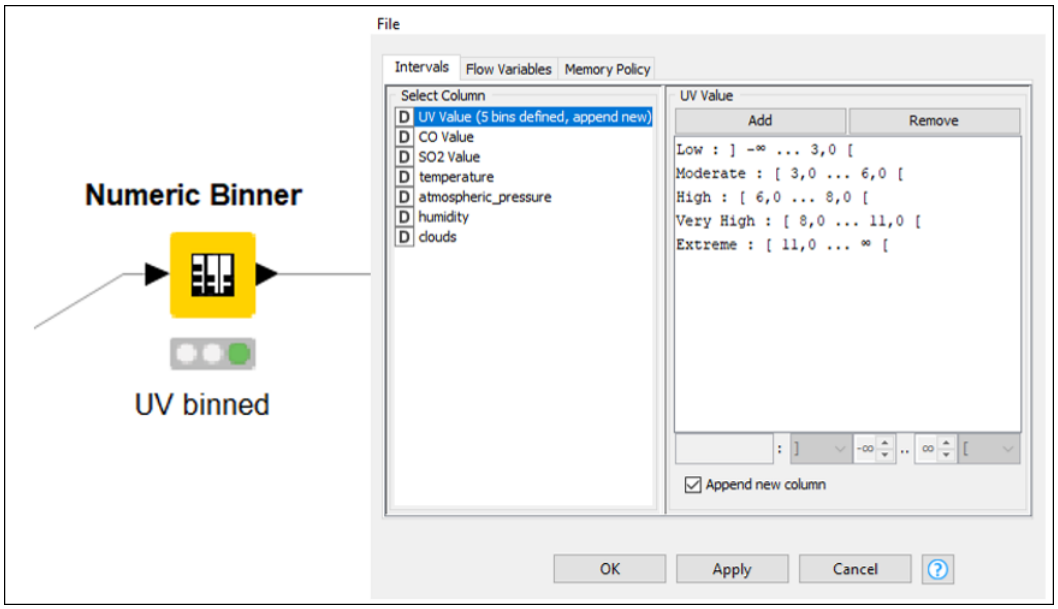


Figure B.10: Numeric binner node and its configuration, at KNIME.

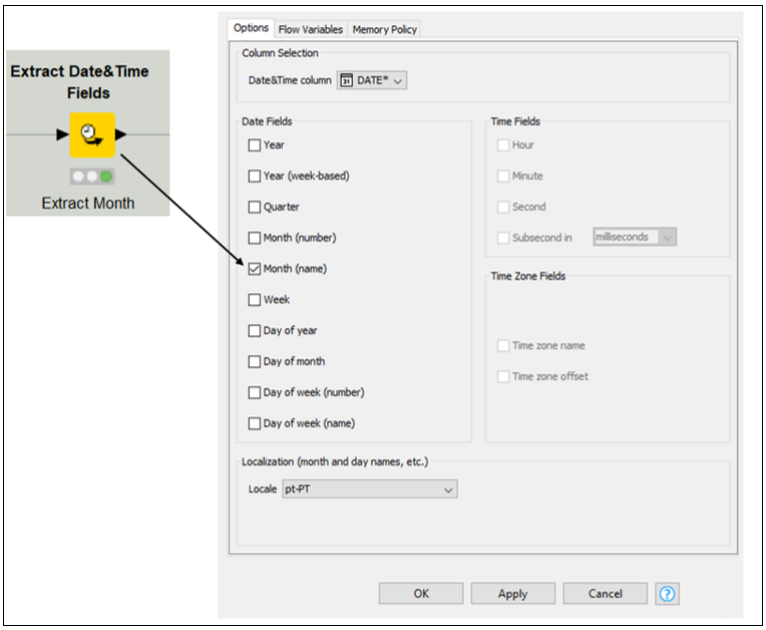


Figure B.11: Extract date&time fields and its configuration, in KNIME.

```
dataset['Date'] = pd.to_datetime(dataset['Date'])
dataset.dtypes
date = dataset['Date']

Year = date.dt.year
Month = date.dt.month
Day = date.dt.day
Hour = date.dt.hour

dataset= dataset.assign(Year = Year, Month = Month, Day = Day, Hour = Hour)

dataset = dataset.drop(columns = 'Date')
```

Figure B.12: Creation of the numeric ”day”, ”month”, ”year”and ”hour”attribute, in python.

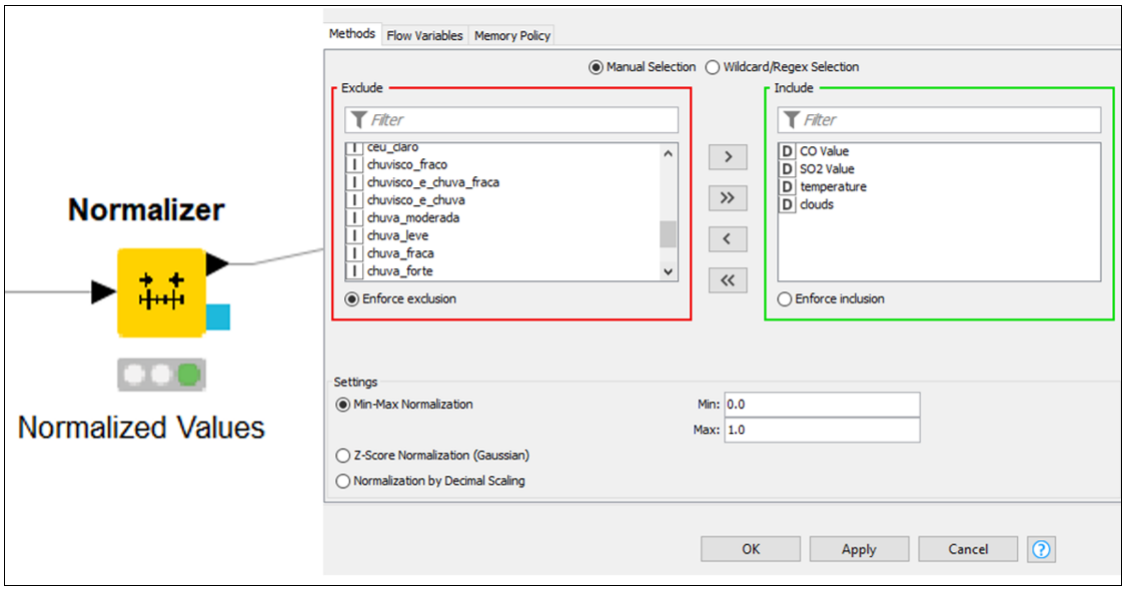


Figure B.13: Normalizer KNIME node and is configuration.

```
# Data Normalization
def data_normalization(df, norm_range=(0, 1)):
    dict_of_scalers = dict()
    for col in df.columns:
        if df[col].dtype != type(object):
            scaler = MinMaxScaler(feature_range=norm_range)
            df[[col]] = scaler.fit_transform(df[[col]])
            dict_of_scalers[col] = scaler
    return dict_of_scalers
```

Figure B.14: Function to normalize data, in python.

```
column_names = ['Date', 'DO-Aerated zone (Line 2)','DO-Anoxic Zone (Line 2)','pH-Secondary Decanter',
'pH-tert. treatment','Temperature-tert. treatment']
raw_dataset = pd.read_csv(path, encoding ='latin-1', header = 0, index_col = False, names=column_names)

dataset = raw_dataset.copy()



dataset_v1 = dataset.copy()

#Replace values equals to 0 and 1by missing values (nan), for the "pH-tert. treatment" attribute.
dataset_v1['pH-tert. treatment'][dataset_v1['pH-tert. treatment']==0] = np.nan
dataset_v1['pH-tert. treatment'][dataset_v1['pH-tert. treatment']==1] = np.nan

#Replace values equals to 0 and 1 by missing values(nan), for the "DO-Anoxic Zone (Line 2)" attribute.
dataset_v1['DO-Anoxic Zone (Line 2)'][dataset_v1['DO-Anoxic Zone (Line 2)']==0] = np.nan
dataset_v1['DO-Anoxic Zone (Line 2)'][dataset_v1['DO-Anoxic Zone (Line 2)']==1] = np.nan

#Remove the missing values defined above
dataset_v2= dataset_v1.dropna(subset=['pH-tert. treatment'])
dataset_v2= dataset_v1.dropna(subset=['pH-tert. treatment'])

#Fill the remaining missing values through liner interpolation
dataset_v2 = dataset_v2.interpolate(method="linear", axis=0).ffill().bfill()
```

Figure B.15: Process to manipulate missing values regarding the water quality dataset.

```
#Create the range of dates over study
dates= pd.date_range("2018-07-24","2020-03-23")
#Transform the list of dates into a dataframe
dates = pd.DataFrame(dates)
#Transform to date time type
dates = dates[0].dt.date
dates = pd.DataFrame(dates)
dates = dates.set_index(0)

#Concat the existing data with the new dates, creating, now, missing values in the dataset
dataset =  pd.concat([dataset,dates], axis =1)

scaler = data_normalization(dataset)

#Fill missing values with a value equal to 1.1
dataset = dataset.fillna(1.1)

dataset = dataset.sort_index(axis = 0)
```

Figure B.16: Handle missing time steps for LSTM predictions.

C. Tuning process appendix

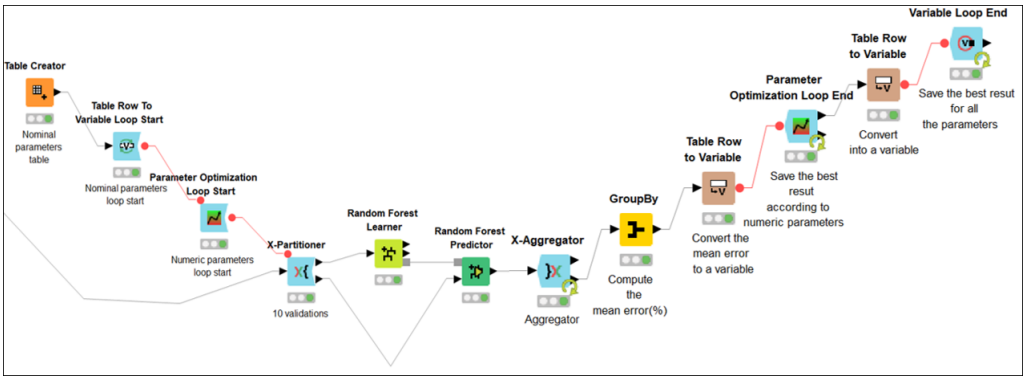


Figure C.1: KNIME Workflow to carry out the regression decision tree tuning process.

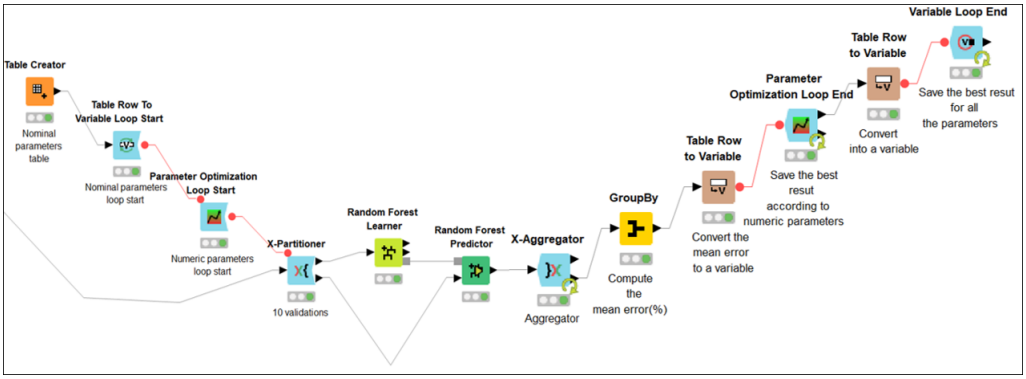


Figure C.2: KNIME Workflow to carry out the classification random forest tuning process.

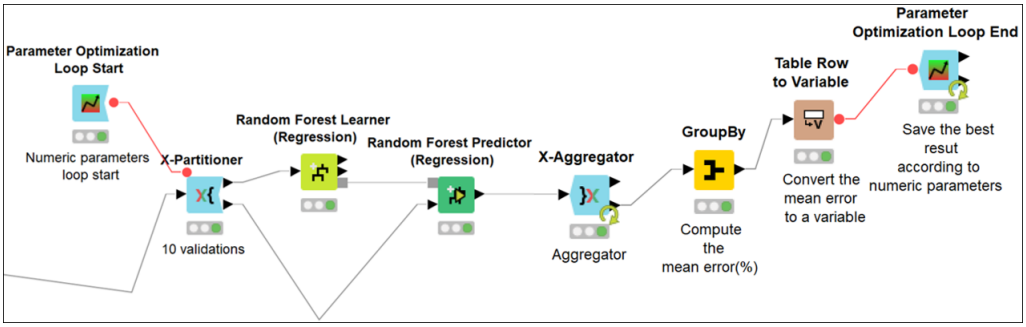


Figure C.3: KNIME Workflow to carry out the regression random forest tuning process.

D. Appendix showing the process to find the ideal number of epochs for UV classification using MLPs

Here are presented the learning curves, used to understand the estimated number of epochs considered ideal for train the MLP classification model, regarding the UV prediciton. These learning curves show the Accuracy value by epoch.

The figure D.1 and D.2 shows the learning curves resulting from the 'relu' activation function.

Activation function - Rectified Linear ('relu')

Lowest

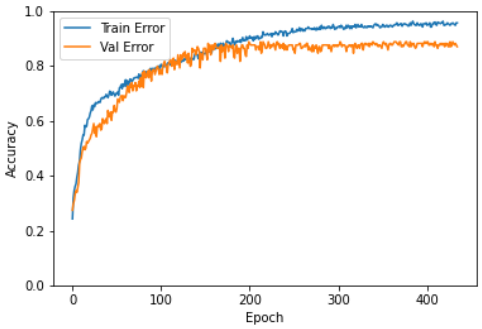


Figure D.1: Learning curve, classification UV prediction, for lowest values-relu.

Highest

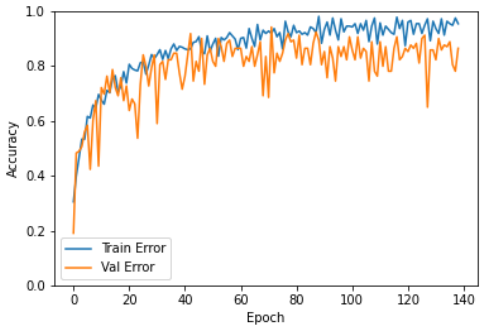


Figure D.2: Learning curve, classification UV prediction, for highest values-relu.

Using the "relu" function is possible to conclude that, for the lowest values, the accuracy starts to stabilize its value around 250 epochs. However, for the highest values, the curve exhibits, around 100 epochs, the higher values. However, this curve shows some anomalous variations in accuracy, probably due to the high learning rate value, which makes it a bit inconclusive.

The figure D.3 and D.4 shows the learning curves, using the activation function 'tanh'.

Activation function - Tanh

Lowest

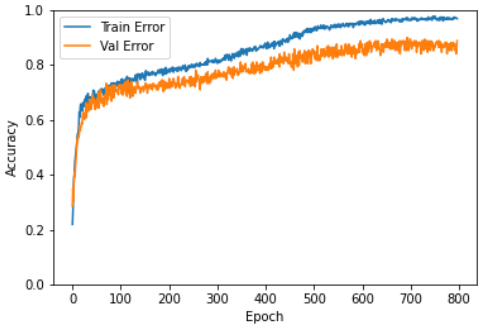


Figure D.3: Learning curve, classification UV prediction, for lowest values-tanh.

Highest

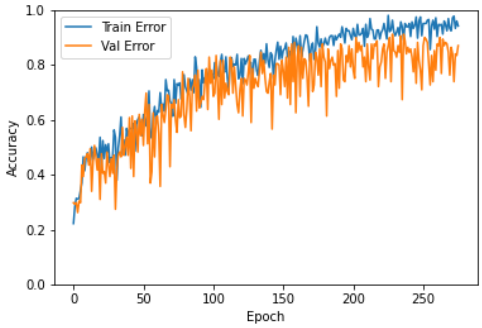


Figure D.4: Learning curve, classification UV prediction, for lowest values-tanh.

To the lowest values, the accuracy stops increasing in a significant way, around 600 epochs. On the other hand, using the highest values, the curve is a bit inconclusive once there are verified a lot of accuracy values variation.

At last, the figure D.5 and D.6 shows the process, using the 'sigmoid' function.

Activation function - Sigmoid

Lowest

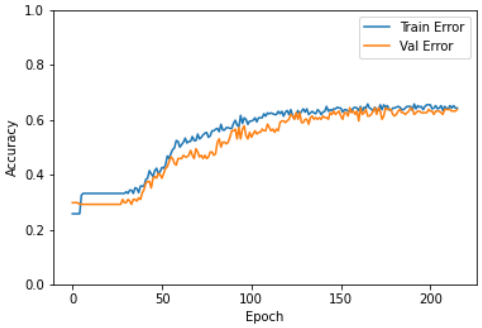


Figure D.5: Learning curve, classification UV prediction, for lowest values-sigmoid.

Highest

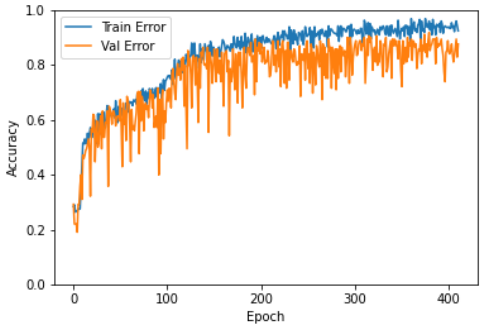


Figure D.6: Learning curve, classification UV prediction, for lowest values-sigmoid.

Analyzing the graphs for the lowest parameters the learning curve stop increasing the accuracy value, around 150 epochs, however, reaches low accuracy values (less than 70%). The second one shows a lot of variations, as the previous two.

Finally, after analyzing all the learning curves, for the multilayer perceptron classification model, the higher number of epochs considered ideal is 600, aiming for a training process through enough number of epochs.

E. Appendix showing the process to find the ideal number of epochs for UV using MLPs

Here, are explored the learning curves, to find the ideal number of epochs for the UV index MLP regression prediction. The following figures shows the iterative process for the “relu” function.

Activation function - Rectified Linear ('relu')

Lowest

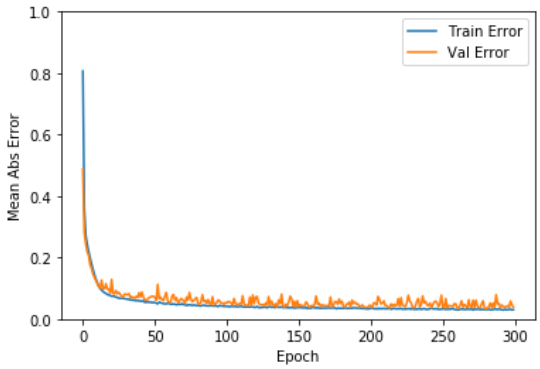


Figure E.1: Learning curve, regression UV prediction, for lowest values, MAE value-relu.

Highest

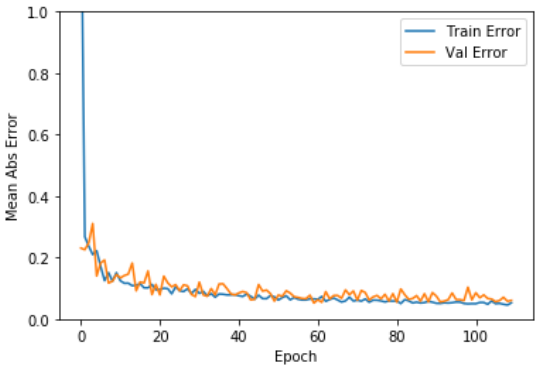


Figure E.2: Learning curve, regression UV prediction, for highest values, MAE value-relu.

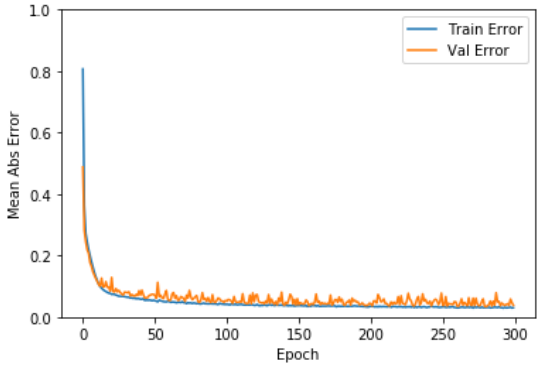


Figure E.3: Learning curve, regression UV prediction, for lowest values, RMSE value-relu.

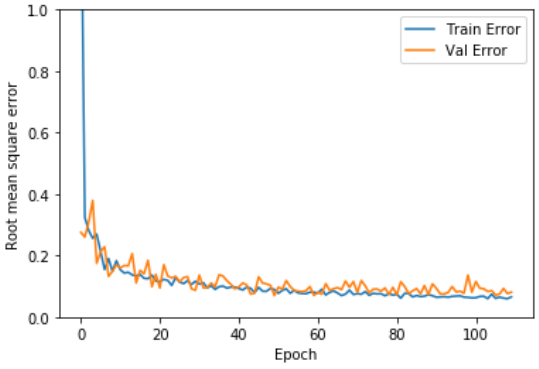


Figure E.4: Learning curve, regression UV prediction, for highest values, RMSE value-relu.

Using the rectified linear function as activation function, it is possible understand that: for lowest values, the number of epochs from there is no significant decrease of MAE and RMSE is around 150 epochs, as the figures E.1 and E.3 shows.

Further, using the highest parameters values, the number of epochs from which the errors stop decreasing significantly is around 70 epochs, figures E.2, and E.4.

Next is shown the iterative process using the "Tanh" activation function.

Activation function-Tanh

Lowers

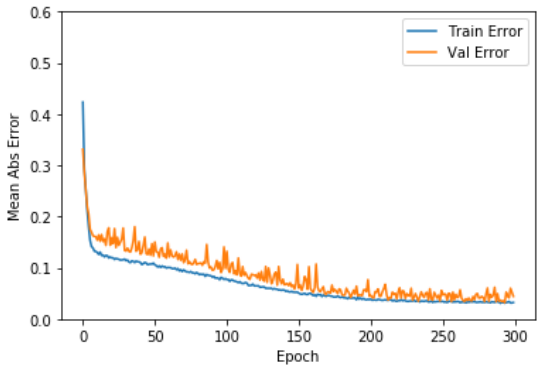


Figure E.5: Learning curve, regression UV prediction, for lowest values, MAE value-tanh.

Highest

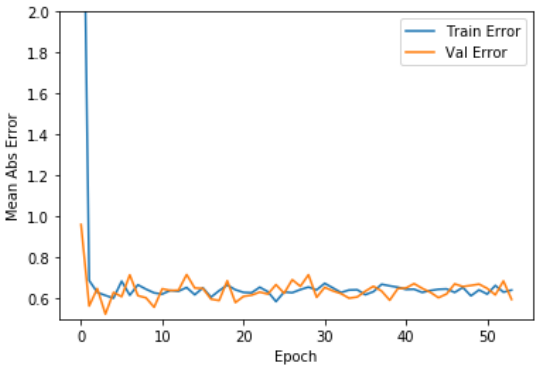


Figure E.6: Learning curve, regression UV prediction, for highest values, MAE value-tanh.

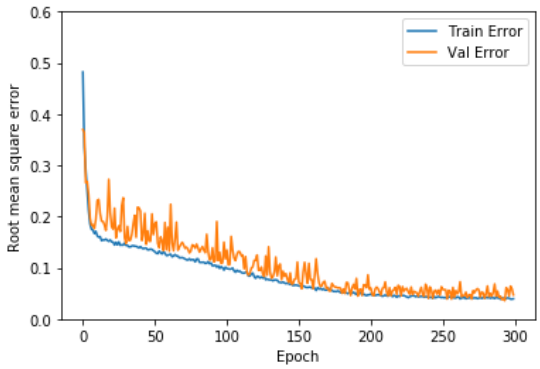


Figure E.7: Learning curve, regression UV prediction, for lowest values, RMSE value-tanh.

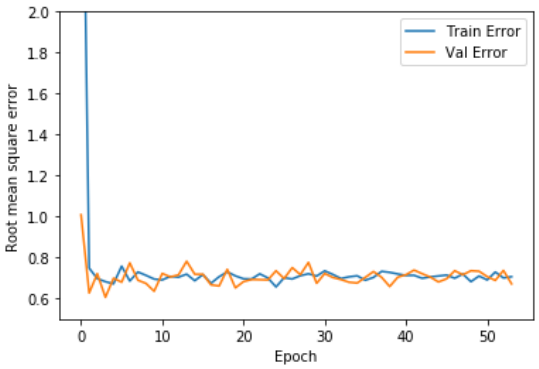


Figure E.8: Learning curve, regression UV prediction, for highest values, RMSE value-tanh.

Using the Tanh activation function, for lowest parameters values, the number of epochs from which there is no significant decrease of MAE and RMSE is around 200 epochs, as the figures E.5 and E.7 shows.

Using the highest parameters values the errors are higher (between 0.6 and 0.8). Here, the error is approximately the same for all the epoch. For that reason, there are no conclusions about the number of epochs in this specific combination of parameters, figures E.6, and E.8.

The next figures show the process implemented using the 'sigmoid' function.

Activation function - Sigmoid

Lowers

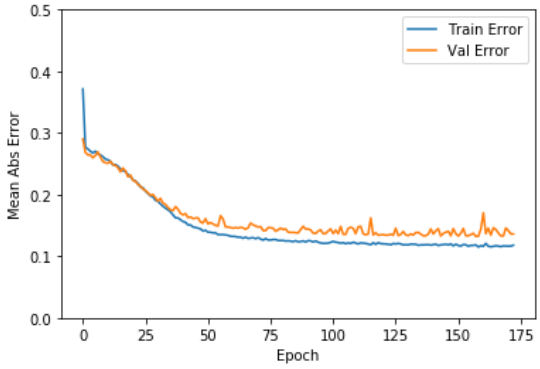


Figure E.9: Learning curve, regression UV prediction, for lowest values, MAE value-sigmoid.

Highest

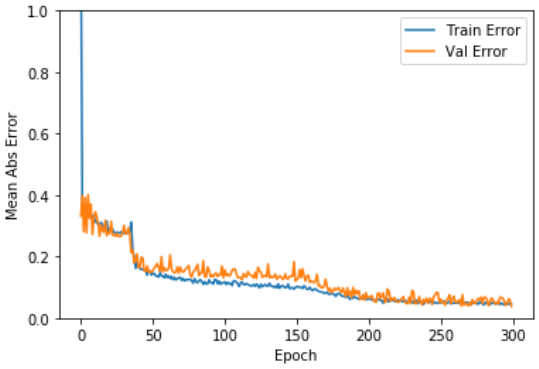


Figure E.10: Learning curve, regression UV prediction, for highest values, MAE value-sigmoid.

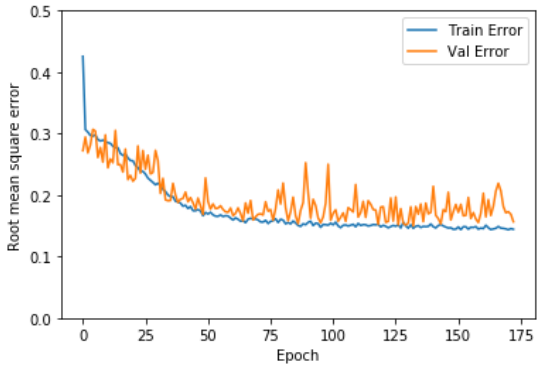


Figure E.11: Learning curve, regression UV prediction, for lowest values, RMSE value-sigmoid.

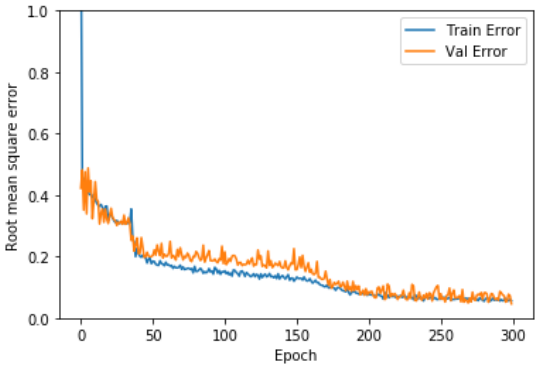


Figure E.12: Learning curve, regression UV prediction, for highest values, RMSE value-sigmoid.

With the sigmoid activation function, the learning curves for lowest parameters values, shows that around 75 epochs the MAE and RMSE values stop present a significant decrease, figures E.9 and E.11 shows.

Using the highest values, the number of epochs from which the errors stop decreasing is around 200, as presented in the figures E.10 and E.12.

Once analyzed all the scenarios, the extreme values show that the higher epochs value register (to which the MAE and RMSE values stop decrease significantly) is 200. Hence, it is possible to conclude that this number will be used to tune the model and deploy the model.

F. Appendix showing the process to find the ideal number of epochs for UV prediction using LSTMs

In the UV index LSTM prediction were used curves from the third iteration of the cross-validation process to create the learning curves. Here, were only evaluated curves using the RMSE as a metric. The figure F.1 and F.2 shows that curves.

Activation function - Rectified Linear ('relu')

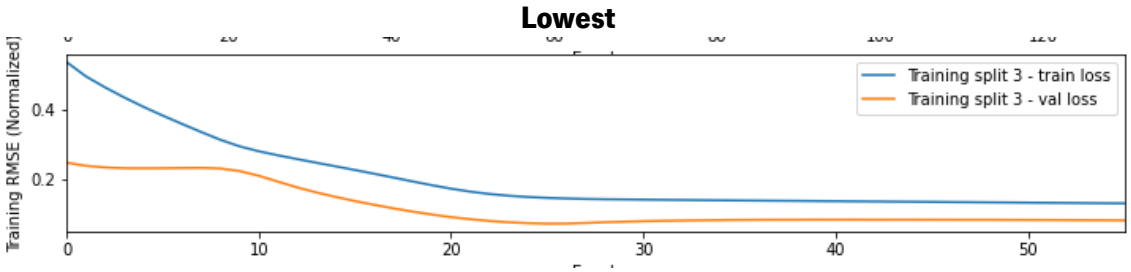


Figure F.1: Learning curve, LSTM UV prediction, for lowest values-relu.

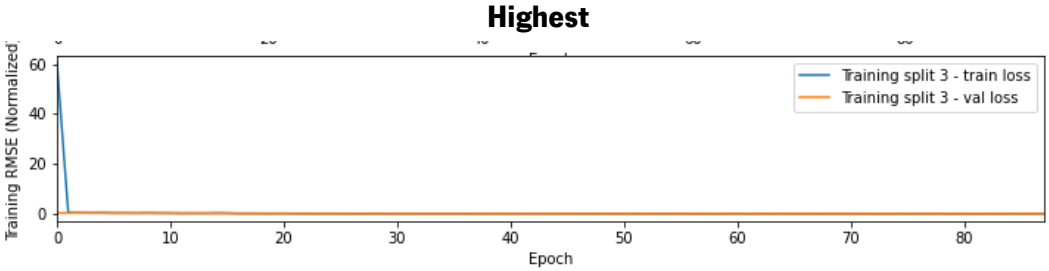


Figure F.2: Learning curve, LSTM UV prediction, for highest values-relu.

Analyzing both curves is possible to conclude that, for the lowest parameters, the value from which the RMSE value starts to stabilize is 20 epochs. Meanwhile, for the higher parameters, the curve is a bit inconclusive.

The following figures (F.3 and F.4) show the process using the “Tanh” function.

Activation function - Tanh

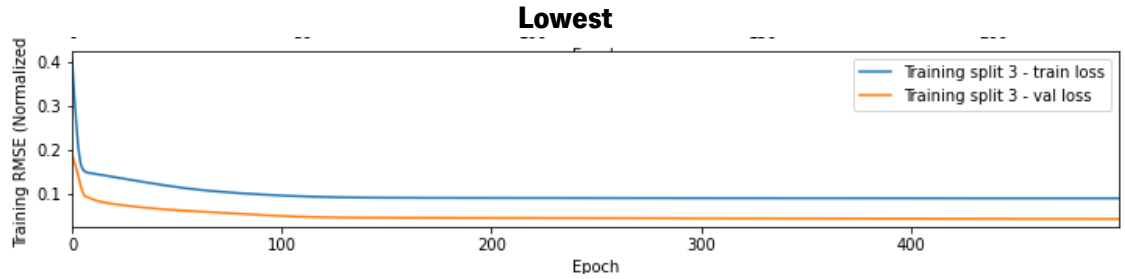


Figure F.3: Learning curve, LSTM UV prediction, for lowest values-tanh.

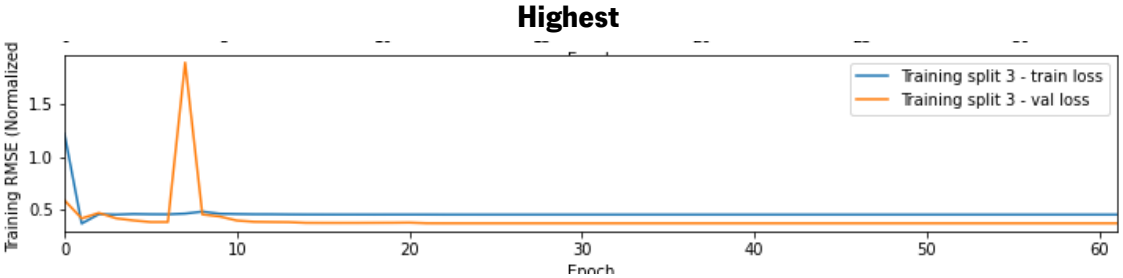


Figure F.4: Learning curve, LSTM UV prediction, for highest values-tanh.

The curves show, for the lowest parameters, a establish from between 100 and 200 epochs. However, for the highest parameters, the curve presents a stabilization around the 10 epochs, showing very small errors.

The figure F.5 and F.6 shows the process using the sigmoid function.

Activation function - Sigmoid

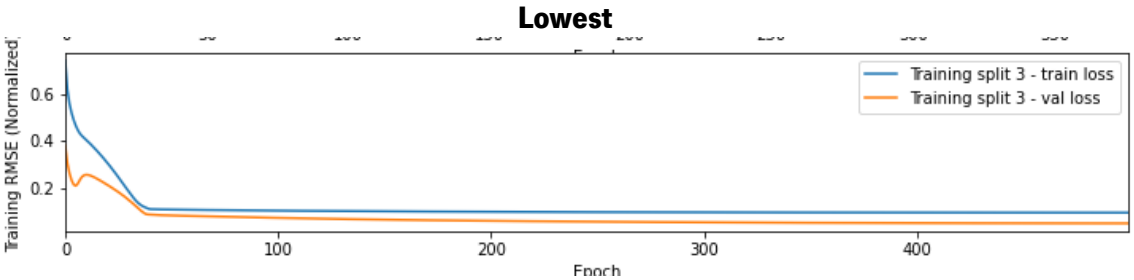


Figure F.5: Learning curve, LSTM UV prediction, for lowest values-sigmoid.

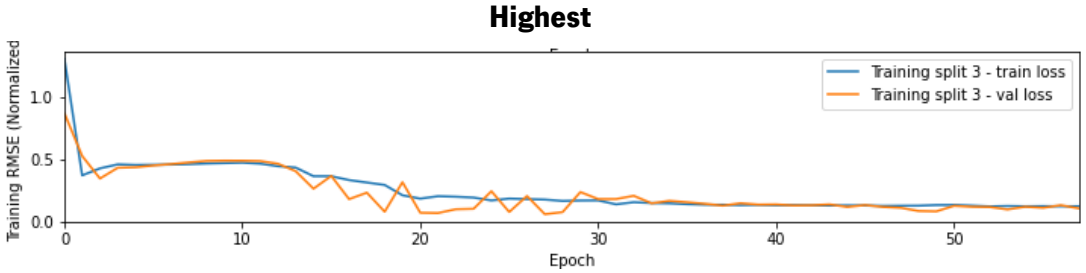


Figure F.6: Learning curve, LSTM UV prediction, for highest values-sigmoid.

Analyzing the curves, for the lowest parameters the curve shows stabilization around 50 epochs. Further, for the highest, the stabilization is around 40 epochs.

Finally, as a conclusion the optimal number to train the LSTM model, for the UV prediction , is around 150 epochs.

G. Appendix showing the process to find the ideal number of epochs for CO prediction using LSTMs

To figure out the ideal number of epochs for the carbon monoxide prediction was used, also, a set of curves, using the RMSE as an evaluation metric.

First, using the "relu" activation function, were generated the curves from the figure G.1 and G.2.

Activation function - Rectified Linear ('relu')

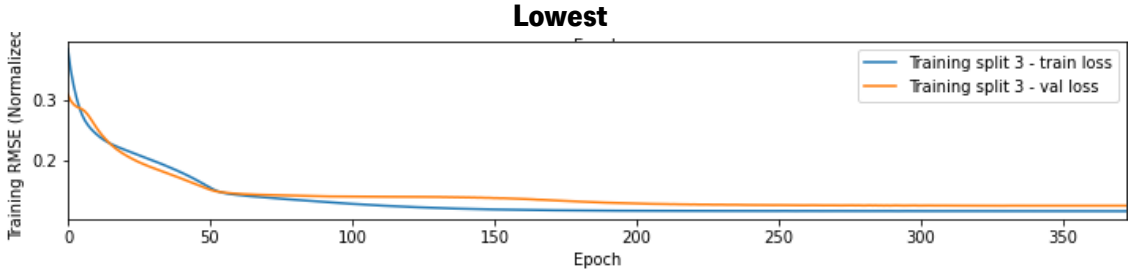


Figure G.1: Learning curve, LSTM CO prediction, for lowest values-relu.

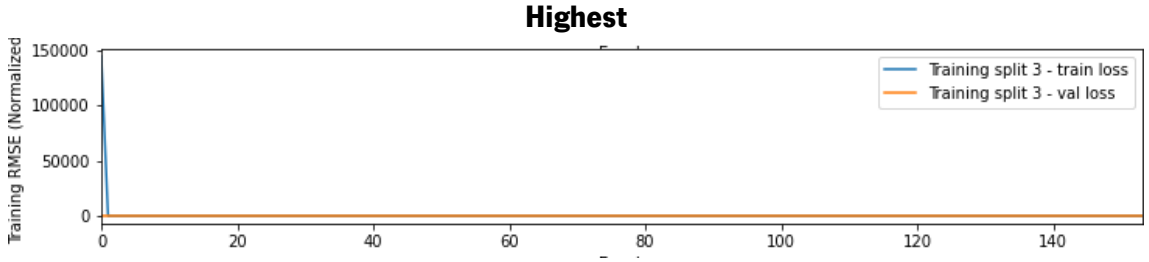


Figure G.2: Learning curve, LSTM CO prediction, for highest values-relu.

Analyzing the curves it is possible to conclude several aspects. For the lowest values, the number of epochs from which starts the stabilization is between 100 and 200. On the other hand, for the highest values, the curve is inconclusive.

The following figures (G.3 and G.4) show the curves resulted using the “tanh” activation function.

Activation function - Tanh

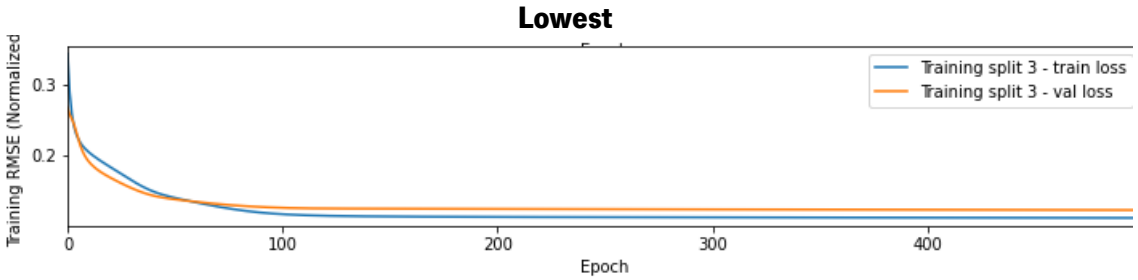


Figure G.3: Learning curve, LSTM CO prediction, for lowest values-tanh.

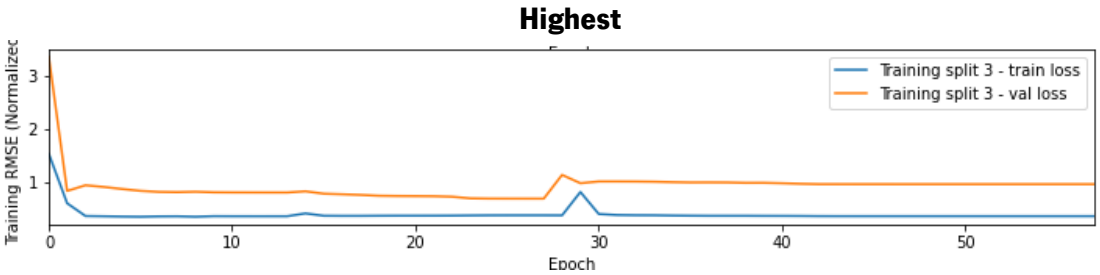


Figure G.4: Learning curve, LSTM CO prediction, for highest values-tanh.

For the lowest values, the RMSE start stops vary significantly around 100 epochs. For the highest ones, this stabilization occurs around 30 epochs.

Last, using the sigmoid activation function resulted the curves from the figure G.5 and figure G.6. Here, is shown the process for the lowest and higher parameters combinations,also.

Activation function - Sigmoid

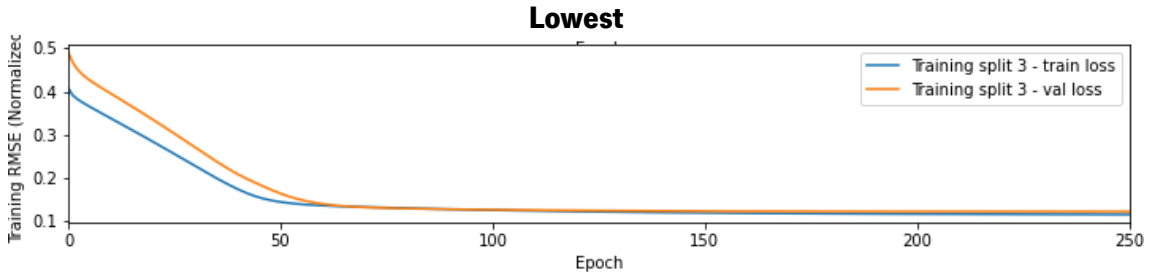


Figure G.5: Learning curve, LSTM CO prediction, for lowest values-sigmoid.

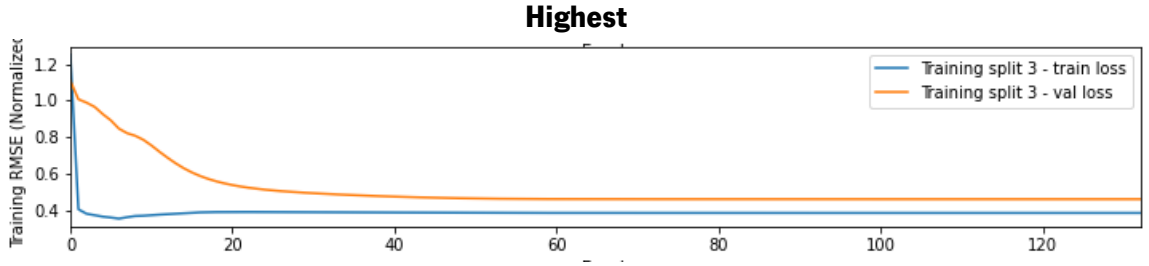


Figure G.6: Learning curve, LSTM CO prediction, for highest values-sigmoid.

For the lowest parameters, the RMSE stabilizes around 60 epochs. Further, analyzing the curve for the highest values, it is possible to conclude that this stabilization occurs about the 40 epochs.

Finally, the number of epochs considered enough to train the LSTM model (regarding the Carbon Monoxide prediction) is 150 epochs.

H. Appendix showing the process to find the number of epochs for water pH using MLPs

To the water pH prediction, the number of epochs is defining, using the MLP regression model. The figure H.1 and H.2, shows the learning curve, using the MAE value to the lowest and highest values, respectively. At the same time, the figure H.3 ad H.4 show the Learning curve, using the RMSE value, for the lowest and higher values. All of this using the 'relu' activation function.

Activation function - Rectified Linear ('relu')

Lowest

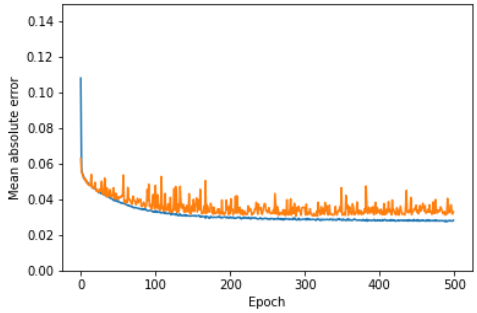


Figure H.1: Learning curve, regression water pH prediction, for lowest values, MAE value-relu.

Highest

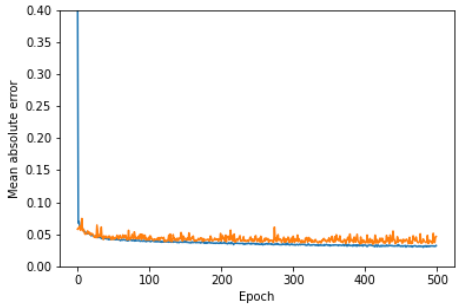


Figure H.2: Learning curve, regression water pH prediction, for highest values, MAE value-relu.

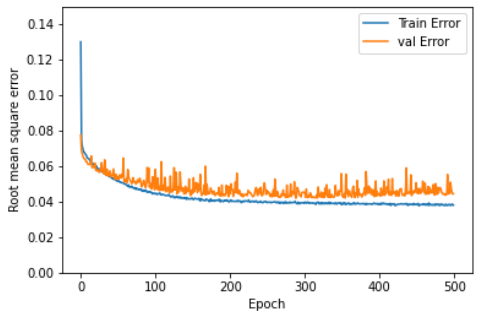


Figure H.3: Learning curve, regression water pH prediction, for lowest values, RMSE value-relu.

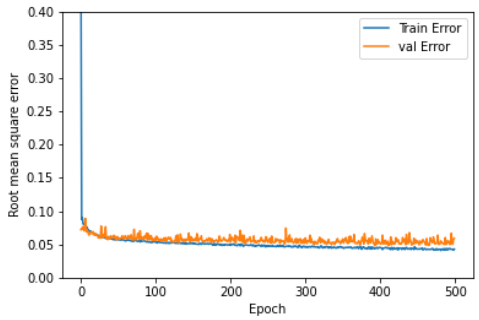


Figure H.4: Learning curve, regression water pH prediction, for highest values, RMSE value-relu.

Analyzing the curve, it is possible to conclude that for the lowest parameter values, the model behaves similarly for both metrics occurring the same to the highest parameters. For the lowest parameters, the number of epochs from which the value of the metrics stops to vary significantly is around 200 epochs. For the highest parameter values, the metrics stop decreasing very soon, that is, for reduced epochs values.

The figure H.5, H.6, H.7 and H.8, shows the learning curves using MEAN and RMSE, for the lowest and highest combinations of parameters, respectively, using the 'Tanh' activation function.

Activation function - Tanh

Lowest

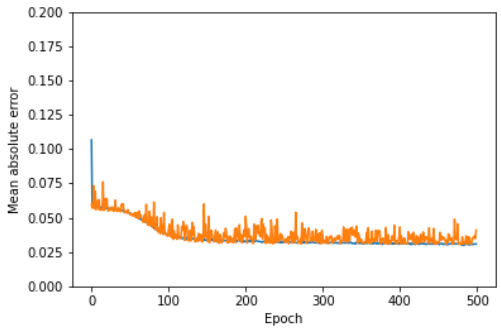


Figure H.5: Learning curve, regression water pH prediction, for lowest values, MAE value-tanh.

Highest

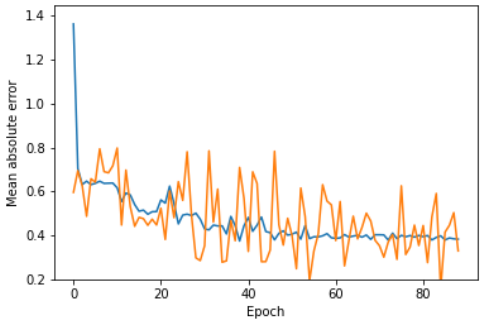


Figure H.6: Learning curve, regression water pH prediction, for highest values, MAE value-tanh.

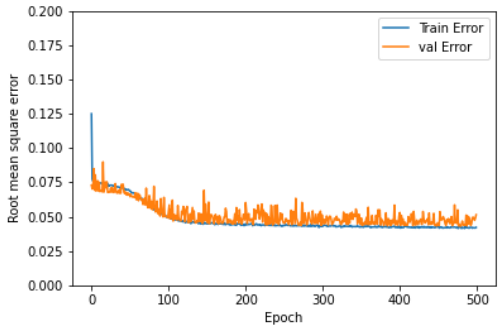


Figure H.7: Learning curve, regression water pH prediction, for lowest values, RMSE value-tanh.

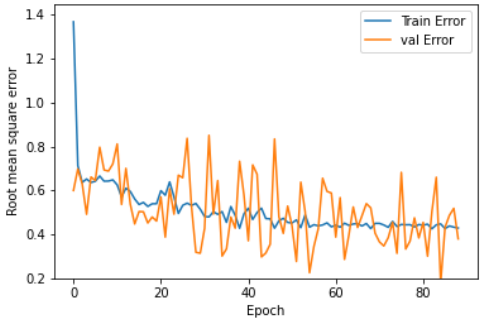


Figure H.8: Learning curve, regression water pH prediction, for highest values, RMSE value-tanh.

Analyzing the curves, for the lowest values of parameters, the number of epochs considering ideal is around 150 epochs. However, in for the highest values, the graphs are inconclusive, once the MAE and the RMSE show a high variation over epochs mainly for the validation data, here, the test dataset.

The graphs H.9, H.10, H.11, and H.12 show the same process explained before, using the sigmoid activation function.

Activation function - Sigmoid

Lowest

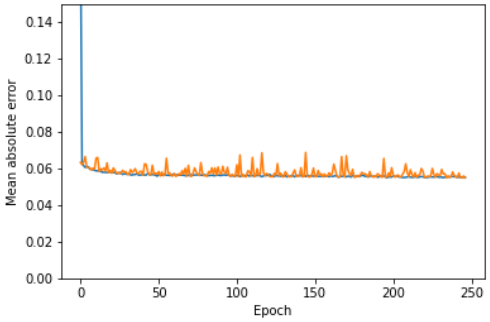


Figure H.9: Learning curve, regression water pH prediction, for lowest values, MAE value-sigmoid.

Highest

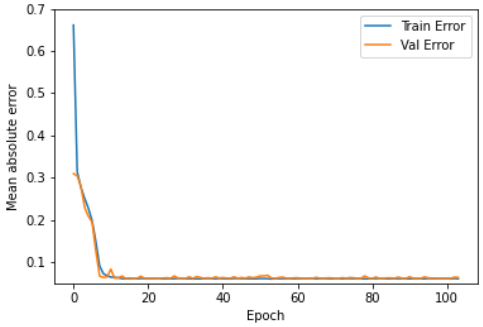


Figure H.10: Learning curve, regression water pH prediction, for highest values, MAE value-sigmoid.

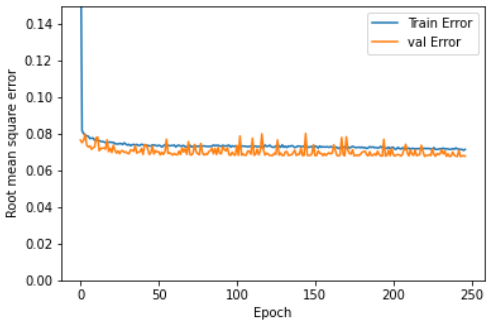


Figure H.11: Learning curve, regression water pH prediction, for lowest values, RMSE value-sigmoid.

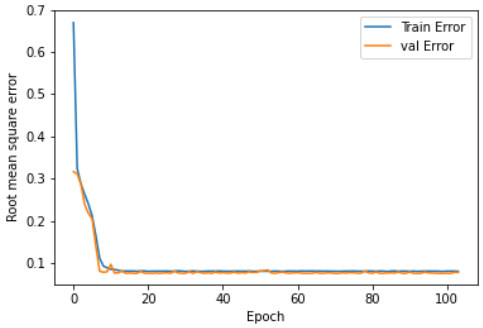


Figure H.12: Learning curve, regression water pH prediction, for highest values, RMSE value-sigmoid.

Here, for the lowest values of parameters, the MAE and RMSE values reach stabilized values fast. The same happens with the highest values, the number of epochs from which the value stop vary is around 10.

Finally, it is possible to conclude that the number ideal to train the model is 200 epochs.