VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Big Data Analysis in Intelligent Buildings within IoT

2020 José Luis Gordillo Relaño

I hereby declare that this master's thesis was written by myself. I have quoted all references I have drawn upon.

Ostrava, April 11, 2020 ……

I hereby agree to the publishing of the master's thesis as per s. 26, 22. 9 of the Study and Examination Regulations for Master's Degree Programmes at VŠB – Technical University of Ostrava.

Ostrava, April 11, 2020 .....

I want to thank all those people who have helped me in the development of this project since, without them, it would not have happened.

Especially to my professors from the University of Córdoba, to my family and finally to my friends, who have been essential support in all the development of my professional career.

Abstract

This bachelor project aims to detect the occupation of Intelligent Building (IB), and Smart Home, optimizing energy waste using data obtained from indirect methods. The leading cause of this project is that most of the software developed for this purpose is based on detecting the occupation of the room using invasive methods. Therefore, to determine the presence of people without being invasive, the use of indirect methods has been determined.

Therefore, it will determine the prediction of the CO2 waveform based on the use of sensors that measure indoor/outdoor temperature and relative humidity. To predict the occupancy, the method used is Support Vector Machine (SVM).

Keywords: Smart Home, prediction of room occupancy, Big data processing, presence of person monitoring, Activities monitoring with indirect methods, IoT

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project Description

The concepts Big Data and the Internet of Things (IoT) are different, but they could not exist without each other. Nowadays, it is more common to understand IoT as a data collection network obtained by sensors.

Nevertheless, due to the large amount of information that these sensors generate in time, and the speed at which it occurs, the concept of Big Data (BD) emerges, being capable of generating, according to IBM and CSA [5], more 2.5 quintillion bytes per day, to the point, that 90% of the world's data has been created during the last two years, is the value of said data, the most relevant aspect to study within Big Data in figure 1, we can see as a measure that increases the volume and complexity of the data, its marginal value affected due to the complexity of exploitation.

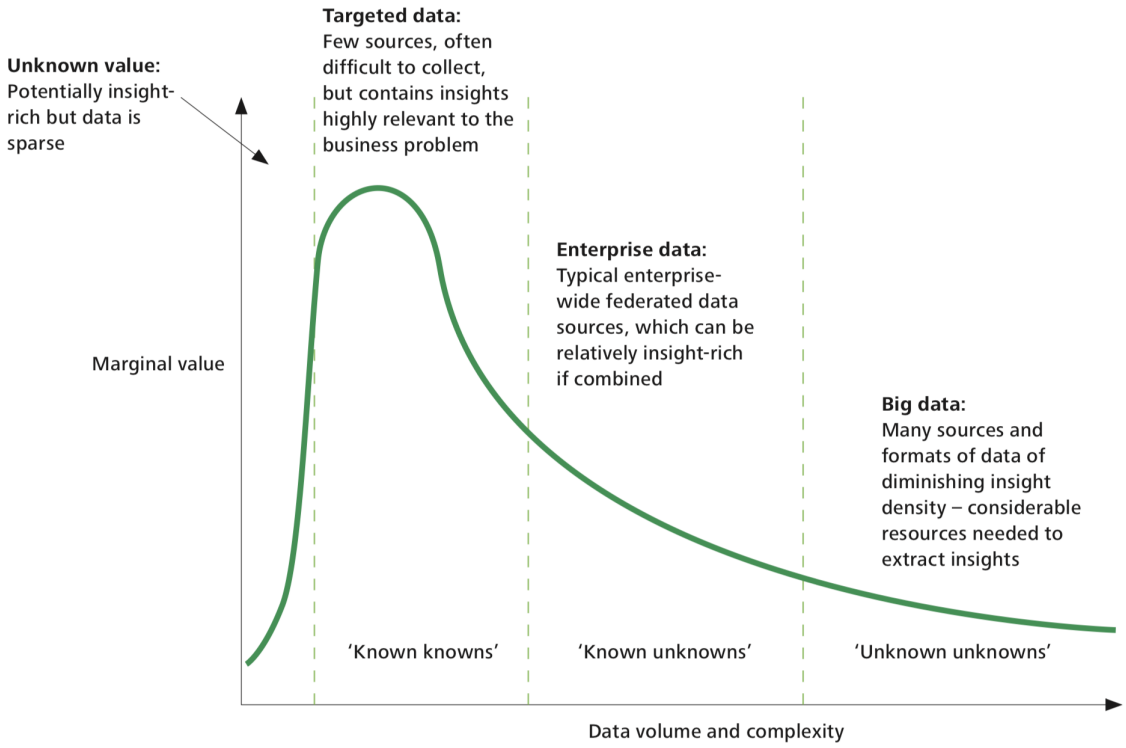


Figure 1: Volume and complexity of the data versus its value [25]

In this investigation, we will focus on the application of IoT and Big Data on the detection of the presence or occupancy of people in Smart Home or intelligent buildings. People monitoring or detection systems play a significant role in the possibility of energy-saving and consumption of this type of buildings [68] [25]. According to a study published by Cisco, the Internet of things will connect around 50 billion devices worldwide in 2020, generating data traffic, which should be saved, analyzed, and prepared for processing [19]. Therefore, we refer

to the treatment and analysis of vast data repositories, so immensely large that it is impossible to treat them with the tools of conventional databases and analytics.

The current data storage paradigm "cloud computing" and the various security concerns in IoT have led to the emergence of alternatives for data storage, organization, and processing, as well as technologies such as blockchain that provide decentralized management of private data, as well as a higher level of security in IoT [6] [57] [61].

Currently, most studies are focused on predicting the presence and number of people through the recognition of the human silhouette [47], as well as through motion detection sensors or even through technologies such as Wi-Fi, Bluetooth [53] [78], or images obtained from video cameras [41].

With this approach, an inconvenience arises, such as the privacy of people [1]. Since they would be continuously monitored, an alternative is proposed to detect if there is the presence of people or not in an IoT, such as the use of no invasive methods using the current infrastructure of IoT or Infrastructure Mediated Sensing (IMS) [26]. These methods will also be useful for the care of older people and the detection of possible falls or problems derived from age [63].

This investigation will focus on these kinds of methods to predict the occupation will be monitored in a non-intrusive way for the person. Through the use of the infrastructure already existing in the building, as well as temperature sensors, Co2, Heating, ventilation, air conditioning (HVAC), surveillance cameras, or consumes of gas, so that the user does not require software-installation [15] [39][8] [50] [66] [79].

So that for recognition of occupation or detection of people in Smart Home, mathematical experiments have been carried out. Some as Linear discriminant analysis (LDA), Classification and Regression Trees (CART), Random Forest (RF), decision trees, rule induction, k-means clustering, and K-nearest neighbor algorithm (KNN) with which promising results have been obtained [24].

1.2 Objectives

The objective of this thesis will be the design of a prediction method based on SVM, to determine the occupancy of the room, using non-invasive methods [23]. This project is based on the prediction of occupancy using data obtained by sensors such as internal temperature and external, level of Co2 in the room, relative humidity, and HVAC levels.

The objective to reach will be to obtain and implement a good prediction model that can decide the presence of people in the room with high accuracy using SVM. Besides, it will be able to reduce energy consumption and increase the energy savings of the Smart Home. In this way, the IoT or Smart Home [58] will be able to control and automate functions such as heating, ventilation, lighting.

1.3 Document structure

The present document has been structured in chapters. In order to offer a global vision of the document, the content of each of these chapters is listed below.

In chapter 1, the description of the project to be carried out is made, as well as the objectives of the work and the structure that follows from this document.

Chapter 2 describes the different phases of the development of the project, as well as a brief description of each of them.

In Chapter 3, a study of the theoretical and mathematical base related to the work carried out is carried out.

In Chapter 4, the entire work of processing the data obtained for the realization of the project is carried out. So that an informative report of the data set is made, explaining, in turn, the procedure performed for the preprocessing of said data

In Chapter 5, the analysis phase is explained in detail, and the design of the model, as well as the different techniques to be performed in order to optimize and improve the model.

In Chapter 6, the construction of the previously designed model is reported, checking and verifying that its operation is expected and correct.

In Chapter 7, the evaluation of the final model will be carried out, where the several tests will be carried out, and a set of results will be obtained that must be subsequently analyzed.

In Chapter 8, an analysis of the results obtained in the evaluation phase will be carried out.

In Chapter 9, the conclusions are drawn from the work are stated, and various functionalities and new lines of research are proposed that would improve and complement the implemented model.

2 Phases of Development of the Project

The following are the phases of which to be included in the process of project development, as is shown in the next figure 2.

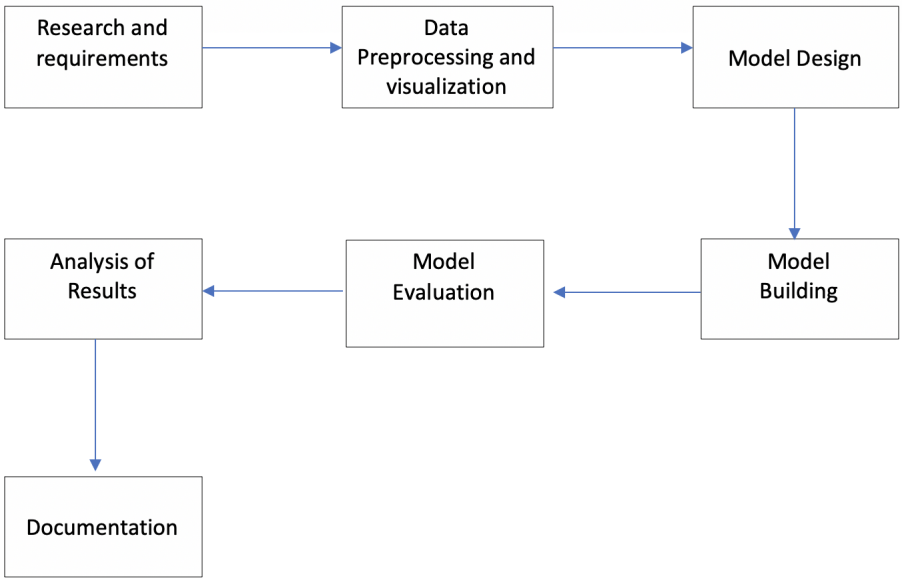


Figure 2: Phases of the project

2.1 Research and Requirements Analysis

In the first phase, the theoretical and mathematical content of the problem will be investigated proposed, as well as the documentation of the necessary software modules to determine the scope of the problem. The objective is to obtain the knowledge necessary for the correct development of the problem.

For this, the material available in the Building Control and Introduction to the Data Mining subjects, as well as the readings recommended by the professors of these subjects, should be consulted. Also in be investigated through the Internet, research, and scientific articles.

In addition to the theoretical content, the Python programming language should be investigated. An investigation will be carried out to determine the possible libraries that allow to carry out different operations, preprocess significant sources of data as well as the correct development of SVM. So that the objectives to fulfill the system to be developed are met.

2.2 Data Preprocessing and Data Visualization

"The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed or made more easily accessible" [62].

Data preprocessing is one of the most important tasks to perform. [37] This is because the data can be impure, that is, they can lead us to the extraction of patterns or rules that are not very useful, and maybe due to:

- Uncompleted data
- Data noise
- Inconsistent data

Data preparation can generate a smaller data set than the original [31], which can improve the efficiency of the Data Mining process that includes:

- Relevant data selection: eliminating duplicate records, eliminating anomalies
- Data Reduction: Selection of characteristics, sampling or selection of instances, discretization, and correlation coefficient.
- Recover incomplete information and outliers treatment.

As shown in figure 3, another of the most critical steps is the visualization of the data, since it offers us an easy way to understand how our data is initially distributed [76].

- Data visualization is a way of displaying complex data graphically.
- The graphics can be precise to locate the implemented models as planned.
- Allow greater ease to compare and interpret data, thus visualizing a large number of them quickly.
- It allows us to have a first global and fast image on how the data is distributed, as well as a time-saving [4].

In our case, it will allow us to choose what type of kernel to implement using SVM.

2.3 Model Design

Once the requirements analysis and preprocessing are finished, we will proceed to design each of the modules and software components of the system to be implemented during project development.

In the same way, they must safely devise the algorithms and mechanisms with which they will solve the problem, and that will be implemented in the software components. For this, a correct preprocessing of the data must have been carried out, as well as a previous study, thus determining which method to design is the most appropriate for our problem, no free lunch theorem [38] [80].

2.4 Model Building

The phase will proceed to implement each of the component's software obtained during the design phase and with which it is intended to give solution to the problem posed.

The Python programming language has been chosen, due to the facilities provided by the programming language itself as well as a large number of libraries that facilitate the processing of data and the implementation of sophisticated mathematical methods and machine learning techniques.

2.5 Model Evaluation

At this point, we will focus on the execution of the algorithm developed with the data obtained in order to obtain results that should be analyzed later. In this phase, it will also be advisable to compare the algorithm with others already implemented in order to know the performance of our algorithm.

2.6 Analysis of Results

This phase in which the operation of the developed system will be tested in order to find errors and improve the system.

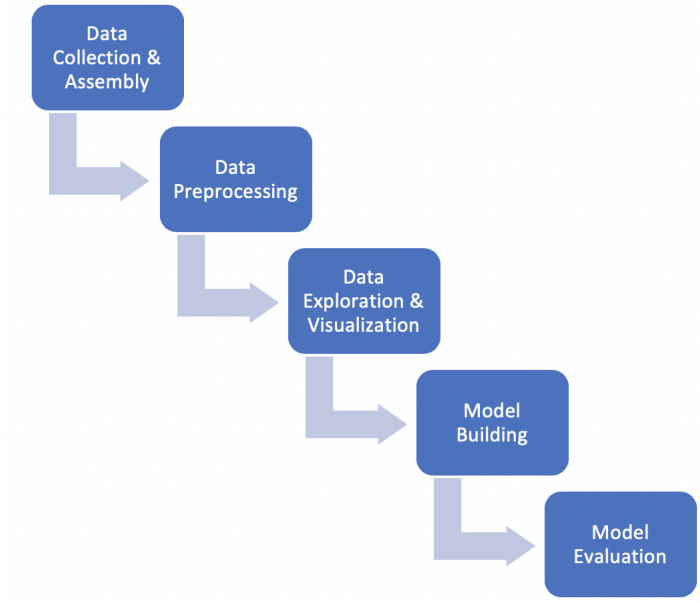


Figure 3: Steps in data manipulation

In turn, the results of the SVM algorithm to be implemented will be obtained, having used the dataset provided by the project tutor. This output will be analyzed, as well as conclusions will be given for them.

3 Research and Requirements Analysis

3.1 Support Vector Machines Maths

"Support vector machines are learning system that uses a hypothesis space of linear functions in a dimensional feature space which is used on classification and regression problems" [22]

Support Vector Machines SVM was developed in the 1990s, within the field of computational science. Although it was initially developed as a binary classification method, its application has extended to multiple regression and classification problems, obtaining excellent performance in the machine learning field.

Support vector machines are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory.

Support vector machines are based on the Maximal Margin Classifier, which, in turn, is based on the concept of the hyperplane.

3.2 Linear Separability

Linear separability is a property of a set of two points. These sets of points are linearly separable if there is at least one line in the plane that is capable of dividing these sets being classified into two classes, as shown in the figure 4.

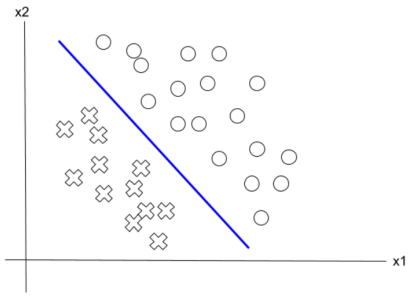


Figure 4: Linearly separable data

Linear separability refers to the fact that classes of patterns with an n-dimensional vector can be separated with a single decision surface [54].

$$\mathbf{x} = (x_1, x_2, ..., x_n) \tag{1}$$

This idea can be generalized to higher-dimensional if a hyperplane replaces the line.

"The optimal hyperplane separates this set of vectors if it is separated without error, and the distance between the closest vector and the hyperplane is maximal" [82]. To describe the separating hyperplane, let us use the following form [20]:

$$y_i(\omega \bullet \mathbf{x}_i + b) - 1 \geq 0 \ \textit{for}\ y_i = +1, -1 \tag{2}$$

It can be proven that the increase in the size of the entry space makes it easier to achieve the linear separability of the data. It is the theorem of Cover [21].

Theorem 1 *The probability that two classes are linear separable approaches 1 when the space dimension of characteristics d samples to infinity and the number of samples N grows bounded by 2 (d + 1).*

So increasing the number of features is more likely that the classes are linearly separable [65]

3.3 Hyperplane and Maximal Margin Classifier

In n$-$*dimensional* space, a hyperplane is defined as a plane sub-space and does not have to go through the origin of dimensions $n-1$.

In a two-dimensional space, the hyperplane is a 1$-$*dimension* subspace, that is, a straight.

In three-dimensional space, a hyperplane is a two$-$*dimensional* subspace; however, for dimensions $n > 3$, the concept of sub-space with $n-1$ dimensions is maintained.

The mathematical definition of a hyperplane in the case of two dimensions, the hyperplane is described according to the equation of a straight-line equation [32]:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0 \tag{3}$$

Given the parameters, β_0, β_1, and β_2, all pairs of values $x = (x1, x2)$ for which equality is met are hyperplane points. This equation can be generalized for $n-$ *dimensions* [32]:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n = 0 \tag{4}$$

Similarly, all the points defined by the vector $(x = x_1, x_2, ..., x_n)$ that meet the equation belong to the hyperplane. The definition of hyperplane for perfectly linearly separable cases results in an infinite number of possible hyperplanes, which makes it necessary to select one of them as an optimal classifier, as shown in figure 5.

The solution is to select the maximal margin hyperplane as the optimal classifier, which corresponds to the hyperplane that is furthest from all training patterns, as shown in figure 6.

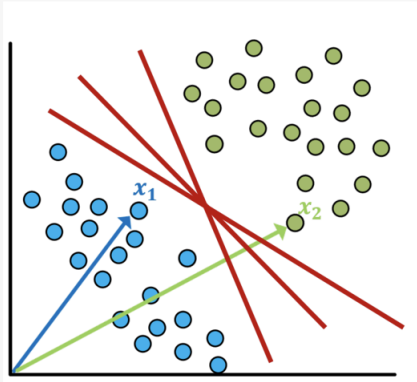


Figure 5: Three possible hyperplanes for data linearly separable.
[16]

To obtain, it is necessary to calculate the perpendicular distance of each observation to a certain hyperplane, and the smaller of these distances or margin determines how far the hyperplane of the training observations is.

The maximal margin hyperplane is defined as the hyperplane that achieves a higher margin, that is, that the minimum distance between the hyperplane and the observations is as considerable as possible.

However, it is not possible to apply it, since there would be infinite hyperplanes against which to measure the distances [69].

The image 6 shows the maximal margin hyperplane for a set of training data. The three equidistant observations regarding the maximal margin hyperplane are found along the dashed lines that indicate the width of the margin.

These are known as support vectors since they are vectors in n-dimensional space that define the maximal margin hyperplane. Any modification in the support vectors implies changes in the maximal margin hyperplane.

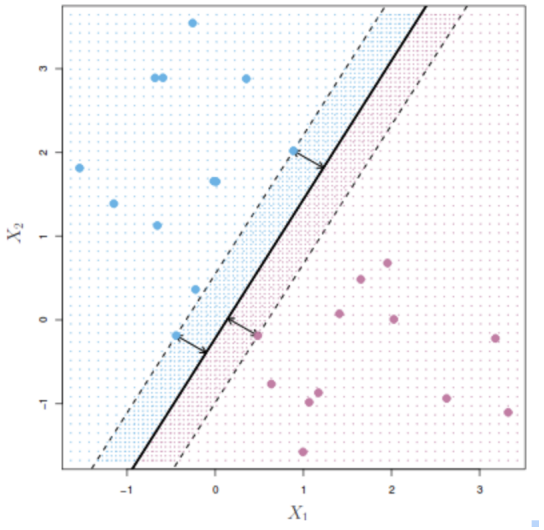


Figure 6: Maximal margin hyperplane [33]

3.4 SVM Optimization Problem - Soft Margin SVM

The main reason to use Soft Margin SVM is that *"the problem with Hard Margin SVM is that it does not tolerate outliers. It does not work with non-linearly separable data because of outliers. For the optimization problem to be solvable, all the constraints have to be satisfied"* [28].
The Maximal Margin Classifier has little practical application since there are rare cases in which the classes are entirely and linearly separable. Even if these conditions are met, in which there is a hyperplane capable of correctly separating the patterns into two classes, there are two drawbacks:

- Since the hyperplane has to separate observations correctly, and it is susceptible to variations in the data. Including a new observation can lead to huge changes in the separation hyperplane (low robustness).
- That the maximal margin hyperplane fits perfectly to the training observations to separate them all correctly usually leads to overfitting problems.

Because of these problems, it is preferable to create a hyperplane-based classifier that is more robust and has fewer overfitting problems.

For this, we will use soft margin classifiers, thus obtaining a plan that almost separates the classes, but allowing it to make some mistakes. In this way, instead of looking for the broadest possible classification range, specific observations are allowed to be on the wrong side of the margin or even of the hyperplane [44] [48].

3.5 Support Vector Machines

SVM obtain excellent results when the separation between classes is approximately linear, being the expansion of dimensions of the original space an alternative for data sets where the separation is not lineal. SVM uses a nonlinear function on attributes of the initial space of variables, using the following function that passes from a one-dimensional space to a two-dimensional space [33].

$$\phi(x_1) = (x_1, x_1^2) \tag{5}$$

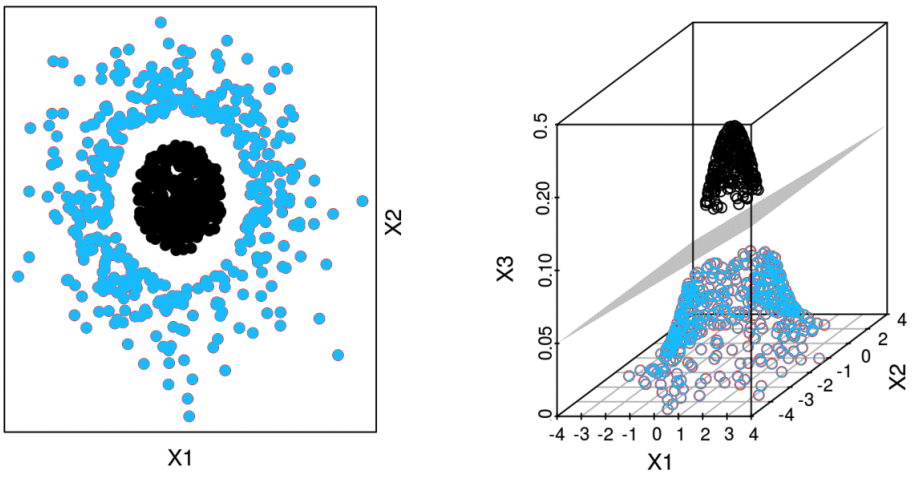


Figure 7: Three-dimensional linear separation

The fact that the data is not linearly separable in the original space does not imply that it is not in a space with a higher number of dimensions. In this way, we transform a group of data whose two-dimensional separation is not linear into a separable set by adding a third dimension, as shown in figure 7.

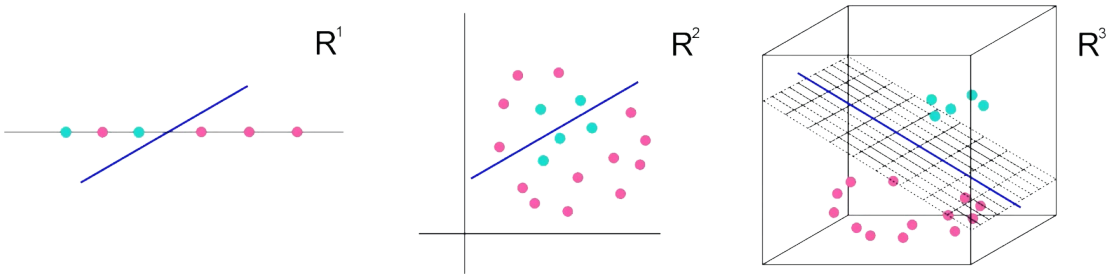


Figure 8: How it works SVM

As we can see in the figure 8, we have a set of data in R^1, which is not possible to separate linearly, so the number of dimensions is increased to R^2. However, we can verify that said data

cannot be separated linearly in the space of dimension R^2 so that the number of dimensions is again increased to space R^3, where now the data set is entirely separable.

3.6 Projection to large spaces

The dimension of a data set can be transformed by combining or modifying any of its dimensions using functions, also called kernel trick. [3].

$$x_i \epsilon R^d \rightarrow \varphi(x_i) \epsilon R^r, r >> d \tag{6}$$

The kernel trick is a mathematical function $\phi(x_1)$ that allows transforming the linearity of a hyperplane, that is, create a new dimension in which a hyperplane can be found that allows classes to be separated as is shown in figure 8, being able to find the following types:

- Linear
- Polynomial
- Radial Base Function
- Sigmoid

The main requirement to define a function of kernel type is that there is a corresponding transformation, such that the kernel function, calculated for a couple of vectors, be equivalent to the scalar product in the transformed space. However, not every function k (x, y) can be used as a core function since it must satisfy the condition of Mercer's condition [45].

Theorem 2 *A symmetric function K(x,y) can be expressed as an inner product*

$$K(x, y) = \phi(x), \phi(y) \tag{7}$$

for some ϕ if and only if K(x,y) is positive semidefinite, i.e.

$$\int K(x, y) \cdot g(x) \cdot g(y) \partial x \partial y \quad \geqslant 0 \quad \forall g \tag{8}$$

or, equivalently:

$$\begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots \\ K(x_2, x_1) & \ddots & \\ \vdots & & \end{bmatrix}$$

is psd for any collection $x_1 \dots x_n$

3.6.1 Linear Kernel

If a linear Kernel is used, the Support Vector Machine classifier obtained is equivalent to the Support Vector Classifier, obtaining the next equation [48].

$$K(x, x') = x \cdot x' \tag{9}$$

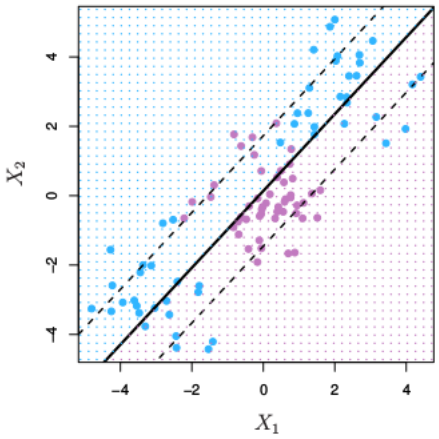


Figure 9: SVM with linear kernel [33]

3.6.2 Polynomial Kernel

When d = 1 is used and c = 0, the result is the same as that of a linear kernel. If d> 1, non-linear decision limits are generated, increasing non-linearity as d increases. It is not usually recommended to use values of d over five due to overfitting problems [48].

$$K(x, x') = (x \cdot x' + c)^d \tag{10}$$

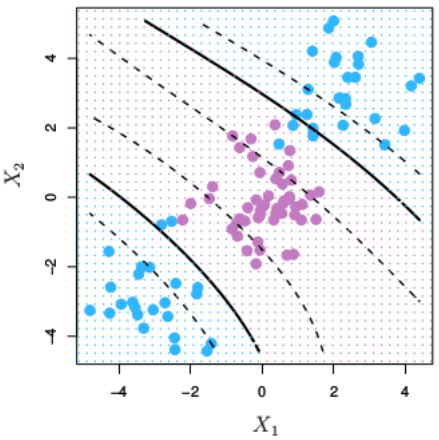


Figure 10: SVM with polynomial kernel [33]

3.6.3 RBF Kernel

It is an actual function whose value depends only on the distance from the origin or from the distance to some center. Each point of the training set creates its bell of Gauss, and the complete form is the sum of the bells of Gauss [33].

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) = exp(-\gamma \|x - x'\|^2) \tag{11}$$

The choice of gamma is critical for SVM performance.

Gamma Value	Points Affected
Low, High sigma	May be far from training examples
High, Low sigma	It should be close to the training examples

Table 1: Kernel RBF Parameters

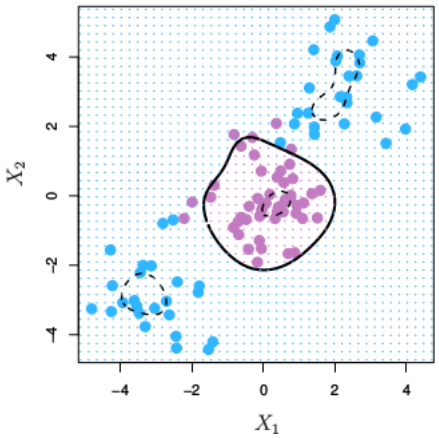


Figure 11: SVM with RBF kernel [33]

The value of γ controls the behavior of the kernel, when it is minimal, the final model is equivalent to that obtained with a linear kernel, as its value increases, so do the flexibility of the model.

The behavior of the model is very susceptible to the gamma parameter because if gamma is too high, the radius of the area of influence of the support vectors only includes the support vector itself, and no amount of regularization with C can avoid overfitting [60].

When gamma is very small, the model is too limited and cannot capture the complexity or "shape" of the data so that the resulting model will behave similarly to a linear model.

Lastly, for intermediate values of gamma, we obtain models of equal performance when C becomes very large.

3.7 KNX

"KNX Association is the creator and owner of the KNX technology – the worldwide standard for all applications in home and building control, ranging from lighting and shutter control to various security systems, heating, ventilation, air conditioning, monitoring, alarming, water control, energy management, smart metering as well as household appliances, audio/video and lots more." [11]

One of the significant advantages of this system is that it has a distributed architecture; that is, a central controller is not needed to control the installation since each element of the system has its intelligence. They communicate with each other, which allows quick modification of the installation. [73]

The study "Potential for energy savings by modern installation systems," [10] carried out by the Institute for Building Automation Systems of the University of Biberach, demonstrates that a KNX-based bus system can achieve a global saving of over 50% as shown in figure 12.

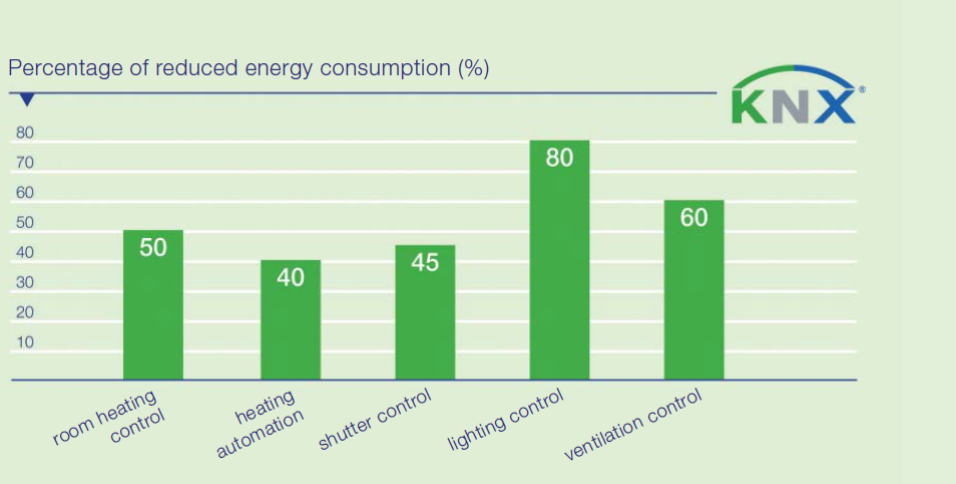


Figure 12: KNX reduced energy comsumption [10]

The implementation of the KNX standard, together with IoT [29], also provides health care benefits, such as accessibility, availability, constant monitoring and specialized care for people who require special services, as well as can be used for fields such as student education [74] in the realization of projects and introduction to Smart Home and IB.

3.8 KNX to IoT connectivity

KNX is a bus system developed for the control and automation of homes and buildings. All devices use the same means of communication and can exchange information via the shared bus.

Each KNX device is a physical object, which, using an individual address and a serial number, allows the device to be identified with interest. These devices can connect to the network using Internet Protocol (IP), PowerLine (PL), Twisted-pair cable (TP), and Radio Frequency (R-F). In this way, they receive, send, process, and exchange data with each other through a KNXnet/IP router. [12]

This study will focus on occupancy monitoring in an intelligent building within IoT [9] so that the prediction will be determined by the data collected through KNX sensors and devices. However, not only can KNX technology be used to detect people in a space, but it is also possible to interconnect with other devices that connect to the network, thus allowing their management and automation, from light bulbs, even heating and ventilation systems, among others, thus increasing comfort and quality of life in the home or office [2] [49].

3.9 How to solve IoT Connectivity in Intelligent Buildings

A smart building has already been defined as a building that intelligently and remotely manages its automation systems to improve comfort, maintenance, accessibility, health, and safety, thus improving the quality of life for the user who uses this type of buildings and saving energy costs [72]. The fundamental concept is to allow the devices to interact with each other and, in turn, with the central control, generally in the cloud, that controls them.

These devices must be unequivocally identifiable within the network so that communication between them is secure through the use of standardized protocols, the following article [46] explains in more detail both the types of protocols available, and the alternatives of technology providers in Cloud.

Currently, there are many solutions to resolve connectivity; however, not all are adaptable to any scenario, intending to solve the needs of the objects connected to that network (IoT). As mentioned above, due to the large amount of data generated by the building, it must be implemented better, it is possible to analyze the usefulness of such data in the field of the IoT, as well as, a higher level of both physical and logical security must be implemented, with the main objective of avoiding possible threats, since with these data, it is possible to conclude the users of the building, as well as the behavior of the building.

3.10 Machine Learning as a Service

By using machine learning cloud services, it is possible to build working models, thus obtaining multiple and valuable perspectives of the predictions obtained with a small team. Among the machine learning platforms that will be addressed in the development of this project, possible decisions that will affect the choice of which infrastructure to take will be considered [18].

First, we find the terminology of machine learning as a service [7] (MLaaS), which we can define as various platforms based on the cloud or cloud computing, and which cover a large part of typical infrastructure problems.

In this way, it is possible from the prediction or classification results obtained to combine them with the internal IT infrastructure, through the use of API REST, such as through data processing or optimization and evaluation of learning models. This is where Cloud-based Machine Learning services come from, being able to find IBM Watson, Azure Machine Learning, and Google Cloud, being the leading cloud computing services for Machine Learning as Service.

3.10.1 What is Azure?

Microsoft Azure Machine Learning Studio is a tool that allows us to create, test, and deploy predictive machine learning solutions straightforwardly and visually. The main interface of Microsoft Azure ML Studio presents a graphical testing environment where different existing modules can be combined to create tests or experiments in order to analyze data, as well as perform different machine learning models, thus allowing the obtaining of results and his subsequent conclusions [55].

The collection of predefined modules is extensive, from data processing and conversion to machine learning algorithms for different tasks such as clustering, classification, or regression. In this way, it is possible to implement complex algorithms in a short time and without the need to code them.

However, it is also adapted for experienced users, who do know data analysis, allowing the modification of Python and R scripts, as well as testing their scripts, thus making a more flexible service for the type and experience of the user. However, as a drawback found, are the input and output restrictions of the script modules, as well as the support of frameworks for Machine Learning, such as TensorFlow or Keras.

3.10.2 What is Google Cloud?

Google Cloud offers ML capabilities for clients with a different set of tools. First of all, it provides different APIs to perform simple automatic operations such as ML classification routines in the areas of language recognition, vision recognition, voice, and translation, being very easy to use for any user regardless of their previous experience. However, it presents a fundamental problem, which is its rigidity, because the tool's handling and adaptation options for a specific need are shallow.

On the other hand, it also offers a machine learning engine, oriented to data experts and with more experience in the field of artificial intelligence, allowing the user to create the models, as well as train and test using the cloud resources of Google, being only limited by the user's knowledge.

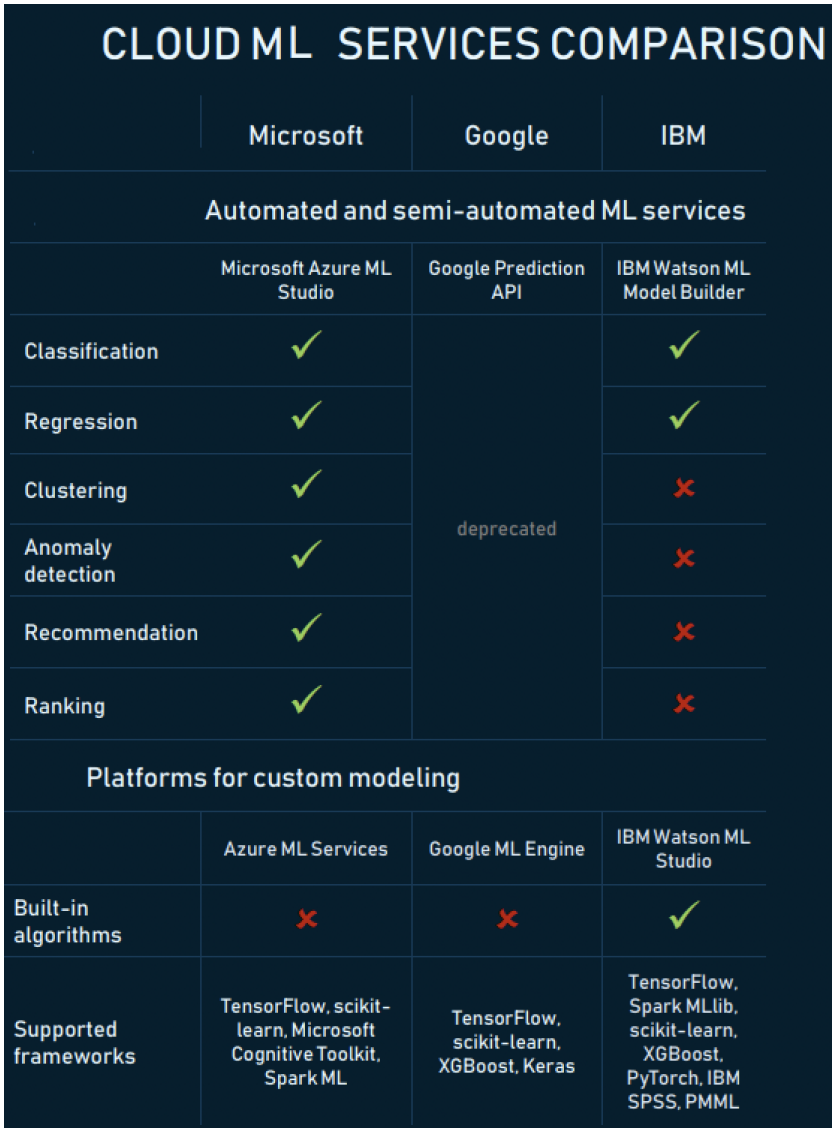


Figure 13: Comparison of Machine Learning as a Service [7]

3.10.3 What is IBM Watson?

IBM Watson Machine learning is the tool offered by IBM, which allows the creation and implementation of learning models. To do this, it allows the use of our data to create, train, and implementing models of machine learning and deep learning. In turn, it gives the possibility of creating different predictive models and subsequently comparing their results, thus executing automated experiments and learning through the use of the Watson Machine Learning engine, thus allowing, through scripts in languages such as Python and R, the creation of said models for expert users [40].

However, it also offers solutions for inexperienced users, providing learning models using visual tools, making it more widely used for all types of users, as well as facilitating the processing and treatment of data, being very useful in the IoT area, where the concept of Big Data arises.

In the figure 13, It is possible to see some of the fundamental characteristics offered by the services mentioned in this research, so the decision to use one or the other platform will depend on the needs that need to be solved, however, for this research, Azure and IBM services are far above what is offered by Google in its most basic plans.

3.11 Implementation of KNX within IoT

This chapter will detail the theoretical operation and implementation of communications for a KNX home automation system together with a previously discussed machine learning platform, including also a brief review of the standard on which this project is based, MQTT.

3.11.1 What is MQTT?

MQTT is an international standard (ISO / IEC PRF 20922) [42] that defines a messaging protocol that implements a pattern of publish-subscribe and built on the TCP / IP protocol. It was designed for the connection of remote devices that required a communication system that was not very demanding neither with the microcontrollers of the connected sensors nor with the necessary bandwidth to send a basic message.

MQTT protocol is based on a publish-subscribe paradigm, any connection in MQTT involves two different kinds of agents, MQTT clients and an a central server or broker that controls the dissemination of these messages.

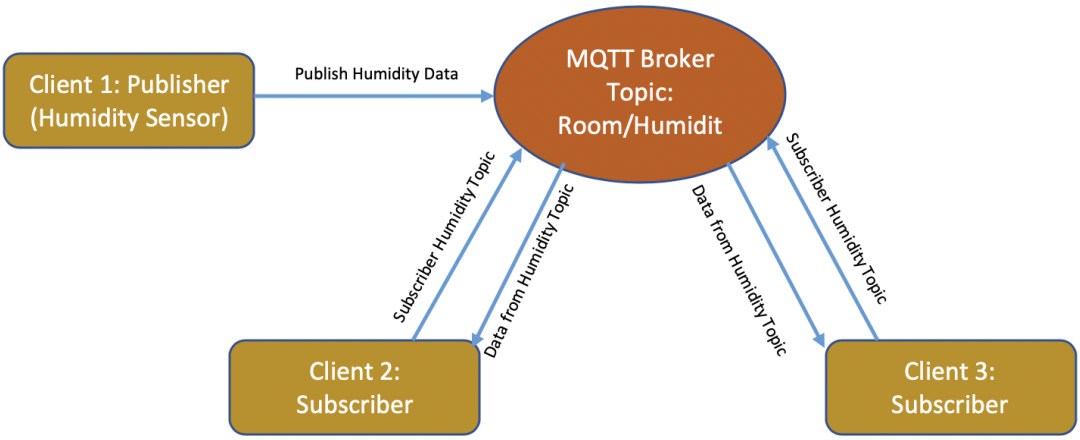


Figure 14: Simple MQTT transmission
[77]

The behavior that defines the protocol can be summarized in the following steps as is shown in above 14:

- One of the clients connected to the server sends a message / publishes application messages with a specific topic, and MQTT subscribers are requesting application messages. The client publisher, can be any KNX device, in the previous example, it has been specified that the publisher will be a humidity sensor.
- The server receives the message and the topic.
- The MQTT broker consults in its internal list of the devices that they are subscribed to that specific topic.
- The server forwards that message to all clients that are subscribed to that topic.

The use of this protocol frees the device from sending a large number of messages since the broadcast is carried out by the server. The server can be a local machine or a server, thus having a higher computing capacity.

3.12 Theoretical Implementation

To carry out an implementation of the proposed system theoretically, the following elements are proposed: Helium Atom as a gateway between KNX devices to Google Cloud / Azure or IBM Watson. Cloud functions where, after receiving the data obtained by the KNX temperature, humidity sensors, it will execute our SVR prediction model, to predict and obtain the CO2 level, and then carry out possible actions, such as, control of opening and closing windows or turn on/off ventilation system.

3.12.1 What is Helium?

Helium was founded in 2013 by Shawn Fanning, Amir Haleem, and Sean Carey, with a mission to make it easier to build connected devices.

In order to build the world's first peer-to-peer wireless network, Helium has gathered a team with a diverse, yet complementary, skillset within radio and hardware, manufacturing, distributed systems, peer-to-peer and blockchain technologies [36]

Helium makes use of blockchain technology intending to facilitate wireless connections. The Helium access point is entirely different from a standard mobile access point, such as that of a mobile phone or a computer [51].

Furthermore, it is that thanks to its "LongFi" technology [75], it is capable of extending nearly 200 times more than a standard WiFi connection, being able to cover large surfaces such as cities, using between 50 and 100 access points, thus allowing more significant growth in cities smart.

Helium Atom is an easy way to connect sensors to channels in the cloud, as it reduces the amount of work in case the expert needs to use cloud services with their integrated devices.

3.12.2 Implementation

To work with Helium Atom, we will use the Python programming language, through which we will use the different libraries, as well as the instructions for its installation, can be found in the Helium repositories [70].

The next step would be to start working with KNX; for this, we have the option of making use of an API, which allows us a gateway with KNX. For this, it is possible to make use of the bobaos project [14], which is intended to solve a KNX problem by using JavaScript, making the connection through the serial port, thus working with KNX data points 15.

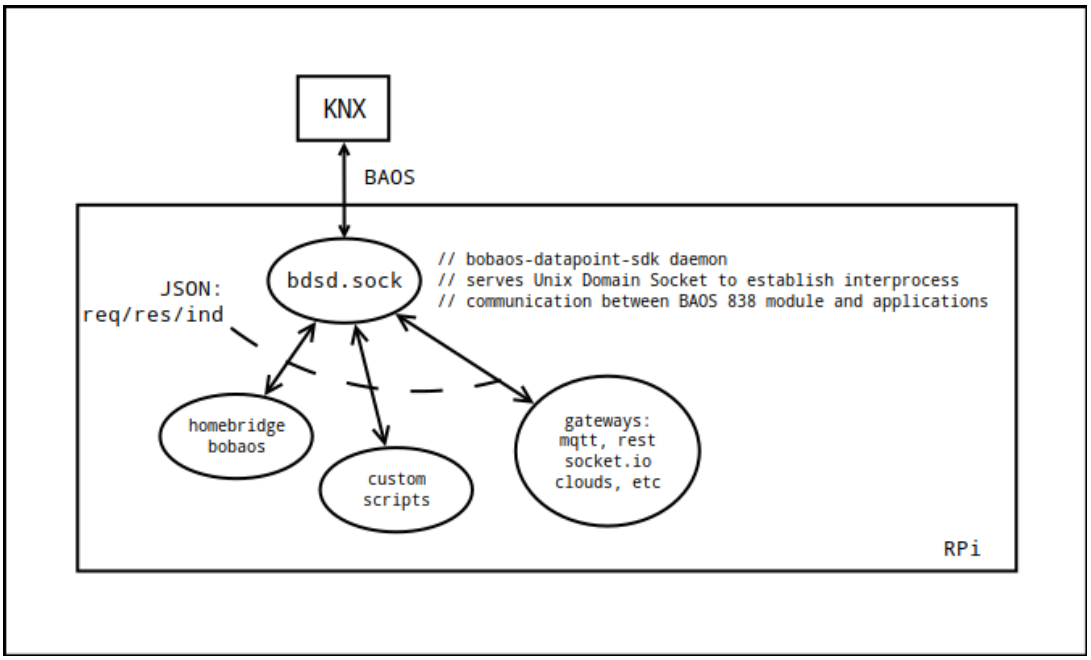


Figure 15: Bobaos connection Scheme
[14]

The next step is to connect to a cloud service, such as the Google Cloud service, which enables full integration with Helium software. In addition to through the Firebase functions, they allow the implementation of the project in a short space of time and in a straightforward way even for inexperienced users, since through the Google IoT Core channel controller, we can manage Google Cloud Pub/Sub triggers, through the methods already implemented [34].

The next step would be the implementation of our already trained model in the google cloud due to Firebase platform allows to implement authorization, information storing by using firestore / storage, machine learning my ML Kit.

So that when data is sent by our KNX sensors, related to the attributes studied, such as indoor/outdoor temperature, as well as humidity, our model predicts the CO2 level, and therefore automatically performs the tasks assigned to said CO2 level (ppm).

4 Data Preprocessing and Visualization

4.1 Libraries

In carrying out this project, both own functions and functions already included in the different libraries included in Python have been used, the most important being explained below:

- **Pandas**: Pandas is a software library written as a NumPy extension for data manipulation and analysis for the Python programming language. In particular, it offers data structures and operations to manipulate number tables and time series.
- **Numpy**: NumPy is a Python extension, which adds more support for vectors and matrices, constituting a library of high-level mathematical functions to operate with those vectors or matrices.
- **Sklearn**: Scikit-learn is a free software machine learning library for the Python programming language; it includes various algorithms of classification, regression, and analysis of groups among which are support vector machines, random forests, Gradient boosting, K-means and DBSCAN.
- **Matplotlib and Seaborn**: Matplotlib is a library for generating graphics from data contained in lists or arrays in the Python programming language and its mathematical extension NumPy. Seaborn is a Python data visualization library based on Matplotlib.
- **Plotly**: Plotly's Python graphing library makes an interactive, publication-quality graph.

4.2 Processing of all datasets

The first step has been the visualization of the obtained datasets. For this, firstly, the number of patterns of said datasets will be visualized, thus every how often the data is extracted, as shown in figure 16

```
File: co2 april.xls          ⟶ Number of Patterns: 32214
File: humidity april.xls     ⟶ Number of Patterns: 32554
File: temperature1 april.xls ⟶ Number of Patterns: 32495
File: temperature2 april.xls ⟶ Number of Patterns: 32171

File: CO2 march.xls          ⟶ Number of Patterns: 25409
File: humidity march.xls     ⟶ Number of Patterns: 25603
File: temperature1 march.xls ⟶ Number of Patterns: 25618
File: temperature2 march.xls ⟶ Number of Patterns: 25398
```

Figure 16: Number of patterns per file

It is appreciated that the number of patterns is different in datasets, and this is because there are times when sensors get data twice or even three times per minute when it should be a measurement every minute.

These individual data sets, two ways of working have been proposed, the first consisting of adding all the rows of the date attribute, which are not found in the dataset, and later joining the rest of the attributes using the date as a union component.

However, after carrying out this process, together with the tests of the subsequent model, better data has been obtained by joining all the files for each month into a single one and subsequently carrying out their processing, so the latter procedure is selected for the study.

The individual data sets have all been merged into a single file, making it easier to process such data further.

The implementation of the SVM algorithm will be carried out after a previous study of how the dataset data is distributed, this being a crucial step. This is because the visualization of said data will help in the selection of the type of kernel to use in the implementation of SVM.

Therefore, a study has been carried out using the different types of graphics available, thus facilitating the understanding of how the dataset data is distributed. To finish, the attributes with which we work in our dataset will be detailed:

- **Indoor CO2**: Measuring Range: 0 ... 2000 ppm, Accuracy at 25 ° C and 1013 mbar < 50 ppm + 2%, Temperature Dependency 2 ppm CO2/°C.
- **Indoor and outdoor temperature**: Accuracy at 20°C, Measuring range: 0 to 50 ° C.
- **Indoor relative humidity**: Type of measurement Capacitance, Measuring range: 0 to 100% RH, Accuracy at 20 ° C, 3% RH.

4.3 Load the dataset

[3]:
```
#reading the CSV file into pandas dataframe
data = pd.read_csv("file_output/april_output.csv", na_values =␣
↪missing_values, sep=",", date_parser=lambda x: datetime.strptime(x, '%d.
↪%m.%Y %H:%M:%S'))
# Fill all missing values to avoid error in displot
data['CO2'] = data['CO2'].fillna(method ='ffill')
data['Temp2'] = data['Temp2'].fillna(method ='ffill')
data['Temp1'] = data['Temp1'].fillna(method ='ffill')
data['Humi'] = data['Humi'].fillna(method ='ffill')
```

[8]:
```
#Check top few records of the dataset
data.head()
```

[8]:

	Date	Temp1	Temp2	Humi	CO2
0	1.4.2019 1:59:59	24.66	23.3	27.84	424.96
1	1.4.2019 2:00:59	24.64	23.3	27.84	424.96
2	1.4.2019 2:01:59	24.66	23.3	27.84	422.72
3	1.4.2019 2:02:59	24.64	23.3	27.84	421.76
4	1.4.2019 2:03:59	24.64	23.3	27.84	419.84

- It shows that there are four independent variables (Date, Temp1, Temp2, Humi) and one dependent variable (CO2).
- All the records are numeric but Date attribute.

[9]:
```
#Check the last few records of the dataset
data.tail()
```

[9]:

	Date	Temp1	Temp2	Humi	CO2
35413	2019-04-25 16:13:00	27.64	26.9	34.9	406.72
35414	2019-04-25 16:14:00	27.62	26.9	34.9	405.76
35415	2019-04-25 16:15:00	27.62	26.9	34.9	406.72
35416	2019-04-25 16:16:00	27.62	26.9	34.9	408.00
35417	2019-04-25 16:17:00	27.62	26.9	34.9	404.80

4.4 Exploratory data quality report April Dataset

4.5 Univariate analysis

4.5.1 Data Types and Description

[10]:
```
#To show the detailed summary
data.info()
```

```
RangeIndex: 32495 entries, 0 to 32494
Data columns (total 5 columns):
Date     32495 non-null object
Temp1    32495 non-null float64
Temp2    32171 non-null float64
Humi     32495 non-null float64
CO2      32214 non-null float64
dtypes: float64(4), object(1)
memory usage: 1.2+ MB
```

- It gives the details about the number of rows (32495), the number of columns (5), data types of information, all columns is float type.
- Memory usage is 1.2 MB+. Also, there are no null values in the data.

[11]:
```
# Data types information
data.dtypes
```

[11]:
```
Date      object
Temp1    float64
Temp2    float64
Humi     float64
CO2      float64
dtype: object
```

- It gives the data types of each column of the dataset
- It is observed how all the attributes are of type float, except the date that is of type Object since it is in date and time format.

[13]:
```
#Analyze the distribution of the dataset
data.describe().T
```

[13]:

	count	mean	std	min	25%	50%	75%	max
Temp1	32495.0	26.281	1.247	22.32	25.22	26.48	27.44	29.24
Temp2	32171.0	25.002	1.202	20.90	24.10	25.20	26.10	27.50
Humi	32495.0	27.184	2.895	17.65	26.67	26.67	27.84	60.00
CO2	32214.0	538.475	157.487	342.72	427.84	481.92	596.80	1332.48

In the figure below, 17, we can see how the Temp1, Humidity, and CO2 data are distributed throughout April.

In these graphs, the X-axis represents the date range from the beginning to the end of April.

On the other hand, the Y-axis describes both the percentage rH(%) of relative humidity in the air, as well as the temperature (ºC) both outdoor and indoor, and finally, the CO2 concentration (ppm) obtained in the dataset.

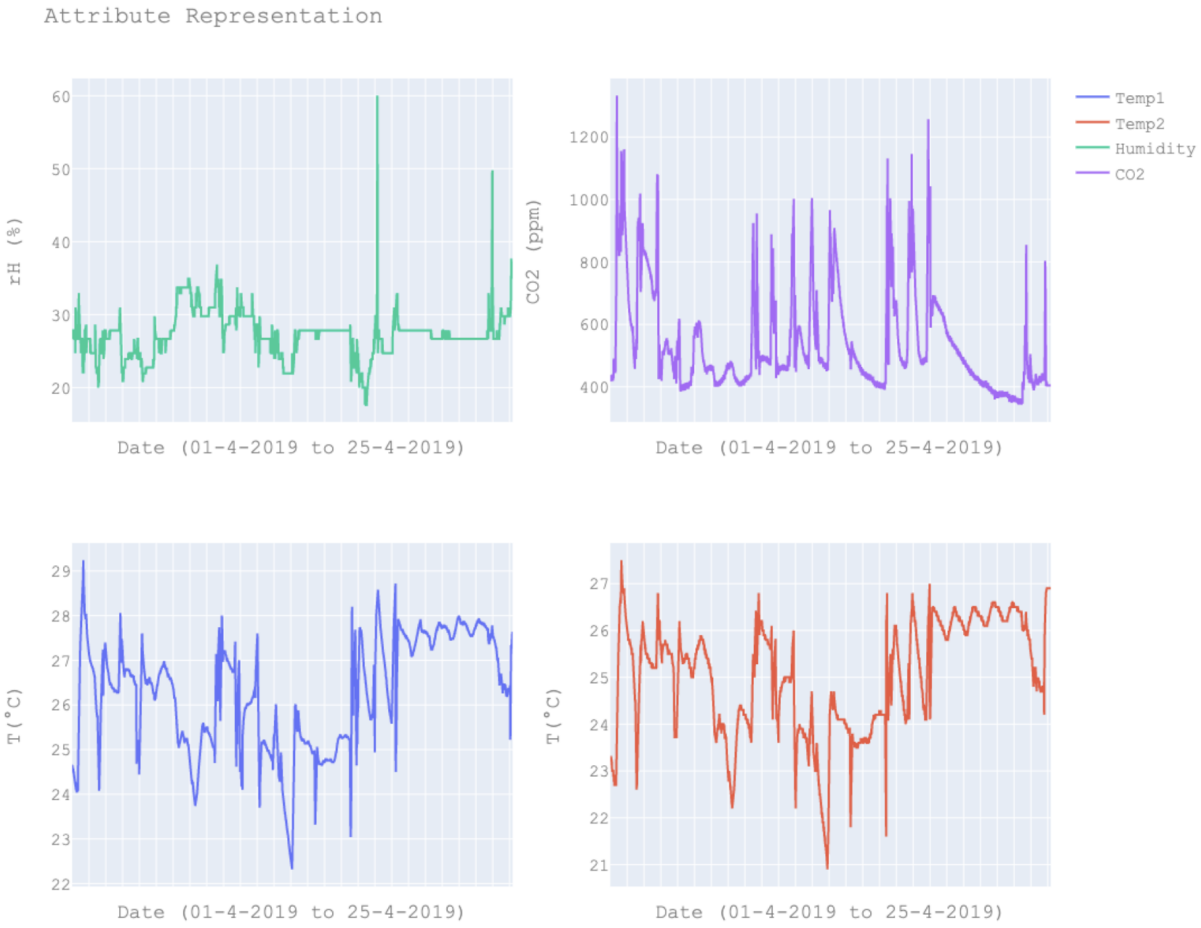


Figure 17: Attribute distribution

- It gives the descriptive statistics (mean, median, mode, percentiles, min, max, standard deviation) and count of the columns of the dataset.

4.5.2 Description of independent attributes

Temp1

Range of values observed

[15]:
```
print('Range of values:', data['Temp1'].max()-data['Temp1'].min())
```

```
Range of values: 6.919999999999998
```

Central values

[16]:
```
print('Minimum temp1:', data['Temp1'].min())
print('Maximum temp1:',data['Temp1'].max())
print('Mean value:', data['Temp1'].mean())
print('Median value:',data['Temp1'].median())
print('Standard deviation:', data['Temp1'].std())
```

```
Minimum temp1: 22.32
Maximum temp1: 29.24
Mean value: 26.28115094629943
Median value: 26.48
Standard deviation: 1.2470823282996288
```

Quartiles

[17]:
```
Q1=data['Temp1'].quantile(q=0.25)
Q3=data['Temp1'].quantile(q=0.75)
print('1st Quartile (Q1) is:', Q1)
print('3st Quartile (Q3) is:', Q3)
print('Interquartile range (IQR) is', Q3-Q1)
```

```
1st Quartile (Q1) is: 25.24
3st Quartile (Q3) is: 27.44
Interquartile range (IQR) is 2.2200000000000024
```

Outlier detection from Interquartile range (IQR) in original data

[18]:
```
L_outliers=Q1-1.5*(Q3-Q1)
U_outliers=Q3+1.5*(Q3-Q1)
print('Lower outliers in Temp1:', L_outliers)
print('Upper outliers in Temp1:', U_outliers)
```

```
Lower outliers in temp1: 21.889999999999993
Upper outliers in temp1: 30.770000000000003
```

[19]:
```
print('Number of outliers in temp1 upper:', data[data['Temp1']>30.
 ↪74]['Temp1'].count())
print('Number of outliers in temp1 lower:', data[data['Temp1']<21.
 ↪94]['Temp1'].count())
print('% of Outlier in temp1 upper:',round(data[data['Temp1']>30.
 ↪74]['Temp1'].count()*100/len(data)), '%')
print('% of Outlier in temp1 lower:',round(data[data['Temp1']<21.
 ↪94]['Temp1'].count()*100/len(data)), '%')
```

```
Number of outliers in Temp1 upper : 0
Number of outliers in Temp1 lower : 0
% of Outlier in Temp1 upper: 0.0 %
% of Outlier in Temp1 lower: 0.0 %
```

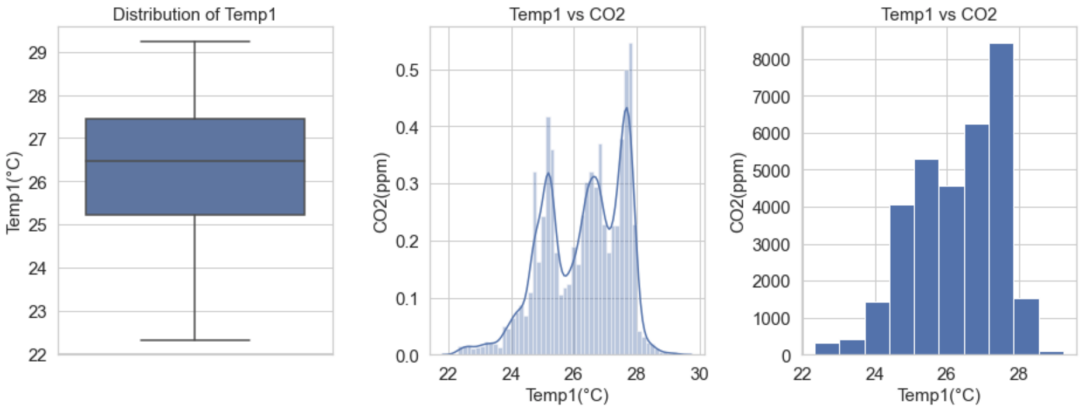


Figure 18: Distribution of Temp1 vs CO2

TEMP2 Range of values observed

[21]:
```
print('Range of values:', data['Temp2'].max()-data['Temp2'].min())
```

```
Range of values:  6.600000000000001
```

Central values4.5.2

[22]:
```
print('Minimum Temp2:', data['Temp2'].min())
print('Maximum Temp2:',data['Temp2'].max())
print('Mean value:', data['Temp2'].mean())
print('Median value:',data['Temp2'].median())
print('Standard deviation:', data['Temp2'].std())
print('Null values:',data['Temp2'].isnull().any())
```

```
Minimum Temp2: 20.9
Maximum Temp2: 27.5
Mean value: 25.002008019645025
Median value: 25.2
Standard deviation: 1.2026502994126629
Null values: True
```

Quartiles

[23]:
```
Q1=data['Temp2'].quantile(q=0.25)
Q3=data['Temp2'].quantile(q=0.75)
print('1st Quartile (Q1) is:', Q1)
print('3st Quartile (Q3) is:', Q3)
print('Interquartile range (IQR) is', Q3-Q1)
```

```
1st Quartile (Q1) is: 24.1
3st Quartile (Q3) is: 26.1
Interquartile range (IQR) is 2.0
```

Outlier detection from Interquartile range (IQR) in original data

[24]:
```
L_outliers=Q1-1.5*(Q3-Q1)
U_outliers=Q3+1.5*(Q3-Q1)
print('Lower outliers in Temp2:', L_outliers)
print('Upper outliers in Temp2:', U_outliers)
```

```
Lower outliers in Temp2: 21.1
Upper outliers in Temp2: 29.1
```

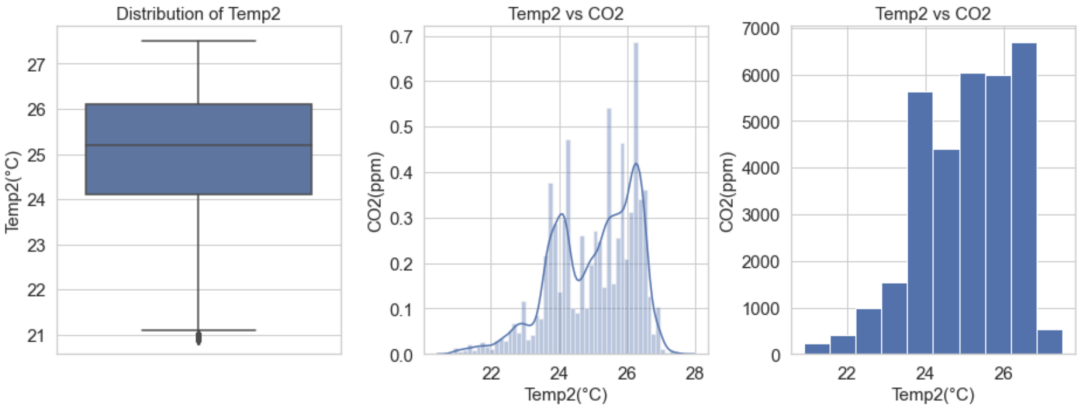


Figure 19: Temp2 vs CO2

[25]:
```
print('Number of outliers in temp2 upper:', data[data['Temp2']>29.
 ↪1]['Temp2'].count())
print('Number of outliers in temp2 lower:', data[data['Temp2']<21.
 ↪1]['Temp2'].count())
print('% of Outlier in temp2 upper:',round(data[data['Temp2']>29.1]['Temp2'].
 ↪count()*100/len(data)), '%')
print('% of Outlier in temp2 lower:',round(data[data['Temp2']<21.1]['Temp2'].
 ↪count()*100/len(data)), '%')
```

```
Number of outliers in temp2 upper : 0
Number of outliers in temp2 lower : 61
% of Outlier in temp2 upper: 0.0 %
% of Outlier in temp2 lower: 0.0 %
```

Humidity Range of values observed

[27]:
```
print('Range of values:', data['Humi'].max()-data['Humi'].min())
```

```
Range of values: 42.35
```

Central values

[28]:
```
print('Minimum Humidity:', data['Humi'].min())
print('Maximum Humidity:',data['Humi'].max())
print('Mean value:', data['Humi'].mean())
print('Median value:',data['Humi'].median())
print('Standard deviation:', data['Humi'].std())
print('Null values:',data['Humi'].isnull().any())
```

```
Minimum Humidity: 17.65
Maximum Humidity: 60.0
Mean value: 27.184417910447763
Median value: 26.67
Standard deviation: 2.8958427007560132
Null values: False
```

Quartiles

[29]:
```
Q1=data['Humi'].quantile(q=0.25)
Q3=data['Humi'].quantile(q=0.75)
print('1st Quartile (Q1) is:', Q1)
print('3st Quartile (Q3) is:', Q3)
print('Interquartile range (IQR) is', Q3-Q1)
```

```
1st Quartile (Q1) is: 26.67
3st Quartile (Q3) is: 27.84
Interquartile range (IQR) is 1.1699999999999982
```

Outlier detection from Interquartile range (IQR) in original data

[30]:
```
L_outliers=Q1-1.5*(Q3-Q1)
U_outliers=Q3+1.5*(Q3-Q1)
print('Lower outliers in Humidity:', L_outliers)
print('Upper outliers in Humidity:', U_outliers)
```

```
Lower outliers in Humidity: 24.915000000000006
Upper outliers in Humidity: 29.595
```

[31]:
```
print('Number of outliers in Humidity upper:', data[data['Humi']>29.
 ↪595]['Humi'].count())
print('Number of outliers in Humidity lower:', data[data['Humi']<24.
 ↪91]['Humi'].count())
print('% of Outlier in Humidity upper:',round(data[data['Humi']>29.
 ↪595]['Humi'].count()*100/len(data)), '%')
print('% of Outlier in Humidity lower:',round(data[data['Humi']<24.
 ↪91]['Humi'].count()*100/len(data)), '%')
```

```
Number of outliers in Humidity upper: 6700
Number of outliers in Humidity lower: 7627
% of Outlier in Humidity upper: 21.0 %
% of Outlier in Humidity lower: 23.0 %
```

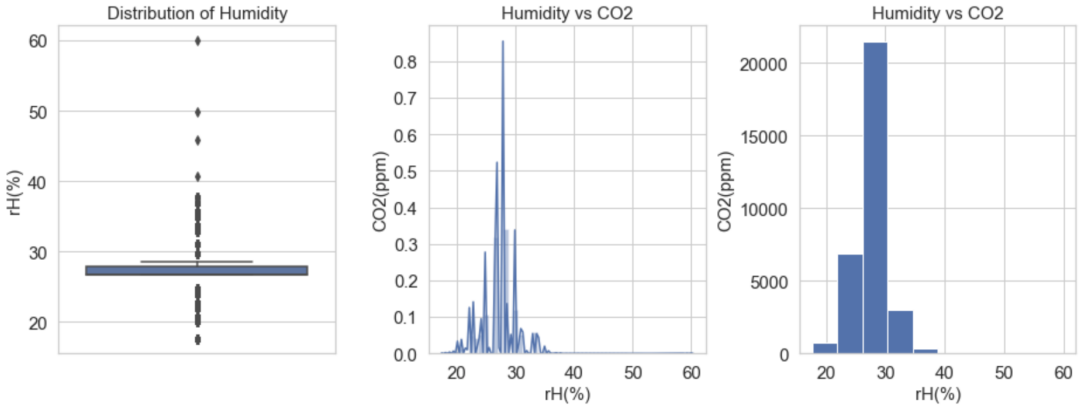


Figure 20: Humidity vs CO2

4.6 Description of dependent attribute

CO2

To carry out the study of whether or not a person is in the area to be investigated, the CO2 level in said space will be studied.

It is possible to determine the arrival or departure time of a person from the studied space.

For this, it is based on the assumption that if the CO2 level decreases, then the person leaves the space, therefore if a person is present in said space, the concentration level of CO2 would increase.

[32]:
```
# Plotting CO2 graphic
fig = px.line(data, x='Date', y='CO2')
fig.show()
```

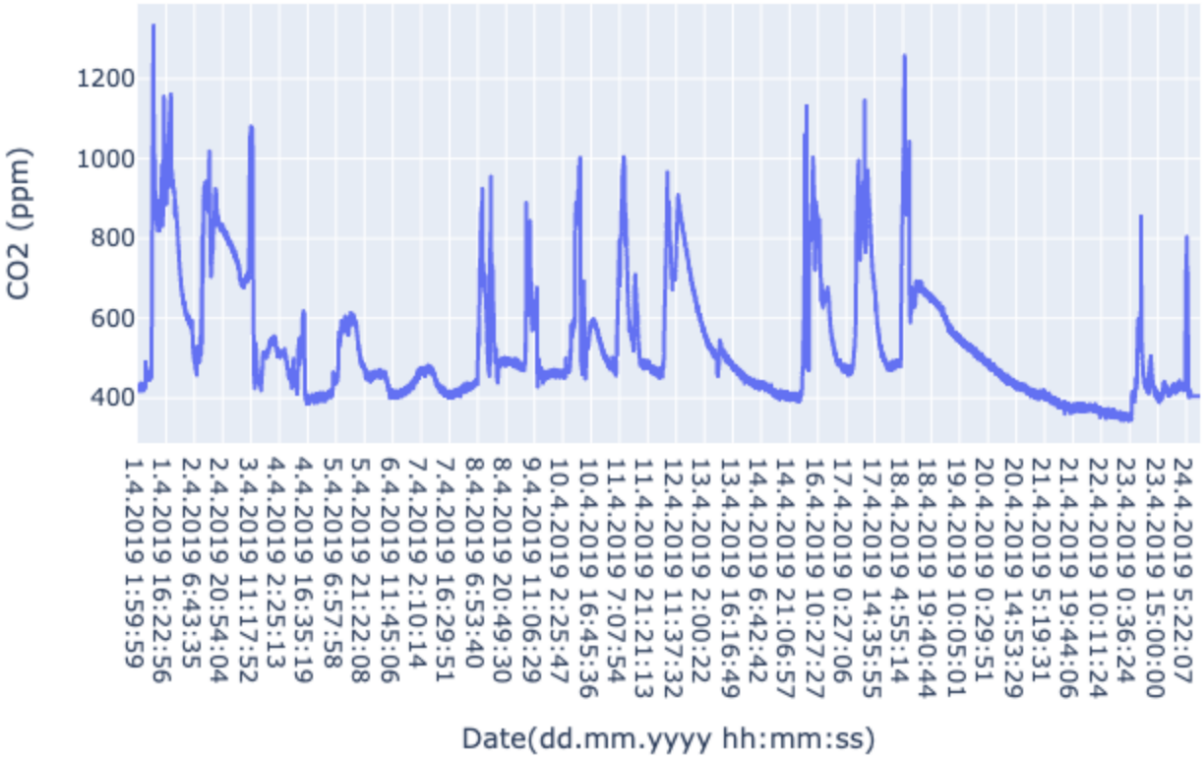


Figure 21: CO2 Distribution in April

Very pronounced peaks are seen on the day 1-04-2019 and the day 18-04-2019, so the following graph will proceed to study one of those days as is shown in figure 21

It can be seen how in the space of time between 00:00 and 03:50, the CO2 concentration is significantly deficient as is shown in figure 22; this maybe since no one is found in the building, in this case, we refer to Ostrava's faculty of electrical engineering and computer science.

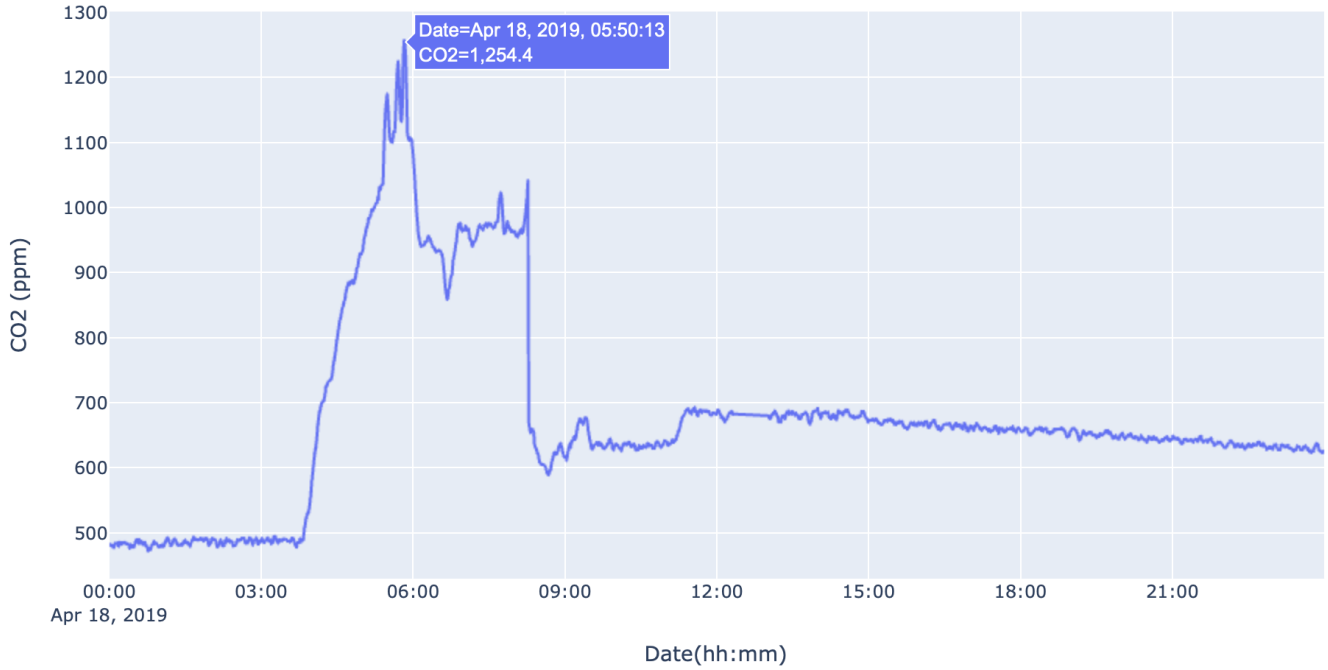


Figure 22: 18 April Co2 Distribution

However, from 04:00 hours until 05:55, the CO2 level increases until reaching the maximum peak of 1255 ppm, which indicates the presence of people in the building. So we can deduce that the rapid increases are due to the presence of people, finding the windows and doors closed and without forced ventilation.

The points where there is a rapid decrease implies that the ventilation systems have been activated, or windows and doors have been opened.

Finally, it is also possible to determine the level of CO2 dispersion, since its progression is slower as it decreases over time, as would be seen in the graph after 3:00 p.m., the ppm levels decrease.

4.7 Multivariate analysis

In the figure shown below, we can determine the following information 23:

- Temp1 has three Gaussians.
- Temp2 has almost two Gaussians.
- CO2 has almost two Gaussians and rightly skewed.
- Humidity has multiple Gaussians.

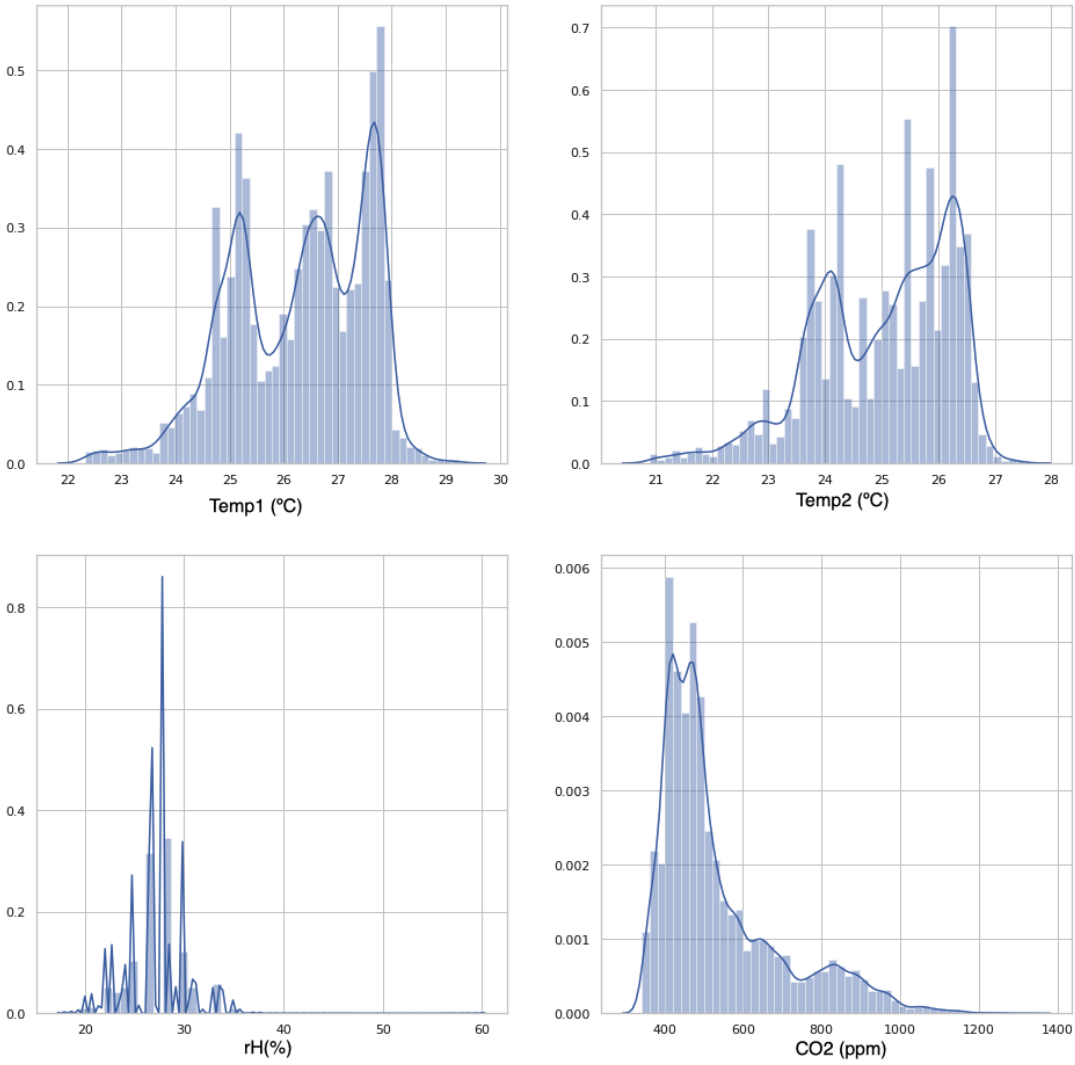


Figure 23: Multivariate attribute analysis

The following figure 24 represents a histogram for each attribute of our dataset so that it allows us to graphically represent each variable in the form of bars, where the surface of each bar is proportional to the frequency of the represented values.

It provides general information on the distribution of the data set concerning each attribute, offering an overview of the preference or trend of the data for a region or a particular value within the spectrum of possible values that the attribute can take.

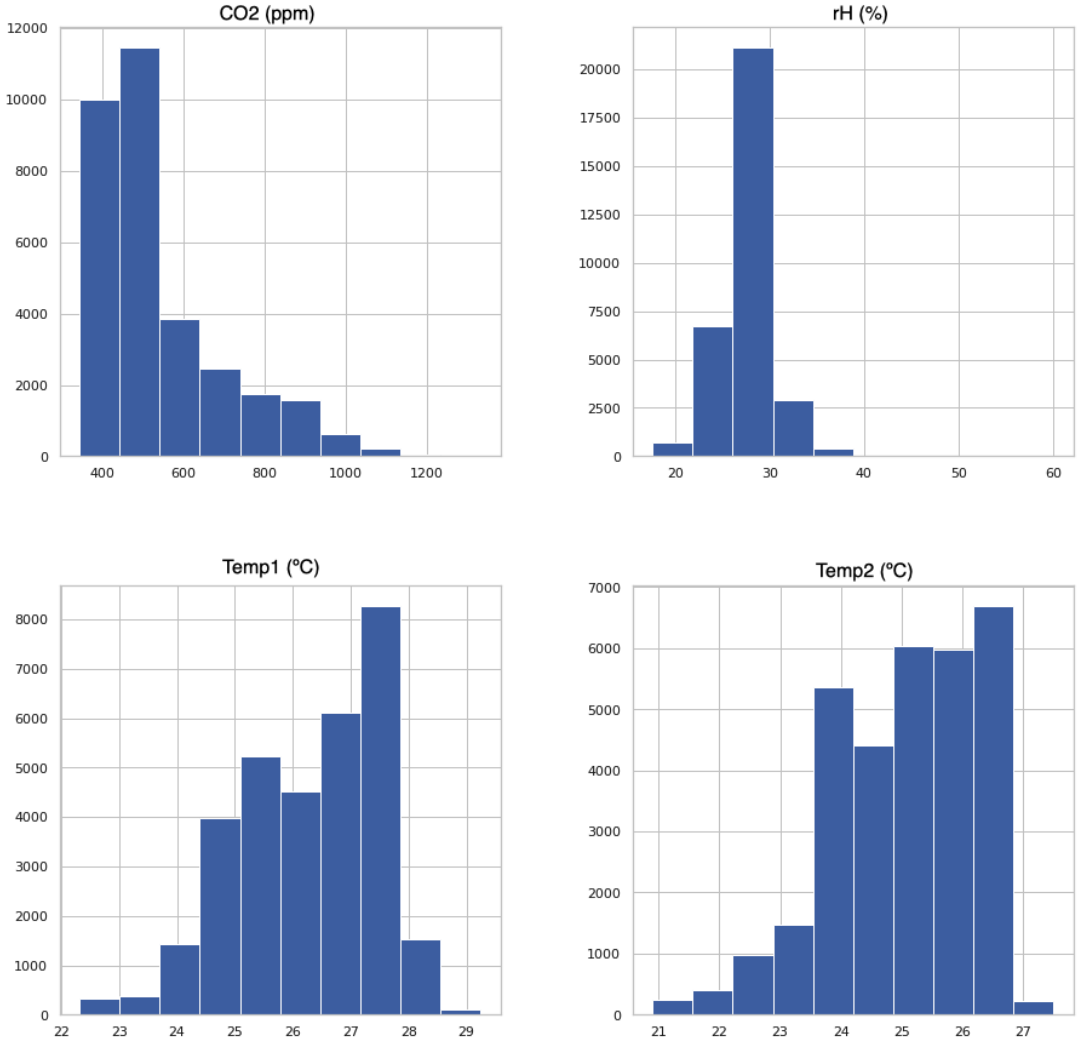


Figure 24: Attribute Histogram

The figure 25 shows the distribution of the data by comparing each attribute concerning the other attributes using the entire data set; the diagonal gives the same information:

Scatter plots

- The Temp1 attribute, It does not show the presence of outliers.
- The Temp2 attribute shows the presence of outliers.
- The Humidity attribute It shows multiple presences of outliers.
- The CO2 attribute is close to a normal curve.

We have not only missing values problem but also outliers the problem in the dataset.

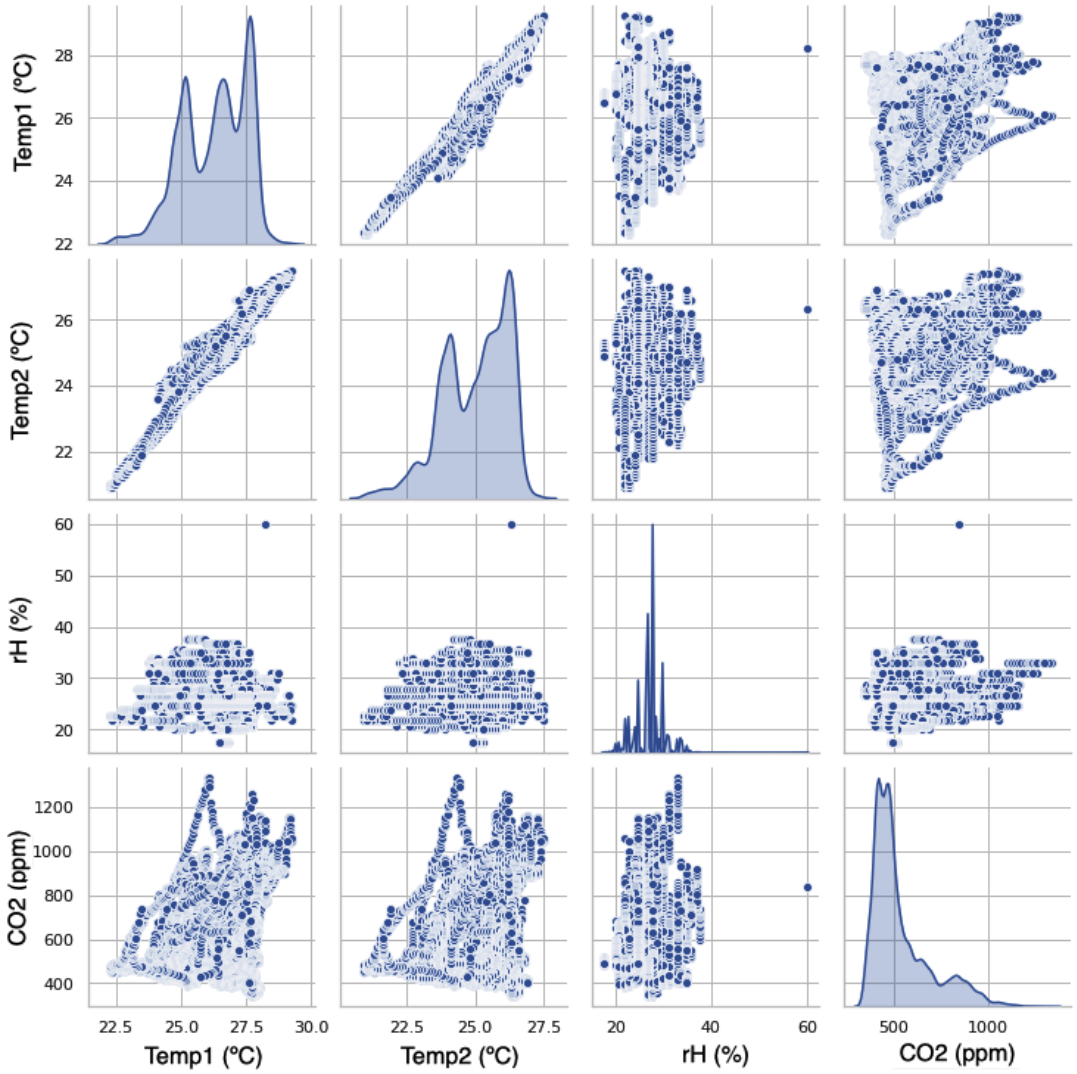


Figure 25: Value distribution per attribute

Off Diagonal Analysis: Relationship between independent attributes

- Temp1 vs. the other attributes: it only shows that can be a high level of correlation with the attribute Temp2, due to both curves are similar.
- Temp2 vs. other independents attribute: This attribute also does not have any significant relation with any other attributes. It almost spread like a cloud
- Humidity vs. other independents attribute: This attribute also does not have any significant relation with any other attributes. It almost spread like a cloud.

The reason why we are doing all this analysis is if we find any kind of dimensions which are very strongly correlated (r-value close to 1 or -1) such dimensions are giving the same information to the algorithms, it is a redundant dimension

[36]:
```
# corrlation matrix
cor=data.corr()
cor
```

[36]:

	Temp1	Temp2	Humi	CO2
Temp1	1.000000	0.849866	-0.057402	-0.001456
Temp2	0.849866	1.000000	-0.005553	0.128610
Humi	-0.057402	-0.005553	1.000000	-0.114620
CO2	-0.001456	0.128610	-0.114620	1.000000

The figure 26 is also giving the same information we observed in pair plot analysis.

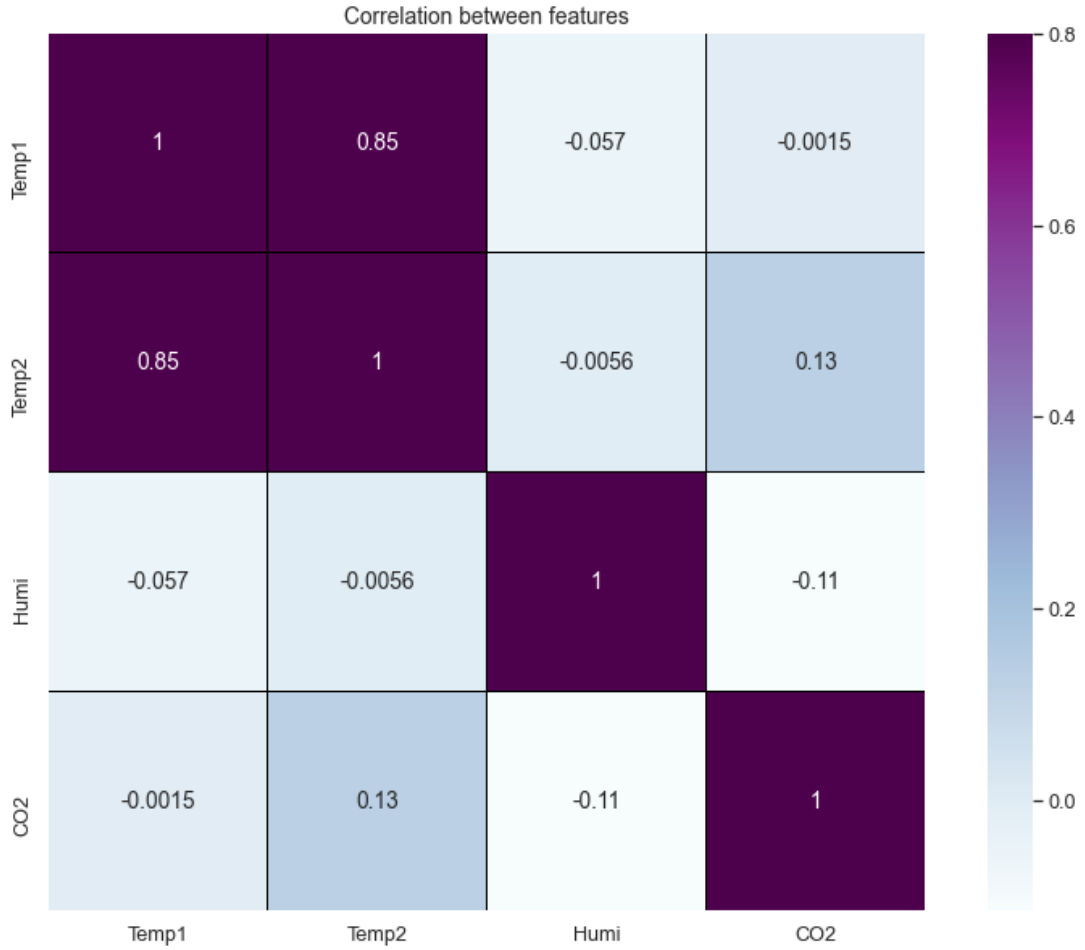


Figure 26: Correlation Matrix

We can see how the correlation between the attributes Temp1 and Temp2 is very high, so it does not provide information, is an attribute that can be eliminated, however, a threshold of 0.95 or higher has been established to eliminate correlated attributes.

4.8 Strategies to handle different data challenges

4.8.0.1 Checking for Missing Values

[39]:
```
#reading the CSV file into pandas dataframe
data = pd.read_csv("file_output/april_output.csv", na_values =␣
 ↪missing_values, sep=",", date_parser=lambda x: datetime.strptime(x, '%d.
 ↪%m.%Y %H:%M:%S'))
#Checking for missing values
del data['Date']
data.isnull().sum()
```

[38]:
```
Temp1    3567
Temp2    3625
Humi     3535
CO2      3433
dtype: int64
```

We can see that there are missing values.

4.8.0.2 Checking for outliers values

In figure 27, we see the box, whose upper edge is Q3, and the lower edge is Q1. Between averages are 50% of occurrences. The height of the box is the interquartile range. Furthermore, the thick line is the median. Above and below are two limits, which are the thresholds for outliers.

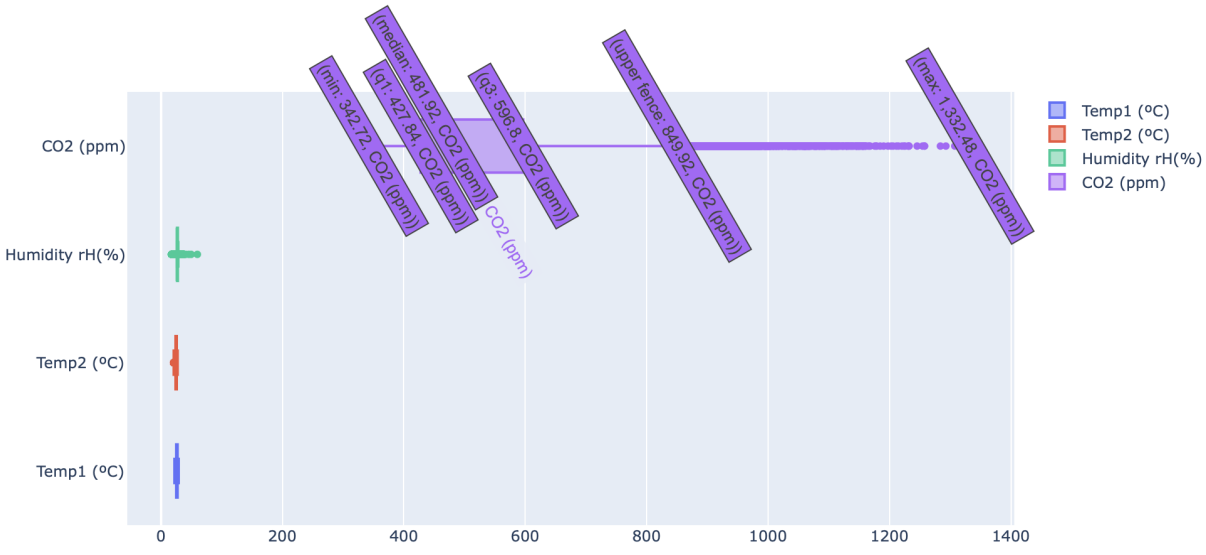


Figure 27: BoxPlot Diagram checking for outliers

- The upper limit is Q3 + 1.5 * IQR, being anything above an outlier.

- The lower limit should be Q1 - 1.5 * IQR, with any pattern below a lower outlier.

It also shows that Humidity, Temp1, Temp2, and CO2 contains outliers.

[40]:
```
#Number of outliers present in the dataset
print('Number of outliers in Temp1: ',df1[((df1.Temp1 - df1.Temp1.mean()) /␣
 ↪df1.Temp1.std()).abs() >3]['Temp1'].count())
print('Number of outliers in Temp2: ',df1[((df1.Temp2 - df1.Temp2.mean()) /␣
 ↪df1.Temp2.std()).abs() >3]['Temp2'].count())
print('Number of outliers in Humidity: ',df1[((df1.Humi - df1.Humi.mean()) /␣
 ↪df1.Humi.std()).abs() >3]['Humi'].count())
print('Number of outliers in CO2: ',df1[((df1.CO2 - df1.CO2.mean()) / df1.
 ↪CO2.std()).abs() >3]['CO2'].count())
```

```
Number of outliers in Temp1: 96
Number of outliers in Temp2: 153
Number of outliers in Humidity: 175
Number of outliers in CO2: 333
```

Here, we have used the Standard deviation method to detect the outliers.If we have any data point that is more than three times the standard deviation; then, those points are very likely to be outliers. We can see that Temp1, Temp2, Humidity, and CO2 contain outliers.

Handling the outliers

The alternatives proposed to deal with outliers will be substituting the outlier for the median, as is shown in figure 28 or eliminating the pattern that has outliers.

In carrying out this research, the elimination of those patterns with outliers have been chosen, since replacing the median does not make sense, since false information is being generated, that has no a priori benefit.

Below an example about how to replace by median works:

It can be seen in the box diagram, as any value that is higher and lower than the quartiles (represented by the intersection of the lines) are outliers.

4.8.0.3 Preprocessing datasets

Preprocessed output files features:

- All attributes are in the same scale between 0 and 1 by normalization.
- Outliers have been deleted.
- Missing Values have been replaced by linear interpolation, because the substitution of missing data using the mean may not be correct, as it is a time series, so a linear interpolation better fits the data model.
- Correlated attributes have been deleted because the correlation indicates the similarity between attributes, so having two very similar attributes is not recommended.

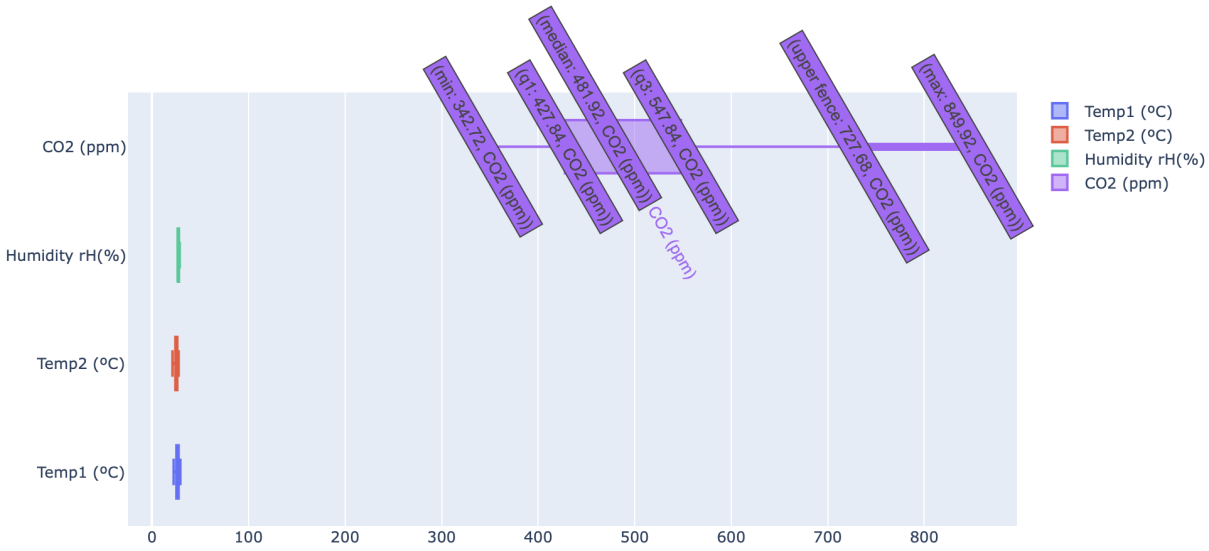


Figure 28: BoxPlot Diagram replacing by median value

- Duplicate values have been removed, due to it could incur the algorithm to show more preference for such patterns.
- Elimination of the 'Date' attribute, since despite being a time series problem, a larger dataset is not available, having only two contiguous months, so it is assumed that there will be no significant changes in the data.

4.8.0.4 Output Dataset Visualization

[45]:
```
# Use april preprocesed file to display info
data = pd.read_csv("./dataset/final/Preprocesed_april_output.csv", sep=",")
# Display the new dataset info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21501 entries, 0 to 21500
Data columns (total 4 columns):
Temp1    21501 non-null float64
Temp2    21501 non-null float64
Humi     21501 non-null float64
CO2      21501 non-null float64
dtypes: float64(4)
memory usage: 672.0 KB
```

[47]:
```
#Analyze the distribution of the dataset
data.describe().T
```

[48]:

	count	mean	std	min	25%	50%	75%	max
Temp1	21501.0	0.552830	0.182828	0.0	0.39701	0.58209	0.71343	1.0
Temp2	21501.0	0.606182	0.202655	0.0	0.45763	0.64407	0.77966	1.0
Humi	21501.0	0.456490	0.188927	0.0	0.30019	0.42511	0.55003	1.0
CO2	21501.0	0.317646	0.235386	0.0	0.14101	0.22494	0.44221	1.0

[48]:
```
# corrlation matrix
cor=data.corr()
cor
```

It is appreciated that the highest correlation of attributes occurs between Temp1 and Temp2, however, being a correlation less than 0.95, it has been decided not to eliminate the attribute, because we will be losing information.

[48]:

	Temp1	Temp2	Humi	CO2
Temp1	1.000000	0.814804	-0.078072	0.027226
Temp2	0.814804	1.000000	-0.028192	0.174716
Humi	-0.078072	-0.028192	1.000000	-0.141705
CO2	0.027226	0.174716	-0.141705	1.000000

The alternatives proposed to deal with outliers will be substituting the outlier for the median or eliminating the pattern that has outliers.

In carrying out this research, the elimination of those patterns with outliers has been chosen 29, since replacing the median does not make sense, since false information is being generated that has no a priori benefit.

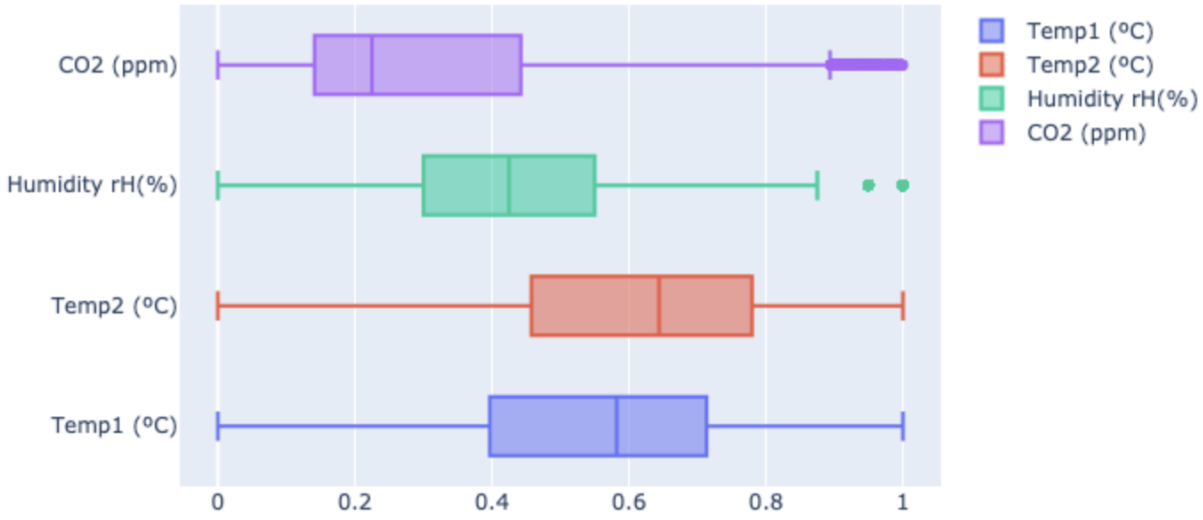
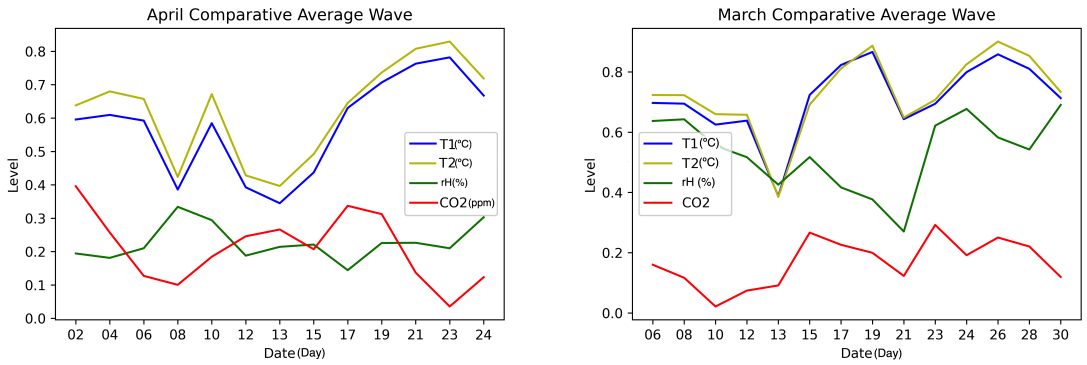


Figure 29: BoxPlot without outliers

Visualization of both data sets has been carried out over time, to see their waveform concerning CO2, obtaining the results shown in the following figure 30. This figure has been established on the X-axis, a period of two days for the Y-axis, this being the mean of the attribute on those days.



(a) Average April Comparative Waveform

(b) Average March Comparative Waveform

Figure 30: Average Comparative Waveforms

Knowing that these measurements are not accurate because the average of between two days is being used, the person being able to spend a long time in one day and very little time in another period has been carried out to estimate the middle levels.

For the data treatment, a missing values recovery process has been carried out, deleting those attributes with more than 80% of missing values and recovering missing values using linear interpolation. [56].

5 Model Design

5.1 SVR Parameters Optimization

To make a fair comparison and show the benefits of adjusting the parameters of our program, we must use an objective function. Otherwise, the test set error could be overly optimistic since the model has been adjusted and tested on the same set [27]. Usually need to adjust or tuning three parameters (C, epsilon, and gamma, if using an RBF kernel); the total number of candidate models in the Grid-Search can be quite large [30].

For example, considering ten candidate values for each parameter, it should test 10 * 10 * 10 candidate SVR models, plus cross-validation on top of that, and the number of models to build and evaluate becomes quite large.

5.1.1 Evaluation of Models

The coefficient determined the quality of the model to replicate the results and the proportion of variation of the results that can be explained by the model [60].

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{n=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{n=1}^{n}(y_i - \bar{y}_i)^2} \tag{12}$$

The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

On the other hand, use has been made of two metrics, such as the Mean Squared Error (MSE), which is non-negative values, the best possible value is 0.0, which indicates the error made between the actual value and the value predicted by the model [60].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \check{Y})^2 \tag{13}$$

The last evaluation metric will be the RMSE, which measures the amount of error between two data sets. That is, it compares a predicted value and an observed or known value, defined by the equation [60]:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\check{y}_i - y_i)^2}{n}} \tag{14}$$

5.2 Optimizing Parameters

5.2.1 Parameters to optimice

- **C**: It is the penalty parameter and regularisation. The higher the penalty value, the model makes fewer mistakes, but it is observed that after a specific penalty value, the model is overtraining.
- **Gamma**: It defines how far influences the calculation of a plausible line of separation. The higher the gamma value, the only near points, are considered. A lower gamma value, far away points are also considered.
- **Epsilon**: It specifies the epsilon-tube within which no penalty is associated with the training loss function with points predicted within a distance epsilon from the actual value.

5.3 Regularization Methods

This approach involves fitting a model involving all p predictors. However, the estimated coefficients are shrunken towards zero relatives to the least-squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance. [33]

Using regularization minimizes the complexity of the model while minimizing the cost function. This results in simpler models that tend to generalize better. Those overly complex models tend to overfit. That is, to find a solution that works very well for training data but very bad for new data.

It is the objective to obtain models that, in addition to learning well, also have an excellent performance with new data.

5.3.1 Ridge Regression

The main problem to be solved in the Regression Ridge application is the estimation of the most suitable value of k. The choice of this parameter involves a balance between the components of bias and variance of the mean squared error when estimating β.

So assuming a linear model, the higher the k, the greater the bias, but the less the variance of the estimator, and the final determination implies a compromise between both terms [43].

The complexity C is measured as the mean square of the model coefficients. As in Lasso, Ridge regularization can be applied to various machine learning techniques.

Ridge will work when the expert thinks that various of the input attributes or features are correlated with each other, making the coefficients end up being smaller.

This decrease in the coefficients minimizes the effect of the correlation between the input attributes and makes the model generalize better, working better when most of the attributes are relevant.

5.3.2 Lasso Regression

Lasso was designed to improve the prediction accuracy and interpretability of statistical regression models by altering the model building process by selecting only a subset of the variables provided for use in the final model [71].

In Lasso regularization, also called L1, complexity C is measured as the mean of the absolute value of the model coefficients. For the case of the root mean square error, this is the complete development for Lasso (L1):

$$J = \frac{1}{M}\sum_{i=1}^{M}(yi - yi)^2 + \alpha\frac{1}{N}\sum_{j=1}^{N}|\omega_j|$$

When using Lasso regularization, it is favored that some of the coefficients end up being 0, which can be useful to discover which of the input attributes are relevant and, in general, to obtain a model that generalizes better.

Lasso can help us to make the selection of input attributes, working better when the attributes are not uncommonly correlated between them.

5.3.3 ElasticNet

Elastic Net arose as a result of the Lasso regression, whose variable selection may be data-dependent and, therefore, unstable. The solution proposed is combining the Ridge and Lasso regression penalties to get the best of both.

ElasticNet will be used when the dataset has a large number of attributes, some of which will be irrelevant, and others will be correlated with each other.

5.3.4 Linear Regression

The objective of a regression model is to try to explain the relationship between a dependent variable (response variable) and a set of independent variables (explanatory variables) $X_1, \ldots X_n$. In a simple linear regression model, an attempt is made to explain the relationship between the response variable Y and a single explanatory variable X.

5.3.5 Shrinkage Methods Comparison

The superiority of one method over the other depends on the scenario in which they are applied, being sometimes very similar. Generally, when only a small number of predictors out of all included have substantial standardized coefficients, and the rest have very small or zero values, SVR generates better models, as is shown in figure 31.

Name	MSE	STD
Linear Regression	0.091777	0.006949
Ridge	0.091772	0.006905
Lasso	- 0.000362	0.000335
ElasticNet	- 0.000362	0.000335
Support Vector Regression	0.417924	0.010753

Figure 31: Shrinkage Methods

If, on the contrary, all the included predictors have non-zero coefficients (they all contribute to the model) and of approximately the same magnitude, Ridge regression (RID) tends to work better.

The precision of the model obtained by Lasso (LSO) or Ridge regression [35] depends on the choice of the λ value used since it determines the degree of penalty.

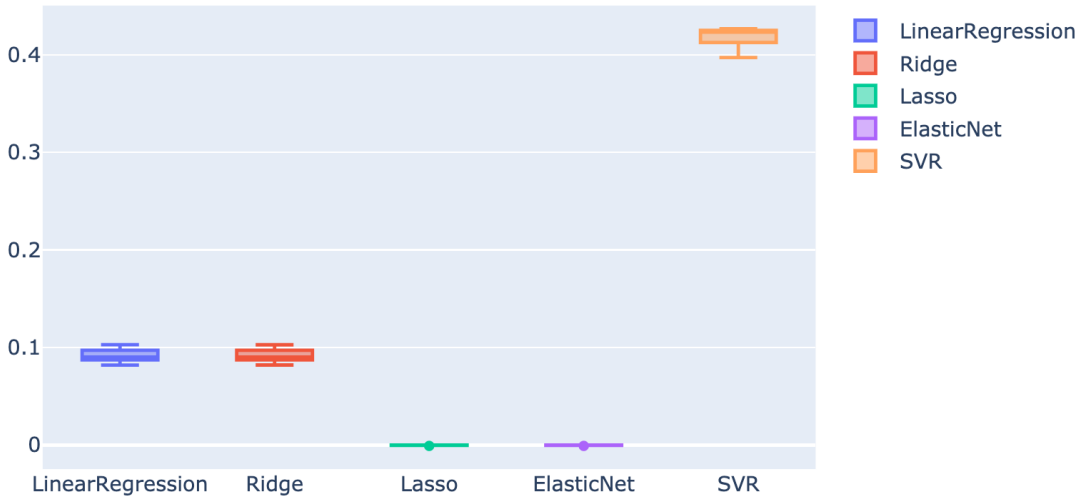


Figure 32: BoxPlot Algorithm Comparison

The way to identify the best method for a given study is to divide the available observations into two groups (training and test).

Following the steps of each of the methods, the model is adjusted using only the training set, and the MSE is calculated using the test set, being the method with the lowest MSE is the one that achieves the best precision.

	Model	Score	Model	Score	Model	Score	Model	Score	Model	Score
0	LinearRegression (LN)	0.0918	standardLN	0.0918	minmaxLN	0.0918	normalizerLN	0.1013	RobustLN	0.0918
1	Ridge (RID)	0.0918	standardRID	0.0918	minmaxRID	0.0918	normalizerRID	0.1013	RobustRID	0.0918
2	Lasso (LSO)	-0.0004	standardLSO	-0.0004	minmaxLSO	-0.0004	normalizerLSO	-0.0004	RobustLSO	-0.0004
3	ElasticNet (EN)	-0.0004	standardEN	-0.0004	minmaxEN	-0.0004	normalizerEN	-0.0004	RobustEN	-0.0004
4	SVR	0.4179	standardSVR	0.4312	minmaxSVR	0.4179	normalizerSVR	0.2036	RobustSVR	0.4225

Figure 33: Standardization score models five-fold

A study of the different types of algorithms has been carried out by varying the number of folds, as is shown in figure 33. Where we can see that there is no significant difference in the number of folds concerning the result obtained, so it has been decided that due to the considerable computation time that ten folds represent, the study will be carried out between two-fold and five-fold 34. K-fold will divide the dataset into a prespecified number of folds, and every sample must be in one and only one fold, being a fold is a subset of your dataset.

	Num Folds			
Algorithm	2	5	10	15
LN	0,3726	0,3725	0,3726	0,3725
RID	0,3726	0,3725	0,3726	0,3725
LSO	-0,0001	-0,0005	-0,0006	-0,0008
EN	-0,0001	-0,0005	-0,0006	-0,0008
SVR	0,6759	0,6829	0,6797	0,6798

Figure 34: Comparision of K-fold

5.3.6 How to optimize the parameters

Nested cross-validation is used to estimate the generalization performance of a full learning pipeline, which includes optimizing hyperparameters. We will use five folds in the outer loop.

Score: Return the coefficient of determination R^2 of the prediction.

The coefficient R^2 is defined in the equation 12, where "u" is the residual sum of squares.

The best possible score is 1.0, and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0 and a value of MSE close to 0.0 as is defined in equation 13.

5.3.7 Defining Score and search methods

The code shown below shows us the metrics used for the evaluation of our model, in which two metrics have been used, the correlation coefficient and the mean square error:

[ ]:
```
def MSE(y_true,y_pred):
    mse = mean_squared_error(y_true, y_pred)
    print ('MSE: %2.3f' % mse)
    return mse
def R2(y_true,y_pred):
     r2 = r2_score(y_true, y_pred)
     print ('R2: %2.3f' % r2)
     return r2
def two_score(y_true,y_pred):
    MSE(y_true,y_pred) #set score here and not below if using MSE in GridCV
    score = R2(y_true,y_pred)
    return score
def two_scorer():
  return make_scorer(two_score, greater_is_better=True) # False = MSE
```

The next code fragment, the following, has defined the Random-Search and Grid-Search class. These classes will be in charge of giving a model, searching for its hyperparameters, either randomly or optimally, for the model.

[ ]:
```
class Random-Search(object):
    def __init__(self,X_train,y_train,model,hyperparameters):
        self.X_train = X_train
        self.y_train = y_train
        self.model = model
        self.hyperparameters = hyperparameters
        self.best_score = []
        self.best_params = []
    def Random-Search(self):
        # Create randomized search 3-fold cross validation and 20 iterations
        cv = 3
        clf = RandomizedSearchCV(self.model,
                                 self.hyperparameters,
                                 random_state=1,
                                 n_iter=20,
                                 cv=cv,
                                 verbose=0,
                                 n_jobs=-1,
                                 scoring=two_scorer(),
                                 )
        # Fit randomized search
        best_model = clf.fit(self.X_train, self.y_train)
        self.best_score.append(best_model.best_score_)
        self.best_params.append(best_model.best_params_)
        return best_model,best_model.best_params_

    def BestModelPridict(self,X_test):
        best_model,_ = self.Random-Search()
        pred = best_model.predict(X_test)
        return pred
```

In turn, the parameters of these methods have been established in order to achieve the best hyperparameters that fit our model and data, such as the number of iterations, the number of cross-validation folds, and the evaluation method.

[ ]:
```
class Grid-Search(object):
    def __init__(self,X_train,y_train,model,hyperparameters):
        self.X_train = X_train
        self.y_train = y_train
        self.model = model
        self.hyperparameters = hyperparameters
        self.best_score = []
        self.best_params = []

    def Grid-Search(self):
        # Create randomized search 3-fold cross validation
        cv = 3
        clf = Grid-SearchCV(self.model,
                                self.hyperparameters,
                                cv=cv,
                                verbose=0,
                                n_jobs=-1,
                                refit=True,
                                error_score=0,
                                scoring=two_scorer(),
                                )
        # Fit randomized search
        best_model = clf.fit(self.X_train, self.y_train)
        self.best_score.append(best_model.best_score_)
        self.best_params.append(best_model.best_params_)
        return best_model,best_model.best_params_

    def BestModelPridict(self,X_test):
        best_model,_ = self.Grid-Search()
        pred = best_model.predict(X_test)
        return pred
```

5.3.8 Grid-Search vs Random Search

The most common method of selecting algorithm parameters is Grid-Search. The Grid-Search is performed by selecting a list of values for each parameter and testing all possible combinations of these values, being an exhaustive search method. However, research has been conducted that claims that a random search in the parameter space can be more effective than a Grid-Search.

In the article developed by Bergstra [13], he states that a random search of the parameter space is more effective than the Grid-Search, being, in turn, competitive with other parameter search methods, such as using population-based algorithms.

The fundamental idea on which the research is based is that in the vast majority of cases, the surface of the objective function is not irregular in all dimensions, some parameters have less effect on the objective function than others, so that knowing this, the search for specific parameters can be given more weight in the Grid-Search.

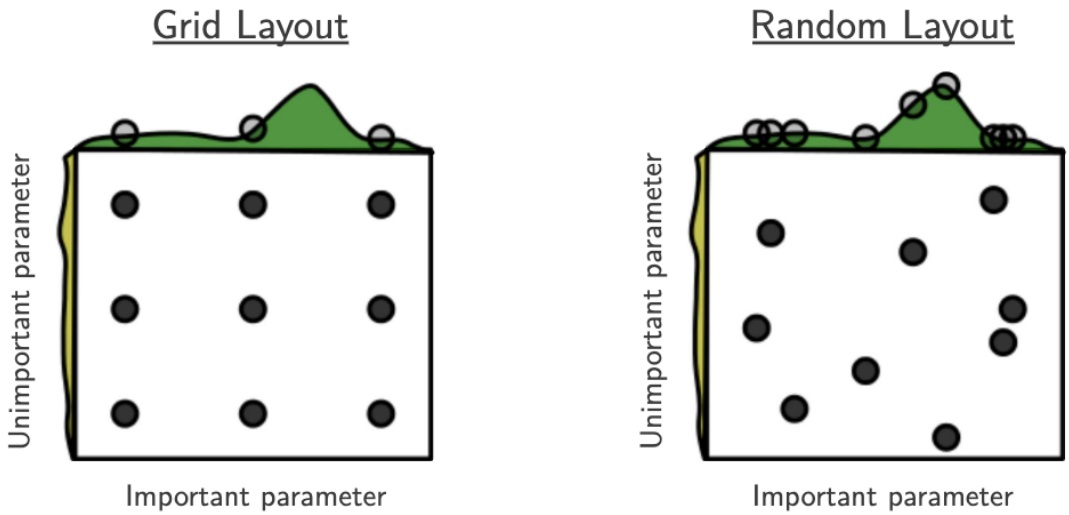


Figure 35: Grid Layout vs Random Layout, source Bergstra Paper

However, this case does not usually occur, so employing the random search, and it allows exploring a higher number of values for each parameter in the same number of tests, as is shown in figure 35. Thus, in this investigation, the search was carried out with both types of searches, and then a comparison was made.

5.3.9 Range of parameters to optimize

A search range of the optimal hyperparameters for SVR has been established, as shown in the following code snippet.

[ ]:
```
# Range of parameters to optimize
kernel = ['rbf']
gamma = np.logspace(-15, 3, num=9, base=2)
tol = [0.0001, 0.001, 0.1, 1]
C = np.logspace(-5, 15, num=11, base=2)
epsilon = np.logspace(-5, 15, num=11, base=2)
hyperparameters= dict(
  kernel = kernel,
  gamma = gamma,
  tol = tol,
  C = C,
  epsilon = epsilon
)
```

5.3.10 Testing with the dataset preprocesed

After executing the Grid-Search method, the following result has been obtained, in which the elapsed time can be seen, being 15 hours, as well as the optimal hyperparameters.

```
(14471, 3) (3618, 2) (14471,) (3618,)
Elapsed Time: 4h 27m 37s R2 Score: [0.9073075779178978]
[{'C': 32.0, 'epsilon': 0.03125, 'gamma': 8.0, 'kernel': 'rbf', 'tol': 0.
↪0001}]
```

On the other hand, after executing the Random-Search method, the following result has been obtained, which is very similar to that previously obtained and in a much shorter time.

```
(14471, 3) (3618, 2) (14471,) (3618,)
Elapsed time: 0h 5m 45s R2 Score: [0.9161303110926914]
[{'C': 86.80322265625, 'epsilon': 0.03125, 'gamma': 46.0693359375, 'kernel':␣
↪'rbf', 'tol': 0.0001}]
```

As previously discussed in the section 5.3.5, obtaining these hyperparameters has been carried out using a kfold and using cross-validation

5.3.11 Parameters Results

In the table shown below, It is possible to see the different results obtained, as well as the values of the SVR parameters obtained.

Kernel	C	Gamma	Epsilon	Tol	Search Method
RBF	32	8	0,03125	0,0001	GS
RBF	86,8032227	46,0693359	0,03125	0,0001	RS

Figure 36: Parameters obtained to test

These values in figure 36 are the ones that will be tested in the testing section, where we will finally determine the precision of our algorithm by predicting the CO2 value and thus determining the presence or not of people in the room.

6 Model Building

6.1 Building the Model

The code shown below shows us the configuration of our SVR model. Where we will divide the dataset into 80% (14471 samples) for training and 20% (3618 samples) for testing which have no effect on training and so provide an independent measure of network performance during and after training, later we will scale the characteristics, through a standard transformation.

Succeeding fit the SVR model according to the given training data and finally perform regression on samples in X using the predict method.

Finally, the score function returns the coefficient of determination R^2 of the prediction.

[69]:
```
data  = pd.read_csv('dataset/final/Preprocesed_march_output.csv', sep=",")
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=44)
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
model = SVR( tol=0.0001, kernel='rbf', gamma=8.0, epsilon=0.01, C=512)
model.fit(X_train, y_train)
# estimate accuracy on validation dataset
X_test = scaler.transform(X_test)
y_pred = model.predict(X_test)
print("Performance on training data:",model.score(X_train, y_train))
print("Performance on test data:",model.score(X_test, y_test))
print('Accuracy:',r2_score(y_test, y_pred))
print('RMSE:',np.sqrt(mean_absolute_error(y_test, y_pred)))
```

```
Performance on training data: 0.9013106611567959
Performance on test data: 0.873823283994796
Accuracy: 0.8738232839947961
MSE: 0.005906655159650698
RMSE: 0.1668545335547436
```

The previously shown code would only be used to make the prediction using the previously processed dataset. However, tests have also been carried out using a division of the original dataset into train and test, these being processed individually and in the same way, as previously mentioned.

The result shown has been one of the best tests obtained, reaching an accuracy of 88% in the test set, with an error of 0.005, so it can be seen that the model has improved considerably concerning the original score.

7 Model Evaluation

7.1 Test Developed

A set of tests have been carried out to obtain the generalization values of our model. For this, a division of the data set has been made into 80% of data for the train and 20% for tests.

The division of the data has been divided in two ways, using a separation using split arrays or matrices into a random train and test subsets. Also, on the other hand, through ShuffleSplit, which will randomly sample the entire data set, in this way, we will be able to obtain the performance of our model by using the validation set.

In turn, the generalization test will be carried out at intervals of days, see one day, three days, one week, two weeks, and a full month.

To be fair tests, the first day and time of each dataset have been selected as the starting day, and tests may also be carried out on random days; however, the tests would not be fair for each model.

The result obtained will be a table for each parameter configuration and method used in the evaluation of the model. So, in said table, it will be possible to visualize both the precision of the model, as well as the values of MSE, RMSE defined by equations 13 and 14, as well as the hyperparameters used and the range of days used for the tests.

7.2 ShuffleSplit Configuration

The ShuffleSplit method, contrary to other cross-validation strategies, random divisions does not guarantee that all folds will be different, although this is still very likely for large data sets [60].

The parameters used for the use of ShuffleSplit have been the following:

- n_split: Number of re-shuffling and splitting iterations, five splits have been used.
- test_size: Represent the proportion of the dataset to include in the test split; in that case, 20% (0.2) is for testing.
- random_state: It is the seed used by the random number generator.

7.3 train_test_split Configuration

This function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model [60].

The parameters used for the use of train_test_split have been the following:

- Arrays : It is a sequence of indexables with the same length or shape. Allowed inputs are lists, NumPy arrays, scipy-sparse matrices, or pandas dataframes.
- test_size: Represent the proportion of the dataset to include in the test split; in that case, 20% (0.2) is for testing.
- random_state: It is the seed used by the random number generator.
- shuffle: Whether or not to shuffle the data before splitting.

7.4 Model performance

Once the tests have been carried out, the results will be shown both in a table and graphical form; in this way, it will be possible to visualize the performance of the model by comparing the predicted results against the real values.

The columns shown in the results tables represents:

- Range of days: Range of days taken from the dataset.
- Accuracy: Model accuracy with the test set.
- Accuracy CV: Model accuracy with cross-validation.
- MSE: MSE value of the model with the test set.
- MSE CV: MSE value of the model with cross-validation.
- RMSE: RMSE value of the model with the test set.
- RMSE CV: RMSE value of the model with cross-validation.
- Kernel, C, Gamma, Epsilon: Hyperparameters of the model.
- Month: The month of the test

Since the values obtained in MSE and RMSE through cross-validation are negative, it must be clarified that the value is negative, due to the function used, a combined score is applied, always maximizing the score, so the tests that must be minimized as in this case, they refuse for the scoring function to work correctly.

Therefore, the score that is returned is negated when it is a result that must be minimized and left positive if it is maximized.

First, the results obtained with the first set of hyperparameters, shown below in the following image 37, both performed using the Grid-Search method.

Range of days	Accuracy	Accuracy CV	MSE	MSE CV	RMSE	RMSE CV	Kernel	C	Gamma	Epsilon	Month
1	0.993153	0.982503	0.000674	−0.001443	0.025954	−0.036684	RBF	32	8	0.03125	April
3	0.962208	0.964692	0.002570	−0.002454	0.050695	−0.048974	RBF	32	8	0.03125	April
7	0.903550	0.892852	0.005867	−0.006658	0.076599	−0.081270	RBF	32	8	0.03125	April
14	0.882652	0.883401	0.006091	−0.006495	0.078046	−0.080502	RBF	32	8	0.03125	April
month	0.826258	0.802874	0.009626	−0.010521	0.098114	−0.102507	RBF	32	8	0.03125	April
1	0.986260	0.986316	0.000532	−0.000571	0.023072	−0.023893	RBF	32	8	0.03125	March
3	0.952545	0.942448	0.001856	−0.002195	0.043079	−0.046678	RBF	32	8	0.03125	March
7	0.903582	0.904436	0.003988	−0.003881	0.063154	−0.062171	RBF	32	8	0.03125	March
14	0.898954	0.931249	0.004064	−0.002838	0.063747	−0.052730	RBF	32	8	0.03125	March
month	0.870837	0.882323	0.006046	−0.005310	0.077759	−0.072678	RBF	32	8	0.03125	March

Figure 37: Grid-Search Model Evaluation

Proceeding with the tests, it has been performed with the following set of parameters obtained 38, this time with the Random-Search method, showing below the results.

Range of days	Accuracy	Accuracy CV	MSE	MSE CV	RMSE	RMSE CV	Kernel	C	Gamma	Epsilon	Month
1	0.993900	0.982445	0.000600	−0.001462	0.024497	−0.036572	RBF	86.803223	46.069336	0.03125	April
3	0.964649	0.969001	0.002404	−0.002157	0.049030	−0.046096	RBF	86.803223	46.069336	0.03125	April
7	0.933308	0.904490	0.004057	−0.005946	0.063696	−0.076769	RBF	86.803223	46.069336	0.03125	April
14	0.906869	0.898765	0.004834	−0.005651	0.069528	−0.074969	RBF	86.803223	46.069336	0.03125	April
month	0.861060	0.828477	0.007698	−0.009156	0.087738	−0.095506	RBF	86.803223	46.069336	0.03125	April
1	0.960439	0.947958	0.001533	−0.002180	0.039149	−0.046613	RBF	86.803223	46.069336	0.03125	March
3	0.905474	0.923635	0.003697	−0.002897	0.060799	−0.053464	RBF	86.803223	46.069336	0.03125	March
7	0.906043	0.880932	0.003887	−0.004863	0.062342	−0.069238	RBF	86.803223	46.069336	0.03125	March
14	0.911971	0.936859	0.003540	−0.002605	0.059500	−0.050730	RBF	86.803223	46.069336	0.03125	March
month	0.912686	0.909603	0.004087	−0.004081	0.063933	−0.063593	RBF	86.803223	46.069336	0.03125	March

Figure 38: Random-Search Model Evaluation

In the previous images, it can be seen how the precision of the model varies considerably using both methods. Thus, it is much more accurate when predicting intervals between one day and three days. However, as the number of days increases, the model begins to decrease the precision.

The best values obtained have been for the second set of hyperparameters, in which we can observe the best values obtained for the set of days.

- Predicting one day: In the prediction for a single day, both models work well, since both have an accuracy of 0.99 using a test set and 0.98 using cross-validation, as well as the RMSE values of both models are similarly obtaining 0.02 with cross-validation in April. However, for March, the values are lower, achieving a better Grid-Search result with a 0.98 compared to the value of the Random-Search 0.94, as well as the respective values of RMSE 0.02 with Grid-Search and 0.04 for Random-Search.

- Predicting three days: In the prediction for three days, it occurs in a similar way to the previous one, since both have an accuracy of 0.96 using test set and cross-validation, as well as the RMSE values of both models are similarly obtaining 0.04 in cross-validation, which are values indicating that our model fits the data well. It can be seen how the prediction for March using Random-Search, has achieved a better score with validation than with test.

- Predicting seven days: In the seven-day prediction, we found a difference between the different methods, since the parameters obtained through Random-Search, have obtained a higher precision than that of Grid-Search, being this of 0.904, as well as a better RMSE than that of Grid-Search with a value of 0.07.

- Predicting fourteen days: In the prediction obtained in the fourteen-day range, it can be seen how Random-Search has obtained similar results through cross-validation, reaching 0.936 and an RMSE of 0.05 for March, compared to the values obtained with Grid-Search also for March. However, for April, the results are down to 0.89 with an RMSE of 0.07.

- Predicting a month: In the prediction of a full month, the results obtained have been the worst, since the best result has been with Random-Search, reaching a validation precision of 0.9; however, the RMSE committed is very low, being 0.06.

7.4.1 Implementation of the practical part experiments

The practical implementation and construction process of the SVR model, together with the verification of the results obtained in the test data are carried out in the following steps:

Step 1: Measurement of non-electrical variables Temp1, Temp2, rH, and CO2 in the Ostrava Faculty of Electrotechnics and Computer Science (FEI) from March 4 to March 31, 2019, from April 1 to April 25, 2019, using operating sensors.

Step 2: Data preprocessing by removing correlated elements, outliers, missing values, normalization, and duplicate patterns.

Step 3: Compare the model to be implemented along with other algorithms to compare its performance.

Step 4: Optimization of hyperparameters of the model to be built to obtain higher performance.

Step 5: Training of the SVR model, for the parameters obtained through the search methods explained in previous sections.

Step 6: Evaluation of the model using the described metrics, as well as through the use of cross-validation.

1. SVR (from March 4, to March 5, 2019) with the tested data (from March 4, to March 5, 2019), as is shown in figure 38. The best results were recorder with the predicted course of CO2 for SVR with Grid-Search hyperparameters, which are indicated in the image and using five fold, MSE = 0.023 (ppm) and R^2 = 0.98 using cross-validation. In the figure shown below 39, we can see the predictions made by the model against the real CO2 values.

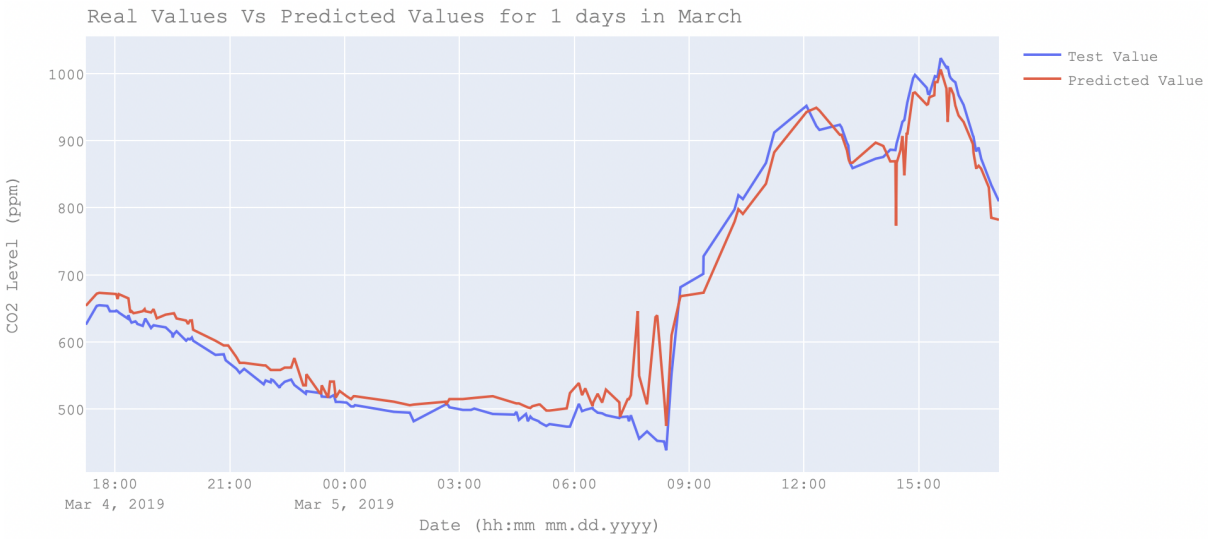


Figure 39: Prediction obtained for a day in March

2. SVR (from April 1, to April 2, 2019) with the tested data (from April 1, to April 2, 2019), as is shown in figures 38 and 37. The best results were recorder with the predicted course of CO2 for SVR with Random-Search hyperparameters which are indicated in the image and using 5 fold, MSE = 0.0365 (ppm) and R^2 = 0.982 using cross-validation. In the figure shown below 40, we can see the predictions made by the model against the real CO2 values.

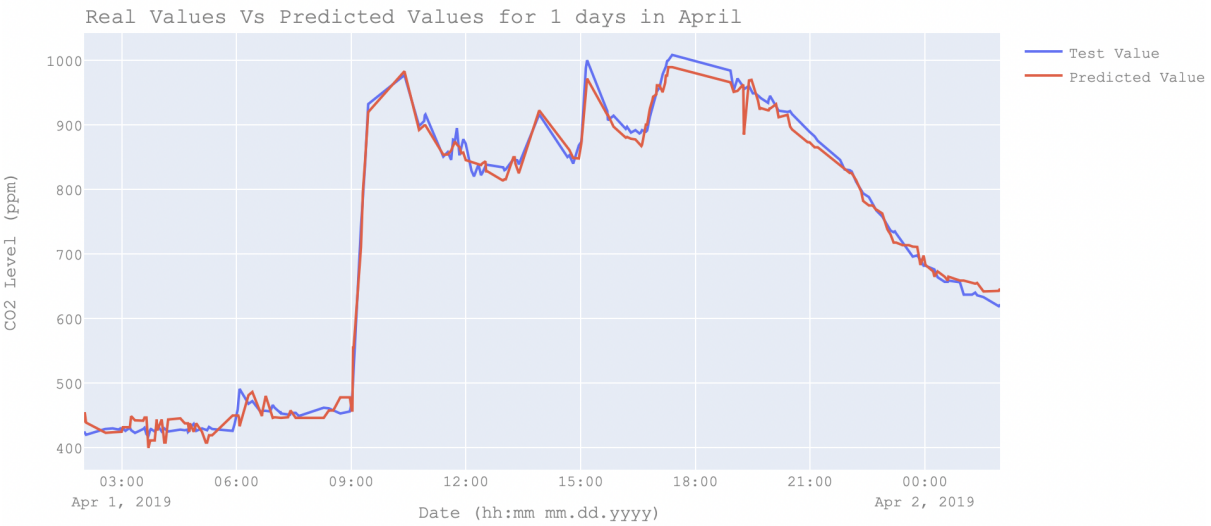


Figure 40: Prediction obtained for a day in April

3. The predictions obtained for a time range of three days have been better for April, regardless of the search method, the best result obtained has been R^2 = 0.96 and an RMSE = 0.04 (ppm) have been obtained, as can be seen in the figures, as is shown in the figure below, 41.

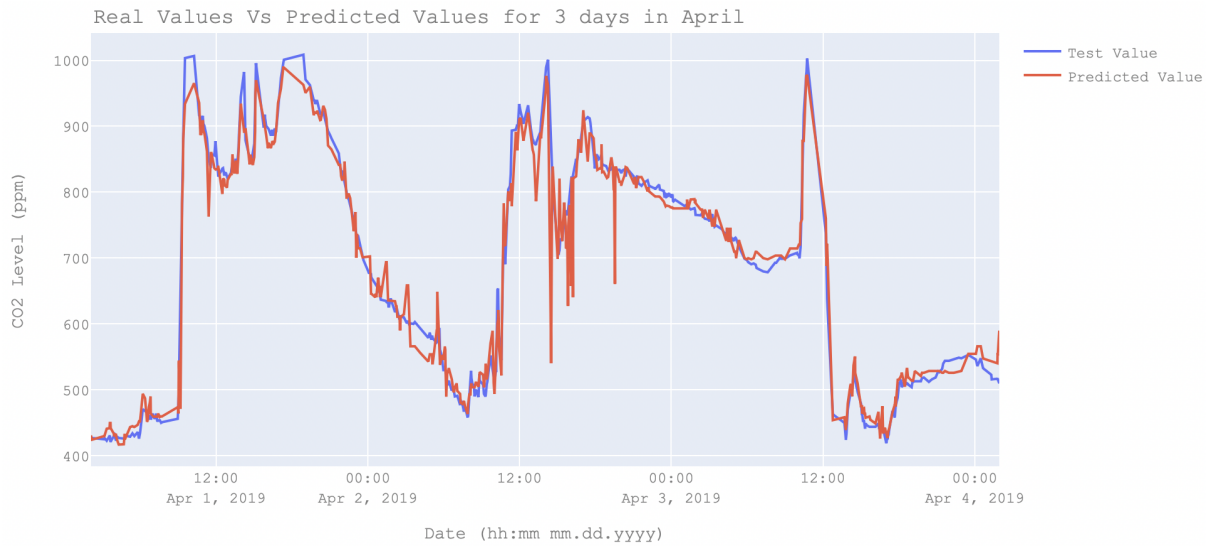


Figure 41: Prediction obtained for three days in April

For March, the best result obtained was R^2 = 0.92 and RMSE = 0.05 (ppm), by using the hyperparameters obtained with Random-Search, as is shown in the next figure 42.

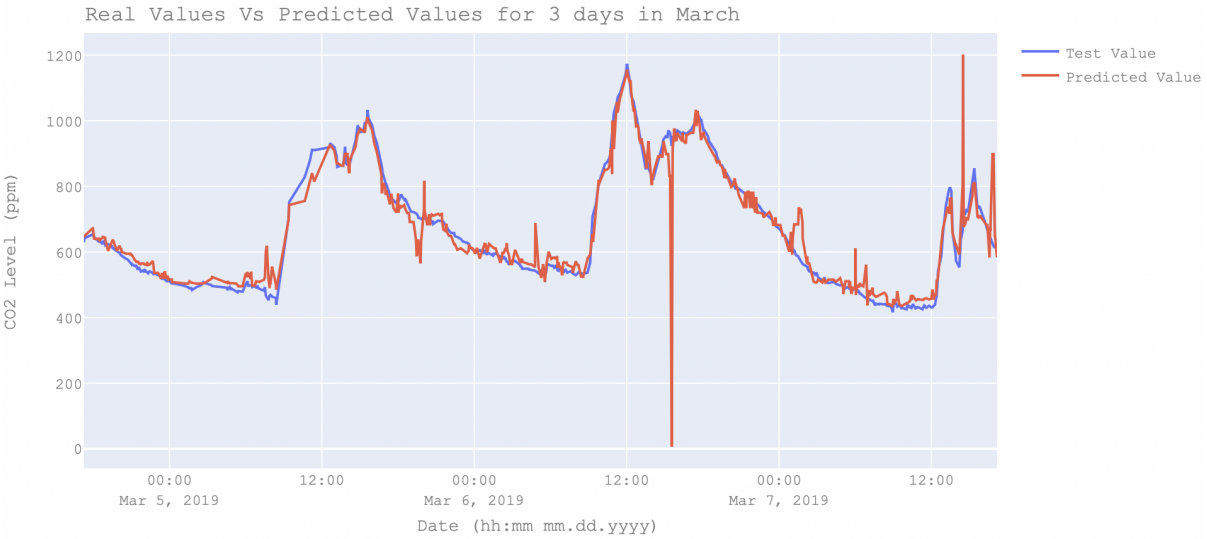


Figure 42: Prediction obtained for three days in March

4. The predictions obtained for a time range of seven days have been better for April, the best result obtained has been R^2 = 0.90 and an RMSE = 0.06 (ppm) have been obtained, as can be seen in the figures using Grid-Search hyperparameters. The figure below, 43 shows the values predicted by the model versus the real values for that range in April.

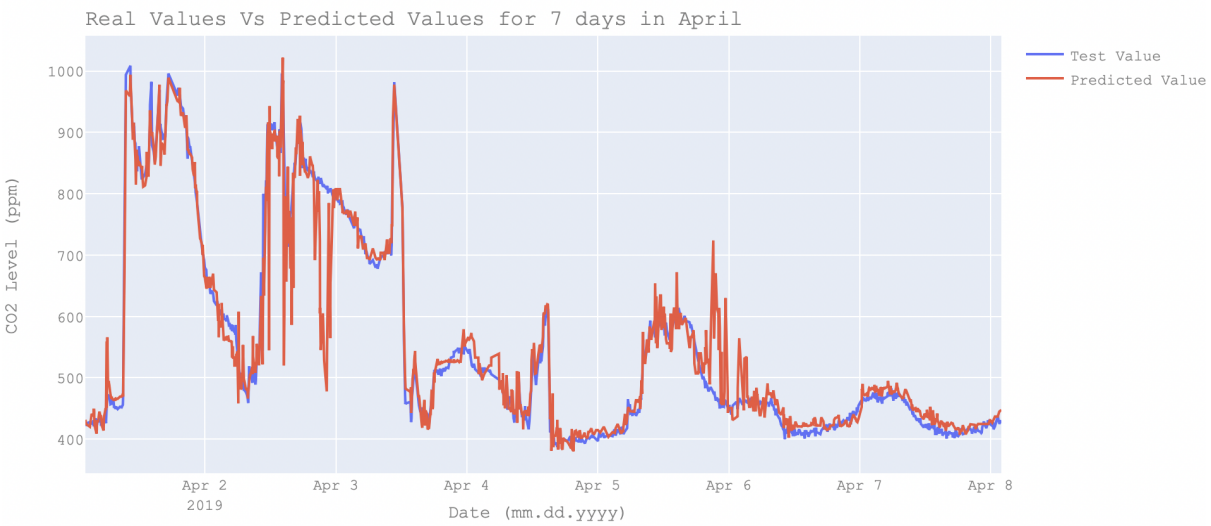


Figure 43: Prediction obtained for seven days in April

For March, the best result obtained was R^2 = 0.90 and RMSE = 0.06 (ppm), by using the hyperparameters obtained with Grid-Search, as is shown in the next figure 44.

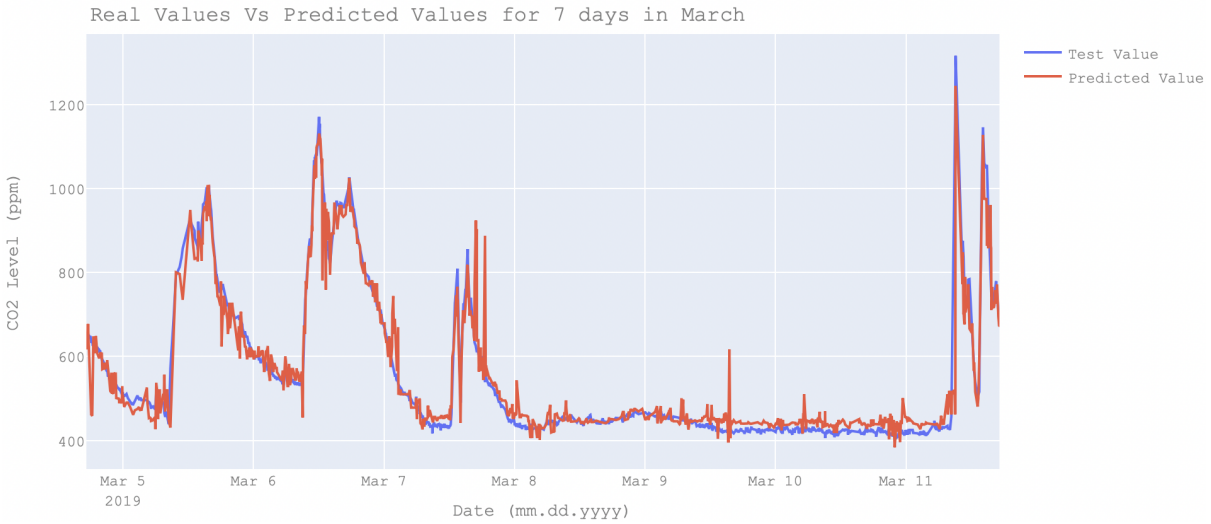


Figure 44: Prediction obtained for seven days in March

5. The predictions obtained for a time range of fourteen days have been better for March, the best result obtained has been $R^2 = 0.93$ and an RMSE = 0.05 (ppm) have been obtained, as can be seen in the figures using Grid-Search hyperparameters. The figure below, 45 shows the values predicted by the model versus the real values for that range in March.

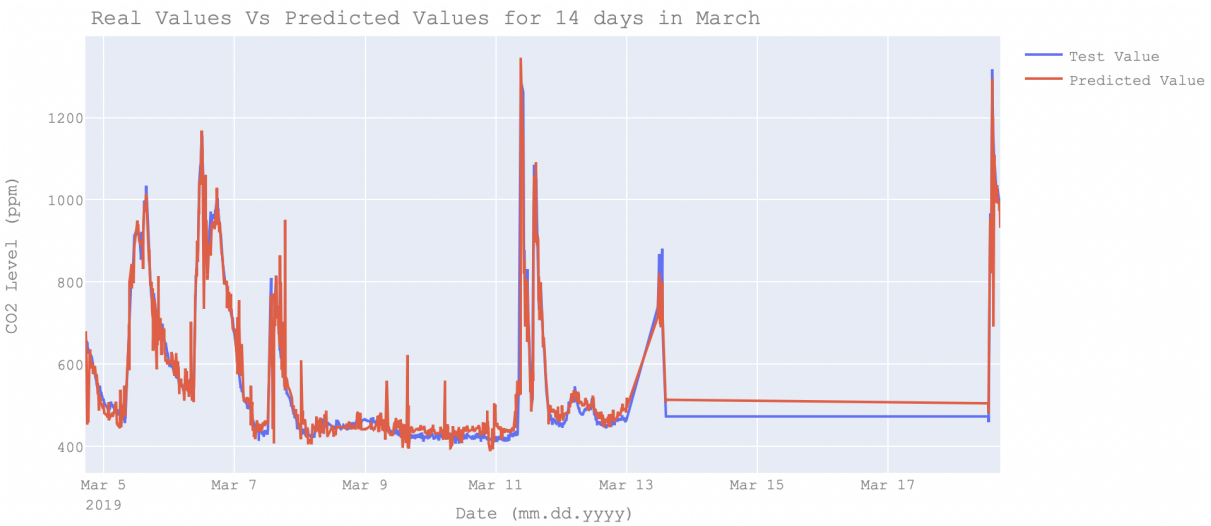


Figure 45: Prediction obtained for fourteen days in march

For April, the best result obtained was $R^2 = 0.88$ and RMSE = 0.08 (ppm), by using the hyperparameters obtained with Grid-Search, as is shown in the next figure 46.

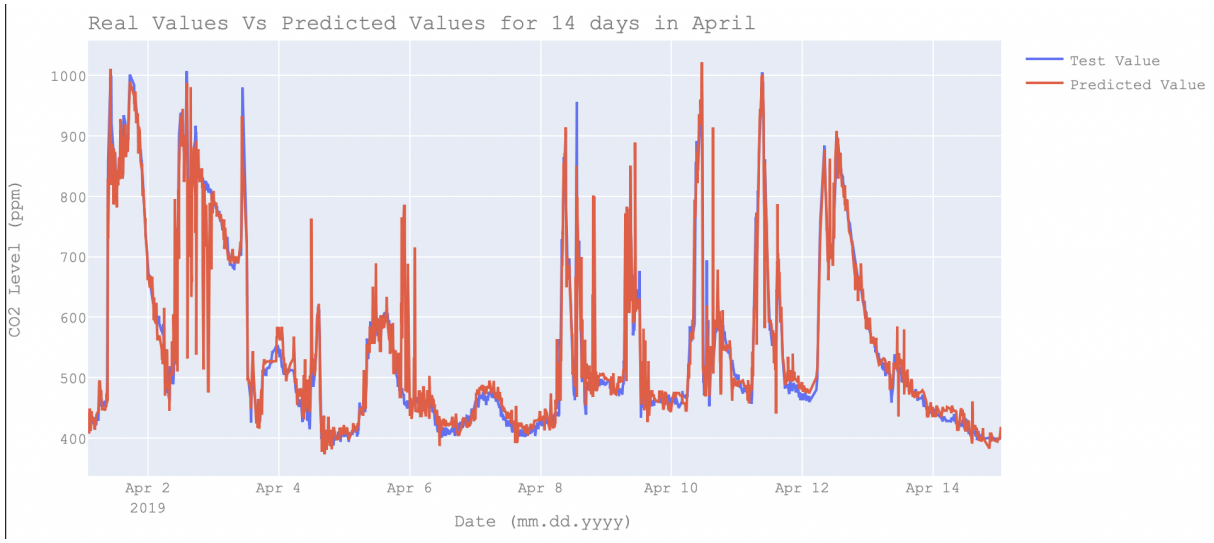


Figure 46: Prediction obtained for fourteen days in April

6. The predictions obtained for a time range of full month have been better for March in figure 47; the best result obtained has been R^2 = 0.90, and an RMSE = 0.06 (ppm) have been obtained, as can be seen in the figures using Random-Search hyperparameters.

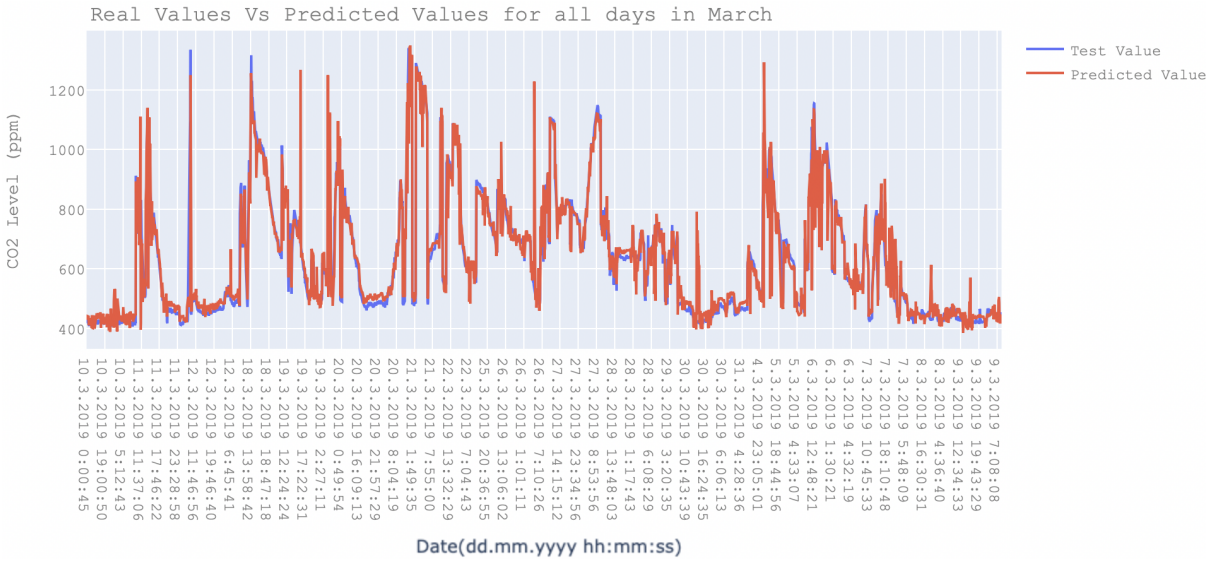


Figure 47: Prediction obtained for March

For April, the best result obtained was R^2 = 0.82 and RMSE = 0.09 (ppm), by using the hyperparameters obtained with Random-Search, as is shown in the next figure 48.

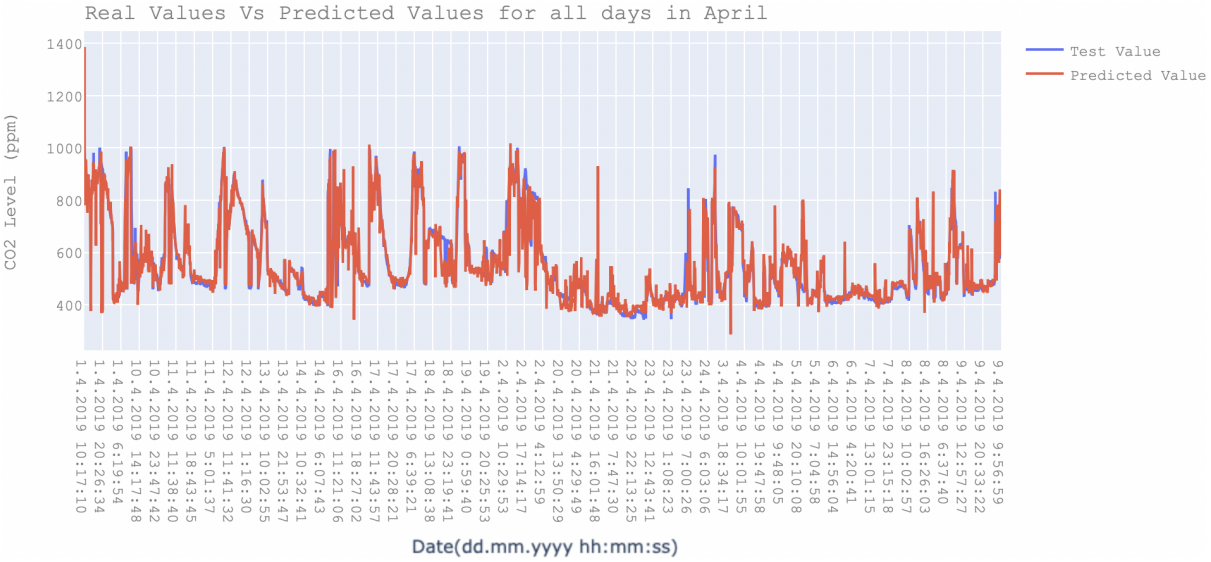


Figure 48: Prediction obtained for April

Step 7: Evaluation of the results achieved.

8 Analysis of Results

The document describes an approach applying support vector machines to monitor the occupation of the designated space, in this case, the Ostrava Faculty of Electrotechnics and Computer Science (FEI).

As shown in figure 34, the tests carried out have been carried out both in April and March, obtaining better results than the first ones.

The first part of the document explains the mathematical procedure on which the operation of SVM is based, as well as the implementation of the KNX system in an IoT environment since the data processed has been obtained through the KNX system and the use of sensors temperature (ºC), relative humidity (% RH) and CO2 sensors (ppm).

As mentioned, the work that most of the time has been dedicated has been to data processing, since finding repeated values, missing values and outliers, as well as attributes at different scales, requires excellent processing for good results.

After said processing, we have tried to optimize the SVR algorithm, by searching for the optimal hyperparameters for our problem, and subsequently training our model using 80% off patterns from the April set, since it has been the one with the best result It has been obtained in the search and subsequent tests.

A standard scaling has been used, even though in the initial tests better results have been obtained with the robust Scaler, the computation time for obtaining the hyperparameters, as well as the execution of the SVR algorithm increases dramatically, which is why which and after seeing the initial results that are seen in the ref, there is no significant difference between the standard and the robust scalers.

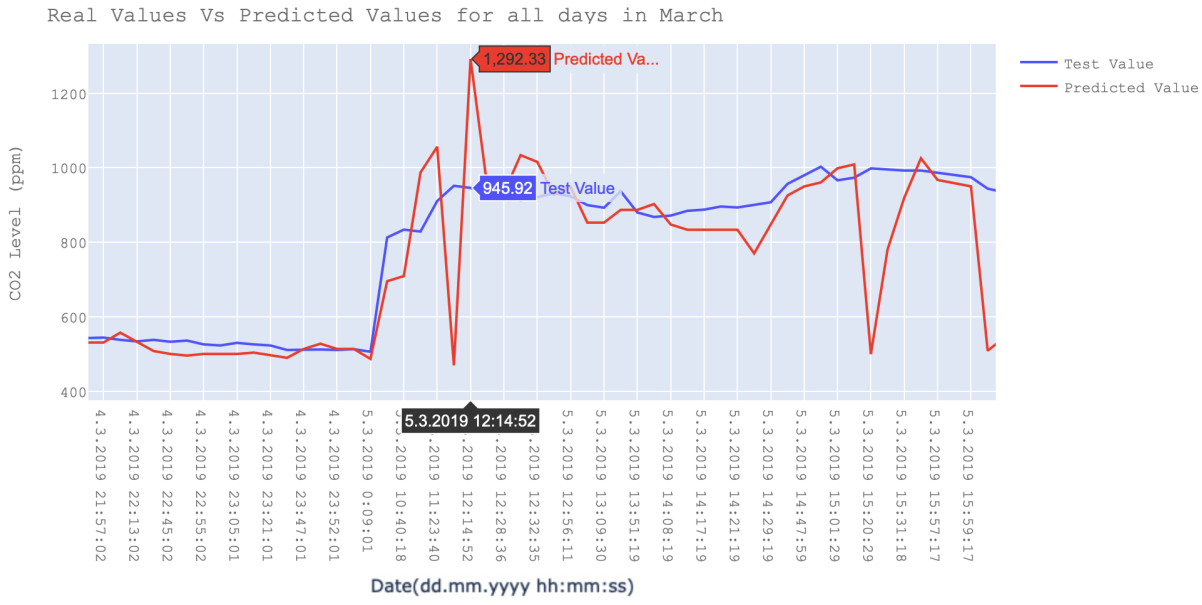


Figure 49: Prediction Peaks

After optimization, it has been shown in figures 38 and 37, how it has been possible to increase the initial result from 0.67 until achieving a precision close and higher than 0.90, together with the Root Mean Squared Error (RMSE) metric also using cross-validation, to check if our model is good using data that you have never seen before.

Lastly, the predicted ppm values have been represented graphically against the real values, obtaining a graph with the denormalized values, to know which real values are those that have been predicted.

It can be seen how at the most extreme peak, as shown in figure 49, it corresponds to March 5 at nine and 12:14, obtaining a high predicted CO2 level with a value of 1292 ppm; however, the real value is 945 ppm. So it may be a time when the windows and doors are closed, with the ventilation system turned off.

It can be seen how, before the abrupt ascent to the maximum peak since 9:00, the CO2 levels are deficient, being around 450 to 500 ppm.

Therefore, this increase can be deduced from the possible presence of people in the room, as previously mentioned, all the closed and off ventilation systems.

Before said peak, it can be seen how the CO2 concentration is medium. The presence of people in the room may remain constant for the next few hours, until, as can be seen in the image, it slowly decreases again, and then rises again.

It is assumed that it is possible that the room would remain empty until 14:00, with ventilation systems, turned off and windows closed.

9 Conclusions

In this chapter, the conclusions of the work carried out are collected, evaluating the results obtained and comparing them with the initial results. Section 9.3 discusses the overall results of the project concerning the knowledge acquired and previous experiences. Section 9.1 analyzes the objectives that have been met and justifies those that have not been possible to meet. Finally, Section 9.2 proposes improvements applicable to the developed system and new lines of research.

9.1 Conclusions regarding the objectives

The article describes the design and verification of the indirect method of predicting the course of CO2 concentration (ppm) using SVR with parameter optimization techniques for monitoring the presence of people in the Ostrava Faculty of Electrotechnics and Computer Science (FEI).

The article further describes the method of Support Vector Regression (SVR) for predicting the course of CO2 from the values measured by the relative temperature and humidity sensors. For estimating CO2 concentration in the air in order to obtain information on the occupancy of individual rooms (arrival time, departure time, number of people).

There are much better methods for predicting human occupation by measuring the level of CO2 (ppm), such as neural networks [67], as well as methods for optimizing model parameters, both regression and classification.

9.2 Future Improvements

As it has already been commented in the development of this research, due to the computation time of the Grid-Search method with a large data set, it is possible that the performance of the optimization of these parameters can be improved by another class of algorithms [64] [52], such as genetic algorithms [83], Particle Swarm Optimization (PSO) [84], possible improvements in Grid-Search, algorithms based on sine-cosine, and multi-objective optimization [59].

In this way, it is possible to achieve optimal hyperparameters in reasonable computation time, thus trying to improve the prediction model to a level of reliability higher than that obtained in the development of this study.

9.3 Personal conclusions

With the realization of this project, I have entered the world of data analysis that each, specifically, with machine learning.

I would like to emphasize that the preparation of this report has been entirely in LATEX. Although I had already used it the previous year in other works, the development of this memory has allowed me to improve the quality of my documentation.

Finally, as machine learning and data analysis was a subject that had not been studied in depth in the subjects of the Degree in Computer Science, I consider that I have learned a lot, from basic notions such as they are the operation of sensors and actuators to methods to process and analyze much more complex data.

Acronyms

BD Big Data. 11

CART Classification and Regression Trees. 12

HVAC Heating, ventilation, air conditioning. 12

IB Intelligent Building. 5, 25

IMS Infrastructure Mediated Sensing. 12

IoT Internet of Things. 11, 12, 26

IP Internet Protocol. 26

KNN K-nearest neighbor algorithm. 12

LDA Linear discriminant analysis. 12

MSE Mean Squared Error. 55, 59, 60

PL PowerLine. 26

PSO Particle Swarm Optimization. 81

R-F Radio Frequency. 26

RF Random Forest. 12

RMSE Root Mean Squared Error. 80

TP Twisted-pair cable. 26

Glossary

Big Data Big Data can be defined as extensive data analysis. An amount of data, so extremely large, that data processing software applications that were traditionally being used are not able to capture, process, and value in a reasonable time.
These data are obtained from several sources, such as:

- Produced by people: emails, social networks, surveys.
- Between machines: Communication between machines such as thermometers, parking meters, vehicle GPS and mobile phones, are communicated through devices to others to which they transmit the data they are collecting.
- Biometrics: fingerprint sensors, retinal scanners, DNA readers, facial recognition, or voice recognition sensors. Its use is widespread in the field of security.

. 11, 83

Intelligent Building Intelligent Buildings are those whose facilities and systems (for air conditioning, lighting, electricity, security, telecommunications, multimedia, computer, and access control) allow integrated and automated management and control, in order to increase the energy efficiency, safety, usability, and accessibility. Wong, Li, and Wang. 5, 83

Internet of Things It is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that have unique identifiers and the ability to transfer data through a network, without requiring human to human or human to computer interactions.. 11, 83

Internet Protocol Digital data communication protocol functionally classified in the network layer according to the international OSI model. Its principal function is the bidirectional use in origin or destination of communication to transmit data through a non-connection oriented protocol.. 26, 83

Mean Squared Error MSE measures the average square error of our predictions. For each point, compute the square difference between the predictions and the target, and then average those values... 55, 83

Particle Swarm Optimization Particle cloud optimization and particle swarm optimization refers to a metaheuristic that evokes the behavior of particles in nature.. 81, 83

PowerLine The use of the existing power network (230V) in a building for data transmission represents a cost-effective means of communication.. 26, 83

Radial Base Function A real function whose values depend only on the distance of the origin.. 22

Radio Frequency They are a type of network where different frequencies (RF channels or radio frequencies) are used to transmit audiovisual content. There are two prominent types of multi-frequency networks, horizontal and vertical.. 26, 83

Root Mean Squared Error The RMSE gives the error value the same dimensionality as the actual and predicted values. The smaller value of RMSE indicates the better performance of the model.. 80, 83

Smart Home It is a smart home, where appliances and air conditioning, ventilation, lighting systems, in addition to audio and video systems or security systems can communicate with each other and can be controlled remotely via the internet. These integrated home automation systems, together with other measurement and control devices such as sensors, can convert a home into a digital home prepared to help control the consumption of the home.. 5, 11, 12, 25

SVM It is a supervised learning algorithm that can be used for binary classification or regression, being very effective in applications such as natural language processing, speech, image recognition, and artificial vision. A support vector machine constructs an optimal hyperplane in the form of a decision surface.
"That the margin of separation between the two classes in the data is maximized. Support vectors refer to a small subset of the training observations that are used as support for the optimal location of the decision surface" Chen et al.. 5, 12, 14, 17, 21

Twisted-pair cable The twisted pair of twisted pairs (TP) is by far the most widely used means of communication in KNX installations. All participants are connected via the bus. The cable has a low cost, and its installation is simple.. 26, 83

References

[1] Kemal Akkaya et al. "IoT-based Occupancy Monitoring Techniques for Energy-Efficient Smart Buildings". In: *2015 IEEE WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE WORKSHOPS (WCNCW)*. IEEE Wireless Communications and Networking Conference Workshops. IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, MAR 09-12, 2015. IEEE. 2015, 58–63. ISBN: 978-1-4799-8760-3 (cit. on p. 12).

[2] Amir H. Alavi et al. "Internet of Things-enabled smart cities: State-of-the-art and future trends". In: *Measurement* 129 (2018), pp. 589–606. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2018.07.067. URL: http://www.sciencedirect.com/science/article/pii/S0263224118306912 (cit. on p. 26).

[3] Isabelle Guyon Alexander Statnikov Douglas Hardin. "A Gentle Introduction to Support Vector Machines in Biomedicine". In: WORLD SCIENTIFIC, 2011, pp. 65–69. DOI: 10.1142/7922. URL: https://pdfs.semanticscholar.org/e528/a9cdde8dc310e0c1acdf2b9201c44d4217b4.pdf (cit. on p. 22).

[4] S. M. Ali et al. "Big data visualization: Tools and challenges". In: *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. Dec. 2016, pp. 656–660. DOI: 10.1109/IC3I.2016.7918044 (cit. on p. 15).

[5] Cloud Security Alliance. *Top Ten Big Data Security and Privacy Challenges*. 2012. URL: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Top_Ten_v1.pdf (cit. on p. 11).

[6] B. Alotaibi. "Utilizing Blockchain to Overcome Cyber Security Concerns in the Internet of Things: A Review". In: *IEEE Sensors Journal* 19.23 (Dec. 2019), pp. 10953–10971. ISSN: 2379-9153. DOI: 10.1109/JSEN.2019.2935035 (cit. on p. 12).

[7] AltexSoft. *Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson*. https://medium.com/datadriveninvestor/machine-learning-as-a-service-mlaas-is-the-next-trend-no-one-is-talking-about-e100973121c1. 2019 (cit. on pp. 27, 28).

[8] Krzysztof Arendt et al. "Room-level occupant counts, airflow and CO 2 data from an office building". In: Nov. 2018, pp. 13–14. DOI: 10.1145/3277868.3277875 (cit. on p. 12).

[9] J.A. Asensio et al. "Emulating home automation installations through component-based web technology". In: *Future Generation Computer Systems* 93 (2019). cited By 2, pp. 777–791. DOI: 10.1016/j.future.2017.09.062. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85032389621&doi=10.1016%2fj.future.2017.09.062&partnerID=40&md5=5124734b2798df4c8bd80a99dc260a54 (cit. on p. 26).

[10] KNX Association. *Energy Efficiency with KNX*. URL: https://www2.knx.org/media/docs/Flyers/Energy-Efficiency-With-KNX/Energy-Efficiency-With-KNX_en.pdf (cit. on p. 25).

[11] KNX Association. *KNX*. URL: https://www2.knx.org/in/ (cit. on p. 25).

[12] KNX Association. *KNX System arguments*. URL: http://knx.com.ua/attachments/article/132/KNX-basic_course_full.pdf (cit. on p. 26).

[13] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." In: *J. Mach. Learn. Res.* 13 (2012), pp. 281–305. URL: http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12 (cit. on p. 63).

[14] Bobaos. *Bobaos KNX datapoint connection*. URL: https://github.com/bobaos/bdsd.python (cit. on p. 32).

[15] Luis Candanedo Ibarra and Veronique Feldheim. "Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models". In: *Energy and Buildings* 112 (Dec. 2015). DOI: 10.1016/j.enbuild.2015.11.071 (cit. on p. 12).

[16] by Caner Savas and Fabio Dovis. "The Impact of Different Kernel Functions on the Performance of Scintillation Detection Based on Support Vector Machines". In: (2019). URL: https://www.mdpi.com/1424-8220/19/23/5219/htm (cit. on p. 18).

[17] Duo Chen et al. "A high-performance seizure detection algorithm based on Discrete Wavelet Transform (DWT) and EEG". In: *PLOS ONE* 12.3 (Mar. 2017), pp. 1–21. DOI: 10.1371/journal.pone.0173138. URL: https://doi.org/10.1371/journal.pone.0173138 (cit. on p. 85).

[18] Michael Chui. *Notes from the AI Frontier: Applications and Value of Deep Learning*. 2018. URL: https://www.mckinsey.com/~/media/McKinsey/Featured%5C%20Insights/Artificial%5C%20Intelligence/Notes%5C%20from%5C%20the%5C%20AI%5C%20frontier%5C%20Applications%5C%20and%5C%20value%5C%20of%5C%20deep%5C%20learning/Notes-from-the-AI-frontier-Insights-from-hundreds-of-use-cases-Discussion-paper.ashx (cit. on p. 27).

[19] Cisco. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. 2015 (cit. on p. 11).

[20] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning*. 1995, pp. 273–297 (cit. on p. 17).

[21] Thomas M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Trans. Electronic Computers* 14 (1965), pp. 326–334 (cit. on p. 18).

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000, p. 7. DOI: 10.1017/CBO9780511801389 (cit. on p. 17).

[23] Antonino Crivello et al. “Detecting occupancy and social interaction via energy and environmental monitoring”. In: *INTERNATIONAL JOURNAL OF SENSOR NETWORKS* 27.1 (2018), 61–69. ISSN: 1748-1279. DOI: {10.1504/IJSNET.2018.10013426} (cit. on p. 12).

[24] Simona D’Oca and Tianzhen Hong. “Occupancy schedules learning process through a data mining framework”. In: *ENERGY AND BUILDINGS* 88 (Feb. 2015), 395–408. ISSN: 0378-7788. DOI: {10.1016/j.enbuild.2014.11.065} (cit. on p. 12).

[25] Deloitte. *Big data Time for a lean approach in financial services*. URL: https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/deloitte-analytics/big-data.pdf (cit. on p. 11).

[26] George Demiris. “Privacy and Social Implications of Distinct Sensing Approaches to Implementing Smart Homes for Older Adults”. In: *2009 ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, VOLS 1-20*. IEEE Engineering in Medicine and Biology Society Conference Proceedings. Annual International Conference of the IEEE-Engineering-in-Medicine-and-Biology-Society, Minneapolis, MN, SEP 03-06, 2009. IEEE Engn Med & Biol Soc. 2009, 4311–4314. ISBN: 978-1-4244-3295-0. DOI: {10.1109/IEMBS.2009.5333800} (cit. on p. 12).

[27] Tatjana Eitrich and Bruno Lang. “Efficient Optimization of Support Vector Machine Learning Parameters for Unbalanced Datasets”. In: *J. Comput. Appl. Math.* 196.2 (Nov. 2006), pp. 425–436. ISSN: 0377-0427. DOI: 10.1016/j.cam.2005.09.009. URL: https://doi.org/10.1016/j.cam.2005.09.009 (cit. on p. 55).

[28] Shuzhan Fan. *Understanding the mathematics behind Support Vector Machines*. https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/. 2018 (cit. on p. 20).

[29] Bahar Farahani, Farshad Firouzi, and Krishnendu Chakrabarty. “Healthcare IoT”. In: Jan. 2020, pp. 515–545. ISBN: 978-3-030-30366-2. DOI: 10.1007/978-3-030-30367-9_11 (cit. on p. 25).

[30] Frauke Friedrichs and Christian Igel. “Evolutionary tuning of multiple SVM parameters”. In: *Neurocomputing* 64 (Mar. 2005), pp. 107–117. DOI: 10.1016/j.neucom.2004.11.022 (cit. on p. 55).

[31] Salvador García, Julián Luengo, and Francisco Herrera. “Data Preparation Basic Models”. In: *Data Preprocessing in Data Mining*. Cham: Springer International Publishing, 2015, pp. 39–57. ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_3. URL: https://doi.org/10.1007/978-3-319-10247-4_3 (cit. on p. 15).

[32] Trevor Hastie Gareth James Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf. 2009, pp. 338–339 (cit. on p. 18).

[33] Trevor Hastie Gareth James Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf. 2009, pp. 346–349 (cit. on pp. 19, 21, 23, 24, 56).

[34] Google. *Cloud Pub/Sub triggers*. URL: https://firebase.google.com/docs/functions/pubsub-events (cit. on p. 32).

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001. URL: https://web.stanford.edu/~hastie/Papers/ESLII.pdf (cit. on p. 58).

[36] Helium. *Welcome to Helium*. URL: https://www.helium.com/about (cit. on p. 31).

[37] Francisco Herrera. *Data Preprocessing*. URL: https://sci2s.ugr.es/sites/default/files/files/publications/books/slides/Summer-School-BDML_Data-Preprocessing-Wroclaw-2015.pdf (cit. on p. 15).

[38] Y.C. Ho and D.L. Pepyne. "Simple explanation of the no-free-lunch theorem and its implications". In: *Journal of Optimization Theory and Applications* 115.3 (2002). cited By 159, pp. 549–570. DOI: 10.1023/A:1021251113462. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0036437286&doi=10.1023%2fA%3a1021251113462&partnerID=40&md5=8e273a200f63fc9336d1fd48da4d47af (cit. on p. 15).

[39] Brodie W. Hobson et al. "Opportunistic occupancy-count estimation using sensor fusion: A case study". In: *BUILDING AND ENVIRONMENT* 159 (July 2019). ISSN: 0360-1323. DOI: {10.1016/j.buildenv.2019.05.032} (cit. on p. 12).

[40] IBM. *Watson Machine Learning*. 2020. URL: https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-overview.html (cit. on p. 29).

[41] Dimosthenis Ioannidis et al. "Building Multi-occupancy Analysis and Visualization Through Data Intensive Processing". In: *ARTIFICIAL INTELLIGENCE APPLICATIONS AND INNOVATIONS, AIAI 2016*. Ed. by Iliadis, L and Maglogiannis, I. Vol. 475. IFIP Advances in Information and Communication Technology. 12th IFIP WG 12.5 International Conference on Artificial Intelligence Applications and Innovations (AIAI), Thessaloniki, GREECE, SEP 16-18, 2016. Int Federat Informat Proc Working Grp 12 5. 2016, 587–599. ISBN: 978-3-319-44944-9. DOI: {10.1007/978-3-319-44944-9_52} (cit. on p. 12).

[42] ISO. *ISO/IEC 20922:2016*. 2016. URL: https://www.iso.org/standard/69466.html (cit. on p. 30).

[43] Alan Julian Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 0387781889 (cit. on p. 56).

[44] Vikramaditya Jakkula. *Tutorial on Support Vector Machine (SVM) Vector Machines*. https://pdfs.semanticscholar.org/7cc8/3e98367721bfb908a8f703ef5379042c4bd9.pdf (cit. on p. 20).

[45] Michael I. Jordan. *Advanced Topics in Learning and Decision Making*. URL: https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf (cit. on p. 22).

[46] M. Jung et al. "Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture". In: *2013 27th International Conference on Advanced Information Networking and Applications Workshops*. 2013, pp. 1361–1367 (cit. on p. 26).

[47] Kibum Kim, Ahmad Jalal, and Maria Mahmood. "Vision-Based Human Activity Recognition System Using Depth Silhouettes: A Smart Home System for Monitoring the Residents". In: *JOURNAL OF ELECTRICAL ENGINEERING & TECHNOLOGY* 14.6 (Nov. 2019), 2567–2573. ISSN: 1975-0102. DOI: {10.1007/s42835-019-00278-8} (cit. on p. 12).

[48] Alexandre Kowalczyk. *Support Vector Machines Succinctly*. http://resurse.devilstrike.ro/facultate/master/support_vector_machines_succinctly.pdf. 2017 (cit. on pp. 20, 23).

[49] A. Krylovskiy. "Internet of Things gateways meet linux containers: Performance evaluation and discussion". In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 222–227 (cit. on p. 26).

[50] Fadwa Lachhab et al. "Context-driven monitoring and control of buildings ventilation systems using big data and Internet of Things-based technologies". In: *PROCEEDINGS OF THE INSTITUTION OF MECHANICAL ENGINEERS PART I-JOURNAL OF SYSTEMS AND CONTROL ENGINEERING* 233.3 (Mar. 2019), 276–288. ISSN: 0959-6518. DOI: {10.1177/0959651818791406} (cit. on p. 12).

[51] Cate Lawrence. *Helium Brings Connectivity Innovation to IoT*. URL: https://dzone.com/articles/helium-brings-connectivity-innovation-to-iot (cit. on p. 31).

[52] Sai Li, Huajing Fang, and Xiaoyong Liu. "Parameter optimization of support vector regression based on sine cosine algorithm". In: *Expert Systems with Applications* 91 (2018), pp. 63–77. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.08.038. URL: http://www.sciencedirect.com/science/article/pii/S0957417417305833 (cit. on p. 81).

[53] Edoardo Longo, Alessandro E. C. Redondi, and Matteo Cesana. "Accurate occupancy estimation with WiFi and bluetooth/BLE packet capture". In: *COMPUTER NETWORKS* 163 (Nov. 2019). ISSN: 1389-1286. DOI: {10.1016/j.comnet.2019.106876} (cit. on p. 12).

[54] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. USA: Society for Industrial and Applied Mathematics, 2000. ISBN: 0898714540 (cit. on p. 17).

[55] Microsoft. *What is Azure Machine Learning?* 2019. URL: https://docs.microsoft.com/es-es/azure/machine-learning/overview-what-is-azure-ml (cit. on p. 27).

[56] Norazian Mohamed Noor et al. “Comparison of Linear Interpolation Method and Mean Method to Replace the Missing Values in Environmental Data Set”. In: *Materials Science Forum* 803 (Aug. 2014), pp. 278–281. DOI: 10.4028/www.scientific.net/MSF.803.278 (cit. on p. 54).

[57] Sachi Nandan Mohanty et al. “An efficient Lightweight integrated Blockchain (ELIB) model for IoT security and privacy”. In: *Future Generation Computer Systems* 102 (2020), pp. 1027–1037. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2019.09.050. URL: http://www.sciencedirect.com/science/article/pii/S0167739X19319843 (cit. on p. 12).

[58] N. Noury et al. “Building an Index of Activity of Inhabitants From Their Activity on the Residential Electrical Power Line”. In: *IEEE Transactions on Information Technology in Biomedicine* 15.5 (Sept. 2011), pp. 758–766. ISSN: 1558-0032. DOI: 10.1109/TITB.2011.2138149 (cit. on p. 12).

[59] L. Pasolli, C. Notarnicola, and L. Bruzzone. “Multi-Objective Parameter Optimization in Support Vector Regression: General Formulation and Application to the Retrieval of Soil Moisture From Remote Sensing Data”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 5.5 (2012), pp. 1495–1508 (cit. on p. 81).

[60] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 24, 55, 68).

[61] Sasa Pesic et al. “BLEMAT: Data Analytics and Machine Learning for Smart Building Occupancy Detection and Prediction”. In: *INTERNATIONAL JOURNAL ON ARTIFICIAL INTELLIGENCE TOOLS* 28.6, SI (Sept. 2019). ISSN: 0218-2130. DOI: {10.1142/S0218213019600054} (cit. on p. 12).

[62] Dorian Pyle. *Data preparation for data mining*. English. Includes bibliographical references and index. San Francisco, Calif. : Morgan Kaufmann Publishers, 1999. ISBN: 1558605290. URL: http://www.loc.gov/catdir/toc/els033/99017280.html (cit. on p. 15).

[63] Ana Belén García Hernando Qin Ni and Iván Pau De la Cruz. *The Elderly's Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development*. https://www.mdpi.com/1424-8220/15/5/11312/htm. 2015 (cit. on p. 12).

[64] Qiujun Huang, Jingli Mao, and Yong Liu. “An improved grid search algorithm of SVR parameters optimization”. In: *2012 IEEE 14th International Conference on Communication Technology*. 2012, pp. 1022–1026 (cit. on p. 81).

[65] F. Samuelson and D. G. Brown. “Application of Cover's theorem to the evaluation of the performance of CI observers”. In: *The 2011 International Joint Conference on Neural Networks*. July 2011, pp. 1020–1026. DOI: 10.1109/IJCNN.2011.6033334 (cit. on p. 18).

[66] James Scott et al. “PreHeat: Controlling Home Heating Using Occupancy Prediction”. In: Sept. 2011, pp. 281–290. DOI: 10.1145/2030112.2030151 (cit. on p. 12).

[67] JP Skön et al. "Modelling indoor air carbon dioxide (CO2) concentration using neural network". In: *methods* 14.15 (2012), p. 16 (cit. on p. 81).

[68] Stephan Spiegel. "Optimization of In-House Energy Demand". In: *SMART INFORMATION SYSTEMS: COMPUTATIONAL INTELLIGENCE FOR REAL-LIFE APPLICATIONS*. Ed. by Hopfgartner, F. Advances in Computer Vision and Pattern Recognition. 2015, 271–289. ISBN: 978-3-319-14178-7. DOI: {10.1007/978-3-319-14178-7_10} (cit. on p. 11).

[69] Enrique J. Carmona Suárez. *Tutorial sobre Máquinas de Vectores Soporte (SVM)*. http://www.ia.uned.es/~ejcarmona/publicaciones/[2013-Carmona]%20SVM.pdf. 2014 (cit. on p. 19).

[70] Helium Systens. *Helium client python documentation*. URL: https://helium.github.io/helium-client-python/ (cit. on p. 32).

[71] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. URL: http://www.jstor.org/stable/2346178 (cit. on p. 57).

[72] MELLISA TOLENTINO. *1950s Smart homes : Future in the past*. URL: https://siliconangle.com/2014/02/05/1950s-smart-homes-future-in-the-past/ (cit. on p. 26).

[73] M.Y. Toylan and E. Cetin. "Design and application of a KNX-based home automation simulator for smart home system education". In: *Computer Applications in Engineering Education* 27.6 (2019). cited By 1, pp. 1465–1484. DOI: 10.1002/cae.22162. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85070799104&doi=10.1002%2fcae.22162&partnerID=40&md5=49c701b1fc67c1a42987c026e2716229 (cit. on p. 25).

[74] Mehmet Toylan and Engin Cetin. "Design and application of a KNX-based home automation simulator for smart home system education". In: *Computer Applications in Engineering Education* 27 (Aug. 2019). DOI: 10.1002/cae.22162 (cit. on p. 25).

[75] Hank Tucker. *Blockchain Startup Makes Wireless Internet Cheaper, Lands Lime Scooters And Nestle As Clients*. URL: https://www.forbes.com/sites/hanktucker/2019/06/12/blockchain-startup-makes-wireless-internet-cheaper-lands-lime-scooters-and-nestle-as-clients/ (cit. on p. 31).

[76] José David Rossi Vellido Alcacena Alfredo Martín. "Seeing is believing: the importance of visualization in real-world machine learning applicationss". In: 2011. ISBN: 978-2-87419-044-5. URL: https://upcommons.upc.edu/bitstream/handle/2117/20273/Vellido.pdf?sequence=1&isAllowed=y (cit. on p. 15).

[77] Felix Walcher. *KNX to MQTT/AMQP*. 2019. URL: https://www.auto.tuwien.ac.at/bib/pdf_TR/TR0194.pdf (cit. on p. 30).

[78] Fulin Wang et al. “Predictive control of indoor environment using occupant number detected by video data and CO2 concentration”. In: *ENERGY AND BUILDINGS* 145 (June 2017), 155–162. ISSN: 0378-7788. DOI: {10.1016/j.enbuild.2017.04.014} (cit. on p. 12).

[79] Wei Wang et al. “Modeling occupancy distribution in large building spaces for HVAC energy efficiency”. In: *Energy Procedia* 152 (Oct. 2018), pp. 1230–1235. DOI: 10.1016/j.egypro.2018.09.174 (cit. on p. 12).

[80] D.H. Wolpert and W.G. Macready. “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997). cited By 4765, pp. 67–82. DOI: 10.1109/4235.585893. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0031118203&doi=10.1109%2f4235.585893&partnerID=40&md5=fac8c56be911367d556066800e863066 (cit. on p. 15).

[81] J.K.W. Wong, H. Li, and S.W. Wang. “Intelligent building research: a review”. In: *Automation in Construction* 14.1 (2005), pp. 143–159. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2004.06.001. URL: http://www.sciencedirect.com/science/article/pii/S0926580504000536 (cit. on p. 84).

[82] Xiangying Wang and Yixin Zhong. “Statistical learning theory and state of the art in SVM”. In: *The Second IEEE International Conference on Cognitive Informatics, 2003. Proceedings.* Aug. 2003, pp. 55–59. DOI: 10.1109/COGINF.2003.1225953 (cit. on p. 17).

[83] D. Zhang et al. “Parameter Optimization for SVR Based on Genetic Algorithm and Simplex Method”. In: *2010 Chinese Conference on Pattern Recognition (CCPR)*. 2010, pp. 1–6 (cit. on p. 81).

[84] X. Zhou and J. Yang. “Parameters optimization of air conditioning load prediction model based on PSO-SVR”. In: *Proceedings of the 32nd Chinese Control Conference*. 2013, pp. 1777–1782 (cit. on p. 81).