

Universidad Autónoma de Querétaro



Facultad De Ingeniería LICENCIATURA EN INGENIERÍA FÍSICA

(ENERO 2019 – MARZO 2023)

Red Neuronal Recurrente de Memoria a Largo y Corto Plazo Optimizada por Procesos Gaussianos para Predecir la Contaminación Atmosférica

Alumno:

Marco Antonio Olgún Sánchez

Presidente:

Dr. Marco Antonio Aceves Fernández

Secretario:

Dr. Saul Tovar Arriaga

Vocal:

Dr. Josué De Jesús Trejo Alonso

21 de marzo de 2023



Dirección General de Bibliotecas y Servicios Digitales
de Información



Red Neuronal Recurrente de Memoria a Largo y Corto
Plazo Optimizada por Procesos Gaussianos para
Predecir la Contaminación Atmosférica

por

Marco Antonio Olguin Sánchez

se distribuye bajo una [Licencia Creative Commons
Atribución-NoComercial-SinDerivadas 4.0
Internacional](#).

Clave RI: IGLIN-243663

Este trabajo está dedicado a mis padres quienes siempre me han apoyado en todo a lo largo de la vida. A mi hermano por siempre ser un apoyo, y al Dr. Aceves por su guía profesional y ética en esta etapa de la vida.

Table of Contents

| | |
|--|-----------|
| 1. Introduction..... | 7 |
| 1.1 Motivation..... | 7 |
| 1.2 Problem Formulation..... | 7 |
| 1.3 Objectives..... | 8 |
| 1.3.1 Specific Objectives..... | 8 |
| 1.4 Hypothesis..... | 8 |
| 1.5 Thesis Structure..... | 8 |
| 2. Literature Survey | 8 |
| 2.1 Airborne Pollution | 8 |
| 2.2 Particulate Matter | 8 |
| 2.2.1 <i>PM2.5</i> | 9 |
| 2.3 Artificial Intelligence..... | 9 |
| 2.3.1 Machine Learning..... | 10 |
| 2.3.2 Deep Learning..... | 10 |
| 2.4 Neural Network Architectures | 11 |
| 2.4.1 Feedforward Networks | 11 |
| 2.4.2 Convolutional Neural Networks | 11 |
| 2.4.3 Recurrent Neural Networks | 12 |
| 2.5 Probability Theory | 13 |
| 2.5.1 Discrete Probability Distributions..... | 14 |
| 2.5.2 Continuous Probability Distributions | 14 |
| 2.5.3 Stochastic Process | 15 |
| 2.6 Gaussian Processes..... | 15 |
| 2.6.1 Gaussian Process in Machine Learning..... | 16 |
| Multiple Imputation by Chained Equations | 17 |
| 3. Methodology..... | 18 |
| 3.1 Requirements | 18 |
| 3.1.1 System Requirements | 18 |
| 3.1.2 Python Libraries | 18 |
| 3.2 Database Acquisition | 18 |
| The SIMAT counts with more than 40 stations (SEDEMA, 2022) to measure the quality of the air in the metropolitan area, Ciudad de México (CDMX) and surrounding area of Estado de México. Also, at the time of completion of this project just the stations show in the <i>Table 1</i> has the capability of measure <i>PM2.5</i> | 18 |
| 3.3 Preprocessing | 19 |
| 3.3.1 Missing Values..... | 19 |

| | |
|---|-----------|
| 4. Results and Discussion | 24 |
| 4.1 LSTM Baseline Model..... | 24 |
| 4.2 LSTM Model Optimized with GP | 28 |
| 4.3 GP Model vs Baseline Model..... | 32 |
| 4.4 Discussion..... | 35 |
| 4.4.1 Significance/Impact..... | 35 |
| 4.4.2 Future Work | 35 |
| Bibliography..... | 35 |

Abstract

Forecasting air pollution is a challenging problem today that requires special attention in large cities since they are home to millions of people who are at risk of respiratory diseases every day. At the same time, there has been exponential growth in the research and application of deep learning, which is useful to treat temporary data such as pollution levels, leaving aside the physical and chemical characteristics of the particles and only focusing on predicting the next levels of contamination. This work seeks to contribute to society by presenting a useful way to optimize recurrent neural networks of the short and long-term memory type through a statistical process (Gaussian processes) for the correct optimization of the processes.

1. Introduction

Recurrent neural networks (RNN), especially Long Short Term Memory (LSTM) have proved their efficiency working on time-dependent values by (as its names indicate) the use of memory (sequences) gives enough information to the network to work properly finding patterns and trends in the values, which are not so obvious at first glance. Also, the gaussian processes are a useful statistical technique that allows the hyperparameters of the network since it has shown that the processing time is reduced and, at the same time, the accuracy may be improved. Afterward, there is airborne pollution, which is a complex system that affects billions of people worldwide, especially in a metropolis like in the case of this study, Mexico City, Mexico. Also, there are a lot of types of particles interacting with each other in chemical, biological, and physical ways. The pollutants that are monitored by the SEDEMA's network in the City of Mexico are nitrogen dioxide (NO_2), Ozone (O_3), sulfur dioxide (SO_2) and particulate matter ($PM_{2.5}$, and PM_{10}). Hence the propose in this project is the use of an RNN-LSTM and optimizing its hyperparameters using Gaussian processes to increase the accuracy in the forecast of airborne pollution instead of the use of the current methods.

1.1 Motivation

Air pollution is a major public health concern in Mexico City metropolitan area (CDMX), and accurate forecasting of pollution levels could help to mitigate the negative health effects of exposure to polluted air. Therefore, our research group is motivated to study the application of long short-term memory networks (LSTMs) for forecasting airborne pollution in CDMX. LSTMs are a powerful type of recurrent neural networks (RNNs) that have been shown to be effective for time-series forecasting, making them a suitable choice for forecasting pollution levels in CDMX. Furthermore, we aim to optimize the LSTM using Gaussian processes (GPs) which are a powerful tool for optimization and uncertainty estimation in machine learning, and have been shown to be particularly effective for optimizing RNNs such as LSTMs. By optimizing the LSTM using GPs, we can take advantage of the probabilistic interpretation of the optimization problem provided by GPs, which can be useful for determining the confidence in the model's predictions and identifying regions of the parameter space where the optimization is particularly uncertain. Additionally, GPs can handle missing or corrupted data, which is a common problem in many RNN applications, and are robust to noise in the data. Our goal is to improve the accuracy of the forecasted pollution using GPs, which can be beneficial in situations where the data is incomplete or corrupted, or noisy. Finally, we also plan to explore the integration of GPs with other machine learning techniques such as Bayesian optimization to enhance the optimization process.

1.2 Problem Formulation

Every day, airborne pollution affects millions of people in large cities and reduces their life expectancy giving to them several respiratory and cardiovascular issues (Stephens, et al., 2008) The total suspended particles (TSP) has effects on human health not only related with the respiratory system, they also lead to cardiovascular diseases (Aceves-Fernandez, Estrada, Pedraza-Ortega, Gorrostiera-Hurtado, & Tovar-Arriaga, 2015). Airborne pollution also have the potential to make histopathological changes in the population (Ferman-Cano, Padialla-Santamaria, Moreno-Venegas, & et all, 2018).

Mexico City has been studied for decades (Stephens, et al., 2008) due to its high levels of pollution that affect more than 20 million people. Considering the relevance of the problem is that it is propose to model and forecast the $PM_{2.5}$ using deep neural networks due to its complexity. Continuing with previously works which tried to solve this topic (Becerra Rico, Aceves-Fernandez, Escalante, & Pedraza-Ortega, 2020) using Long-Short Term Memory (LSTM), this time is used gaussian processes to tune the hyperparameters of the networks giving more accuracy and reducing the time looking for it.

1.3 Objectives

In this project, the general objective is to apply the gaussian processes theory into a recurrent neural network and show the advantages as well its disadvantages.

1.3.1 Specific Objectives

The specific objectives of this project are as follows:

- Acquire data.
- Preprocess the raw data dealing with missing values.
- Generate a robust and adjustable recurrent neural network model.
- Apply the gaussian process to the model.
- Benchmark the optimized model with a baseline model non-optimized.
- Analyze the results of the model in order to determine the accuracy.

1.4 Hypothesis

For it's own statistical nature, gaussian process could be an efficient way to optimize the hyperparameter since them could seen as a multivariable function. An optimize LSTM model could be more accurate in the time than a simple LSTM, and also with more time in the forecasting window.

1.5 Thesis Structure

The thesis is organized as follows:

- The first chapter presents the motivation of the project, the problem to be dealt with, the objectives of the project and the hypothesis presented to deal with the project.
- The second chapter presents a summary of the knowledge necessary to deal with the subject.
- The third chapter explains the methodology followed to carry out the project.
- The fourth chapter presents the results of this project, as well as presents the discussion of these.
- Finally, the bibliography used and the annexes are presented.

2. Literature Survey

2.1 Airborne Pollution

Particulates, and biological molecules. These substances can come from a variety of sources, including industrial and agricultural activities, vehicle exhaust, and natural sources. Airborne pollution can have a range of negative effects on human health, including respiratory and cardiovascular disease, cancer, and other illnesses (Arbex, Nacimiento Saldiva, Amador Pereira, & Ferreira Braga, 2010), (Guo, et al., 2010). It can also damage the environment, including plants, animals, and natural habitats. This contamination is accompanied by the same pollutant particles that have a useful life depending on physical parameters (size and shape) and their chemical composition. Governments and organizations around the world are working to reduce airborne pollution through regulations, monitoring, and other measures.

2.2 Particulate Matter

Particulate matter, also known as particle pollution or PM, is a term used to describe a mixture of solid particles and liquid droplets found in the air. These particles can come from a variety of sources, including vehicle exhaust, construction activities, and natural sources like dust and pollen. Particulate matter can be divided into

two main categories: PM_{10} , which refers to particles with a diameter of 10 micrometers or less, and $PM_{2.5}$, which refers to particles with a diameter of 2.5 micrometers or less. $PM_{2.5}$ is considered more harmful to human health because it is small enough to be inhaled deep into the lungs. Exposure to high levels of particulate matter can cause respiratory and cardiovascular problems, and it has been linked to a range of other health effects. The United States Environmental Protection Agency (EPA) defines particulate matter (PM) as:

The fine particles that remain suspended in the air. These in turn are divided into three categories according to their size: coarse particles (PM_{10}), fine particles ($PM_{2.5}$) and ultrafine particles ($PM_{0.1}$) (EPA, 2022).

2.2.1 $PM_{2.5}$

According to the published literature, we surmise that particulate matter (PM) concentration, individually, may be less important than components in explaining health effects. $PM_{2.5}$ (aerodynamic diameter $< 2.5 \mu m$) had similar cytotoxicity (e.g., cell viability reduction, oxidative damage, inflammatory effects and genetic toxicity) on different types of cells.

2.3 Artificial Intelligence

Artificial intelligence (AI) is the simulation of human intelligence in machines that are programmed to think and act like humans (Demuth, Beale, De Jess, & Hagan, 2014). Artificial intelligence (AI) was born with the perceptron in 1949 by Donald Hebb. Subsequently, research on it increased during the 1960s with the arrival of computers, but due to the limitations that it presented for its time, it was not until the end of the nineties that there was more interesting research and applications. These intelligent machines are capable of learning, adapting, and making decisions based on the data they receive.

AI has been a topic of fascination and speculation for decades, and with the rapid advancements in technology in recent years, it has become a reality in many fields. AI is used in a variety of applications, such as speech recognition, image and facial recognition, natural language processing, and autonomous vehicles.

One of the main goals of AI research is to create machines that can think and act intelligently, without being explicitly programmed to do so. This is known as "general AI," and it is a highly ambitious and challenging goal. Another of the most well-known approaches to achieving general AI is through the use of machine learning algorithms. These algorithms allow machines to learn from data, without being explicitly programmed. This is done by feeding the machine large amounts of data and letting it "learn" the patterns and relationships in the data (Russell & Norvig, 2010).

Another approach to AI is through the use of artificial neural networks, which are inspired by the structure and function of the human brain. These networks consist of multiple interconnected nodes, known as neurons, which are able to process and transmit information. Through a process called "training", the network is able to learn from data and make predictions or decisions based on that data (Charu, 2018), (Poole & Mackworth, 2017).

Despite the impressive capabilities of AI, it is important to recognize that it is still a developing field, and there are many challenges and limitations to overcome. One of the main challenges is ensuring that AI systems are able to make decisions that are ethical and fair. Another challenge is ensuring that AI systems are able to explain their decision-making processes, so that humans can understand and trust them (Burkov, 2019), (Amodei & Olah, 2018).

In conclusion, artificial intelligence is a rapidly developing field that has the potential to revolutionize many aspects of our lives. While there are many challenges and limitations to overcome, the future of AI is exciting and holds great promise.

2.3.1 Machine Learning

Machine learning is a subset of artificial intelligence that involves training algorithms to make predictions or take actions based on data. In other words, machine learning algorithms use data to learn for themselves, rather than being explicitly programmed with a set of rules. This allows them to improve over time, making more accurate predictions or taking more effective actions.

One key difference between machine learning and artificial intelligence is that machine learning algorithms are limited to the specific task they were trained for, whereas artificial intelligence refers to the ability of a machine or system to exhibit intelligent behavior that is comparable to that of a human. In other words, artificial intelligence is a broader concept that encompasses many different techniques and technologies, including machine learning (Russell & Norvig, 2010), (Goodfellow, Bengio, & Courville, 2016), (Mitchell, 1997).

Another important difference is that machine learning algorithms require a large amount of data to learn from, whereas artificial intelligence can incorporate a wider range of information, including unstructured data and subjective information such as human emotions and intentions (Nilsson, 1998), (Negnevitsky, 2005), (Burkov, 2019), (Luger, 1994)

IBM gives a more accurate definition, it follows:

The information output that is generated when you train your machine learning algorithm on data. After training, providing a model with an input will give it an output. For example, a predictive algorithm will create a predictive model.
(IBM, 2022)

2.3.2 Deep Learning

Deep learning is a subset of machine learning that involves training algorithms called neural networks to make predictions or take actions based on data. These neural networks are designed to mimic the structure and function of the human brain, and they can learn to recognize patterns in data and make decisions based on that data (Goodfellow, Bengio, & Courville, 2016).

One key advantage of deep learning is that it can handle very large and complex datasets, which makes it well-suited to a wide range of applications. For example, deep learning algorithms can be used for image recognition, natural language processing, and many other tasks that require processing large amounts of data (Berg, Berg, & Yang, 2018), (Goldbeg, 2017), (Brownlee, 2019).

Another advantage of deep learning is that it can learn from unstructured data, such as images or text, without the need for manual feature extraction. This allows deep learning algorithms to learn from data directly, without the need for human intervention (Singh, 2018), (Chollet, 2018), (Gibson & Patterson, 2018).

Deep learning is a powerful tool for making predictions or taking actions based on data, and it has a wide range of applications in many different fields. With its ability to handle large and complex datasets, deep learning has the potential to revolutionize many areas of applications in many different fields. With its ability to handle large and complex datasets, deep learning has the potential to revolutionize many areas of science and technology.

2.4 Neural Network Architectures

Neural networks are a type of algorithm used in machine learning. They are inspired by the structure and function of the human brain, and they are composed of many interconnected processing units called neurons. Neural networks can be organized into different architectures, which refer to the way the neurons are connected and arranged into layers. Some common architectures include feedforward networks, convolutional neural networks, and recurrent neural networks (Charu, 2018). The choice of architecture for a neural network depends on the specific task it will be used for and the type of data it will be applied to. Different architectures can have different properties and capabilities, and they can be combined in various ways to create more complex and powerful models (Goodfellow, Bengio, & Courville, 2016).

2.4.1 Feedforward Networks

Feedforward networks are a type of artificial neural network that consists of multiple layers of interconnected neurons. These networks are called "feedforward" because the information flows through the network in a single direction, from the input layer to the output layer, without any loops or feedback connections (Charu, 2018), (Goodfellow, Bengio, & Courville, 2016).

The mathematical foundation of feedforward networks is based on the concept of an artificial neuron, which is a simple mathematical model of a biological neuron. An artificial neuron receives input from other neurons, processes this input using a set of weights and biases, and produces an output signal (Chollet, 2018). The output signal is then passed on to other neurons in the network. The behavior of an artificial neuron can be described mathematically using the following equations:

$$\begin{aligned} a_i^{(l)} &= g \left(\sum_{j=1}^{n_l} w_{i,j}^{(l)} a_j^{(l-1)} + b_i^{(l)} \right) \\ \hat{y} &= g \left(\sum_{i=1}^{n_L} w_i^{(L)} a_i^{(L-1)} + b^{(L)} \right) \end{aligned} \tag{1}$$

where $a_i^{(l)}$ is the activation of the i -th neuron in the l th layer, $g(\cdot)$ is the activation function, $w_{i,j}^{(l)}$ is the weight from the j -th neuron in the $(l-1)$ th layer to the i -th neuron in the l -th layer, $b_i^{(l)}$ is the bias of the i -th neuron in the l -th layer, n_l is the number of neurons in the l -th layer, n_L is the number of neurons in the output layer, \hat{y} is the predicted output of the network, and $w_i^{(L)}$ and $b^{(L)}$ are the weights and bias of the output layer, respectively.

2.4.2 Convolutional Neural Networks

A convolutional neural network (CNN) is a type of artificial neural network that is commonly used in the field of computer vision. This type of network is designed to process data that has a grid-like structure, such as an image. The mathematical foundation of a CNN is the convolution operation, which is a mathematical function that takes two inputs: a matrix of weights and a feature map. The convolution operation applies the weights to the feature map, producing a new feature map called the convolved feature. This process is repeated for each convolutional layer in the network, with the output of one layer serving as the input for the next (Goodfellow, Bengio, & Courville, 2016). The convolution operation is defined as follows:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m]$$

Here, f and g are the input feature map and the matrix of weights, respectively, and n is the index of the element in the output feature map.

In a CNN, the convolution operation is typically followed by a non-linear activation function, such as the rectified linear unit (ReLU), which is defined as:

$$\text{ReLU}(x) = \max(0, x) \quad (2)$$

This function converts all negative values in the output feature map to zero, allowing the network to learn more complex patterns in the data \cite{lecun1990handwritten}. In addition to the convolution operation, CNNs also use pooling layers, which are used to reduce the spatial dimensions of the feature map (Fukushima, 1980). This is done by applying a pooling function, such as max pooling (Nair and Hinton, 2010), to the feature map. This function picks the maximum value from each subregion of the feature map, effectively reducing the number of parameters that the network needs to learn (Nair & Hinton, 2010).

The overall architecture of a CNN is composed of multiple convolutional and pooling layers, followed by one or more fully connected layers at the end. The output of the CNN is typically a vector of probabilities that represents the likelihood that each class is present in the input data. In summary, the mathematical foundations of a convolutional neural network are the convolution operation and the use of non-linear activation functions and pooling layers. These techniques allow the network to learn complex patterns in data and make accurate predictions.

This function converts all negative values in the output feature map to zero, allowing the network to learn more complex patterns in the data. This technique is called ReLU (Rectified Linear Unit) activation function (Goodfellow, Bengio, & Courville, 2016). In addition to the convolution operation, CNNs also use pooling layers, which are used to reduce the spatial dimensions of the feature map. This is done by applying a pooling function, such as max pooling, which picks the maximum value from each subregion of the feature map. This effectively reduces the number of parameters that the network needs to learn, and also helps to reduce overfitting.

2.4.3 Recurrent Neural Networks

Recurrent neural networks (RNN) are a type of neural network that is designed to process sequential data, such as text, audio, or time series data. In a recurrent neural network, the neurons are organized into layers, and the connections between the neurons form a directed cycle, allowing information to flow in both directions. This allows the network to incorporate information from previous time steps, which is important for tasks like language modeling and speech recognition, where the context and meaning of the data depend on the order in which it is presented. Recurrent neural networks can be difficult to train, but they are powerful models that have achieved state-of-the-art performance on many challenging tasks. In practical applications, the use of symbolic values is more common. In a recurrent neural network, there is a one-to-one correspondence between the layers of the network and specific positions in the sequence. The position in the sequence is also known as its timestamp. Finally, RNNs are complete Turing, this means that this type of network can simulate any algorithm with sufficient data and computational resources (Siegelmann & Sontag, 1995).

2.4.3.1 Long-Short Term Memory

To represent the hidden states of the k -th hidden states (layer) the notation $h_t^{-(k)}$ is used and to simplify the notation it will be assumed that the input layer \bar{x}_t can be denoted by $h_t^{-(0)}$ (this layer is not hidden) (Charu, 2018). To obtain good results, a hidden vector of dimension p must also be included, which will be denoted by

$\bar{c}_t^{(k)}$ and refers to the state of the cell. The state of the trap can be observed as the long-term memory within the network. The matrix that updates the values is denoted by $W^{(k)}$ and is used to permute the column vectors $[h_t^{-(k-1)}, h_{t-1}^{-(k)}]^T$. The matrix that is obtained always results in dimensions $4p \times 2p$. A $2p$ size vector is then pre-multiplied by the $W^{(k)}$ matrix resulting in a $4p$ vector. Now to find the updates we have the following; for setting up intermediates the equation 4, for selectively forget and add to long-term memory eq. 5, for selectively leak long-term memory to hidden state eq. 6.

$$\begin{array}{l} \text{Input Gate:} \\ \text{Forget Gate:} \\ \text{Output Gate:} \\ \text{New C - State} \end{array} \begin{bmatrix} \bar{i} \\ \bar{f} \\ \bar{o} \\ \bar{c} \end{bmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad (3)$$

$$\bar{c}_t^{(k)} = \bar{f} \odot \bar{c}_{t-1}^{(k)} + \bar{i} \odot \bar{c} \quad (4)$$

$$\bar{h}_t^{(k)} = \bar{o} \odot \tanh(\bar{c}_t^{(k)}) \quad (5)$$

The final equation does not always use the tanh activation function. The following alternative update may be used:

$$\bar{h}_t^{(k)} = \bar{o} \odot \bar{c}_t^{(k)} \quad (2)$$

Additionally, clarify that LSTM is an algorithm that belongs to recurrent neural networks or RNN (Kuri-Monge, Aceves-Fernandez, Ramirez-Montanez, & Pedraza-Ortega, 2021). The RNN's refer to neural networks that take their previous state as input, this means that the neural network will have two inputs, the new information entered in the network and its previous state. With this model we can have short-term memory in the neural network. These neural networks have applications in sequential predictions, that is, predictions that depend on a temporal variable.

2.5 Probability Theory

Probability theory is a branch of mathematics that deals with the study of random phenomena (William, 1968). It is concerned with quantifying the likelihood of different outcomes of a random event, and provides a framework for making predictions and decisions based on uncertain information (Kolmogorov, 1933). The theory has a wide range of applications in many different fields, including statistics (Cox, 1961), finance (Hull, 2017), engineering (Kleinrock, 1975), and computer science (Sipser, 2013). It is used to model and analyze complex systems, such as weather patterns (Wilks, 2011), stock market fluctuations (Campbell & Shiller, 1997), and the behavior of large networks (Albert & Barabasi, 2009).

One of the key concepts in probability theory is the idea of a probability distribution (Feller, 1968), which describes the likelihood of different outcomes of a random event. These distributions can be either discrete or continuous (Kolmogorov, 1933), depending on the number of possible outcomes.

Discrete probability distributions, such as the binomial (Cramer, 1946) and Poisson (Poisson, 1837) distributions, are used to model the probability of a certain number of outcomes occurring in a fixed number of

trials or events. Continuous probability distributions, such as the normal distribution (Gauss, 1809), are used to model the probability of a continuous variable, such as the height of a person or the temperature of a room.

Another important concept in probability theory is the idea of independence (Kolmogorov, 1933) which describes the relationship between different random events. Two events are considered independent if the occurrence of one event does not affect the likelihood of the other event occurring. Probability theory also includes the concept of conditional probability (Gillespie, 2015), which is the probability of an event occurring given that another event has already occurred. This allows us to make more refined predictions about the likelihood of different outcomes based on additional information.

Probability theory is a powerful and versatile tool (Feller, 1968) that allows us to make predictions and decisions based on uncertain information. Its applications are vast (Kleinrock, 1975) and continue to expand, making it an important area of study in both mathematics and the wider world (von Mises, 1957).

2.5.1 Discrete Probability Distributions

Discrete probability distributions are a fundamental concept in probability theory, which is the mathematical study of random phenomena. These distributions are used to model the probability of different outcomes of a random experiment, such as the probability of rolling a certain number on a die or the probability of a coin toss resulting in heads or tails (Jay L. Devore, 2012).

Unlike continuous probability distributions, which can take on any value within a certain range, discrete probability distributions can only take on a countable number of values (Ross, 2004). For example, the possible outcomes of a coin toss are heads or tails, which are two discrete values (Geoffrey & Stizaker, 2001).

One of the most common examples of a discrete probability distribution is the binomial distribution, which is used to model the probability of a certain number of successes in a fixed number of independent trials. For example, the binomial distribution could be used to model the probability of flipping heads ten times in a row with a fair coin (Agresti & Finlay, 2008).

Another important example of a discrete probability distribution is the Poisson distribution, which is used to model the probability of a certain number of events occurring in a fixed time period. This distribution is often used to model the probability of a certain number of customers arriving at a store within a given hour, for example (Ross, 2004).

In addition to the binomial and Poisson distributions, there are many other examples of discrete probability distributions, including the geometric distribution, the negative binomial distribution, and the hypergeometric distribution. Each of these distributions has its own unique characteristics and is used in different situations to model the probability of different outcomes.

Discrete probability distributions are an important concept in probability theory that allow us to mathematically model the probability of different outcomes of a random experiment. By understanding these distributions and how they work, we can make more informed decisions based on the likelihood of different events occurring.

2.5.2 Continuous Probability Distributions

Continuous probability distributions are a fundamental concept in probability theory, which is the mathematical study of random phenomena. These distributions are used to model the probability of different outcomes of a random experiment, such as the probability of a person's height falling within a certain range or the probability of a certain temperature being measured.

Unlike discrete probability distributions, which can only take on a countable number of values, continuous probability distributions can take on any value within a certain range. For example, the possible outcomes of measuring a person's height are continuous, since a person's height can be any value within a certain range (e.g. from 0 to 2 meters). The normal distribution, also known as the Gaussian distribution, is symmetrical and bell-

shaped, and is often used to model the probability of a continuous variable, such as a person's height or IQ score. Another important example of a continuous probability distribution is the exponential distribution, which is used to model the probability of the time between events occurring in a certain process. This distribution is often used to model the probability of the time between arrivals at a store or the time between failures in a machine.

In addition to the normal and exponential distributions, there are many other examples of continuous probability distributions, including the gamma distribution, the beta distribution, and the chi-squared distribution. Each of these distributions has its own unique characteristics and is used in different situations to model the probability of different outcomes.

Continuous probability distributions are an important concept in probability theory that allow us to mathematically model the probability of different outcomes of a random experiment. By understanding these distributions and how they work, we can make more informed decisions based on the likelihood of different events occurring.

2.5.3 Stochastic Process

A stochastic process is a mathematical model used to describe the behavior of a system or process that is subject to random variables. It is a collection of random variables that evolve over time according to certain rules. Stochastic processes are commonly used in many fields, including finance, physics, biology, and engineering. They are often used to model the behavior of complex systems that are subject to random influences, such as the stock market, weather patterns, or biological populations.

It is formally defined to be a collection of random variables defined on a common probability space (ω, \mathcal{F}, P) and indexed by the elements of a parameter set T . The set T will in this book generally be one of these (Lamperti, 1997):

$$\mathbb{R}^1, \mathbb{R}^+ = [0, \infty), Z = \{\dots, -1, 0, 1, 2, \dots\}, \text{ or } Z^+ = \{0, 1, 2, \dots\}$$

There are several key characteristics of stochastic processes. First, they are defined over a period of time, which could be discrete (such as a series of time steps) or continuous (such as a range of values). Second, they are probabilistic, meaning that their behavior is described in terms of probabilities. Third, they are often described using a set of mathematical equations or rules that define how the random variables evolve over time.

Stochastic processes are a powerful tool for modeling and analyzing complex systems that are subject to random influences. They provide a way to understand the underlying mechanisms that drive the behavior of these systems, and to make predictions about their future behavior.

2.6 Gaussian Processes

A Gaussian process is a mathematical concept used to model the behavior of random processes. It is a type of stochastic process, which is a mathematical way of describing the evolution of a random variable over time. A Gaussian process is completely specified by its mean function and covariance function. The mean function determines the expected value of the random process at any point in time, while the covariance function determines the relationship between the values of the random process at different points in time.

One of the key properties of a Gaussian process is that it is a "zero mean" process, which means that the expected value of the process at any point in time is zero. This property makes Gaussian processes well-suited for modeling noise and uncertainty in data, since they can capture the random variation present in the data without introducing any bias. Another important property of a Gaussian process is that it is a "stationary" process, which means that its statistical properties do not change over time. This property makes Gaussian processes useful for modeling the behavior of systems that exhibit a certain degree of regularity or predictability.

In addition to these properties, Gaussian processes have several other important characteristics that make them useful for modeling a wide range of phenomena. For example, they are "smooth" processes, which means that the values of the process change gradually over time. They are also "Markovian" processes, which means that the future evolution of the process is determined only by its current state, not by its past history.

Gaussian processes have many applications in various fields, including machine learning, signal processing, and finance. They are often used as a nonparametric approach to modeling data, since they do not require the specification of any particular functional form. Instead, they can be used to model complex patterns in data without making any strong assumptions about the underlying structure of the data.

Overall, Gaussian processes are a powerful and versatile tool for modeling the behavior of random processes. Their ability to capture uncertainty and regularity in data makes them a valuable tool for making predictions and decisions based on uncertain information.

2.6.1 Gaussian Process in Machine Learning

In machine learning, Gaussian processes (Rasmussen & Williams, Gaussian Processes for Machine Learning, 2006) are a powerful and widely-used tool for modeling complex patterns in data without making strong assumptions about the underlying structure of the data. Gaussian processes are a nonparametric approach to modeling data, which means that they do not require the specification of any particular functional form. Instead, they can be used to model complex patterns in data without making any strong assumptions about the underlying structure of the data.

One of the main uses of Gaussian processes in machine learning is in regression (Rasmussen & Nickisch, 2008), which is the task of estimating the relationship between a dependent variable and one or more independent variables. Gaussian processes can be used to model the underlying function that relates the dependent and independent variables, allowing for the estimation of the dependent variable at any point in the domain of the independent variable. This can be useful for making predictions about the value of the dependent variable based on new values of the independent variable.

Another important use of Gaussian processes in machine learning is in classification (Rasmussen & Nickisch, 2008), which is the task of assigning a class label to a given input. Gaussian processes can be used to model the relationship between the input and the class labels, allowing for the assignment of a class label to a new input based on its similarity to the training data. This can be useful for making predictions about the class labels of new data points. In addition to regression and classification, Gaussian processes have many other applications in machine learning, including density estimation (Rasmussen & Nickisch, 2008), clustering (Rasmussen & Nickisch, 2008), and anomaly detection. Neural networks, specifically deep learning architectures, have been widely recognized as a powerful tool for modeling complex patterns in data. They have been successfully applied in various fields such as natural language processing (NLP) (Graves, 2013), computer vision (Ng & Katanforoosh, 2015), and time series analysis \cite{sharma2017deep}.

One of the key advantages of using Gaussian processes in machine learning is their ability to optimize the values of hyperparameters \cite{Snoek2012}, which are parameters that control the behavior of the model. Hyperparameters are an important part of many machine learning algorithms, and the values of these parameters can have a significant impact on the performance of the model. In order to achieve good performance, it is often necessary to carefully tune the values of the hyperparameters to match the characteristics of the data.

By using Gaussian processes to optimize the values of hyperparameters, machine learning algorithms can achieve improved performance on a wide range of tasks. This is because Gaussian processes are able to capture the complex patterns and relationships present in the data, allowing the model to learn the optimal values of the hyperparameters that best fit the characteristics of the data. The ability of Gaussian processes to optimize the values of hyperparameters is a key advantage of using these processes in machine learning. By using a Bayesian optimization approach, Gaussian processes can help machine learning algorithms to achieve improved performance on a wide range of tasks.

Multiple Imputation by Chained Equations

Multiple imputation by chained equations is a statistical method used to handle missing data in a dataset. This method involves creating multiple versions of the dataset, each with imputed (i.e. estimated) values for the missing data, and then combining the results from these multiple versions to produce a single set of inferences.

Dealing with the missing data problem often leads to two general approaches for imputing multivariate data: joint modeling (JM) and fully conditional specification (FCS), also known as MICE (Van & Groothuis-Oudshoorn, 2011). It is known that is a JM type problem when we must specify a multivariate distribution for the missing data and obtain the imputation of its conditional distributions through Markov Monte Carlo chains (MCMC) techniques. On the other hand, it is FCS, which specifies the multivariate imputation model on a variable-by-variable basis using a set of conditional densities, one for each incomplete variable. The imputation starts by iterating over the conditional densities, usually, a low number of iterations is enough. In order to explain the model let's use the following notation \cite{van}: Let Y_j with $(j = 1, \dots, p)$ be one of p incomplete variables and $Y = (Y_1, \dots, Y_p)$. The observed and missing parts of Y_j are denoted by Y_j^{obs} and Y_j^{mis} , respectively, then $Y_j^{obs} = (Y_1^{obs}, \dots, Y_p^{obs})$, and $Y_j^{mis} = (Y_1^{mis}, \dots, Y_p^{mis})$, these are the observed and missing data respectively in Y . The number of imputation is $m \geq 1$. The h -th imputed data set is denoted as $Y^{(h)}$ where $h = 1, \dots, m$. Now let $Y_{-j} = (Y_1, \dots, Y_{j-1}, Y_{j+1}, \dots, Y_p)$ denote the collection of the $p-1$ variables in Y except Y_j . Finally, let Q denote the quantity of scientific interest. Mice algorithm has three main steps: imputation, analysis, and pooling. The analysis starts in an incomplete data set Y_{obs} . The second step is to compute Q on each imputed data set, here the model is applied to $Y^{(1)}, \dots, Y^{(m)}$ in the general identical. Finally, in the three step is to pool the m estimates $\hat{Q}^{(1)}, \dots, \hat{Q}^{(m)}$ into one estimate \bar{Q} and estimate its variance.

The resulting imputed datasets are then combined to produce a single, more complete dataset that is used for analysis. This approach can help to reduce bias and improve the accuracy of the results, compared to other methods for dealing with missing data. MICE is often used in fields such as sociology, economics, and public health, where data collection is complex and incomplete data is common.

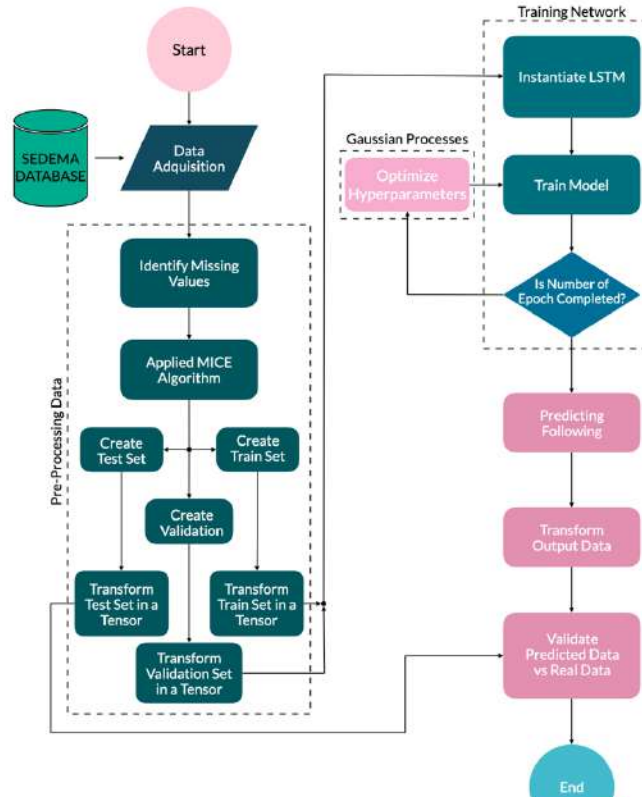


Figure 1: Implemented Workflow

3. Methodology

This section details the step by step from the acquisition of the data to the results of the optimized model. This follows the flowchart shown in the Figure 1: *Implemented Workflow* Figure 1. Additionally, the specifications of the equipment used are detailed, as well as the recommended requirements.

3.1 Requirements

3.1.1 System Requirements

To generate, train and implement the model, it is necessary to satisfy the next following requirements in order to create a proper environment:

- The computer must have Intel Core I5 (11th generation) or higher, M1 processor or higher.
- The computer must have installed whether Ubuntu 16.04 or later, MacOS Mojave or later, or Windows 7 or later as operative System.
- The computer must have installed Python 3.8 or later.
- The computer must have 8GB of RAM.

3.1.2 Python Libraries

This project was developed in the programming language Python 3.8 using the next libraries:

- Tensorflow.
- Pytorch.
- Sklearn.
- Numpy.
- Pandas.
- Matplotlib.
- Seaborn.

3.2 Database Acquisition

For PM_{10} and $PM_{2.5}$ the dataset used were provide by the Atmospheric Monitoring System (SIMAT for its acronym in spanish) SEDEMA, and from SIMAT is obtained the data of Automatic Environmental Monitoring System (RAMA, for its acronym in Spanish) which is special because it contains the levels of PM in different monitoring stations in the City of Mexico.

The SIMAT counts with more than 40 stations (SEDEMA, 2022) to measure the quality of the air in the metropolitan area, Ciudad de México (CDMX) and surrounding area of Estado de México. Also, at the time of completion of this project just the stations show in the Table 1 has the capability of measure $PM_{2.5}$.

Table 1: List of names and abbreviations of the stations used

| Key | Name Station | Location |
|-----|----------------------------|------------------|
| AJM | Ajusco Medio | CDMX |
| BJU | Benito Juárez | CDMX |
| FAR | FES Aragón | Estado de México |
| GAM | Gustavo A. Madero | CDMX |
| HGM | Hospital General de México | CDMX |
| INN | Investigaciones Nucleares | Estado de México |
| MER | Merced | CDMX |

| | | |
|-----|-----------------------|------------------|
| MGH | Miguel Hidalgo | CDMX |
| MON | Montecillo | Estado de México |
| NEZ | Nezahualcóyotl | Estado de México |
| PED | Pedregal | CDMX |
| SAC | Santiago Acahualtepec | CDMX |
| SAG | San Agustín | Estado de México |
| SFE | Santa Fe | CDMX |
| TLA | Tlanepantla | Estado de México |
| UAX | UAM Xochimilco | CDMX |
| UIZ | UAM Iztapalapa | CDMX |
| XAL | Xalostoc | Estado de México |

The sites used in this work are the following: Northeast (Gustavo A. Madero - GAM, FES Aragón - FAR, Xalostoc - XAL), Northwest (Tlanepantla - TLA,), Center (Hospital General de México - HGM, Merced - MER), Southeast (Nezahualcóyotl - NEZ, Santiago Acahualtepec - SAC) and Southwest (Ajusco Medio - AJM, Pedregal - PED, Santa Fe, SFE). The map of the monitoring sites is shown in Figure 2.

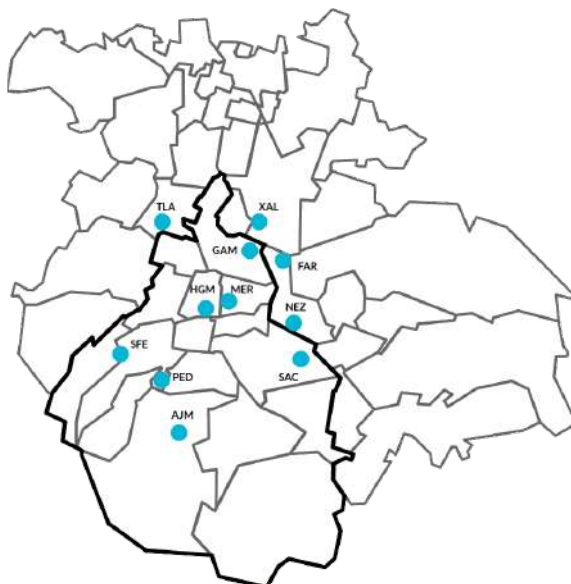


Figure 2: Location of the SIMAT stations used for this project across the metropolitan area (*Gobierno de la Ciudad de México, 2023*).

3.3 Preprocessing

Since the data which the model receive must be in a tensor form and with non-missing values, it is required to look up into the data.

3.3.1 Missing Values

This is the first step in the preprocessing of the data. Data were used to create the dataset has a total amount of 8760 values. Since each station has different percentages of missing values in them only datasets with more than 70% of non-missing values where select to be imputed and used them to train the models, leaving us only with the stations reported in Table 2 The Figure 3 shows the missing values found across the entire data base. The reason for classifying the data from the stations in this way is because the greater the number of data to impute, the dataset will move further and further from reality. Missing values appear due to sensor failure which reports values of -99 or Nan.

Table 2: Percentage of Missing Values found in each station

| Station | Percentage of Missing Values |
|---------|------------------------------|
| AJM | 7.42% |
| BJU | 57.82% |
| FAR | 25.091 |
| GAM | 14.76% |
| HGM | 19.132% |
| INN | 62.295% |
| MER | 9.422% |
| MGH | 62.626% |
| MON | 15.445% |
| NEZ | 16.153 |
| PED | 14.349% |
| SAC | 28.333% |
| SAG | 67.26% |
| SFE | 19.874% |
| TLA | 21.804% |
| UAX | 36.233% |
| UIZ | 29.338 |
| XAL | 16.062% |

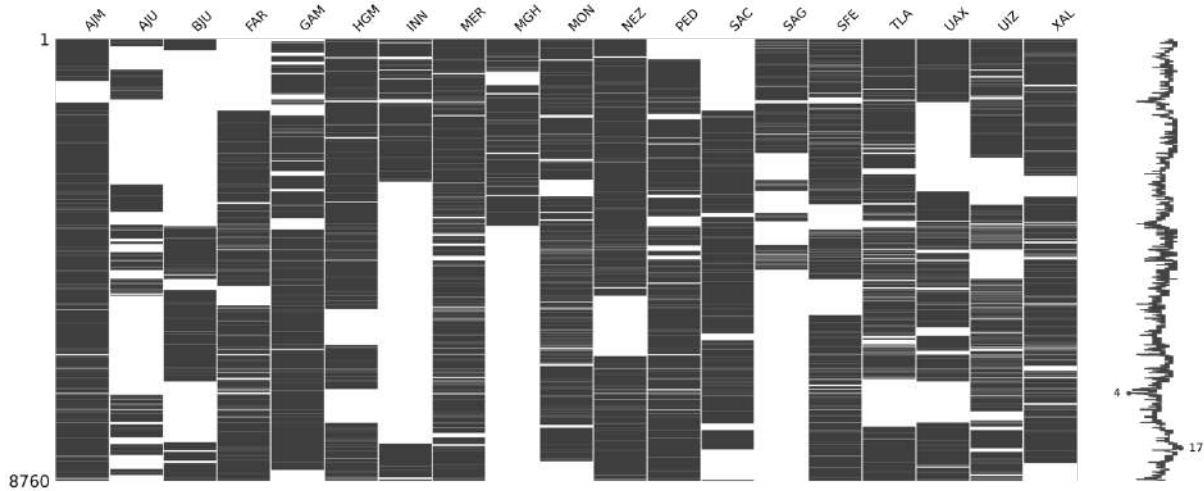
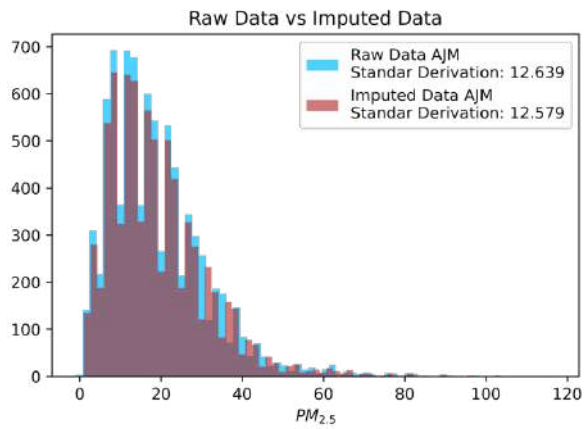
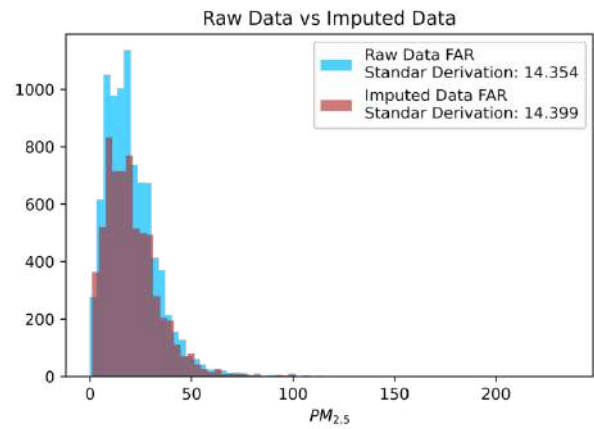


Figure 3: Missing values across the database.

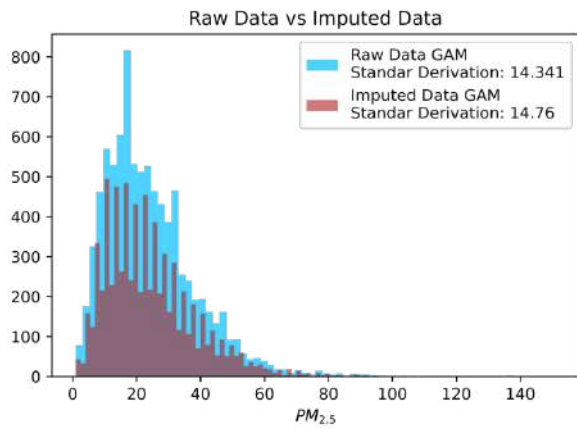
After the imputation, the goal of imputation is to create a complete dataset that can be used for analysis, modeling, and other purposes. It is important that the statistical properties of the data do not change after imputation because we want to maintain the same distributional properties and relationships between variables that were present in the original dataset. If the statistical properties of the data change after imputation, then any analysis or modeling done on the imputed dataset may be biased or incorrect. To verify the statistic variations where not affect the dataset selected, the standard derivation where check for each dataset. This is show in figure 4.



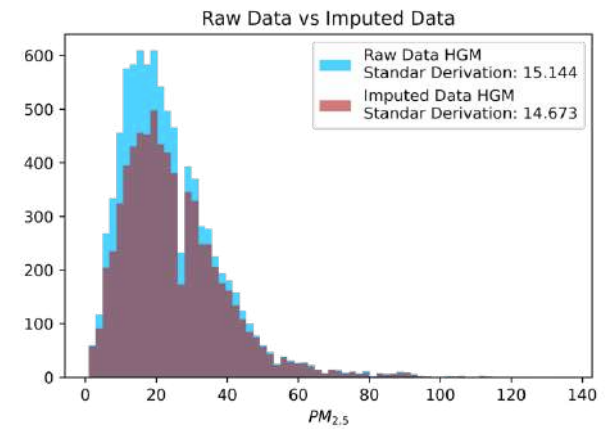
(b) AJM station



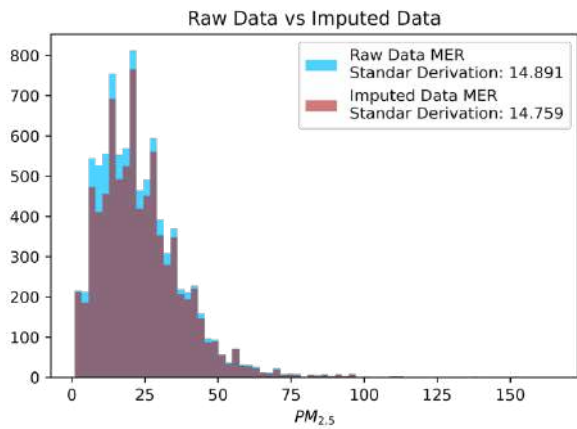
(a) FAR station



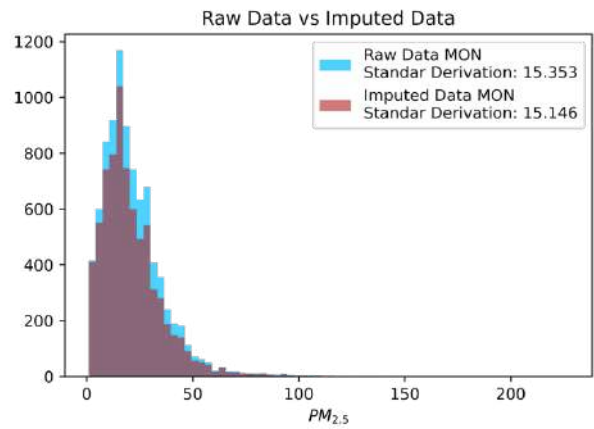
(c) FAR station



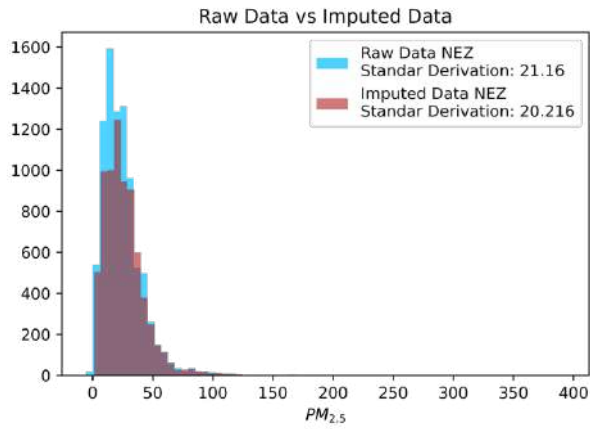
(d) HGM station



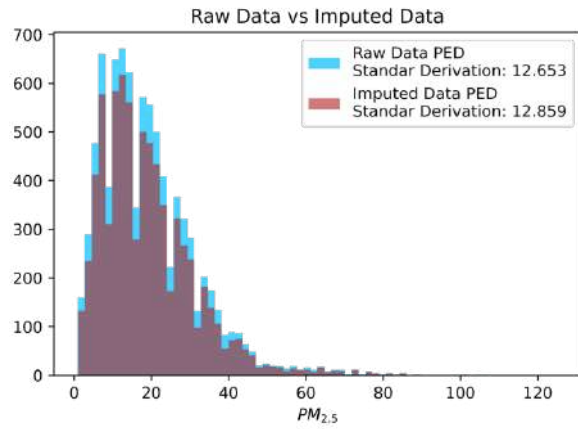
(e) MER station



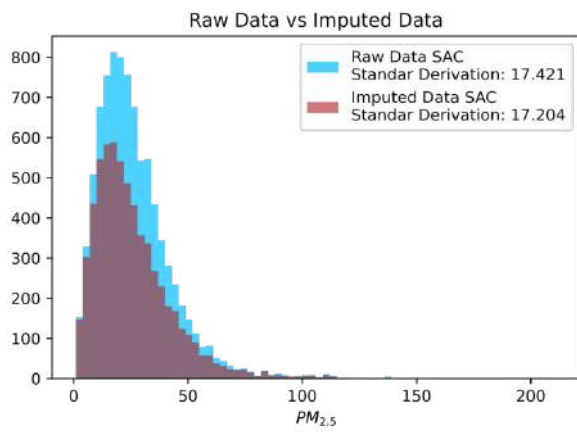
(d) MON station



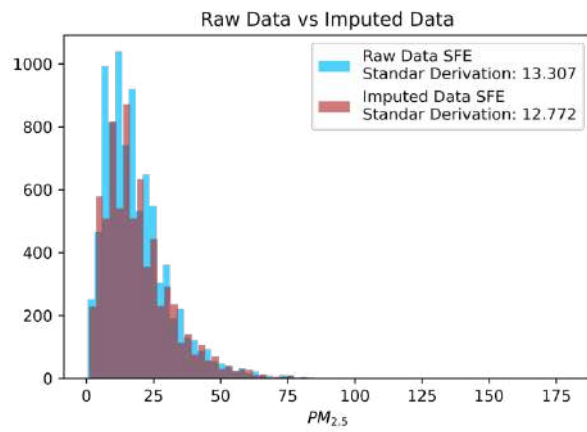
(f) NEZ station



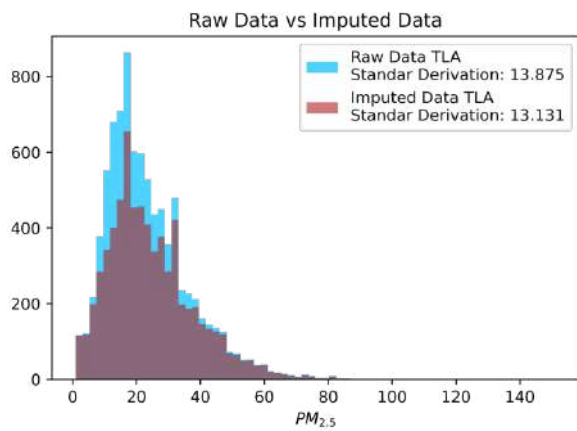
(g) PED station



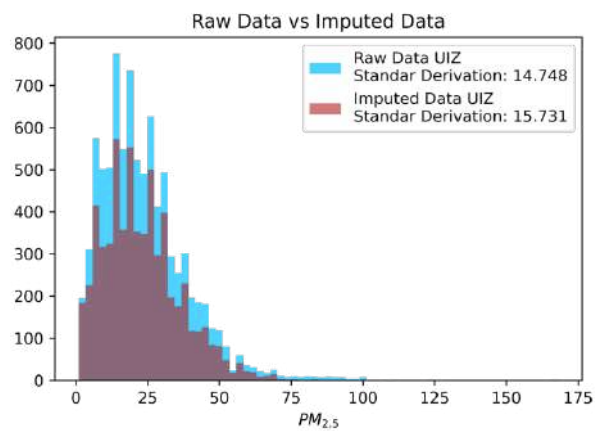
(h) SAC station



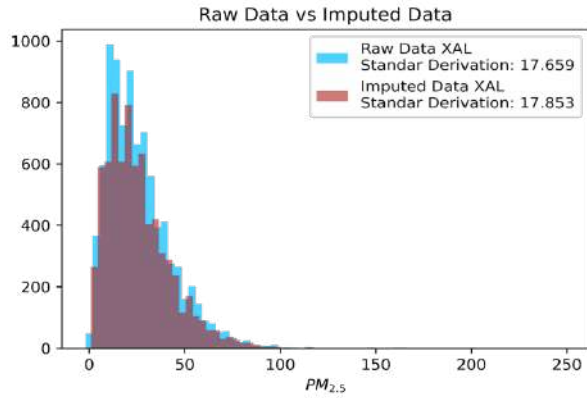
(i) SFE station



(j) TLA station



(k) UIZ station



(1) XAL station

Figure 4: Standard derivation across the dataset selcted before and after being imputed.

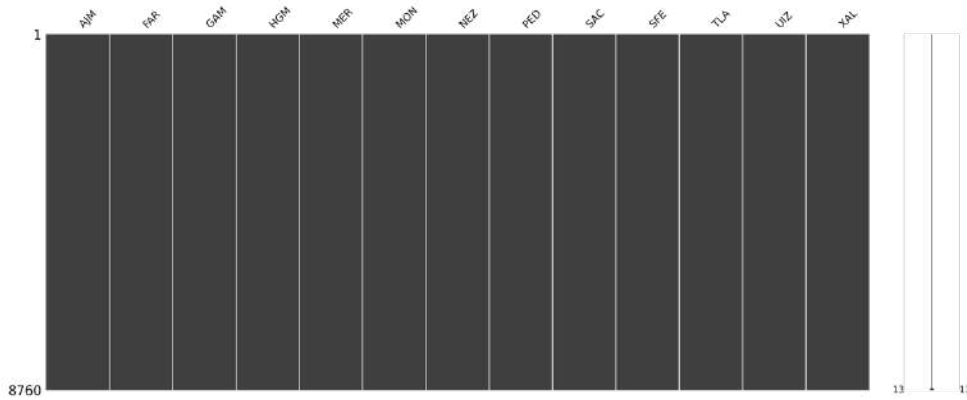


Figure 5: Result of imputed stations.

In figure 5 is shown how the dataset were fill and has not missing values now. Continuing with the stats of the data, in the table is shown the data before and after being imputed. The objective is to show that the imputation does not change the statistical properties of the data, this is important because you don't want to create a sequence with no relation with the original.

Table 3: Change in the statistics of the data before and after being imputed.

| Station | Mean Before Imputation | Mean After Imputation | Standard Deviation Before Imputation | Standard Deviation After Imputation |
|---------|------------------------|-----------------------|--------------------------------------|-------------------------------------|
| AJM | 19.322 | 19.154 | 12.638 | 12.579 |
| FAR | 20.890 | 20.737 | 14.340 | 14.298 |
| GAM | 24.102 | 24.293 | 14.340 | 14.760 |
| HGM | 25.081 | 24.026 | 15.144 | 14.673 |
| MER | 23.939 | 23.558 | 14.891 | 14.758 |
| MON | 21.336 | 21.876 | 15.352 | 15.758 |
| NEZ | 26.726 | 24.931 | 21.726 | 10.216 |
| PED | 19.236 | 19.454 | 12.653 | 12.216 |
| SAC | 25.286 | 26.243 | 17.421 | 17.203 |
| SFE | 19.286 | 18.617 | 13.206 | 12.771 |

| | | | | |
|-----|--------|--------|--------|--------|
| TLA | 24.642 | 23.510 | 13.747 | 13.131 |
| UIZ | 23.344 | 24.066 | 14.747 | 15.730 |
| XAL | 25.918 | 26.255 | 17.659 | 17.853 |

4. Results and Discussion

To determine which methodology is better when these techniques are implemented in the LSTM network, we train 25 times each model and compare the results of its training. For the baseline model, the simple LSTM were used 64 units, a learning rate of 0.001, the activation function was ReLu, using 10 epochs, and a batch size of 64. Instead, for the models which use the gaussian process optimization the space of search was: 64 baseline units + 64 extra units, a variation in the learning rate from 0.001 to 0.02 in logarithm increase, ReLu as the activation function and 10 baseline epochs +20, and finally a batch size from 20 to 128.

Table 4: Hyperparameters of the baseline model and the GP model.

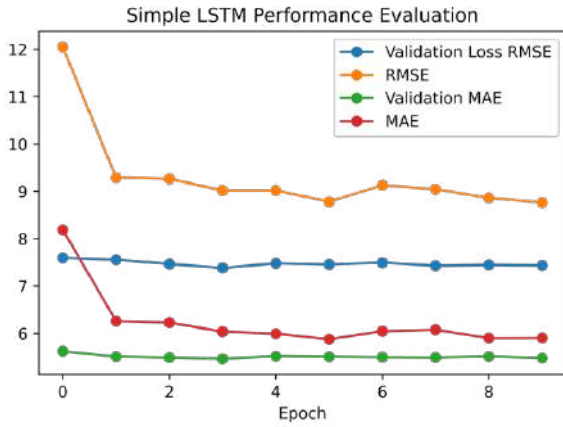
| Model | Units | Learning Rate | Activation Function | Number of Epochs | Batch Size |
|----------------|-------|---------------|---------------------|------------------|------------|
| Baseline Model | 64 | 0.001 | ReLu | 20 | 64 |
| Best GP Model | 82 | 0.001044133 | ReLu | 10 | 220 |

4.1 LSTM Baseline Model

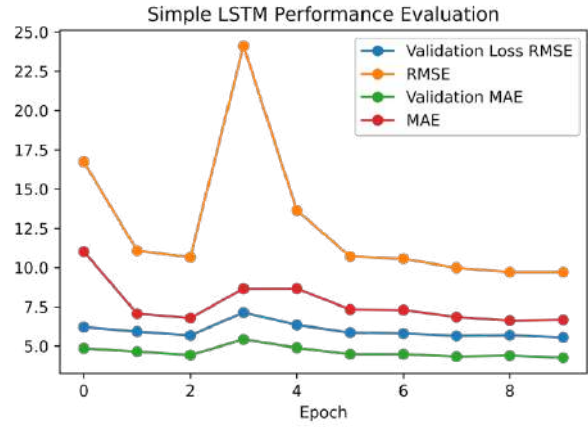
This model was trained according to the methodology presented. The model, in general, gives good results but not always. For the case of AJM station this model had a fast convergence in each epoch. The $PM_{2.5}$ forecast for the validation dataset is good enough as is expected in these cases for a LSTM but the RMSE and MAE are still to be considerable. Another good reason point for this model is the time that requires to execute the model with an average of 45 seconds.

As it was mentioned, the forecast is good enough but not always, this because with no optimization in the processes there are not control qualities for the results and it can be too different in every try due the own nature of the deep learning mechanisms.

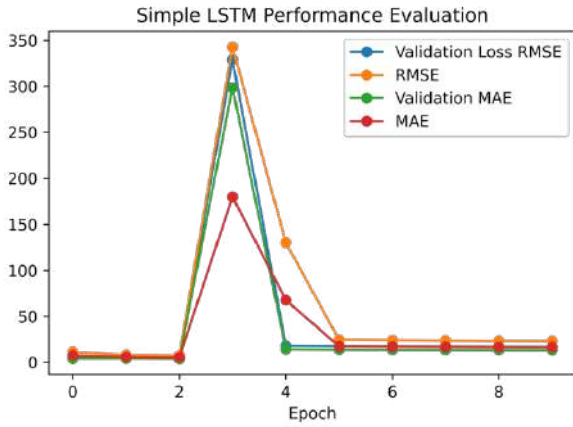
The model was executed 25 times in order to obtain general statistics about the performance of the model. The average of this results is shown in the table 5 and discussed in the chapter 4.3. As in the figure 6 is shown, the results of the model is good enough, most of the time the model converge 5 epochs, and there is not a big difference in the performance of the model for the different stations.



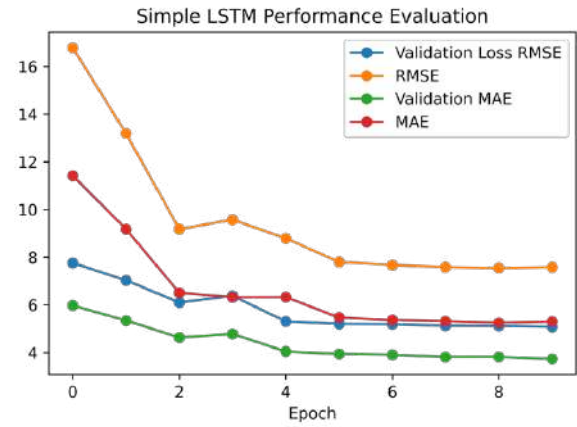
(a) AJM station baseline model metrics.



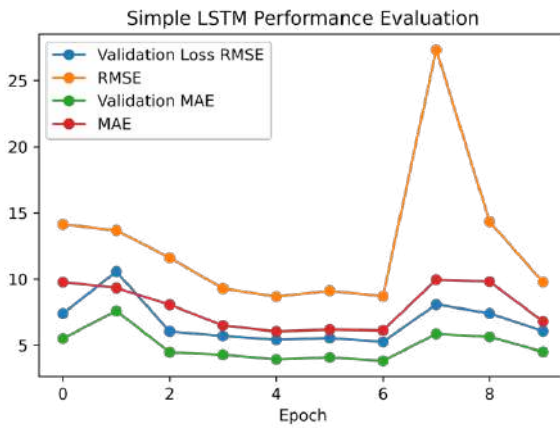
(b) FAR station baseline model metrics.



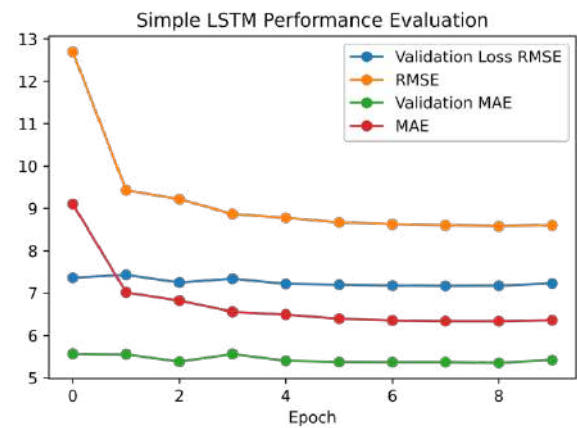
(c) MON station baseline model metrics.



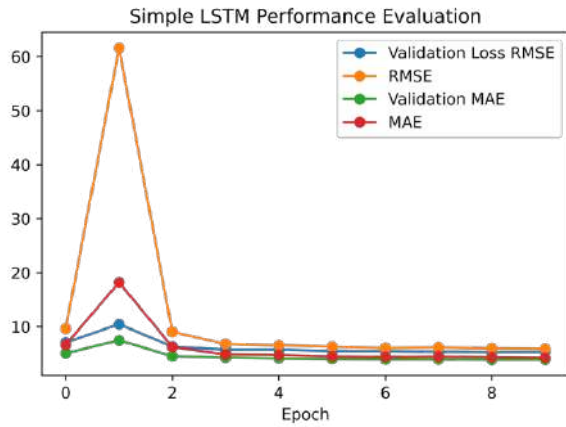
(d) NEZ station baseline model metrics.



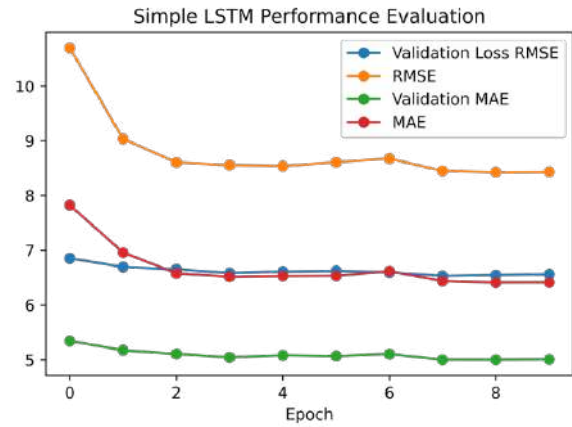
(e) PED station baseline model metrics.



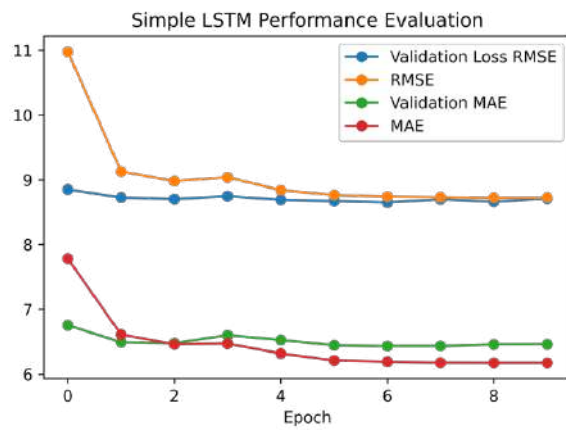
(f) SAC station baseline model metrics.



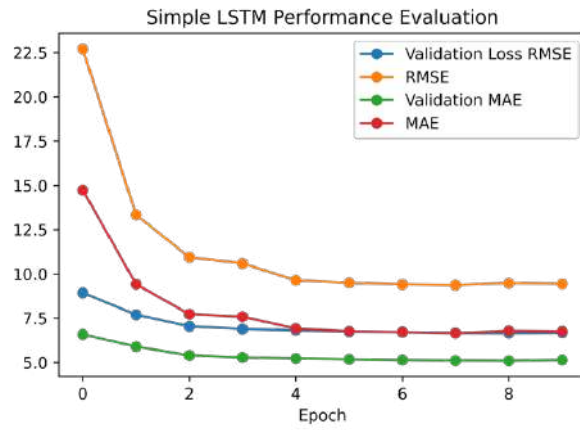
(g) SFE station baseline model metrics.



(h) TLA station baseline model metrics.

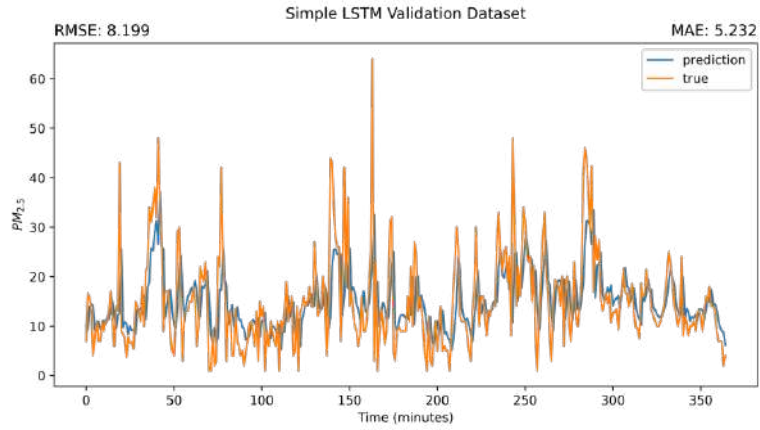


(i) UIZ station baseline model metrics.

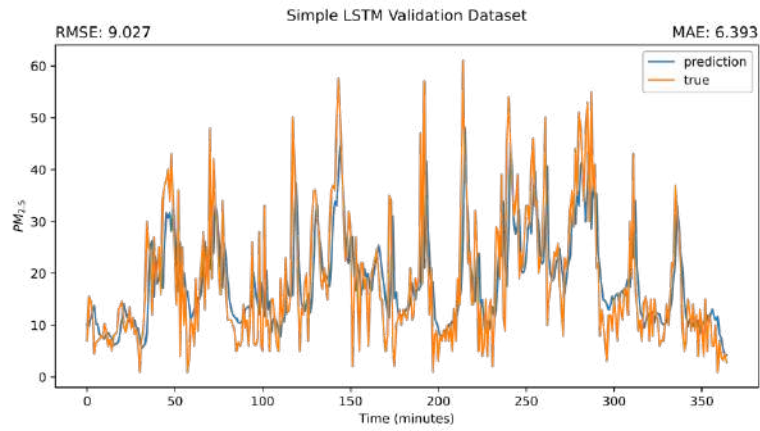


(j) XAL station baseline model metrics.

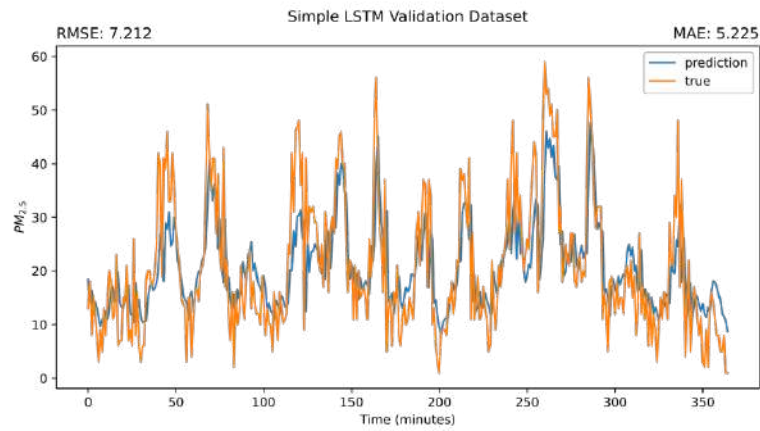
Figure 6: Metrics of baseline model in each station.



(a) AJM Baseline model forecast.



(b) UIZ Baseline model forecast.



(c) TLA Baseline model forecast.

Figure 7: Forecast performance of the baseline model.

4.2 LSTM Model Optimized with GP

The optimized models with GP present a more consistent behavior, without significant variations between the results of the different stations. Although a point that should be considered is the average execution time, this being 4 times longer than the baseline model.

Gaussian processes (GPs) have been gaining popularity in recent years as a powerful tool for optimization and uncertainty estimation in machine learning. One of the areas where GPs have been particularly successful is in the optimization of recurrent neural networks (RNNs) such as long short-term memory networks (LSTMs). In this article, we will discuss the advantages of using GPs for optimizing LSTMs.

One of the main benefits of using GPs for optimizing LSTMs is that they provide a probabilistic interpretation of the optimization problem. This allows for the estimation of uncertainty in the model parameters, which is useful for determining the confidence in the model's predictions and identifying regions of the parameter space where the optimization is particularly uncertain. This can be particularly useful in applications where the model's predictions need to be robust and reliable, such as in healthcare or finance.

Another advantage of GPs is that they can handle non-convex optimization problems, which are common in RNNs due to the complex, non-linear dynamics of the underlying system. This means that GPs can find global optima, rather than getting trapped in local optima like traditional optimization methods. This can be beneficial when the objective function of the optimization problem is complex and has multiple local optima.

GPs are also computationally efficient, even for large datasets, making them well-suited for RNNs with many parameters. They also scale well with the number of data points, which is important for RNNs that are trained on large datasets. This can be beneficial when the data is large and it's not possible to use other optimization techniques.

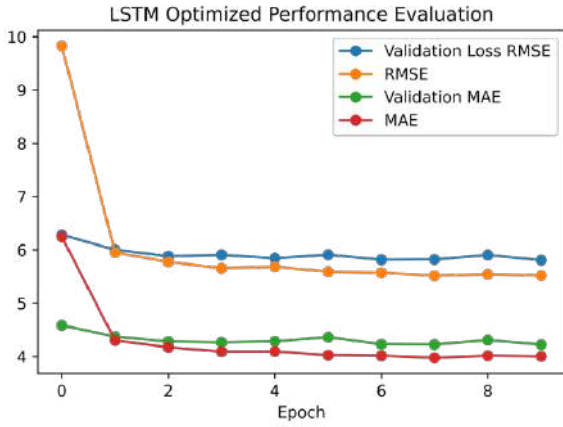
In addition, GPs can incorporate prior knowledge about the structure of the problem, such as smoothness or periodicity, into the optimization process. This can help to constrain the search space and improve the optimization results. This can be particularly useful in applications where the structure of the problem is known, such as in time-series analysis.

Furthermore, GPs can handle missing or corrupted data, which is a common problem in many RNN applications. This can be beneficial in situations where the data is incomplete or corrupted, and traditional optimization methods would fail.

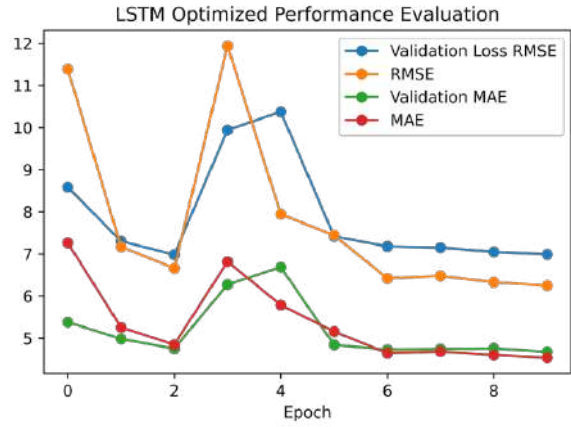
Another advantage of GPs is that they are more robust than traditional optimization methods when it comes to the presence of noise in the data. This can be beneficial in situations where the data is noisy, and traditional optimization methods would fail.

Finally, GPs can be integrated with other machine learning techniques such as Bayesian optimization to enhance the optimization process. This can be beneficial when the optimization problem is complex and needs to be solved using multiple techniques.

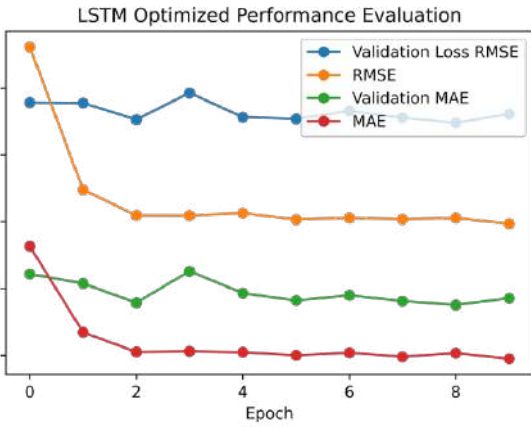
In conclusion, GPs provide several advantages over traditional optimization methods when it comes to optimizing LSTMs. They provide a probabilistic interpretation of the optimization problem, handle non-convex optimization problems, are computationally efficient, incorporate prior knowledge, handle missing or corrupted data, are robust to noise and can be integrated with other machine learning techniques. These advantages make GPs a valuable tool for optimizing LSTMs in various applications.



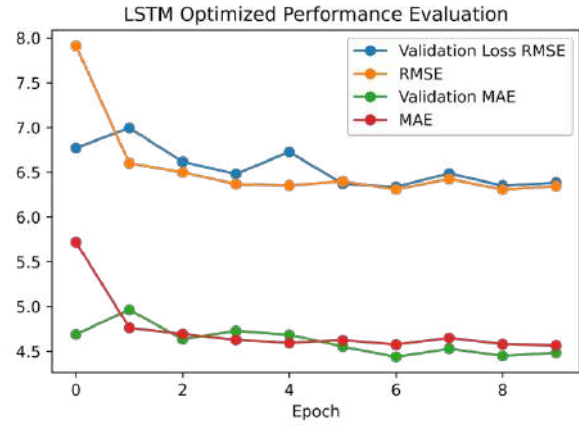
(a) AJM GP Optimized model.



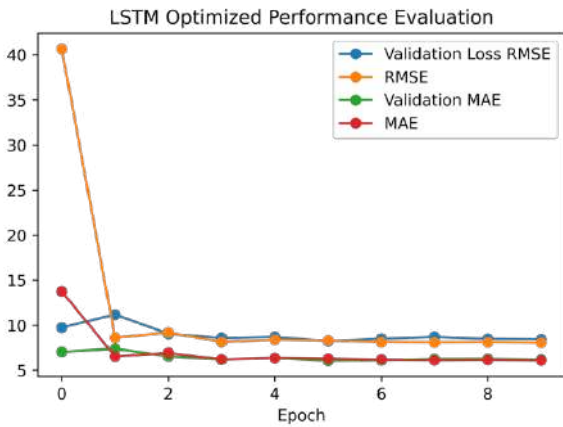
(b) FAR GP Optimized model.



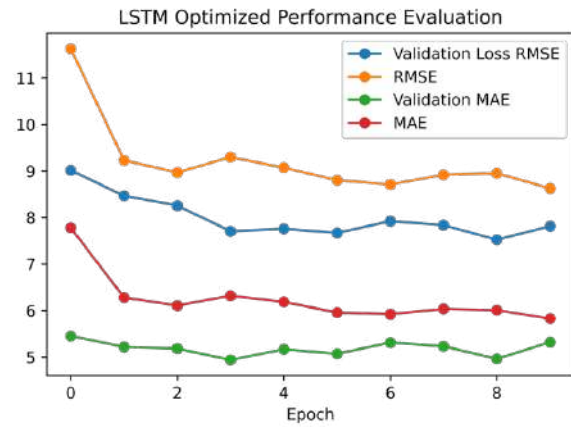
(c) GAM GP Optimized model.



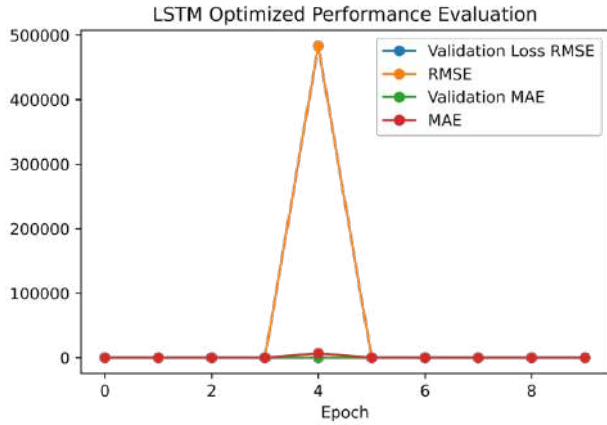
(d) HGM GP Optimized model.



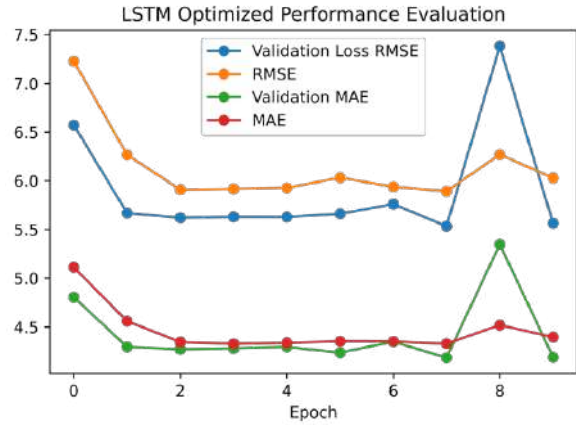
(e) MER GP Optimized model.



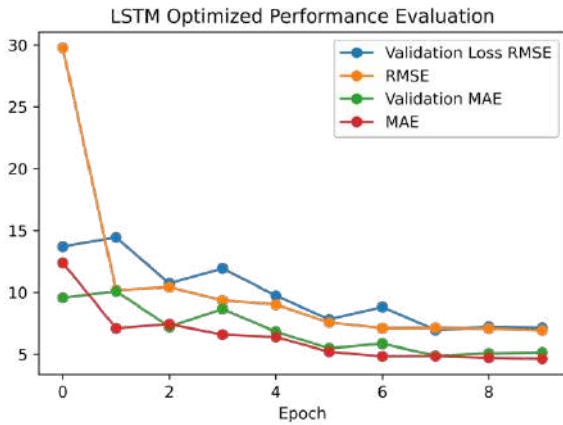
(f) MON GP Optimized model.



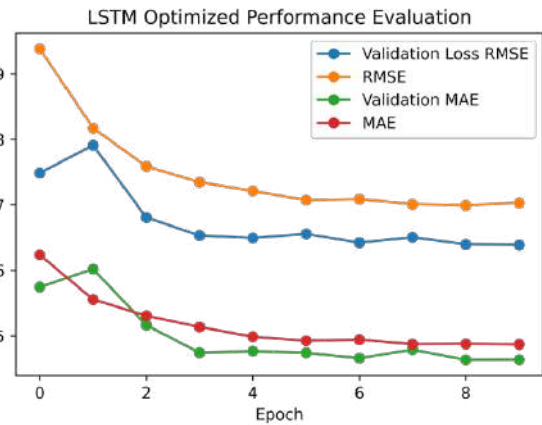
(g) MON GP Optimized model.



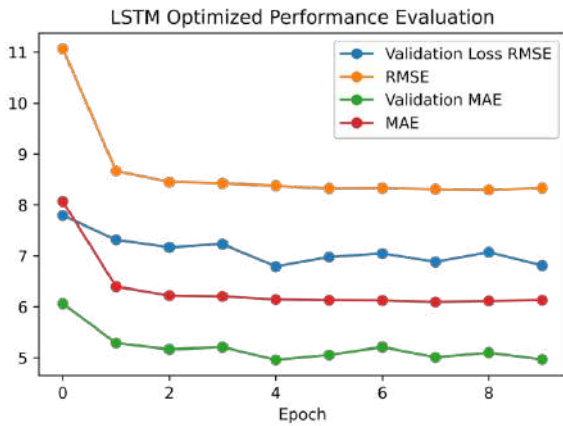
(h) MON GP Optimized model.



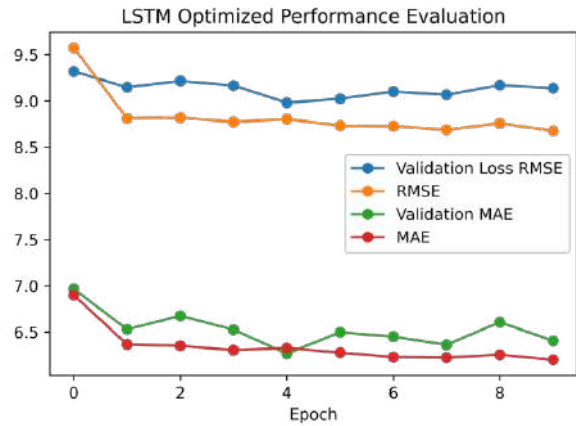
(i) SAC GP Optimized model.



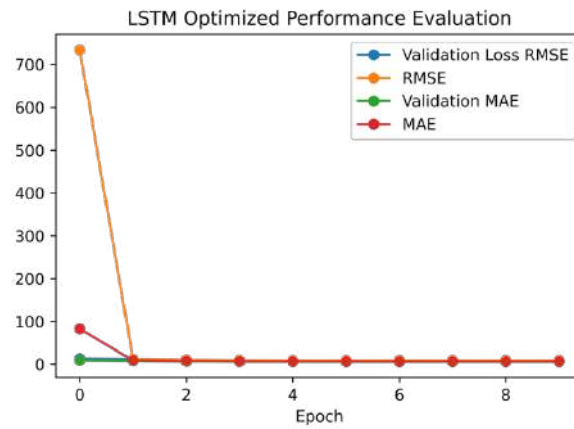
(j) SFE GP Optimized model.



(k) TLA GP Optimized model.

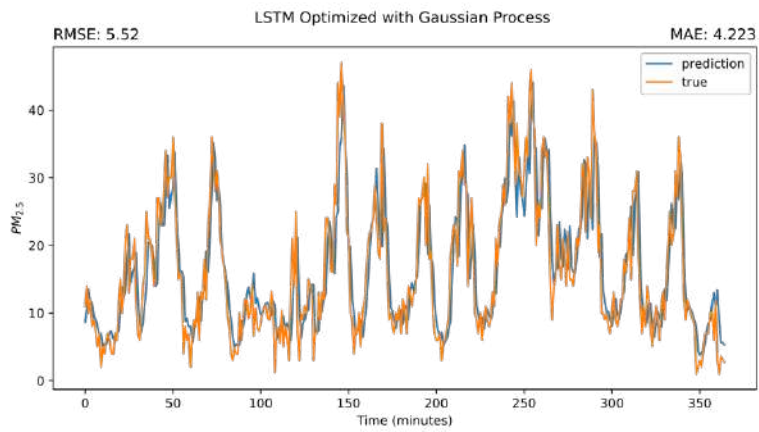


(l) UIZ GP Optimized model.

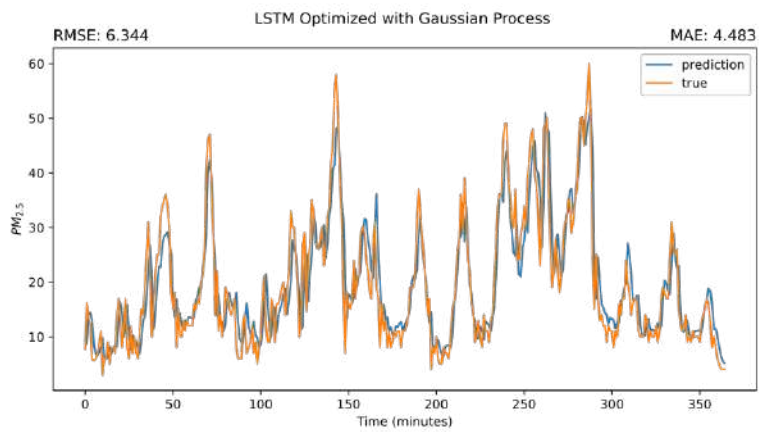


(m) XAL GP Optimized model.

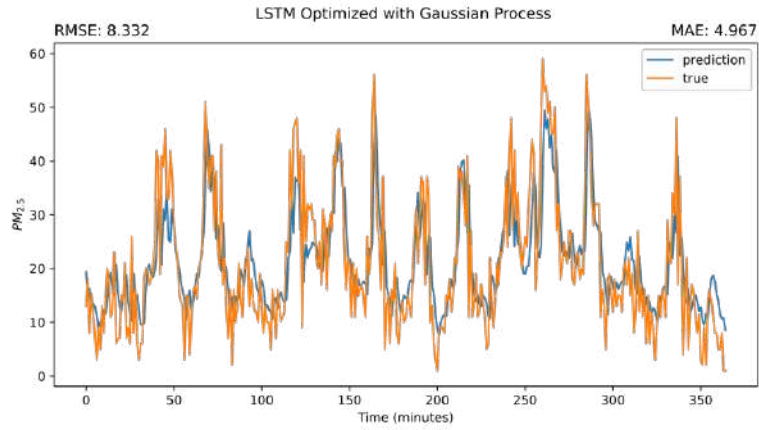
Figure 8: GP model metrics.



(a) AJM GP model forecast.



(b) GAM GP model forecast.



(c) XAL GP model forecast.

Figure 9: GP model forecast performance.

4.3 GP Model vs Baseline Model

In order to confirm that the GP model has a better performance than the baseline model, the entire process was run 25 times. The results show a clear improvement when GP is applied to the model.

Table 5: Performance of each model

| Model | Loss | MAE | MSE | Time (s) |
|---------|---------|--------|---------|----------|
| LSTM | 16.5258 | 0.6852 | 16.5258 | 1.0562 |
| LSTM+GP | 6.1341 | 0.3104 | 6.1341 | 34.1076 |

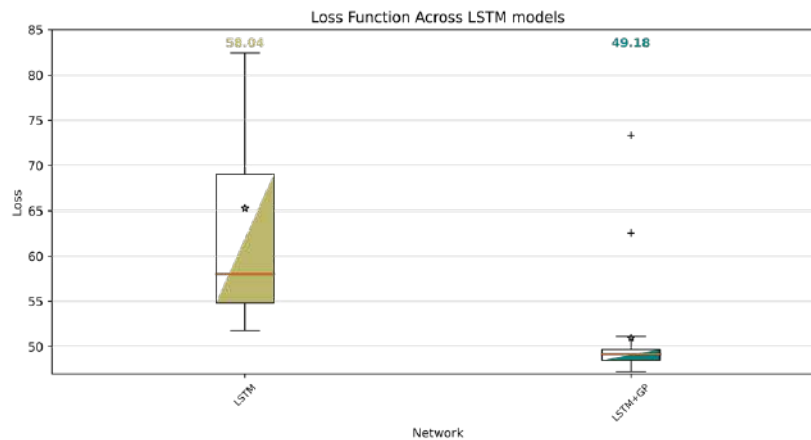


Figure 6: Comparative of the loss function between LSTM models in the train test

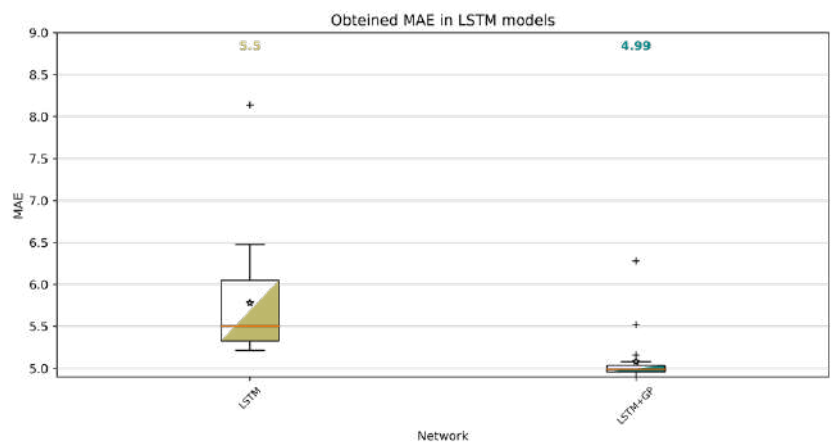


Figure 7: Comparative of the MAE between LSTM models

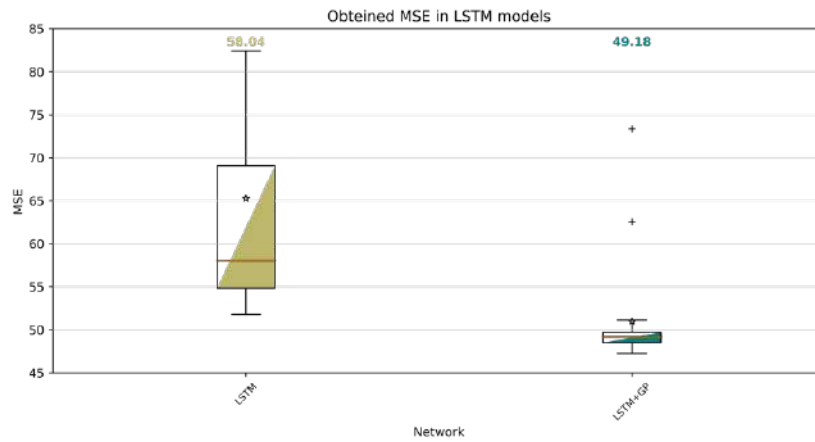


Figure 8: Comparative of the MSE between LSTM models

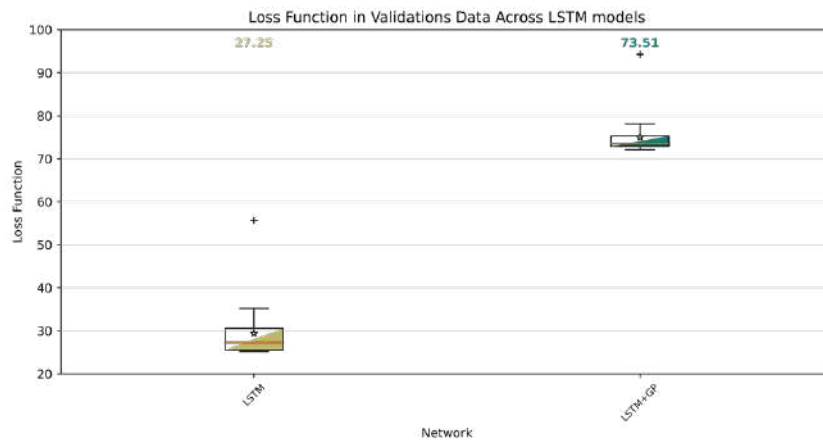


Figure 9: Comparative of the loss function between LSTM models in the validation dataset

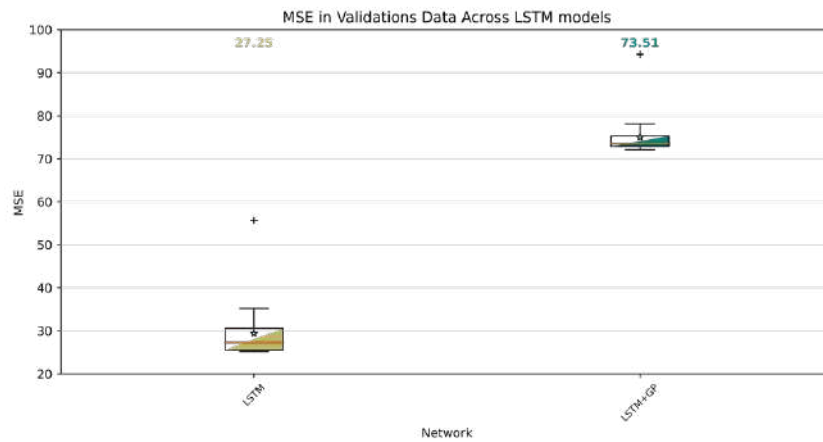


Figure 10: Comparative of the MSE between LSTM models in the validation dataset

4.4 Discussion

4.4.1 Significance/Impact

This project provides a useful tool for Mexico City and the State of Mexico, since it gives greater certainty about the analysis of air quality and if a prediction can be made beyond the dataset, it could prevent exceedances and give time to society, industry and government to take preventive measures.

The result of this project lead to publish a chapter in the IntechOpen Series Artificial Intelligence, Volume 15, on topic Swarm Intelligence Recent Advances and Current Applications.

4.4.2 Future Work

There are many points to improve and to expand the utility of the model such as the forecast of the leave of absence, the implementation of attention mechanism and the time embedding. Also to make predictions beyond the dataset.

For future work, we will seek to add service mechanisms to the LSTM network and use the time-to-vec methodology to temporal data to improve the input features of the network, seeking to have a wider temporal window.

Bibliography

- Aceves-Fernandez, M., Estrada, A., Pedraza-Ortega, J., Gorrostiera-Hurtado, E., & Tovar-Arriaga, S. (2015). Design and implementation of ant colony algorithms to enhance airborne pollution models. *International Journal of Environmental Science and Toxicology Research*, 23, 22-28.
- Agresti, A., & Finlay, B. (2008). *Statistical Methods for the Social Sciences*. Prentice Hall.
- Albert, R., & Barabasi, A. L. (2009). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 47-97.
- Amodei, D., & Olah, C. (2018). *Explainable Artificial Intelligence (XAI)*. Unknown.
- Arbex, M. A., Nacimiento Saldiva, P. H., Amador Pereira, L. A., & Ferreira Braga, A. L. (2010). Impact of outdoor biomass air pollution on hypertension hospital admissions. *Journal of Epidemiology and Community Health*, 64, 573-579.
- Becerra Rico, J., Aceves-Fernandez, M., Escalante, K., & Pedraza-Ortega, J. C. (09 de 2020). Airborne particle pollution predictive model using Gated Recurrent Unit (GRU) deep neural networks. *Earth Science Informatics*, 13, 1-14.
- Berg, A. K., Berg, T. L., & Yang, L. (2018). *Deep Learning for Computer Vision*. CRC Press.
- Brownlee, J. (2019). *Deep Learning for Time Series Forecasting*. Machine Learning Mastery.
- Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.
- Campbell, J. Y., & Shiller, R. J. (1997). *Valuation of Financial Assets: Review of Literature*. National Bureau of Economic Research.
- Charu, C. A. (2018). *Neural Networks and Deep Learning*. New York Springer.
- Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.
- Cox, D. R. (1961). *The Algebra of Probable Inference*. Johns Hopkins University Press.
- Cramer, H. (1946). *Mathematical Methods of Statistics*. Princeton University Press.
- Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). *Neural Network Design*. 2014, Stillwater, OK, USA: Martin Hagan.
- EPA. (20 de 11 de 2022). *Particle Matter (PM) Basics*. Obtenido de epa.gov: <https://www.epa.gov/pm-pollution/particulate-matter-pm-basics>
- Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons.
- Ferman-Cano, F., Padialla-Santamaria, F., Moreno-Venegas, L., & et all. (2018). *Metaplasia de vías aéreas asociada a tabaquismo y contaminación ambiental mediante esputo* (Vol. 56). Rev Med Inst Mex Seguro Soc.

- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36, 193-202.
- Gauss, C. F. (1809). *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. Perthes.
- Geoffrey, G., & Stizaker, D. (2001). *Probability and Random Processes*. OUP Oxford.
- Gibson, A., & Patterson, J. (2018). *Deep Learning: A Practitioner's Approach*. O'Reilly Media.
- Gillespie, D. (2015). *Probability: An Introduction*. John Wiley & Sons.
- Gobierno de la Ciudad de México. (2023, 01 23). *Direccion de Monitoreo Atmosférico* . Retrieved from <http://www.aire.cdmx.gob.mx/default.php>
- Goldbeg, Y. (2017). *Deep Learning for Natural Language Processing*. Synthesis Lectures on Human Language Technologies.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Graves, A. (2013). Natural Language Processing with Deep Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.
- Guo, Y., Barnett, A. G., Zhang, Y., Tong, S., Yu, W., & Pan, X. (2010). The short-term effect of air pollution on cardiovascular mortality in Tianjin, China: comparison of time series and case–crossover analyses. *Science of the Total Environment*, 409, 300-306.
- Hull, J. C. (2017). *Options, Futures and Other Derivatives*. Pearson.
- IBM. (8 de 10 de 2022). IBM. Obtenido de [ibm.com](https://www.ibm.com/mx-es/cloud/learn/machine-learning?mhsrc=ibmsearch_a&mhq=¿que%20es%20machine%20learning%3F): https://www.ibm.com/mx-es/cloud/learn/machine-learning?mhsrc=ibmsearch_a&mhq=¿que%20es%20machine%20learning%3F
- Jay L. Devore, K. N. (2012). *Modern Mathematical Statistics with Applicat.* Springer New York, NY.
- Kleinrock, L. (1975). *Queueing Systems, Volume 1: Theory*. John Wiley & Sons.
- Kolmogorov, A. (1933). *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer.
- Kuri-Monge, J. G., Aceves-Fernandez, M. A., Ramirez-Montanez, J. A., & Pedraza-Ortega, J. C. (2021). Capability of a Recurrent Deep Neural Network Optimized by Swarm Intelligence Techniques to Predict Exceedances of Airborne Pollution (PMx) in Largely Populated Areas. *International Conference on Information Technology, ICIT 2021, Amman, Jordan, July 14-15, 2021* (págs. 61-68). IEEE.
- Lamperti, J. (1997). *Stochastic Processes*. Springer New York, NY.
- Luger, G. F. (1994). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley Longman Publishing Co., Inc.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807-814.
- Negnevitsky, M. (2005). *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson Education Limited.
- Ng, A., & Katanforoosh, K. (2015). Deep Learning for Computer Vision. *arXiv preprint arXiv:1507.05909*.
- Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers.
- Poisson, S. D. (1837). Recherches sur la probabilité des jugements en matière criminelle et en matière civile, présentées à la faculté des sciences de l'Université de Paris. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 1-121.
- Poole, D., & Mackworth, A. (2017). *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press.
- Rasmussen, C. E., & Nickisch, H. (2008). Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 2035-2078.
- Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. MIT Press.
- Ross, S. M. (2004). *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier.

- Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Sanchez-Caraballo, J. M. (18 de 10 de 2022). *Características fisicoquímicas de los gases y partículas contaminantes del aire. Su impacto en el asma*. Obtenido de Scielo.org: <http://www.scielo.org.co/pdf/iat/v25n4/v25n4a07.pdf>
- SEDEMA. (18 de 09 de 2022). *Red automática de monitoreo ambiental*. Obtenido de <http://www.aire.cdmx.gob.mx>
- Siegelmann, H. T., & Sontag, E. D. (1995). On the Computational Power of Neural Nets. *Journal of Computer and System Sciences*, 50(1), 132-150.
- Singh, H. (2018). *Deep Learning for Computer Vision: A Practical Guide*. Apress.
- Sipser, M. (2013). *Introduction to the Theory of Computation*. Cengage Learning.
- Stephens, S., Madronich, S., Wu, F., Olson, J., Ramos, R., Retaman, A., & Munoz, R. (2008). Weekly Patterns of Mexico City's surface concentrations of CO, NO_x, PM₁₀ and O₃ during 1986-2007. *Atmospheric Chemistry and Physics Discussion*, 8457-8384.
- Van, B. S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 1-67.
- von Mises, R. (1957). *Probability, Statistics and Truth*. Dover Publications.
- Wilks, D. S. (2011). *Statistical Methods in the Atmospheric Sciences*. Academic Press.
- William, F. (1968). *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons.

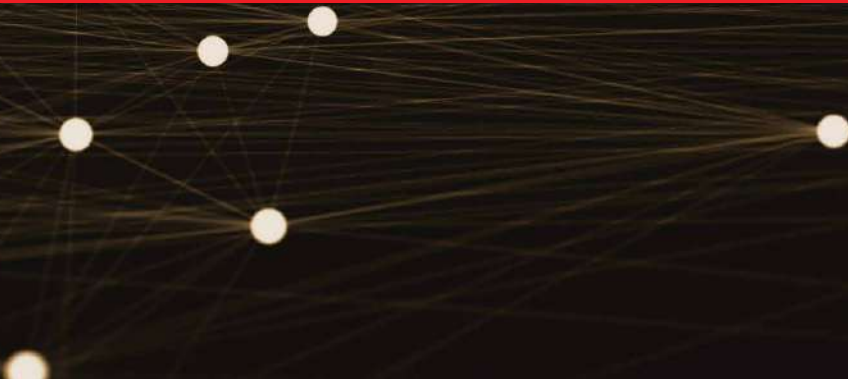
IntechOpen

IntechOpen Series
Artificial Intelligence, Volume 15

Swarm Intelligence

Recent Advances and Current Applications

Edited by Marco Antonio Aceves-Fernández



Perspective Chapter: Airborne Pollution (PM_{2.5}) Forecasting Using Long Short-Term Memory Deep Recurrent Neural Network Optimized by Gaussian Process

Marco Antonio Olguin-Sanchez,

Marco Antonio Aceves-Fernández,

Jesus Carlos Pedraza-Ortega and Juan Manuel Ramos-Arreguín

Abstract

Forecasting air pollution is a challenging problem today that requires special attention in large cities since they are home to millions of people who are at risk of respiratory diseases every day. At the same time, there has been exponential growth in the research and application of deep learning, which is useful to treat temporary data such as pollution levels, leaving aside the physical and chemical characteristics of the particles and only focusing on predicting the next levels of contamination. This work seeks to contribute to society by presenting a useful way to optimize recurrent neural networks of the short and long-term memory type through a statistical process (Gaussian processes) for the correct optimization of the processes.

Keywords: deep learning, gaussian process, optimization, recurrent neural network, long-short term memory, airborne pollution

1. Introduction

Recurrent neural networks (RNN), especially Long Short Term Memory (LSTM), have proved their efficiency in working on time-dependent values by (as its names indicate) the use of memory (sequences) gives enough information to the network to work properly finding patterns and trends in the values, which are not so obvious at first glance. Also, the gaussian processes are a useful statistical technique that allows the hyperparameters of the network since it has shown that the processing time is reduced and, at the same time, the accuracy may be improved [1]. Afterward, there is airborne pollution, which is a complex system that affects billions of people worldwide, especially in a metropolis such as Hotan, China. Shanghai, China, Ghaziabad, India, or in the case of this study, Mexico City, Mexico. Also, there we have a lot of

types of particles interacting with each other in chemical, biological, and physical ways. The pollutants that are monitored by the SEDEMA's network in the City of Mexico are nitrogen dioxide (NO_2), Ozone (O_3), sulfur dioxide (SO_2) and particulate matter ($PM_{2.5}$, and PM_{10}). Hence, we propose the use of an RNN-LSTM and optimizing its hyperparameters using Gaussian processes to increase the accuracy in the forecast of airborne pollution instead of the use of the current method.

2. Literature review

2.1 Airborne pollution

Air pollution has been a problem that has been increasing in recent decades mainly in large cities, bringing with it respiratory diseases [2, 3]. This contamination is accompanied by the same pollutant particles that have a useful life depending on physical parameters (size and shape) and their chemical composition. Mexico City has been studied for decades [4] due to its high levels of pollution that affect more than 20 million people. The sites used in this work are the following: Northeast (Gustavo A. Madero—GAM, FES Aragón—FAR, Xalostoc—XAL), Northwest (Tlalnepantla—TLA,), Center (Hospital General de México—HGM, Merced—MER), Southeast (Nezahualcóyotl—NEZ, Santiago Acahualtepec—SAC) and Southwest (Ajusco Medio—AJM, Pedregal—PED, Santa Fe, SFE). The map of the monitoring sites is shown in **Figure 1**.

2.1.1 Multiple imputation by chained equations (MICE)

Dealing with the missing data problem often leads to two general approaches for imputing multivariate data: Joint modeling (JM) and fully conditional specification

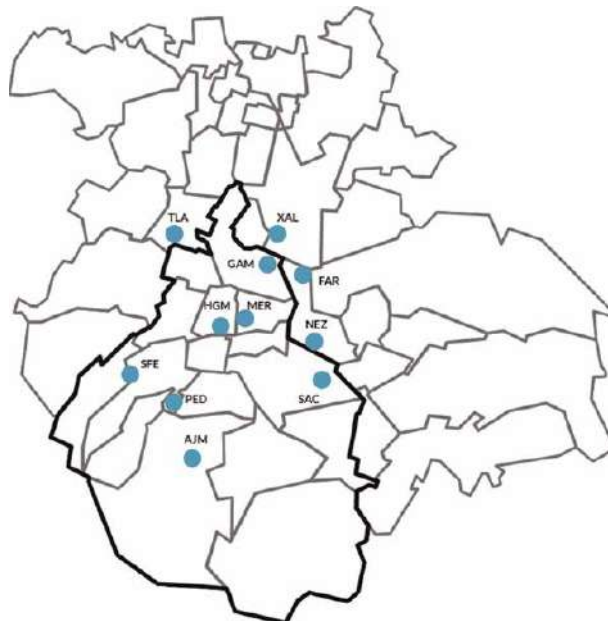


Figure 1.
Location of the monitoring sites in Mexico City.

(FCS), also known as multivariate imputation by chained equations (MICE) [5]. It is known that is a JM-type problem when we must specify a multivariate distribution for the missing data and obtain the imputation of its conditional distributions through Markov Monte Carlo chains (MCMC) techniques. On the other hand, it is FCS, which specifies the multivariate imputation model on a variable-by-variable basis using a set of conditional densities, one for each incomplete variable. The imputation starts by iterating over the conditional densities, usually, a low number of iterations is enough. In order to explain the model, let us use the following notation [5]: Let Y_j with $(j = 1, \dots, p)$ be one of p incomplete variables and $Y = (Y_1, \dots, Y_p)$. The observed and missing parts of Y_j are denoted by Y_j^{obs} and Y_j^{mis} , respectively, then $Y^{obs} = (Y_1^{obs}, \dots, Y_p^{obs})$ and $Y^{mis} = (Y_1^{mis}, \dots, Y_p^{mis})$, these are the observed and missing data respectively in Y . The number of imputations is $m \geq 1$. The h -th imputed data set is denoted as $Y^{(h)}$ where $h = 1, \dots, m$. Now let $Y_{-j} = (Y_1, \dots, Y_{j-1}, Y_{j+1}, \dots, Y_p)$ denote the collection of the $p - 1$ variables in Y except Y_j . Finally, let Q denote the quantity of scientific interest. The mice algorithm has three main steps: imputation, analysis, and pooling. The analysis starts with an incomplete data set Y_{obs} . The second step is to compute Q on each imputed data set, here the model is applied to $Y^{(1)}, \dots, Y^{(m)}$ in the general identical. Finally, the third step is to pool the m estimates $\hat{Q}^{(1)}, \dots, \hat{Q}^{(m)}$ into one estimate \hat{Q} and estimate its variance.

2.1.2 Recurrent neural networks

Recurring neural networks (better known as RNN) can be used for any type of data. In practical applications, the use of symbolic values is more common. In a recurrent neural network, there is a one-to-one correspondence between the layers of the network and specific positions in the sequence. The position in the sequence is also known as its timestamp. Finally, RNNs are complete Turing, which means that this type of network can simulate any algorithm with sufficient data and computational resources [6]. A representation of this kind of network is shown in **Figure 2**.

2.1.3 Long-short term memory (LSTM)

To represent the hidden states of the k th hidden states (layer) the notation $h_t^{-(k)}$ is used and to simplify the notation it will be assumed that the input layer \bar{x}_t can be denoted by $h_t^{-(0)}$ (this layer is not hidden) [7]. To obtain good results, a hidden vector of dimension p must also be included, which will be denoted by $c_t^{(k)}$ and refers to the state of the cell. The state of the trap can be observed as the long-term memory within the network. The matrix that updates the values is denoted by $W^{(k)}$ and is used to permute the column vectors $[h_t^{-(k-1)}, h_{t-1}^{-(k)}]^T$. The matrix that is obtained always results in dimensions $4p \times 2p$. A $2p$ size vector is then premultiplied by the $W^{(k)}$ matrix resulting in a $4p$ vector. Now to find the updates we have the following; for setting up intermediates Eq. 1, for selectively forget and add to long-term memory eq. 2, for selectively leak long-term memory to hidden state (Eq. 3).

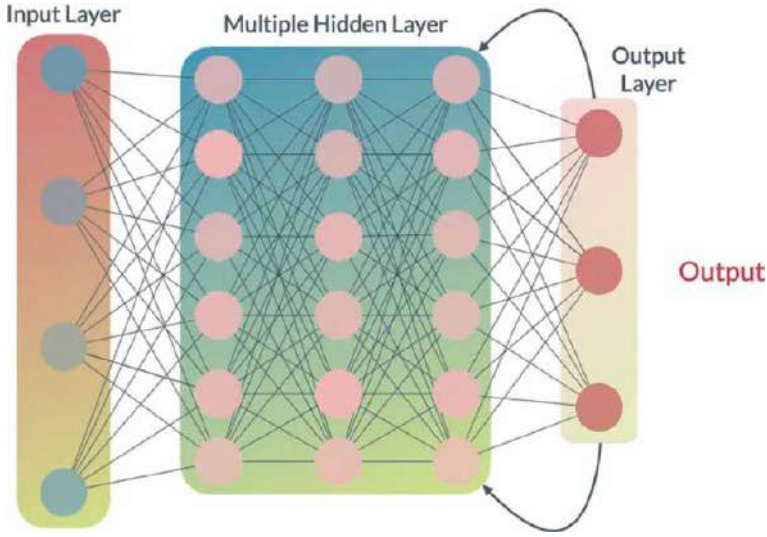


Figure 2.
Graphic representation of a recurrent neural network.

$$\begin{matrix} \text{InputGate} : \\ \text{ForgetGate} : \\ \text{OutputGate} : \\ \text{NewC - State} : \end{matrix} \begin{bmatrix} \bar{i} \\ \bar{f} \\ \bar{o} \\ \bar{c} \end{bmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{sigm} \end{pmatrix} W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^k \end{bmatrix} \quad (1)$$

$$\bar{c}_t^{(k)} = \bar{f} \odot \bar{c}_{t-1}^{(k)} + \bar{i} \odot \bar{c} \quad (2)$$

$$\bar{h}_t^{(k)} = \bar{o} \odot \tanh(\bar{c}_t^{(k)}) \quad (3)$$

Additionally, clarify that LSTM is an algorithm that belongs to recurrent neural networks or RNN [8]. The RNN's refer to neural networks that take their previous state as input, this means that the neural network will have two inputs, the new information entered into the network and its previous state, which is shown in **Figure 2**. With this model we can have short-term memory in the neural network [9]. These neural networks have applications in sequential predictions, that is, predictions that depend on a temporal variable.

2.2 Bayesian optimization using Gaussian processes

2.2.1 Multidimensional Gaussian distribution

To talk about Gaussian processes, we must first define a multivariable Gaussian distribution in several dimensions. Formally, this distribution is expressed as in Eq. 4:

$$p(x) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (4)$$

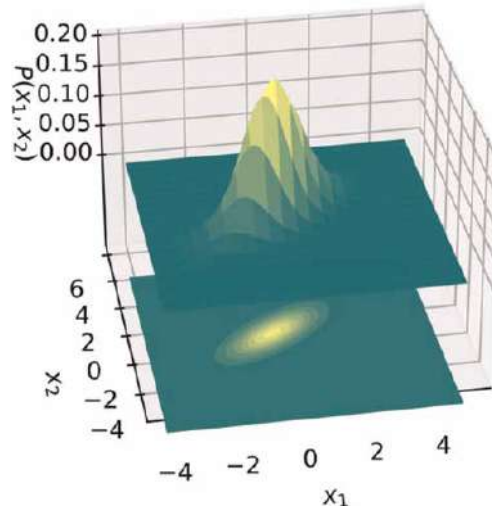


Figure 3.
Graphic example of a multidimensional gaussian distribution.

Where D is the number of dimensions, x is the variables, μ is the average vector, Σ is the covariance matrix. Gaussian processes try to model a function f given a set of points [10]. Traditional nonlinear regression machine learning methods usually give a function that they think best fits these observations. But, there may be more than one function that fits the observations equally well. When we have more observation points, we use our posterior-anterior as our anterior, we use these new observations to update our posterior. This is the Gaussian process. A Gaussian process is a probability distribution over possible functions that fit a set of points. Because we have the probability distribution over all possible functions, we can calculate the means as the function and calculate the variance to show how confident we are when we make predictions using the function (**Figure 3**).

2.2.2 Gaussian process

Because we have the probability distribution over all possible functions, we can calculate the means as the function and calculate the variance to show how confident when predictions are made using the function as demonstrated by Wang [10], we must take into account that:

- I.The (later) functions are updated with new observations.
- II.The mean calculated by the posterior distribution of the possible functions is the function used for the regression.

The function is modeled by a multivariable Gaussian of the form shown in Eq. 5:

$$P(f|X) = N(f|\mu, K) \quad (5)$$

Where $X = [x, \dots, x_n], f =, \dots, f(x_n), \mu =, \dots, m(x_n), K_{i,j} = k(x_i, x_j)$. Being X the points of the observed data, m represents the average function and K represents a definite positive kernel.

3. Materials and methods

3.1 Materials

The data used to train the model were obtained from the Atmospheric Monitoring System (SIMAT for its acronym in Spanish) database is conformed of four subsystems: RAMA, REDMA, REDMET, and REDDA, all are given by its website but we just focus on the Automatic Environmental Monitoring Network (RAMA for its acronym in Spanish).

3.2 Methodology

3.2.1 Data acquisition and preprocessing data

First, the database on air quality, RAMA (SEDEMA, 2021) [11] must be downloaded, which is public. This file (dataset) contains values captured by all stations capable of monitoring $PM_{2.5}$. In case the dataset contains more variables in addition to the one already mentioned, a preprocessing process must be carried out. First, the values of interest ($PM_{2.5}$ and air direction) should be classified, excluding any other. Once the data has been classified, it will be necessary to determine if there are missing data and in which cases it is convenient to impute because if the amount of data to be imputed exceeds 40% of the total data, it is advisable not to use that station since there is a loss very large data and a case of over-learning or data that does not reflect reality could be presented. To impute the missing data, the MICE algorithm will be used and once the algorithm has been applied, a new dataset will have to be generated with the imputed data (complete). Because the RNN-LSTM works by taking a tensor as input and already having an absent dataset of missing data, now it will be necessary to divide this data into three different datasets which would serve for training, testing, and data validation. To conclude with the preprocessing, the data will be normalized and later converted into tensors. To normalize the data, the min-max normalization will be used, which takes the minimum value of the data as “zero” and the maximum value as “one” and it is based on these that the normalization is performed. For tensors, they must take into account the batch value, which in turn takes values of 2^n with $n \in \mathbb{N}$. The value of the sequence to use as a parameter should also be considered when creating the tensors.

3.2.2 Instantiate LSTM and optimize the model

To begin with the training and optimization of the network, we are going to start the network with values of the hyperparameters selected at random, this is only to make the network start and work since later the values of each selected hyperparameter will be rewritten in optimization until the optimal point is found within the search space that is established. By having the data imputed, divided,

normalized, and transformed into tensors now, using the training dataset, as its name implies, we will begin the process of training the network, which will go hand in hand with the number of iterations (points) that have been assigned to the optimization process. In each iteration, an adjustment will be made in some of the hyperparameters, and saving the score of each model to end up with the best one.

3.2.3 Model evaluation

Having already a trained and optimized model, we can now determine the efficiency of the model by calculating the RMSE that the model has by comparing the predicted data against the real data. To generate predictions, we are going to use the evaluation set (**Figure 4**).

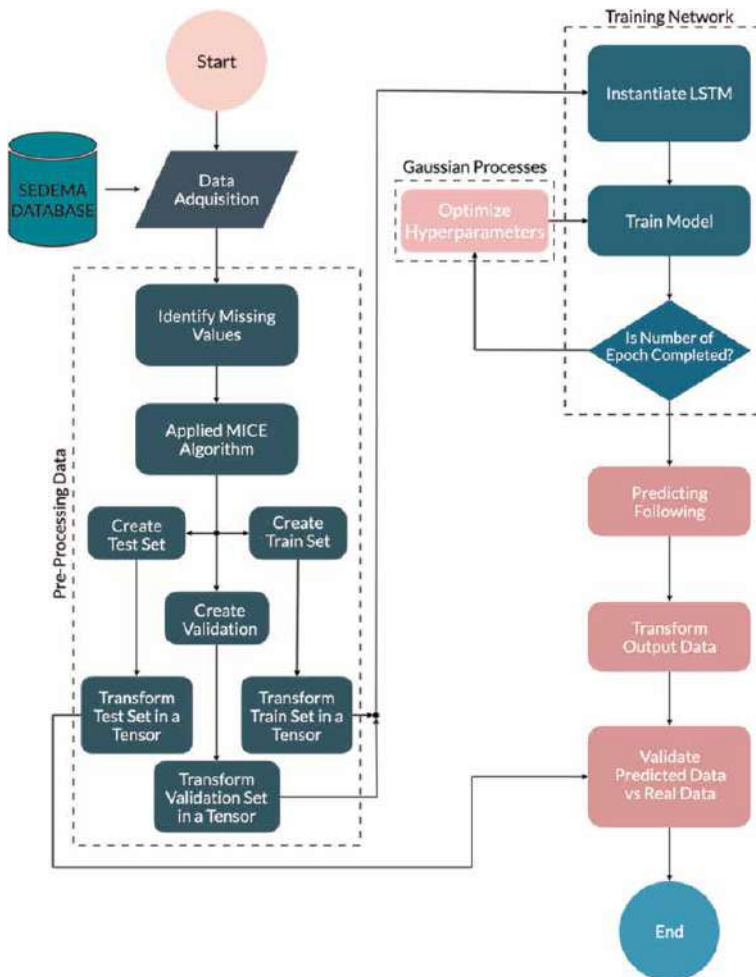


Figure 4.
Proposed methodology.

4. Results and discussion

There was a considerable amount of missing values all across the data. After applying the MICE algorithm the data full fill. In order to confirm that the imputation was successfully checked that the distribution of the data did not change as is shown in **Figures 5** and **6**. These processes will be repeated for each station. This process was repeated for all the stations which were selected getting similar results in all cases. Once the preprocessing was finished, the training was started and by optimizing the model, a search space was stabilized for each hyperparameter of the RNN-LSTM considered. Now, with the model ready and tested, the results of the optimization process can be seen within the network. As is shown in **Figures 7** and **8**, for the LSTM optimized by GP, the loss function during training decreases rapidly in each epoch until it converges and the change between epochs is no longer so noticeable compared with simple LSTM (**Figures 9** and **10**). In both cases when the converges are reached, it means that the network has already stopped learning. Finally, the validation set is used to estimate the skills of the network obtained in its training for the prediction of $PM_{2.5}$ levels. This is shown for the LSTM optimized by GP in **Figures 11** and **12**, and

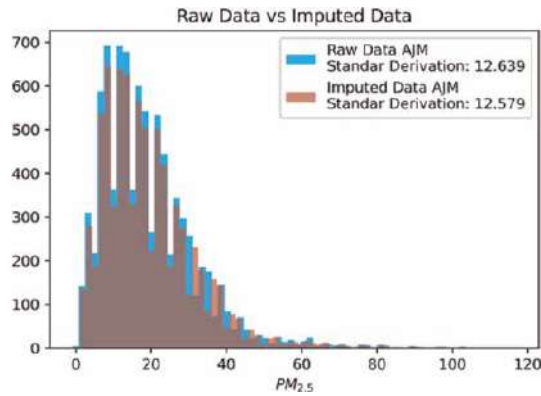


Figure 5. Data distribution of AJM station before and after being imputed.

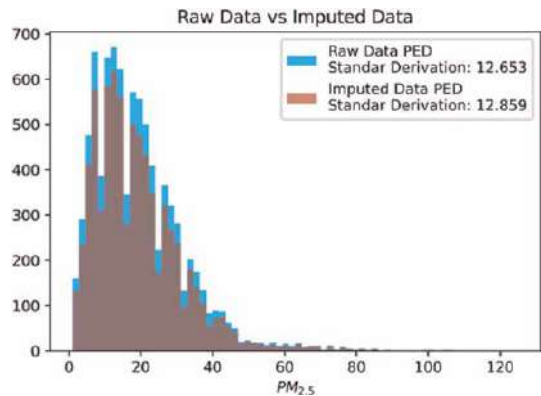


Figure 6. Data distribution of PED station before and after being imputed.

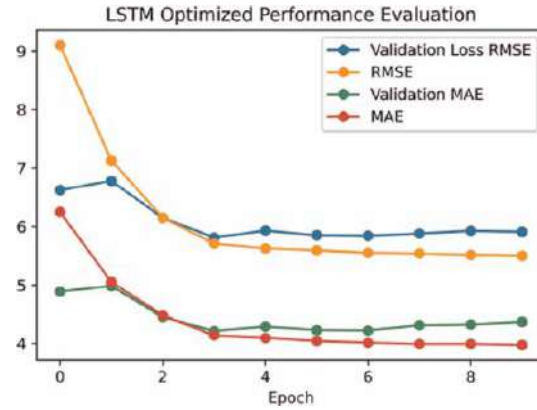


Figure 7.
Metrics obtained with LSTM model optimized by Gaussian process for the AJM station.

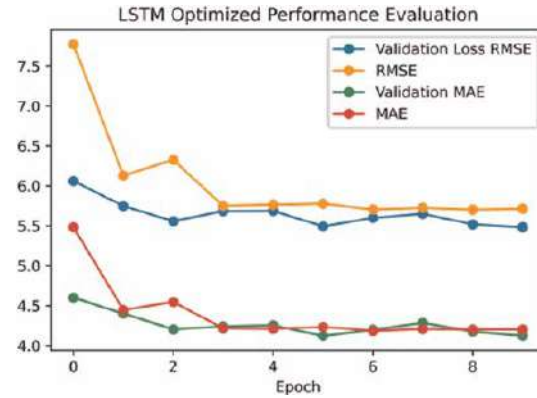


Figure 8.
Metrics obtained with LSTM model optimized by Gaussian process for the PED station.

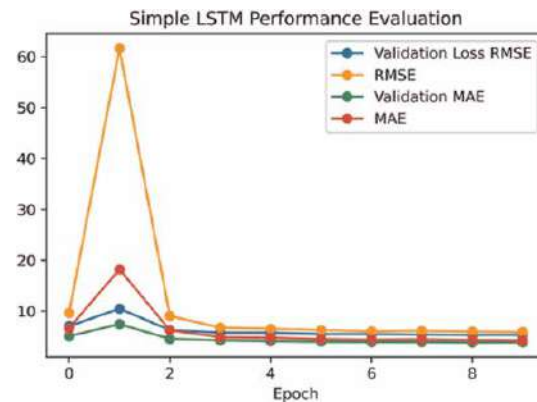


Figure 9.
Metrics obtained with simple LSTM model for the AJM station.

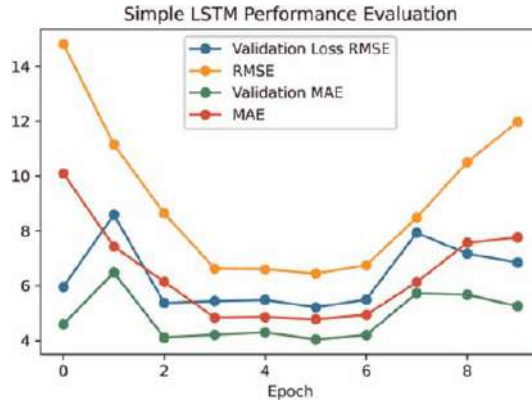


Figure 10 Metrics obtained with simple LSTM model for the PED station.

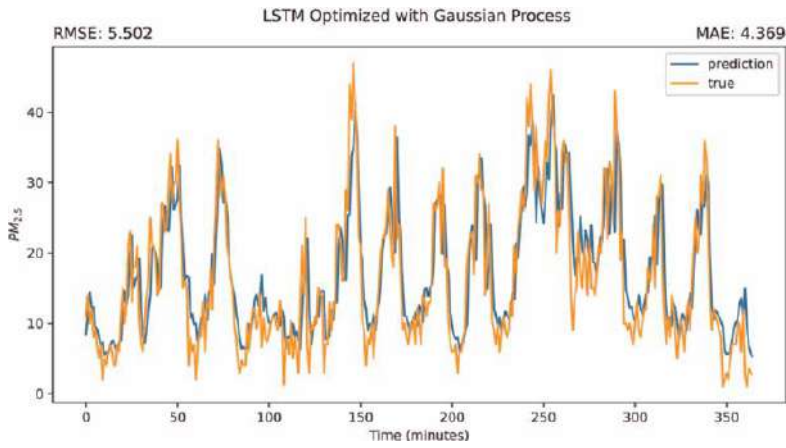


Figure 11. LSTM optimized by gaussian process, validation forecast for AJM station.

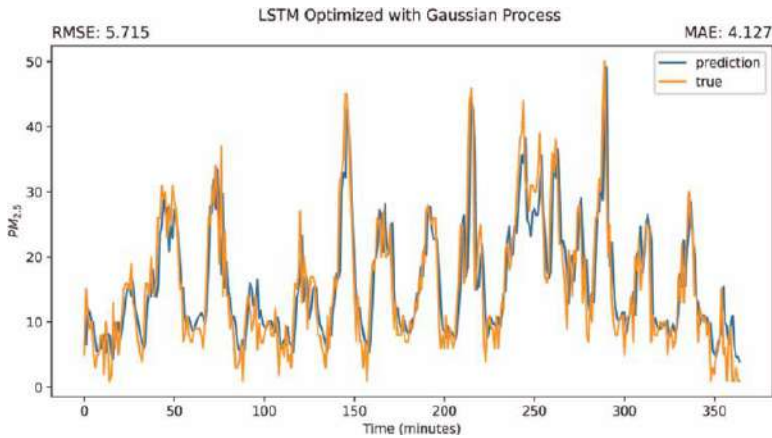


Figure 12. LSTM optimized by gaussian process, validation forecast for PED station.

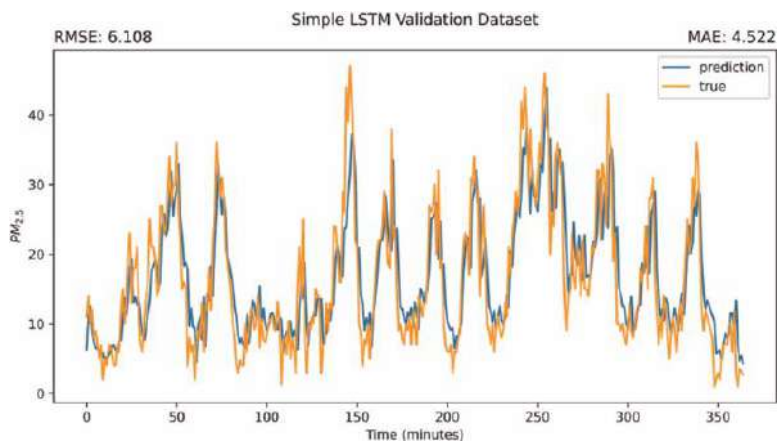


Figure 13.
Simple LSTM, validation forecast for PED station.

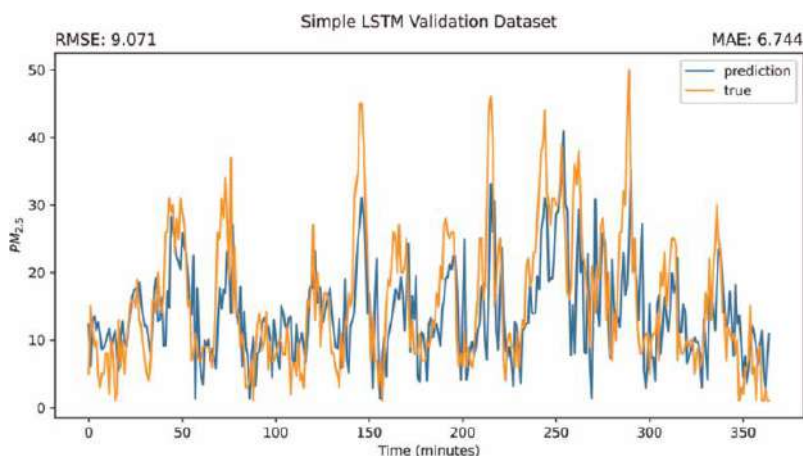


Figure 14.
Simple LSTM, validation forecast for PED station.

for the simple LSTM the results are shown in **Figures 13** and **14**. The results show are from AJM and PED stations.

5. Conclusion

The potential of using Gaussian Processes to improve the hyperparameters tuning is based on a strong mathematical model compared with other methods of hyperparameter tuning; this gives more confidence when looking for an optimized deep learning model. On the other hand, to use this method requires more time to compute the Bayesian search and it may be a thing to consider.

About the project, it does not pretend to end the airborne pollution problem, but it introduces a useful tool for the government and the citizen in order to prevent and plane his day because the model can make predictions along more than three hundred hours with high accuracy, so this gives 12 days to the corresponding users to avoid or prevent the contamination levels.

Conflict of interest

The authors declare no conflict of interest

Consent for publication


Not applicable

Author details

Marco Antonio Olguin-Sanchez, Marco Antonio Aceves-Fernández*,
Jesus Carlos Pedraza-Ortega and Juan Manuel Ramos-Arreguín
Autonomous University of Queretaro, Queretaro, Mexico

*Address all correspondence to: marco.aceves@uaq.mx

IntechOpen

© 2022 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Sánchez CJ. Características físico-químicas de los gases y partículas con- taminantes del aire. Su impacto en el asma. [Internet]. Available from: www.scielo.org.co/scielo.php?script=sciabstract&pid=S0121-0793201200040007&lng=en&nrm=iso&tlng=es [Accessed: October 10, 2022]
- [2] Arbex MA, Saldiva PH, Pereira LA, Braga AL. Impact of outdoor biomass air pollution on hypertension hospital admissions. *Journal of Epidemiology and Community Health*. 2010;**64**(7):573-579. DOI: 10.1136/jech.2009.094342
- [3] Guo Y, Barnett AG, Zhang Y, Tong S, Weiwei Y, Pan X. The short-term effect of air pollution on cardiovascular mortality in Tianjin, China: Comparison of time series and case–crossover analyses. *Science of The Total Environment*. 2010;**409**(2):300-306. DOI: 10.1016/j.scitotenv.2010.10.013
- [4] Stephens SM, Sasha W, Olson F, Ramos J, Retama R, Armando, et al. Weekly patterns of México City’s surface concentrations of CO, NO_x, PM₁₀, and O₃ during 1986-2007. *Atmospheric Chemistry and Physics*. 2008;**8**: 5313-5325. DOI: 10.5194/acpd-8-8357-2008
- [5] van Buuren S, Groothuis-Oudshoorn K. Mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*. 2011;**45**(3):1-67. DOI: 10.18637/jss.v045.i03
- [6] Siegelmann HT, Sontag ED. On the computational power of neural nets. *Association for Computing Machinery*. 1992;**1995**:440-449. DOI: 10.1145/130385.13043
- [7] Aggarwal CC. *Neural Networks and Deep Learning*. Midtown Manhattan, New York City: Springer Cham; 2018. pp. 293-294. DOI: 10.1007/978-3-319-94463-0
- [8] Kuri-Monge GJ, Aceves-Fernández MA, Ramírez-Montañez JA, Pedraza-Ortega JC. Capability of a recurrent deep neural network optimized by swarm intelligence techniques to predict exceedances of airborne pollution (PM_x) in largely populated areas. In: 2021 International Conference on Information Technology (ICIT), 2021. Amman, Jordan: IEEE; 2021. pp. 61-68
- [9] Ramírez Montañez JA, Aceves Fernandez MA, Arriaga ST, Ramos Arreguin JM, Salini Calderon GA. Evaluation of a recurrent neural network LSTM for the detection of exceedances of particles PM₁₀. In: 2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). Mexico City, Mexico: IEEE; 2019. pp. 1-6
- [10] Wang J. An Intuitive Tutorial to Gaussian Processes Regression. [Internet]. 2020. Available from: https://www.researchgate.net/publication/344359964_An_Intuitive_Tutorial_to_Gaussian_Processes_Regression [Accessed: February 22, 2022]
- [11] SEDEMA. Red automática de monitoreo ambiental [Internet]. 2018. Available from: <http://www.aire.cdmx.gob.mx/> [Accessed: February 23, 2021]