



UNIVERSITAT OBERTA DE CATALUNYA (UOC)

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS (*Data Science*)

TRABAJO FINAL DE MÁSTER

ÁREA: 5

Detección eficiente de anomalías en sensores de CO₂ en tiempo real

Un sistema jerárquico con computación Edge y Cloud para mejorar la calidad del aire en interiores

Autor: Javier Gallego Fernández

Tutor: Sergio Trilles Oliver

Profesor: Albert Solé Ribalta

Santiago de Compostela, 21 de junio de 2023

Créditos/Copyright



Esta obra está sujeta a una licencia de Reconocimiento - NoComercial - SinObraDerivada 3.0 España de [CreativeCommons](#).

FICHA DEL TRABAJO FINAL

Título del trabajo:	Detección eficiente de anomalías en sensores de CO2 en tiempo real
Nombre del autor:	Javier Gallego Fernández
Nombre del colaborador/a docente:	Sergio Trilles Oliver
Nombre del PRA:	Albert Solé Ribalta
Fecha de entrega (mm/aaaa):	07/2023
Titulación o programa:	Máster Universitario en Ciencia de Datos
Área del Trabajo Final:	Área 5
Idioma del trabajo:	Español
Palabras clave	anomalía, CO2, calidad del aire

Abstract

Monitoring the concentration of carbon dioxide (CO₂) is essential to ensure good indoor air quality. However, detecting anomalies in CO₂ sensors can be difficult due to variations in data patterns and processing limitations of the devices. This work presents a hierarchical system of Machine Learning models that uses Edge and Cloud computing to detect anomalies in CO₂ sensors. The system consists of a more limited model that runs on the device (Edge) and a more complex model that runs on the Cloud. The selection of the work mode is done in the server, allowing efficient and accurate detection of anomalies in CO₂ sensors. The results show that this system provides a scalable and effective solution for detecting CO₂ sensor anomalies in real-time, which can significantly improve indoor air quality.

Keywords: anomaly, CO₂, air quality, detection, IoT, edge computing, cloud computing, machine learning

Resumen

El seguimiento de la concentración de dióxido de carbono (CO2) es esencial para garantizar una buena calidad del aire en interiores. Sin embargo, la detección de anomalías en los sensores de CO2 puede ser difícil debido a las variaciones en los patrones de los datos y a los límites en la capacidad de procesamiento de los dispositivos. En este trabajo se presenta un sistema de jerarquía de modelos de Machine Learning que utiliza la computación Edge y Cloud para detectar anomalías en los sensores de CO2. El sistema se compone de un modelo más limitado que se ejecuta en el dispositivo (Edge) y de un modelo más complejo que se ejecuta en el Cloud. La selección del modo de trabajo se realiza desde el servidor, lo que permite una detección eficiente y precisa de las anomalías en los sensores de CO2. Los resultados muestran que este sistema proporciona una solución escalable y efectiva para la detección de anomalías en sensores de CO2 en tiempo real, lo que puede mejorar significativamente la calidad del aire en interiores.

Palabras clave: anomalía, CO2, calidad del aire, detección, IoT, computación edge, computación cloud, machine learning

Índice general

Abstract	v
Resumen	vii
Índice	ix
Listado de Figuras	xI
Listado de Tablas	1
1. Introducción	3
1.1. Descripción y justificación de la relevancia de la propuesta	3
1.2. Motivación personal	4
1.3. Definición de los objetivos	4
1.3.1. Objetivo principal	4
1.3.2. Objetivos secundarios	4
1.4. Descripción de la metodología	5
1.5. Impacto en sostenibilidad, ético-social y de diversidad del Trabajo	6
1.6. Planificación temporal	7
2. Estado del arte	9
2.1. Concepto de anomalía	9
2.2. Trabajos relacionados	11
2.2.1. Modelos de detección de anomalías	11
2.2.2. Framework de selección de modelos	14
2.3. Conclusiones	14
3. Análisis y diseño	17
3.1. Fuentes de datos	17
3.2. Tecnologías y herramientas	19

3.2.1. Lenguajes de programación	19
3.2.2. Bibliotecas	19
3.2.3. Frameworks	20
3.2.4. Otras herramientas	20
3.3. Hardware	20
3.4. Diagramas del sistema	21
4. Procesado y análisis del dataset	23
4.1. Exploración inicial	23
4.2. Limpieza de datos	24
4.3. Análisis de datos	26
5. Generación de los modelos	37
5.1. Creación de los conjuntos de datos	37
5.1.1. Imputación de datos	37
5.1.2. Etiquetado de datos	38
5.1.3. Eliminación de anomalías	39
5.1.4. Normalización de datos	40
5.1.5. Creación de ventanas deslizantes	41
5.1.6. Subconjuntos de datos	42
5.2. Creación de los modelos	43
5.2.1. Modelo Coud: LSTM-AE	44
5.2.2. Modelo Standalone: LSTM-AE	47
5.2.3. OneClass - SVM	48
5.2.4. Isolation Forest	49
5.3. Comparativa de modelos	49
6. Experimentos realizados	53
7. Conclusiones	55
Appendices	57
A. Repositorio de Código	59
Bibliografía	61

Índice de figuras

1.1.	Diagrama de proceso que muestra las diferentes fases de CRISP-DM [1]	6
1.2.	Diagrama de Gantt que ilustra la planificación temporal del proyecto. <i>Fuente:</i> <i>Elaboración propia</i>	8
2.1.	Tipos de anomalía en una serie temporal de CO2.	10
3.1.	Mapa con la ubicación de los colegios en los que se desplegaron los sensores.	18
3.2.	Arquitectura del sistema.	21
3.3.	Diagrama de secuencia del modo Cloud.	22
3.4.	Diagrama de secuencia del modo Standalone.	22
4.1.	Nombres de los sensores de cada escuela.	26
4.2.	Boxplot de la variable CO2.	27
4.3.	Gráfico circular con la distribución de los sensores.	29
4.4.	Gráfico circular con la distribución de los días de la semana.	30
4.5.	Máximos de CO2 mensuales.	31
4.6.	Máximos de CO2 por día de la semana.	32
4.7.	Máximos de CO2 por hora del día.	32
4.8.	Distribución de los valores de CO2 en dos franjas horarias.	33
4.9.	Distribución de los valores de CO2 en cada sensor de Sant Miquel.	34
4.10.	Distribución de los valores de CO2 en cada sensor de Albea.	34
4.11.	Test de Shapiro-Wilk.	35
4.12.	Mapa de calor con la correlación entre variables.	36
5.1.	Serie temporal del sensor <i>CO2_04</i> de Albea antes y después de la imputación de datos.	38
5.2.	Distribución de los datos de CO2 con umbrales definidos por la regla 3-(IQR).	39
5.3.	Ejemplo de <i>sliding window</i> con paso 1 y longitud de secuencia 6.	41
5.4.	Proceso general de detección de anomalías con LSTM_AE.	44

5.5.	Topología del modelo LSTM_AE de 3 capas.	44
5.6.	Importancia de los hiperparámetros para minimizar el error de reconstrucción máximo	46
5.7.	Topología del modelo LSTM_AE de 1 capa implementado con TensorFlow Lite.	47
5.8.	Curva ROC de los modelos desarrollados.	51
6.1.	Entorno de pruebas	53
6.2.	<i>Dashboard</i> que muestra la serie temporal de mediciones de CO ₂	54

Índice de tablas

2.1. Modelos de <i>deep learning</i> para detección de anomalías en series temporales univariadas. Adaptado de [2].	12
3.1. Datos de los sensores de las escuelas	18
4.1. Datos de los sensores de CEIP L'Albea.	23
4.2. Datos de los sensores de CEIP Sant Miquel.	24
4.3. Lista de atributos del dataset inicial.	25
4.4. Dataset final.	25
4.5. Análisis descriptivo del dataset.	28
5.1. Combinación de hiperparámetros del modelo LSTM_AE de 3 capas con su error de reconstrucción máximo	46
5.2. Tabla comparativa con las métricas obtenidas por todos los modelos desarrollados.	50
5.3. Tabla comparativa con los consumos medios de CPU y memoria RAM de los modelos.	51

Capítulo 1

Introducción

1.1. Descripción y justificación de la relevancia de la propuesta

El Internet de las Cosas (IoT, en sus siglas en inglés) es una tecnología que permite la interconexión y comunicación entre objetos cotidianos, desde electrodomésticos hasta vehículos, mediante sensores, software y conectividad de red [3][4]. Esta innovación tecnológica ha sido posible gracias al rápido avance en la miniaturización de sensores, el desarrollo de redes de comunicación de alta velocidad y la computación en la nube.

El término IoT fue acuñado por Kevin Ashton [5] en el año 1999 y refleja correctamente esta tendencia de interconectar todo tipo de objetos en diferentes sectores, desde el hogar hasta la industria, pasando por la salud y el transporte. En este contexto, los sensores juegan un papel fundamental en la recopilación de datos del mundo real. Estos instrumentos capaces de medir distintas magnitudes son ampliamente utilizados por su capacidad de conectarse a la red y transmitir datos en tiempo real, lo que facilita la toma de decisiones y la interacción con el entorno.

En este trabajo nos centraremos en un tipo de dispositivos concreto, los sensores de dióxido de carbono (CO₂) para interiores. Existen estudios que relacionan los niveles de CO₂ en espacios cerrados, como aulas escolares o hospitales, con distintas dolencias físicas [6]. También se ha comprobado que la concentración de CO₂ y otros aerosoles en interiores está correlacionada con la del SARS-CoV-2¹ [7].

Llevar un control de los niveles de CO₂ es una tarea que puede realizarse mediante la implementación de algoritmos de *Machine Learning* (ML) para la detección de anomalías. Ya que los dispositivos de CO₂ comerciales son destinados al público general y deben ser asequibles

¹Tipo de coronavirus causante de la COVID-19

para su compra, tienen una capacidad de cómputo y memoria limitada para no encarecerlos en exceso. Esto dificulta la tarea de aprovechar su potencial hardware y desplegar el software de detección de anomalías en el propio equipo.

El uso de sistemas interconectados, abre la posibilidad de desplazar la tarea de cálculo hacia la nube, eliminando el problema mencionado anteriormente. Esta solución sería ideal si no fuera por una razón. El envío de datos por la red introduce cierta latencia que puede provocar un retraso en la detección.

Este estudio pretende construir un sistema que solvente o mitigue en gran parte los problemas de ambas propuestas. Para ello, se aprovechará tanto la computación en la nube como la computación en el propio dispositivo.

1.2. Motivación personal

En esta sección se recogen las razones por las que se ha escogido este proyecto para la realización del *Trabajo de Fin de Máster* del *Máster en Ciencia de Datos* de la UOC.

- **Interés personal:** el mundo del *edge computing* y los dispositivos IoT es un campo emergente que me despierta una gran curiosidad, por lo que realizar un proyecto de este alcance puede suponer un gran aprendizaje sobre el tema.
- **Experiencia profesional:** he trabajado para una empresa encargada de la fabricación de dispositivos, desde el hardware hasta el software. Siempre he participado en el desarrollo de soluciones software que interactúen con el sistema y me llama la idea de introducir estas soluciones en el propio equipo.
- **Interés social:** me gustaría aportar valor a este ámbito, ya que me parece que podría repercutir en la salud y bienestar de la sociedad.

1.3. Definición de los objetivos

1.3.1. Objetivo principal

La **hipótesis** de este estudio es que se pueden detectar anomalías en tiempo real en las mediciones de sensores de CO₂.

1.3.2. Objetivos secundarios

Para conseguir confirmar nuestra **hipótesis**, se definen los siguientes objetivos parciales:

- **Conocer y analizar** los estudios previos más relevantes sobre **detección de anomalías** en dispositivos IoT para establecer las bases del proyecto.
- **Crear un conjunto de datos** a partir de una o varias fuentes para alimentar nuestro modelo.
- **Crear un modelo de detección de anomalías** capaz de ejecutarse en el dispositivo IoT y otro sin restricciones de recursos.
- **Crear** un mecanismo de **selección de modelos** en función del dato de entrada y comparar la eficiencia de usarlo en contraste con usar cada modelo individualmente.

1.4. Descripción de la metodología

La selección de la metodología de trabajo es una tarea crítica para la elaboración de todo proyecto de Ciencia de Datos. En este ámbito, existen numerosas estrategias como **KDD**, **SEMMA** o **CRISP-DM** [8]. La metodología **Cross-Industry Standard Process for Data Mining (CRISP-DM)** [9] es una buena opción para nuestro problema debido a su enfoque iterativo y estructurado. Por su amplio uso y estructuración, existen multitud de recursos que facilitan su implementación y su carácter iterativo permite que se vaya ajustando según el proyecto avanza y evoluciona. La Figura 1.1 muestra las distintas fases de este enfoque.

Estas etapas son las siguientes:

1. **Entendimiento del negocio:** En esta fase, se busca comprender el problema que se quiere resolver. Se debe tener una idea clara de los objetivos del proyecto y el contexto en el que se aplicará la solución.
2. **Entendimiento de los datos:** Una vez que se tiene claro el problema que se busca resolver, es necesario explorar los datos disponibles. En esta fase se busca examinar la calidad de los datos, su relevancia para el problema concreto, su disponibilidad e identificar posibles patrones ocultos.
3. **Preparación de los datos:** En esta fase, se limpian, integran y transforman los datos para que sean aptos para el modelado. Es necesario preparar los datos de manera que se ajusten a las necesidades del proyecto y se puedan utilizar para las etapas siguientes.
4. **Modelado de los datos:** Con los datos preparados, se crearán los modelos para la tarea de detección de anomalías. Además, se dividirán los datos en los conjuntos de entrenamiento, validación y test, se refinrarán los modelos para obtener el mejor rendimiento y se realizará su entrenamiento.

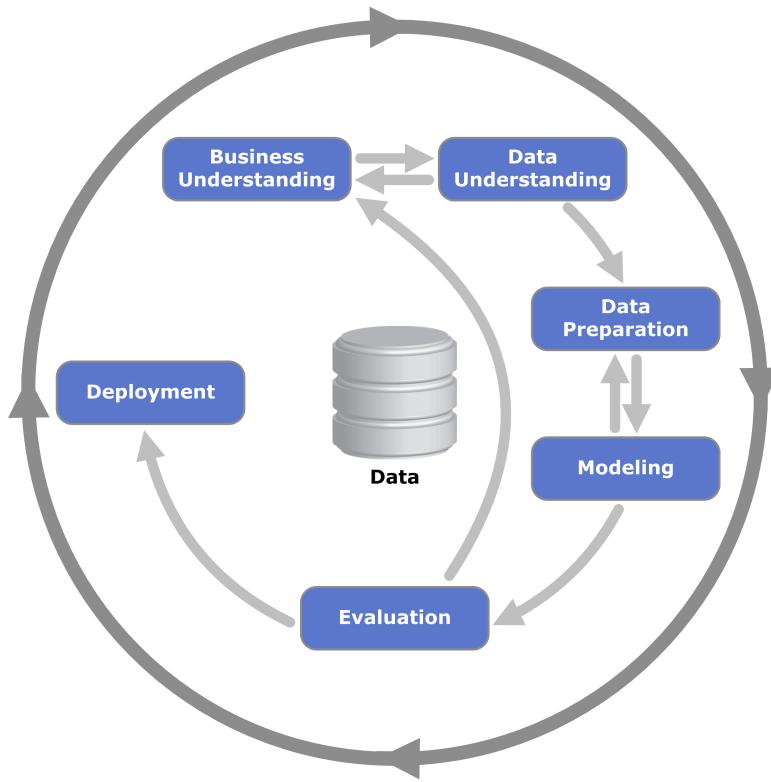


Figura 1.1: Diagrama de proceso que muestra las diferentes fases de CRISP-DM [1]

5. **Evaluación:** Una vez que se ha creado y entrenado el modelo, es necesario evaluar su rendimiento utilizando las métricas definidas. Se debe validar el modelo y asegurar su calidad. Si no se logra el desempeño buscado, se debe volver a realizar otra iteración del proceso completo.
6. **Despliegue:** En la última fase, se implementa la solución en el entorno de producción. Se asegura que la solución sea compatible con el entorno y se hacen pruebas para asegurar que funciona correctamente. En este proyecto, se desplegará en un dispositivo de medición de CO₂ y en alguna solución *cloud*.

1.5. Impacto en sostenibilidad, ético-social y de diversidad del Trabajo

Este trabajo incorpora la **Competencia de Compromiso Ético y Global (CCEG)** debido a la importancia de actuar de manera honesta, ética, sostenible, socialmente responsable y respetuosa con los derechos humanos y la diversidad. Para realizar la implementación de la CCEG se utilizarán los **Objetivos de Desarrollo Sostenible (ODS)** [10] dejando de

manifiesto aquellos con los que este proyecto se alinea y encuadrándolos en las dimensiones que aborda la CCEG.

- **Dimensión sostenibilidad.** El proyecto desarrollado busca implementar una solución innovadora para la detección de anomalías en sensores de CO₂, lo que contribuye al avance de la infraestructura y la industria sostenible, por lo que está especialmente alineado con el ODS 9, *Industria, innovación e infraestructura*. Del mismo modo, se favorece la creación de ambientes sanos y seguros para las personas, como se manifiesta en el ODS 11, *Ciudades y comunidades sostenibles*.

Al controlar la calidad del aire en interiores, se puede fomentar un consumo responsable (ODS 12 *Producción y consumo responsable*) al identificar las causas que repercuten en el impacto ambiental, reduciendo las emisiones de CO₂ (ODS 13 *Acción por el clima*) de los edificios.

- **Dimensión comportamiento ético y de responsabilidad social.** Como ya se ha mencionado antes, los niveles de CO₂ afectan directamente a la salud de las personas. Con este trabajo se pretende lograr un mejor bienestar en espacios cerrados como aulas o oficinas. Esto va ligado a los ODS 3 *Salud y bienestar* y ODS 8 *Trabajo decente y crecimiento económico*.
- **Dimensión diversidad, género y derechos humanos.** Esta dimensión es, al igual que las interiores, de una gran importancia al promover la igualdad y la inclusión, y reconocer la importancia de los derechos humanos y la diversidad cultural de nuestra sociedad. Los objetivos y resultados de este proyecto no están directamente relacionados con esta dimensión, pero se ha tratado de aportar lo máximo a esta causa al considerar la perspectiva de género (ODS 5 *Igualdad de género*) durante todo el proceso. Por ello, se ha hecho especial esfuerzo en considerar todas las aportaciones previas independientemente del género, se ha utilizado un lenguaje inclusivo y se han utilizado/adaptado los datos para prevenir la aparición de sesgos. Por ejemplo, podrían aparecer sesgos al entrenar modelos con datos donde se incorpore el género de las personas o se consideren datasets de espacios donde haya una menor cantidad de mujeres.

1.6. Planificación temporal

El proyecto se ha planificado para cumplir con los plazos de entrega marcados por la asignatura que engloba el desarrollo del *Trabajo de Fin de Máster* y los objetivos definidos en

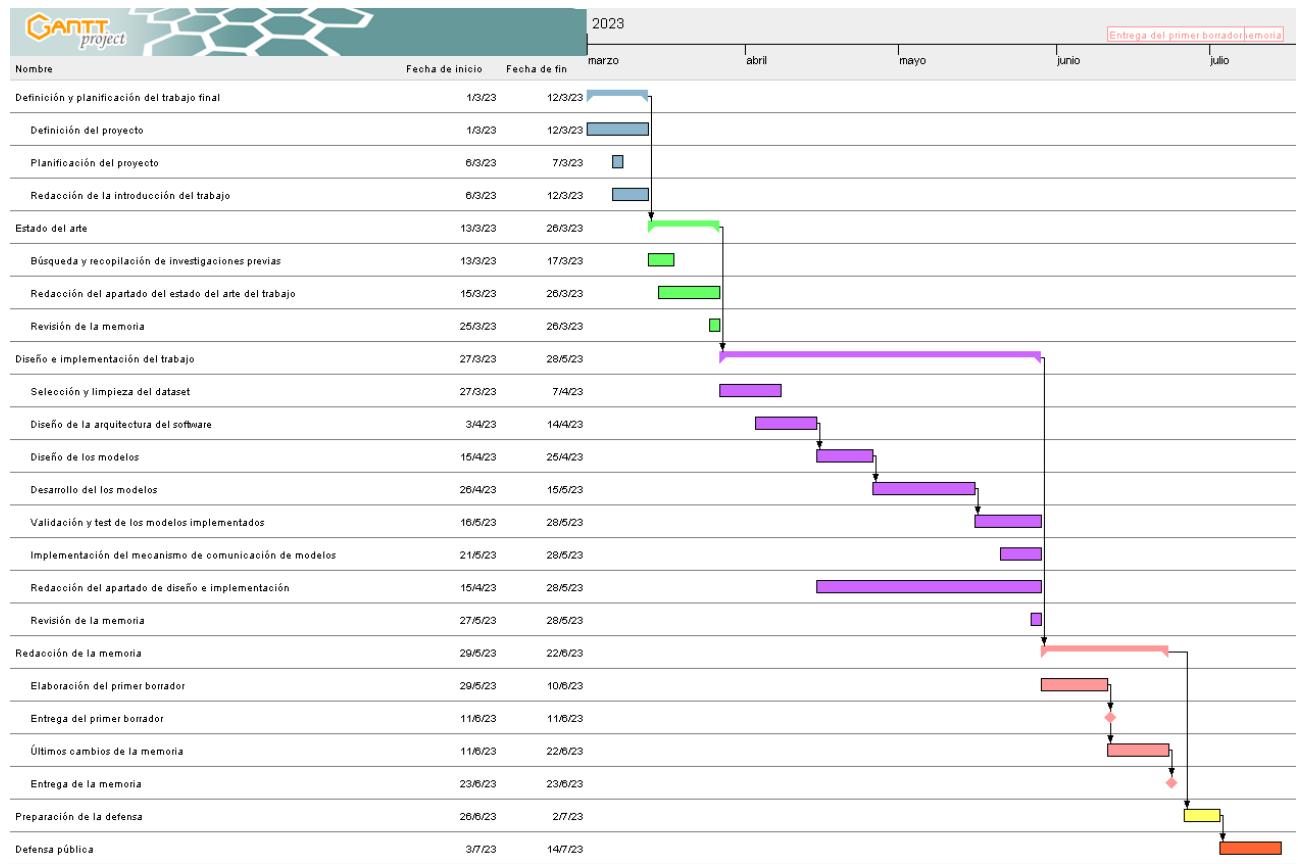


Figura 1.2: Diagrama de Gantt que ilustra la planificación temporal del proyecto. *Fuente: Elaboración propia*

secciones anteriores. Se ha dividido en seis bloques definidos por la asignatura y en sus subtareas. Además, se considera el carácter incremental de la memoria y en cada fase se incorpora una tarea de revisión de la redacción realizada hasta ese momento.

En la Figura 1.2 se puede ver el **Diagrama de Gantt** con la planificación temporal de las tareas identificadas para el correcto cumplimiento de los objetivos marcados.

Capítulo 2

Estado del arte

La detección de anomalías en series temporales y en tiempo real es un campo de gran interés actualmente, por lo que existe una gran bibliografía que aborda el tema desde distintos ámbitos. En este capítulo se recopilan una serie de artículos que exploran el **estado del arte** de esta temática, con el objetivo de comprender el contexto actual de investigación y sacar conclusiones de ello.

Para la búsqueda se han utilizado principalmente dos buscadores especializados en documentos académicos, *Google Scholar*¹ y *Web of Science*².

Los artículos analizados se han agrupado en las siguientes secciones teniendo en cuenta las partes del proyecto en las que tienen un impacto. Un artículo puede estar mencionado en varias secciones debido a sus aportaciones. A continuación, se incluye un breve resumen de cada uno de los apartados:

- **Concepto de anomalía:** Este apartado busca profundizar en este término, buscando delimitar lo que se entiende por anomalía.
- **Metodología:** Se exploran distintas implementaciones para resolver nuestro problema, tanto el de detección de anomalías como el de selección del modelo más adecuado.
- **Conclusiones:** Por último, se hace un resumen con una serie de directrices a tener en cuenta en la elaboración del proyecto.

2.1. Concepto de anomalía

El primer paso para elaborar este proyecto es investigar lo que se entiende por anomalía. En [11] se hace una distinción en tres grupos: **anomalía puntual**, **anomalía contextual** y

¹<https://scholar.google.es/>

²<https://www.webofscience.com/>

anomalía colectiva. Las anomalías **puntuales** son aquellas que se corresponden con la definición clásica de *outlier* [12], es decir, una observación puntual que se aleja significativamente del resto de observaciones. Una serie temporal volverá a su estado normal en pocas observaciones. Puede ser producida por ruido en el sistema o un evento externo. En nuestro caso podría ser una subida puntual de los niveles de CO₂, producida por una fuga o por la llegada de un gran número de personas a la localización en la que se encuentra el sensor. Las **contextuales** son aquellas que se desvían de los patrones de la serie temporal, aunque sus valores extremos se encuentren dentro del rango de valores normales. Por último, las **colectivas** son **colecciones de observaciones** que pueden no ser anomalías individualmente, pero sí como grupo. En [2] se profundiza en esta última categoría distinguiendo entre **estacionales**, desviaciones en la *estacionalidad* de la serie temporal, **de tendencia**, aquellos cambios que afectan a la *tendencia* de la serie, y **de forma**, son secuencias que difieren en *forma* a la del total de la serie. Para profundizar en esta distinción, se ha elaborado la Figura 2.1. En esta gráfica se muestra una serie temporal con los tipos de anomalía explicados anteriormente. Se observa que las anomalías puntuales son mediciones que se alejan del resto de observaciones. Por otro lado, las anomalías colectivas se encuentran dentro del rango normal pero tienen un valor constante, lo que podría significar que el sensor no funciona correctamente y registra siempre el mismo valor. Por último, las anomalías contextuales también se encuentran dentro del rango normal, pero para nuestro contexto en el que se está impartiendo una clase en ese momento, los valores son demasiado bajos, indicando que probablemente exista algún problema.

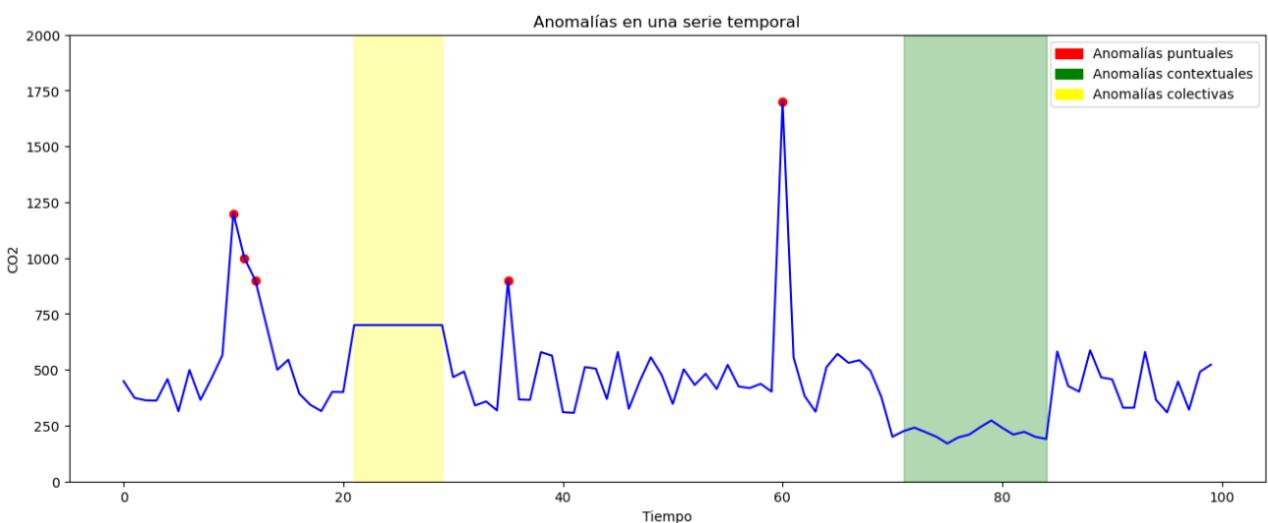


Figura 2.1: Tipos de anomalía en una serie temporal de CO₂.

2.2. Trabajos relacionados

En esta sección empezaremos por analizar distintos métodos para realizar la detección de anomalías, explicando varios modelos y en qué situaciones se comportan mejor. Tras esto, veremos distintas propuestas para el mecanismo encargado de seleccionar el mejor modelo de detección en tiempo real.

2.2.1. Modelos de detección de anomalías

En [11] y [2] se realiza una clasificación de técnicas de *deep learning* para detección de anomalías. Nos centraremos en destacar las soluciones para datos univariados³ y para series temporales, ya que son las que se ajustan a la definición y alcance del proyecto. En [2] se realiza una breve descripción de aspectos de los modelos que se deben tener en cuenta antes de la clasificación. El primero es el **esquema de aprendizaje**. Actualmente existen tres esquemas para los problemas de detección de anomalías en función de si contamos con datos etiquetados. El **aprendizaje supervisado** necesita ser alimentado con datos etiquetados. Esto puede no ser posible en escenarios reales debido a la falta de estas marcas que identifiquen cada medición del sensor como anomalía. El **aprendizaje no supervisado** no tiene este problema al no necesitar datos etiquetados. La mayoría de soluciones siguen este esquema por la naturaleza de los datos de sensores. Un punto intermedio entre estas dos alternativas es el **aprendizaje semisupervisado**, en el que se utilizan datos *normales*⁴ para la etapa de entrenamiento. Esto hace posible que el modelo *aprenda* la distribución de los datos normales y detecte aquellos que se desvén de esta distribución. Otro aspecto es la **entrada** que recibe el modelo. Los modelos para detección de anomalías en series temporales pueden recibir tanto **puntos individuales** como **ventanas temporales**. Estas ventanas pueden estar ordenadas o mezcladas dependiendo de la aplicación y pueden usarse para comparar secuencias de observaciones en lugar de puntos. Esto va de la mano con la definición de anomalía, ya que detectar **anomalías colectivas** puede ser tarea difícil si tratamos cada medición del sensor de manera individual. Con estos puntos aclarados podemos realizar la clasificación de los modelos de *deep learning*. El artículo realiza la clasificación en dos grupos:

- **Modelos basados en pronósticos:** Estas soluciones buscan predecir una observación o secuencia de observaciones futuras. La detección de anomalías se realiza comparando las diferencias entre estas predicciones y los datos reales medidos.
- **Modelos basados en reconstrucción:** Estos modelos codifican las secuencias de datos

³Los datos univariados son aquellos que se refieren a una sola variable o característica, en este caso el CO₂.

⁴Se consideran datos normales aquellos en que la existencia de anomalías es nula o aproximadamente nula.

de entrada en *espacios latentes*⁵. Las reconstrucciones de los datos se comparan con los datos medidos y se calcula el error de reconstrucción. Este error se utiliza para clasificar la entrada como anomalía o dato normal.

En la Tabla 2.1 podemos ver ejemplos de modelos que siguen esta clasificación.

Clasificación	Modelo	Esquema	Entrada	Anomalía
Pronóstico	LSTM [13]	Semisupervisado	Punto	Colectiva
	LSTM-AD [14]	No supervisado	Punto	Puntual
Reconstrucción	LSTM-VAE [15]	Semisupervisado	Ventana	Colectiva
	LSTM-AE [16]	Semisupervisado	Ventana	Puntual

Tabla 2.1: Modelos de *deep learning* para detección de anomalías en series temporales univariadas. Adaptado de [2].

En el artículo [15] se utiliza una de las soluciones mencionadas. En concreto, se implementa un **LSTM-VAE**. Este nombre hace referencia a dos arquitecturas de redes neuronales, la *Long Short-Term Memory* (LSTM) [17] y el *Variational Autoencoder* (VAE) [18]. Las LSTM son *Redes Neuronales Recurrentes* (RNN) que resuelven el problema de la desaparición del gradiente⁶ que se presenta en las RNN tradicionales. Son útiles para el procesamiento de secuencias de datos a largo plazo, lo que se ajusta a nuestro proyecto en particular. Los VAE son una variante de los *autoencoders* (AE) en los que el *espacio latente* se representa mediante una distribución de probabilidad. En este artículo se alimenta el VAE con ventanas temporales consecutivas, por ejemplo para el instante t tenemos la ventana $w_t = [x_0, x_1, x_2, \dots, x_t]$, donde x es un punto de la serie temporal. Estos datos de entrada deben ser normales durante la fase de entrenamiento buscando minimizar el error de predicción. Para detectar una ventana temporal anómala, se compara la ventana real con la reconstruida por el VAE. Si esta diferencia supera el umbral de error, se identifica como anómala. Este umbral se define utilizando un conjunto de datos de validación que contenga datos normales y anómalos y la métrica que mejor desempeño tenga en el problema en cuestión. Por ejemplo, **F1-score**.

En [16] utilizan un **LSTM-AE** que sigue una arquitectura y funcionamiento similares al artículo anterior [15], pero en este caso se utiliza un AE en lugar de un VAE. La principal diferencia es que en un AE se codifica la entrada en el *espacio latente* y no su distribución probabilística. Este modelo tienen una jerarquía de LSTM tanto en el *encoder* como en el *decoder*. Cada observación de la ventana de entrada es procesada por una LSTM y, a su vez,

⁵Los espacios latentes son representaciones de menor dimensionalidad de los datos de entrada.

⁶La desaparición del gradiente se refiere al error en la actualización de los pesos de las redes neuronales cuando los pesos se vuelven demasiado pequeños.

cada LSTM decide si quedarse o descartar la salida de la anterior de una manera secuencial. Esto se hace con la finalidad de plegar varias secuencias temporales y obtener las características codificadas. El *decodificador* procesará esto y se obtendrá la reconstrucción de las ventanas de entrada. Para calcular el umbral del error de reconstrucción, se considera el mayor error obtenido con los datos normales de validación tras el entrenamiento.

En ambas implementaciones [15][16] se realiza un preprocesado de los datos que consiste en construir los datasets de entrenamiento y de test. Primero se limpian los datos eliminando posibles registros duplicados y valores nulos. Después se dividen los dos datasets. Para el de entrenamiento se eliminan los registros que contienen anomalías, debido al carácter semisupervisado mencionado con anterioridad. En el de test se añade una etiqueta indicando si el registro es una anomalía, esto se hace con la finalidad de medir el rendimiento del modelo. Las métricas utilizadas para medir este desempeño han sido:

- **Exactitud:** Es el ratio entre las predicciones correctas (suma de *verdaderos positivos* (TP) y *verdaderos negativos* (TN)) y las predicciones totales.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

- **Precisión:** Es una métrica utilizada para saber qué porcentaje de observaciones se han clasificado como anomalías siendo realmente anomalías.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.2)$$

- **Recall:** Es el ratio de TP. Se utiliza para saber cuantas anomalías (reales) se clasifican correctamente.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.3)$$

- **F1-score:** Esta métrica es la media armónica de las medidas de *precisión* y *recall*. Es útil para evaluar nuestro modelo buscando maximizar ambas métricas a la vez.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.4)$$

- **AUC-ROC:** Esta métrica calcula el *área bajo la curva* (AUC) que refleja la relación entre la tasa de TP y la de *falsos positivos* (FP). Es una buena herramienta para medir el acierto en la predicción de eventos binarios [19], como pueden ser las anomalías.

$$\text{AUC-ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \quad (2.5)$$

2.2.2. Framework de selección de modelos

Los dispositivos IoT tienen unas especificaciones de recursos limitadas que dificultan el despliegue de modelos de cierta complejidad. En [20] se presenta un framework para automatizar la tarea de evaluar el modelo desarrollado y generar código específico para la plataforma IoT en cuestión⁷. Además se hace un análisis de los aspectos que se deben considerar a la hora de estimar los requisitos de memoria de un modelo. Afirma que considerar sólo el tamaño del modelo en si es demasiado optimista y que se deben tener en cuenta las dependencias externas de librerías para las funciones de activación y el flujo de datos. Estas limitaciones provocan que los modelos se tengan que ajustar, pudiendo producir un deterioro en su rendimiento. En [21] se propone un método de selección de modelos en tiempo real para mitigar este problema. La idea consiste en implementar una jerarquía de modelos en la que el más complejo, y por tanto el que más recursos necesita, se aloja fuera del dispositivo, típicamente en alguna solución *cloud*. El modelo con menos complejidad se encontrará en el propio dispositivo y entre estos dos pueden implementarse el número de capas que se deseen. El mecanismo encargado de seleccionar el mejor modelo en función del dato de entrada es un algoritmo de aprendizaje por refuerzo que busca maximizar la recompensa esperada del modelo a seleccionar. Esta recompensa se calcula mediante la **exactitud** de detectar la entrada menos el coste de emplear un modelo en particular. Este coste viene dado por la latencia de la capa en la que se encuentra el modelo. De esta manera se penaliza el hecho de usar las capas más lentas aunque tengan un mejor comportamiento, buscando un equilibrio entre rendimiento e instantaneidad de la respuesta.

Existen otros métodos utilizados para este fin. En [22] se utiliza una secuencia de modelos basados en *K-Nearest Neighbors* (KNN) con tantos KNN como modelos de detección. El primer algoritmo decide si se debe utilizar el primer modelo, si decide que no se pasa al siguiente y se continúa hasta el final. En [23] se vuelve a utilizar un agente de aprendizaje por refuerzo, en concreto un algoritmo *Q-learning*.

2.3. Conclusiones

Esta revisión del estado del arte del ámbito del proyecto permite tener una visión global de las metodologías y tecnologías que se están utilizando en la actualidad para abordar el problema de la detección de anomalías en sensores de CO₂. Como resumen, se extraen una serie puntos a tener en cuenta en futuras fases del *Trabajo Final de Máster*:

- La importancia de **definir** el concepto de **anomalía**. Hemos visto que existen distintas interpretaciones de esta palabra que hacen que el enfoque cambie completamente. Existen

⁷<https://github.com/BenSliwa/LIMITS>

modelos que funcionan mejor para un tipo de anomalías que para otros, por lo que la implementación de este proyecto depende totalmente de este factor.

- La **preparación** de los **datos de entrada** es una tarea muy importante y depende del esquema que utilicemos. Si utilizamos un algoritmo de aprendizaje supervisado necesitaremos tener datos correctamente etiquetados, mientras que si utilizamos otro esquema puede que necesitemos limpiar de anomalías nuestro conjunto de entrenamiento.
- Una práctica que ofrece buenos resultados es el uso de celdas **LSTM**. Estas redes capturan dependencias a largo plazo entre los elementos de la serie temporal producida por los sensores de CO₂. Por otro lado, las técnicas basadas en **reconstrucción** que utilizan **autoencoders** obtienen muy buenos resultados de rendimiento.
- Los dispositivos IoT tienen **recursos limitados**, por lo que se debe analizar correctamente el hardware en el que va a ser desplegado el modelo para adaptarse a él.
- Los métodos de **selección de modelos** examinados **penalizan la latencia** de la respuesta. Es necesario establecer un criterio para nuestro problema que indique el grado de perjuicio que representa el retardo en la clasificación de una anomalía de CO₂.

Capítulo 3

Análisis y diseño

En este capítulo, se lleva a cabo el análisis de las fuentes de datos empleadas tanto para el entrenamiento como para la evaluación de los modelos. Además, se describen las herramientas y tecnologías utilizadas durante el desarrollo, y se establece la arquitectura del sistema que se va a desarrollar.

3.1. Fuentes de datos

La selección y preparación de las fuentes de datos es una fase crítica en el desarrollo del proyecto, ya que son fundamentales para la construcción y validación de los modelos, pudiendo influir significativamente en los resultados obtenidos si no se lleva a cabo correctamente. En este apartado se describirán las fuentes seleccionadas, incluyendo su origen, calidad y una explicación de por qué se ajustan a este proyecto. En el siguiente capítulo se realizará el preprocesamiento, limpieza y análisis de estos datos.

Los requisitos del trabajo marcan las características de los datos necesarias. En este caso, se buscan mediciones de CO₂ en interiores que deben ser continuas y a lo largo del tiempo. Para utilizar modelos que aprovechen la secuencialidad de los datos, como las **LSTM**, necesitamos un volumen de datos considerable. Además, sería interesante contar con datos de más de un sensor para poder validar los modelos. Para este proyecto no se necesitan otros atributos ya que el objetivo es que funcione en un sensor de CO₂ y no necesite de otro tipo de medidores.

Con estas premisas se han revisado distintos repositorios como *Kaggle*¹ o *Zenodo*². De las fuentes analizadas se ha seleccionado un dataset con datos de CO₂ de distintas aulas de dos escuelas españolas [24]. En concreto, los colegios se ubican en las localidades de **Villafamés** y **Vall d'Alba** en **Castellón, España**. La Figura 3.1 muestra la ubicación de estas localidades

¹<https://www.kaggle.com/>

²<https://zenodo.org/>

en relación con el resto del país.

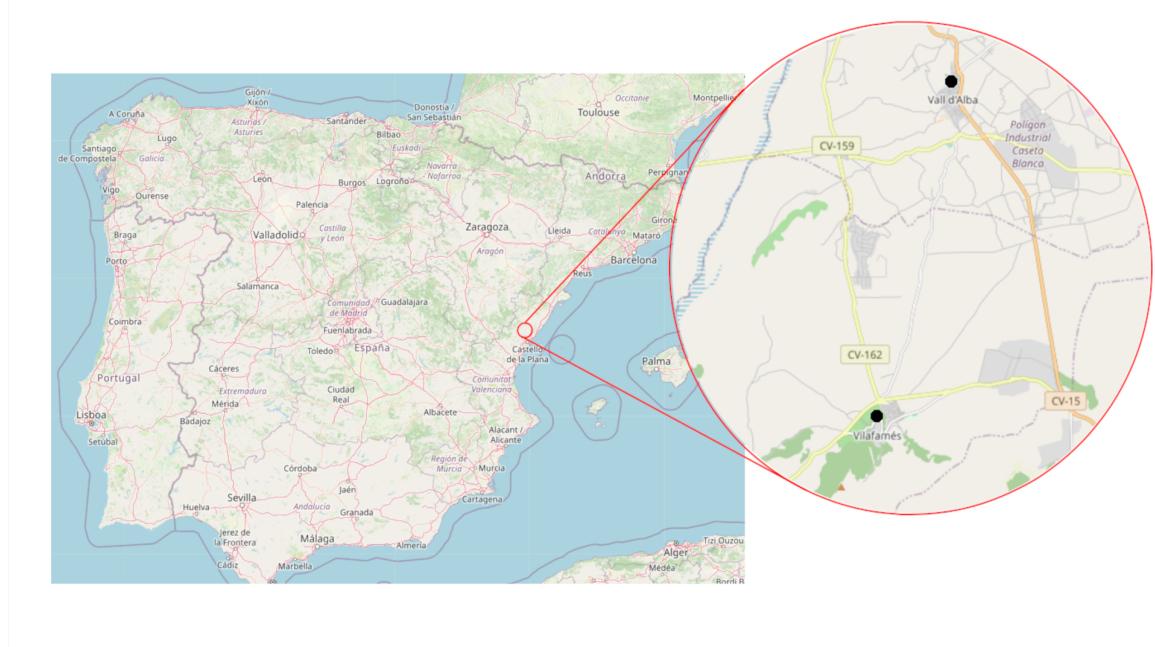


Figura 3.1: Mapa con la ubicación de los colegios en los que se desplegaron los sensores.

En la Tabla 3.1 se puede comprobar que además de los datos de CO₂ tenemos otros atributos como el sensor que ha realizado la medición, campos relativos a la fecha de la medición, e incluso, datos de temperatura y humedad. Se tienen 80,000 registros por lo que el volumen de datos es suficiente y al contar con distintas localizaciones y sensores podemos preparar correctamente nuestros conjuntos de entrenamiento, validación y test.

Por todo esto, se considera suficiente usar esta única fuente de datos para el proyecto y no existe la necesidad de integrar otras propuestas.

published_at	date_time	date	month	w.day	time	hour	min	sec	temp	hum	co2	bat	sensor_id
2021-06-01T14:30:06.735Z	2021-06-01T14:30:06	2021-06-01	Jun	Tue	14:30:06	14	30	6	29.84	46.33	390	85.19	CO2_01
2021-06-01T14:15:06.784Z	2021-06-01T14:15:06	2021-06-01	Jun	Tue	14:15:06	14	15	6	29.82	47.53	391	81.14	CO2_01
2021-06-01T14:20:06.733Z	2021-06-01T14:20:06	2021-06-01	Jun	Tue	14:20:06	14	20	6	29.83	47.24	391	82.49	CO2_01
2021-06-01T14:35:06.728Z	2021-06-01T14:35:06	2021-06-01	Jun	Tue	14:35:06	14	35	6	29.87	46.07	393	86.54	CO2_01
2021-06-01T14:40:06.729Z	2021-06-01T14:40:06	2021-06-01	Jun	Tue	14:40:06	14	40	6	29.96	45.92	395	87.90	CO2_01
2021-06-01T14:10:06.758Z	2021-06-01T14:10:06	2021-06-01	Jun	Tue	14:40:06	14	10	6	29.75	47.73	399	79.69	CO2_01

Tabla 3.1: Datos de los sensores de las escuelas

3.2. Tecnologías y herramientas

En esta sección se hace un desglose de los medios y utilidades utilizadas para la realización del *Trabajo de Fin de Máster*, en concreto de las fases de desarrollo y despliegue del proyecto.

3.2.1. Lenguajes de programación

- **Python:** Es un lenguaje de programación interpretado, de alto nivel y multiparadigma. Se utiliza la versión 3.7.12.
- **C++:** Es un lenguaje de programación de propósito general que se utiliza ampliamente, entre otras muchas aplicaciones, en dispositivos con recursos limitados por su eficiencia.
- **Javascript:** Es un lenguaje de programación interpretado, orientado a objetos y de alto nivel que se utiliza principalmente en el desarrollo web.

3.2.2. Bibliotecas

- **Pandas:** Es una biblioteca de código abierto para análisis de datos en Python que proporciona estructuras de datos flexibles y de alto rendimiento para trabajar con datos [25].
- **Matplotlib:** Biblioteca de visualización de datos en Python, que permite generar distintos gráficos y visualizaciones [26].
- **Seaborn:** Otra biblioteca de visualización que se basa en Matplotlib [27].
- **Scipy:** Proporciona herramientas matemáticas en Python [28].
- **Scikit-learn:** Incluye funcionalidades de aprendizaje automático y preprocesamiento de datos en Python [29].
- **MicroMLGen:** Permite exportar modelos de sklearn a lenguaje C++ de una manera automática [30].
- **Tensorflow:** En conjunto con Keras, permite la creación y entrenamiento de redes neuronales en Python [31].
- **Optuna:** Biblioteca que facilita la búsqueda y análisis de los mejores hiperparámetros para los modelos creados [32].

3.2.3. Frameworks

- **FastAPI:** Framework web moderno y rápido para construir APIs en Python [33].
- **ESP-IDF:** Facilita el desarrollo de sistemas embebidos basados en microcontroladores Espressif [34].

3.2.4. Otras herramientas

- **Miniconda:** Gestor de paquetes y entorno de conda de tamaño reducido, que permite la instalación y gestión de paquetes y dependencias de manera eficiente en Python [35].
- **kaggle:** Plataforma en línea que ofrece un servicio para ejecutar Jupyter Notebooks en un entorno con GPU.
- **Visual Studio Code:** Editor de código abierto que facilita el trabajo con ficheros de distintos tipos y lenguajes de programación.³
- **Jupyter Notebook:** Aplicación web de código abierto que permite crear documentos que contienen código, texto y visualizaciones.⁴

3.3. Hardware

Para desplegar el sistema elaborado se ha utilizado un PC para albergar el servidor, que podría estar en un servidor local o incluso en algún servicio *cloud* y un microcontrolador que podría estar integrado en un sensor de CO₂.

- **PC:** para el desarrollo del proyecto se ha utilizado un ordenador portátil con Windows 11 Home y las siguientes características:
 - CPU Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz (8 CPUs), 2.30 GHz.
 - Memoria RAM de 16,0 GB.
 - SSD INTEL SSDPEKNW512G8 de 512 GB.
 - Tarjeta gráfica NVIDIA GeForce MX250.
- **Microcontrolador:** se ha utilizado la placa de desarrollo **ESP32-S3-DevKitC-1-N32R8V** que incluye el microcontrolador (MCU) **ESP32-S3-WROOM-2-N32R8V**. Este MCU contiene un módulo Wi-Fi y Bluetooth que permiten conectarlo con otros dispositivos de una manera sencilla. Las especificaciones pueden encontrarse en: [ESP32-S3-DevKitC-1](https://www.espressif.com/en/products/esp32-s3-devkitc-1-n32r8v).

³<https://code.visualstudio.com/>

⁴<https://jupyter.org/>

3.4. Diagramas del sistema

En esta sección se proporcionan una serie de diagramas que facilitan la comprensión del sistema desarrollado y proporcionan una visión general de su funcionamiento.

En la Figura 3.2 se muestra la **arquitectura del sistema**. Esta plataforma se compone de un **sensor** que tiene un **MCU** capaz de realizar la tarea de detección de anomalías o de enviar los datos del sensor a un **servidor** alojado en *Internet* o en la red local. Este sistema de medición puede estar conectado a una serie de elementos como una **pantalla** o un mecanismo de **alarmas** que avise al usuario de que se ha producido una anomalía. También es capaz de enviar los datos al servidor para que sea este el que realice la tarea de detección y los usuarios puedan visualizar los resultados en distintos dispositivos.

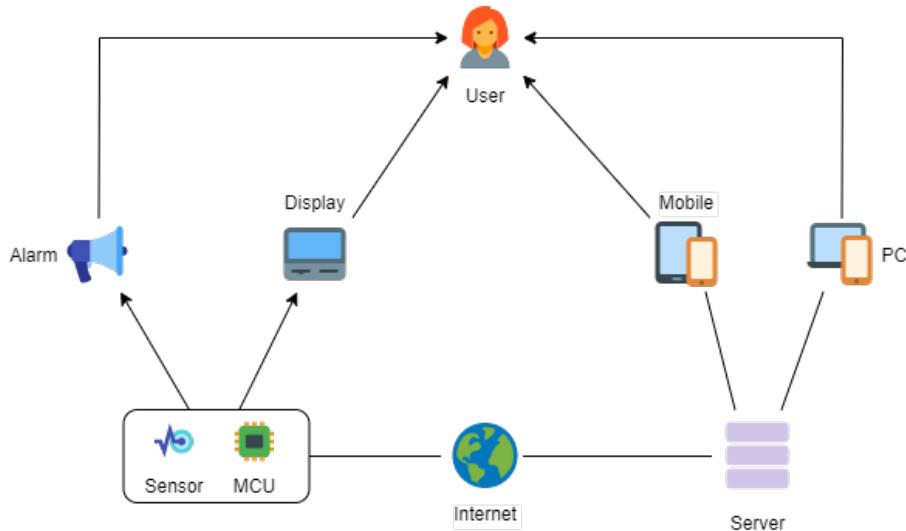


Figura 3.2: Arquitectura del sistema.

De este diagrama se puede ver que tenemos dos modos de funcionamiento. El primero se puede ver en la Figura 3.3 y lo llamaremos **modo Cloud**. En este modo el MCU pregunta al servidor en qué modo debe trabajar, si está configurado en modo Cloud el MCU envía los datos del sensor al servidor para que un modelo realice la detección de anomalías. Una vez que se ha procesado la secuencia de datos, el servidor informa a los **clientes WEB** conectados para que dibujen los nuevos puntos en la gráfica temporal indicando cuáles son anomalías.

En el caso de que el servidor no responda o el modo configurado sea el **modo Standalone** se llevará a cabo la secuencia mostrada en la Figura 3.4. En este caso el MCU utilizará un modelo embebido para realizar la detección y enviará los resultados al servidor, en caso de que esté disponible, e informará a los clientes que tenga conectados en la red local, como pantallas o alarmas.

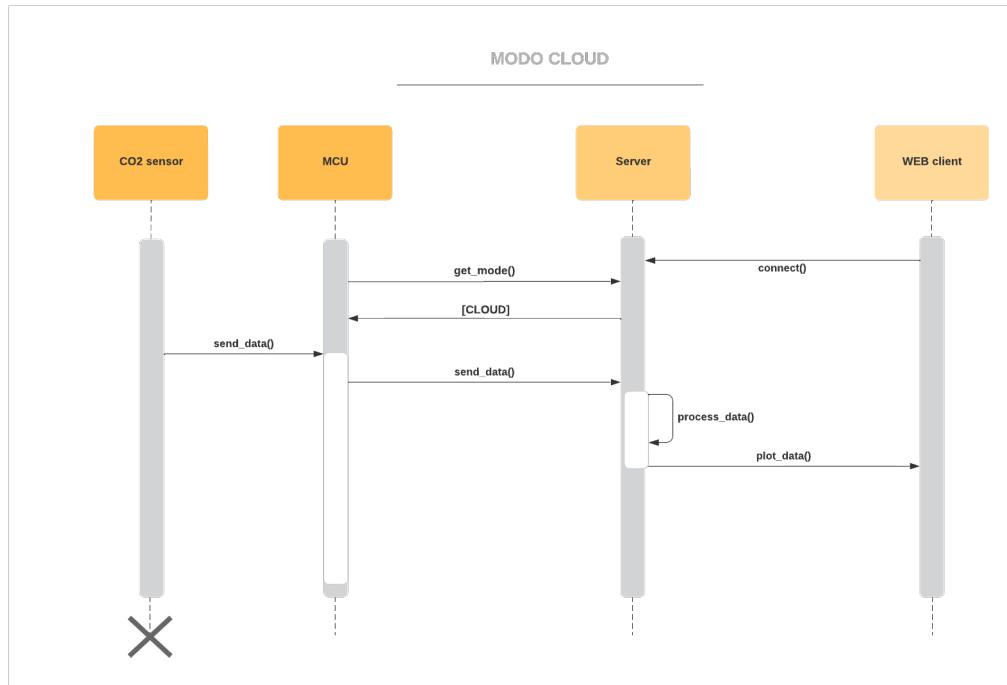


Figura 3.3: Diagrama de secuencia del modo Cloud.

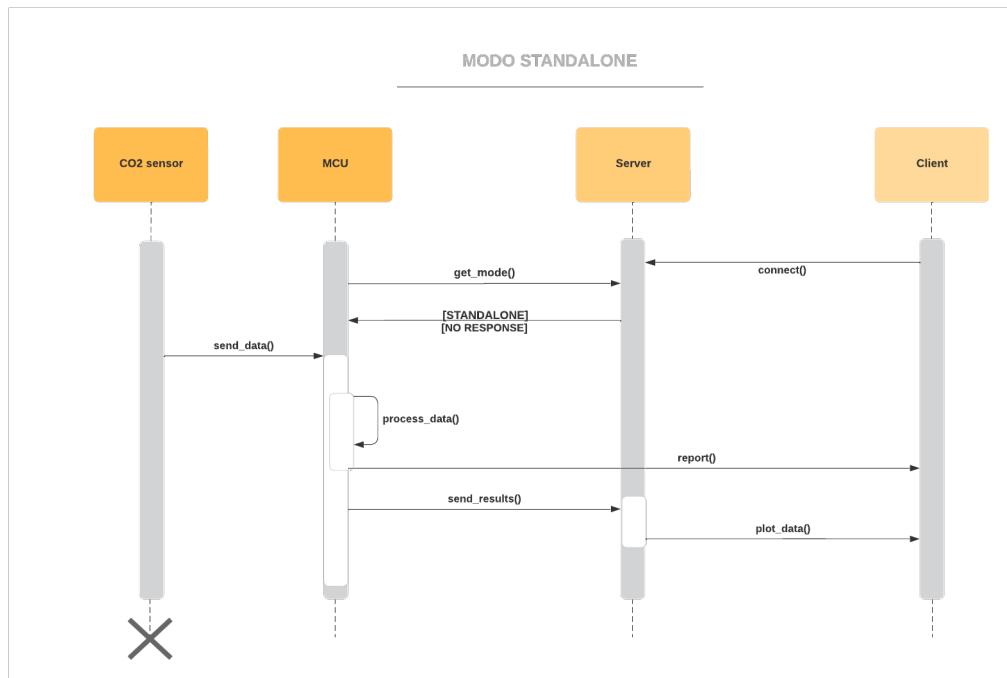


Figura 3.4: Diagrama de secuencia del modo Standalone.

Capítulo 4

Procesado y análisis del dataset

En este capítulo se abordan todas las etapas previas a la creación de los modelos relacionadas con los datos. Se lleva a cabo la exploración, limpieza y análisis de los mismos. Todo este proceso se encuentra documentado en la notebook llamada *models* en el repositorio de **GitHub**, accesible en el siguiente enlace: [models](#).

4.1. Exploración inicial

Nuestro dataset está formado por dos conjuntos de datos que se corresponden cada uno con una escuela de primaria de Castellón, España. En la Tabla 4.1 y en la Tabla 4.2 podemos ver los primeros registros de cada conjunto. A simple vista, se percibe que cuentan con los mismos 14 atributos, a excepción del atributo **time** que solo está disponible en los datos de **CEIP L’Albea**.

published_at	date_time	date	month	w_day	time	hour	min	sec	temp	hum	co2	bat	sensor_id
2021-06-01T14:30:06.735Z	2021-06-01T14:30:06	2021-06-01	Jun	Tue	14:30:06	14	30	6	29.84	46.33	390	85.19	CO2_01
2021-06-01T14:15:06.784Z	2021-06-01T14:15:06	2021-06-01	Jun	Tue	14:15:06	14	15	6	29.82	47.53	391	81.14	CO2_01
2021-06-01T14:20:06.733Z	2021-06-01T14:20:06	2021-06-01	Jun	Tue	14:20:06	14	20	6	29.83	47.24	391	82.49	CO2_01
2021-06-01T14:35:06.728Z	2021-06-01T14:35:06	2021-06-01	Jun	Tue	14:35:06	14	35	6	29.87	46.07	393	86.54	CO2_01
2021-06-01T14:40:06.729Z	2021-06-01T14:40:06	2021-06-01	Jun	Tue	14:40:06	14	40	6	29.96	45.92	395	87.90	CO2_01

Tabla 4.1: Datos de los sensores de CEIP L’Albea.

Observando estos datos y las descripciones recogidas en la Tabla 4.3 podemos extraer las siguientes conclusiones para futuros apartados:

published_at	date_time	date	month	w_day	hour	mint	sec	temp	hum	co2	bat	sensor_id
2021-05-03T07:00:05.241Z	2021-05-03T07:00:05	5/3/2021	May	Mon	7	0	5	26.551460	39.382935	302	99.55	CO2_06
2021-05-03T07:05:05.348Z	2021-05-03T07:05:05	5/3/2021	May	Mon	7	5	5	26.479362	39.572144	301	99.55	CO2_06
2021-05-03T07:20:05.162Z	2021-05-03T07:20:05	5/3/2021	May	Mon	7	20	5	26.578163	40.089417	376	99.55	CO2_06
2021-05-03T07:25:05.047Z	2021-05-03T07:25:05	5/3/2021	May	Mon	7	25	5	26.749062	40.321350	520	99.55	CO2_06
2021-05-03T07:35:04.554Z	2021-05-03T07:35:04	5/3/2021	May	Mon	7	35	5	27.160294	39.421082	393	99.55	CO2_06

Tabla 4.2: Datos de los sensores de CEIP Sant Miquel.

- El campo *date* tiene formatos distintos en ambos datasets, se debe tener en cuenta para interpretar correctamente las fechas.
- Los campos *date*, *time*, *hour*, *mint* y *sec* se pueden obtener del campo *date_time* por lo que a priori no ofrecen nada nuevo.
- La fecha de publicación no es relevante para nuestro análisis.
- Las mediciones de humedad y temperatura pueden ser interesantes para el análisis de los datos pero no se utilizarán para el entrenamiento de los modelos, ya que se busca que se pueda utilizar simplemente con un sensor de CO2.

4.2. Limpieza de datos

Tras realizar la primera aproximación a nuestros datos, se procede a prepararlos para su análisis. La limpieza de datos es una fase clave ya que sin ella los resultados del análisis pueden estar sesgados o ser inexactos. En esta fase se realizan diversas tareas, como la eliminación de valores nulos, la detección y eliminación de duplicados, la corrección de errores tipográficos y la eliminación de outliers si procede, entre otras.

Lo primero que hacemos es añadir una nueva columna **location** a nuestros conjuntos que indica la escuela en la que se ha realizado la medición (*albea* o *santmiquel*). Esto es importante ya que nuestros modelos deben ser entrenados siguiendo una coherencia temporal y espacial entre los datos. Tras unir los conjuntos, nos quedamos con los atributos que nos aportan cierta información para nuestro análisis. En la Tabla 4.4 podemos ver los atributos con los que nos quedamos.

Observando los sensores con los que contamos (Figura 4.1) el uso de la nueva columna **location** cobra mayor importancia, ya que no es posible identificar la escuela por el nombre del

Atributo	Descripción
published_at	Fecha y hora con precisión de milisegundos, en formato ISO 8601 [36] en la que el sensor informó de la medición
date_time	Marca de tiempo con el formato año-mes-día hora:minutos:segundos en formato ISO 8601, en la que se realizó la medición
date	Fecha en el formato mes/día/año o año-mes-día en la que se realizó la medición
month	Mes en el que se tomó el dato, expresado en texto
w_day	Día de la semana de la medición, expresado en texto
time	Marca de tiempo con formato hh:mm:ss que indica la hora exacta de la medición.
hour	Variable numérica que indica la hora del día en la que se hizo la medición, en formato de 24 horas
min	Variable numérica que indica los minutos de la hora en la que se hizo la medición
sec	Variable numérica que indica los segundos de la hora en la que se hizo la medición
temp	Temperatura registrada en grados Celsius
hum	Humedad relativa registrada en porcentaje
co2	Concentración de dióxido de carbono medida en partes por millón (ppm)
bat	Nivel de batería restante del sensor expresado en porcentaje
sensor_id	Identificador único del sensor que generó los datos

Tabla 4.3: Lista de atributos del dataset inicial.

date_time	co2	sensor_id	month	w_day	hour	location
2021-05-03T07:00:05	302	CO2_06	May	Mon	7	santmiquel
2021-05-03T07:05:05	301	CO2_06	May	Mon	7	santmiquel
2021-05-03T07:20:05	376	CO2_06	May	Mon	7	santmiquel
2021-05-03T07:25:05	520	CO2_06	May	Mon	7	santmiquel
2021-05-03T07:35:04	393	CO2_06	May	Mon	7	santmiquel

Tabla 4.4: Dataset final.

sensor. Además, se percibe que los textos tienen espacios al principio por lo que es necesario recortarlos para eliminar los espacios en blanco.

```
albea      [ CO2_01, CO2_02, CO2_03, CO2_04, CO2_05, CO2_06]
santmiquel [ CO2_06, CO2_04, CO2_01, CO2_03, CO2_02, CO2_05]
```

Figura 4.1: Nombres de los sensores de cada escuela.

El siguiente paso es eliminar los registros duplicados. Debido a que los sensores de distintas ubicaciones comparten nombre, es necesario analizar cuidadosamente el criterio a seguir para realizar este proceso. En este caso, se eliminan los registros repetidos que comparten *date_time*, *sensor_id* y *location*, manteniendo el primer registro encontrado. Existen un total de **582 registros** que se eliminan del dataset. La búsqueda de **valores nulos** nos indica que no existen valores nulos ni vacíos en ninguna de nuestras variables. También se comprueba que no existan valores de CO2 iguales a 0 ya que podrían ser falta o errores de medición. El tratamiento de los *outliers* también es una fase crítica de este proyecto. Esto se debe a que el objetivo es encontrar anomalías en las series temporales de CO2 y los *outliers* son un tipo de anomalías. Debido a esto, en este punto se buscan valores que puedan deberse a errores de medición, como valores extremadamente elevados, que no tendrían sentido que se produjesen en nuestro entorno. En los artículos [37], [38] y [39] se analiza la calidad del aire en escuelas y otros espacios cerrados y se observa que los valores de CO2 se encuentran entre la concentración de fondo normal en el aire exterior (entre 250 y 400 ppm) y un ambiente cargado (que llega a los 3000 ppm y en alguna ocasión a los 4000 ppm). En la Figura 4.2 vemos que los valores de CO2 de nuestras escuelas se encuentran en el rango [250, 2000] ppm, por lo que asumimos que todos los valores son realistas.

Por último, con nuestros datos procesados tras la limpieza, los ordenamos por las variables *location*, *sensor_id* y *date_time* para tener secuencias temporales de cada sensor.

4.3. Análisis de datos

En este apartado utilizaremos el dataset creado para extraer conclusiones que nos ayuden a comprender mejor los patrones, tendencias y relaciones presentes en los datos. En primer lugar, realizaremos un análisis descriptivo para obtener una visión general de las características y propiedades de los atributos. Esto incluye calcular medidas de tendencia central, como la media, mediana y moda, así como medidas de dispersión, como la desviación estándar y el rango. En la Tabla 4.5 podemos ver este análisis del que sacamos las siguientes conclusiones:

- Tenemos un total de 72,877 registros, lo que indica que contamos con una cantidad significativa de datos para nuestro análisis. Esta muestra amplia nos brinda una buena

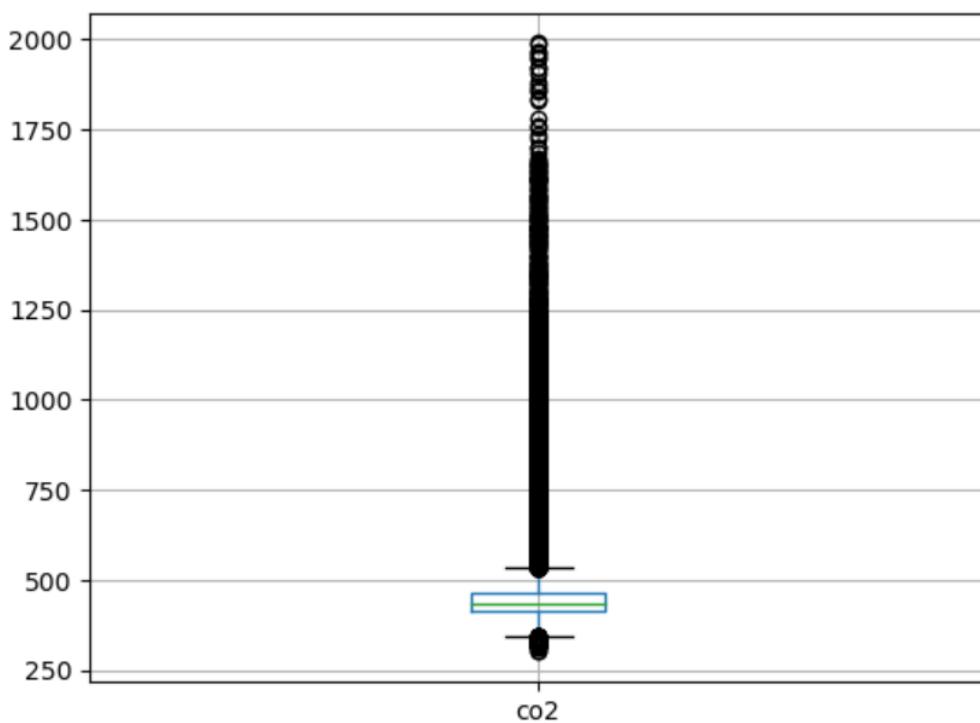


Figura 4.2: Boxplot de la variable CO2.

representación de las mediciones de CO2 realizadas en el periodo de tiempo considerado.

- Los percentiles nos proporcionan información sobre la distribución de los niveles de CO2. El percentil 25 (25 %) se encuentra en 416 ppm, lo que significa que el 25 % de las mediciones de CO2 son inferiores a este valor. Por otro lado, el percentil 75 (75 %) se encuentra en 464 ppm, lo que indica que el 75 % de las mediciones no superan este valor. Esto sugiere que la mayoría de las mediciones de CO2 se encuentran en un rango relativamente estrecho.
- Se observa que la desviación estándar de 107.28 ppm es más del doble del rango intercuartil de 48 ppm. Esto sugiere que existe una variabilidad significativa en los niveles de CO2 que se puede deber a factores como la ocupación y la ventilación.
- Nuestro dataset abarca mediciones tomadas desde el 3 de Mayo de 2021 al 23 de Junio de 2021, lo que nos proporciona datos de dos meses distintos. Esta diversidad temporal nos permitirá analizar posibles variaciones estacionales en los niveles de CO2 y comprender mejor su comportamiento a lo largo del tiempo.
- En cuanto a los días de la semana, encontramos datos de los 7 días, lo cual es interesante ya que nos brinda una visión completa de los niveles de CO2 en diferentes momentos

de la semana. Es relevante destacar que estos datos incluyen días laborables y días no laborables, lo que podría influir en las mediciones, especialmente considerando que las escuelas están vacías durante los fines de semana y días festivos.

- Es importante mencionar que las mediciones se realizan las 24 horas del día, lo que nos permite capturar la variabilidad de los niveles de CO₂ a lo largo de todo el día. Esta información es crucial para comprender los patrones de concentración de CO₂ en el ambiente interior y evaluar su relación con factores como la ocupación.
- Como ya se mencionó anteriormente, disponemos de 6 sensores distintos ubicados en dos escuelas, lo que nos brinda un total de 14 sensores de CO₂ distintos. Esta distribución de sensores nos permite realizar un análisis detallado de las mediciones en diferentes ubicaciones y evaluar posibles variaciones espaciales en los niveles de CO₂.

	date_time	co2	sensor_id	month	w_day	hour	location
count	72877	72877.000000	72877	72877	72877	72877.000000	72877
unique	14685	NaN	6	2	7	NaN	2
top	2021-05-25 16:15:00	NaN	CO2_02	May	Fri	NaN	santmiquel
freq	6	NaN	12948	38477	10920	NaN	38477
first	2021-05-03 07:00:00	NaN	NaN	NaN	NaN	NaN	NaN
last	2021-06-23 23:55:00	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	463.413697	NaN	NaN	NaN	11.577288	NaN
std	NaN	107.279452	NaN	NaN	NaN	6.913305	NaN
min	NaN	301.000000	NaN	NaN	NaN	0.000000	NaN
25 %	NaN	416.000000	NaN	NaN	NaN	6.000000	NaN
50 %	NaN	434.000000	NaN	NaN	NaN	12.000000	NaN
75 %	NaN	464.000000	NaN	NaN	NaN	18.000000	NaN
max	NaN	1989.000000	NaN	NaN	NaN	23.000000	NaN

Tabla 4.5: Análisis descriptivo del dataset.

Es evidente que la distribución de las mediciones por sensor en el dataset es un factor importante a considerar. Un dataset balanceado, en términos de la cantidad de mediciones por sensor, es crucial para obtener conclusiones sólidas y representativas. En la Figura 4.3, se muestra un gráfico circular que representa la proporción de datos de cada sensor en el conjunto de datos total. Según esta representación visual, se observa que la cantidad de datos de cada sensor oscila entre el 5.8 % y el 10.3 % del total. Este hallazgo indica que la distribución de las

mediciones por sensor es relativamente equilibrada, lo cual es una buena señal. Si la cantidad de datos de cada sensor estuviera muy desequilibrada, es decir, si algunos sensores tuvieran una cantidad significativamente mayor o menor de mediciones en comparación con otros, esto podría sesgar los análisis y las conclusiones obtenidas, además de perjudicar la generalización de nuestros modelos. Al contar con una distribución más uniforme de los datos entre los sensores, podemos tener mayor confianza en que las conclusiones extraídas de nuestro análisis reflejarán de manera más precisa y equitativa el comportamiento de los diferentes sensores en el entorno de estudio.

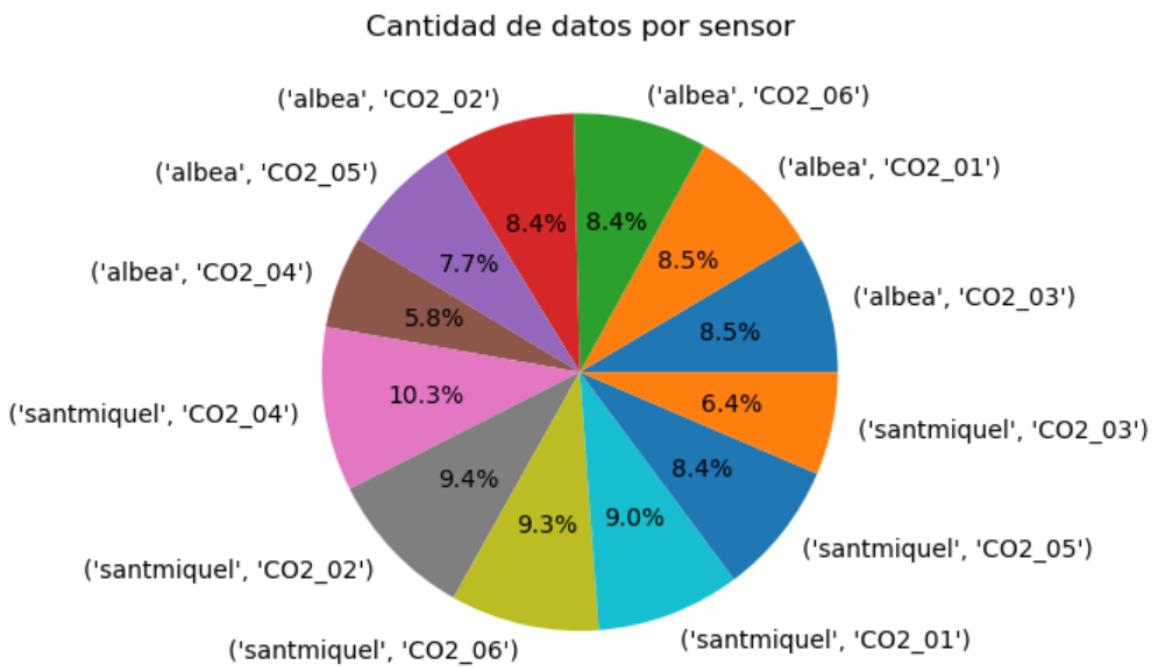


Figura 4.3: Gráfico circular con la distribución de los sensores.

Además de la distribución equilibrada de los datos por sensor, es igualmente importante analizar la distribución de las mediciones a lo largo de los días de la semana. Un conjunto de datos balanceado en términos de la cantidad de mediciones por día de la semana asegura que nuestras conclusiones sean representativas en todos los días, evitando posibles sesgos relacionados con patrones específicos de ciertos días.

En la Figura 4.4, se muestra otro gráfico circular que ilustra la proporción de datos disponibles para cada día de la semana. Al observar este gráfico, se aprecia que los datos de todos los días de la semana están presentes en el conjunto de datos, y que la cantidad de mediciones por día de la semana es similar. Esto nos permite obtener una visión más completa y precisa del comportamiento de los niveles de CO₂ en relación con los diferentes días de la semana en

el entorno de estudio y poder relacionarlo con otros factores como el número de ocupantes de la clase.

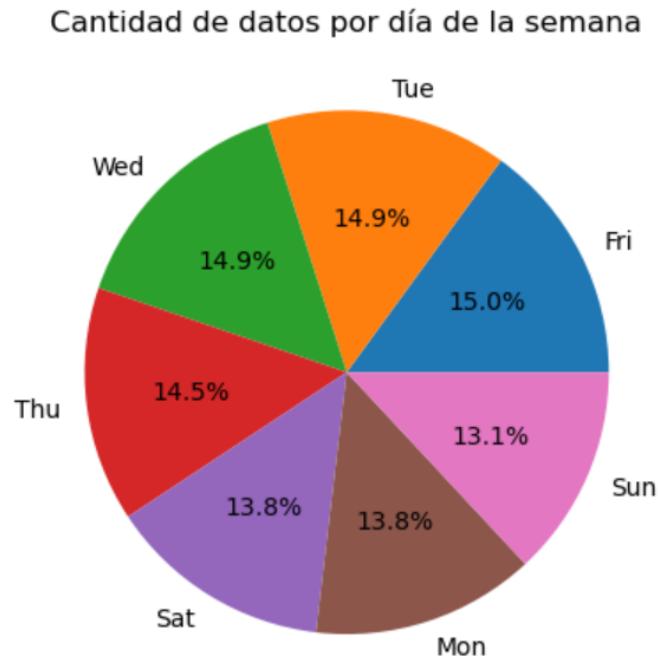


Figura 4.4: Gráfico circular con la distribución de los días de la semana.

El siguiente paso es examinar los datos para ver cada cuánto tiempo se realizan las mediciones. Una frecuencia baja puede dar lugar a la pérdida de información importante si ocurren cambios o fluctuaciones significativas en los datos en un intervalo de tiempo menor a la frecuencia de medición. En nuestro caso, se comprueba que los registros se realizan cada **5 minutos**. Si bien frecuencias más elevadas, como tomar mediciones cada 1 minuto o incluso cada 30 segundos, podrían ser más óptimas para capturar con precisión los patrones temporales más finos, es importante considerar el contexto de nuestro caso particular. En el caso de la detección de anomalías de CO₂, la frecuencia de muestreo de 5 minutos es suficiente para identificar patrones generales y anomalías significativas en el tiempo.

Continuando con el análisis temporal de los datos, hemos examinado los valores máximos de CO₂ registrados en cada mes. En la Figura 4.5 se presentan los resultados. Como se mencionó anteriormente, nuestro conjunto de datos abarca dos meses del año 2021: mayo y junio. Al analizar los valores máximos de CO₂ registrados, se observa una diferencia significativa entre ambos meses. En mayo, el valor máximo de CO₂ se sitúa alrededor de los 1250 ppm, mientras que en junio se alcanzan casi los 2000 ppm. Estos resultados sugieren que puede haber diferencias estacionales o factores externos que influyen en los niveles de CO₂ en las clases durante mayo y junio. Es importante no sacar conclusiones precipitadas ya que durante el análisis se ha

determinado que existe una correlación total entre el mes y la localidad, es decir, los datos de Sant Miquel se han tomado durante el mes de mayo y los de Albea durante el mes de Junio. Debido a esto no se puede determinar si las variaciones entre meses se deben a algún factor meteorológico o estacional o, simplemente a las propiedades de cada entorno determinado (ventilación, número de ocupantes por clase, política de apertura de ventanas...).

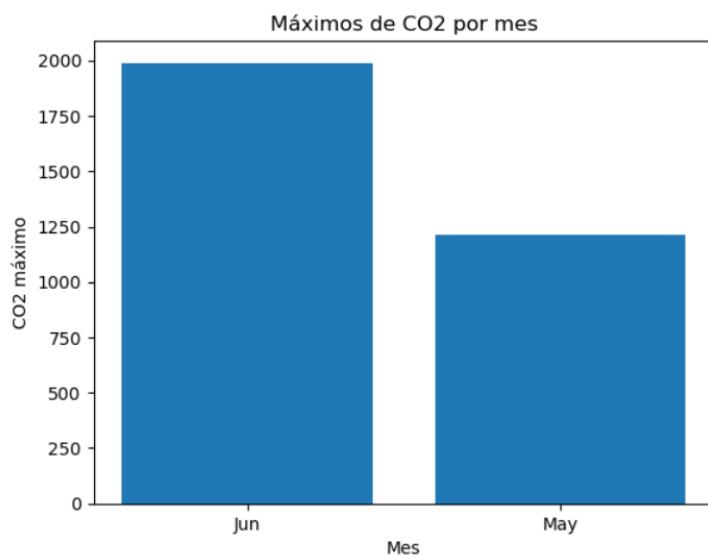


Figura 4.5: Máximos de CO2 mensuales.

En cuanto a los máximos de CO₂ por cada día de la semana, se ha observado una diferencia notable entre los días laborables (de lunes a viernes) y los fines de semana. En la Figura 4.6 se muestran los máximos para cada día y se ha encontrado que para los días laborables los valores máximos de CO₂ se mantienen en un rango muy similar, oscilando entre 1600 y 2000 ppm. Por otro lado, durante los fines de semana, se ha observado una reducción significativa en los niveles de CO₂, con valores en torno a los 500 ppm. Esta disminución indica una clara diferencia en la calidad del aire durante los días no laborables en comparación con los días de la semana. Estos resultados pueden atribuirse a la forma de operar en las escuelas. Durante los días laborables, en las escuelas se están impartiendo clases y tienen una mayor concentración de personas, lo que conduce a una acumulación de CO₂ en el ambiente. Por el contrario, durante los fines de semana, es probable que estos lugares estén menos ocupados o incluso cerrados, por lo que los niveles de CO₂ disminuyen significativamente.

Para acabar con el análisis temporal, en la Figura 4.7 se presentan los máximos de CO₂ para cada hora del día. Como era de suponer, los valores más elevados se encuentran durante la mañana y primeras horas de la tarde, lo que se corresponde con un horario habitual de una escuela de primaria. Las fluctuaciones en estas horas pueden deberse a factores que vacían las aulas como el recreo (probablemente en la franja de las 12:00) o la hora de comer (a las 13:00).

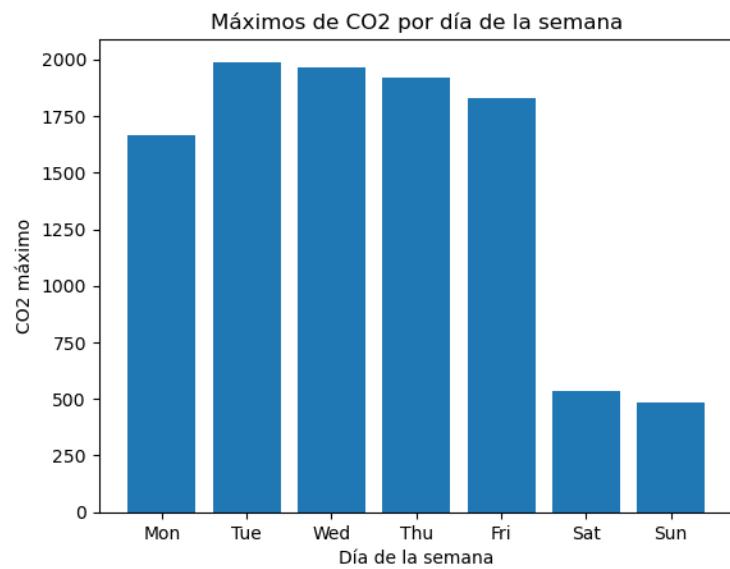


Figura 4.6: Máximos de CO₂ por día de la semana.

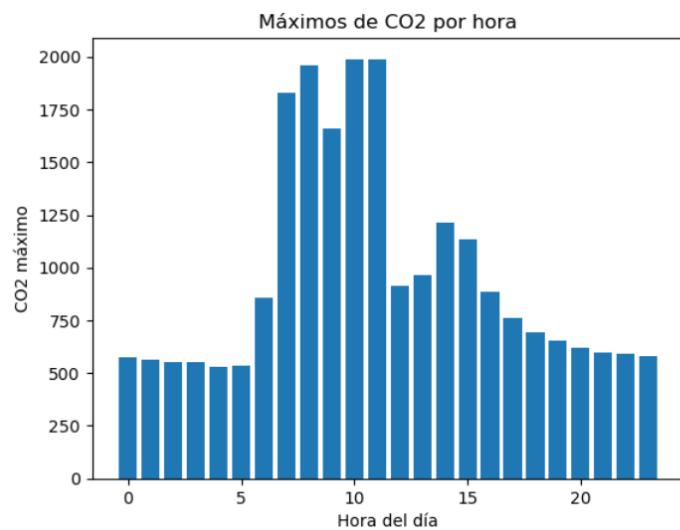


Figura 4.7: Máximos de CO₂ por hora del día.

Estas conclusiones se han analizado conjuntamente con el artículo asociado a la fuente de los datos [40] en el que se especifican los horarios de las clases en las que se desplegaron los sensores. Debido a esto, se han validado las conclusiones tomadas, aunque se comprueba que existen valores elevados de CO₂ en las horas previas al comienzo de las clases. Esto podría deberse a que el personal docente acuda antes para preparar las clases o a que el personal de limpieza realice sus labores en este horario. Para descartar que estos valores máximos se deben a un fenómeno puntual que se hubiese producido en estas horas se ha elaborado la Figura 4.8 en la que se compara la distribución de los valores de CO₂ en la franja horaria previa al comienzo de las clases con una franja horaria que cae dentro de los horarios. Se puede comprobar que las distribuciones son muy semejantes, por lo que parece que los máximos no se deben a un hecho aislado y debe ser algo que se realiza habitualmente.

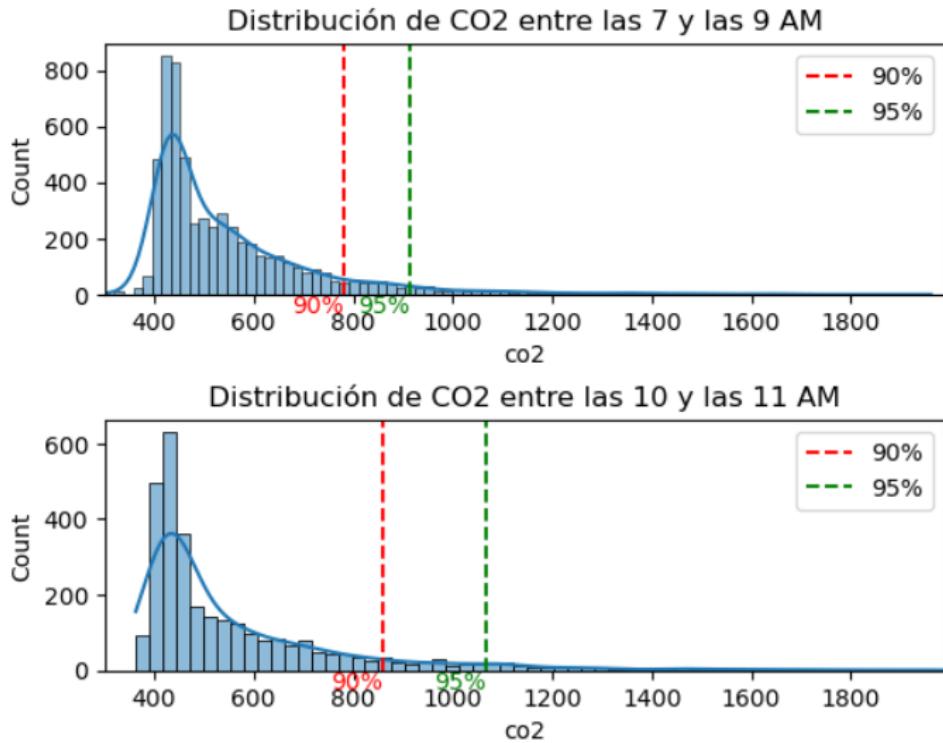


Figura 4.8: Distribución de los valores de CO₂ en dos franjas horarias.

Siguiendo con el estudio, se procede a realizar un análisis de normalidad de los datos. Este análisis resulta relevante debido a que algunos modelos y métodos estadísticos requieren que los datos sigan una distribución normal o se ajusten a ciertas suposiciones de normalidad. La normalidad de los datos implica que siguen una distribución gaussiana, lo cual facilita el uso de técnicas estadísticas paramétricas y la interpretación de los resultados obtenidos. El análisis de normalidad nos proporcionará información acerca de la forma de distribución de los datos y nos ayudará a seleccionar los modelos y métodos más adecuados. Si aplicamos el **Teorema**

del Límite central (TLC) [41] podemos asumir normalidad en nuestras series temporales de CO₂ al ser suficientemente grandes (normalmente mayores a 30 elementos). Aún así, se decide profundizar en esta cuestión por lo que en las Figuras 4.9 y 4.10 se presentan las distribuciones de las mediciones de CO₂ de cada sensor. En ellas se puede observar que ninguna tiene una forma de campana por lo que parece que **no siguen una distribución normal**. Esto se debe a la presencia de gran cantidad de valores elevados que se alejan de la media.

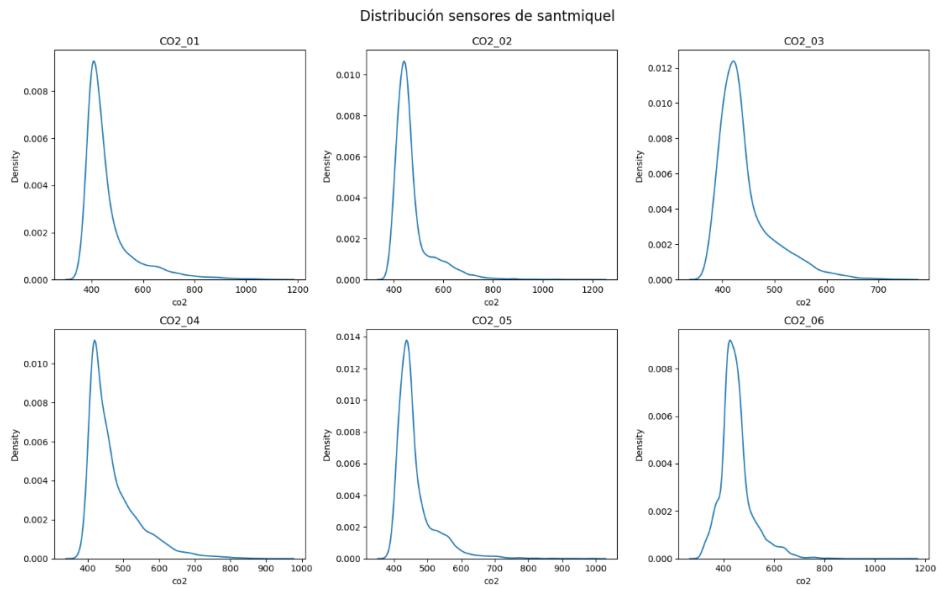


Figura 4.9: Distribución de los valores de CO₂ en cada sensor de Sant Miquel.

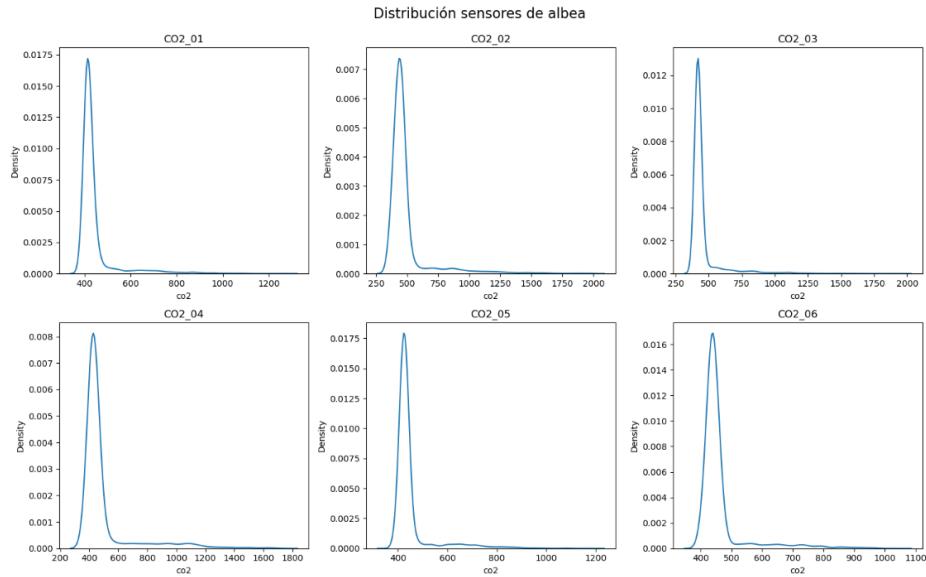


Figura 4.10: Distribución de los valores de CO₂ en cada sensor de Albea.

Para disipar cualquier duda sobre la normalidad de los datos, se aplica el **Test de Shapiro-Wilk** [42]. Este test es ampliamente utilizado para evaluar si una muestra de datos sigue una distribución normal. En nuestro análisis, se obtuvo un resultado interesante (Figura 4.11), ya que el valor de *p-valor* para todas las muestras fue igual a 0. El *p-valor* es una medida que indica la evidencia en contra de la hipótesis nula, que en este caso sería la hipótesis de normalidad de los datos. Un *p-valor* bajo, como el obtenido, sugiere que existe evidencia significativa para rechazar la hipótesis nula y **concluir** que los datos **no siguen una distribución normal**.

```
Sensor CO2_01 - albea: 0.0 (p-value)
Sensor CO2_02 - albea: 0.0 (p-value)
Sensor CO2_03 - albea: 0.0 (p-value)
Sensor CO2_04 - albea: 0.0 (p-value)
Sensor CO2_05 - albea: 0.0 (p-value)
Sensor CO2_06 - albea: 0.0 (p-value)
Sensor CO2_01 - santmiquel: 0.0 (p-value)
Sensor CO2_02 - santmiquel: 0.0 (p-value)
Sensor CO2_03 - santmiquel: 0.0 (p-value)
Sensor CO2_04 - santmiquel: 0.0 (p-value)
Sensor CO2_05 - santmiquel: 0.0 (p-value)
Sensor CO2_06 - santmiquel: 0.0 (p-value)
```

Figura 4.11: Test de Shapiro-Wilk.

La última parte de esta sección consiste en un análisis de correlación de los datos. Para ello, se recuperan las variables de temperatura y humedad que fueron eliminadas durante la etapa de limpieza de datos. Este análisis es relevante para determinar si existe alguna relación o dependencia entre la concentración de CO₂ y estas variables. Para el análisis se utiliza el **coeficiente de correlación de Spearman** por la falta de normalidad de nuestros datos. Esto se debe a que es un test no parámetrico, por lo que a diferencia del **coeficiente de correlación de Pearson**, no asume una distribución normal de los datos, lo que lo hace más adecuado para nuestro caso. En la Figura 4.12 se muestra un mapa de calor que representa los coeficientes de correlación de Spearman para cada par de variables. Se observa que todos los coeficientes se encuentran cercanos a 0, lo cual indica una falta de correlación significativa entre las variables.

Es llamativo que la temperatura y los niveles de CO₂ no tengan correlación, ya que cabría esperar que temperaturas elevadas favoreciesen la apertura de ventanas u otras medidas de ventilación que redujesen los niveles de CO₂.

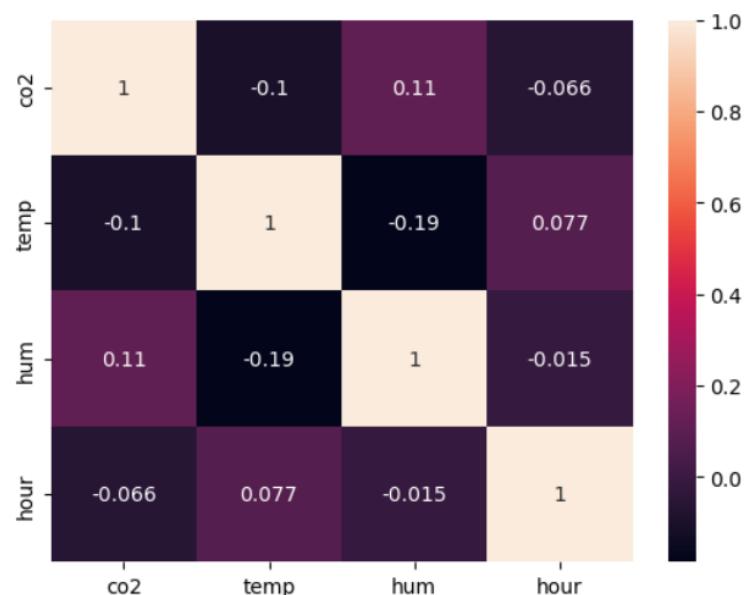


Figura 4.12: Mapa de calor con la correlación entre variables.

Capítulo 5

Generación de los modelos

En este capítulo se abordan todos los procesos llevados a cabo para la creación y evaluación de los modelos desarrollados. También es necesario realizar tareas previas de procesado de datos para crear los conjuntos que usarán después los modelos. Esto se incluye en esta fase y no en la anterior debido a que se puede considerar que estas tareas dependen de los modelos seleccionados. El código se encuentra en el fichero `models` y los conjuntos de datos en la carpeta `datasets` del repositorio de [GitHub](#).

5.1. Creación de los conjuntos de datos

En esta fase crearemos los conjuntos de entrenamiento (*train*), validación (*val*) y prueba (*test*) para alimentar a nuestros modelos. Este proceso debe realizarse de manera cuidadosa y siguiendo ciertos criterios para garantizar la calidad y representatividad de los datos. Es necesario aplicar las siguientes modificaciones previas:

- **Imputación de datos.**
- **Etiquetado de datos** del conjunto de *test*.
- **Eliminación de anomalías** en los conjuntos de *train* y *val*.
- **Normalización de datos.**
- **Creación** de ventanas deslizantes *sliding windows* de los datos de *train* y *val*.

5.1.1. Imputación de datos

El proceso de imputación de datos consiste en llenar los valores faltantes en un conjunto de datos con estimaciones u otros valores. En nuestro caso, tenemos series temporales con

intervalos fijos de 5 minutos y la falta de puntos intermedios podría perjudicar el desempeño de los modelos si existen patrones o tendencias importantes en los datos a nivel temporal. Tras examinar los datos, se comprueba que existen valores faltantes en las series temporales de todos los sensores. En el artículo [43] se presentan distintas técnicas de imputación de valores faltantes. Entre estas técnicas, se incluyen el relleno con la media o la mediana, el uso de valores aleatorios dentro de un rango determinado y la aplicación de interpolación lineal. En este caso, se ha decidido utilizar la técnica de **interpolación lineal** para imputar los valores faltantes en las series temporales. Esta decisión se debe a la necesidad de que exista cierta coherencia temporal en nuestra serie, y que no se introduzcan valores que se alejan considerablemente de sus vecinos cercanos. En la Figura 5.1 tenemos la serie temporal de uno de los sensores antes y después de la imputación de datos. Se puede observar que se han incorporado un total de **2.115 nuevos valores** que se corresponden con el 33 % del total y aún así **no se perciben cambios en su forma general**.

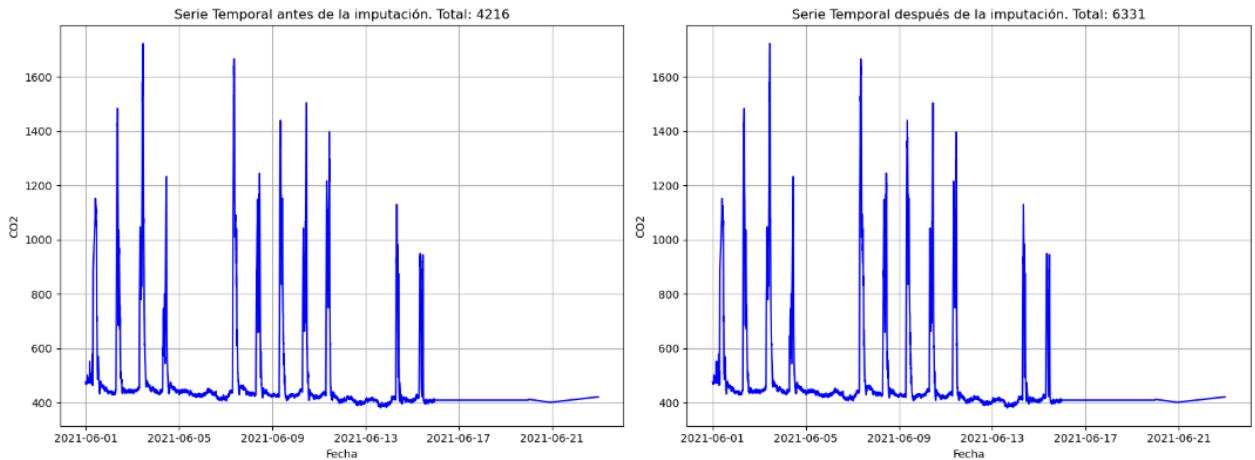


Figura 5.1: Serie temporal del sensor *CO2_04* de Albea antes y después de la imputación de datos.

5.1.2. Etiquetado de datos

Los modelos que se utilizarán para la tarea de detección de anomalías forman parte de la categoría de **métodos semi-supervisados** o **métodos no supervisados**, por lo que no es necesario etiquetar los datos para realizar el proceso de entrenamiento. Con todo, se realiza esta tarea para eliminar las anomalías de los conjuntos de *train* y *val* para los modelos LSTM-AE y para la fase de evaluación de resultados. Examinando otros artículos como [44] o [16] se observa que se utiliza la **regla 3-sigma** para etiquetar las anomalías en los datos de CO2.

La regla 3-sigma es una técnica estadística que se basa en la desviación estándar de los datos para identificar valores atípicos o anomalías. Esta regla establece que la mayoría de los datos se encuentran dentro de tres desviaciones estándar de la media en una distribución normal. Debido a que nuestros datos no sigue una distribución normal, se utiliza la **regla 3-(IQR)** [45] que no utiliza la media ni la distribución estándar. En su lugar, cataloga como anomalías los datos que se alejan 3 veces el rango intercuartílico del tercer cuartil, por lo que el umbral será $Q3 + 3 * IQR$. Como se observa en la Figura 5.2, el **umbral** se fija en **611 ppm**, lo que supone que el 5.94 % de los datos más extremos se identifican como valores anómalos.

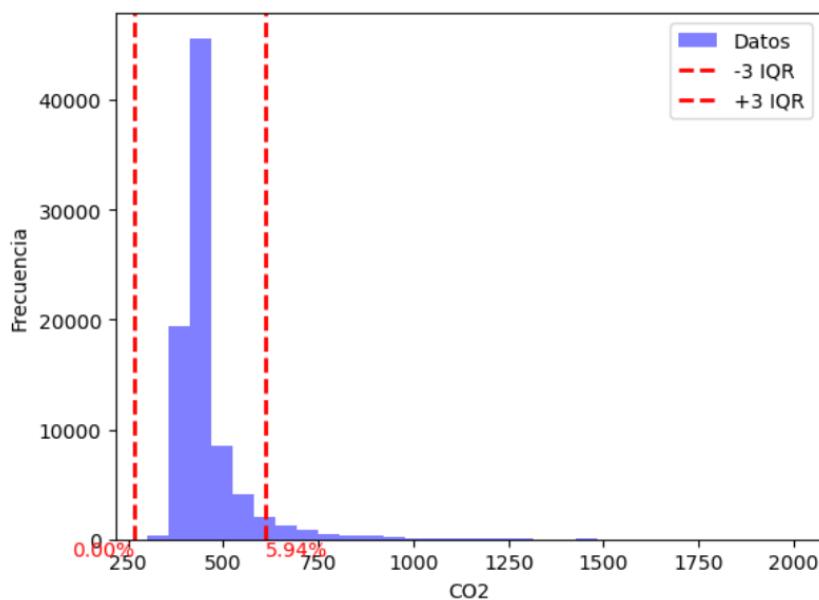


Figura 5.2: Distribución de los datos de CO2 con umbrales definidos por la regla 3-(IQR).

5.1.3. Eliminación de anomalías

Tras realizar el proceso de etiquetado, hemos identificado valores anómalos que deben ser eliminados de los conjuntos de *train* y *val* para la fase de entrenamiento. Como ya hemos mencionado anteriormente, queremos mantener la secuencia de valores con intervalos fijos de 5 minutos, por lo que no es suficiente con eliminarlos del *dataset*. En este caso decidimos crear la función `clean_anomalies` que utiliza el método de **k-Nearest Neighbor (kNN)** [46] con **k igual a 5** para la imputación de estos valores. Para ello se realiza un proceso iterativo en cada sensor que consiste en imputar los valores anómalos, recalcular las anomalías y repetirlo hasta que no existan más. Este método se basa en encontrar los k vecinos más cercanos a un valor anómalo y utilizar sus valores para estimar el valor faltante, asegurando que los valores imputados sean consistentes con el resto de la serie.

5.1.4. Normalización de datos

La normalización de datos es una tarea importante cuando se trabaja con múltiples variables que tienen diferentes escalas o unidades de medida. Sin una normalización adecuada, las diferencias en las escalas de las variables pueden afectar negativamente el rendimiento de los algoritmos de aprendizaje automático y distorsionar los resultados del análisis. En nuestro estudio solo tenemos la variable de CO₂, por lo que esta problemática no está presente. Aún así, se decide aplicar normalización para eliminar posibles impactos negativos de las diferentes escalas de las mediciones de CO₂ y reducir la complejidad computacional del entrenamiento de los modelos. Se han evaluado distintas técnicas de normalización como:

- **Normalización min-max:** Esta técnica reescaliza las variables para que estén en un rango específico, generalmente entre 0 y 1.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (5.1)$$

- **Normalización Z-score:** Esta técnica reescaliza las variables para que tengan una media de cero y una desviación estándar de uno.

$$x_{\text{scaled}} = \frac{x - \text{mean}(x)}{\text{std}(x)} \quad (5.2)$$

- **Normalización robusta:** Técnica que divide cada valor en el conjunto de datos por el IQR, de manera que los valores se escalan en función de la dispersión relativa de los datos en lugar de utilizar la desviación estándar.

$$x_{\text{scaled}} = \frac{x - Q1}{Q3 - Q1} \quad (5.3)$$

La elección de la técnica de normalización depende de las características y naturaleza de los datos, y debe seleccionarse correctamente para no introducir sesgos o distorsiones en el análisis. En este proyecto debemos seleccionar un método que no depende de que los datos tengan una distribución normal, además debe ser robusto ante la presencia de anomalías. En este contexto, la **normalización robusta** parece la **mejor opción**. La normalización robusta, a diferencia de las otras normalizaciones evaluadas que utilizan la media, la desviación estándar o los extremos, se basa en medidas de ubicación y dispersión resistentes a los valores atípicos. Esto es especialmente relevante cuando se trabaja con conjuntos de datos que pueden contener valores extremos, los cuales pueden distorsionar significativamente las estadísticas y métricas utilizadas en el análisis. Para el proceso de normalización se utiliza la clase *RobustScaler* de la

librería *Sklearn*, creando el *scaler* sobre los datos con anomalías. Es importante guardar este objeto ya que debe acompañar a los modelos desarrollados en la fase de despliegue, ya que para realizar la inferencia los datos deben tener la misma escala que durante el entrenamiento.

5.1.5. Creación de ventanas deslizantes

El objetivo de este proyecto es ser capaz de detectar anomalías de CO₂ en los datos. Las **anomalías puntuales** pueden detectarse con modelos que no capturen los patrones temporales subyacentes, pero para capturar **anomalías contextuales**, que pueden estar formadas por más de una medición consecutiva y no sobrepasar el umbral marcado anteriormente, es necesario alimentar a los modelos con secuencias temporales de datos. Para lograr esto, utilizaremos el enfoque de **ventanas deslizantes**.

Las ventanas deslizantes nos permiten dividir la serie temporal en secuencias más pequeñas y consecutivas, lo que nos brinda una visión más detallada de los patrones temporales presentes en los datos. Cada ventana consiste en un subconjunto de observaciones consecutivas y se desliza a lo largo de la serie temporal con un cierto paso o intervalo. En la Figura 5.3 se muestra un ejemplo de una ventana deslizante en la que la longitud de las secuencias es de 6 elementos y el paso es de 1 elemento, es decir, cada secuencia incorpora un nuevo valor con respecto a su predecesora, eliminando el valor más lejano temporalmente. La ventaja de utilizar ventanas deslizantes con respecto a dividir el *dataset* en subsecuencias que no comparten elementos radica en que estamos aumentando la información contextual de cada punto, al pasarlo al modelo en distintas posiciones de la secuencia. Además, limita que las anomalías contextuales no queden divididas en dos secuencias distintas, siempre que el tamaño de la secuencia sea suficiente.

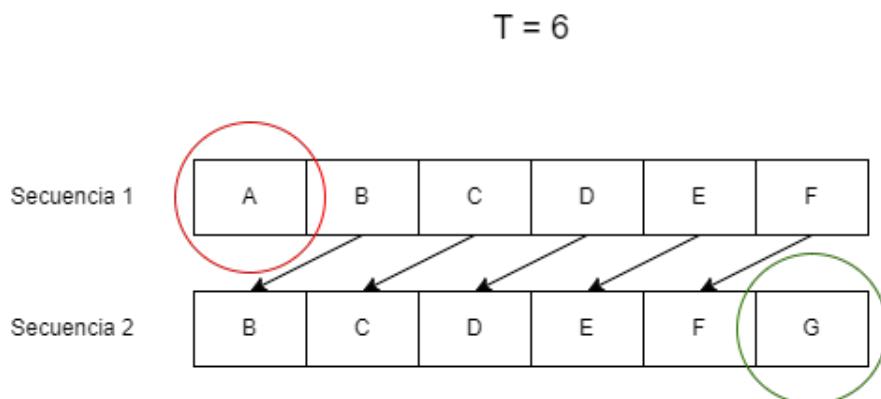


Figura 5.3: Ejemplo de *sliding window* con paso 1 y longitud de secuencia 6.

A la hora de crear las secuencias, se tiene que definir su tamaño. Este tamaño debe ser suficientemente grande para capturar la información relevante y los patrones temporales en

los datos, pero también debe ser lo suficientemente pequeño para evitar la pérdida de detalles importantes. Como punto de partida, se marca el valor de **12 elementos** como **tamaño de ventana**. Se toma esta decisión porque contamos con intervalos de 5 minutos, lo que nos daría ventanas de 1 hora. Aún así, durante la fase de creación de los modelos se experimentará con este valor para tratar de obtener los mejores resultados.

5.1.6. Subconjuntos de datos

Una vez que tenemos el *dataset* preparado y todos los procesos previos realizados, se deben preparar los conjuntos de *train*, *val* y *test* para el entrenamiento y evaluación de los modelos. Los modelos desarrollados se incluyen dentro de los métodos de aprendizaje **semi-supervisado** y **no supervisado**. Esta diferenciación provoca que los conjuntos de datos sean distintos. En los siguientes subapartados se detallan las diferencias entre los conjuntos generados para cada metodología.

5.1.6.1. Modelos semi-supervisados

En este proyecto, se ha tomado la decisión de asignar los datos de manera proporcional, utilizando una política en la que se destinan los datos de 6 sensores para el conjunto de entrenamiento, 2 sensores para el conjunto de validación y 4 sensores para el conjunto de prueba. Esta decisión se basa en el análisis de la cantidad de datos disponibles y en la necesidad de contar con una cantidad suficiente de datos tanto para el entrenamiento como para la evaluación de los modelos. La asignación de 6 sensores para el entrenamiento y 2 sensores para la validación se considera adecuada, ya que proporciona una cantidad significativa de datos para el proceso de aprendizaje y ajuste de los modelos. Los 6 sensores de entrenamiento se corresponden con la localización de Sant Miquel y el resto con los de Albea. Al utilizar únicamente los datos de Sant Miquel para el entrenamiento, se busca evitar un sesgo excesivo hacia una ubicación específica y permitir que los modelos aprendan patrones generales y características del CO₂ que puedan ser aplicables a diferentes contextos. Al final de este proceso se utiliza la función `create_sets` para que nuestros datos se organicen de la siguiente manera:

- **train_data**: Valores de CO₂ de Sant Miquel ordenados por sensor y temporalmente, normalizados y agrupados en ventanas deslizantes de longitud 12.
- **val_data**: Valores de CO₂ de los sensores *CO2_01* y *CO2_02* de Albea ordenados por sensor y temporalmente, normalizados y agrupados en ventanas deslizantes de longitud 12.

- `test_data`: Valores de CO₂ de los sensores de Albea restantes ordenados por sensor y temporalmente pero sin normalizar y agrupados en subsecuencias de longitud 12 pero sin deslizamiento. Se guardan estos datos sin normalizar para facilitar la interpretación de las visualizaciones de los resultados.
- `test_data_norm`: Valores de `test_data` normalizados.
- `test_data_labels`: Variable `target` asociada a los conjuntos `test_data` y `test_data_norm` que se utilizará para la evaluación de los modelos.

5.1.6.2. Modelos no supervisados

En este caso, se sigue la misma política que para los métodos semi-supervisados, con una pequeña variación. En lugar de utilizar un conjunto de validación, se elimina esta etapa y se mantienen las anomalías en el conjunto de entrenamiento. Esto se debe a que estos modelos tienen la capacidad de discriminar las anomalías durante el entrenamiento, sin requerir una limpieza previa de las mismas. Es importante destacar que estos modelos no son capaces de identificar patrones temporales en los datos, lo cual implica que no tiene sentido aplicar el enfoque de ventanas deslizantes. Aunque son más rápidos y simples, pierden la capacidad de detectar anomalías contextuales. En resumen, los modelos no supervisados utilizados son más rápidos y simples, pero no pueden capturar patrones temporales y no son adecuados para la detección de anomalías contextuales. En este caso obtenemos los subconjuntos siguientes:

- `train_data`: Valores de CO₂ de Sant Miquel y los dos primeros sensores de Albea ordenados por sensor y temporalmente y normalizados.
- `test_data`: Valores de CO₂ de los sensores de Albea restantes ordenados por sensor y temporalmente pero sin normalizar. Se guardan estos datos sin normalizar para facilitar la interpretación de las visualizaciones de los resultados.
- `test_data_norm`: Valores de `test_data` normalizados.
- `test_data_labels`: Variable `target` asociada a los conjuntos `test_data` y `test_data_norm` que se utilizará para la evaluación de los modelos.

5.2. Creación de los modelos

En esta sección se detalla el funcionamiento de los modelos desarrollados, explicando como se ha realizado la búsqueda de los mejores **hiperparámetros** y los puntos fuertes y débiles de cada modelo.

5.2.1. Modelo Coud: LSTM-AE

El primero de los modelo desarrollados es un **Autoencoder basado en LSTMs (LSTM_AE)**. Los Autoencoders son una clase de redes neuronales que se utilizan para reducir la dimensionalidad de la entrada y volver a reconstruirla. En este caso, se ha implementado un Autoencoder utilizando celdas LSTM, lo que ayudará a capturar patrones temporales en las secuencias de datos. El objetivo principal de este modelo es aprender una representación latente de los datos de entrada y luego generar una reconstrucción lo más fiel posible de los mismos. Al alimentar el modelo con datos normales, aprenderá a reconstruir muy bien estos datos pero no aquellos que se escapan a esta normalidad. El objetivo es determinar un **umbral máximo de error** en la reconstrucción a partir del cuál el dato se considerará como anómalo. En la Figura 5.4 se ilustra este proceso. Para determinar el valor de este umbral, se decide utilizar el **error máximo** de la reconstrucción en el **conjunto de validación** tras la fase de entrenamiento.

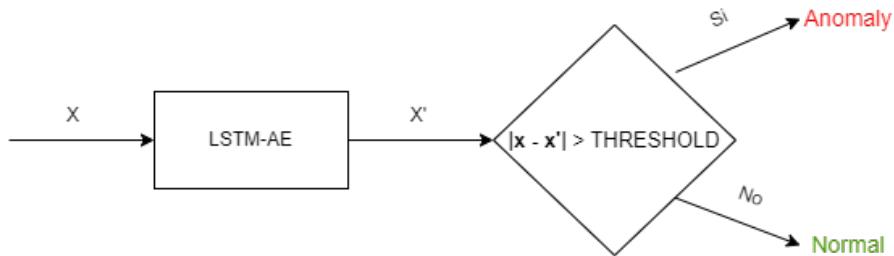


Figura 5.4: Proceso general de detección de anomalías con LSTM_AE.

En cuanto a la arquitectura de la red, se ha decidido utilizar **Keras** haciendo uso de las capas que se pueden ver en la Figura 5.5.



Figura 5.5: Topología del modelo LSTM_AE de 3 capas.

- **InputLayer:** Define la forma de los datos de entrada del modelo. En este caso, se espera que los datos de entrada sean de forma $(?, t, 1)$. Esto indica que el modelo se alimentará con lotes de tamaño variable de secuencias de longitud t y una sola característica, que en este caso es la medida de CO₂.
- **Codificador con 3 capas LSTM:** Se ha decidido utilizar un codificador compuesto por **3 capas LSTM**, cada una con una activación **tanh**. El objetivo de estas capas LSTM es

reducir la dimensionalidad de los datos de entrada y capturar patrones y características relevantes en las secuencias temporales de CO₂. Cada capa LSTM tiene una dimensionalidad menor que la anterior, lo que permite una representación latente más compacta.

- **RepeatVector:** El RepeatVector es una capa que repite la secuencia de salida del codificador múltiples (t) veces. En este caso, se utiliza para expandir la representación latente aprendida por el codificador y prepararla para el decodificador.
- **Decodificador con 3 capas LSTM:** El decodificador consta de **3 capas LSTM** que tienen una estructura similar al codificador. Estas capas LSTM tienen la tarea de reconstruir las secuencias temporales de CO₂ a partir de la representación latente expandida.
- **TimeDistributed con capa Dense:** Se utiliza para aplicar una **operación densa** a cada uno de los pasos de tiempo en la secuencia de salida del decodificador. Esto permite generar **una salida para cada punto de tiempo** en la secuencia reconstruida.

Además de esto se ha utilizado el optimizador **Adam** y la métrica del *error absoluto medio* para la función de pérdida.

Para encontrar los mejores hiperparámetros para esta arquitectura, se ha utilizado la librería **Optuna**. Esta librería permite comparar el desempeño del modelo con distintas combinaciones de hiperparámetros. En este caso, se busca minimizar la salida de la función **objective**. Esta función entrena el modelo con una selección de hiperparámetros y devuelve el **error de reconstrucción máximo** en el conjunto de *val*. Se han realizado múltiples pruebas para encontrar la mejor combinación, pero el modelo se ha entrenado con un número de épocas reducido para mantener los tiempos en un rango asumible. Esto produce que las métricas obtenidas no sean las óptimas, pero nos da una idea de las aportaciones que realiza cada hiperparámetro al desempeño del modelo. En la Tabla 5.1 se muestra un resumen con parte de las pruebas que se han realizado. Se puede ver que los mejores resultados se obtienen con los *learning rates* de 0.001, los tamaños de secuencia pequeños y los tamaños de lote de 32 o 64.

En la Figura 5.6 se observa la importancia de cada hiperparámetro a la hora de obtener los mejores resultados del **error de reconstrucción**. Parece que la longitud de la secuencia influye notablemente en el resultado, siendo más fácil reconstruir secuencias cortas. El número de unidades de las capas LSTM, el *learning rate* y el tamaño de los lotes son los que tienen menos peso en el rendimiento. Es importante tener esto en mente para escoger el conjunto de parámetros con los que realizaremos el entrenamiento.

Finalmente, la combinación de hiperparámetros seleccionada es la siguiente:

- **Batch size: 32.** Ya que se ha visto que esto no influye notablemente en el rendimiento del modelo. Además se selecciona el tamaño que ha obtenido los mejores resultados.

batch size	LSTM units	t	learning rate	Max error
16	64,32,16	20	0.001	3.84
16	128,64,32	12	0.0001	4.07
16	64,32,32	12	0.0001	3.46
32	64,32,16	20	0.0001	3.76
32	128,64,32	20	0.0001	3.50
32	128,64,32	12	0.0001	3.40
32	128,64,32	5	0.001	0.71
64	128,64,32	20	0.0001	3.63
64	128,64,32	12	0.001	3.92
64	64,32,16	12	0.001	3.63

Tabla 5.1: Combinación de hiperparámetros del modelo LSTM_AE de 3 capas con su **error de reconstrucción máximo**.

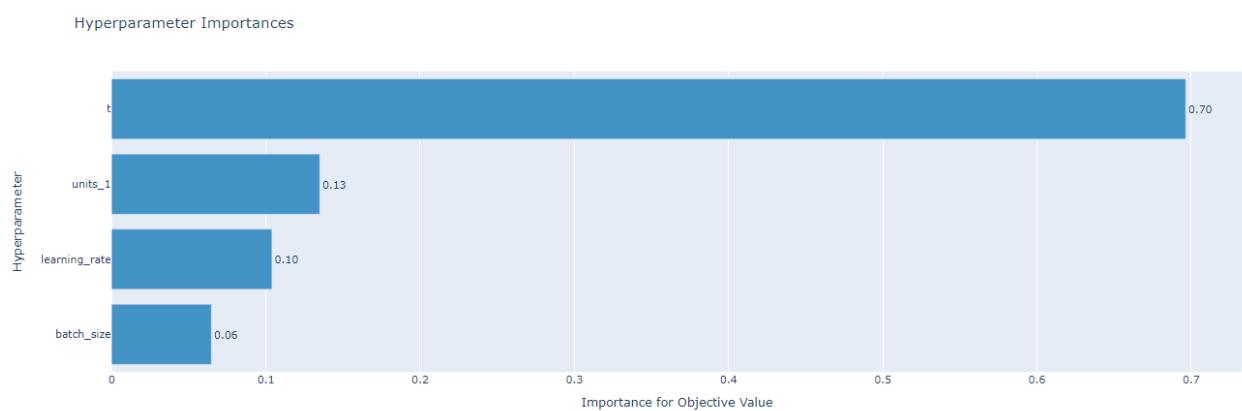


Figura 5.6: Importancia de los hiperparámetros para minimizar el **error de reconstrucción máximo**.

- **LSTM units: 64, 32, 16.** El número de unidades de las capas LSTM que se han probado parece que no tienen un peso significativo en el rendimiento del modelo. Debido a esto no se escoge la combinación de (128, 64, 32), buscando reducir el tamaño de la red y el tiempo de ejecución.
- **t: 6.** Parece que tamaños pequeños de secuencia funcionan mejor que tamaños grandes. Debido a esto se marca el valor de 6 como una secuencia significativamente grande para capturar patrones temporales pero no en exceso, para que no afecte al rendimiento del modelo.
- **Learning rate: 0.001.** Es uno de los factores que menos parece influir en los resultados, por lo que se decide utilizar el valor que ha obtenido los mejores desempeños.

Con esta configuración, el modelo **LSTM-AE** tiene un total de **65,729 parámetros**.

5.2.2. Modelo Standalone: LSTM-AE

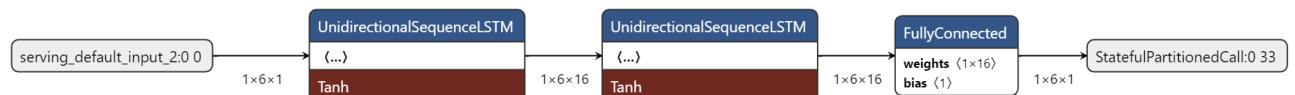


Figura 5.7: Topología del modelo LSTM_AE de 1 capa implementado con TensorFlow Lite.

Este modelo es una simplificación del anterior, adaptado específicamente para su implementación en la placa **ESP32-S3-DevKitC-1-N32R8V**. Aunque la estructura general del modelo se mantiene, se han realizado modificaciones para reducir la complejidad y ajustarse a las limitaciones de recursos de la placa. Como se puede ver en la Figura 5.7, algunas capas se han eliminado o simplificado para optimizar la utilización de memoria y capacidad de procesamiento del microcontrolador. A continuación, se analizan los cambios realizados en profundidad:

- **Reducción del número de capas LSTM:** Para adaptar el modelo a la placa del MCU, se ha reducido el número de capas LSTM tanto en el codificador como en el decodificador. Esto permite simplificar la estructura del modelo y reducir la carga computacional necesaria para su ejecución en la placa. Esto limita la capacidad del modelo para capturar patrones y relaciones más complejas y sutiles en los datos.
- **Número de unidades de la capa LSTM:** De nuevo, buscando reducir la memoria que ocupa el modelo, se establece a **16** el **número de unidades de la capa LSTM**. Además, se consigue que el espacio latente sea más compacto.

- **Establecer el tamaño del lote a 1:** Esto implica que el modelo procesará una secuencia de datos a la vez, tanto durante el entrenamiento como en la inferencia. El motivo de hacer esto es que al usar lotes, el modelo de **Tensorflow Lite** generado contiene capas que no son compatibles con la librería **Tensorflow Lite for Microcontrollers**.
- **Eliminar la capa RepeatVector:** Esta capa se utiliza en el modelo original para replicar el espacio latente antes de pasarlo al decodificador. Sin embargo, su eliminación no afecta significativamente a la capacidad del modelo mientras que reduce su tamaño en memoria.
- **Eliminar la capa TimeDistributed:** Al igual que la capa anterior, se elimina para reducir el tamaño del modelo y conseguir adaptarlo para el microcontrolador.
- **Reducción de las épocas de entrenamiento:** Al eliminar los lotes durante el entrenamiento, este proceso se vuelve más costoso temporalmente. Debido a esto, que establece un límite de **30 épocas**.

El modelo desarrollado tiene un total de **3,281 parámetros**, lo que supone una reducción del 95 % del número de parámetros con respecto al modelo anterior. Además, se realiza un proceso de conversión del modelo a **TensorFlow Lite** y posteriormente se convierte en un fichero hexadecimal legible en lenguaje **C++**. Este proceso se realiza con el objetivo de incrustar el modelo desarrollado en el entorno de la placa del MCU.

5.2.3. OneClass - SVM

El modelo **OC-SVM (One Class Support Vector Machine)** [47] es un método de aprendizaje **no supervisado** que funciona creando una *hiperesfera*¹ que engloba a los datos de una clase y todos los que quedan fuera se consideran **outliers**. Este modelo se ha utilizado ampliamente en problemas de detección de anomalías ([48]) debido a que su funcionamiento se ajusta muy bien a lo que se busca en estos problemas. La *hiperesfera* se ajusta maximizando el margen entre los datos y la frontera de decisión. El objetivo es encontrar la *hiperesfera* que mejor encapsule la mayor parte de los datos *normales*, minimizando así la cantidad de anomalías que quedan fuera de ella. Para esto, se debe tener en cuenta que el modelo necesita saber la proporción máxima de anomalías que presenta el conjunto de entrenamiento.

Para la creación del modelo se ha utilizado la clase **OneClassSVM** de **sklearn**. Esta clase necesita recibir el parámetro **nu**. Este parámetro controla la proporción de puntos considerados como anomalías en el conjunto de datos de entrenamiento, por lo que se debería realizar una búsqueda exhaustiva para encontrar el mejor valor. En nuestro problema de detección de

¹Generalización de la esfera a un espacio de dimensión arbitraria.

anomalías no tenemos forma de saber, sin usar los datos de *test*, la calidad de los parámetros elegidos. Esto se debe a que, durante el entrenamiento, no sabemos si el modelo ha clasificado correctamente al no tener datos etiquetados. Usar el conjunto de *test* para validar los hiperparámetros es una **mala práctica** por lo que se descarta.

Por todo esto, es necesario volver a la fase de análisis y preparación de los datos para entender qué proporción de los mismos se espera que sean anomalías. En esa fase se utilizó la **regla 3-(IQR)** para fijar el umbral a partir del cuál los datos se consideraban anomalías, quedando fuera de los rangos normales un **5.94 %** de los datos. Se decide utilizar ese umbral, tanto para este modelo como para el siguiente.

Una vez entrenado el modelo, se utiliza la herramienta **MicroMLGen** para trasladar el modelo a lenguaje **C++** y poder usarlo en el MCU.

5.2.4. Isolation Forest

Al igual que el modelo anterior, se trata de un método de aprendizaje **no supervisado** que no requiere de datos etiquetados o entrenamiento sin datos anómalos. El modelo **Isolation Forest** [49] funciona con la premisa de que es **más difícil aislar** puntos **anómalos** que puntos normales. Consiste en ir dividiendo los datos en dos árboles por un umbral aleatorio hasta que se llegue a algún criterio de parada como la profundidad o el número de árboles. Tras esto, se evalúa la *anormalidad* de cada punto en función de su altura de aislamiento. Los datos anómalos son más fáciles de aislar del resto, por lo que cuanto más alta sea la altura de un punto, más probabilidades tiene de ser una anomalía.

La librería **sklearn** proporciona la clase **IsolationForest**, a la que es necesario pasarle el parámetro **contamination**. Este parámetro es similar al **nu** del modelo anterior, ya que indica la proporción de anomalías del conjunto de datos. Debido a esto, se marca el umbral de **5.94 %** definido en el apartado anterior.

5.3. Comparativa de modelos

En este apartado, se presenta una recopilación de los resultados obtenidos por cada modelo analizado sobre el conjunto de *test*. Se han utilizado los mejores hiperparámetros, explicados en el apartado anterior. En la Tabla 5.2 se recopilan estos resultados.

Se puede ver que todos los modelos obtienen una **accuracy** elevada, lo que no es indicativo de un buen desempeño (ni tan poco malo). Esto se debe al **desbalanceo** que existe entre el número de datos **normales** y el número de **anomalías**. Por ejemplo, si un modelo clasifica todos los datos de *test* como datos normales, su accuracy será muy elevada, ya que el porcentaje de anomalías es muy reducido.

Modelo	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
LSTM-AE (Cloud)	1625	23797	147	117	0.99	0.917	0.933	0.925
LSTM-AE (Standalone)	254	3995	32	0	0.993	0.888	1	0.941
Isolation Forest	1506	23793	266	123	0.985	0.85	0.924	0.886
OC-SVM	1088	23879	684	37	0.972	0.614	0.967	0.751

Tabla 5.2: Tabla comparativa con las métricas obtenidas por todos los modelos desarrollados.

El resto de métricas nos ayudan con este problema. La **precisión** indica el porcentaje de datos que se han clasificado como anomalías y son realmente anomalías y el **recall** indica cuantas anomalías se clasifican correctamente. Está claro que un buen modelo de detección de anomalías debe maximizar estas dos métricas, por lo que se añade el **F1-score**, la media harmónica de ambas.

En términos del F1-score, se observa que los modelos **LSTM-AE** han obtenido **resultados excelentes**, mientras que el modelo **Isolation Forest** ha obtenido un **buen resultado**. Sin embargo, el modelo **OC-SVM** muestra **margen de mejora**, especialmente en su baja precisión, lo que indica que ha clasificado un gran número de observaciones normales como anomalías.

Llama la atención que el modelo **LSTM-AE** más simple es el que ha obtenido un mejor F1-score, provocado por su puntuación perfecta de recall, lo que significa que ha clasificado correctamente todas las anomalías.

Para completar el análisis de resultados, se presenta en la Figura 5.8 la **curva ROC** de todos los modelos. En este caso, el modelo **LSTM-AE** de tres capas ha logrado el mejor resultado y muy cercano a 1, lo que indica que es el mejor distinguiendo entre las clases (datos normales y anomalías) de los datos de CO2.

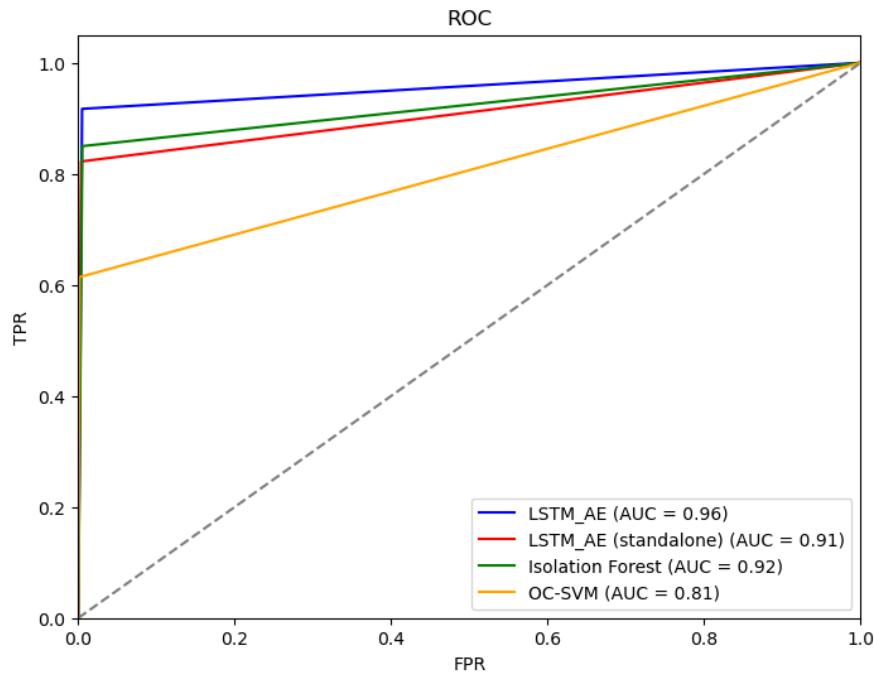


Figura 5.8: **Curva ROC** de los modelos desarrollados.

Además, se ha realizado una prueba de carga con cada modelo para evaluar sus consumos medios de **CPU** y **memoria RAM**. En el [repositorio GitHub](#) se pueden encontrar las pruebas, scripts en *python* que cargan los modelos para realizar la inferencia sobre los datos de *test* en bucle y con un paso de un segundo. El entorno de ejecución de las pruebas ha sido el **PC** definido en la sección 3.3 y se ha utilizado la herramienta **Monitor de Recursos** nativa de **Windows 11** para realizar el seguimiento de los procesos.

En la Tabla 5.3 se muestran los resultados, en los que se comprueba que el modelo **LSTM-AE (cloud)** tiene un consumo medio muy superior al resto. Los demás modelos presentan un uso de CPU reducido, por lo que son los más apropiados para desplegar en el MCU. Es importante destacar que los datos de *test* se cargan en memoria, lo cual afecta y aumenta su ocupación para todos los modelos.

Modelo	CPU (%)	Memoria RAM (KB)
LSTM-AE (Cloud)	8.80	420.852
LSTM-AE (Standalone)	0.01	282.060
Isolation Forest	0.23	123.144
OC-SVM	0.04	91.448

Tabla 5.3: Tabla comparativa con los consumos medios de CPU y memoria RAM de los modelos.

Debido a los resultados obtenidos, se ha decidido desplegar el modelo **LSTM-AE (cloud)** en el servidor y el modelo **LSTM-AE (Standalone)** en el MCU. Esta decisión se ha tomado basándose en que son los modelos con mejores rendimientos y que el consumo de recursos del modelo LSTM-AE (Standalone) es adecuado para las limitaciones del MCU.

Capítulo 6

Experimentos realizados

En este capítulo se detalla el entorno de pruebas desplegado para validar el sistema desarrollado durante la elaboración del *Trabajo de Fin de Máster*. Como ya se ha mencionado en la sección 3.3, se ha utilizado el MCU **ESP32-S3-WROOM-2-N32R8V** y un **PC** para el despliegue del sistema. En la Figura 6.1 se ve el **MCU** conectado al **PC** para aprovisionarlo de alimentación y poder ver la **CONSOLA MCU** en tiempo real. En la pantalla auxiliar se muestra la consola del **SERVIDOR** que recibe los datos del MCU mediante Wi-Fi y el **DASHBOARD** que se corresponde con la vista del *cliente web*. En este caso el cliente y el servidor están en la misma máquina física, pero podrían estar en distintos dispositivos dentro de la misma red.

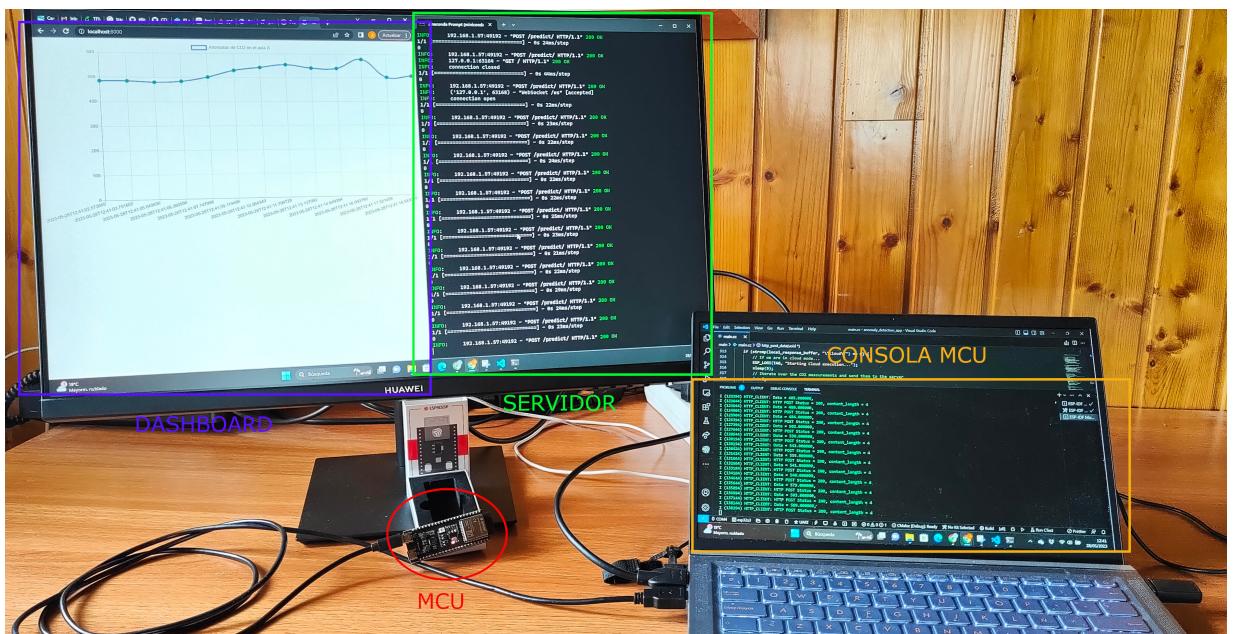


Figura 6.1: Entorno de pruebas

En la Figura 6.2 se muestra el *dashboard* con la serie temporal de mediciones de CO₂ para analizarlo en mayor profundidad.

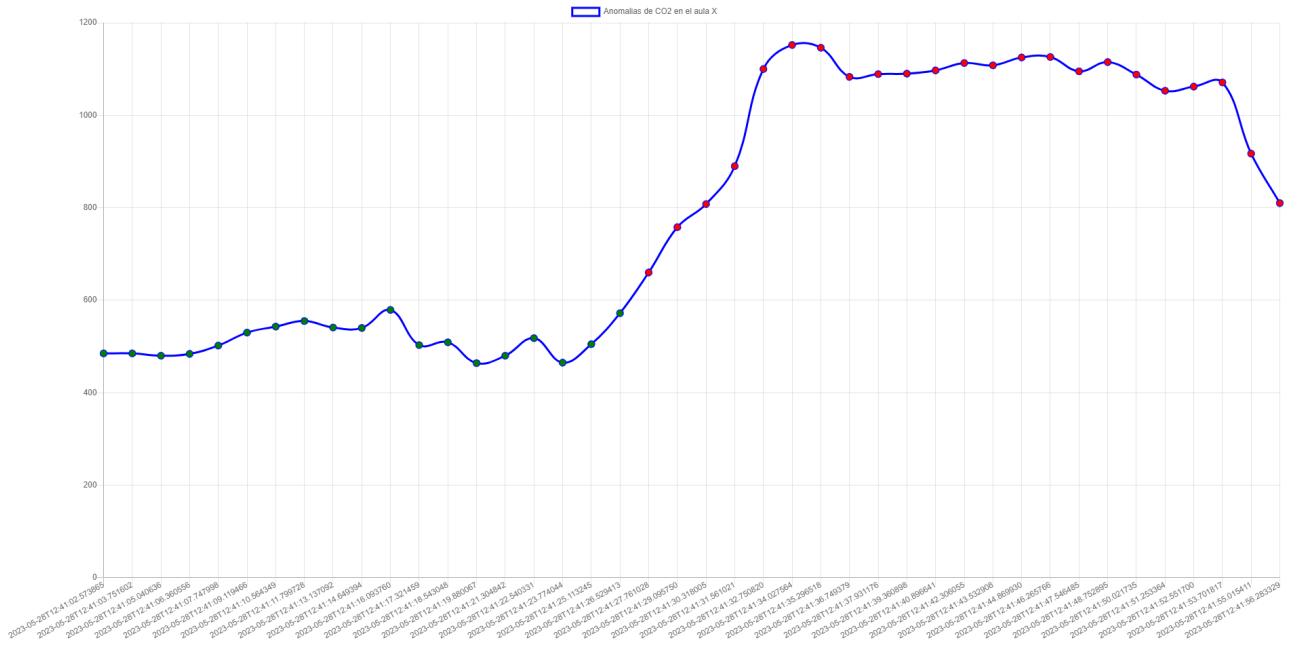


Figura 6.2: *Dashboard* que muestra la serie temporal de mediciones de CO₂.

El cliente web está *escuchando* constantemente y se actualiza automáticamente cuando el servidor procesa un nuevo punto de CO₂. Para la serie temporal se ha decidido utilizar el color **verde** para aquellos puntos que se han clasificado como *normales* y el color **rojo** para las mediciones de CO₂ catalogadas como *anomalías*.

Capítulo 7

Conclusiones

En este proyecto se ha desarrollado un **sistema de detección de anomalías** en tiempo real en las mediciones **de sensores de CO₂**. Debido a esto, se ha alcanzado el **objetivo principal** definido al inicio del trabajo. En cuanto a los **objetivos secundarios**, todos se han cumplido excepto el último, que consistía en implementar un mecanismo automático de selección de modelos donde el propio MCU fuese capaz de decidir si trabajar en modo *Cloud* o *Standalone*. Esto podría aportar más valor al sistema dotándolo de capacidad para auto-ajustarse y adaptarse en función de la situación. Aún así, no ha sido algo determinante debido a los buenos resultados de los modelos seleccionados, logrando tanto el modelo desplegado en el MCU como el alojado en el servidor valores de **F1-score** superiores al **90 %**. Estos resultados han sido posibles debido al **análisis del estado del arte** donde se han visto distintas soluciones que han servido como punto de partida para el proyecto. La **exploración de los conjuntos de datos** y la **preparación del dataset** también han sido fases críticas que han sido **iteradas varias veces**, logrando el objetivo de aumentar el rendimiento de los modelos. Para este proceso ha sido de gran ayuda la implementación de la metodología **CRISP-DM**. Por último, en el desarrollo de los modelos ha sido de gran ayuda la librería **Tensorflow Lite for Microcontrollers**, que ha permitido trasladar un modelo de *deep learning* a un espacio con recursos limitados como es la placa del MCU. En cuanto a la **planificación**, se ha intentado seguir los tiempos marcados al inicio del proyecto y, por lo general, se ha conseguido, a excepción de la tarea relacionada con el **mecanismo de selección de modelos**. Estos días han sido absorbidos por fases previas debido a la necesidad de realizar iteraciones del proceso. En resumen, se ha logrado desarrollar el sistema de **detección de anomalías de CO₂** y se ha desplegado en un entorno controlado, verificando su correcto funcionamiento. Los dos modos de trabajo (*Cloud* y *Standalone*) han sido probados, lo que permite trabajar con el MCU aunque no exista conexión con el servidor.

7.1. Líneas de trabajo futuras

Durante el desarrollo del TFM han ido apareciendo nuevas líneas de trabajo que no se han implementado por falta de tiempo o por no ajustarse al alcance del proyecto. Sería interesante explorar estas funcionalidades o mejoras en trabajos futuros:

- **Mecanismo de selección de modelos.** Como ya se ha mencionado, esta funcionalidad no se ha implementado por falta de tiempo, pero sería interesante profundizar en ella ya que podría suponer una mejora en la eficacia del sistema.
- **Mejora del cliente web.** El *dashboard* con la serie temporal de CO₂ es muy útil para que el usuario explore las mediciones en cada momento de tiempo. Esta vista puede refinarse para utilizar estilos y colores que aporten una mayor usabilidad. Adicionalmente, es posible generar otras visualizaciones con diferentes tipos de gráficos informativos. Por ejemplo, se podría crear un gráfico de sectores que muestre el porcentaje de anomalías registradas en la última semana.
- **Crear modelos predictivos.** Los modelos desarrollados podrían acompañarse de modelos predictivos que permitan anticiparse aún más a la ocurrencia de anomalías. Ambos modelos podrían coexistir y trabajar en conjunto para proporcionar una detección y predicción más precisa de las anomalías.
- **Extender a otros escenarios.** Los modelos han sido entrenados utilizando datos de escuelas de primaria. Sin embargo, sería interesante explorar la posibilidad de aplicar las arquitecturas desarrolladas en otros casos de uso donde la detección de anomalías de CO₂ es de vital importancia, como en el contexto de minas subterráneas.

Appendices

Apéndice A

Repositorio de Código

El código desarrollado se encuentra en el repositorio de **GitHub**:

<https://github.com/jgallego9/CO2-indoor-anomaly-detection>

Bibliografía

- [1] Commons W. CRISP-DM Process Diagram; 2013. Available from: https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png [cited 03-12-23].
- [2] Darban ZZ, Webb GI, Pan S, Aggarwal CC, Salehi M. Deep Learning for Time Series Anomaly Detection: A Survey; 2022.
- [3] Granell C, Kamilaris A, Kotsev A, Ostermann FO, Trilles S. Internet of things. Manual of digital earth. 2020:387-423.
- [4] Trilles S, González-Pérez A, Huerta J. An IoT platform based on microservices and serverless paradigms for smart farming purposes. Sensors. 2020;20(8):2418.
- [5] Ashton K. That ‘Internet of Things’ Thing; 2009. Available from: <https://www.rfidjournal.com/that-internet-of-things-thing> [cited 03-11-2023].
- [6] Seppänen OA, Fisk WJ, Mendell MJ. Association of Ventilation Rates and CO₂ Concentrations with Health and Other Responses in Commercial and Institutional Buildings. Indoor Air. 1999;9(4):226-52. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-0668.1999.00003.x>.
- [7] Peng Z, Jimenez JL. Exhaled CO₂ as a COVID-19 infection risk proxy for different indoor environments and activities. Environmental Science & Technology Letters. 2021;8(5):392-7.
- [8] Wirth R, Hipp J. CRISP-DM: Towards a standard process model for data mining. In: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. vol. 1. Manchester; 2000. p. 29-39.
- [9] Chapman P, Clinton J, Kerber R, Khabaza T, Reinartz T, Shearer C, et al. The CRISP-DM user guide. In: 4th CRISP-DM SIG Workshop in Brussels in March. vol. 1999. sn; 1999. .

- [10] Unidas N. Objetivos de Desarrollo Sostenible;. Available from: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/> [cited 03-12-23].
- [11] Cook AA, Misirlı G, Fan Z. Anomaly Detection for IoT Time-Series Data: A Survey. IEEE Internet of Things Journal. 2020;7(7):6481-94.
- [12] Hawkins DM. Identification of outliers. vol. 11. Springer; 1980.
- [13] Bontemps L, Cao VL, McDermott J, Le-Khac NA. Collective anomaly detection based on long short-term memory recurrent neural networks. In: Future Data and Security Engineering: Third International Conference, FDSE 2016, Can Tho City, Vietnam, November 23-25, 2016, Proceedings 3. Springer; 2016. p. 141-52.
- [14] Malhotra P, Vig L, Shroff G, Agarwal P, et al. Long Short Term Memory Networks for Anomaly Detection in Time Series. In: ESANN. vol. 2015; 2015. p. 89.
- [15] Lin S, Clark R, Birke R, Schönborn S, Trigoni N, Roberts S. Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2020. p. 4322-6.
- [16] Wei Y, Jang-Jaccard J, Xu W, Sabrina F, Camtepe S, Boulic M. LSTM-Autoencoder-Based Anomaly Detection for Indoor Air Quality Time-Series Data. IEEE Sensors Journal. 2023;23(4):3787-800.
- [17] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735-80.
- [18] Kingma DP, Welling M. Auto-Encoding Variational Bayes; 2022.
- [19] BIS. Evaluación de indicadores adelantados mediante el área AUC;. Available from: https://www.bis.org/publ/qtrpdf/r_qt1403z_es.htm [cited 02-25-23].
- [20] Sliwa B, Piatkowski N, Wietfeld C. LIMITS: Lightweight machine learning for IoT systems with resource limitations. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE; 2020. p. 1-7.
- [21] Ngo MV, Luo T, Quek TQS. Adaptive Anomaly Detection for Internet of Things in Hierarchical Edge Computing: A Contextual-Bandit Approach. ACM TRANSACTIONS ON INTERNET OF THINGS. 2022 FEB;3(1).

- [22] Marco VS, Taylor B, Wang Z, Elkhatib Y. Optimizing Deep Learning Inference on Embedded Systems Through Adaptive Model Selection. *ACM TRANSACTIONS ON Embedded COMPUTING SYSTEMS*. 2020 FEB;19(1).
- [23] Feng C, Sun M, Zhang J. Reinforced Deterministic and Probabilistic Load Forecasting via Q -Learning Dynamic Model Selection. *IEEE Transactions on Smart Grid*. 2020;11(2):1377-86.
- [24] Trilles S. Co2 concentration, temperature and humidity in primary classrooms during the Covid-19 Safety Measures in Spain. Zenodo; 2021. Available from: <https://doi.org/10.5281/zenodo.5062837>.
- [25] pandas development team T. pandas-dev/pandas: Pandas. Zenodo; 2020. Latest. Available from: <https://doi.org/10.5281/zenodo.3509134>.
- [26] Hunter JD. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2007;9(3):90-5.
- [27] Waskom ML. seaborn: statistical data visualization. *Journal of Open Source Software*. 2021;6(60):3021. Available from: <https://doi.org/10.21105/joss.03021>.
- [28] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020;17:261-72.
- [29] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825-30.
- [30] EloquentArduino. micromlgen: Generate C code for machine learning models; 2023. <https://github.com/eloquentarduino/micromlgen>.
- [31] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al.. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. Software available from tensorflow.org. Available from: <https://www.tensorflow.org/>.
- [32] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework; 2019.
- [33] Ramírez S. FastAPI; 2023. Versión 0.95.2. Available from: <https://fastapi.tiangolo.com/>.

- [34] Systems E. ESP-IDF; 2023. Versión 4.0.2. Available from: <https://github.com/espressif/esp-idf>.
- [35] Anaconda Software Distribution. Anaconda Inc.; 2023. Available from: <https://docs.anaconda.com/>.
- [36] Ashton K. That ‘Internet of Things’ Thing. ISO; 2019. Available from: <https://www.iso.org/iso-8601-date-and-time-format.html> [cited 05-14-2023].
- [37] Atarodi Z, Karimyan K, Gupta VK, Abbasi M, Moradi M. Evaluation of indoor air quality and its symptoms in office building – A case study of Mashhad, Iran. Data in Brief. 2018;20:74-9. Available from: <https://www.sciencedirect.com/science/article/pii/S2352340918308308>.
- [38] Asif A, Zeeshan M, Jahanzaib M. Indoor temperature, relative humidity and CO₂ levels assessment in academic buildings with different heating, ventilation and air-conditioning systems. Building and Environment. 2018;133:83-90. Available from: <https://www.sciencedirect.com/science/article/pii/S036013231830060X>.
- [39] Griffiths M, Eftekhari M. Control of CO₂ in a naturally ventilated classroom. Energy and Buildings. 2008;40(4):556-60. Available from: <https://www.sciencedirect.com/science/article/pii/S0378778807001387>.
- [40] Trilles S, Juan P, Chaudhuri S, Fortea ABV. Data on CO₂, temperature and air humidity records in Spanish classrooms during the reopening of schools in the COVID-19 pandemic. Data in Brief. 2021;39:107489.
- [41] Kwak SG, Kim JH. Central limit theorem: the cornerstone of modern statistics. Korean journal of anesthesiology. 2017;70(2):144-56.
- [42] SHAPIRO SS, WILK MB. An analysis of variance test for normality (complete samples)†. Biometrika. 1965 12;52(3-4):591-611. Available from: <https://doi.org/10.1093/biomet/52.3-4.591>.
- [43] Zhang Z. Missing data imputation: focusing on single imputation. Annals of translational medicine. 2016;4(1).
- [44] Liu Y, Pang Z, Karlsson M, Gong S. Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control. Building and Environment. 2020;183:107212. Available from: <https://www.sciencedirect.com/science/article/pii/S0360132320305837>.

- [45] Understanding Outliers. University of Florida; 2022. <https://bolt.mph.ufl.edu/6050-6052/unit-1/one-quantitative-variable-introduction/understanding-outliers/>.
- [46] Zhang S. Nearest neighbor selection for iteratively kNN imputation. Journal of Systems and Software. 2012;85(11):2541-52. Available from: <https://www.sciencedirect.com/science/article/pii/S0164121212001586>.
- [47] Schölkopf B, Williamson RC, Smola A, Shawe-Taylor J, Platt J. Support vector method for novelty detection. Advances in neural information processing systems. 1999;12.
- [48] Li KL, Huang HK, Tian SF, Xu W. Improving one-class SVM for anomaly detection. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693). vol. 5; 2003. p. 3077-81 Vol.5.
- [49] Liu FT, Ting KM, Zhou ZH. Isolation forest. In: 2008 eighth ieee international conference on data mining. IEEE; 2008. p. 413-22.