Trabajo Fin de Máster
Máster en Ingeniería Electrónica, Robótica y Automática

Aprendizaje automático para la detección de anomalías en sensores de bajo coste

Autor: Adrián Rocha Íñigo

Tutores: José Manuel García Campos

Daniel Gutiérrez Reina

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster
Máster en Ingeniería Electrónica, Robótica y Automática

Aprendizaje automático para la detección de anomalías en sensores de bajo coste

Autor:
Adrián Rocha Íñigo

Tutores:
José Manuel García Campos
Profesor Interino Sustituto

Daniel Gutiérrez Reina
Profesor Contratado Doctor

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster: Aprendizaje automático para la detección de anomalías en sensores de bajo coste

Autor: Adrián Rocha Íñigo
Tutores: José Manuel García Campos
Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Estas palabras constituyen el punto final del Máster en Ingeniería Electrónica, Robótica y Automática, el cual, además de diluir fronteras en ámbitos del conocimiento que despiertan en mí gran interés, me ha permitido dar los primeros pasos en el mundo laboral. Por ello, me gustaría dar las gracias a todos los compañeros de Fundación Ayesa, quienes, desde el primer día, han conseguido que trabajar a su lado haya sido una experiencia enriquecedora.

En especial, a José Manuel García, por dedicarme su valioso tiempo y conocimiento, además de su inestimable ayuda y enseñanzas más allá del contexto de este trabajo.

Por último, y más importante, gracias a mis padres, por hacerme la persona que soy hoy, por enseñarme a aspirar cada vez más alto y, porque sin sus esfuerzos, nada de esto sería posible. A mi hermana, por ser el mejor ejemplo a seguir. Y a mi pareja, porque su apoyo incondicional constituye la base de mis éxitos.

Adrián Rocha Íñigo
Ingeniero Industrial

Máster en Ingeniería Electrónica, Robótica y Automática

Sevilla, 2021

Resumen

Este trabajo nace a partir del proyecto *Agricultura 4.0*, en el cual se despliegan redes de sensores de bajo coste en entornos agrícolas. Este tipo de sensores de baja calidad son más propensos a introducir anomalías entre sus mediciones, es decir, valores fuera de lo común que no se ajustan al comportamiento esperado. Este hecho supone un problema, puesto que dificulta la explotación posterior de los datos registrados y, por ello, surge la necesidad real de mejorar la fiabilidad de las mediciones tomadas.

Para solventar dicho problema, en este trabajo se desarrolla una metodología para la detección de anomalías en las mediciones tomadas por sensores de bajo coste. La base del método consiste en convertir el problema de detección de anomalías en una serie temporal en un problema de identificación de outliers basado en una nube de puntos. De esta manera, se consigue resolver la cuestión original mediante las técnicas conocidas para dar solución al problema derivado. En concreto, se ha empleado el método de los k vecinos más cercanos, al cual se le ha aplicado una serie de ideas propias para mejorar su desempeño en este contexto.

Además, para poder evaluar la metodología propuesta, se ha elaborado, a modo de banco de pruebas o testbed, una red de sensores inalámbrica de bajo coste y fácilmente escalable, cuya implementación también es descrita en este documento. Esta, además, también podrá ser usada como entorno controlado para estudiar, desarrollar, poner a prueba y comparar nuevas técnicas para la detección de anomalías. Los resultados obtenidos son prometedores, puesto que reflejan una exactitud, sensibilidad y especificidad superiores al 95 %, por lo que la solución obtenida, tras los ajustes requeridos, será usada en el proyecto *Agricultura 4.0* para solventar el problema original.

Abstract

This work was born from the *Agricultura 4.0* project, in which low-cost sensor networks are deployed in agricultural environments. This type of low quality sensors are more prone to introduce anomalies among their measurements, i.e., out-of-the-ordinary values that do not conform to the expected behavior. This fact is a problem, since it hinders the subsequent exploitation of the recorded data and, for this reason, there is a real need to improve the reliability of the measurements taken.

In order to solve this problem, this work proposes a methodology for the detection of anomalies in the measurements that are taken by low-cost sensors. The basis of the method consists of converting the anomaly detection problem, which is a time serie problem, into an outlier identification problem, which is based on a point cloud. In this way, using this approach is possible to solve this problem applying the known techniques, in terms of anomaly detection. Specifically, the k-nearest neighbors method is the selected technique in this work. Moreover, a set of approaches have been proposed and validated in order to improve the performance of the selected anomaly detection algorithm.

In addition, in order to evaluate the proposed methodology, a low-cost and easily scalable wireless sensor network has been developed as a testbed, whose implementation is also described in this document. It can be also used as a controlled environment to study, develop, test, and compare new techniques for anomaly detection. The achieved results are promising, they reflect an accuracy, sensitivity and specificity higher than 95 %. For these reasons, this methodology, after the required adjustments, will be used in the *Agricultura 4.0* project to solve the original problem.

Índice

Índice de Figuras

Índice de Tablas

Índice de Códigos

1 Introducción

El presente trabajo nace a partir del proyecto *Agricultura 4.0 - Apoyo a la producción familiar campesina ecológica en Paraguay mediante una red de dispositivos inteligentes basada en IoT*, subvencionado por la Agencia Española de Cooperación Internacional para el Desarrollo (AECID).

El objetivo general de este es contribuir a mejorar los entornos productivos de la agricultura familiar campesina en Paraguay, la cual está marcada por el acceso desigual a la propiedad de la tierra y la incapacidad de competir en los mercados.

Para ello, se ha propuesto mejorar la productividad y competitividad incorporando iniciativas innovadoras de bajo impacto ambiental que optimizan y racionalizan el uso de recursos, principalmente, de recursos hídricos. La solución adoptada consiste en el despliegue de una red de dispositivos inteligentes con capacidad de medida, cómputo y comunicación, basada en tecnología del Internet de las Cosas (IoT), junto a una plataforma de telemetría para la monitorización, en tiempo real, de las condiciones del cultivo. Además, mediante algoritmos de Inteligencia Artificial y Machine Learning sobre el Big Data generado y almacenado en la nube por la red se sensores, serán predichas las necesidades reales de riego a corto y largo plazo, junto a otros aspecto del desempeño global del sistema. Por último, el desarrollo y aplicación de sistemas de control en tiempo real del riego permitirán la automatización de las bombas hidráulicas y servoválvulas, realizando un riego óptimo de acuerdo con las necesidades detectadas y los costes asociadas a la operación.

1.1 Motivación y objetivos del trabajo

Una de las características innovadoras del proyecto mencionado es la sustitución de los sistemas tradicionales de medida por otros de bajo coste que permiten incrementar el número de mediciones tanto en el espacio como en el tiempo, además de posibilitar ser usados en explotaciones más pequeñas con menos recursos económicos.

El hecho de usar sensores de bajo coste tiene consecuencias directas sobre la robustez del sistema y la fiabilidad de la información recopilada. Es por ello que nace la necesidad de desarrollar y aplicar algoritmos para paliar dichas consecuencias. Esta necesidad es la que se abarca en este texto.

Concretamente, se va a tratar la detección automática de anomalías entre las mediciones tomadas por los sensores de bajo coste, entendiendo como medición anómala aquella que no encaja o no se ajusta al comportamiento esperado [1], o bien, aquella que parece ser inconsistente con el resto de datos de los que se dispone [2].

De este modo, si suponemos un sensor de temperatura en Sevilla a mediodía, una medición de 20ºC podría ser considerada correcta si es en el mes de marzo, pero no si fuese en agosto, ya que ni sería la temperatura esperada ni se adaptaría a los datos existentes.

Nótese que el momento de la medición no es el único factor que define el contexto en el cual debe encajar la medida para que no sea considerada anómala. Otras circunstancias también deben ser tenidas en cuenta; así, siguiendo con el mismo ejemplo de la temperatura, en épocas de inestabilidad climática pueden darse diferencias de temperatura de más de una decena de grados entre días consecutivos, por lo que la radiación solar, la cantidad de precipitaciones, la humedad relativa u otras variables podrían ser requeridas para precisar el contexto de la medición y mejorar la detección.

1.2 Solución propuesta

La metodología para detectar anomalías entre las mediciones tomadas por sensores de bajo coste constituye el contenido principal de este texto. Nuestra propuesta, en primera instancia, transforma en una nube de puntos la serie temporal obtenida a partir de las mediciones de los sensores. Para ello, aprovecha la variación estacional, también conocida como variación cíclica regular, que esta última presenta: divide todas las mediciones en periodos de variación regular y los superpone sin desfasar, consiguiendo que todas las mediciones que estén separadas el mismo tiempo respecto al inicio de su periodo se sitúen donde mismo en el nuevo eje temporal derivado de la transformación.

Una vez obtenida la nube, el problema de identificar anomalías se convierte en un problema de detección de outliers, por lo que las técnicas usadas para tal fin pueden ser aplicadas en este escenario. Se tratará concretamente con el método de los k vecinos más cercanos, al cual se le plantearán modificaciones y mejoras para incrementar su desempeño, tales como: la modificación del cálculo de la distancia, la determinación automática del parámetro k y la actualización periódica del modelo, entre otras que más adelante se desarrollarán.

Con el objetivo de validar el algoritmo de detección de anomalías anteriormente mencionado, así como las mejoras propuestas para adaptarlo al problema planteado en *Agricultura 4.0*, se ha diseñado y desplegado una red de sensores, en adelante llamado banco de pruebas. En la figura 1.1 se refleja la composición del sistema, los nodos recogen los datos proporcionados por los sensores y los envía a la base de datos centralizada para almacenarlos.

Los elementos principales del banco de prueba son dos:

- **Nodos**: Cada uno es un ordenador de placa única, concretamente, una Raspberry Pi. Esta registra las mediciones de temperatura y humedad relativa tomadas por los sensores que se les ha conectado y, posteriormente, las almacena en una base de datos local propia o las envía a la base de datos centralizada.

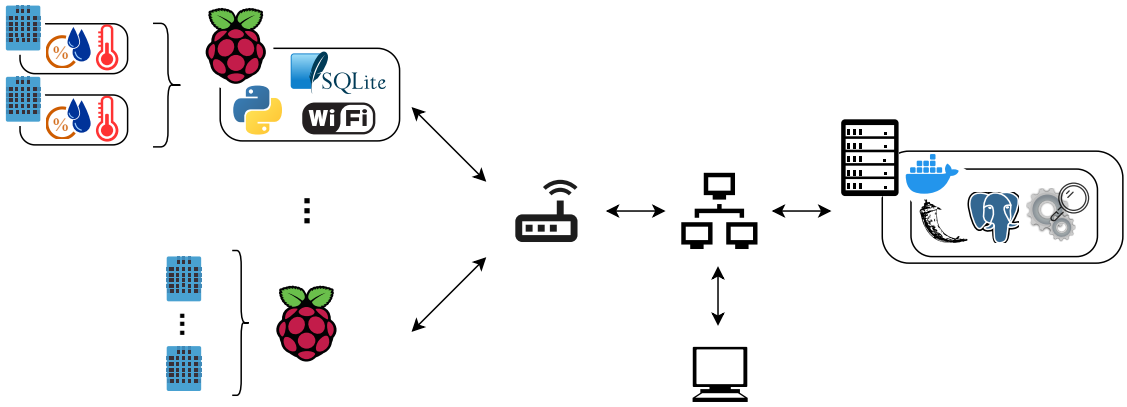

Figura 1.1 Representación esquemática del sistema desarrollado.

- **Servidor**: En este se ejecuta, gracias a la tecnología de Docker [3], dos contenedores virtuales:
  - Un contenedor comprende la base de datos centralizada en la que se almacenan todas las mediciones registradas por la red de sensores. Además, también ejecuta una API de desarrollo propio para facilitar y restringir el acceso, tanto en escritura como en lectura, a la base de datos. De esta manera, los nodos pueden enviar sus datos para que sean almacenados y, también, otras aplicaciones pueden hacer uso de ellos.
  - El otro contenedor ejecuta de forma ininterrumpida el algoritmo de detección de anomalías. Este último también accede a los datos registrados y guarda los resultados generados por medio de la API descrita anteriormente.

1.3 Contenido del trabajo

Tras presentar el contexto de este trabajo, los objetivos que persigue y una breve presentación sobre la solución adoptada, el contenido restante del documento sigue la siguiente estructura.

En el capítulo 2 se expone una concisa revisión sobre diferentes técnicas y aproximaciones para resolver el problema de la detección de anomalías. En el capítulo 3 se describe la red de sensores desplegada para recopilar los datos de temperatura y humedad con los que se ha puesto a prueba nuestra metodología para detectar anomalías desarrollada en el capítulo 4. En el capítulo 5 se muestran los resultados obtenidos con la metodología propuesta. En el capítulo 6 se describe cómo se ha integrado la herramienta de detección, propuesta en el capítulo 4, con la red de sensores y los servicios de comunicaciones y de almacenamiento de datos. Por último, en los capítulo 7 y 8, se recogen las conclusiones finales derivadas de la realización del trabajo, junto a las lecciones aprendidas y las posibles líneas de investigación a continuar en el futuro, respectivamente.

2 Estado del arte

La detección de anomalías ha sido ampliamente estudiada y aplicada a lo largo de los años en un gran número de campos y ramas del conocimiento. En seguridad informática se usa para identificar intentos de intrusión, acceso indebido a la información o extraños patrones de tráfico en una red [4, 5]. En ingeniería estructural se emplea para filtrar y limpiar las grandes cantidades de datos que producen los sistemas de monitorización de la salud estructural en las infraestructuras civiles [6, 7]. En medicina se utiliza en el análisis de imágenes médicas para asistir en el reconocimiento de enfermedades y patologías [8, 9]. Como último ejemplo, en el dominio financiero se aplica para la detección de fraudes financieros de varios tipos [10].

En el estudio sobre la detección de anomalías realizado por Chandola [1] se pueden consultar múltiples clasificaciones de las técnicas empleadas: según si el resultado que proporcionan es una clasificación de todas las observaciones indicando cuáles son anómalas o cuáles no, o bien, si el resultado es un valor numérico que cuantifica la anormalidad de cada observación; según si la detección es supervisada, semisupervisada o no supervisada; según si los datos tienen alguna dependencia espacio-temporal o no; según si las anomalías son puntuales y por ello pueden considerarse anómalas con respecto al resto de datos, o bien son contextuales y también se debe tener en cuenta su contexto. No obstante, la que más desarrolla el autor es la que divide las técnicas según sus bases:

- **Técnicas basadas en clasificación**: Durante la etapa de entrenamiento generan un clasificador a partir de un conjunto de observaciones previamente etiquetadas. Luego, el clasificador es usado para asignar una de las clases existentes a cada nueva muestra. En el contexto de detección de anomalías, las clases son normal y anormal. Estas técnicas se pueden subdividir en las que se basan en redes neuronales, en redes bayesianas, en las máquinas de vectores de soporte y en un conjunto de reglas.
  - Entre ellas: Multi layered Perceptrons, Neural Trees, Radial Basis Function-Based (RBF)...
- **Técnicas basadas en la vecindad**: Se fundamentan en el cálculo de distancias. Por un lado, están las que estudian la densidad relativa del vecindario de cada observación para evaluar su anormalidad y, por otro lado, las que lo hacen aprovechando las distancias a sus vecinos. En este último grupo está contenida nuestra aproximación.
  - Entre ellas: k-Nearest Neighbours (kNN), Local Outlier Factor (LOF)...

- **Técnicas basadas en clustering**: Estas técnicas agrupan en conjuntos las observaciones que son similares entre sí. Se dividen en tres según la naturaleza de las anomalías que se presuma: los datos normales sí forman conjuntos, pero las anomalías no pertenecen a ninguno; los datos normales se encuentran cerca del centro del conjunto, no siendo así para las anomalías; los datos no anómalos forman conjuntos densos y poblados, mientras que las anomalías forman sus propios conjuntos que son pequeños y dispersos.

  - Entre ellas: Self-Organizing Maps (SOM), k-Means, Expectation Maximization (EM)...

- **Técnicas basadas en estadística**: Generan un modelo probabilístico a partir de la datos disponibles y, posteriormente, determinan si un dato nuevo es anómalo o no en función de si era suficientemente probable que ocurriera según el modelo inferido previamente. Se dividen en técnicas paramétricas y no paramétricas según se tenga conocimiento o no de la distribución estadística real del modelo.

  - Entre ellas: Gaussian Model-Based, Regression Model-Based, Kernel Function-Based...

- **Técnicas basadas en la teoría de la información**: Analizan la información contenida en los datos y detectan como anómalos aquellos que introducen información discordante. Herramientas como la Complejidad de Kolmogórov son las usadas para analizar la información.

  - Entre ellas: Local Search Algorithm (LSA), Entropy-Based...

- **Técnicas basadas en el espectro**: Se basan en agrupar características de los datos para trasladarlos a un espacio de menor dimensión en el cual se amplifique o se haga más evidente la diferencia entre datos anómalos y no anómalos.

  - Entre ellas: Principal Component Analysis (PCA), Compact Matrix Decomposition (CMD)...

Para el caso concreto de la detección de anomalías en sensores, en [11] se propone una técnica denominada Long Short Term Memory Networks based Encoder-Decoder (EncDec-AD). En esta, un modelo aprende de la serie temporal registrada por el sensor y, luego, es capaz de continuarla para instantes futuros. El error entre el valor medido y el esperado es usado para detectar las anomalías. Los autores exponen que esta técnica es robusta y puede detectar anomalías en series temporales de cualquier longitud que sean predecibles, impredecibles, periódicas, no periódicas o quasi-periódicas.

En [12] se describe un método en dos fases capaz de hacer frente a situaciones en las que los sensores generan un gran volumen de datos. En la primera fase, aplican un algoritmo de detección basado en la propia medición, el cual es menos preciso, pero consume menos recursos y es más rápido, por lo que es apto para la detección en tiempo real. Luego, en el postprocesamiento de los datos que constituye la segunda fase, hacen uso de un algoritmo basado en el contexto de la medición, que es mucho más lento, pero aporta mayor precisión a la detección. Según los autores, los resultados preliminares son prometedores tanto en entornos simulados como en escenarios reales.

En [13] se estudia la detección de anomalías en los datos registrados por una red inalámbrica de sensores cuya fiabilidad está comprometida por las duras condiciones ambientales, las interferencias electromagnéticas y la baja calidad de los sensores. Para ello, hacen uso de una red bayesiana con la que capturan las correlaciones espacio temporales entre las observaciones de un sensor y sus vecinos, con el objetivo de detectar anomalías o datos ausentes.

Como último ejemplo, se presenta la técnica denominada Segmented Sequence Analysis (SSA) [14], la cual resuelve la detección de anomalías mediante la identificación de patrones inusuales en la serie temporal obtenida de las mediciones, haciendo así posible la detección de anomalías de larga duración. La base del método reside en comparar las medidas tomadas con una serie temporal de referencia previamente calculada, y marcar como anormales los tramos en los que la disimilitud entre ambas sea relevante. Las pruebas realizadas concluyen que esta aproximación, la cual puede ser aplicada casi en tiempo real, es eficiente, robusta y presenta buena precisión en datos reales.

3 Adquisición de datos

En la mayor parte de desarrollos que involucran aprendizaje automático, incluida la propuesta en este trabajo para la detección de anomalías en sensores de bajo coste, se tiene como factor común la necesidad de disponer de datos. Es decir, tener acceso a información recabada de hechos pasados de la misma naturaleza que el problema tratado. Esto se debe a que el aprovechamiento de dichos datos nos ayuda a generar conocimiento para dar lugar a nuevos datos, extraer conclusiones, tomar decisiones o crear nuevos recursos. En definitiva, a resolver el problema y validar la solución alcanzada.

En nuestra aplicación, estos datos no son más que el histórico de las mediciones pasadas tomadas por los sensores de coste reducido bajo estudio, u otros similares, a lo largo del tiempo. Así pues, las magnitudes registradas serán diferentes según la índole del sensor, por ejemplo: grados en termómetros, velocidad en anemómetros y adimensionales en higrómetros. A día de hoy es fácil acceder a datos de este tipo, como el registro de la temperatura media diaria en más de 300 ciudades desde 1995 que ofrece la universidad de Dayton [15], o el registro de diversas variables meteorológicas que ofrece la AEMET mediante su plataforma OpenData [16].

No obstante, para dar solución al problema planteado en *Agricultura 4.0*, se ha optado por construir un sistema que simule al real y, por ello, se ha diseñado e implementado un sistema de adquisición de datos que, además, será usado en el problema real. De esta manera, se consigue acercar el detector de anomalías a su aplicación final, la detección en sensores de bajo coste, así como ponerlo a prueba y validar o rechazar sus resultados. Además, se logra total libertad en cuanto a las características de los datos tomados, como la frecuencia de muestreo, las magnitudes medidas, la localización física de las mediciones, las condiciones ambientales, e incluso, la inserción de registros anómalos mediante la alteración física del sensor o su entorno.

3.1 Banco de pruebas

Para elaborar el sistema de adquisición de datos se ha primado, entre otros factores, el bajo coste, la disponibilidad y reutilización de los componentes, y el uso de software gratuito. El resultado se ilustra de forma esquemática en la figura previa 1.1, donde se puede comprobar que existen tres componentes bien diferenciados: los nodos, a la izquierda; las comunicaciones, en el centro; y el almacenamiento de datos, entre otros elementos, a la derecha. Los tres se describen en mayor profundidad en los siguientes apartados.

3.1.1 Nodos de la red de sensores

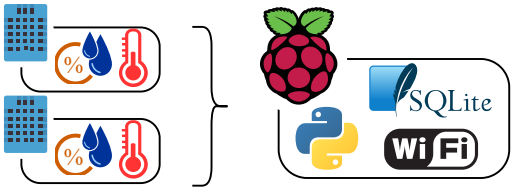

Figura 3.1 Nodo de la red desplegada.

Implementación física

Los nodos, figura 3.1, son los elementos principales de medición. Cada uno de ellos está constituido fundamentalmente por una Raspberry Pi. Estos ordenadores de placa única y tamaño reducido ejecutan todos los programas que son necesarios para dotar de la funcionalidad requerida al nodo. Además, mediante sus pines de conexión de propósito general, gestionan los sensores utilizados para la monitorización de las magnitudes físicas de interés, las cuales pueden ser cualquiera además de la temperatura y humedad relativa que se representan en la figura, ya que la arquitectura se ha diseñado para que sea lo más escalable posible.

El modelo de Raspberry Pi empleado es el 3A+ [17], aunque todos los demás también cumplen con los requisitos del proyecto y pueden ser usados indistintamente. Por otro lado, los sensores utilizados son los integrados en el módulo DHT11 [18], el cual incluye elementos de tipo resistivo para la medida de la humedad relativa y temperatura. Sus características se recogen en la tabla 3.1.

Tabla 3.1 Especificaciones módulo DHT11.

Parámetro	**Valor**	
	Temperatura	Humedad
Rango	0 ºC - 50 ºC	20 %RH - 90 %RH
Resolución	1 ºC	1 %RH
Precisión	± 5 ºC	± 2 %RH
Repetibilidad	± 1 ºC	± 1 %RH

La principal ventaja de este módulo es que integra todos los elementos electrónicos necesarios para hacer uso de sus sensores. Esto permite que se pueda conectar directamente a los pines de alimentación que dispone la Raspberry Pi para dicho fin y, además, gracias a que incorpora un microprocesador, se puede acceder a los valores registrados de temperatura y humedad de forma sencilla con otro pin mediante una comunicación digital secuencial. A modo de ejemplo, se ilustra en la figura 3.2 la conexión de dos módulos sensores.

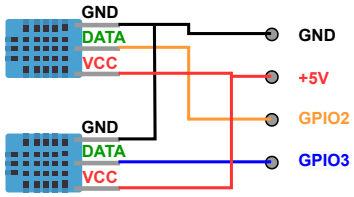

Figura 3.2 Ejemplo de conexionado de dos módulos DHT11 en una Raspberry Pi.

Programación de los nodos

A continuación, una vez descrita la implementación física de los nodos, se procede a describir la programación de la funcionalidad requerida de estos dispositivos.

Se ha empleado el lenguaje de programación Python, el cual es un lenguaje interpretado multiparadigma que está respaldado por una gran comunidad. Esto último es de vital importancia, puesto que es posible descargar y hacer uso de numerosas aportaciones, en forma de paquetes de código abierto, que facilitan la realización de diversas tareas. Gracias al uso de algunos de estos paquete se ha simplificado en gran medida el desarrollo llevado a cabo, siendo los más relevantes mencionados a lo largo de este texto.

El programa escrito para ser desplegado en cada nodo realiza la siguiente operación, figura 3.3. Primero lee un archivo de configuración para capturar diferentes parámetros que definen el funcionamiento del nodo y, luego, lanza un proceso de ejecución por cada módulo sensor conectado, lado izquierdo de la figura. La función que ejecuta cada proceso es la misma: lee los valores de temperatura y humedad registrados por el sensor al que está asociado, los envía por HTTP y, si la transmisión es correcta, espera un tiempo predefinido para volver a registrar y enviar nuevas mediciones. En caso contrario, almacena los datos en una base de datos local SQLite3. Por último, en el momento que un intento futuro de transmisión tenga éxito, se enviarán también todos los registros almacenados que pudiera haber, véase el lado derecho de la figura. Para mayor claridad, se incluye el pseudocódigo 3.1 con el proceso de inicialización, el 3.2 con el proceso de medición de cada sensor y el 3.3 con el proceso de volcado de la base de datos local.

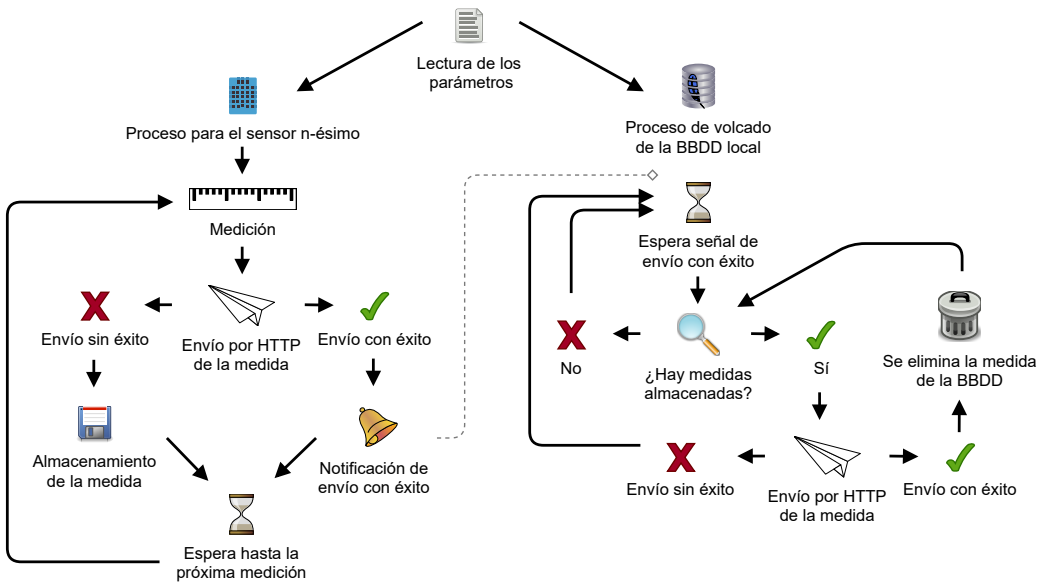

Figura 3.3 Diagrama de flujo de la funcionalidad de los nodos.

Código 3.1 Pseudocódigo del programa principal ejecutado en los nodos.

```
Leer los parámetros del archivo de configuración
Establecer la conexión con los sensores
Ejecutar el proceso para el volcado de la base de datos en local
Para cada uno de los módulos conectados:
   Ejecutar el proceso de medición del módulo sensor
Esperar a la finalización de los procesos de medición
Detener de forma segura el proceso de volcado
Cerrar la conexión con los sensores
```

Código 3.2 Pseudocódigo del programa principal ejecutado en los nodos.

```
Mientras no se alcance la fecha final de medición establecida:
   Leer la temperatura y humedad registrada por el sensor
   Enviar los datos por HTTP
   Si el envío tiene éxito:
      Notificar al proceso de volcado de la base de datos local
   Si el envío no tiene éxito:
      Almacenar la medición en la base de datos local
   Esperar el tiempo definido entre mediciones
```

Código 3.3 Pseudocódigo del programa principal ejecutado en los nodos.

```
Hacer siempre:
   Esperar notificación de envío con éxito
   Si hay datos almacenados:
      Por cada dato almacenado:
         Enviar dato por HTTP
         Si el envío tiene éxito:
            Eliminar el registro de la basa de datos local
         Si el envío no tiene éxito:
            Interrumpir ejecución del bucle interno
```

Un aspecto importante es que no solo se envían las mediciones tomadas por los sensores, sino que también se transmite una marca de tiempo del instante en el que se tomaron, junto a identificadores del nodo y del módulo sensor. Esto es especialmente útil, ya que facilita el posterior análisis y explotación de los datos.

La división de la ejecución del programa en múltiples procesos comunicados entre ellos se consigue con los módulos *multiprocessing* [19] y *signal* [20]. Gracias a esta estructura se logra que el tratamiento por separado de cada dispositivo DHT11, junto a la transmisión de los datos almacenados, no interfieran entre sí, favoreciendo que todas las mediciones se realicen en intervalos regulares. Además, si uno de los sensores dejase de funcionar correctamente, tampoco afectaría al resto de elementos.

En cuanto a la base de datos local, los principales motivos por los que se ha empleado una SQLite son los siguientes: su reducida demanda de recursos computacionales, sus características intrínsecas de atomicidad, consistencia, aislamiento y durabilidad, y su compatibilidad con aplicaciones multiprocesos. Para la lectura y escritura de los datos se ha empleado el módulo *sqlite3* [21], el cual permite una sencilla interacción con la base de datos mediante el lenguaje SQL.

Para terminar, se hace especial mención al paquete de Python *adafruit_dht* [22], puesto que establece una capa de abstracción entre el módulo DHT11 y la Raspberry Pi que facilita la integración de ambos sin tener que lidiar explícitamente con la comunicación de tipo secuencial entre dispositivos.

Configuración de los nodos

La configuración del nodo se divide en dos etapas. La primera de ellas se centra en disponer al sistema operativo de la Raspberry Pi de todos los recursos, configuraciones y dependencias necesarias para ejecutar la aplicación desarrollada. Esto incluye, por ejemplo, la instalación de los paquetes requeridos por el sistema, el intérprete de Python y los módulos usados, además de la configuración de la conexión inalámbrica por WIFI y el inicio automático de la aplicación de medición cada vez que el dispositivo se inicia.

La segunda etapa está relacionada con la configuración de la propia aplicación de medida. Para ello, se dispone de un fichero de configuración editable por el usuario que permite personalizar el comportamiento de cada nodo sin tener que modificar el código fuente. Entre las variables que permite editar se encuentran: el identificador del nodo, el identificador de cada uno de los módulos sensores conectados, el intervalo de muestreo, la fecha final de medición y la dirección a la cual se envían los datos.

3.1.2 Almacenamiento de datos

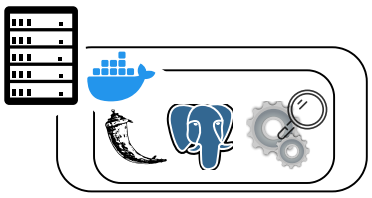

Figura 3.4 Arquitectura del servidor central.

Respecto a la persistencia de los datos registrados, ya se ha comentado previamente como cada nodo incluye una base de datos local SQLite3. No obstante, este almacenamiento es de carácter temporal y su principal objetivo es emular una cola no volátil. Haciendo uso de ella se consigue que las magnitudes físicas medidas no se pierdan cuando no existe conexión a la red de área local, no se puedan enviar los datos o, incluso, cuando al nodo se le retira la alimentación eléctrica.

Así pues, para el almacenamiento permanente y centralizado de los datos se hace uso del sistema de gestión de bases de datos relacional orientado a objetos PostgreSQL. Este se ejecuta en un servidor permanentemente conectado a la misma red de área local que usan los nodos, por lo que todos ellos tienen acceso directo a él para poder guardar sus medidas. Su estructura de datos se puede consultar en el diagrama entidad-relación del anexo A.

En esta base de datos se tiene una tabla para guardar los datos brutos, es decir, tal como son recibidos por parte de los nodos. El empleo de identificadores de nodo y sensor es lo que hace posible usar una

única tabla, ya que permiten la trazabilidad de los datos. Además, el uso de estos también facilita la escalabilidad del sistema, puesto que mientras ninguno sea repetido, la base de datos no supondrá ningún problema a la hora de ampliar la red de nodos. En la tabla 3.2 se muestra un extracto de los datos registrados.

Tabla 3.2 Tabla de datos brutos.

id measurement	id rp	id sensor	temperature	humidity	measurement date	save date
2019	2	2	24	46	2021-04-28 11:26:03.67	2021-04-28 11:26:03.72
2020	45	2	1	23	2021-04-28 11:26:03.67	2021-04-28 11:26:03.76
2021	2	2	24	46	2021-04-28 11:28:45.61	2021-04-28 11:28:45.66
2022	48	1	1	21	2021-04-28 11:29:53.81	2021-04-28 11:29:53.94
2021	2	2	24	46	2021-04-28 11:28:45.61	2021-04-28 11:28:45.66
2023	45	2	1	23	2021-04-28 11:28:45.61	2021-04-28 11:28:45.67

Se puede comprobar como cada fila de la tabla, es decir, cada registro, es asociado a un identificador único formado por un entero secuencial. Es importante mencionar que este número no es generado por el nodo, el cual es totalmente ajeno a estos valores, sino que es determinado por la propia base de datos.

Asimismo, en la base de datos existen otras tres tablas generadas a partir de la anterior. La primera de ellas almacena los datos resultantes de un preprocesamiento básico a modo de limpieza de los datos brutos. Su contenido es prácticamente igual a la tabla original excepto por la siguiente salvedad: se borran las entradas duplicadas si las hubiese, y se detectan y rellenan intervalos de medición no registrados. Para realizar esto último se sigue una sencilla regla, si no se tiene constancia de hasta 4 registros consecutivos, se completan con la media aritmética de los valores medidos en los dos registros frontera del hueco detectado. Por contra, si el número es mayor a 4, los registros se completan con un valor igual a NULL.

Las dos tablas restantes son agregaciones calculadas a partir de los resultados del preprocesamiento. En ellas se recoge la media y desviación típica de las magnitudes medidas, en una tabla con granularidad horaria y en la otra diaria. En las tablas 3.3 y 3.4 se recogen los campos más relevantes de ambas, quedando excluidos otros como el número de mediciones que engloba cada agregación o el número de huecos en dicho intervalo.

3.1.3 Comunicaciones

Todo los elementos que componen el sistema están comunicados entre sí mediante una red de área local. Los nodos hacen uso de la tarjeta de red inalámbrica WIFI que integran para conectarse a cualquiera de los puntos de acceso que extienden la red. Por otro lado, el servidor que ejecuta la base de datos PostgreSQL se conecta mediante el cableado Ethernet de la misma red.

Tabla 3.3 Tabla de agregaciones horarias.

id aggregation	id rp	id sensor	temperature mean	temperature std deviation	humidity mean	humidity std deviation	hour	date
537	1	1	21.000	0.000	42.750	0.452	7	2021-05-10
538	1	1	21.833	0.389	41.000	0.000	8	2021-05-10
606	2	1	22.250	0.452	40.500	0.798	7	2021-05-10
607	2	1	25.500	0.798	36.750	1.055	8	2021-05-10
675	2	2	25.167	0.389	37.833	0.389	7	2021-05-10
676	2	2	26.833	0.577	35.333	0.778	8	2021-05-10

Tabla 3.4 Tabla de agregaciones diarias.

id aggregation	id rp	id sensor	temperature mean	temperature std deviation	humidity mean	humidity std deviation	date
2	1	1	21.451	4.326	33.191	7.528	2021-05-01
3	1	1	22.829	5.176	31.794	9.497	2021-05-02
8	2	1	22.660	2.345	31.944	3.802	2021-05-01
9	2	1	22.583	2.321	32.285	5.352	2021-05-02
14	2	2	24.455	1.500	30.892	2.567	2021-05-01
15	2	2	24.243	1.428	31.174	3.803	2021-05-02

3.1.4 Herramienta de extracción de datos

Para facilitar el intercambio de datos entre los diferentes elementos se ha desarrollado una API basada en la tecnología RESTful, la cual es ejecutada en el mismo servidor que la base de datos. Estas siglas provienen del inglés de *interfaz de programación de aplicaciones* y de *transferencia de estado representacional*, respectivamente.

Sobre este concepto y su desarrollo se puede leer en la obra de Massé [23]. En ella, el autor describe una API como un intermediario que continuamente escucha y responde las peticiones de parte del cliente a los servicios web, entendiendo estos últimos como los recursos que tienen el propósito de cubrir las necesidades de los sitios web u otras aplicaciones. Con este fin, la API pone a disposición un conjunto de datos y funciones con el objetivo de facilitar la interacción entre programas informáticos que intercambian información.

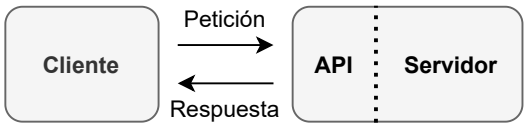

Figura 3.5 Esquema cliente-servidor con una API como figura intermediaria.

Massé continúa exponiendo REST como el término acuñado por Fielding [24] para su propia propuesta de arquitectura web que comúnmente es aplicada al diseño de APIs, de ahí el término REST API. En dicha arquitectura, los elementos que pone a disposición la API se conocen como recursos, y estos son accedidos utilizando un identificador de recurso uniforme, es decir, un endpoint, el cual no es más que una dirección web o URI. Además, los componentes de la red que interactúan,

cliente y servidor, se comunican generalmente mediante el protocolo de transferencia de hipertexto, HTTP, a través del cual se intercambian los recursos haciendo uso de diferentes representaciones, siendo los formatos HTML, XML y JSON los más empleados.

Una definición más rigurosa se puede leer en la disertación de Fielding [24], donde se recogen los 5 principios de esta arquitectura que se han intentado condensar en el párrafo anterior: arquitectura cliente-servidor, ausencia de estado, habilitación y uso de la caché, sistema dividido por capas e interfaz uniforme.

En el contexto de este trabajo, la API permite aislar la base de datos centralizada de los nodos de medición, ya que ninguna de las partes tiene conocimiento de cómo está implementada la otra. Esto facilita y aporta flexibilidad al desarrollo, puesto que las modificaciones y evoluciones que puedan darse en alguna de las partes no interfieren en la otra, siempre que ambas acaten el funcionamiento de la API que actúa de intermediaria.

Un ejemplo de mensaje que envía cualquier nodo al servidor a través de la API para almacenar permanentemente una medición se recoge en la figura 3.6.

```
{
    "id_rp": 1,
    "id_sensor": 2,
    "temperature": 28.2,
    "humidity": 42,
    "measured_date": "2021-06-21T14:28:14.23"
}
```

Figura 3.6 Ejemplo de JSON enviado por el nodo.

Este mensaje es enviado en el cuerpo de un mensaje HTTP hacia un endpoint configurado para tal fin en la API, en este caso concreto, la dirección es: *http://IP:PUERTO/Raw/Record/*. Una vez que es recibido, esta procesa el mensaje y le aplica las transformaciones necesarias para almacenarlo en la base de datos; por ejemplo, le añade una marca de tiempo con el instante en el que se guarda. Por último, le envía una respuesta al nodo para que este tenga constancia de que los datos han sido, o no, almacenados con éxito.

La programación de la API se ha realizado también en Python mediante el módulo *Flask* [25]. Este permite un desarrollo sencillo y ágil, en el que por medio de decoradores se pueden asignar endpoints a cualquier función. De esta manera, cuando un mensaje es enviado a la URI asociada a dicho endpoint, la función es ejecutada recibiendo el cuerpo del mensaje y su resultado enviado como respuesta.

Por otro lado, se ha empleado el módulo *SQLalchemy* [26] para llevar a cabo las operaciones sobre la base de datos. Este incorpora una utilidad ORM, mapeo objeto-relacional, que permite tratar las tablas de la base de datos como si fuesen objetos del programa: cada tabla se corresponde con una clase, la cual tiene un atributo por cada columna. De esta manera, cada fila almacenada se representa mediante una instancia de la clase correspondiente. Esta arquitectura es uno de los motivos principales de la gran flexibilidad que aporta este módulo, dando lugar a una serie de ventajas como, por ejemplo, evitar el uso explícito de sentencias SQL, y añadir una capa de abstracción que hace que la explotación de la base de datos sea independiente de cuál se use concretamente.

Se han aprovechado las inmensurables posibilidades que aporta el uso de ambos módulos en la programación de APIs para ampliar la de este proyecto. De esta manera, no se emplea solo para almacenar los datos registrados por los nodos, sino también para la lectura de estos. Así, cualquier aplicación o algoritmo para la visualización o explotación de los datos puede hacer uso de algún endpoint de ella para recuperar la información. Los diferentes puntos de acceso se han implementado según surgía la necesidad, siendo uno de ellos el que permite recuperar los datos brutos en forma de JSON: *http://IP:PUERTO/Raw/Read/<id_rp>?id_sensor=*&sdate=*&edate=**. Este debe recibir obligatoriamente un identificador de nodo para hacer un filtrado por él y, opcionalmente, un identificador del sensor y los límites temporales de búsqueda para realizar otros filtros adicionales.

Para terminar, cabe destacar la capa de seguridad que aporta el uso de la API como intermediaria, puesto que esta define y limita cómo los datos son recuperados, modificados o almacenados; quedando así restringidas las operaciones que otros dispositivos de la red pueden realizar sobre la base de datos.

3.2 Validación de la solución

A modo de validación del sistema de adquisición desarrollado se representan a continuación las mediciones de temperatura y humedad tomadas por dos nodos situados en estancias diferentes de una oficina. Uno posee un único módulo sensor y el otro dos.

Los datos mostrados corresponden con los registrados entre el 01/05/2021 y el 15/05/2021 con frecuencia de muestreo de 5 minutos, además de los agregados por horas y días.

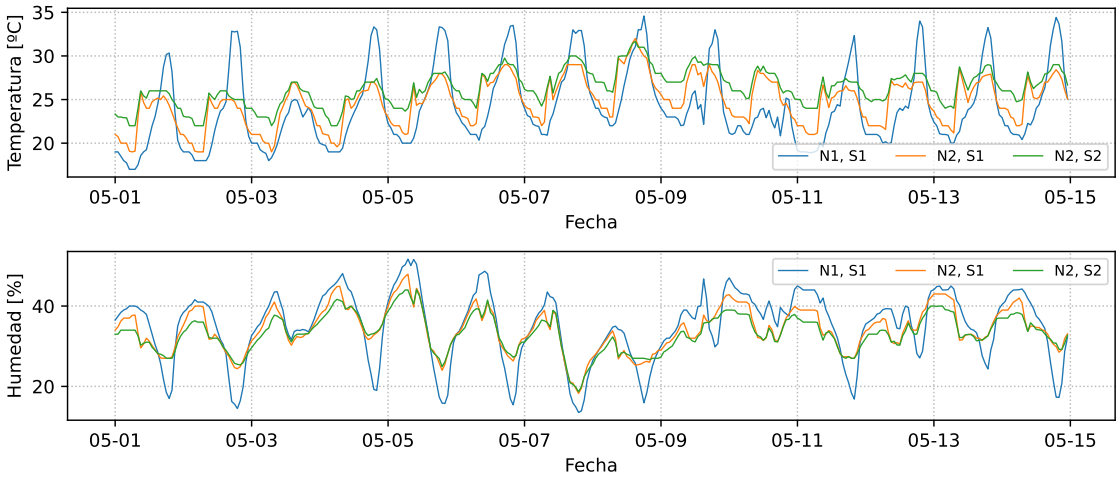

Figura 3.7 Datos brutos registrados por el sistema de adquisición.

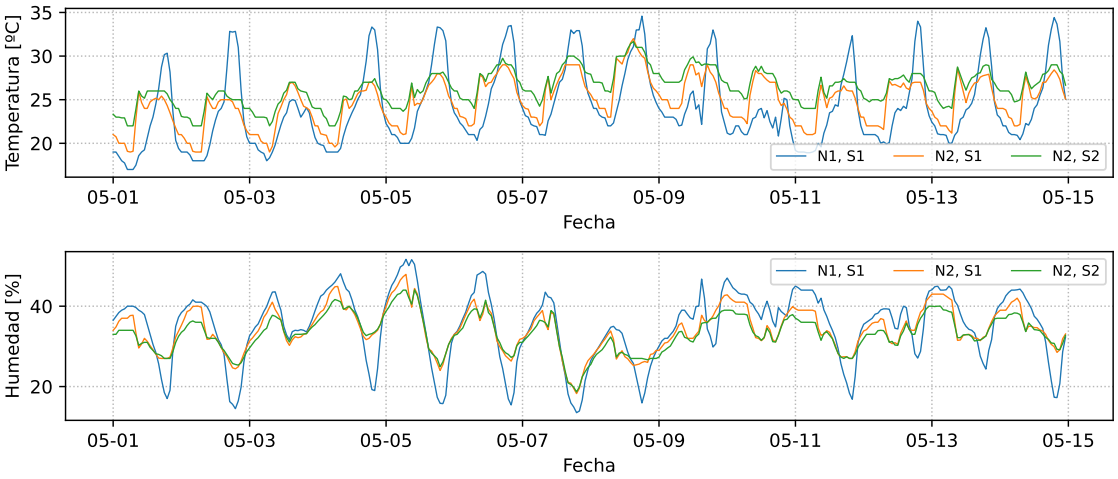

Figura 3.8 Datos registrados agregados por horas.

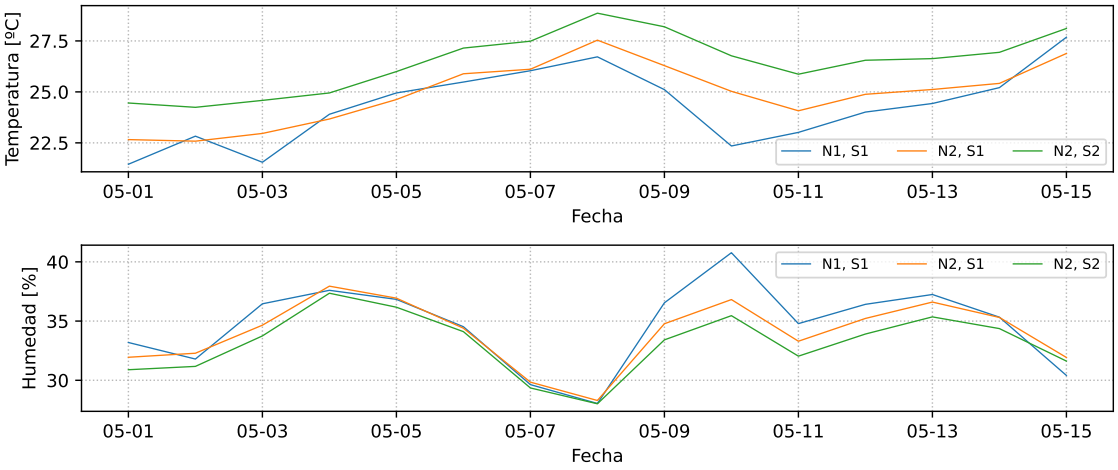

Figura 3.9 Datos registrados agregados por días.

4 Metodología para la detección de anomalías

En este capítulo se recogen los fundamentos de la metodología propuesta para la detección de anomalías, mediante la cual será posible detectar mediciones anómalas en sensores de bajo coste, es decir, mediciones fuera de lo común, o no esperadas, en el contexto en el que se dan. De este modo, se logra mejorar la fiabilidad de las mediciones de estos dispositivos, ya que permite realizar un filtrado de los datos según sean normales o no, lo que también contribuye a mejorar el posterior aprovechamiento de estos por otros algoritmos. Primero se va a exponer la base sobre la que se construye la herramienta de detección junto al método de lo k vecinos más cercanos y, posteriormente, una serie de modificaciones y ampliaciones que se le han realizado para tratar de mejorar su desempeño.

4.1 Bases de la metodología

El objetivo principal del proyecto, la detección de anomalías en sensores de bajo coste, fija la naturaleza de serie temporal que tendrán los datos con los que se va a tratar. Así, independientemente del tipo de sensor que se use, se tendrá una sucesión ordenada de valores indexados mediante marcas de tiempo.

En otras palabras, cada medición del sensor da lugar a un punto, x, contenido en un espacio vectorial de como mínimo dimensión dos: la dimensión de la magnitud medida y la del tiempo. Nótese que esto mismo aplica aunque el sensor mida más de una magnitud física y dé lugar a un espacio vectorial de mayor dimensión, puesto que los puntos resultantes pueden ser descompuestos.

$$x_t = (\text{magnitud física},\ \text{tiempo}), \quad x_t \in \mathscr{R}^2 \tag{4.1}$$

$$x'_t = (\text{magnitud física 1},\ \text{magnitud física 2},\ \text{tiempo}), \quad x'_t \in \mathscr{R}^3 \tag{4.2}$$

$$x'_t \quad \longleftrightarrow \quad \begin{array}{l} x'_{t,1} = (\text{magnitud física 1},\ \text{tiempo}), \quad x'_{t,1} \in \mathscr{R}^2 \\ x'_{t,2} = (\text{magnitud física 2},\ \text{tiempo}), \quad x'_{t,2} \in \mathscr{R}^2 \end{array} \tag{4.3}$$

Normalmente, trabajar con datos en forma de serie temporal añade una extra de dificultad. Por este motivo, el punto de partida imprescindible de la metodología propuesta consiste en una transformación inicial de los datos de partida, la cual tiene como objetivo eliminar de forma parcial la vinculación de estos con el tiempo sin perder la información que la variable temporal aporta. Para ello, dicha transformación se vale de la variación cíclica regular que puedan presentar los valores de la serie temporal, es decir, la componente que se repite de manera periódica cada cierto tiempo. Un ejemplo de serie temporal con variación cíclica es el volumen de tráfico, ya que su evolución a lo largo del día sigue siempre un patrón similar.

La transformación mencionada consiste en dividir la serie temporal en ciclos regulares de variación, para luego superponerlos sin desfase y conformar una nube de puntos sin ningún orden temporal. De esta manera, reutilizando el ejemplo anterior sobre el tráfico del cual se sabe que tiene un periodo de 24 horas, la transformación consistiría básicamente en eliminar las cifras más significativas que las horas en la variable tiempo, esto es, eliminar el día, mes y año: 2021-06-27 13:20:15 pasaría a ser 13:20:15.

En la figura 4.1 se muestran dos gráficas para clarificar lo expuesto previamente. A la izquierda se tiene un fragmento de la evolución de la temperatura registrada por uno de los nodos, pudiendo comprobarse como esta tiene una variación regular cada día. A la derecha se representan los mismos datos tras aplicarle la transformación, donde ya no presentan ninguna sucesión temporal y forman una nube de puntos. El nuevo eje de abscisas recoge la hora del día en la que se tomó cada medición, por lo que todos los registros tomados a la misma hora en diferentes días están alineados verticalmente.

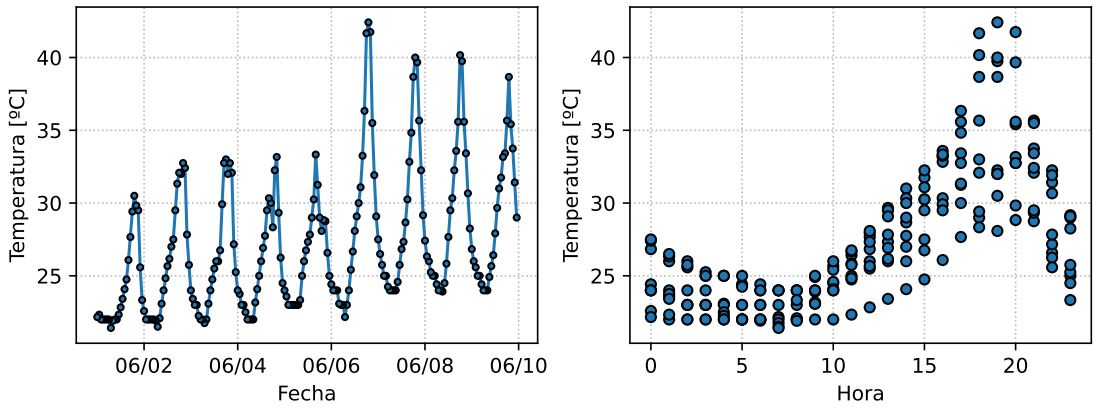

Figura 4.1 Serie temporal de la temperatura antes y después de la transformación.

Como ya se ha mencionado, el objetivo de la transformación es dar lugar a una nube de puntos, entendiendo esta como una aglomeración en el espacio vectorial. La obtención de dicha nube es la base de esta metodología, puesto que permite convertir el problema de detección anomalías en un problema de detección de outliers, es decir, valores atípicos que son numéricamente distantes del resto de datos. Gracias a esto, se abre la posibilidad de utilizar las herramientas conocidas en la detección de outliers para la detección de anomalías en series temporales.

Un hecho a tener en cuenta es que, la detección de outliers y, por tanto, de anomalías, arrojará mejores resultados mientras mejor definida esté la nube de puntos. Asimismo, como dicha nube proviene de la serie temporal original, la detección será mejor, cuanto más marcada sea la variación cíclica regular. Si los datos no siguen un patrón a lo largo del tiempo, no se puede llevar a cabo una superposición que dé lugar a una nube y, por consiguiente, no se pueden detectar anomalías siguiendo esta metodología.

Las técnicas para la detección de outliers son muchas y variadas, entre ellas cabe destacar: Local Outlier Factor (LOF) [27], Angle-Based Outlier Detection (ABOD) [28], Feature Bagging [29], etc. No obstante, la que se va a aplicar en este trabajo es conocida por el nombre de los k vecinos más cercanos, k-NN, y su elección está motivada por lo sencilla e intuitiva que resulta, además de por sus buenos resultados en otras aplicaciones [30, 31, 32]. Esta técnica, junto al resto de la metodología, se desarrolla en el siguiente apartado.

4.1.1 Método de los k vecinos más cercanos

Este método, más conocido por la abreviatura k-NN proveniente de su nombre en inglés k-nearest neighbours, fue presentado por primera vez por E.Fix y J. L. Hodges para resolver problemas de clasificación [33].

Su principio de funcionamiento se basa en la noción de proximidad espacial entre observaciones. Primero, comienza estableciendo un conjunto de prototipos a partir de observaciones cuya salida, también denominada clase, es conocida. Luego, esta base de conocimiento es empleada para inferir la clase de nuevas observaciones. Para ello, durante la clasificación, asigna a cada nueva muestra la clase de aquellos prototipos con los que guarda mayor relación, es decir, la clase de los prototipos que están más próximos. En concreto, el parámetro k que aparece en el nombre hace referencia al número de prototipos vecinos más cercanos de cada nueva observación que se tienen en cuenta para realizar la clasificación.

A continuación, se expone un ejemplo para clarificar el funcionamiento del método. Se ha supuesto un problema de clasificación en un espacio de características de dimensión tres. Dos de ellas son las variables de entrada, y están representadas mediante el eje de abscisas y el de ordenadas. La variable restante es la de salida y su valor es ilustrado mediante el símbolo de cada punto. Así pues, en este ejemplo, se tienen 13 prototipos que conforman la base de conocimiento y se quiere clasificar una nueva muestra, representada por la cruz central, en una de las tres clases existentes: azul, amarillo o verde.

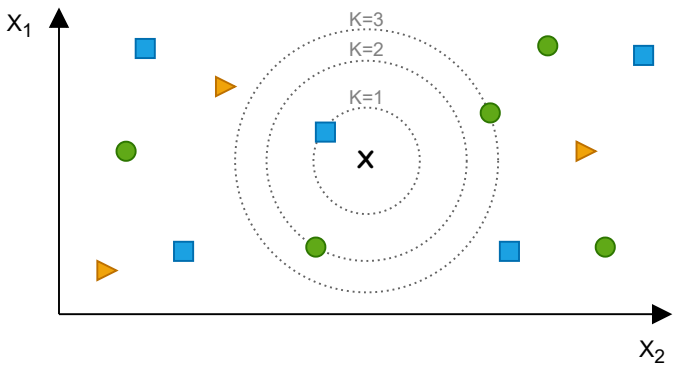

Figura 4.2 Ejemplo gráfico de clasificación mediante k-NN .

Si k es igual a 1, el prototipo vecino más cercano pertenece a la clase azul, por lo que se predice que la muestra con salida desconocida pertenece también a esa clase. Si k es igual a 2, los vecinos más cercanos son uno azul y otro verde, por lo que hay que incluir algún criterio más para solventar la indeterminación. Este podría ser, por ejemplo, escoger la clase que minimice la suma de las

distancias de los prototipos que pertenecen a ella a la muestra desconocida. De este modo, la salida predicha sería de nuevo azul. El resultado cambia si k es 3, puesto que la clase verde se convierte en la mayoritaria y esta pasa a ser la asignada como salida de la muestra tratada.

En definitiva, para un tamaño de vecindad genérico k, la salida predicha, $\hat{y}_i$, es igual a la más repetida entre los k prototipos más cercanos. Esto se expresa matemáticamente en la ecuación 4.4, donde $\mathscr{Y}$ es el conjunto de todas las clases posibles, $\delta_{y,\,y_j}$ es una delta de Kronecker cuyo valor es cero cuando las dos salidas que compara son diferentes y uno cuando son iguales, $\mathscr{K}$ es la vecindad formada por los k prototipos más cercanos, y ω_j corresponde con unos pesos de ponderación que ayudan a solventar las situaciones de empate. Un ejemplo de estos pesos puede ser la inversa de la distancia, así, conforme más próximo esté el prototipo, mayor importancia tendrá.

$$\hat{y}_i = \arg\max_{y \in \mathscr{Y}} \sum_{j \in \mathscr{K}} \omega_j \delta_{y,\,y_j} \tag{4.4}$$

Nótese que esta herramienta puede ser aplicada en problemas de clasificación con un número cualquiera de variables. En este caso se ha escogido tres por el simple hecho de poder realizar la representación en un plano. Otro aspecto importante es que en ningún momento se define cómo se calcula la distancia, por lo que cualquier métrica puede ser empleada.

Una vez presentada la definición original de los k vecinos más cercanos se hace evidente que no se puede utilizar directamente para la detección de outliers. Tal y como se ha descrito, por cada observación con salida desconocida, el método devuelve una clase. No obstante, para la detección de anomalías, el método debería devolver un valor numérico que cuantifique la anormalidad de cada observación [34, 35]: mientras mayor sea este número, más anormal será la observación y más probabilidad tendrá de ser una anomalía.

Existen diversas formas de modificar el método para calcular la anormalidad de una muestra x_i en función de su vecindad formada por los k vecinos más cercanos. Entre ellas destacan tres: usar como anormalidad la media de las distancias a los k vecinos, usar la mediana, o directamente usar la distancia al vecino k-ésimo. A continuación y en el mismo orden, se expresan matemáticamente, donde $\mathscr{K}$ es el conjunto de las k vecinos más cercanos y $d(x_i, x_j)$ es la distancia entre dos observaciones.

$$a(x_i) = \frac{1}{k} \sum_{k \in \mathscr{K}} d(x_i, k) \tag{4.5}$$

$$a(x_i) = Median(\{d(x_i, k) \mid k \in \mathscr{K}\}) \tag{4.6}$$

$$a(x_i) = \max_{k \in \mathscr{K}} d(x_i, k) \tag{4.7}$$

En la figura 4.3 se refleja el resultado obtenido en el mismo conjunto de datos usado en la figura 4.1. El número de vecinos considerados es 15 y la anormalidad se ha calculado según la ecuación 4.6 con la distancia euclídea. El valor de la anormalidad se representa mediante la escala de colores de la derecha.

A modo de resumen, se destacan los pasos a seguir para emplear el método de los k vecinos más cercanos para la detección de anomalías.

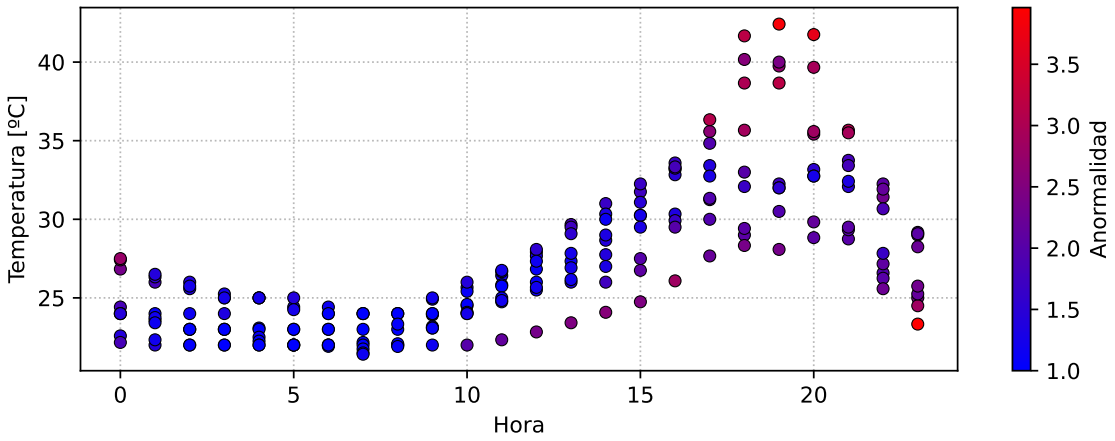

Figura 4.3 Anormalidad de los prototipos en la nube de puntos.

1. **Transformación de los datos**: Se aplica la transformación que permite aprovechar la variación cíclica de la serie temporal para dar lugar a una nube de puntos.

2. **Selección de prototipos**: Consiste en definir un conjunto de observaciones libres de anomalías para ser usado como prototipos en futuras predicciones. A los elementos de este conjunto se les puede calcular la anormalidad utilizando alguna de las tres expresiones anteriores y buscando vecinos entre ellos mismos. Esto último es lo que se refleja en la figura 4.3.

3. **Predicción**: Para calcular la anormalidad de nuevas observaciones, se hace uso de la misma expresión que en el paso previo buscando vecinos entre los prototipos.

4.2 Mejoras del algoritmo

En las sucesivos apartados se describen las bases de una metodología para mejorar los resultados en la detección de anomalías tomando como base el bien conocido algoritmo de los k vecinos más cercanos descrito previamente.

4.2.1 Ampliación del conjunto de datos mediante copias desfasadas

En la anterior figura 4.3 se puede comprobar que las observaciones con mayor valor de anomalía se concentran en la frontera de la nube de puntos; lo que era de esperar, ya que no tendría sentido considerar como anómala una observación situada en el centro de la nube de puntos rodeada por otras muchas más. Para este caso en concreto, las mayores anormalidades se concentran en las mediciones más cálidas situadas en torno a las 18 horas,en las más frías de las 10 en adelante y en los puntos situados en la vertical correspondiente a las 0 y 23 horas.

Que el algoritmo haya devuelto como más anómalas las observaciones de la frontera no es casualidad ni particular de este ejemplo en concreto. Para explicar este fenómeno se va a suponer una nube de puntos de densidad uniforme como la que se presenta en la figura 4.4. Si se considera una vecindad de tamaño 4, los vecinos más cercanos de los puntos con fondo rojo y verde son los que tienen los bordes del mismo color, quedando la distancia media de ambos puntos a sus vecinos igual a 1

para el rojo y 1.103 para el verde. Como consecuencia, el punto verde situado en la frontera es más anómalo. Gráficamente se comprende por qué ocurre esto: si se trazan circunferencias con centro en cada uno de los puntos, es más probable que el radio requerido para envolver todos los vecinos sea menor para un punto de interior que para otro en la frontera, siempre que la distribución de las observaciones en la nube tenga una densidad más o menos uniforme. Esto ocurre porque los puntos de interior tienen vecinos a todo su alrededor, mientras que los de exterior solo los tienen en una dirección y necesitan ampliar su distancia de búsqueda para encontrar todos los vecinos requeridos. En definitiva, estar cerca de la frontera es un factor penalizador, lo cual es un efecto deseable y de esperar.

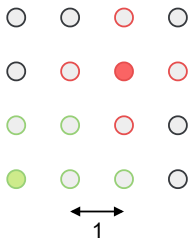

Figura 4.4 Representación del incremento de la anormalidad por emplazamiento en la frontera.

Una vez estudiado este fenómeno, resulta interesante reflexionar sobre dónde está situada realmente la frontera. Para el eje de la magnitud física medida es sencillo, los puntos situados en los límites constituyen la frontera de la nube de puntos y deberían ser penalizados.

No ocurre lo mismo en el eje del tiempo. Al transformar la serie temporal en una nube de puntos, se pierde en cierta medida la información del orden temporal de cada una de las observaciones. En concreto, al dividir la secuencia en periodos de variación regular, las observaciones al final de un periodo no reconocen que las del inicio del periodo siguiente son su vecinos. En el ejemplo que se está usando, el dividir la serie temporal en días hace que las medidas tomadas a las 0 horas no consideren las tomadas a las 23 horas como sus vecinas, cuando realmente están separadas por una única hora. Este efecto no ocurre solo en los puntos de las horas extremas, así pues, una medición a las 16 horas considera que está más próxima de otra a las 3 que a las 2. Nótese que, a medida que se aumenta el número de vecinos considerados, mayor es la cantidad de veces que ocurre este tipo de errores.

Para compensar la información perdida sobre la consecución de los periodos de variación regular superpuestos para formar la nube se propone extender el conjunto de datos. Mejor dicho, triplicar el conjunto de prototipos usados para predecir futuras observaciones. Una copia quedaría tal cual se ha realizado hasta ahora, ocupando un periodo completo en el eje temporal comenzando por 0. Las otras dos copias son desfasadas para que el comienzo y final de la original coincida con el final y comienzo de cada copia, respectivamente. Para nuestro conjunto de datos, una copia se mantendría de 0 a 23 horas, y se crearían dos copias idénticas: una desfasada a la franja de 24 a 47 horas, y a la otra a la de -25 a -1 horas.

Con esta técnica se consigue solventar las incorrecciones en la búsqueda de los vecinos más cercanos que tienen lugar debido a la transformación inicial de los datos. En la figura 4.5 se muestra el resultado obtenido tras su aplicación. En ella solo se muestra completamente la copia original de la nube de puntos, puesto que las otras dos no aportan información gráfica extra. Se puede constatar como esta modificación tiene el efecto deseado, ningún punto ve alterado su valor de anormalidad excepto aquellos que se sitúan en las horas extremas del día, ya que han dejado de ser frontera y reconocen que tienen vecinos a ambos lados.

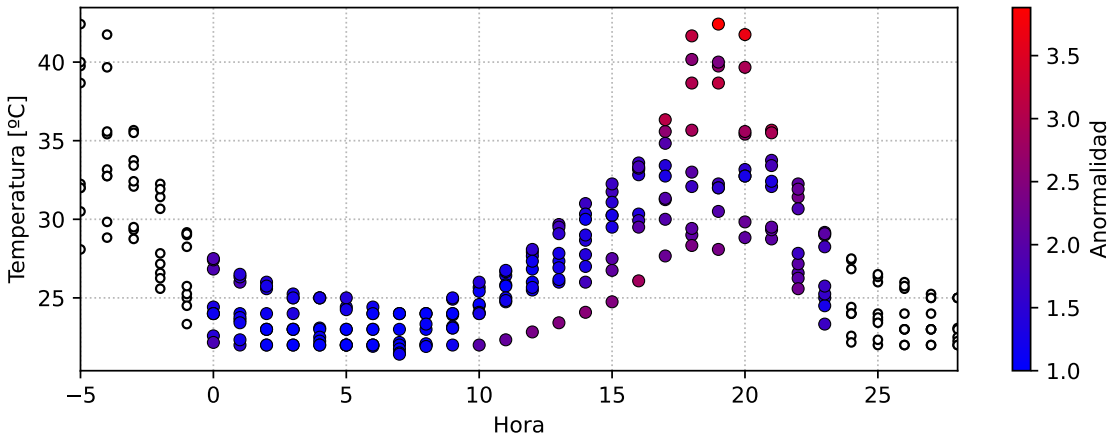

Figura 4.5 Anormalidad de los prototipos en la nube de puntos tras su ampliación.

Resulta evidente que modificar de esta forma los prototipos que conforman la base de conocimiento no altera la forma de predecir el valor de anormalidad de futuras muestras. Estás estarán todas comprendidas en el periodo de tiempo original sin desfasar y su posición condicionará si debe tener en cuenta alguna de las copias desfasadas para encontrar su vecindad.

4.2.2 Mejora en el cálculo de la distancia

Si se estudia con atención la modificación previa se pueden advertir varias deficiencias. La primera de ellas es su ineficiencia, ya que aumentar el número de prototipos requiere más recursos computacionales e incrementa el tiempo de ejecución del algoritmo de búsqueda de vecinos. En segundo lugar, si el número de vecinos considerados es muy grande, puede ocurrir que dos puntos de diferentes copias representativos de la misma observación sean considerados al mismo tiempo, lo cual implica contar erróneamente el mismo vecino más de una vez.

Para solventarlas estas cuestiones, se propone una solución basada en la modificación de la expresión del cálculo de la distancia. Para explicar esta, se va a usar la gráfica del figura 4.6. En ella se han situado cuatro puntos provenientes de una serie temporal con un periodo de variación cíclica regular de un día, por lo que al igual que en el ejemplo usado a lo largo de esta memoria, su variable temporal corresponde a la hora del día en la que se toma la medición y comprende desde las 0 hasta las 23 horas.

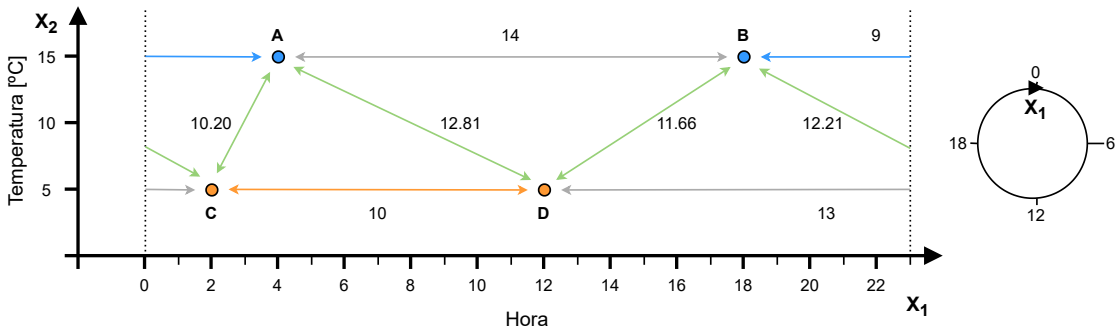

Figura 4.6 Modificación del cálculo de la distancia.

La distancia euclídea entre los puntos naranjas y los azules es de 10 y 14 unidades, respectivamente. No obstante, este cálculo no tiene en cuenta que el inicio de uno de los ciclos de variación regular de la serie temporal original es continuación directa del final de otro ciclo, tal y como se ha descrito en el apartado anterior. Considerando esto, la distancia entre los naranjas no se ve afectada, pero sí entre los azules, que ahora pasa a ser 9.

Para reflejar este efecto en el cálculo de la distancia se puede considerar que los puntos no se sitúan en un plano, sino en la superficie de un cilindro. La altura de este estaría relacionada con la magnitud física medida, y el diámetro con la variable temporal. De este modo, el eje del tiempo dejaría de ser lineal y pasaría a ser cíclico, como se refleja a la derecha de la gráfica.

Tras esta transformación, la forma de proceder para calcular la distancia sería la siguiente: medirla en el sentido creciente y decreciente del eje temporal y escoger la menor de ambas. No obstante, esto resulta complicado de incluir en las expresiones matemáticas, por lo que se plantea una alternativa que deriva en el mismo resultado. Se comienza calculando la distancia en el sentido creciente y, si el resultado es menor que la mitad del eje temporal, es decir, la mitad del periodo del ciclo, esa sería la distancia buscada. No obstante, si no cumpliese dicha condición, la distancia correcta sería la medida en el sentido contrario, la cual es igual al periodo del ciclo menos la distancia previamente calculada.

Esta casuística se recoge en la función de la expresión 4.8, donde T es el periodo del ciclo de variación regular de la serie temporal. La función recibe el valor de la distancia medida en uno de los dos sentidos, y devuelve esta misma o la distancia en el sentido contrario según cual sea la correcta.

$$f(x) = \begin{cases} x & si \quad x <= T/2 \\ T-x & si \quad x > T/2 \end{cases} \tag{4.8}$$

Es importante destacar que esta transformación solo aplica a la dimensión temporal, debiendo dejar el resto intactas. En la gráfica de la figura 4.6 anterior, también se incluye el valor de la distancia entre los puntos de diferente color. Se puede apreciar como ninguna de ellas experimenta cambios con esta transformación excepto la comprendida entre los puntos B y C. Aún en este caso, la distancia medida en el eje de la magnitud física no cambia.

En la tabla 4.1 se recoge la modificación de alguna de las métricas más empleadas según la expresión 4.8 anteriormente descrita. En ellas se puede reafirmar como la transformación solo aplica a la dimensión temporal, la cual se ha considerado que ocupa la posición n-ésima.

Tabla 4.1 Modificación de diferentes métricas.

Nombre	Estándar	Modificada
Euclídea	$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$	$d(X,Y) = \sqrt{\sum_{i=1}^{n-1}(x_i - y_i)^2 + f(\lvert x_n - y_n\rvert)^2}$
Euclídea al cuadrado	$d(X,Y) = \sum_{i=1}^{n}(x_i - y_i)^2$	$d(X,Y) = \sum_{i=1}^{n-1}(x_i - y_i)^2 + f(\lvert x_n - y_n\rvert)^2$
Manhattan	$d(X,Y) = \sum_{i=1}^{n}\lvert x_i - y_i\rvert$	$d(X,Y) = \sum_{i=1}^{n-1}\lvert x_i - y_i\rvert + f(\lvert x_n - y_n\rvert)$
Chebyshov	$d(X,Y) = \text{máx}_{i=1}^{n}\lvert x_i - y_i\rvert$	$d(X,Y) = \text{máx}\{\text{máx}_{i=1}^{n-1}\lvert x_i - y_i\rvert,\ f(\lvert x_n - y_n\rvert)\}$

Para acabar, se incluye de nuevo en la figura 4.7 el resultado del cálculo de las anomalías de los prototipos de los ejemplos anteriores haciendo uso de la distancia euclídea modificada. Se pone de relieve como el resultado es similar a los anteriores, a excepción de que se reduce las anormalidades de los puntos de las horas extremas por interpretar correctamente que no pertenecen a la frontera.

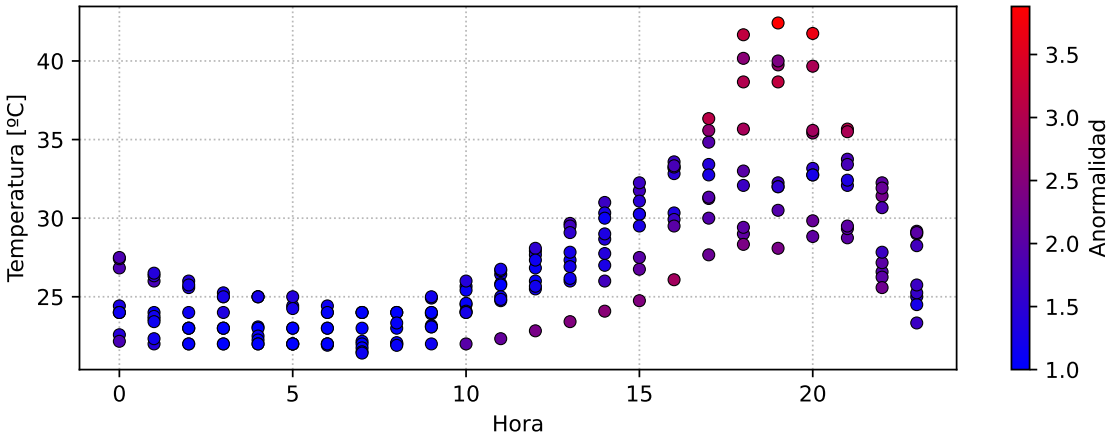

Figura 4.7 Anormalidad de los prototipos en la nube de puntos tras modificar el cálculo de distancias.

4.2.3 Cálculo del umbral de anormalidad

Hasta este punto, la herramienta nos permite calcular un valor de anomalía para cada uno de los prototipos que conforman la base de conocimiento y, aún más importante, para cada una de las nuevas observaciones que se van tomando.

Esto hace posible establecer una lista de todas las observaciones ordenadas en función de cómo de anómalas son. No obstante, no indica explícitamente si alguna de ellas es una anomalía.

Parece razonable pensar que, si una medición es realmente una anomalía, todas las demás que tengan un valor de anormalidad igual o mayor también lo serán. De aquí nace la idea de establecer un umbral de anormalidad para marcar el límite entre las mediciones anómalas y no anómalas. De esta manera, el problema de identificar las anomalías en función de la anormalidad se traduce en establecer un umbral que las delimite.

Establecer el umbral de forma manual mediante prueba y error puede llegar a ser una tarea costosa, con el inconveniente de que, por lo general, el valor obtenido en un problema no es extrapolable a otros. Por ello, en este trabajo se propone un método para situarlo de forma automática empleando el conocimiento que condensan los prototipos.

Si volvemos atrás, en el apartado 4.1.1 se describe cómo se puede calcular, durante la fase de entrenamiento, el valor de la anormalidad de cada medición que conforma el conjunto de prototipos del modelo. Haciendo uso de esto, se propone fijar el umbral en una anormalidad igual a la del prototipo más anormal, es decir, en el mayor valor de anormalidad de entre todos los prototipos usados para el entrenamiento y las posteriores predicciones. De este modo, se está presuponiendo que ninguno de los prototipos que conforman la base de conocimiento es anómalo y, de entre todas las mediciones cuya anormalidad sea calculada mediante ellos, solo aquellas que superen el umbral lo serán.

En la figura 4.8 se recupera de nuevo el ejemplo de la temperatura para presentar el resultado obtenido. Para generarla se ha usado la distancia euclídea modificada, una vecindad de tamaño 15 y el valor máximo de anormalidad de los prototipos para fijar el umbral.

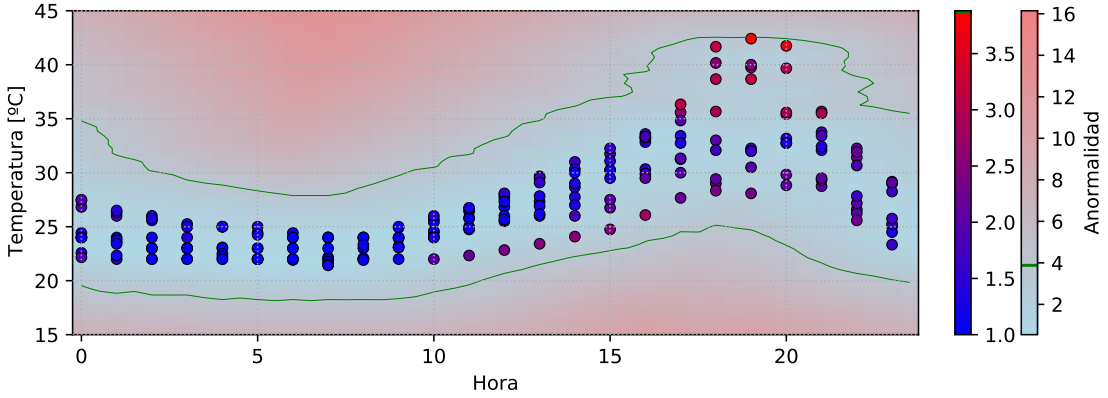

Figura 4.8 Límite fijado por el umbral entre observaciones anómalas y no anómalas.

En la nueva gráfica se representa la misma información que en las anteriores: la posición de cada prototipo junto a su anormalidad según una escala de colores. A esto se le ha añadido un mapa de calor que simboliza la anormalidad que tendría una nueva medición que se situase en cada punto del plano. Así pues, si en este mapa se marcan los puntos cuya anormalidad es igual a la del prototipo más anormal, se obtienen las curvas de color verde. En otras palabras, la curva verde corresponde al umbral que se ha fijado por medio del prototipo más anormal para poder identificar las anomalías. Cualquier nueva observación que supere el umbral estará situado al otro lado de la frontera que marca la curva y será considerada como una anomalía.

Esta curva, a partir de ahora curva de predicción, pasará a ser una superficie, o su equivalente en un espacio vectorial de más de tres dimensiones, cuando el número de variables de variables de entrada del problema sea más de una sin contar el tiempo.

Desplazamiento del umbral

Es posible que se dé el caso en el que el umbral calculado mediante el método anterior sea demasiado restrictivo o permisivo, provocando que se identifiquen muchas observaciones como anomalías cuando realmente no lo son, o al contrario, respectivamente. Ambas situaciones se pueden paliar en cierta medida aplicando un factor al umbral, amplificando o disminuyendo su valor según sea mayor o menor a 1.

En la figura 4.9 se ilustra el resultado de aplicar este factor. En ella se puede apreciar como modificar de esta forma el umbral no provoca variaciones significativas en la forma de la curva de predicción, sino que su efecto consiste principalmente en un desplazamiento que la acerca o la aleja de la nube de puntos. Si el factor es mayor a 1, el umbral crece, la curva se aleja y se detectan menos anomalías. El efecto contrario sucede si el factor es menor a 1.

La introducción de este parámetro puede parecer una contradicción, puesto que originalmente se buscaba no tener que fijar de forma manual el umbral. Sin embargo, nótese la diferencia entre la situación inicial, en la que era necesario determinar un valor absoluto, y esta, en la que realmente se está estableciendo variaciones relativas de un umbral situado de manera automática, lo cual facilita enormemente la tarea.

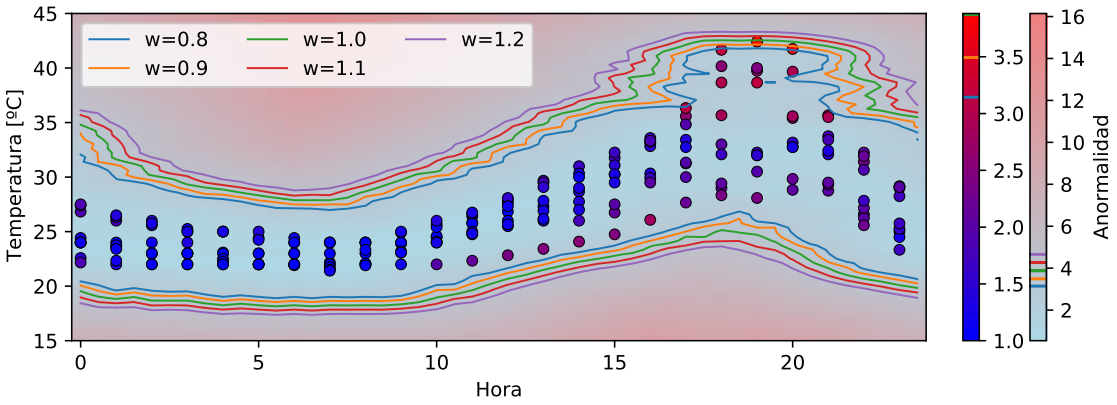

Figura 4.9 Desplazamiento relativo del umbral.

Por último, parece lógico pensar que la curva de predicción pase siempre justo por encima del prototipo con mayor anormalidad, sin embargo, esto no es así, tal y como se puede comprobar en la figura 4.9. Para entender por qué, es necesario recordar que el método de los k vecinos se divide en dos fases, entrenamiento y clasificación. Durante la primera, los prototipos son almacenados y se obtienen sus anormalidades mediante el cálculo de las distancias entre ellos. En la segunda, la anormalidad de una nueva medición se obtiene a partir de las distancias con los prototipos previos. Como en cada caso se calculan las distancias con conjuntos diferentes, el valor de anormalidad para el mismo punto del plano no es el igual en ambas fases. Así pues, que la curva de predicción se obtenga en la segunda y la anormalidad del prototipo en la primera, hace que no coincidan en el espacio.

Este efecto es más acusado cuanto menor es el número de vecinos considerados. En el caso extremo de k igual a 1, ningún prototipo tendrá anormalidad 0 si no existen dos prototipos coincidentes. No obstante, si durante la predicción una medición se sitúa justo sobre un prototipo, le será asignada una anormalidad nula.

4.2.4 Elección del número de vecinos

El único hiperparámetro del método de los k vecinos más cercanos es precisamente k, el número de vecinos a considerar. Si este se incrementa, es necesario buscar más vecinos, por lo que se requiere aumentar la distancia de búsqueda para encontrarlos y, en consecuencia, las anormalidades resultantes son mayores. No obstante, como todas las observaciones sufren el mismo efecto, las variaciones relativas de anormalidad entre ellas no se ven tan afectadas.

En la imagen 4.10 se recoge el impacto que tiene la variación de este hiperparámetro sobre la curva de predicción fijada por el prototipo con mayor anormalidad, el cual puede ser diferente para cada valor de k. En esta ocasión, la forma de la curva sí sufre variaciones significativas, así que el efecto no es un simple desplazamiento como antes.

A continuación, se propone también un método para calcular de forma automática el número de vecinos a considerar para resolver el problema. Este se basa en cuantificar la variación de la curva de predicción a medida que se incrementa la vecindad. El número buscado se tendrá cuando, al incrementar k en una unidad, la curva sufra una variación relativa menor a un límite fijado.

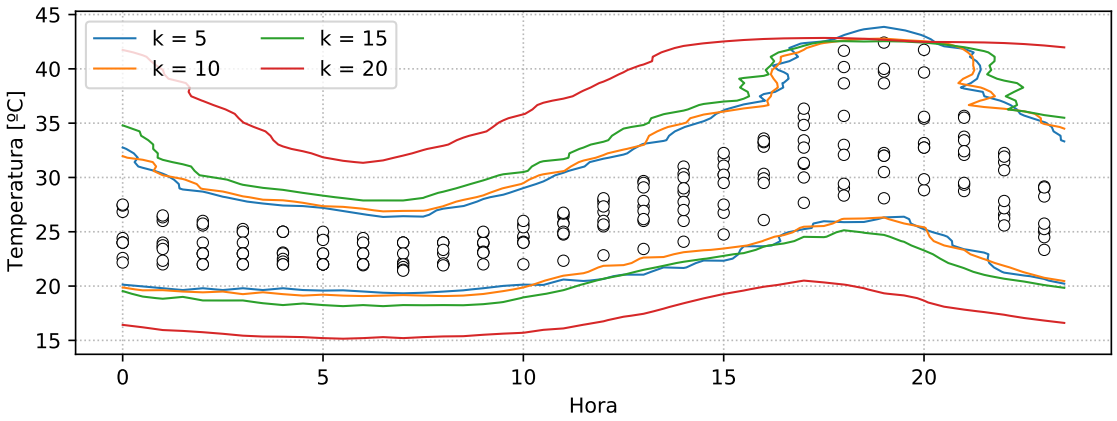

Figura 4.10 Efecto del parámetro k en el límite que separa las mediciones anómalas.

En primer lugar, para cuantificar la variación entre dos curvas de predicción, A y B, se ha optado por emplear la distancia de Hausdorff. Esta se define como la máxima distancia dirigida de Hausdorff entre las curvas A y B, expresión 4.9.

$$H(A,B) = \text{máx}\{h(A,B), h(B,A)\} \tag{4.9}$$

La distancia dirigida de Hausdorff entre dos curvas A y B, expresión 4.10, se calcula como la máxima de las mínimas distancias entre cada punto de A con todos los puntos de B. Esto es, primero se elabora un conjunto que almacena un valor por cada punto de A: la distancia desde el punto de A que se esté tratando al punto de B que esté más cerca. Luego, la distancia dirigida se calcula como el máximo del conjunto anterior. Su interpretación física corresponde con la mínima distancia que una persona situada sobre la curva A debería recorrer para poder alcanzar la curva B. Este concepto puede ser extendido también a superficies, considerando todos los puntos que la definen.

$$h(A,B) = \underset{a\in A}{\text{máx}}\ \underset{b\in B}{\text{mín}}\, d(a,b) \tag{4.10}$$

Una vez se tiene la herramienta para cuantificar la variación entre curvas de predicción se construye el conjunto ordenado $\mathscr{H}$, donde $H(k_i,k_j)$ es la distancia de Hausdorff entre las curvas de predicción resultantes de usar el método de los k vecinos más cercanos con k igual a i y j:

$$\mathscr{H} = \{H(1,2), H(2,3), H(3,4), ..., H(k_i,k_j), ...\} \tag{4.11}$$

Luego, se construye un nuevo conjunto $\mathscr{V}_H$ que contiene la variación relativa entre dos elementos consecutivos de $\mathscr{H}$:

$$\mathscr{V}_H = \{v_1, v_2, ..., v_i = \frac{|H(i,i+1) - H(i+1,i+2)|}{H(i,i+1)}, ...\} \tag{4.12}$$

Finalmente, el índice del primer elemento de $\mathscr{V}$ cuyo valor sea menor a la variación relativa máxima admisible fijará el número de vecinos a usar. Esto es, si el elemento que cumple dicha condición es v_n, el número de vecinos a usar será $k = n$.

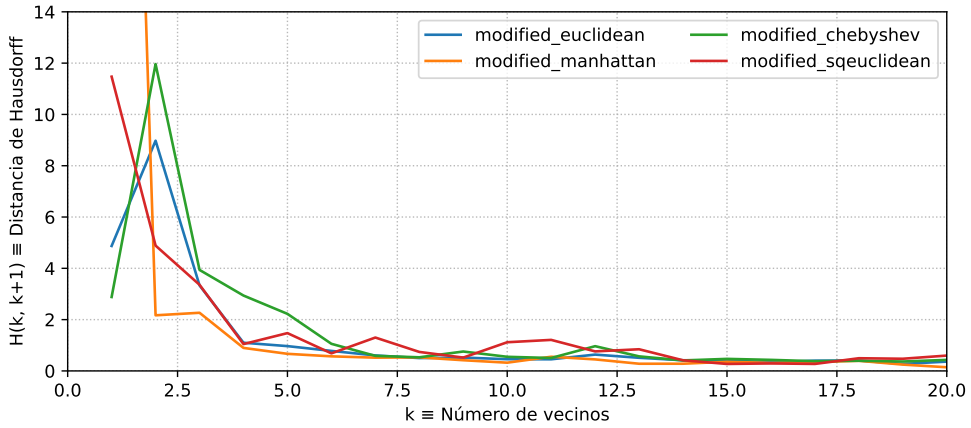

Figura 4.11 Distancia entre curvas de predicción consecutivas respecto al número de vecinos.

En la gráfica de la figura 4.11 se representa el conjunto $\mathscr{H}$ obtenido haciendo uso de los mismos datos que en los ejemplos anteriores. Cada línea representa una métrica modificada empleada para el k-NN.

Se aprecia como la distancia de Hausdorff sigue una tendencia decreciente hasta estabilizarse. El hecho de establecer un umbral de variación relativa nos permite evitar la zona inicial transitoria y obtener un modelo cuyo comportamiento ya se ha estabilizado respecto a otros con un número parecido de vecinos. En este caso concreto, el valor buscado de k está aproximadamente entre los 5 y 8 vecinos, puesto que en dicho rango las curvas comienzan a ser constantes.

4.2.5 División del periodo de variación regular en horquillas temporales

Si se presta atención a la figura 4.8 se puede constatar que la curva de predicción resultante es demasiado restrictiva en las horas próximas a la medición de mayor anormalidad y, por otro lado, demasiado permisiva en el resto del rango del eje temporal. Esto puede ser una fuente de errores durante la predicción, ya que una única medición tomada en una hora concreta del día, o del periodo de variación regular que se está tratando, está condicionando las predicciones realizadas en cualquier otra hora.

Por otro lado, se puede ver como la dispersión de las mediciones no es constante a lo largo del día. Aproximadamente desde las 4 hasta las 12 horas, se forma una nube de puntos más compacta y mejor definida, mientras que el resto del tiempo las observaciones están más dispersas. Esto refleja que es más común que, de un día para otro, la variación de temperatura sea más acusada solo en determinadas horas del día.

Si se combinan los dos hechos anteriores, parece razonable dividir el día, o el periodo de variación regular que se esté tratando, en horquillas temporales. Cada horquilla estará asociada a un modelo de detección diferente, y su curva de predicción estará definida por el prototipo de mayor anormalidad de entre los que contiene la horquilla. No obstante, el modelo de cada horquilla debe ser entrenado con todos los prototipos disponibles, al igual que se hacía antes, y no solo con los que están dentro de la horquilla, para que así realicen correctamente la búsqueda de vecinos. Finalmente, cuando aparezca una nueva observación, se tratará con el modelo asignado a la horquilla temporal a la que pertenezca.

En la figura 4.12 se muestra el resultado de implementar dicha modificación. Se han establecido 5 horquillas diferentes, y se puede apreciar como en todas ellas la curva de predicción está mejor adaptada y tiene un seguimiento más adecuado de la tendencia de la nube de puntos.

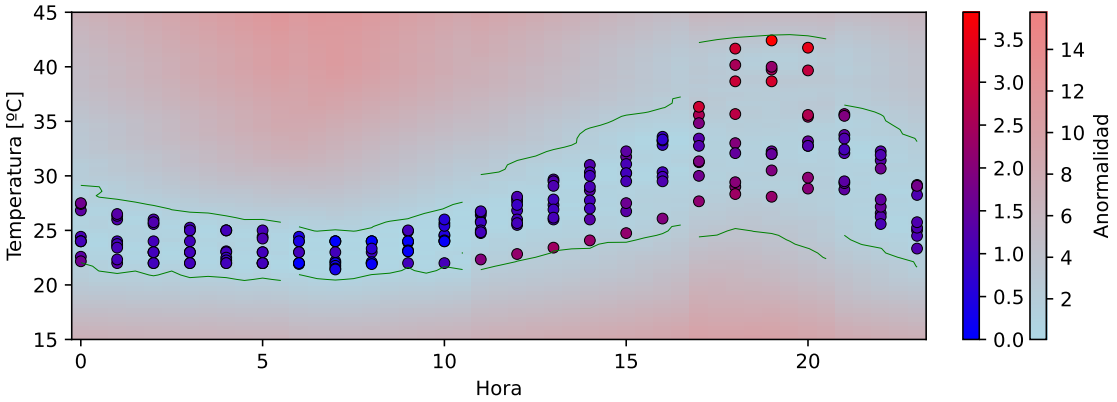

Figura 4.12 Distancia entre curvas de predicción consecutivas respecto al número de vecinos.

Es importante destacar que esta modificación obliga a entrenar un modelo por cada horquilla temporal, haciendo uso en todos ellos del conjunto completo de prototipos. Estos modelos se diferenciarán principalmente en el número de vecinos que contemplan, aunque cualquier otra variación, por ejemplo, la métrica empleada, es admisible. El método expuesto anteriormente para inferir el valor de k puede ser aplicado de forma individual en cada horquilla. Por otro lado, se tendrán tantas curvas de predicción como franjas, estando cada una de ellas definida por una observación que esté contenida en la franja.

4.2.6 Actualización temporal de los modelos

Que las variables tratadas tengan un periodo de variación regular más o menos estable no impide que, además, sufran variaciones de pequeña magnitud que se van acumulando de forma constante a lo largo del tiempo. Esto puede provocar que la serie temporal difiera mucho consigo misma cuando se compara en instantes muy alejados.

Esto se entiende fácilmente con el ejemplo de la temperatura media diaria. A pesar de que las de dos días consecutivos son muy similares, no ocurre lo mismo cuando se compara la registrada un día de diciembre con otra de julio. La temperatura experimenta cambios considerables a largo plazo aun manteniendo su variación cíclica durante todos los días.

Por este motivo, parece lógico pensar que el desempeño del detector de anomalías será pobre cuando se aplique sobre nuevas observaciones muy alejadas temporalmente de aquellas que fueron empleadas como prototipos durante el entrenamiento. Para paliar este efecto se propone actualizar de forma periódica los modelos de detección basados en los k vecinos más cercanos que conforman el detector.

La idea es la siguiente. Primero se entrenan los modelos del detector haciendo uso de observaciones lo más cercanas posible a las mediciones que se van a analizar en busca de anomalías. Luego, conforme se van estudiando las nuevas observaciones en orden cronológico, se mantiene la cuenta

de cuántas de ellas no son clasificadas como anómalas. Cuando este registro supera un cierto número, n, se procede a actualizar los modelos de predicción. Para ello, se modifican los prototipos usados para definir el conjunto de entrenamiento: se eliminan los n prototipos más antiguos, y se añaden las n observaciones no anómalas más recientes. Un valor de referencia para n podría ser el número de observaciones que constituyen un ciclo de variación regular.

De esta manera, se consigue que el detector evolucione y se adapte a las variaciones que pueda presentar la magnitud medida.

5 Implementación de la metodología

La metodología propuesta para la detección de anomalías se ha implementado haciendo uso del lenguaje Python y el paradigma de programación orientado a objetos. El programa desarrollado incluye la base del método y todas las ampliaciones y modificaciones descritas anteriormente. La implementación se ha realizado de una forma modular y escalable, con el fin de que la solución obtenida pueda ser ampliada con la adición de nuevos módulos que trabajen con los ya existentes.

Entre los módulos de Python que se han empleado destacan los siguientes:

- **Pandas** [36]: para facilitar el tratamiento de las series temporales mediante las herramientas y estructuras de datos avanzadas que integra.
- **Numpy** [37]: para mejorar la eficiencia de las operaciones lógicas matemáticas complejas que involucran grandes cantidades de datos.
- **Datetime** [38]: para el tratamiento y comparación de las marcas de tiempo.
- **Requests** [39]: para poder comunicarse con la API.
- **Numba** [40]: para compilar las funciones de cálculo de distancias modificadas. Su uso ha permitido mejorar en un orden de magnitud el tiempo de ejecución.
- **PyOD** [41]: este es el más importante, puesto que incluye el método de los k vecinos más cercanos para la detección de outliers en nubes de puntos. Integra un algoritmo eficiente para la búsqueda de vecinos y el cálculo de la anormalidad.

Además, la aplicación se ha desarrollado de la forma más general posible para que se pueda aplicar, a priori, en cualquier problema de detección de anomalías en sensores. Para lograr esto, a la hora de ejecutarse, se le deben proporcionar diversos parámetros para adaptarla tanto al contexto en el que se va a utilizar como a los datos que va a recibir: endpoints de la API, nombre de las variables que va a tratar, división en horquillas temporales, tipo de modelo de detección de outliers a emplear en cada horquilla junto a todos sus parámetros, la métrica a usar, el rango de número de vecinos entre los que se debe buscar si se desea que este hiperparámetro sea calculado de forma automática, periodo de tiempo que define el rango de observaciones que conformarán el primer conjunto de entrenamiento, periodo de reentrenamiento, etc.

Para almacenar los resultados del algoritmo, se ha utilizado la misma base de datos centralizada que emplea el banco de pruebas descrito en el capítulo 3. Para ello, se han creado dos nuevas tablas, las cuales están reflejadas en el modelo entidad-relación del anexo A. Una guarda el identificador de cada agregación tratada, además de si se ha clasificado como anomalía o no, la puntuación

de anormalidad obtenida y un identificador del modelo con el que ha sido tratada. La segunda tabla almacena el mismo identificador de modelo junto a los parámetros que lo definen: número de vecinos, métrica, prototipos usados para el entrenamiento, etc.

Como es evidente, el acceso a la base de datos se continúa realizando por medio de la misma API del banco de pruebas, la cual también se ejecuta en el servidor central. Por este motivo, la funcionalidad de esta ha sido ampliada: se han añadido nuevos endpoints para cubrir las necesidad sobre las tablas nuevas y las que ya existían previamente.

La funcionalidad que integra la aplicación es la siguiente, figura 5.1. Primero, lee todos los parámetros de entrada y configura el detector. Luego, recupera todas las mediciones que forman el primer conjunto de entrenamiento, es decir, los prototipos. Después, realiza el entrenamiento, teniendo en cuenta las horquillas temporales y todos los parámetros previamente dados. Una vez están preparados los modelos de detección, recupera cronológicamente las mediciones almacenadas en la base de datos para detectar anomalías entre ellas. Cuando se supera un cierto número de mediciones detectadas como no anómalas, se actualizan los modelos mediante otra etapa de entrenamiento en la que se eliminan los prototipos más antiguos y se añaden las nuevas observaciones no anómalas. Este ciclo se repite de forma ininterrumpida, y todos los resultados que se van generando son enviados a la API para que los almacene en las tablas correspondientes.

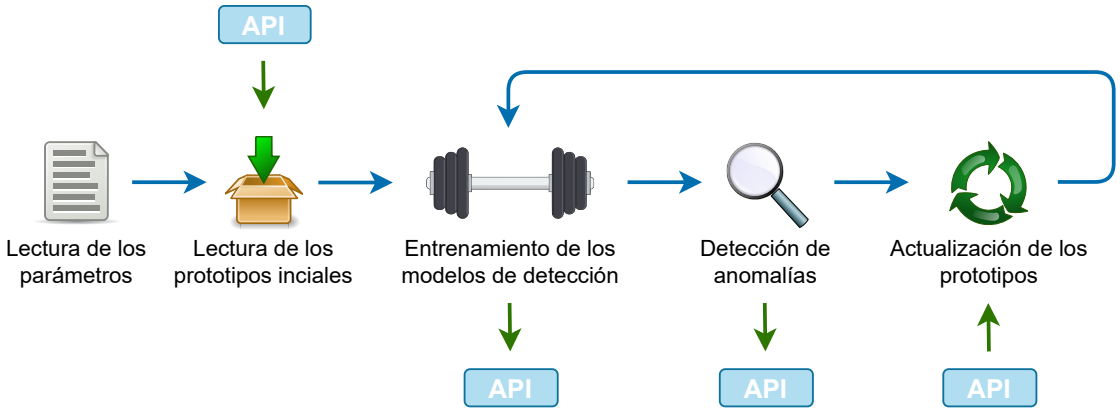

Figura 5.1 Funcionamiento de la aplicación de detección de anomalías.

Un aspecto muy importante es que, para hacer esta solución aún más general y flexible, todos los programas que se ejecutan en el servidor lo hacen dentro de contenedores virtuales aprovechando la tecnología de Docker [3], una plataforma de código abierto con la capacidad de separar las aplicaciones de la infraestructura y que, por tanto, facilita su desarrollo, despliegue y ejecución.

Docker permite crear y ejecutar dichos contenedores, los cuales no son más que entornos virtuales aislados y autosuficientes que ejecutan de manera independiente las aplicaciones que se deseen. La principal diferencia de estos contenedores con las máquinas virtuales es que realizan la virtualización a nivel de sistema operativo, y no a nivel de hardware. Aun así, dos contenedores corriendo al mismo tiempo en el mismo equipo no saben que están compartiendo recursos. Esto es gracias a que Docker hace uso de las herramientas del kernel Linux compartido por los contenedores para añadir capas de abstracción y aislar la visión que tiene cada uno del sistema operativo que usan en común. Evidentemente, utilizar el mismo núcleo para todos los contenedores es mucho más eficiente, respecto a la demanda de recursos, que virtualizar un sistema operativo completo por cada máquina virtual.

Por tanto, si en un contenedor se incluyen las aplicaciones desarrolladas junto a todas sus dependencias, tendremos la certeza de que estas funcionarán en cualquier equipo que ejecute el contenedor, independientemente de su sistema operativo y de las utilidades que tenga instaladas. De esta manera, el despliegue es tan sencillo como crear un contenedor con todos los recursos necesarios y distribuirlo entre todos los equipos que deban ejecutarlo.

El flujo de trabajo a seguir cuando se usa Docker se ilustra en la figura 5.2. Primero, por cada servicio que se quiera aislar en un contenedor es necesario escribir un fichero conocido como *Dockerfile*, en el cual se define: la imagen de Linux en la que se basa el contenedor, los archivos que contendrá, las configuraciones requeridas, la instalación de las dependencias, los accesos que tiene a los datos guardados en la máquina huésped, comandos o programas que debe ejecutar, etc. Luego, Docker usa el *Dockerfile* para construir la imagen del contenedor definido, siendo esta un tipo de plantilla que precisa cómo se generan los contenedores: Cuando una imagen es lanzada, se crea una instancia ejecutable de esta, esto es, un contenedor. En este punto, ya se tendría la aplicación corriendo de manera aislada en un contenedor, no obstante, lo común es tener diferentes servicios que se relacionan entre sí ejecutándose en contenedores diferentes. Por este motivo, no se lanzan directamente a partir de la imagen, sino que se crea un nuevo fichero denominado *docker-compose*. Este último contiene en formato yaml la lista de los contenedores que deben ser lanzados, indicando para cada uno de ellos la imagen a partir de cual se genera, junto a otros parámetros como el mapeo de puertos que se hace entre la máquina huésped y el contenedor en cuestión. Una vez se ejecute este último archivo, se lanzarán todos los contenedores requeridos.

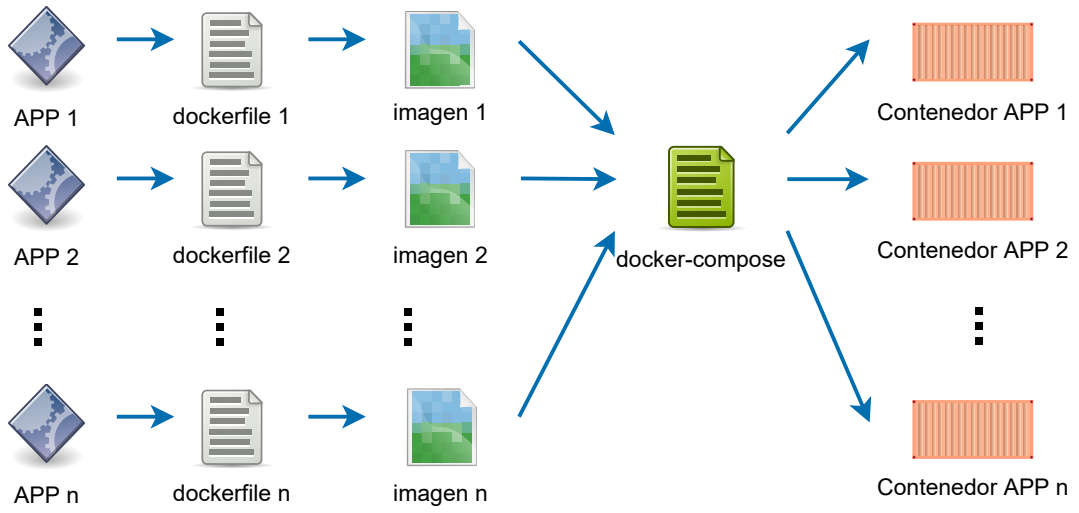

Figura 5.2 Flujo de trabajo con Docker.

Así es como se ha procedido en este trabajo. Por un lado, se usa un contenedor que engloba la base de datos centralizada PostgreSQL, junto a la API que habilita y limita el acceso a ella en escritura y lectura. Por otro lado, en otro contenedor, se ejecuta el algoritmo con la metodología propuesta para la detección de anomalías. Esta arquitectura queda reflejada de manera esquemática en la figura 5.3. De esta manera, se ha llegado a una solución extensible y flexible, muy fácil de desplegar, replicar, configurar y explotar, ya que se puede ejecutar en cualquier ordenador personal, estación de trabajo, servidor o servicio en la nube.

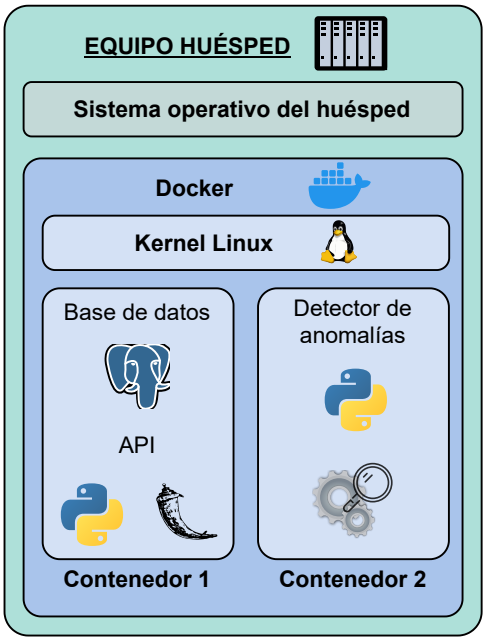

Figura 5.3 Arquitectura implementada en el servidor.

6 Resultados y evaluación de la metodología

En este capítulo se exponen los resultados de aplicar, sobre los datos de temperatura y humedad registrados por el banco de pruebas descrito en el capítulo 3, la metodología para la detección de anomalías en sensores de bajo coste presentada en el capítulo 4.

6.1 Resultados sobre los datos originales

A continuación, se muestran los resultados obtenidos haciendo uso de los datos registrados sin modificar, es decir, tal y como son enviados por el sensor. En concreto, se han empleado las agregaciones horarias, de la temperatura y humedad, registradas en el periodo comprendido entre el día 1 de mayo de 2021-05-01 y el 24 de junio de 2021-06-24. Esto da lugar a un conjunto de datos con 1320 registros consecutivos separados temporalmente 1 hora.

Para comenzar, las variables se han tratado por separado: primero se han buscado anomalías en los valores de temperatura y, luego, en los de humedad. En cada caso, se han usado los 7 primeros días, 168 registros, como conjunto inicial de entrenamiento, por lo que realmente solo se han analizado 1152 registros. Además, tal y como se describió en el apartado 4.2.6, los modelos k-NN en los que se basa el detector son actualizados periódicamente, en este caso concreto, cada vez que 24 registros son clasificados como no anómalos. También se han usado 5 horquillas temporales con límites en las horas: 0, 6, 11, 17, 21, y 23

Los resultados se recogen en la tabla 6.1, donde cada fila constituye una prueba caracterizada por: una de las métricas modificadas en el apartado 4.2.2 y un factor de ponderación del umbral marcado por el prototipo más anómalo de cada horquilla, apartado 4.2.3. El número de vecinos considerados se ha fijado en 1 para todos los casos.

La columna errores representa el total de anomalías que son detectadas. Se consideran errores, en este caso concreto, porque tenemos la certeza de que el sensor no ha fallado en ningún momento; además, al tratarse de datos agregados por horas, se reduce aún más el efecto de cualquier desviación que alguna medición haya podido tener respecto a su valor esperado. Por este motivo, mientras menor sea la columna errores, mejor será el desempeño. El valor porcentual es también el número de errores, pero normalizado respecto al total de observaciones que se están analizando: 1152.

Tabla 6.1 Resultados con un único vecino con los datos originales tratando las magnitudes por separado.

Métrica	Ponderación	Errores	
		Temperatura	Humedad
euclídea mod.	1.0	255 [22.13 %]	380 [32.99 %]
euclídea mod.	1.1	187 [16.23 %]	**328 [28.47 %]**
euclídea mod.	1.2	**165 [14.32 %]**	**318 [27.60 %]**
manhattan mod.	1.0	252 [21.87 %]	375 [32.55 %]
manhattan mod.	1.1	225 [19.53 %]	353 [30.64 %]
manhattan mod.	1.2	215 [18.66 %]	**271 [23.52 %]**
chebyshev mod.	1.0	225 [19.53 %]	403 [34.98 %]
chebyshev mod.	1.1	**123 [10.68 %]**	356 [30.90 %]
chebyshev mod.	1.2	**107 [09.29 %]**	345 [29.95 %]
sqeuclídea mod.	1.0	272 [23.61 %]	391 [33.94 %]
sqeuclídea mod.	1.1	251 [21.79 %]	380 [32.99 %]
sqeuclídea mod.	1.2	204 [17.71 %]	369 [32.03 %]

De los resultados obtenidos se extraen varias conclusiones. La primera de ellas es que, a medida que aumenta el factor de ponderación del umbral que marca el límite de la anormalidad para clasificar las observaciones en anómalas o no, menos errores se comenten. Esto era de esperar, puesto que incrementar el umbral hace que la curva de predicción se aleje de la nube de puntos y, en consecuencia, detecte menos anomalías. En otras palabras, reduce la sensibilidad del detector. La segunda conclusión es que el uso de la métrica euclídea al cuadrado es la que arroja peores resultados en global. Respecto a la mejor métrica, no está claro. Chebyshev es evidentemente la mejor en las temperaturas, pero presenta un resultado casi tan pobre como la euclídea al cuadrado con las humedades. Así pues, en promedio entre los dos problemas, es la distancia euclídea la que sobresale.

A continuación, en la tabla 6.2, se recogen los resultados en las mismas condiciones que los anteriores, con la única diferencia que, en cada caso, el número de vecinos considerados en cada horquilla es calculado automáticamente mediante la distancia de Hausdorff, apartado 4.2.4. Además, en esta ocasión, sí tiene sentido considerar también el método empleado para calcular la anormalidad de cada observación, apartado 4.1.1. Esto no ocurre en el caso anterior ya que, si solo se tiene en cuenta un vecino, la media, la mediana y el máximo de las distancias a él coinciden.

Lo primero que se observa es que existen casos que reducen aún más el mínimo porcentaje de error obtenido previamente. Esto quiere decir que la idea planteada para buscar un número adecuado de vecinos haciendo uso de la distancia de Hausdorff presenta resultados satisfactorios. Respecto al método empleado para calcular la anormalidad, parece que aquel que usa solo la distancia al vecino más lejano es la que mejor comportamiento tiene: "largest", 13 veces la mejor y 1 vez la peor; "mean", solo 2 veces la mejor y 17 veces la peor; y "median", 9 veces la mejor y 7 veces la peor.

En cuanto a la métrica, vuelve a existir consenso en que la euclídea al cuadrado presenta los peores resultados, mientras que la euclídea destaca positivamente en ambos problemas, la Chebyshev lo hace en el de temperatura y la Manhattan en el de humedades. Por último, el efecto del factor de ponderación es el mismo que en el caso anterior, mientras mayor es, menos errores se cometen.

Tabla 6.2 Resultados con el cálculo automático de vecinos sobre los datos originales.

Métrica	Método	Ponderación	Errores	
			Temperatura	Humedad
euclídea mod.	largest	1.0	258 [22.40 %]	381 [33.07 %]
euclídea mod.	mean	1.0	319 [27.69 %]	412 [35.76 %]
euclídea mod.	median	1.0	328 [28.47 %]	376 [32.64 %]
euclídea mod.	largest	1.1	193 [16.75 %]	281 [24.39 %]
euclídea mod.	mean	1.1	240 [20.83 %]	339 [29.43 %]
euclídea mod.	median	1.1	202 [17.53 %]	320 [27.78 %]
euclídea mod.	largest	1.2	**090 [07.81 %]**	**264 [22.92 %]**
euclídea mod.	mean	1.2	125 [10.85 %]	289 [25.09 %]
euclídea mod.	median	1.2	**123 [10.68 %]**	**243 [21.09 %]**
manhattan mod.	largest	1.0	255 [22.14 %]	334 [28.99 %]
manhattan mod.	mean	1.0	283 [24.57 %]	358 [31.08 %]
manhattan mod.	median	1.0	240 [20.83 %]	392 [34.03 %]
manhattan mod.	largest	1.1	**119 [10.33 %]**	269 [23.35 %]
manhattan mod.	mean	1.1	186 [16.15 %]	320 [27.78 %]
manhattan mod.	median	1.1	200 [17.36 %]	311 [27.00 %]
manhattan mod.	largest	1.2	135 [11.72 %]	**242 [21.01 %]**
manhattan mod.	mean	1.2	132 [11.46 %]	**267 [23.18 %]**
manhattan mod.	median	1.2	138 [11.98 %]	**237 [20.57 %]**
chebyshev mod.	largest	1.0	161 [13.98 %]	352 [30.56 %]
chebyshev mod.	mean	1.0	306 [26.56 %]	366 [31.77 %]
chebyshev mod.	median	1.0	224 [19.44 %]	374 [32.47 %]
chebyshev mod.	largest	1.1	133 [11.55 %]	328 [28.47 %]
chebyshev mod.	mean	1.1	219 [19.01 %]	317 [27.52 %]
chebyshev mod.	median	1.1	217 [18.84 %]	294 [25.52 %]
chebyshev mod.	largest	1.2	**117 [10.16 %]**	267 [23.18 %]
chebyshev mod.	mean	1.2	**104 [09.03 %]**	278 [24.13 %]
chebyshev mod.	median	1.2	166 [14.41 %]	278 [24.13 %]
sqeuclídea mod.	largest	1.0	275 [23.87 %]	346 [30.03 %]
sqeuclídea mod.	mean	1.0	320 [27.78 %]	380 [32.99 %]
sqeuclídea mod.	median	1.0	316 [27.43 %]	321 [27.86 %]
sqeuclídea mod.	largest	1.1	225 [19.53 %]	332 [28.82 %]
sqeuclídea mod.	mean	1.1	278 [24.13 %]	363 [31.51 %]
sqeuclídea mod.	median	1.1	222 [19.27 %]	325 [28.21 %]
sqeuclídea mod.	largest	1.2	123 [10.68 %]	327 [28.39 %]
sqeuclídea mod.	mean	1.2	205 [17.80 %]	334 [28.99 %]
sqeuclídea mod.	median	1.2	198 [17.19 %]	324 [28.12 %]

Además, se han repetido las pruebas anteriores fijando el número de vecinos a 1, pero tratando conjuntamente las variables temperatura y humedad con un único detector, tabla 6.3. Ahora, por tanto, el tamaño del espacio de característica es tres en vez de dos, por lo que los puntos obtenidos de las mediciones se sitúan en el espacio en vez de en el plano. Por ello, la curva de predicción pasa a ser una superficie que envuelve la nube de puntos tridimensional.

Es fácil observar que esta agregación no es recomendable, al menos en este problema. Los resultados son, para todos los casos, peores que los obtenidos anteriormente en la magnitud individual con resultados más desfavorables, es decir, en la humedad.

Tabla 6.3 Resultados con un único vecino tratando conjuntamente temperatura y humedad sobre los datos originales.

Métrica	Ponderación	Errores
euclídea mod.	1.0	543 [47.14 %]
euclídea mod.	1.1	460 [39.93 %]
euclídea mod.	1.2	**333 [28.91 %]**
manhattan mod.	1.0	455 [39.50 %]
manhattan mod.	1.1	**387 [33.59 %]**
manhattan mod.	1.2	**322 [27.95 %]**
chebyshev mod.	1.0	464 [40.28 %]
chebyshev mod.	1.1	433 [37.59 %]
chebyshev mod.	1.2	415 [36.02 %]
sqeuclídea mod.	1.0	502 [43.58 %]
sqeuclídea mod.	1.1	473 [41.06 %]
sqeuclídea mod.	1.2	445 [38.63 %]

Para concluir, se añade en la figura 6.1 una comparación del aspecto que tienen las nubes de puntos resultantes de transformar la serie temporal de la temperatura, a la izquierda, y humedad, a la derecha. Se puede intuir que la causa de que los resultados con la humedad sean más pobres que con la temperatura reside en la mayor dispersión de la nube de puntos, además de un comportamiento intrínseco más anormal en las horas finales del día.

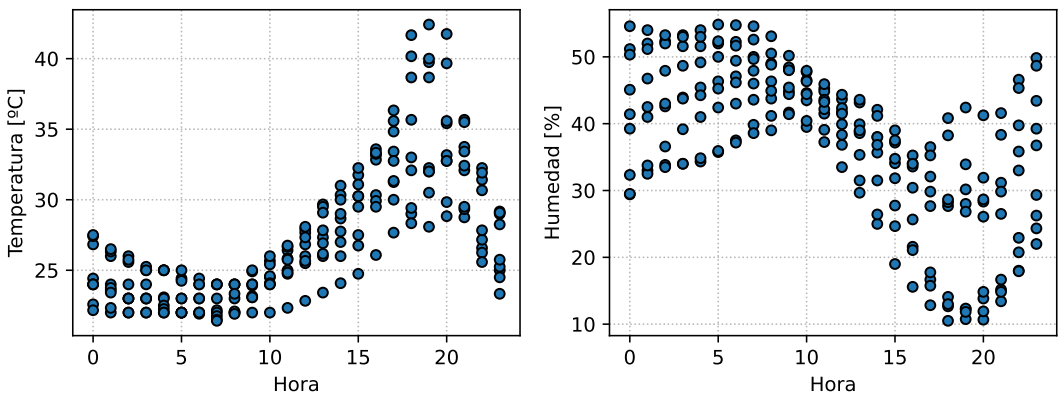

Figura 6.1 Nubes de puntos resultantes de las series temporales de temperatura y humedad.

6.2 Resultados sobre los datos modificados con anomalías

A continuación, se realiza un estudio similar a los anteriores, pero sobre un conjunto de datos diferente. En concreto, se ha partido del anterior y se han escogido al azar 95 observaciones, las cuales han sido modificadas para sacarlas de la nube de puntos y situarlas en la periferia, es decir, se han convertido en anomalías.

Se ha considerado solo el caso de la distancia de Chebyshev con un número fijo de vecinos igual a 1. En la tabla 6.4 se recogen los resultados obtenidos, siendo la primera columna el factor de ponderación del umbral, y las restantes: TP, verdaderos positivos; TN, verdaderos negativos; FP, falsos positivos; y FN, falsos negativos. En este caso, se considera positivo una medición anómala.

Tabla 6.4 Resultados obtenidos sobre un conjunto de datos con anomalías.

Ponderación	TP	TN	FP	FN
1.00	94	1037	175	01
1.25	94	1145	067	01
1.50	93	1184	028	02
1.75	**92**	**1190**	**022**	**03**
2.00	76	1187	025	19

Los resultados obtenidos son muy favorables: a medida que se incrementa el factor de ponderación, el decremento de los falsos positivos es mucho más acusado que el incremento de los falsos negativos. Por lo que aumentar el valor de este parámetro permite, hasta cierto punto, mejorar el desempeño del detector, ya que se equivoca menos tratando las mediciones normales como anómalas, al mismo tiempo que es capaz de discernir casi la totalidad de anomalías. En concreto, para una ponderación de 1.75, el detector presenta un 98 % de exactitud, un 81 % de precisión, un 97 % de sensibilidad y un 98 % de especifidad. Siendo estas métricas calculadas de la siguiente manera:

- **Exactitud**: porcentaje de predicciones acertadas.

$$Exactitud = (TP+TN)/(TP+FP+FN+TN)$$

- **Precisión**: porcentaje de anomalías detectadas correctamente.

$$Precisión = (TP)/(TP+FP)$$

- **Sensibilidad**: porcentaje de anomalías detectadas, o tasa de verdaderos positivos.

$$Sensibilidad = (TP)/(TP+FN)$$

- **Especifidad**: porcentaje de observaciones no anómalas detectadas, o tasa de verdaderos negativos.

$$Especifidad = (TN)/(TN+FP)$$

Por si resulta de interés, en la siguiente secuencia de imágenes se recoge el resultado del algoritmo para el caso concreto de la ponderación igual a 1.75. Los puntos representan mediciones realmente no anómalas, y las cruces mediciones que sí son anomalías. Respecto a los colores, el azul marca las mediciones que conforman los prototipos usados para predecir, el verde, las nuevas mediciones que son identificadas como no anómalas; y el rojo, las nuevas que se detectan como anomalías. La combinación de ambos atributos da lugar a:

- Círculo rojo: Falso positivo.
- Cruz roja: Verdadero positivo.
- Círculo verde: Verdadero negativo.
- Cruz verde: Falso positivo.
- Círculo azul: Previo verdadero negativo usado como prototipo para predecir.
- Cruz azul: Previo falso negativo usado como prototipo para predecir.

Además, el orden de las gráficas viene dado por los números de su encabezado, comenzando en los más pequeños y terminando en los mayores. El paso de una a la siguiente indica que los modelos k-NN han sido reentrenados, lo que ocurre, en concreto, cada 24 detecciones no anómalas.

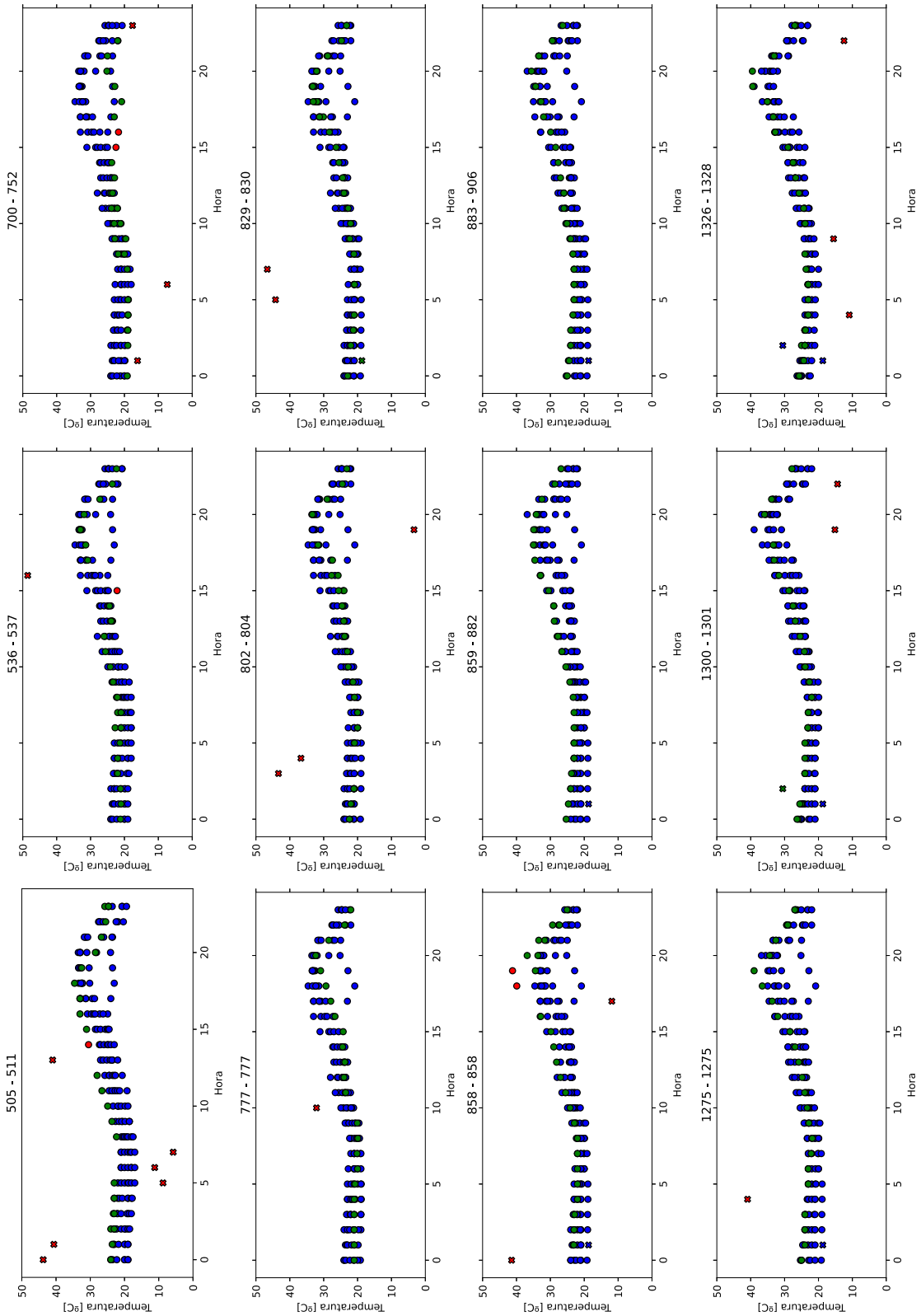

Figura 6.2 Aplicación del algoritmo sobre los datos con anomalías - Parte 1.

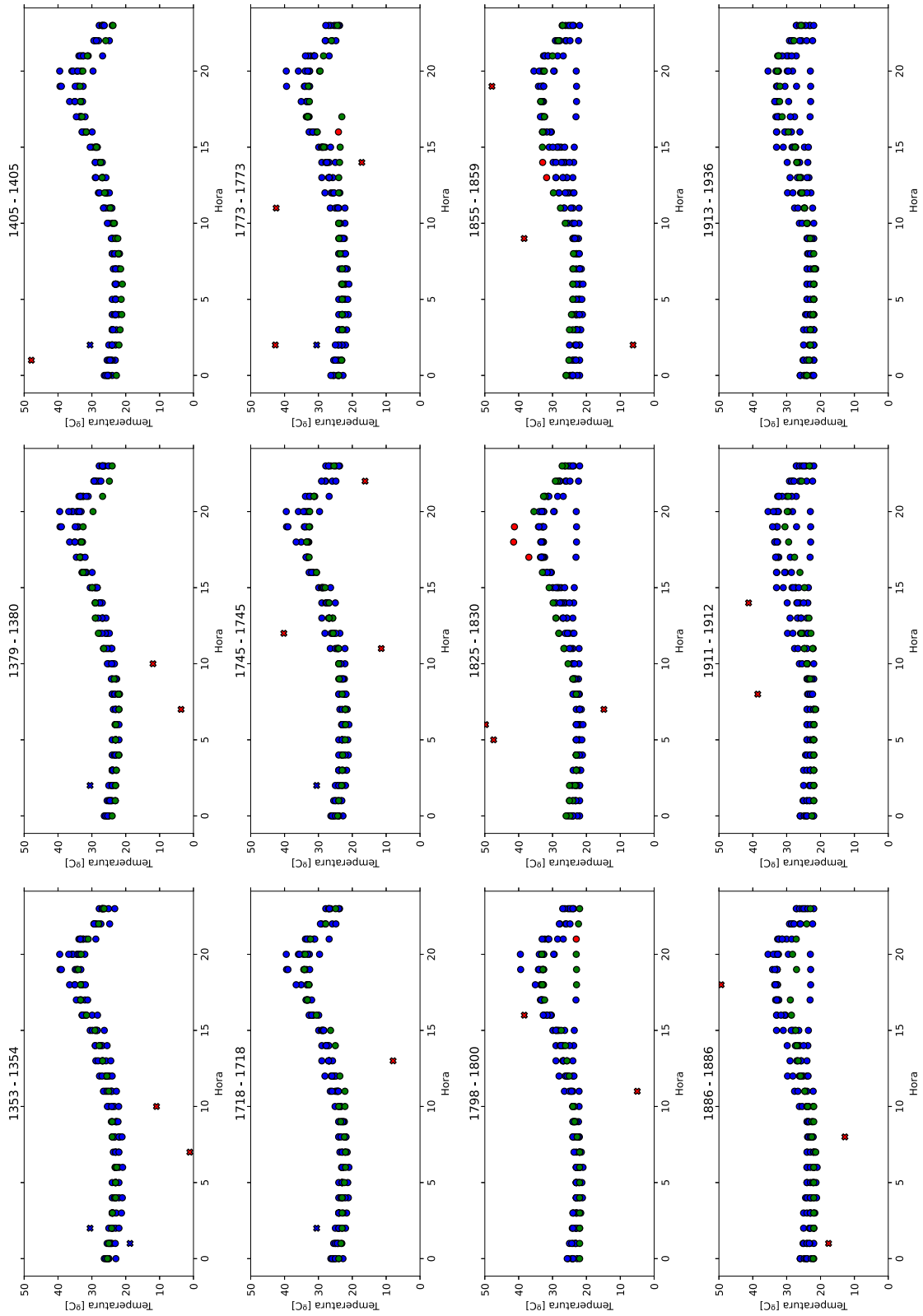

Figura 6.3 Aplicación del algoritmo sobre los datos con anomalías - Parte 2.

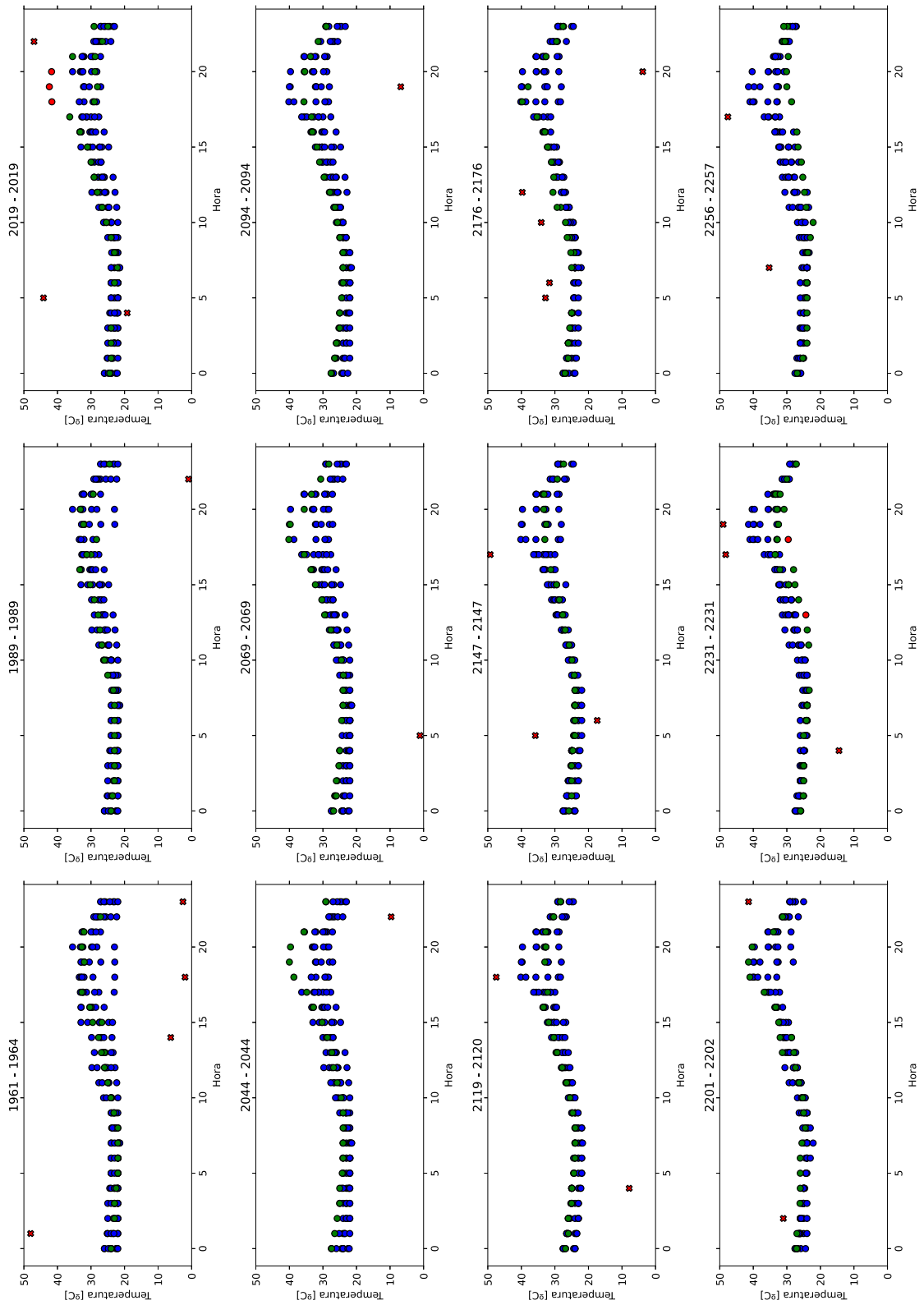

Figura 6.4 Aplicación del algoritmo sobre los datos con anomalías - Parte 3.

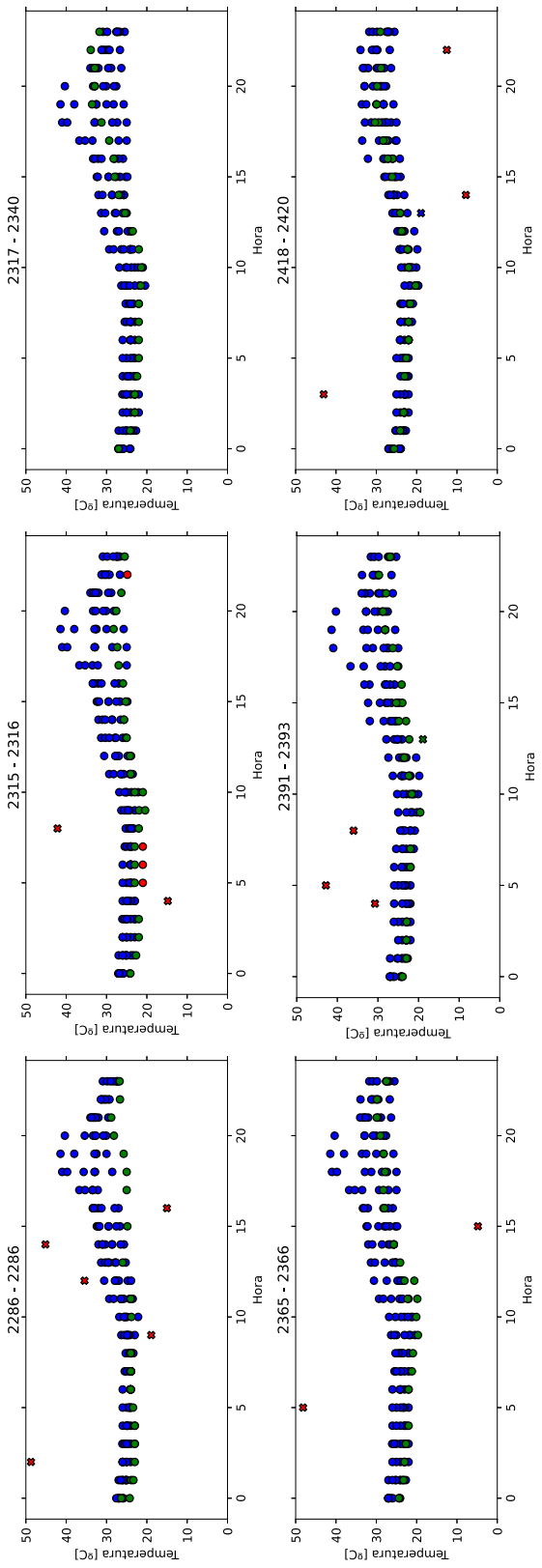

Figura 6.5 Aplicación del algoritmo sobre los datos con anomalías - Parte 4.

7 Conclusiones

El objetivo principal de este trabajo consistía en poder detectar las mediciones anómalas tomadas por sensores de bajo coste y calidad, para así mejorar la fiabilidad de estos dispositivos y facilitar el aprovechamiento posterior de los datos generados.

Como resultado, se ha desarrollado una metodología para la detección de anomalías basada en la transformación del problema original en otro de identificación de outliers, por lo que las técnicas conocidas para resolver este último pueden ser aplicadas en este contexto. En concreto, se ha empleado el método de los k vecinos más cercanos, al cual se le han aplicado una serie de actualizaciones e ideas propias para incrementar su desempeño. En concreto:

- Se ha estudiado la modificación del cálculo de distancias para mejorar la búsqueda de vecinos.
- Se ha estudiado cómo clasificar las observaciones en anómalas y no anómalas en función de su grado de anormalidad dado por el k-NN.
- Se ha estudiado cómo calcular automáticamente el hiperparámetro k, es decir, el número de vecinos considerados.
- Se ha estudiado la fragmentación de la nube de puntos para tratar cada región con un modelo k-NN diferente.
- Se ha estudiado la actualización periódica en el tiempo del detector de anomalías para adecuarlo a las tendencias de variación, lentas y constantes, que sufren las variables registradas por los sensores.

Para poner a prueba la metodología descrita se ha desplegado una red de sensores a modo de banco de pruebas. A nivel físico, cada nodo está constituido por sensores de bajo coste conectados a una Raspberry Pi que se comunica mediante Wi-Fi con un servidor central donde se almacenan los datos y se ejecuta la implementación del detector de anomalías. Con el objetivo de hacer este sistema escalable, flexible y fácil de replicar, se han establecido, durante el desarrollo, diferentes capas de abstracción mediante el uso de APIs y herramientas como Docker.

Los resultados derivados de implementar y evaluar la metodología en el banco de pruebas descrito son prometedores. Destacando, sobre todo, los obtenidos en las mediciones de temperatura, donde se han conseguido valores de exactitud, precisión y sensibilidad superiores al 95 %. Aun así, se puede inferir que el método estudiado tiene aún capacidad de mejora, la cual se puede conseguir con un ajuste más fino de los parámetros o con el estudio de nuevas ideas como la normalización de las variables.

8 Lecciones aprendidas y líneas futuras

Gracias a la diversidad temática de las cuestiones que se han ido planteando y resolviendo a medida que se avanzaba en este proyecto, se han adquirido conocimientos de naturaleza muy variada.

Cuando los nodos de la red se comenzaron a implementar, fue necesario investigar sobre asuntos propios de la Raspberry Pi y su sistema operativo como, por ejemplo, lanzar tras el inicio aplicaciones de manera automática, o el tratamiento de sus pines de uso general para poderla conectar a elementos externos. Además de esto, surgió el primer contacto con las bases de datos relacionales, por el uso de SQLite y el lenguaje SQL.

El conocimiento sobre bases de datos se amplió cuando, posteriormente, se optó por usar PostgreSQL para centralizar todos los registros en un servidor. De esta manera, se aprendió a crear, gestionar, modificar y consultar bases de datos de mayor envergadura.

Tras esto, surgió la necesidad de comunicar los nodos con el servidor, lo que nos brindó la oportunidad de conocer y estudiar las incontables bondades de las APIs y, también, cómo implementarlas. Además, se tuvo un primer contacto con el protocolo de comunicaciones HTTP.

Luego, al comenzar con la detección de anomalías, se leyó una cantidad importante de artículos y estudios de esta temática, lo que nos permitió ampliar conocimientos sobre técnicas de inteligencia artificial y aprendizaje automático. Con la implementación de nuestra metodología, también se tuvo un acercamiento a Docker, mediante el cual se ha desarrollado una estructura basada en contenedores.

Por último, especial mención merece Python. Este lenguaje de programación ha sido el protagonista principal de todas las etapas y elementos que engloba el trabajo. Su uso intensivo nos ha permitido elevar considerablemente el nivel de conocimiento sobre él y sobre un gran número de módulos de los que se ha hecho uso, algunos muy famosos como *numpy*, *pandas*, *matplotlib*, *flask*, *sqlalchemy*, *multiprocessing*, *dash*, *scikit-learn*, *ScyPy*, etc.

Aunque el presente trabajo finalice aquí, deja las puertas abiertas a futuras líneas de investigación. La más directa consistiría en aplicar la metodología expuesta en nuevos problemas, con datos de diferente naturaleza, otros tipos de sensores, con un periodo de variación regular diferente o cualquier otra variación que permita estudiar la sensibilidad del desempeño ante dichas alteraciones.

Por otro lado, se propone continuar con las mismas magnitudes tratadas hasta ahora, temperatura y humedad, y añadir otras más que ayuden a definir el contexto en el que se han tomado las mediciones y que, por consiguiente, contribuyan a afinar la detección de anomalías. Así, por ejemplo, se podrían incluir la radiación solar, la cantidad de precipitaciones, la velocidad y dirección del viento, el porcentaje de nubes, la presión atmosférica, etc. Estas podrían ser tratadas como nuevas variable de entrada que amplían el espacio de características, o como variables de decisión para determinar qué detector debe tratar cada observación. Esta última idea consiste en entrenar varios modelos y usar uno u otro en función de las nuevas variables. Por ejemplo, se podría entrenar un modelo usando solo los datos registrados de temperatura en los días que no ha llovido, y otro modelo con las temperaturas de los días que si ha llovido. Luego, para determinar si una observación futura es anómala, se estudiará con un modelo u otro en función de si llovía o no el día que fue tomada.

Con relación al método de los k vecinos más cercanos, se podría estudiar qué efecto tiene la normalización de las variables de entrada, incluida la temporal, en los resultados obtenidos. La mayoría de métricas se fundamentan en la suma de las diferencia de valores de cada dimensión, por lo que las variaciones de las variables de mayor orden de magnitud podrían enmascarar y anular la información aportada por la variación de las variables que se desarrollan en escalas de medida más pequeñas. Por ejemplo, si en el problema se tratase con la variable temperatura en grados centígrados y con la humedad relativa en por unidad, el mínimo cambio que se puede dar en la primera, variación de un grado, afecta al cálculo de la distancia tanto como el máximo cambio que se puede dar en la segunda, pasar de la mínima humedad igual a 0 a la máxima igual a 1, o viceversa. La variable temperatura sería claramente dominante y la vecindad sería determinada por ella sin importar prácticamente el valor de la humedad.

Este hecho se representa en el ejemplo de la figura 8.1. En él se observa como un cambio de apenas el 14 % en la temperatura es capaz de enmascarar una variación del 100 % en la humedad: la distancia entre los puntos solo se modifica un 2.4 %.

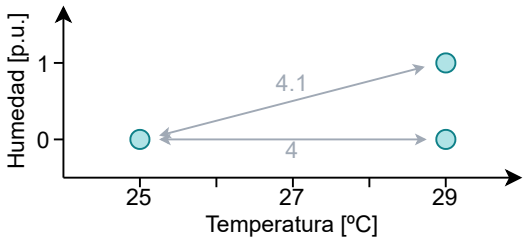

Figura 8.1 Ejemplo gráfico para demostrar la importancia de normalizar las variables.

Para normalizar el espacio de características se pueden emplear unidades tipificadas, expresión 8.1. Estas transforman las variables de entrada en otras con media nula y desviación típica igual a uno, por lo que los datos se vuelven independientes de la unidad o escala escogida. Otra opción consistiría en usar el máximo y mínimo de cada variable para normalizarla en el intervalo $[0, 1]$, expresión 8.2. Si por algún motivo no se quisiese modificar directamente el valor de las variables, se pueden conseguir resultados similares ponderado cada dimensión en la ecuación del cálculo de la distancia para otorgar más peso a aquellas cuya escala sea menor. Esto último puede ser especialmente útil en el eje temporal, puesto que permite dilatarlo o contraerlo sin afectar a nada más, haciéndolo más o menos representativo respecto al resto de variables.

$$X' = \frac{X - \mu}{\sigma} \tag{8.1}$$

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{8.2}$$

Otra alternativa de estudio consistiría en sustituir el detector de outliers basado en el método de los k vecinos más cercano por otro que también se pueda aplicar a la nube de puntos obtenida a partir de la transformación inicial de los datos originales. Por ejemplo, se podría empezar probando por el conocido como Local Outlier Factor (LOF) [27], el cual calcula la anormalidad de cada observación en función de la densidad de población de su área; Angle-Based Outlier Detection (ABOD) [28], que hace uso de la varianza del ángulo formado entre pares de puntos en vez de la distancia entre ellos; y Feature Bagging [29], que primero busca outliers en múltiples subconjuntos de variables de entrada y luego combina los resultados para dar un valor de anormalidad final a cada observación. Siguiendo la filosofía del último ejemplo de combinar múltiples resultados, también se propone hacer uso de más de un detector y aprovechar los resultados de todos ellos para llegar a una conclusión final de cuáles son las observaciones anómalas. De esta manera, se aprovecharía de manera conjunta las ventajas de cada método.

Por último, aparte de estudiar y comparar el desempeño de la metodología con otros detectores, sería interesante hacer la misma comparación con otras aproximaciones a la detección de anomalías totalmente diferentes. Así se podrían poner los resultados obtenidos en contexto para conocer cómo de buenos o malos son respecto a otros.

Anexo A

Modelo ERD de la base de datos

En la figura A.1 se muestra el modelo entidad-relación de la base de datos PostgreSQL desplegada en el servidor. El contenido de las tablas y sus campos es el siguiente:

- **ft_agricultura**: Contiene los datos tal como son enviados por el nodo.
  - *id_measurement*: Identificador único de cada medición.
  - *id_rp*: Identificador del nodo que ha generado la medición.
  - *id_sensor*: Identificador del sensor dentro del nodo que ha generado la medición.
  - *temperature*: Temperatura medida.
  - *measurement_date*: Marca de tiempo del instante de medición.
  - *humidity*: Humedad relativa medida.
  - *save_date*: Marca de tiempo del instante de almacenamiento en la BBDD de la medición.
- **agricultura**: Surge de un preprocesamiento a partir de la tabla anterior. En concreto, se completan los huecos temporales en los que se ausenten una o más mediciones.
  - *is_synthetic*: Indica si la medición ha sido generada a partir de los datos de otras para completar un hueco de registros ausentes.
  - *is_null*: Indica si se ha generado una medición vacía para completar un hueco de registros ausentes.
- **fta_agricultura_1day**: Contiene las agregaciones con granularidad diaria calculadas a partir de la tabla de datos preprocesados.
  - *temperature_mean*: Media aritmética de las temperaturas que engloba la agregación.
  - *temperature_variance*: Desviación típica de las temperaturas agregadas.
  - *humidity_mean*: Media aritmética de las humedades que engloba la agregación.
  - *humidity_variance*: Desviación típica de las humedades agregadas.
  - *date*: Día de la agregación.
  - *any_synthethic*: Indica si para la agregación se ha usado alguna medición sintética.
  - *qty_synthetic*: Número de mediciones sintéticas.
  - *any_null*: Indica si para la agregación se ha usado alguna medición vacía.
  - *qty_null*: Número de mediciones vacías.
  - *aggregtion_size*: Numero de mediciones que engloba la agregación.

- **fta_agricultura_1hour**: Contiene las agregaciones con granularidad horaria calculadas a partir de la tabla de datos preprocesados.
  - *aggregation_hour*: Hora de la agregación.
- **anomaly_detection**: Almacena el resultado de aplicar el detector de anomalías sobre la tabla de agregaciones horarias.
  - *id_detection*: Identificador único de la detección realizada sobre una agregación.
  - *id_aggregation*: Identificador de la agregación horaria estudiada.
  - *is_anomaly*: Valor binario que refleja si el detector considera la agregación como anómala o no.
  - *abnormality_score*: Valor que cuantifica la anormalidad de la agregación.
  - *id_model*: Identificador del modelo empleado para tratar la agregación.
- **models**: Guarda información relativa a los modelos usados durante la detección.
  - *n_neighbors*: Parámetro k, número de vecinos, del modelo.
  - *metric*: Métrica empleada por el modelo.
  - *first_datetime*: Marca de tiempo de la agregación más antigua usada como prototipo del modelo.
  - *last_datetime*: Marca de tiempo de la agregación más reciente usada como prototipo del modelo.
  - *qty_prototypes*: Cantidad de prototipos que emplea el modelo.

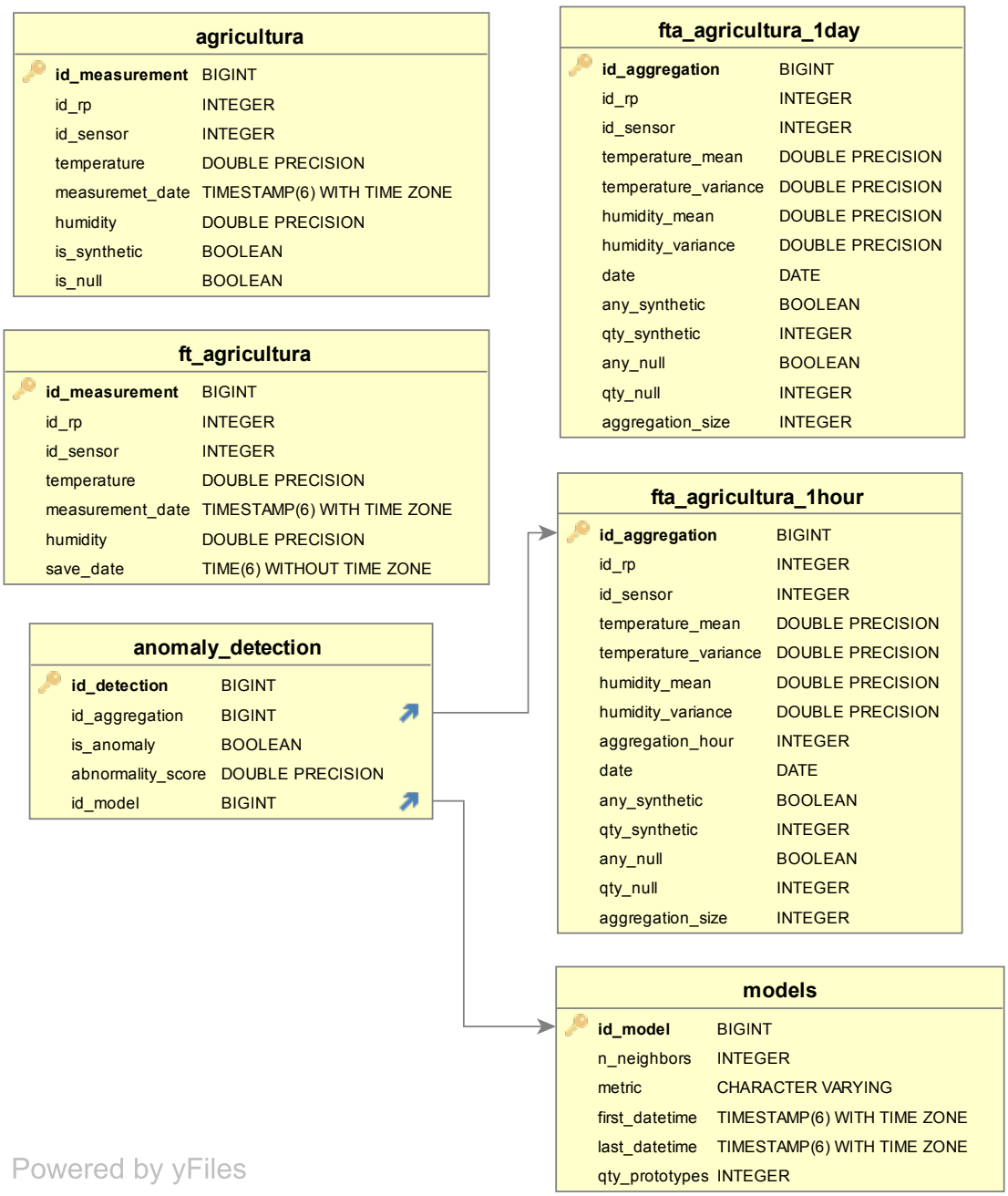

Figura A.1 Modelo de la base de datos en el servidor.

Bibliografía

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[2] R. Pincus, "Barnett, v., and lewis t.: Outliers in statistical data. 3rd edition. j. wiley & sons 1994, xvii. 582 pp." *Biometrical Journal*, vol. 37, no. 2, pp. 256–256, 1995. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/bimj.4710370219

[3] "Docker," accessed: 2021-07-07. [Online]. Available: https://www.docker.com/

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[5] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, 2003.

[6] Z. Tang, Z. Chen, Y. Bao, and H. Li, "Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring," *Structural Control and Health Monitoring*, vol. 26, no. 1, p. e2296, 2019, e2296 STC-18-0112.R1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/stc.2296

[7] L. H. Nguyen and J.-A. Goulet, "Anomaly detection with the switching kalman filter for structural health monitoring," *Structural Control and Health Monitoring*, vol. 25, no. 4, p. e2136, 2018, e2136 stc.2136. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/stc.2136

[8] A. Taboada-Crispi, H. Sahli, D. Hernandez-Pacheco, and A. Falcon-Ruiz, "Anomaly detection in medical image analysis," in *Handbook of research on advanced techniques in diagnostic imaging and biomedical applications*. IGI Global, 2009, pp. 426–446.

[9] Y. Kukita, J. Uchida, S. Oba, K. Nishino, T. Kumagai, K. Taniguchi, T. Okuyama, F. Imamura, and K. Kato, "Quantitative identification of mutant alleles derived from lung cancer in plasma cell-free dna via anomaly detection using deep sequencing data," *PloS one*, vol. 8, no. 11, p. e81468, 2013.

[10] M. Ahmed, A. Mahmood, and M. Islam, "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278–288, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X15000023

[11] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," 2016.

[12] M. A. Hayes and M. A. Capretz, "Contextual anomaly detection in big sensor data," in *2014 IEEE International Congress on Big Data*, 2014, pp. 64–71.

[13] D. Janakiram, A. Kumar, and A. M. Reddy V., "Outlier detection in wireless sensor networks using bayesian belief networks," in *2006 1st International Conference on Communication Systems Software Middleware*, 2006, pp. 1–6.

[14] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan, "Online anomaly detection for sensor systems: A simple and efficient approach," *Performance Evaluation*, vol. 67, no. 11, pp. 1059–1075, 2010, performance 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016653161000115X

[15] "Average daily temperature archive," niversity of Dayton, Environmental Protection Agency. [Online]. Available: https://academic.udayton.edu/kissock/http/Weather/default.htm

[16] "Opendata." [Online]. Available: https://opendata.aemet.es/centrodedescargas/inicio

[17] "Raspberry pi 3 model a+," accessed: 2021-07-07. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/

[18] "Dht11 sip packaged temperature and humidity sensor," accessed: 2021-07-07. [Online]. Available: http://www.aosong.com/en/products-21.html

[19] Python Software Foundation, "multiprocessing - Process-based parallelism," accessed: 2021-07-07. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html

[20] ——, "signal - Set handlers for asynchronous events," accessed: 2021-07-07. [Online]. Available: https://docs.python.org/3/library/signal.html

[21] ——, "sqlite3 - DB-API 2.0 interface for SQLite databases," accessed: 2021-07-07. [Online]. Available: https://docs.python.org/3/library/sqlite3.html

[22] T. DiCola, "Adafruit python dht sensor library," accessed: 2021-07-07. [Online]. Available: https://github.com/adafruit/Adafruit_Python_DHT/

[23] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.

[24] R. T. Fielding and R. N. Taylor, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[25] A. Ronacher, "Flask's documentation," accessed: 2021-07-07. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/

[26] M. Bayer, "The Python SQL Toolkit and Object Relational Mapper," accessed: 2021-07-07. [Online]. Available: https://www.sqlalchemy.org/

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, May 2000. [Online]. Available: https://doi.org/10.1145/335191.335388

[28] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 444–452. [Online]. Available: https://doi.org/10.1145/1401890.1401946

[29] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 157–166. [Online]. Available: https://doi.org/10.1145/1081870.1081891

[30] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320307000027

[31] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996.

[32] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126. ICML United States, 2003.

[33] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989. [Online]. Available: http://www.jstor.org/stable/1403797

[34] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, no. 2, p. 427–438, May 2000. [Online]. Available: https://doi.org/10.1145/335191.335437

[35] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Principles of Data Mining and Knowledge Discovery*, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 15–27.

[36] "Python data analysis library," accessed: 2021-07-07. [Online]. Available: https://pandas.pydata.org/

[37] "The fundamental package for scientific computing with python," accessed: 2021-07-07. [Online]. Available: https://numpy.org/

[38] Python Software Foundation, "datetime — Basic date and time types," accessed: 2021-07-07. [Online]. Available: https://docs.python.org/3/library/datetime.html

[39] “Requests: HTTP for Humans,” accessed: 2021-07-07. [Online]. Available: https://docs.python-requests.org/en/master/

[40] “Numba: a High Performance Python Compiler,” accessed: 2021-07-07. [Online]. Available: http://numba.pydata.org/

[41] Y. Zhao, Z. Nasrullah, and Z. Li, “Pyod: A python toolbox for scalable outlier detection,” *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html