Trabajo de Fin de Master:

Detección de Anomalías en Series Temporales

Autor:

Sergio Pastor Medrano

Directores:

Dr. Rafael del Hoyo

David Abadía

Ponente:

Dra. María Beatriz Lacruz Casaucau

Universidad de Zaragoza

Facultad de Ciencias

Departamento de Métodos Estadísticos

Noviembre 2022

Agradecimientos

A mi madre por su apoyo incondicional en casa
y a mi padre por el apoyo desde la distancia.

.

A mis tutores David y Rafael, por la paciencia, pero sobre todo
por la confianza de que todo iba a salir bien.

.

A todos mis compañeros del departamento de Big Data y Sistemas Cognitivos,
todo es mucho más fácil cuando uno se rodea de los mejores, y vosotros lo sois.

Resumen

La detección de anomalías es uno de los temas más populares en el mundo de la ciencia de datos por sus múltiples aplicaciones prácticas. En concreto, el estudio de anomalías en series temporales es un problema ampliamente investigado y desarrollado a lo largo de la historia, nutriéndose tanto de técnicas estadísticas como de los algoritmos de aprendizaje automático y profundo que han ido surgiendo con los años. Sin embargo, existen muy pocos trabajos en la literatura que comparen técnicas de detección de anomalías procedentes de métodos estadísticos, algoritmos de aprendizaje automático y algoritmos de aprendizaje profundo. Por eso, en este trabajo, se analizan y desarrollan algoritmos procedentes de los tres campos mencionados anteriormente, complementados con transformaciones de las series temporales, con el objetivo de analizar la efectividad de cada algoritmo para diversas situaciones y anomalías. El análisis del desempeño de cada algoritmo se realizará a través de conjuntos de series temporales públicos con anomalías identificadas y pertenecientes a distintas categorías; desarrollando un formalismo matemático válido para llevar a cabo dicha tarea, y utilizando métricas capaces de representar adecuadamente el desempeño de los modelos de detección de anomalías.

Anomaly detection is one of the most popular topics in the world of data science due to its many practical applications. Specifically, the study of anomalies in time series is a problem that has been widely researched and developed throughout history, drawing on both statistical techniques and deep and machine learning algorithms that have emerged over the years. However, there are very few works in the literature comparing anomaly detection techniques from statistical methods, machine learning algorithms and deep learning algorithms. Therefore, in this paper, algorithms from the three fields mentioned above are analyzed and developed, complemented with time series transformations, with the objective of analyzing the effectiveness of each algorithm for various situations and anomalies. The analysis of the performance of each algorithm will be performed through sets of public time series with identified anomalies and belonging to different categories; developing a valid mathematical formalism to carry out such task, and using metrics capable of adequately representing the performance of the anomaly detection models.

Índice general

Capítulo 1

Introducción

Nos encontramos en un mundo en constante evolución y digitalización en el que la inmensa cantidad de datos que se atesoran excede la capacidad humana de analizarlos manualmente. Esto ha provocado el innegable avance de la Inteligencia Artificial, a través de tecnologías como el aprendizaje automático y aprendizaje profundo, que permiten el análisis y tratamiento automático de los datos, así como explotar el 100 % de la información que esconden. La Inteligencia Artificial es ya una herramienta fundamental en muchos campos como la salud [1], aplicaciones financieras como la segmentación de clientes [2] e incluso arte con los nuevos modelos de generación de imágenes tales como Dalle de OpenAi [3] o Stable Diffusion de Stability.Ai [4].
Una de estas aplicaciones de la Inteligencia Artificial es la detección de anomalías en los datos. Si definimos burdamente una anomalía como un punto que difiere de la gran mayoría de puntos restante, la detección de anomalías consiste en separar dichos puntos de los puntos considerados como 'normales'.

En este capítulo se explica el porqué de la elección del tema del trabajo, así como los objetivos deseados a cumplir con el desarrollo de este.

1.1. Motivación

La elección de este trabajo se da principalmente debido a mi experiencia laboral en el Instituto Tecnológico de Aragón, lugar donde he llevado a cabo mis prácticas en el departamento de Inteligencia Artificial. Dentro de este departamento he desarrollado tareas de minería de datos que incluyen el análisis, visualización y modelado de estos. En concreto, he participado en varios proyectos en el que se trabajaba en el campo de las series temporales y la detección de anomalías, lo que causó un gran interés en mí.

Tras ver la comunidad tan grande que existe actualmente en el mundo de la ciencia de datos, decidí contribuir con una revisión y ligera modificación de algoritmos clásicos y recientes de detección de anomalías procedentes de las diferentes ramas de la inteligencia artificial. De esta manera, pretendía ganar conocimiento y poder entender y modelar problemas mediante técnicas

de aprendizaje automático y aprendizaje profundo.
Estas modificaciones incluyen cambios sobre la serie de datos a través de diferenciaciones y transformaciones de los datos, así como cambios sobre el algoritmo, ya sea a través de ventanas o incorporación de reglas para la selección de anomalías. Además, quería ser capaz de poder organizar los métodos estudiados según la metodología que utilizaban para la detección de anomalías, no segmentar los algoritmos según si pertenecían a métodos de aprendizaje profundo, aprendizaje automático o métodos estadísticos.
Debido a mi especialización laboral en la detección de anomalías en series temporales, quería comparar los resultados obtenidos por los diferentes métodos para diversas series, ganando experiencia para identificar qué metodología aplicar en futuros proyectos según la tipología de la serie temporal. Además, estaba interesado en establecer una formulación matemática detallada para la detección de anomalías. De esta manera se pretendía entender el problema matemáticamente y mejorar el conocimiento básico de campos como el aprendizaje automático y profundo.
Finalmente, debido a mi formación en visualización de datos y creación de pequeñas interfaces interactivas o *web apps* con lenguaje de programación Python, quería ser capaz de desarrollar para la comunidad y para la empresa una interfaz sencilla y fácil de usar donde poder visualizar el desempeño de algoritmos de detección de anomalías en series temporales.

1.2. Objetivos

El objetivo principal de este trabajo es analizar el comportamiento general de los diferentes algoritmos de detección de anomalías al aplicarlos en series temporales univariantes. Se estudiarán diferentes técnicas de detección de anomalías que incluyen desde métodos puramente estadísticos, pasando por técnicas de aprendizaje automático o *Machine Learning* y finalizando con métodos avanzados y actuales de Aprendizaje Profundo o *Deep Learning*.
De esta manera se quiere analizar la utilidad de estos algoritmos ante anomalías en series temporales de diversa naturaleza. Para llevar a cabo este objetivo principal, este se desglosó en varios:

- Estudiar la literatura existente sobre la detección de anomalías y su aplicación a series temporales.
- Definir el problema de la detección de anomalías en términos matemáticos, para contar con un formalismo claro a seguir por cada algoritmo y válido para cualquier tipo de problema dentro del tema del trabajo.
- Encontrar un conjunto de datos de series temporales univariantes con anomalías etiquetadas utilizado en trabajos similares que sea válido para analizar el comportamiento de los algoritmos definidos en el trabajo.
- Evaluar de manera efectiva el desempeño de los algoritmos a través de métricas que sean válidas para clases no balanceadas.
- Incrementar el estado del arte actual, comparando desde modelos sencillos computacionalmente basados en la estadística; hasta complejos modelos basados en las

arquitecturas recientes del aprendizaje profundo, valorando las ventajas y desventajas de cada uno.

- Incluir transformaciones en las series de datos como diferenciaciones o uso de ventanas móviles que faciliten la tarea de la detección de anomalías al transformar la serie temporal.
- Comparar el desempeño de cada algoritmo en diferentes escenarios para poder extraer conclusiones en función de las características temporales de la serie y las anomalías.
- Desarrollar una herramienta interactiva y válida para cualquier serie temporal univariante importada por el usuario, que facilite el estudio de anomalías, así como su detección mediante el conjunto de algoritmos descritos, pudiendo visualizar de manera clara y sencilla el desempeño de cada uno de ellos a modo de selección inicial de modelos.

Capítulo 2

Contexto y estado del arte

En este capítulo se analiza el concepto clave del trabajo: la detección de anomalías, explicando las características básicas que debe cumplir una anomalía para ser considerada como tal, así como clasificando estas anomalías según su tipología. Además, se introduce al lector al tipo de series de datos sobre las que se va a estudiar la detección de anomalías: las series temporales, mostrando las diferentes características y componentes que pueden poseer, los conceptos de ventanas móviles y diferenciación de la serie, claves en el uso de los algoritmos del trabajo, y finalizando con la explicación algunos modelos básicos utilizados en series temporales para mostrar que valores anteriores y variables aleatorias pueden utilizarse para la modelización de series temporales.

2.1. Detección de anomalías

La detección de anomalías, aparte de ser una tarea necesaria siempre que se va a trabajar con datos, es fundamental y protagonista en campos como la medicina, donde es clave para la detección de enfermedades y tumores, así como para eliminar el ruido de la señal o imágenes médicas estudiadas [5]; en ciberseguridad donde una anomalía puede representar una intrusión o ataque al sistema informático e incluso en actividades financieras donde la detección de una anomalía en las transacciones puede significar evitar fraudes [6].

2.1.1. ¿Qué es una anomalía?

Para poder formular matemáticamente el problema de la detección de anomalías, así como diseñar una metodología válida a aplicar, es necesario definir el concepto de *anomalía*. Sin embargo, no existe una definición única en la literatura.
La definición más común es la expresada por Chandola *et al* [7]: "Las anomalías son patrones en los datos que no se ajustan a una noción bien definida de comportamiento normal". Barnert y Lewis [8] fueron más específicos al definir una anomalía como "Una observación o conjunto de observaciones que aparentan ser inconsistentes con el resto de las observaciones de la serie de datos".
En otras palabras, puede pensarse en ellos como aquellos puntos generados por mecanismos diferentes al del resto de puntos. Todas estas definiciones están de acuerdo en dos características

[9]:

- La distribución espacial de los puntos anómalos difiere sustancialmente de la distribución general de la serie de datos.
- Las anomalías representan un porcentaje muy pequeño de la serie de datos.

Por ejemplo, en la Figura 2.1, se representan los puntos pertenecientes a un conjunto de datos bidimensional. Existen dos regiones claramente diferenciadas en las que se encuentran agrupados la gran mayoría de los puntos, pudiendo ser denominadas, por tanto, como regiones normales, pues los puntos contenidos representan el comportamiento general de la serie.

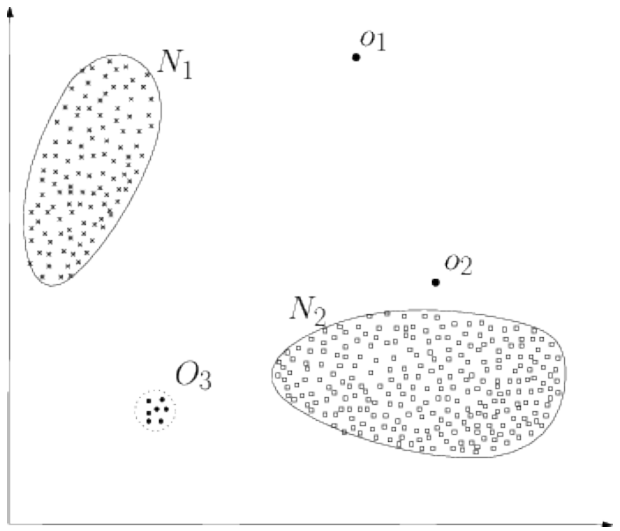


Figura 2.1: Anomalías en un espacio bidimensional. Fuente: [7]

En el caso contrario tenemos los puntos o_1 y o_2, totalmente aislados del resto y, por tanto, de la distribución general de la serie, siendo dos anomalías claras.

Además, los puntos en la región (O_3) también son puntos anómalos, pues aunque sí que se encuentran en una región formada por más de un punto, el porcentaje de puntos pertenecientes a esta región no es comparable al de las regiones (N_1) y (N_2), claramente separados de (N_3), por lo que provienen de una distribución diferente a la distribución de la mayoría de los puntos de la serie.

2.1.2. Tipos de Anomalías

Es posible encontrarse con anomalías muy diferentes; sin embargo, existen tres tipos de anomalías que son las más representativas [7]:

1. **Anomalías puntuales:** Puntos individuales que se consideran anómalos con respecto al resto de los datos. Como ejemplo aplicado a la vida real, un ingreso de dinero muy grande en la cuenta bancaria de un individuo con respecto del resto de sus ingresos bancarios. De esta manera, dado el instante de tiempo t, y una ventana de tamaño $2w + 1$, $w \in \mathbb{R}$

suficientemente grande, y centrada en t , un punto X_t es considerado como un punto anómalo de carácter puntual si se desvía significativamente del resto de puntos de la ventana $\{X_{t-w}, X_{t-w+1}, ..., X_{t+w-1}, X_{t+w}\}$.

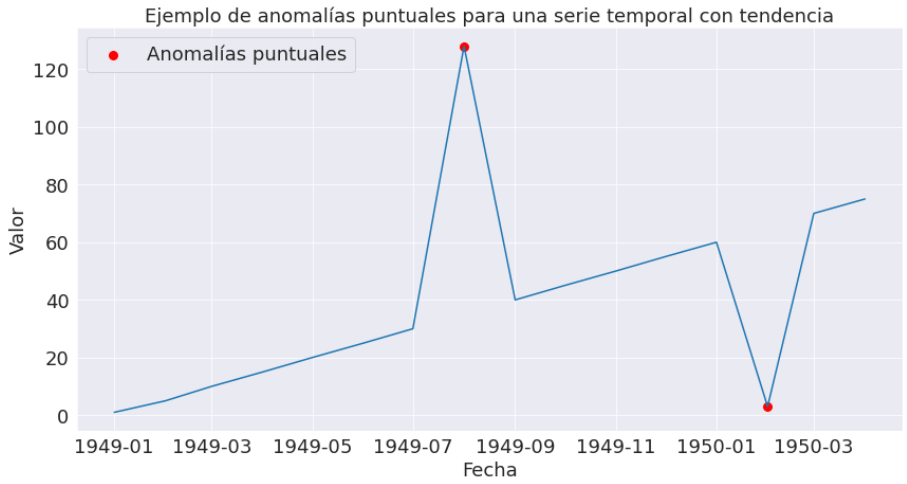


Figura 2.2: Anomalías puntuales para una serie temporal con tendencia positiva en el tiempo. Fuente: propia.

2. **Anomalías colectivas:** Se trata de puntos consecutivos en el tiempo cuyo comportamiento es inusual, aunque cada punto no tiene por qué ser necesariamente una anomalía puntual. En la Figura 2.3 vemos el ejemplo de una serie temporal correspondiente a un electrocardiograma. La región en rojo representa una anomalía debido a que la serie permanece en torno a un cierto valor por un periodo anormalmente largo de tiempo que rompe el comportamiento estacional de la serie, aunque dicho valor sea del mismo orden que el resto de valores de la serie.

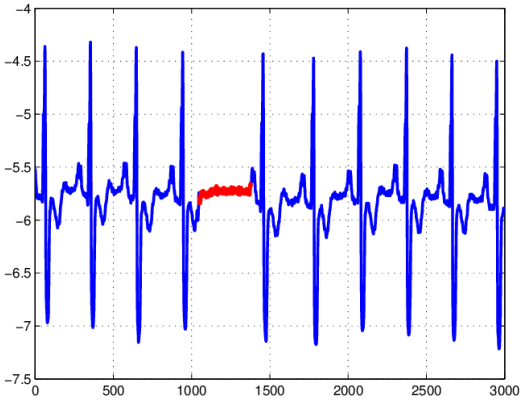


Figura 2.3: Anomalía colectiva para un electrocardiograma humano. Fuente: [7].

3. **Anomalías Contextuales:** Se trata de datos que pueden parecer normales al principio, pero se consideran anomalías en sus respectivos contextos. Un punto anómalo para un contexto puede ser no-anómalo para otro contexto. Por ejemplo, una temperatura media en Alemania de 35 grados no tiene que ser necesariamente una anomalía si nos encontramos

en el mes de agosto, pero sin duda lo será para un mes invernal. En la Figura 2.4, extraída de [10], se puede observar un ejemplo de cada una de las tres anomalías explicadas. La anomalía contextual se encuentra dentro del rango de valores considerados como 'normal' y de haberse dado en alguno de los instantes t_1, t_2, t_3 o t_4 no se consideraría anomalía; sin embargo, en el contexto de la curva sinusoidal que describe la serie temporal, se trata de un punto anómalo que no sigue dicho comportamiento.

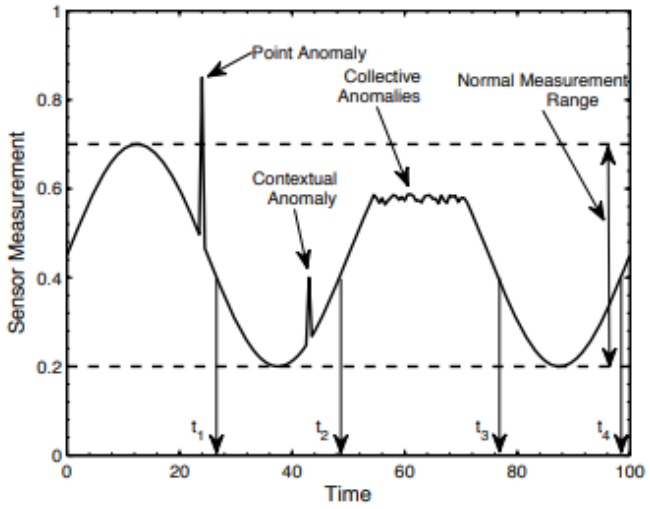


Figura 2.4: Anomalías puntual, contextual y colectiva recogidas por un sensor de red inalámbrico. Fuente: [10].

2.2. Series Temporales

Las diferentes técnicas de detección de anomalías dependen directamente de la tipología de los datos de estudio. No es lo mismo tratar con datos sin correlación espacio-temporal como pueden ser estudios clínicos, qué datos con correlación espacial como imágenes o datos con correlaciones temporales como las series de datos.
Es en este último ejemplo, en las series temporales donde se centra este trabajo. Existen multitud de procesos cuyos datos mantienen correlaciones temporales, como puede ser las temperaturas de una ciudad o las concentraciones de contaminantes en el aire. En estos casos, la detección de anomalías tiene que adaptarse y ser capaz de 'entender' estas relaciones entre los datos pasados y futuros. Las diferentes series de datos que se han analizado en este trabajo son series temporales. Por ello, es necesario definir y explicar en que consisten las series temporales. Sin embargo, antes se debe definir un concepto previo necesario como es el concepto de **proceso estocástico** [11]:

Definición 2.2.1 (**Proceso Estocástico**)**.** Dado un espacio de probabilidad (Γ, F, P) se denomina proceso estocástico a toda familia de variables o vectores aleatorios $\{X_t\}_{t\in\mathbb{T}}$ definidas en ese espacio. Esa familia está caracterizada por tres elementos principales:

- El conjunto de índices, $\mathbb{T}$ el cual puede ser discreto ($\mathbb{T} = 0, 1, 2, ...,$) o continuo ($\mathbb{T} = [0, \infty)$). Este conjunto de índices puede ser unidimensional o multidimensional.
- El espacio de estados $\mathbb{S}$ definido como el conjunto donde toman valores las variables

aleatorias que forman el proceso. Al igual que el conjunto $\mathbb{T}$, $\mathbb{S}$ también puede ser discreto o continuo, y unidimensional o multidimensional.

- La relación de dependencia entre las variables aleatorias del proceso.

Un proceso estocástico queda caracterizado por la distribución conjunta de los vectores de variables aleatorias $(X_1, ..., X_T)$ para todo T. Estas distribuciones se denominan distribuciones finito-dimensionales y determinan las distribuciones marginales de cada X_T. Una serie temporal es un proceso estocástico donde los elementos están ordenados cronológicamente.
El problema de las series temporales es que determinar la distribución conjunta de estos procesos es complicado.

Un proceso estocástico básico que se utiliza en la definición de otros muchos es el *ruido blanco*. Este proceso está formado por variables incorreladas e idénticamente distribuidas, con media nula y varianza σ^2, por lo que frecuentemente se denomina al ruido blanco como $\mathrm{WN}(0, \sigma^2)$. La clave es que si se impone la hipótesis de normalidad, la incorrelación es equivalente a la independencia, por lo que se trata de un proceso de observaciones independientes e idénticamente distribuidas.

Los elementos de una serie temporal no han sido generados por procesos independientes: las observaciones pasadas influyen en las observaciones presentes. Para identificar el grado de dependencia entre valores pasados y presentes se utilizan dos funciones: la función de autocorrelación y la función de autocorrelación parcial.

La función de **autocorrelación (ACF)** mide la correlación entre dos variables del mismo proceso estocástico separadas por h periodos. Esta función de autocorrelación para un proceso estacionario en el retardo h se define como $\rho(h) = Corr[X_{t+h}, X_t] = \gamma(h)/\gamma(0)$ con $\gamma(h)$ la función de autocovarianzas. Para estimar la función de autocovarianzas se utiliza su correspondiente valor muestral $\hat{\gamma}(h) = \frac{1}{T-|h|}\sum_{t=1}^{T-|h|}(x_{t+|h|} - \bar{x})(x_t - \bar{x})$ de manera que $\hat{\rho}(h) = \hat{\gamma}(h)/\hat{\gamma}(0)$.

La función de **autocorrelación parcial (PACF)** mide la correlación entre dos variables del mismo proceso estocástico separadas por h periodos cuando se obvia la dependencia creada por los retardos intermedios existentes entre ambas. Esto es equivalente a medir la autocorrelación entre dos variables sin tener en cuenta los efectos de las variables intermedias.

2.2.1. Tipos de Series Temporales

Las series temporales pueden adoptar diferentes comportamientos a lo largo del tiempo [12]:

- **Serie estacionaria**: Serie estable con valores que oscilan en torno a un nivel medio **fijo** con una variabilidad **constante** en el tiempo. Un ejemplo de serie estacionaria es la generada por los resultados de apostar a cara o cruz a lo largo del tiempo. Matemáticamente, una

serie temporal $\{X_1, \ldots, X_T\}$ es estacionaria si cumple las siguientes condiciones:

$$E[X_t] = \mu \; \forall \, t = \{1, \ldots, T\} \tag{2.1}$$

$$Var(X_t) = \sigma^2 \; \forall \, t = \{1, \ldots, T\} \tag{2.2}$$

$$Cov[X_t, X_s] = \gamma \; \forall \, t, s = \{1, \ldots, T\} \tag{2.3}$$

■ **Series no estacionarias en media**

- **Serie con Tendencia**: Una serie cuya media evoluciona a lo largo del tiempo. Un modelo simple para describir una serie con tendencia es

$$X_t = m_t + Y_t, \tag{2.4}$$

donde m_t es la componente de tendencia e Y_t es una serie estacionaria de media nula. Un ejemplo de serie con tendencia es el número personas mayores de 80 años en el mundo lo largo del tiempo. [13].

- **Serie estacional**: Una serie estacional es una serie cuyo comportamiento se repite de manera periódica. Un modelo simple de serie estacional es:

$$X_t = s_t + Y_t, \tag{2.5}$$

con s_t la componente estacional de periodo S: $s_{t-S} = s_t = s_{t+S}$, e Y_t una serie estacionaria de media nula. Un ejemplo de serie estacional es la temperatura media mensual de un municipio, con una clara periodicidad anual.

- **Serie con tendencia y componente estacional**: Se trata de una serie con ambos comportamientos. Un modelo básico, por tanto, consiste en

$$X_t = s_t + m_t + Y_t, \tag{2.6}$$

donde m_t es la componente de tendencia, s_t la componente estacional de periodo S, e Y_t es una serie estacionaria de media nula.

■ **Series no estacionarias en varianza**: Series en las que la varianza de X_t depende del instante t.

2.2.2. Diferenciación de la serie

Una manera sencilla de eliminar la tendencia y la componente estacional de una serie es aplicar el operador diferencia ∇ tal que $\nabla X_t = X_t - X_{t-1}$. Una serie con tendencia polinómica de orden k puede convertirse en una serie estacionaria al aplicar el operador ∇ k veces, ∇^k. Supóngase qué $k = 1$, y sea la serie $X_t = m_t + Y_t$ con $m_t = a_0 + a_1 t$ e Y_t una serie estacionaria. Entonces $\nabla m_t = m_t - m_{t-1} = a_1$ independientemente de t, y $\nabla Y_t = Y_t'$ una serie estacionaria, pues Y_t lo era, por lo que $\nabla X_t = \nabla m_t + \nabla Y_t$ es estacionaria. Para eliminar la componente estacional se utiliza un proceso similar al utilizar el operador ∇_S tal que $\nabla_S X_t = X_t - X_{t-S}$, donde S es el periodo de la estacionalidad de la serie.
Hay que tener en cuenta que al diferenciar la serie se está cambiando la estructura de correlación

y perdiendo la información de las componentes de tendencia y estacionalidad.
Esta manera de convertir una serie temporal en serie estacionaria será utilizada más adelante. En concreto, en este trabajo se ha trabajado solamente con diferenciaciones de orden 1, es decir, solo se aplicaba el operador ∇ una vez; debido a que se trataba con 367 series temporales y para la gran mayoría bastaba con diferenciar en primer orden para conseguir una serie estacionaria.

2.2.3. Ventanas móviles

El comportamiento de una serie temporal puede cambiar a lo largo del tiempo. Sin embargo, es posible capturar el comportamiento temporal de la serie a través de ventanas móviles. Una ventana móvil consiste en un subconjunto de instantes de tiempo consecutivos de la serie temporal que se va desplazando en el sentido del eje temporal.

Sea $\mathcal{D} = \{X_t \mid X_t \in \mathbb{R},\, t \in \{1, ..., T\}\}$ la serie temporal; dada una ventana w y un instante de tiempo t, estas ventanas móviles pueden ser centradas o asimétricas. Las ventanas móviles centradas se encuentran con el elemento de la serie correspondiente al instante t como centro de la ventana, lo que implica que el número de elementos dentro de la ventana debe ser un número impar.
Una ventana centrada en t de tamaño $2w+1$ es $\{X_{t-w}, X_{t-w+1}, ..., X_t, ..., X_{t+w-1}, X_{t+w}\}$ con $w < t < T - w$.
Dada una ventana móvil centrada en t y de longitud $2w+1$ para la serie de datos $\mathcal{D}$, el valor promedio del subconjunto de datos que conforman la ventana móvil es lo que se denomina media móvil centrada en t de longitud $2w+1$, $\mu^C(2w+1)_t$, matemáticamente:

$$\mu^C(2w+1)_t = \frac{X_{t-w} + X_{t-w+1} + ... + X_t + ... + X_{t+w-1} + X_{t+w}}{2w+1}. \tag{2.7}$$

Así, el subíndice de la media móvil t coincide con el del valor central X_t. Por ello, no se pueden calcular operaciones correspondientes a las ventanas móviles centradas de los w primeros puntos de la serie ni de los w últimos.

Por otra parte, las ventanas móviles pueden ser asimétricas. Una ventana asimétrica de tamaño w en t es $\{X_{t-w+1}, X_{t-w+2}, ..., X_{t-1}, X_t\}$ con $w \leq t$.
Dada la ventana móvil asimétrica de longitud w de la misma serie $\mathcal{D}$, el valor promedio de dicha ventana se denomina media móvil asimétrica en t de longitud w, $\mu^A(w)_t$, matemáticamente:

$$\mu^A(w)_t = \frac{X_{t-w+1} + X_{t-w+2} + ... + X_{t-1} + X_t}{w}. \tag{2.8}$$

En este caso, el subíndice de la media móvil t coincide con el subíndice del último elemento de la ventana asimétrica X_t, por lo que no se pueden calcular operaciones para los w primeros puntos de la serie.

2.2.4. Procesos estacionarios

Es posible modelar una serie temporal estacionaria a partir de variables adicionales como valores anteriores o medias móviles. Los procesos paramétricos más importantes de series temporales estacionarias pueden clasificarse en [12]:

- **Procesos AR:** los procesos autorregresivos surgen de la idea de un modelo de regresión lineal que permite explicar una variable en un instante t, X_t, en función del valor de esa variable en instantes anteriores $X_{t-1}, X_{t-2}, \ldots$.
  Un ejemplo es el proceso AR(p):

$$X_t = \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + Z_t \ , \tag{2.9}$$

  con $Z_t \sim \text{WN}(0, \sigma^2)$ y todas las raíces del polinomio $1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p = 0$ fuera del círculo unitario (necesario para la condición de estacionaridad).
  La ventaja de estos procesos es la capacidad de almacenar la información de múltiples retardos y establecer una pauta de decrecimiento fija para sus coeficientes. A esta ventaja se le denomina *memoria* larga.

- **Procesos MA:** estos procesos surgen bajo la idea de que es posible explicar una variable en un instante t, X_t, a partir de variables aleatorias $\text{WN}(0, \sigma^2)$ generadas en instantes anteriores.
  Un ejemplo es el proceso MA(q):

$$X_t = \mu + Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \ , \tag{2.10}$$

  con μ la media de la serie temporal, $Z_i \sim \text{WN}(0, \sigma^2),\ i = \{t-q, \ldots, t\}$ y $\theta_1, \ldots, \theta_q$ los parámetros constantes del modelo.
  Los procesos MA permiten que solo unos pocos coeficientes sean distintos de cero, pero con valores arbitrarios, lo que se conoce como *memoria* corta

- **Procesos ARMA** De la unión de los dos procesos anteriores, surgen los procesos ARMA, los más importantes de los procesos de series temporales estacionarios; pues combinan las ventajas de los AR y de los MA, obteniendo la capacidad de representar procesos cuyos primeros coeficientes son cualesquiera (memoria corta) mientras que los siguientes decrecen según unas leyes simples (memoria larga).
  Un ejemplo es el modelo ARMA(p,q):

$$X_t = \mu + Z_t + \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \ , \tag{2.11}$$

  con μ la media de la serie temporal, $Z_t \sim \text{WN}(0, \sigma^2)$ y los polinomios $1 - \phi_1 z - \ldots - \phi_p z^p$ y $1 + \theta_1 z + \ldots \theta_q z^q$ sin raíces comunes.
  Estos procesos permiten métodos simples de predicción, así como modelar muchos tipos de procesos; sin embargo, existen $p + q + 2$ parámetros, lo cual hace complejo el modelado en muchas ocasiones.

Estos modelos son válidos solo para series temporales estacionarias. Sin embargo, muchas series temporales pueden ser transformadas para convertirse en series estacionarias, de manera que pueden añadirse estos mecanismos a los procesos anteriores para cumplir la condición de estacionariedad. Los dos procesos más importantes son los procesos **ARIMA** o modelos autorregresivos integrados de media móvil, en los que se añade el parámetro d que indica el número de veces que se ha de aplicar el operador ∇ antes de modelar el proceso como ARMA; y los procesos **SARIMA** o *seasonal ARIMA* que permiten representar series que tienen un comportamiento estacional, para lo que se añaden los parámetros d, D, P, Q, necesarios para

capturar las componentes de tendencia y de estacionalidad.

Tanto estos dos procesos, como los anteriores, conllevan un estudio intenso de los parámetros a través de tendencias, medias móviles, funciones de autocorrelación y autocorrelación parcial, diferenciación, etc. Esto supone un procedimiento muy costoso la construcción de modelos paramétricos para series temporales. Es por ello que los modelos de aprendizaje automático y aprendizaje profundo, los cuales son capaces de aprender por sí mismos las complicadas relaciones entre las variables y construir los mejores modelos que se ajustan a la serie temporal, han supuesto un punto de inflexión en la modelización de series temporales.

Capítulo 3

Fundamentos

En este capítulo se recogen conceptos necesarios para comprender las sucesivas secciones del trabajo. Primero se detallan las métricas estudiadas en el trabajo, tanto para el apartado de clasificación (la selección de los puntos anómalos) como para los algoritmos que necesiten las funciones de pérdida clásicas de la regresión. Posteriormente, se explica el pre-procesado adoptado para alguno de los algoritmos utilizados. Finalmente, se explican los algoritmos de aprendizaje automático y aprendizaje supervisado utilizados en este proyecto, haciendo especial énfasis en el aprendizaje profundo y en las diferentes arquitecturas revisadas para la detección de anomalías.

3.1. Métricas utilizadas para la detección de anomalías

En este trabajo se han utilizado tanto métricas de clasificación, para evaluar el desempeño de los métodos a la hora de clasificar los puntos como anómalos o no anómalos, como métricas de regresión, necesarias para algoritmos como las redes neuronales.

3.1.1. Métricas de clasificación binaria

Independientemente de los formalismos y subprocesos elegidos para llevar a cabo la detección de anomalía, el paso final consiste en clasificar un punto como punto anómalo o normal.
Esto no es más que un problema de clasificación binaria debido a que un elemento del conjunto de datos puede pertenecer a la clase *Anomalía* (clase positiva) o *no-anómalo* o normal (clase negativa).

La matriz de confusión es una herramienta que permite visualizar y resumir el desempeño de un algoritmo de clasificación. Mediante esta simple matriz se comparan los valores reales y las predicciones realizadas por el modelo de clasificación según si estas son predicciones verdaderas o falsas. La matriz de confusión clásica para clasificación binaria se encuentra representada en la Figura 3.1.

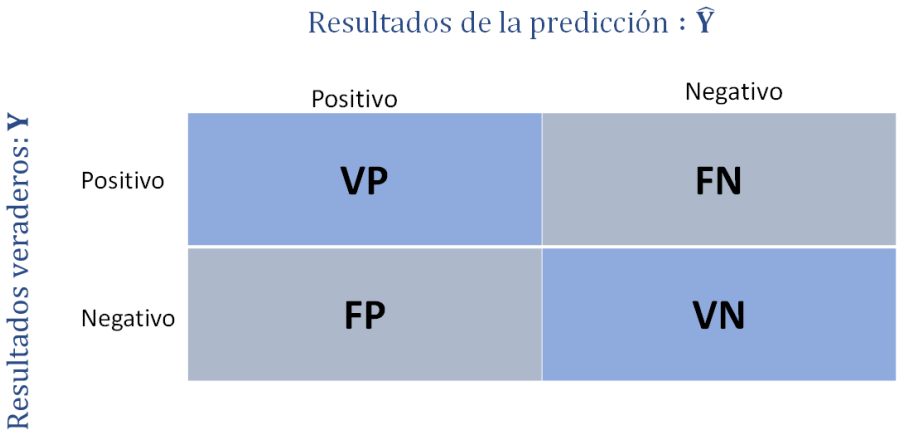


Figura 3.1: Matriz de confusión clásica para un problema de clasificación binaria. Fuente: propia.

Donde:

- **Verdaderos Positivos (VP):** número de elementos cuya clase real es Positiva y la clase predicha fue Positiva.
- **Verdaderos Negativos (VN):** número de elementos cuya clase real es Negativa y la clase predicha fue Negativa.
- **Falsos Positivos (FP):** número de elementos cuya clase real es Negativa, pero la clase predicha fue Positiva. Este error de clasificación se denomina error de tipo I.
- **Falsos Negativos (FN):** número de elementos cuya clase real es Positiva, pero la clase predicha fue Negativa. Este error de clasificación se denomina error de tipo II.

El escenario ideal es aquel en el que el número de falsos positivos y falsos negativos es 0 y, por tanto, solamente los valores de la diagonal principal son no nulos.
Sin embargo, en los problemas de detección de anomalías existe un inconveniente crítico: las clases están generalmente desbalanceadas. En los problemas de clasificación, se dice que las clases son desbalanceadas o no-balanceadas cuando la proporción de individuos en cada clase no es similar. Esta situación, unida a los diferentes contextos e importancias que puedan representar las clases, supone que no exista una métrica única ideal, sino que dependiendo del problema y del objetivo será más acertada una u otra. Las métricas más populares son:

- **Precisión:** Conocida generalmente como *accuracy*, esta métrica indica el número de elementos clasificados correctamente frente al número total de elementos:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} . \tag{3.1}$$

  Este valor se encuentra comprendido entre 0 (precisión nula) y 1 (precisión absoluta)

- **Sensibilidad o Tasa de Verdaderos Positivos:** Conocida en inglés como *Sensibility* o *Recall*, indica la proporción de casos positivos correctamente detectados

$$\text{Sensibilidad} = \frac{VP}{VP + FN} . \tag{3.2}$$

En este trabajo, responde a la pregunta: ¿Qué proporción de casos verdaderamente anómalos fueron identificados por el modelo?

- **Exhaustividad:** Conocida generalmente como *precision*, muestra el número de casos positivos identificados correctamente en comparación con todos los casos positivos predichos.

$$\text{Exhaustividad} = \frac{VP}{VP + FP} \ . \tag{3.3}$$

En el caso del trabajo, responde a la pregunta: ¿Qué proporción de casos anómalos predichos por el modelo son realmente anómalos?

- **Especificidad o Tasa de Verdaderos Negativos:** Conocida en inglés como *Especificity*, muestra la proporción de casos negativos correctamente identificados por el modelo.

$$\text{Especificidad} = \frac{VN}{VN + FP} \ . \tag{3.4}$$

En este trabajo, responde a la pregunta: ¿Qué proporción de puntos verdaderamente no-anómalos fue identificado por el modelo?

- **Puntuación F1:** Representa el desempeño general del modelo al combinar exhaustividad y sensibilidad

$$F1 = 2 \cdot \frac{\text{Sensibilidad } \cdot \text{Exhaustividad}}{\text{Sensibilidad } + \text{Exhaustividad}} \ . \tag{3.5}$$

Simon B et al [14] usaron esta métrica para medir la calidad de su clasificador de anomalías en imágenes de rayos X de electrodos de pilas de combustible.

- **Precision balanceada (*Balanced Accuracy*)**

$$\textbf{Precisión Balanceada} = \frac{\text{Sensibilidad} + Especificidad}{2} \ . \tag{3.6}$$

Esta métrica resulta útil para problemas de clasificación binaria, especialmente en conjuntos de datos cuyas clases están muy desbalanceadas. La métrica de precisión balanceada se sobrepone a este problema al ponderar las clases positivas y negativas para obtener un resultado equilibrado, a pesar de que una clase sea más numerosa que la otra.

- **Área bajo la curva ROC**

Esta curva permite visualizar el desempeño del clasificador para distintos umbrales de corte, la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos para dichos umbrales.
La curva ROC es una curva de probabilidad. Su métrica asociada, AUC, es el área bajo dicha curva y representa el grado de separabilidad del clasificador.
Para poder dibujar la curva ROC usamos distintos valores σ como umbrales, lo que da lugar a diferentes duplas de $(1 - \text{Especificidad}, \text{Sensibilidad})$ para cada σ. Estos valores permiten trazar una curva que comienza en el origen y termina en el punto $(1, 1)$. La métrica correspondiente, AUC, es el área debajo de dicha curva. Si este AUC es cercano a 0.5, el modelo no tiene poder de discriminación entre las clases; mientras que si es cercano a 1, el modelo tiene una gran capacidad de separar las clases.

En la Figura 3.2 se pueden observar tres curvas ROC distintas. La curva A tiene un poder de discriminación mayor que la curva B, es decir, hay una mayor proporción de verdaderos positivos que de falsos positivos para cada punto de corte. La curva C es la línea de no discriminación, ya que para cualquier punto de corte, el modelo arroja igual proporción de verdaderos positivos que de falsos positivos (AUC = 0.5)

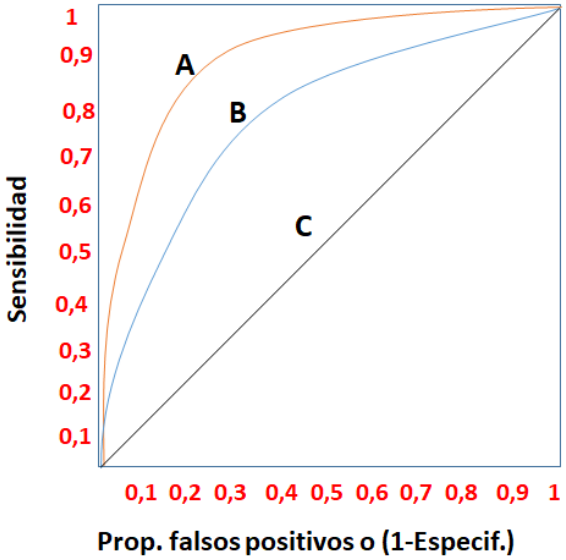


Figura 3.2: Comparación 3 curvas ROC Distintas. Fuente: [15]

Es importante elegir la métrica adecuada. En este caso no tiene sentido hablar de métricas como precisión, pues las clases están tremendamente desbalanceadas. Supongamos que el porcentaje de anomalías es del 1 %. Si se escogiese que todos los puntos son no-anómalos se obtendría una precisión de 0.99. Por ello, en este trabajo se han estudiado como métricas la precisión balanceada, la puntuación *F1* y el área bajo la curva ROC o AUC. Finalmente, se decidió utilizar como métrica de referencia la puntuación *F1*, pues esta se centra en el desempeño del algoritmo en torno a la clasificación de la clase positiva, en este caso la clase anómala. De esta manera se solventa el problema del desbalance de clases. Además, se comparó el área bajo la curva ROC entre los clasificadores, pues esta métrica es independiente de umbrales y solo tienen en cuenta las probabilidades o puntuaciones estimadas por el modelo para cada punto. Sin embargo, no ha sido la métrica elegida, puesto que en casos de clases tan desbalanceadas como el caso de estudio. En [16] se estudia este problema donde se concluye que en problemas donde el desbalance sea muy acusado se debe trabajar con métricas que relacionen la sensibilidad y la exhaustividad.

3.1.2. Métricas de regresión

A pesar de que el último paso en la detección de anomalías es la clasificación de los puntos según sean anómalos o no; para tomar esta decisión es necesario de pasos previos y algoritmos capaces de asociar a cada punto un valor numérico que determine cómo de probable es que ese punto sea una anomalía. Algunos de estos algoritmos se basan en la reconstrucción y predicción de la serie, es decir, corresponden con algoritmos de regresión que buscan predecir

valores continuos. Para medir el desempeño de estos algoritmos de regresión es necesario utilizar métricas propias de este tipo de problemas. Estas métricas se utilizan a lo largo del trabajo, pues corresponden con las métricas para los modelos de regresión, así como las funciones de pérdida necesarias para el aprendizaje de algunos modelos de aprendizaje automático y aprendizaje profundo.

- **Error Cuadrático Medio (*Mean Squared Error-MSE*)** Es una función de pérdida muy importante para el ajuste de algoritmos de regresión utilizando el marco teórico de *mínimos cuadrados* con la intención de minimizar el valor medio de la diferencia al cuadrado entre observaciones y predicciones.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \ . \tag{3.7}$$

- **Error Absoluto Medio (*Mean Absolute Error-MAE*)** Esta función de pérdida representa el promedio de las diferencias en valor absoluto entre observaciones y predicciones

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\|y_i - \hat{y}_i\| \ . \tag{3.8}$$

  Esta función de perdida es lineal, al contrario que (3.7), que promediaba las diferencias al cuadrado. Esto significa que pondera todas las diferencias por igual sin verse tan afectada por los valores extremos como el error cuadrático medio.

3.2. Preprocesamiento de los datos

Dependiendo de los algoritmos a utilizar, las series temporales deben ser pre-procesadas para un correcto funcionamiento del algoritmo. Este preprocesamiento puede conllevar tanto una transformación numérica del rango de valores de la serie, reduciendo el tiempo necesario para la convergencia de algunos algoritmos, así como convertir los datos a una escala de valores más fácil de interpretar para el algoritmo; como una transformación de la estructura de la serie de datos, aumentando el número de variables disponibles para el algoritmo al tomar una decisión.

3.2.1. Transformación numérica

Algunos de los métodos estudiados en este trabajo requieren de un preprocesamiento de los datos para reducir el tiempo computacional y evitar problemas de convergencia en algoritmos de aprendizaje automático y aprendizaje profundo. Existen diferentes formas de pre-procesar los datos, las más conocidas son:

- **Estandarización:** Un conjunto de datos está estandarizado si su media es nula y su varianza es la unidad. De esta manera sea $\mathcal{D}$ el conjunto de datos, y sean μ, σ la media y desviación estándar de $\mathcal{D}$, la serie estandarizada es:

$$x_i' = \frac{x_i - \mu}{\sigma} \ , \ \forall x_i \in \mathcal{D} \ , \tag{3.9}$$

  siendo una de las formas más populares de procesar los datos.

- **Normalización mediante el rango:** Para convertir todas las variables de la serie de datos al mismo rango de valores, es posible normalizar los datos de manera que estos se encuentren comprendidos entre 0 y 1:

$$x'_i = \frac{x_i - x_{\text{mín}}}{x_{\text{máx}} - x_{\text{mín}}} \ , \ \forall x_i \in \mathcal{D} \ . \tag{3.10}$$

  En este trabajo se ha optado por la normalización de los datos para los modelos de aprendizaje automático y aprendizaje profundo, igual que se hizo en [17] donde se utilizaron algoritmos de aprendizaje profundo para la detección de anomalías aplicadas en la gestión de la cadena de suministro.

3.2.2. Transformación estructural. Uso de valores pasados como variables adicionales

El tema de trabajo trata acerca de métodos de detección de anomalía en series temporales. Muchos de los métodos y algoritmos utilizados en el trabajo requieren del uso de los valores anteriores como variables adicionales para que dichos algoritmos sean capaces de tratar con las dependencias temporales entre los datos. Para poder 'alimentar' estos algoritmos con los valores anteriores como variables, se hace uso de las ventanas móviles explicadas en 2.2.3. En concreto, las ventanas móviles asimétricas desplazándose un instante de tiempo hacia adelante. Sea $\mathcal{D}$ la serie temporal unidimensional de longitud T: $\mathcal{D} = \{X_t \mid X_t \in \mathbb{R},\ t \in \{1, \ldots, T\}\}$, a través del uso de ventanas móviles asimétricas de tamaño w con $0 < w \leq T$, se convierte la serie a una serie de longitud $T - w + 1$ con w variables $\mathcal{W} = \{\mathbf{W}_t \mid \mathbf{W}_t \in \mathbb{R}^w,\ t \in \{w, \ldots, T\}\}$ con $\mathbf{W}_t = (X_{t-w+1}, \ldots, X_t)$. No todos los métodos van a requerir los mismos pasos, por ejemplo, los métodos estadísticos no se aplican a la serie normalizada ni usando los valores anteriores como variables adicionales.

3.3. Algoritmos Aprendizaje Automático

Hoy en día, las técnicas de aprendizaje automático están presentes en cualquier tarea relacionada con el tratamiento y modelización de datos. Esto se debe a la capacidad que tienen de adaptarse a grandes cantidades de datos y dimensiones de estos, especialmente para tratar con series con una gran cantidad de datos. Debido a la importancia que tiene en algunos sectores la detección de anomalías, se han utilizado algoritmos e incluso desarrollado otros únicamente para esta tarea. Los algoritmos utilizados en este trabajo son:

3.3.1. Árboles de Aislamiento (*Isolation Forest*)

Uno de los algoritmos más famosos de aprendizaje automático especializado en la detección de anomalías es el algoritmo de árboles de aislamiento o *Isolation Forest* [18]. Este algoritmo es comúnmente utilizado de manera semi-supervisada: se entrena el modelo sobre un subconjunto de entrenamiento en el que todos los datos son datos no-anómalos, y se aplica el modelo sobre un subconjunto en el que se desconoce cuáles son las muestra anómalas, pero si se sabe que son la clase minoritaria. Es lo que se conoce como *one-class classification*. Sin embargo, puede

ser utilizada como algoritmo no supervisado, obteniendo resultados similares debido al gran desequilibrio entre clases en las que la clase anómala representa un porcentaje ínfimo.
Para entender algunos de los conceptos explicados, en el apéndice D se explican los árboles de regresión y clasificación de manera breve. Los árboles de clasificación son similares, pero en vez de predecir una variable cuantitativa se predice una variable cualitativa que representa la clase predicha para la observación.

El funcionamiento es el siguiente: el algoritmo construye un conjunto de árboles de decisión binarios denominados *iTrees* que aíslan puntos. Como las anomalías son más propensas a ser aisladas que los puntos no anómalos, es mucho más probable que estas se encuentren más cerca de la raíz de un árbol. A este conjunto de árboles binarios se le denomina *iForest*.
Supóngase el conjunto de observaciones d-dimesionales $\mathcal{D} = \{x_t \mid x_t \in \mathbb{R}^d,\ t \in \{1, ..., T\},\ d \geq 1\}$.
La detección de anomalías utilizando los árboles de aislamiento consiste en un proceso en dos etapas:

- En la primera etapa, *etapa de entrenamiento*, se utiliza un subconjunto de datos de tamaño m, $D' = \{x'_1, ..., x'_m\}$, escogido aleatoriamente sin reemplazamiento del conjunto original D ($D' \subset D$). El algoritmo divide de forma recursiva el subconjunto D'. Para dividir este subconjunto, este se va segmentando en subconjuntos más pequeños, seleccionando aleatoriamente una variable y un valor aleatorio entre el máximo y el mínimo valor de la variable seleccionada para cada subconjunto. Esta partición recursiva puede representarse como una estructura de árbol en la que un nodo se divide en dos según si el atributo seleccionado es mayor que el valor de corte o no. El proceso termina cuando todos los puntos del nodo tienen el mismo valor para el atributo seleccionado o solamente queda un punto en el nodo. Finalmente, se devuelve el conjunto de árboles construido. En el caso de series temporales unidimensionales, debido a que solo existe una variable de estudio, el subconjunto D' se divide de forma recursiva al ir eligiendo distintos valores de corte para esa variable.

- En la segunda etapa, *etapa de evaluación*, se obtiene la puntuación de anomalía o *anomaly score* de cada una de las muestras del conjunto restante. Esta puntuación se obtiene de la profundidad promedio, $E[h(x)]$, donde $h(x)$ representa la profundidad del punto x (número de divisiones necesarias para aislar el punto x). De esta manera $E[h(x)]$ representa el promedio de cada una de las profundidades $h(x)$ obtenidas para cada *iTree* del conjunto de árboles. Finalmente, el *anomaly score* se define como

$$s(x, m) = 2^{\frac{E[h(x)]}{c(m)}}, \tag{3.11}$$

  siendo $c(m)$ la media de $h(x)$ dado m, necesario para normalizar el valor de $h(x)$. De esta manera se obtiene un valor numérico válido para puntuar cómo de posible es que un punto sea una anomalía.

En la Figura 3.3 se representa un esquema de este tipo de árboles de aislamiento:

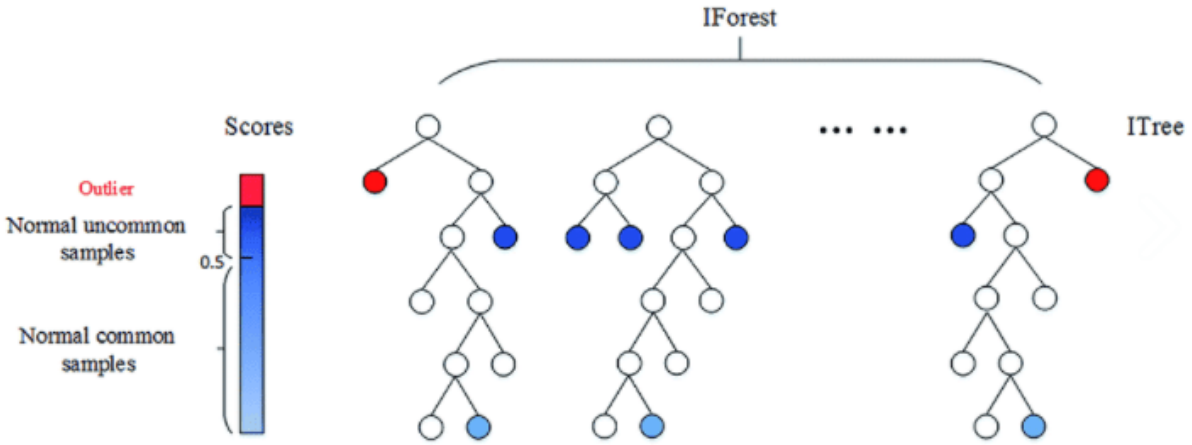


Figura 3.3: Esquema algoritmo *Isolation Forest*. Fuente: [19]

3.3.2. *One Class Support Vector Machine* (OC-SVM)

Este algoritmo consiste en una variación de los clasificadores máquinas de vectores soporte o *Support Vector Machines* (SVM) [20]. Un clasificador SVM es capaz de construir un hiperplano o conjunto de hiperplanos en un espacio transformado. Este hiperplano separa los puntos según las clases a las que pertenecen. La popularidad de estos clasificadores reside en que si los datos no son separables linealmente en el espacio original, los clasificadores SVM utilizan lo que se denomina el truco del kernel para crear hiperplanos capaces de separar linealmente los datos en un espacio transformado. Para ello utilizan una función no lineal Φ que proyecta los datos del espacio original a un espacio transformado F. Después, dicho hiperplano es proyectado de vuelta al espacio original, donde tendrá la forma de una curva no lineal. Para entender los clasificadores SVM, se explican los conceptos básicos de una forma similar a la realizada en [21]:

Sea el conjunto de observaciones d-dimensionales $\mathcal{D} = \{\mathbf{x_t} \mid \mathbf{x_t} \in \mathbb{R}^d,\ t \in \{1, .., n\},\ d \geq 1\}$ e $y_t \in \{-1, 1\}$ la clase a la que pertenece la t-ésima observación. El hiperplano se representa con la ecuación $\mathbf{w}^T\mathbf{x} + b = 0$, con $\mathbf{w} \in F$ el vector normal y $b \in \mathbb{R}$ el término de sesgo o *bias*. Este hiperplano determina el margen entre las dos clases: todos los puntos de datos para la clase -1 están en un lado y todos los puntos de datos para la clase 1 en el otro. El margen es la distancia perpendicular entre el hiperplano separador y el hiperplano que pasa sobre los puntos más cercanos, denominados vectores soporte. La distancia desde el punto más cercano de cada clase al hiperplano es igual; así, el hiperplano construido busca el margen máximo.
La función $K(\mathbf{x}, \mathbf{x_i}) = \Phi(\mathbf{x})^T\Phi(\mathbf{x_i})$ se conoce como la función kernel. Esta función se utiliza en lo que se conoce como el truco del kernel y es lo que le da a las SVM un poder de separación tan grande al proyectar los datos a un espacio transformado en el que si es posible encontrar un hiperplano de separación. El espacio de características F puede tener una dimensión ilimitada y, por lo tanto, el hiperplano que separa los datos puede ser muy complejo.
En la Figura 3.4 puede observarse un ejemplo de hiperplano en el espacio transformado para un clasificador SVM.

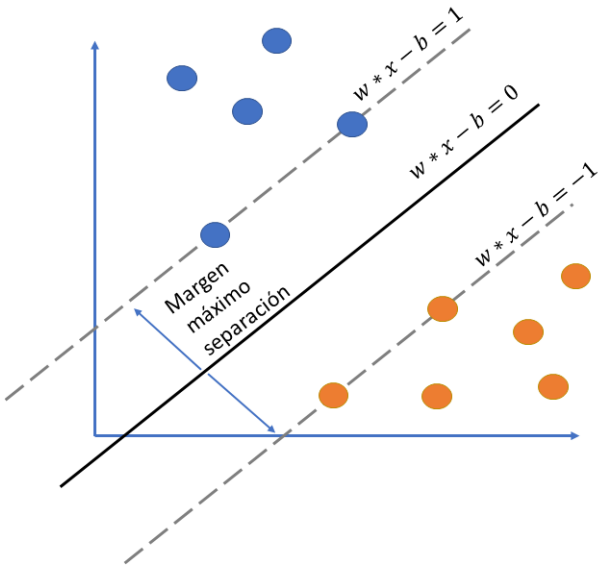


Figura 3.4: Representación hiperplano de separación en el espacio transformado F. Las líneas discontinuas marcan el margen máximo de separación, siendo los puntos que se encuentran sobre dichas líneas discontinuas los vectores soporte. Fuente: propia.

Para evitar que el clasificador sufra sobreajuste, se introduce el vector de variables de holgura $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$. Estas variables facilitan crear un margen suave permitiendo que algunos puntos se encuentren dentro de dicho margen. La constante $C > 0$, parámetro de regularización, determina el compromiso entre maximizar el margen y el número de puntos que se encuentran dentro de dicho margen, permitiendo algunos errores en la clasificación a la vez que penalizándolos.

El algoritmo *One Class SVM* desarrollado por Schölkopf y Williamson [22] separa todos los puntos de datos del origen (en el espacio transformado F) y maximiza la distancia desde este hiperplano hasta el origen. Esto da como resultado una función binaria que captura regiones en el espacio de datos original donde reside la densidad de probabilidad de los datos. Por lo tanto, la función devuelve $+1$ en una región 'pequeña' en la que se encuentran la mayoría de puntos, y -1 en cualquier otro lugar.

La función objetivo de dicho algoritmo es:

$$\min_{\mathbf{w},\xi,\rho} \frac{\|\mathbf{w}\|^2}{2} + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho\ , \tag{3.12}$$

sujeto a

$$\langle \mathbf{w}, \Phi(\mathbf{x_i})\rangle \geq \rho - \xi_i, \quad \forall i = 1, ..., n\ , \tag{3.13}$$

$$\epsilon_i \geq 0, \quad \forall i = 1, ..., n\ , \tag{3.14}$$

con ρ el *offset* del hiperplano en el espacio transformado.
En este caso, la variable C es sustituida por $\nu \in (0, 1]$, de manera que $\nu \propto \frac{1}{C}$ y sirve como cota superior para la fracción de puntos anómalos, así como cota inferior para el número de puntos

utilizados como vectores soporte.
Usando multiplicadores de Lagrange y una función kernel para resolver, la función de decisión de un clasificador SVM para $\mathbf{x} = \{x_1, .., x_d\}$ adopta la forma:

$$f(x) = sgn\left(\langle \mathbf{w}, \Phi(\mathbf{x_i})\rangle - \rho\right) = sgn\left(\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x_i}) - \rho\right). \tag{3.15}$$

con $\alpha_i,\ \alpha_i > 0\ \forall\ i = 1, ..., n$ los multiplicadores de Lagrange.

Este algoritmo crea un hiperplano caracterizado por $\mathbf{w}$ y ρ, cuya distancia al origen es máxima para el espacio de características F y el cual separa todos los puntos del origen, separando los valores más *Anómalos* del resto.

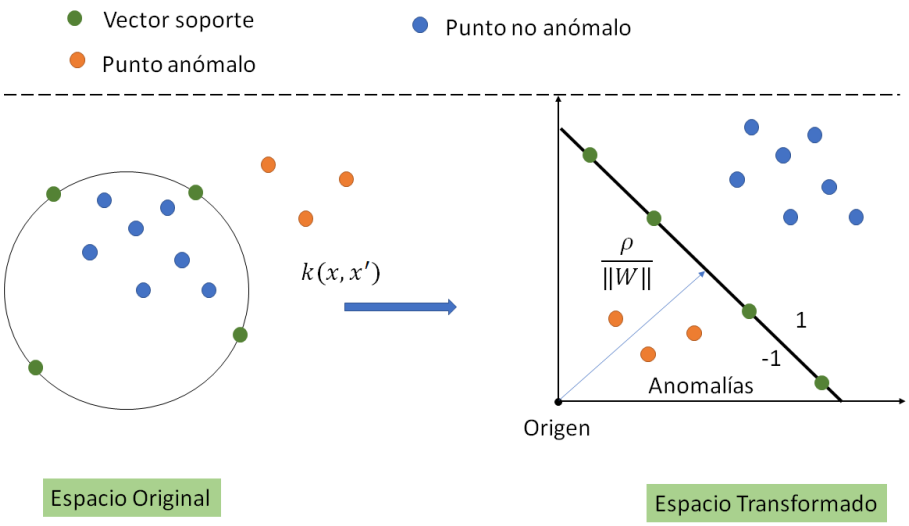


Figura 3.5: Hiperplano en el espacio original y en el espacio transformado. Fuente: propia.

3.3.3. Factor Anómalo Local (LOF)

Este algoritmo publicado por Breuning et al. [23] conocido como *Local Outlier Factor*-(LOF) consiste en una técnica de detección de anomalías basada en densidades locales. Como bien se indica en el artículo, esta puntuación es local, en tanto que depende de cuánto de aislado se encuentra el punto con respecto a sus vecinos más cercanos. De esta manera, la distancia del punto a sus vecinos sirve para estimar la densidad local. Comparando esta con las de sus vecinos, es posible identificar como valores anómalos aquellos con una densidad sustancialmente menor que sus vecinos. Esta distancia local se estima como la distancia en la que el punto puede ser 'alcanzado' por sus vecinos.

Sea el conjunto de observaciones d-dimensionales de longitud T, $\mathcal{D} = \{x_t \mid x_t \in \mathbb{R}^d, t \in \{1, \ldots, T\},\ d \geq 1\}$, y x una observación del conjunto $\mathcal{D}$, $x \in \mathcal{D}$; para calcular el *Factor Anómalo Local* de x primero se deben definir los siguientes conceptos:

Definición 3.3.1 (*k-distancia* de un punto x)**.** La distancia entre dos puntos x y x' del conjunto $\mathcal{D}$ se denomina $d(x, x')$ donde d es la métrica o función de distancia elegida; por ejemplo la

distancia Euclídea:

$$d(x, x') = \sqrt{\sum_{i=1}^{T}(x_i - x'_i)^2} \ . \tag{3.16}$$

Para el punto x, la $\mathbf{k - distancia}$ es la distancia $d(x, x')$ entre x y su vecino más lejano $x' \in \mathcal{D}$ en las siguientes condiciones:

1. Al menos, k puntos $x'' \in \mathcal{D} \setminus \{x\}$ cumplen $d(x, x'') \leq d(x, x')$;
2. Como mucho, $K - 1$ puntos $x'' \in \mathcal{D} \setminus \{x\}$ cumplen $d(x, x'') < d(x, x')$.

Definición 3.3.2 (*k-vecinos* de x)**.** El significado de los *k-vecinos* de x es cualquier punto x' cuya distancia al punto x, $d(x, x')$, es menor o igual que la $k - distancia(x)$, así se define $N_{k-distancia(x)}(x)$ como

$$N_{k-distancia(x)}(x) = \{x' \mid x' \in \mathcal{D}, d(x, x') < \delta_k\} \ . \tag{3.17}$$

Definición 3.3.3 (Distancia de alcance de x con respecto a x')**.** Sea $k \in \mathbb{N}_+$ la distancia de alcance de un punto x con respecto a x' se define como:

$$RD_k(x, x') = \text{máx}(k - distancia(x'), d(x, x')\} \ . \tag{3.18}$$

Esta distancia de alcance de x con respecto de x' es, por tanto, la distancia entre x y x' a no ser que esta sea menor que la *k-distancia* de x'. En este caso, todos los puntos que pertenecen a los *k-vecinos* de x' se consideran igualmente distantes.

Con estos conceptos ya conocidos, se llega a las dos expresiones más importantes:

Definición 3.3.4 (densidad de accesibilidad local)**.**

$$LRD_k(x) = \left(\frac{\sum_{x' \in N_{k-distancia(x)}} RD_k(x, x')}{\mid N_{k-distancia}(x) \mid} \right)^{-1} \ . \tag{3.19}$$

Esto no es más que el inverso de la distancia de accesibilidad promedio de x con respecto a sus vecinos, es decir, la distancia a la que el punto x puede ser alcanzado desde sus vecinos.

Finalmente, se obtiene la puntuación de anomalía conocida como *factor anómalo local* al comparar las densidades de accesibilidad locales

Definición 3.3.5 (Factor anómalo local, LOF)**.**

$$LOF_k(x) = \frac{\sum_{x' \in N_{k-distancia(x)}} \frac{LRD_k(x')}{LRD_k(x)}}{\mid N_{k-distancia}(x) \mid} \ , \tag{3.20}$$

que no es más que la densidad de accesibilidad local promedio de los vecinos divida por la densidad de accesibilidad local del punto x

Un valor LOF cercano a 1 indica que el punto tienen una densidad similar a sus vecinos, y, por lo tanto, no puede considerarse una anomalía. En cambio, un punto con valor LOF mayor que 1 se considera una anomalía, pues su densidad es menor que la de sus vecinos.

3.3.4. *Extreme Gradient Boosting* (XGBoost)

Este método utiliza un ensamblaje de regresores de tipo árbol. En el apéndice D se explica más en detalle en que consisten los árboles de regresión.
En aprendizaje automático se denomina ensamblaje de modelos a un modelo en el que se combinan las predicciones de varios modelos para realizar una predicción final. La palabra *Boosting* hace referencia a la forma en la que se construyen el modelo: se combinan regresores 'débiles' para constituir un regresor 'fuerte' o *robusto*. Los regresores débiles se añaden de forma iterativa, teniendo asociados unos pesos que están relacionados con la exactitud de sus predicciones. Además, cada vez que se añade un nuevo regresor se cambia la estructura de los pesos y se da una mayor importancia a aquellos puntos en los que más fallaron los anteriores regresores débiles. Así se consigue que los nuevos regresores se centren más en estos datos. Finalmente, la predicción del ensamblaje consiste en una suma ponderada de las predicciones de los regresores 'débiles'.
En el artículo original [24] se explica en qué consiste el algoritmo *XGboost*:
dado un conjunto de datos de longitud n, con d variables $\mathcal{D} = \{(\mathbf{x}_i \mid \mathbf{x_i} \in \mathbb{R}^d,\ i \in \{1, \dots, n\},\ d \geq 1\}$ y n variables y_i a predecir, un ensamblaje de regresores de tipo árbol utiliza K funciones predictivas (K regresores) para realizar una predicción $\hat{y}_i$:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}, \tag{3.21}$$

donde $\mathcal{F}$ representa el espacio de los árboles de regresión. Su función de pérdida a minimizar durante el entrenamiento es:

$$\begin{aligned} \mathcal{L}(\phi) &= \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \\ \text{donde } \Omega(f) &= \gamma T + \frac{1}{2}\lambda \|\mathbf{w}\|^2, \end{aligned} \tag{3.22}$$

con l una función de pérdida (convexa y diferenciable) que cuantifica la diferencia entre la predicción $\hat{y}_i$ y el valor real y_i; y Ω el término que penaliza la complejidad del regresor para evitar el sobreajuste del modelo. Cada $f_k \in \mathcal{F}$ corresponde con un árbol de regresión distinto con T hojas y $\mathbf{w}$ el vector de pesos para sus hojas.
La ecuación de la función de pérdida del ensamblaje incluye términos que no pueden ser optimizados haciendo uso de métodos clásicos en el espacio euclídeo. Para solventar este problema, el modelo se entrena de manera iterativa. Sea $\hat{y}_i^{(t)}$ la predicción para el i-ésimo elemento de $\mathcal{D}$ en la t-ésima iteración, será necesario añadir f_t para poder minimizar la siguiente función de pérdida:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t), \tag{3.23}$$

donde f_t es la función que más mejora el modelo de acuerdo a la ecuación (3.23). Para poder resolver esta ecuación se utiliza una aproximación de Taylor de segundo orden:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t), \tag{3.24}$$

con g_i la derivada de la función de pérdida $\partial_{\hat{y}_i(t-1)} l(y_i, \hat{y}_i^{(t-1)})$ y h_i la segunda derivada $\partial^2_{\hat{y}_i(t-1)} l(y_i, \hat{y}_i^{(t-1)})$, obteniendo de esta manera una función de pérdida a minimizar en cada

iteración.

Existen varios artículos donde se utiliza la arquitectura basada en ensamblajes de modelos de tipo árbol del XGBoost para poder utilizar modelos de regresión y predecir el valor de series temporales. En [25] se compara el uso del algoritmo de XGBoost para la predicción de series temporales relacionadas con el consumo eléctrico con otros modelos clásicos como el modelo *ARIMA* o regresores de tipo *Random Forest* demostrando como el algoritmo XGBoost era superior en numerosos escenarios.

3.4. Aprendizaje Profundo

Al igual que los algoritmos de aprendizaje automático, los algoritmos de aprendizaje profundo son capaces de lidiar con complicados problemas y grandes volúmenes de datos, aprendiendo y adaptándose a cualquier tarea.
Estos algoritmos han supuesto un antes y un después en el campo de la inteligencia artificial, convirtiéndose en los métodos más populares para campos como el procesamiento del lenguaje, visión por ordenador e incluso tareas sencillas de clasificación y regresión normalmente abordadas por técnicas de aprendizaje supervisado. La detección de anomalías en series temporales también se nutre de estos algoritmos, pues permiten modelar relaciones complejas y no lineales entre los datos, suponiendo una gran ventaja para poder comprender las complicadas dependencias temporales de los datos.

3.4.1. Redes Neuronales Artificiales

El aprendizaje profundo surge gracias a las redes neuronales artificiales, algoritmos matemáticos inspirados en las neuronas cerebrales y sus asociaciones en redes, formadas por un conjunto de capas interconectadas entre ellas y que siguen una arquitectura específica.

Para entender los diferentes tipos de arquitecturas utilizados es necesario comprender primero el concepto de perceptrón multicapa, el modelo más famoso de esta rama de la inteligencia artificial. Debido a su arquitectura en la que la información 'fluye' hacia delante, estos modelos se denominan redes *feedforward*, pues la información fluye en un solo sentido (y no existe retroalimentación). Su estructura es muy sencilla: cuenta con tres tipos de capas que permiten mapear una entrada a una salida.

- **Capa de Entrada:** Formada por las neuronas de entrada. Cada neurona de entrada se corresponde con un elemento del vector de datos de entrada $(x_1, x_2, ..., x_w)$.
- **Capa de salida:** Formada por las neuronas de salida que constituyen el vector o elemento de salida final.
- **Capas Ocultas:** Estas capas intermedias están formadas por neuronas en las que la información que entra proviene de capas anteriores y la que sale avanza a capas posteriores. El vector asociado a cada una de estas capas se conoce como vector o *estado* oculto $\mathbf{h}$.

Es precisamente gracias a estas capas ocultas por lo que se denomina *Aprendizaje Profundo*, pues estos algoritmos cuentan con la capacidad de trabajar con muchas de estas capas ocultas, en las que se realizan la mayor parte de las operaciones necesarias para mapear las entradas o *inputs* a las salidas u *outputs*.

3.4.1.1. Conexiones entre neuronas

Cada una de las neuronas de las capas ocultas y de las capas de salida contienen un peso y un *bias*, necesarios para el aprendizaje y la conexión entre las distintas capas. En la Figura 3.6 puede observarse un ejemplo de perceptrón multicapa que mapea un vector de entrada $\mathbf{x} = (x_1, x_2)$ en una salida $y \in \mathbb{R}$, con una capa de entrada con dos neuronas, una capa oculta con tres neuronas y una capa de salida con una neurona

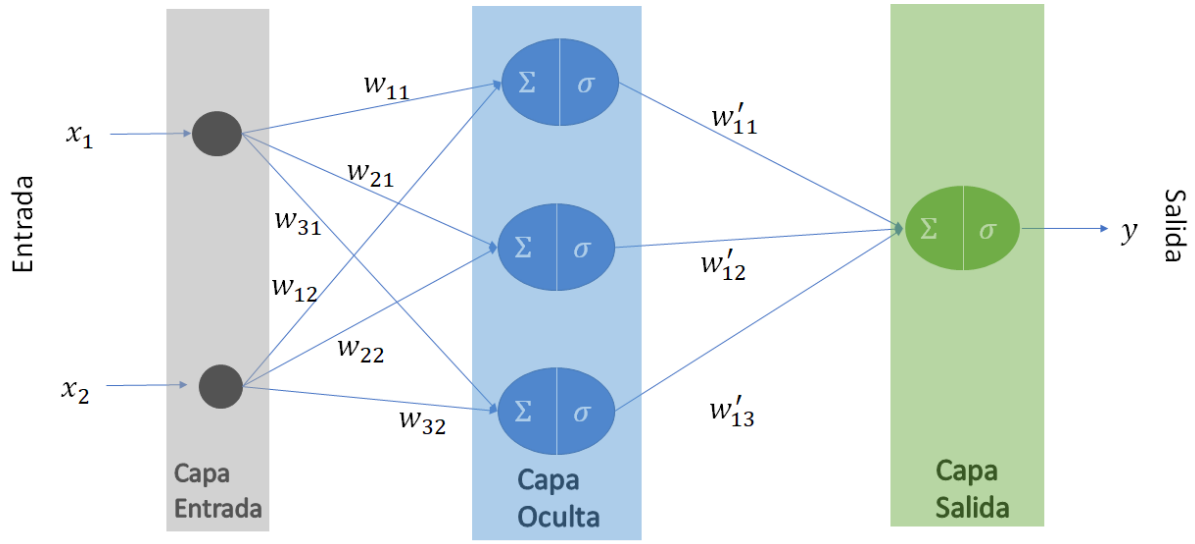


Figura 3.6: Perceptrón multicapa con una capa oculta con 3 neuronas. Fuente: propia.

Es posible recoger las transformaciones que se dan en cada una de las neuronas artificiales de las capas ocultas y de la capa de salida a través de la siguiente ecuación:

$$y_i = \sigma\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right), \tag{3.25}$$

siendo n el número de entradas que recibe la neurona i-ésima, w_{ij} los pesos asociados a la entrada j-ésima x_j, b_i el elemento i-ésimo del vector de *bias* o sesgo, σ la función de activación escogida e y_i la salida de dicha neurona. Por ejemplo, para la neurona de salida, la salida $y_i = y$ y las entradas x_j son las salidas de las neuronas de la capa oculta h_1, h_2, h_3 con sus pesos asociados $W'_{11}, W'_{12}, W'_{13}$; mientras que para la primera neurona de la capa de entrada, la salida es h_1, la entrada es x_1, x_2 y los pesos asociados a dicha entrada son W_{11}, W_{12}.

La ecuación (3.25) muestra las interacciones que se dan en la neurona: la neurona recibe unas entradas junto a unos pesos, que asignan la importancia que tienen estas entradas en la

respectiva neurona. Tras ello se 'activa' la neurona a través la suma de las entradas ponderadas por los pesos, más un valor de *bias* o sesgo. Finalmente, se aplica la función de activación escogida sobre la suma anterior para obtener la salida de dicha neurona.

Estas funciones de activación son claves en el aprendizaje profundo, pues permiten establecer relaciones *no lineales* entre los datos, transformando la salida de la celda previa antes de 'entrar' en la siguiente celda. Su capacidad de utilizar funciones no lineales permite a la red comprender los complejos patrones en la serie de datos. Los pesos representan la contribución de cada neurona de la capa anterior a las neuronas de la capa siguiente, es decir, qué influencia tiene cada una de las entradas de una neurona en la salida de esta; mientras que los vectores *bias* (sesgo) garantizan que aunque la entrada de una neurona sea cero, su salida no lo será.

De esta manera, en la Figura 3.6, la transformación que se aplica al vector de entrada $\mathbf{x}$ para obtener el vector oculto $\mathbf{h}$ es $\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$, mientras que la transformación que sufre el vector oculto $\mathbf{h}$ para obtener la salida y es $\sigma'(\mathbf{W}'\mathbf{h} + \mathbf{b}')$ con $\mathbf{W}, \mathbf{W}'$ las matrices de pesos, $\mathbf{b}, \mathbf{b}'$ los vectores de *bias* y σ, σ' las funciones de activación.

3.4.1.2. Método de aprendizaje

En el aprendizaje profundo, las redes neuronales aprenden de manera automática cómo resolver el problema planteado. El método de aprendizaje es sencillo: a la red se le muestra un conjunto de datos, llamado conjunto de datos de entrenamiento y compuesto por distintos vectores de entrada y sus salidas deseadas correspondientes; de manera que la red va modificando los pesos asociados a cada neurona para tratar de minimizar la diferencia entre la salida de la red y la salida deseada (esta diferencia es lo que se denomina error o *loss*).

Para llevar esta minimización a cabo, se utiliza el algoritmo conocido como el algoritmo de retropropagación de errores o *back propagation*. En él, una vez que la red obtiene un valor de salida en la capa de salida, los errores obtenidos se propagan hacia atrás (desde la salida hasta la entrada) actualizando los pesos de las diferentes neuronas mediante el gradiente de la función de error. Este proceso se repite de manera iterativa a medida que disminuye el error.
En el algoritmo de retropropagación de errores intervienen dos componentes: la función de pérdida o *loss function* y el algoritmo de optimización.

La **función de pérdida** hace referencia a la manera de calcular y cuantificar el error entre la salida de la red y la salida deseada. Un ejemplo es el error cuadrático medio explicado en (3.7).

El **algoritmo de optimización** hace referencia al algoritmo utilizado para encontrar los valores más adecuados con los que ajustar los pesos de la red con el objetivo de minimizar la función de pérdida escogida. De esta manera se calcula la aportación de cada neurona al error. Los pesos se van actualizando a una velocidad de acuerdo a la tasa de aprendizaje $\alpha \in [0, 1]$. Una tasa de aprendizaje cercana a 0 indica que los pesos cambian muy despacio, de manera que el error converge de forma lenta al error mínimo. Por el contrario, valores más cercanos

a uno dan lugar a una actualización de los pesos mucho mayor, disminuyendo el tiempo a la convergencia, pero aumentando los riesgos de que los pesos finales se alejen de los pesos óptimos, cayendo en mínimos locales que no corresponden con la mejor solución posible. Un ejemplo de algoritmos de optimización es el algoritmo *Adam*, explicado en [26]

Los entrenamientos de las redes neuronales son procesos costosos desde un punto de vista computacional, debido al cálculo secuencial de gradientes sobre un número cada vez mayor de neuronas. Por ello, en este trabajo se ha recurrido al entrenamiento tipo *bache*. En este tipo de entrenamiento el conjunto de entrenamiento se separa en subconjuntos de menor tamaño llamados baches, de manera que en vez de actualizar los pesos para cada entrada y salida del conjunto de entrenamiento, estos se actualizan para cada bache. El número de veces que la red procesa el conjunto de entrenamiento completo se denomina número de épocas o *epochs*.

3.4.2. Redes Neuronales Recurrentes

Los métodos clásicos de aprendizaje profundo no son capaces de tratar adecuadamente con datos secuenciales, pues asumen que los elementos de una serie de datos se generan de forma independiente y no se encuentran correlados con valores anteriores.

Los datos secuenciales hacen referencia a cualquier serie de datos cuyos elementos estén ordenados en secuencias y, por tanto, dependan de otros en la serie debido a su orden [27]. Ejemplos de datos secuenciales son las secuencias del ADN, clips de vídeo o los caracteres de un texto. Para dichas series de datos surgen los modelos secuenciales; modelos en los que los datos de entrada y/o los de salida son también secuencias de datos.

Dentro de estos modelos, las redes neuronales recurrentes *RNN* (*Recurrent Neural Network*) son un tipo de redes neuronales muy útiles para problemas secuenciales como es el caso del Procesamiento del Lenguaje Natural (PLN), reconocimiento de voz o la predicción de series temporales, pues la salida de estas redes neuronales depende de los datos de entrada actuales y de los datos tratados con anterioridad. Con esto se consigue que la información no fluya en un sentido único, sino que se produce una retroalimentación de la información al disponer de conexiones hacia atrás adicionales.

La propiedad más importante de las RNN es que tienen memoria interna a modo de bucles de retroalimentación que permiten que la información persista. Esto es diferencial en comparación con los métodos convencionales de redes neuronales, puesto que en las RNNs las características (los pesos) son *compartidos* a lo largo del tiempo. De esta manera, pueden recordar sus entradas anteriores utilizando datos históricos para el cálculo de los pesos.

En la Figura 3.7 se muestra un ejemplo básico de una red neuronal recurrente A que conecta una entrada x_t con una salida h_t:

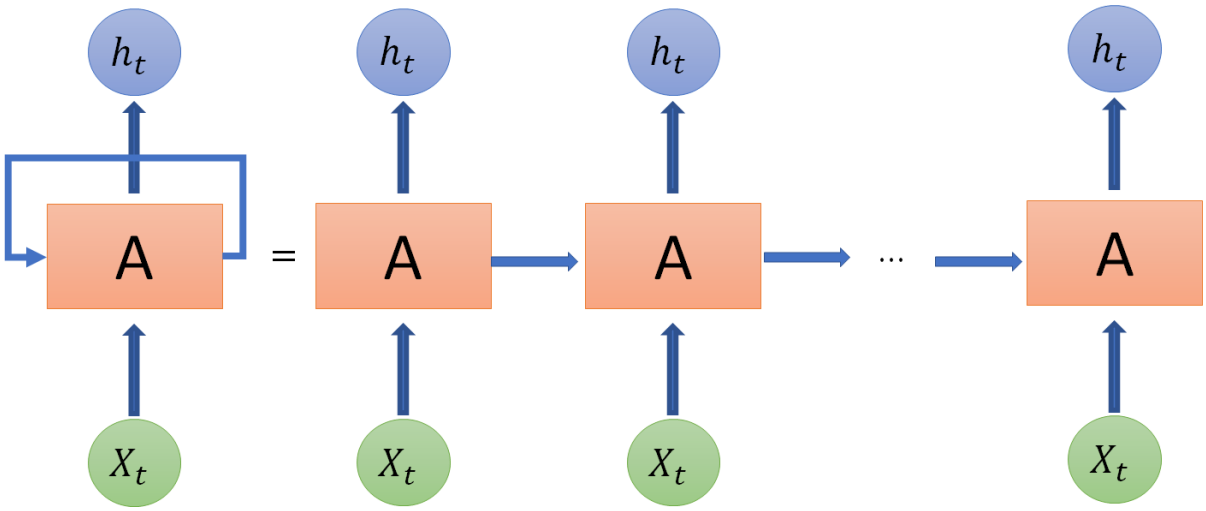


Figura 3.7: Funcionamiento interno de un fragmento de red neuronal recurrente. Fuente: propia.

Donde x_t es la entrada (input) actual y h_t el estado oculto actual. Estos bucles de retroalimentación permiten que la información sea transmitida desde un punto de la red al siguiente. Equivalentemente, las redes recurrentes se pueden entender como múltiples copias de redes neuronales corrientes, cada una pasando un 'mensaje' a su sucesora.

En la Figura 3.8 se muestra una unidad de la arquitectura típica de RNN convencional. La celda toma como entrada el resultado de la celda anterior y la entrada actual x_t, y utiliza como función de activación la función tangente hiperbólica $tanh(x)$

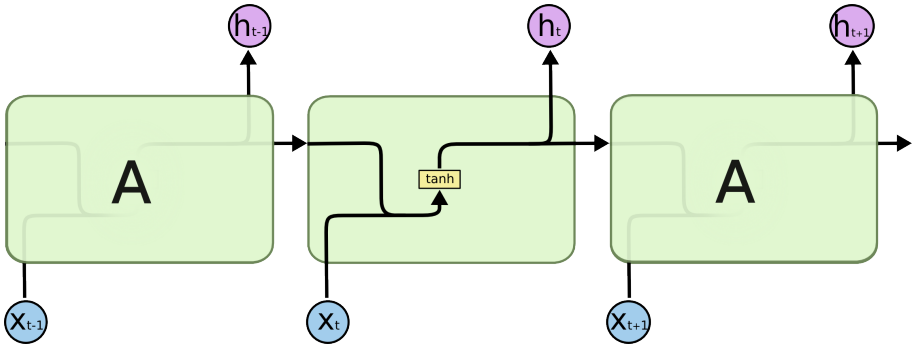


Figura 3.8: Módulo de repetición de red neuronal recurrente básica. Fuente: [28].

Las *RNNs* son muy útiles cuando necesitamos utilizar información reciente para poder realizar una tarea actual. Sin embargo, este no es el caso en la mayoría de los problemas. Una RNN expuesta a secuencias largas (con muchos pasos temporales previos) tiende a perder la información, puesto que es incapaz de almacenar la información de secuencias largas focalizándose solo en la información más reciente. Este problema se conoce como *el problema del desvanecimiento del gradiente* . Para entrenar modelos de redes neuronales recurrentes estándar, utilizamos el algoritmo de retropropagación para calcular gradiente en cada paso temporal y actualizar de esta manera los pesos de las celdas. El problema es que en sí el cambio anterior es pequeño, el gradiente también lo será y así; el gradiente se irá 'desvaneciendo' al adoptar cada vez valores de menor magnitud, reduciendo la efectividad de esta actualización de los pesos e

incluso impidiendo que la red pueda finalizar su entrenamiento.
Además, existe el problema contrario, denominado *Exploding gradient* en el que las diferencias se van acrecentando cada vez más. Para hacer frente a estos problemas surgieron nuevas estructuras basadas en redes recurrentes:

3.4.2.1. Redes *Long Short Term Memory* (LSTM)

Las redes LSTM o *Long Short Term Memory* [29] son capaces de hacer frente al problema anterior, pues su arquitectura les permite contar con memoria a largo y corto plazo y, por lo tanto, ser capaces de aprender de largos patrones temporales.

Al igual que las redes neuronales recurrentes, las LSTM también tiene una estructura de bloques, solo que en este caso, el módulo repetidor A de la arquitectura LSTM es más complejo que el de la RNN básica. En concreto, el módulo LSTM consta de cuatro capas: una **celda de estado**, y tres puertas: **puerta de entrada**, **puerta de salida** y **puerta de olvido**. En la Figura 3.9 puede verse el esquema del módulo repetir de una red LSTM, muy diferente al visto en la Figura 3.8.

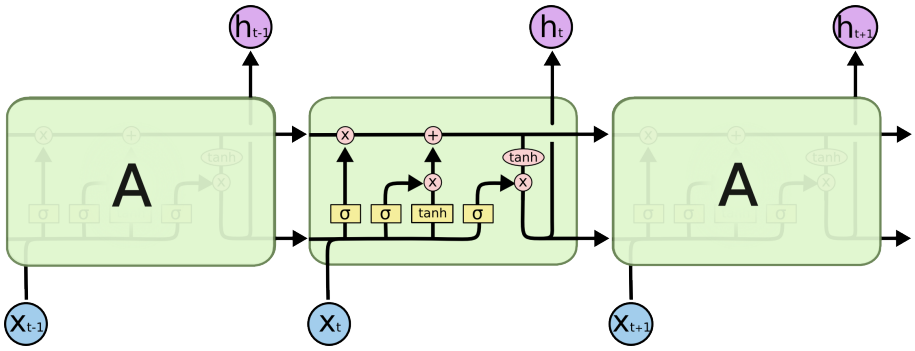


Figura 3.9: Módulo de repetición de una LSTM. Fuente: [28].

El estado de la celda actúa como banda transportadora, mientras que las tres puertas mencionadas anteriormente son capas capaces de añadir o eliminar datos al estado de la celda, regulando el flujo de información que entra y sale. Este comportamiento de la celda puede observarse en la Figura 3.10:

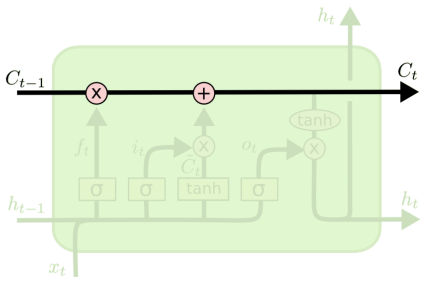


Figura 3.10: Evolución del estado de la celda a lo largo del módulo LSTM. Fuente: [28].

Estas puertas son redes neuronales que funcionan como compuertas: si están completamente abiertas permiten el paso de la información; mientras que si están completamente cerradas bloquean la información por completo.

Cada una de estas puertas está formada por tres elementos: una red neuronal, una función de activación sigmoide $f(x) = \frac{1}{1+e^{-x}}$ denominada σ que confiere el comportamiento de compuerta al devolver un valor entre 0 y 1 (cantidad de información que puede pasar en tanto por uno) y un elemento multiplicador denominado $\odot$ que representa el producto de Hadamard. De esta manera, las puertas construyen vectores de activación a través del vector de entrada $\mathbf{x}_t$ y el estado oculto o *vector de salida* del instante anterior, $\mathbf{h}_{t-1}$ cuyas componentes son valores numéricos entre 0 y 1, y que interactúan con el estado de la celda $\mathbf{c}_{t-1}$, añadiendo, eliminando y guardando información para producir el nuevo estado de la celda $\mathbf{c}_t$ y posteriormente el estado oculto o vector de salida en el instante t $\mathbf{h}_t$.

Para cada instante de tiempo t, entran tres vectores: el estado de la celda anterior $\mathbf{c}_{t-1}$, el estado oculto anterior $\mathbf{h}_{t-1}$ y la entrada del conjunto de datos original en el instante de tiempo t $\mathbf{x}_t$; y salen dos vectores, el estado de la celda y el estado oculto en el instante t ($\mathbf{c}_t$ y $\mathbf{h}_t$ respectivamente). El aprendizaje se realiza de la siguiente manera:
Primero actúa la *puerta de olvido*, decidiendo que información se va a descartar y, por lo tanto, no pasará al estado de la celda. Para ello, utiliza la función sigmoide σ con el estado oculto anterior y la entrada actual para generar el vector $\mathbf{f}_t$ *vector de activación de la puerta de olvido*, en el que cada componente está formada por un número entre 0 y 1 :

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right), \tag{3.26}$$

con $(\mathbf{W}_f, \mathbf{b}_f)$ coeficientes que se aprenden durante el entrenamiento. Un valor de 0 para una componente de $\mathbf{f}_t$ hace que se elimine completamente la información de la correspondiente componente del estado anterior $\mathbf{c}_{t-1}$, mientras que un valor de 1 hace que esa información se mantenga.

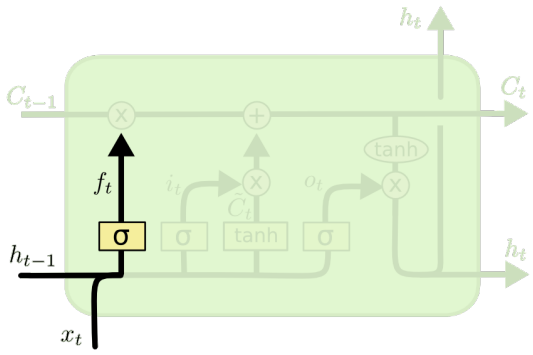


Figura 3.11: La capa *forget* decide que información no entra al estado de la celda. [28]

A continuación, se debe decidir qué información nueva va a ser almacenada en el estado de la celda. Este proceso se compone en dos partes: en primer lugar actúa la *puerta de entrada*, una capa σ que decide que información debe ser actualizada a través del vector $\mathbf{i}_t$ (*vector de*

activación de la puerta de entrada):

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right), \tag{3.27}$$

con $(\mathbf{W}_i, \mathbf{b}_i)$ coeficientes que se aprenden durante el entrenamiento.
En segundo lugar, una capa *tanh* genera un vector de nuevos valores candidatos $\bar{\mathbf{c}}_t$ (*vector de activación del estado de la celda*) que pueden ser añadidos al estado de la celda

$$\bar{\mathbf{c}}_t = tanh\left(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right), \tag{3.28}$$

con $(\mathbf{W}_c, \mathbf{b}_c)$ coeficientes que se aprenden durante el entrenamiento.

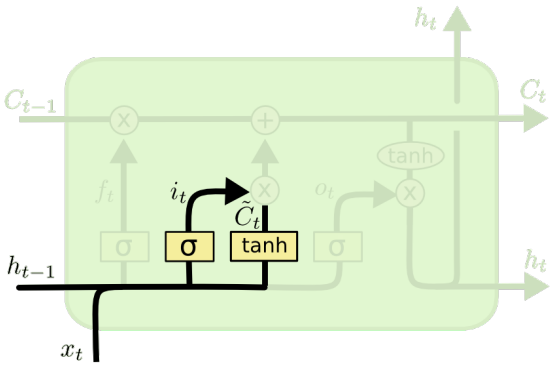


Figura 3.12: La capa de entrada decide que información nueva entra. $\bar{\mathbf{c}}_t$ consiste en el vector de candidatos a ser añadidos al estado de la celda. Fuente: [28].

Ahora elegimos entre estos valores, multiplicando el estado anterior por $\mathbf{f}_t$, 'olvidando' la información que se había decidido en la *puerta de olvido*, y añadiendo el vector $\bar{\mathbf{c}}_t$ multiplicado componente a componente por $\mathbf{i}_t$, es decir, los nuevos candidatos, pero escalados según cuánto queremos actualizar cada componente del estado de la celda; de manera que la memoria actualizada es:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \bar{\mathbf{c}}_t \tag{3.29}$$

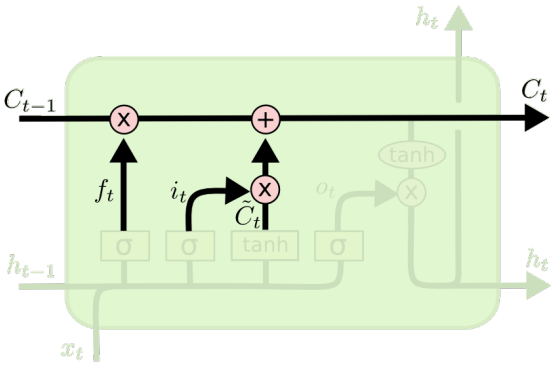


Figura 3.13: Se actualiza el nuevo estado de la celda. Fuente: [28].

Finalmente, se debe decidir cuál va a ser la salida; es decir, el nuevo estado oculto $\mathbf{h}_t$. Para ello, la *puerta de salida* utiliza la función σ que decide que componentes del estado de la celda deben salir a través del vector $\mathbf{o}_t$ (*vector de activación de la puerta de salida*).

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right), \tag{3.30}$$

con $(\mathbf{W}_o, \mathbf{b}_o)$ coeficientes que se aprenden durante el entrenamiento.

Después, el estado de la celda $\mathbf{c}_t$ pasa a través de una capa *tanh* que escala los valores entre -1 y 1 y se multiplica por el vector que filtraba la información que salía $\mathbf{o}_t$. De esta manera, el estado oculto en el instante t, que es la salida de la red LSTM en dicho instante, es

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{3.31}$$

.

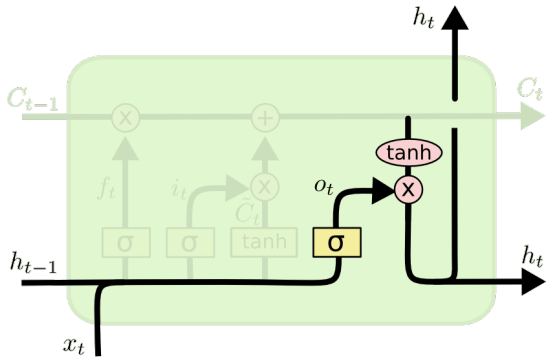


Figura 3.14: Salida del módulo de repetición. Corresponde con el estado oculto en el instante t, $\mathbf{h}_t$. Fuente: [28].

3.4.2.2. Redes *Gated Recurrent Unit* (GRU)

En 2014, Cho et al. [30] introdujeron el concepto de GRU (*Gated Recurrent Unit*) como una versión simplificada de la LSTM. GRU acopla las puertas de entrada y olvido en una sola puerta, llamada puerta de actualización destinada a definir que información anterior se va a mantener y cuyo vector de activación asociado se denomina $\mathbf{z}_t$. Además, elimina la puerta de salida e incorpora una nueva puerta, llamada puerta de reinicio, que define como incorporar a la entrada $\mathbf{x}_t$ la información del estado oculto anterior $\mathbf{h}_{t-1}$ a través de su vector de activación $\mathbf{r}_t$. De esta manera desaparece el estado de la celda y, por lo tanto, su vector asociado $\mathbf{c}_t$ utilizando como entradas únicamente los vectores $\mathbf{x}_t$ y $\mathbf{h}_{t-1}$; y como salida únicamente el estado oculto $\mathbf{h}_t$. En la Figura 3.15 se puede observar el módulo de repetición de una estructura tipo GRU con sus respectivas componentes.

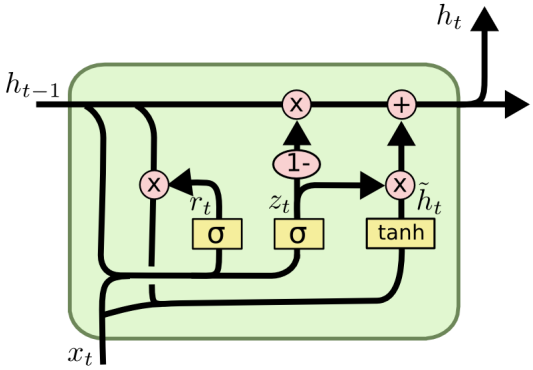


Figura 3.15: Módulo de repetición de una arquitectura GRU. Fuente: [28].

Las respectivas ecuaciones del módulo de repetición de una estructura tipo GRU son:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_z \cdot \mathbf{h}_{t-1}, \mathbf{x}_t\right]) \tag{3.32}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_r \cdot \left[\mathbf{h}_{t-1}, \mathbf{x}_t\right]\right) \tag{3.33}$$

$$\bar{\mathbf{h}}_t = tanh\left(\mathbf{W} \cdot \left[\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t\right]\right) \tag{3.34}$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t \tag{3.35}$$

Artículos como [31] [32] muestran como las estructuras tipo GRU ofrecen mejores resultados que las estructuras LSTM en conjuntos de datos más pequeños.

3.4.2.3. Redes *Bidirectional* LSTM (BI-LSTM)

En las redes LSTM y GRU la información solo fluye en una dirección; ya sea hacia adelante o hacia atrás. Para poder utilizar ambas direcciones (*pasado a futuro* y *futuro a pasado*) se utilizan las variantes bidireccionales. En ellas se entrenan dos modelos, de manera que el primer modelo aprende sobre la secuencia proporcionada, y el segundo modelo aprende sobre la secuencia inversa. Finalmente, se combinan los dos estados ocultos finales para realizar una predicción, por lo que en cada punto se está preservando información del pasado y del futuro. De esta forma, el algoritmo tiene un contexto mucho mayor para entender el problema y realizar una predicción más acertada. Un ejemplo básico para entender este concepto es completar una frase. Imaginemos que queremos completar la frase *voy a comer al ...*, existen múltiples opciones para esa frase; sin embargo, si contamos también con la frase del futuro *... porque me apetece mucho una hamburguesa Big Mac* es mucho más sencillo para el algoritmo encontrar la respuesta correcta. El modelo hacia adelante se alimentaría de la primera secuencia y el modelo hacia atrás de la segunda, para finalmente combinar los resultados en una predicción.

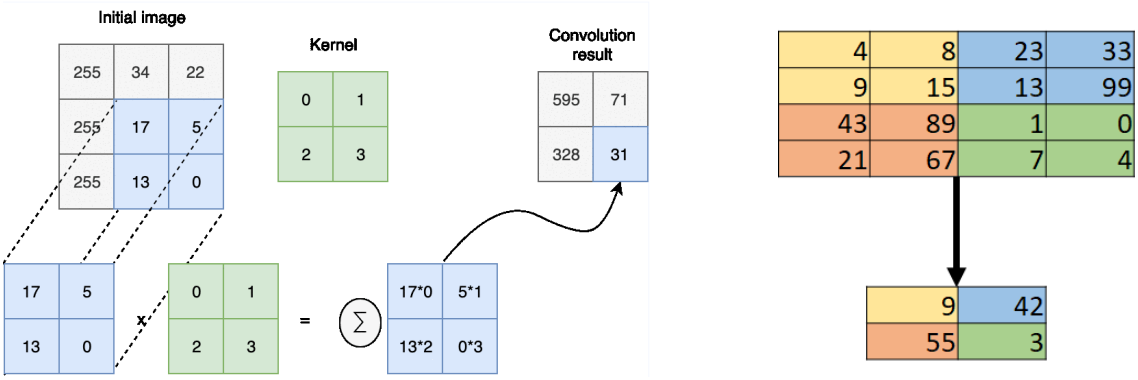
3.4.2.4. *Convolutional Neural Networks* LSTM (CNN-LSTM)

Se denomina redes convolucionales (CNN, *Convolutional Neural Networks*) a aquellas que hacen uso de las capas convolucionales. En estas capas convolucionales, las neuronas no se encuentran totalmente conectadas, sino que solo recibe información de algunas neuronas de la capa anterior. Esto supone la existencia de neuronas especializadas en conjuntos de neuronas de capas anteriores. Aparte de la ventaja computacional que supone(menos neuronas a optimizar),

se ha aprovechado este tipo de redes para la extracción de variables de alto nivel en campos como la visión por ordenador o el procesamiento del lenguaje. Esta arquitectura combina las *CNN*, utilizadas para la extracción de variables temporales, con las LSTM para la predicción de series temporales.
La arquitectura de las redes convolucionales está formada por las mismas capas que las redes *feedforward* más las capas convolucionales y las capas de *pooling* o sub muestreo.

En las **capas convolucionales**, los pesos son sustituidos por *filtros* o *kernel*, matrices de tamaño prefijado que interactúan con los datos de entrada a través de operaciones de convolución.

Las **capas de *pooling*** están situadas tras las capas convolucionales y permiten reducir la dimensionalidad de los datos para disminuir el coste computacional. Estas capas convierten una región de la entrada original de los datos en un valor numérico, según el tipo de *pooling* escogido. En este trabajo se ha utilizado como tipo de *pooling* el método *average pooling* en el que se toma el valor medio de la región.
En la Figura 3.16 es posible observar el efecto de ambas capas.



(a) (b)

Figura 3.16: (a) Ejemplo de aplicación de un filtro convolucional 2D de longitud 2. Fuente: [33]. (b) Ejemplo *average pooling* tamaño 2x2. Fuente: propia

Las cuatro arquitecturas utilizadas para la detección de anomalías por métodos de predicción de la serie ajustan sus pesos a través del algoritmo de retro-propagación para minimizar la función de pérdida escogida; en el trabajo: MSE (3.7). De esta manera, a través de estos pesos y de las funciones de activación no lineales de las diferentes neuronas, los métodos de aprendizaje profundo son capaces de lidiar con las complicadas dependencias temporales entre los datos.

Recientemente, debido a la popularidad de los algoritmos de aprendizaje profundo y de la disponibilidad de recursos tecnológicos necesarios para llevarlos a cabo (mejora de las gpu, disponibilidad de máquinas virtuales como *Google collab*, aumento de algoritmos *open-source*) se empezó a aplicar dichos algoritmos a tareas como la visión por ordenador, clasificación de emociones y tratamiento de datos secuenciales en forma de predicción o traducción de textos. Esto motivó a los diferentes investigadores a desarrollar algoritmos de aprendizaje profundo que pudieran detectar anomalías mediante métodos de predicción de la serie con redes recurrentes

como los trabajos [34] o [35].

3.4.3. *Autoencoders*

Un *Autoencoder* es un modelo de red neuronal que busca aprender una representación comprimida de los datos de entrada.
Este método de detección de anomalías basado en la reconstrucción de la serie utiliza una estructura de red neuronal para reducir la dimensionalidad de la serie y proyectarla en un espacio de dimensionalidad reducida.

La arquitectura de un *Autoencoder* consta de una primera parte (Codificación) que aprende una representación comprimida de la serie de datos mapeando dicha serie a un espacio oculto de menor dimensionalidad, y una segunda parte (Decodificación) que proyecta la serie de nuevo al espacio original. De esta manera, sea $\mathcal{D}$ el conjunto de datos, Θ la función de codificación y Ψ la función de decodificación, el conjunto de datos reconstruido por el *Autoencoder* $\hat{\mathcal{D}}$ es [9]:

$$\hat{\mathcal{D}} = \Psi(\Theta(\mathcal{D})) \tag{3.36}$$

En la Figura 3.17 se puede observar el esquema de un *Autoencoder* genérico.

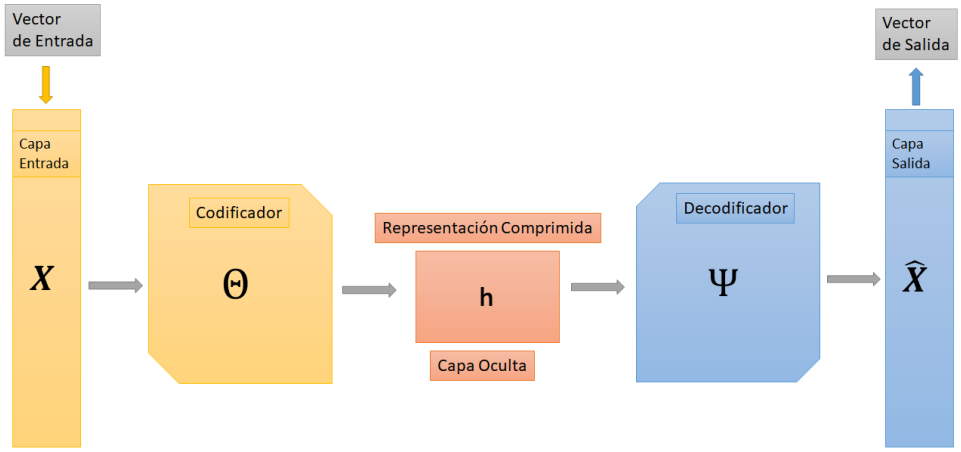


Figura 3.17: Esquema del funcionamiento de un *Autoencoder* genérico. Fuente: propia

El *Autoencoder* básico está formado por la capa de entrada $\mathbf{x} \in \mathbb{R}^m$, la capa oculta $\mathbf{h} \in \mathbb{R}^k$, y la capa de salida $\hat{\mathbf{x}} \in \mathbb{R}^m$ con la misma dimensionalidad que la capa de entrada. Los datos que entran a la capa de entrada son proyectados mediante la operación

$$\mathbf{h} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{3.37}$$

a un espacio comprimido (capa oculta) ($k < m$), y posteriormente, esta representación comprimida de los datos es proyectada de nuevo al espacio original (capa de salida) mediante la operación

$$\hat{\mathbf{x}} = \hat{\sigma}\left(\hat{\mathbf{W}}\mathbf{h} + \hat{\mathbf{b}}\right) \tag{3.38}$$

Con $\sigma, \hat{\sigma}$, las funciones de activación correspondientes; $\mathbf{W}, \hat{\mathbf{W}}$ la matriz que contiene los pesos y $\mathbf{b}, \hat{\mathbf{b}}$ los vectores *bias*. Estos *Autoencoders* se entrenan por el algoritmo de retropropagación

para que la salida sea idéntica a la entrada, ajustando los pesos de cada una de las diferentes capas para minimizar la función de pérdida escogida.

En el *Autoencoder* básico, los bloques de codificación y decodificación están formados por capas ocultas idénticas a las vistas en las redes neuronales artificiales; sin embargo, pueden usarse estructuras más complejas como las redes recurrentes, que sean capaces de capturar el comportamiento normal de los datos secuenciales.

3.4.3.1. LSTM *Autoencoder*

Un *Autoencoder* LSTM es una implementación del *Autoencoder* estudiado previamente aplicado a datos secuenciales a través de una arquitectura *Long Short Term Memory* como codificación-decodificación. La principal ventaja respecto al *Autoencoder* simple es la capacidad que le proporciona la estructura LSTM para aprender de las relaciones temporales de los datos de entrada.

La estructura es idéntica al *Autoencoder* simple, pero, en este caso, los perceptrones multicapa que conformaban el codificador y decodificador son sustituidos por estructuras de redes neuronales recurrentes, en concreto por redes LSTM como las estudiadas en 3.4.2.1. Esto le permite aprender relaciones temporales entre los datos de entrada que se producen a lo largo de grandes periodos temporales, lo que le permite al codificador entender mejor las características temporales de la serie y transformar a la serie original al espacio comprendido por estas características aprendidas.

Capítulo 4

Metodología a seguir

En este capítulo se describe el formalismo desarrollado y seguido en este trabajo. Primero se habla de las tres técnicas existentes para la detección de anomalías desde el punto de vista del aprendizaje; después se formula el problema matemáticamente, mostrando el valor numérico clave del problema: la **puntuación de anomalía**. Además, se explica como se ha decidido agrupar los diferentes algoritmos según los métodos de cálculo de puntuaciones de anomalías a los que pertenecen. Para finalizar, se explican los cuatro conjuntos de datos utilizados en este trabajo, los cuales sirven para valorar los algoritmos ante diversas series temporales y anomalías.

4.1. Metodologías para la Detección de Anomalías

Cada una de las técnicas de detección de anomalías pertenece a una de las tres categorías básicas:

- **Detección de anomalías como aprendizaje supervisado:** Es análogo al aprendizaje supervisado para los problemas de clasificación, en tanto que se asume que se dispone de etiquetas para discernir cada punto como anomalía o no. Esta manera de llevar a cabo la detección de anomalías requiere modelar tanto el comportamiento normal como el comportamiento anormal. De esta manera, el modelo en cuestión aprende del conjunto de entrenamiento, tratando de encontrar patrones para ambos tipos de comportamiento. Sin embargo, normalmente existe un gran des-balance entre el número de puntos pertenecientes a la clase anómala y no-anómala, debido a que las anomalías son eventos no comunes que ocurren con una frecuencia mucho menor que los eventos no-anómalos. Esto puede generar un sesgo en el modelo de clasificación que afecte a su funcionamiento. Existen trabajos [36] que utilizan modelos de generación de anomalías basadas en las características de las anomalías existentes y etiquetadas para crear un balanceo de las clases y poder utilizar métodos de clasificación para clases balanceadas.
- **Detección de anomalías como aprendizaje semi supervisado:** En el lenguaje semi-supervisado, un modelo se entrena solo con datos normales (sin anomalías). Cuando el modelo entrenado se utiliza en los nuevos puntos de datos, puede predecir si el nuevo punto de datos es normal o no. Para ello es necesario que aunque no se conozca la naturaleza del nuevo conjunto de datos, sí se asuma que el número de datos anómalos es mucho menor

que el número de datos no-anómalos. A este tipo de técnicas se les conoce también como *one class classification*.

- **Detección de anomalías como aprendizaje no supervisado:** En este tipo de técnicas se desconoce la naturaleza de los datos, tanto en las etapas de entrenamiento como de predicción. El modelo asume que la mayoría de los datos son normales y asigna cómo de probable es que cada punto sea anómalo.

Este trabajo se ha realizado según aprendizaje no supervisado; es decir, no se ha proporcionado a ningún algoritmo información alguna sobre si un punto era anómalo o no; ni se ha entrenado cada modelo con datos no-anómalos. Estas técnicas de detección de anomalías de forma no-supervisada se caracterizan por asociar un valor numérico, equivalente a una puntuación, a cada punto de la serie que indicase como de anómalo consideraba el modelo a dicho punto. Consecuentemente, las anomalías pueden ser asociadas a aquellos puntos con las puntuaciones más altas.

4.2. Formulación del problema

El objetivo es sencillo: detectar aquellos puntos extraños que se desvían significativamente de la distribución general de la serie de datos. Por ello es preciso disponer de un valor continuo que indique la magnitud de esta 'desviación' como el grado de anomalía de un punto. Este concepto es lo que en la literatura revisada se denomina puntuación de anomalía o *anomaly score*. Trabajos como [34] o [37] utilizan este término para puntuar cómo de probable es que un punto sea ánomalo.
Teniendo esto en cuenta, en este trabajo se ha definido matemáticamente la detección de anomalías de manera similar a [9]:
Sea $\mathcal{D}$ la serie temporal n-dimensional de longitud T, $\mathcal{D} = \{\mathbf{x}_t \mid \mathbf{x}_t \in \mathbb{R}^n, t \in \{1, \ldots, T\}, n \geq 1\}$, y $\mathbf{x}$ un elemento de dicha serie temporal, $\mathbf{x} \in \mathcal{D}$, la función ϕ se define como:

$$\begin{aligned} \phi \ &: \mathbb{R}^n \to \mathbb{R} \\ &\phi(\mathbf{x}) = \gamma \end{aligned} \tag{4.1}$$

donde γ es la puntuación de anomalía asignada por ϕ para $\mathbf{x}$.

El siguiente paso es determinar si dicha 'puntuación' es suficiente para considerar a $\mathbf{x}$ como anómalo. Para ello, el valor continuo γ debe ser convertido a un valor binario que sirva como etiqueta para diferenciar los puntos anómalos de los no-anómalos.

Esta clasificación binaria es llevada a cabo a través de un umbral o *threshold* $\theta \in \mathbb{R}$ tal que todos los puntos cuya puntuación de anomalía supere este umbral se consideraran anomalías. De esta manera, el formalismo de detección de anomalías se define a través de la

función Φ:

$$\Phi \ : \ \mathbb{R}^n \times \mathbb{R} \to \{0, 1\} \tag{4.2}$$

$$\Phi(\mathbf{x}, \theta) = \begin{cases} 1 & \text{Si } \phi(\mathbf{x}) > \theta \\ \\ 0 & \text{En otro caso} \end{cases} \tag{4.3}$$

Donde 1 identifica a x como punto anómalo y 0 como punto no anómalo.

Una vez se ha definido en qué consiste desde el punto de vista matemático la detección de anomalías, es momento de explicar como llevar a cabo este proceso. En este trabajo se estudian distintos métodos y algoritmos capaces de asociar una puntuación de anomalía a cada punto de la serie temporal. El algoritmo ideal es aquel que asocia las puntuaciones más altas a las anomalías reales, de manera que existe un valor umbral θ tal que todos los puntos de la serie cuya puntuación de anomalía calculada sea mayor que dicho umbral son puntos anómalos.
Este trabajo se centra en el **cálculo de las puntuaciones de anomalía**, más que en la elección de umbral, pues esta elección es un problema dependiente de la tipología del proyecto y de los datos, existiendo múltiples maneras de escogerlo diferentes según las necesidades del proyecto implicado. Una vez obtenida la puntuación de anomalía, se valora el desempeño del algoritmo a través de métricas de clasificación binarias. En este trabajo se ha escogido la puntuación *F1* como la métrica clave para comparar el desempeño de los modelos, por lo razonado en 3.1.1.
Estas métricas se obtienen al elegir como umbral el umbral ideal, definido como:

Definición 4.2.1 (**Umbral Ideal**)**.** Se denomina umbral ideal al valor θ para la puntuación de anomalía calculada por el algoritmo, tal que la puntuación *F1* que se obtiene al realizar la clasificación de los puntos con dicho umbral es la máxima posible para esa puntuación de anomalía.

De esta manera se está valorando cómo de bien calcula cada algoritmo la puntuación de anomalía. Un valor alto significa que existe un umbral capaz de separar con éxito los datos anómalos de los datos no anómalos, lo cual significa que las puntuaciones de anomalía más altas se están asociando a los puntos anómalos; mientras que un valor bajo significa que el mejor umbral posible para esa puntuación de anomalía no es capaz de separar correctamente los puntos anómalos y, por tanto, el desempeño del modelo para puntuar cómo de anómalo es un punto es peor que en el caso anterior (no asocia las puntuaciones más altas a los puntos anómalos).

Para realizar el trabajo, se han estudiado diferentes métodos capaces de calcular una puntuación de anomalía. Estos métodos para el cálculo de la puntuación de anomalía abarcan desde métodos estadísticos basados en una medida clásica como es la puntuación-Z: tomando esta puntuación como medida de la desviación de un punto respecto del comportamiento promedio de una serie; pasando por métodos de aprendizaje automático que calculan la puntuación de anomalía de forma no supervisada; y finalizando con métodos que calculan la puntuación de anomalía reconstruyendo la serie, ya sea a través de predicciones o a través de proyecciones a un espacio comprimido.

En la Tabla 4.1 se representa un esquema para categorizar cada uno de los algoritmos según el método que utilizan para calcular la puntuación de anomalía. Los algoritmos de aprendizaje automático calculan la puntuación de anomalía basándose en diferentes técnicas no supervisadas (*LOF* se basa en densidades, *OC-SVM* se basa en máquinas vector soporte y los árboles de aislamiento se basan en la profundidad de los nodos de los árboles construidos), pero los tres algoritmos se han implementado exclusivamente a través de la función de la librería de *Python Sklearn* de forma idéntica y utilizando métodos puramente no-supervisados, por lo que se ha decidido agruparlos en la misma categoría.

Tabla 4.1: Categorías de los diferentes métodos estudiados. Los métodos resaltados se han aplicado tanto sobre la serie original normalizada como sobre la serie normalizada y diferenciada.

Métodos estadísticos	**Puntuación-Z** Puntuación-Z móvil
Métodos aprendizaje automático no supervisado	**LOF** **OC-SVM** **Árboles de aislamiento**
Métodos basados en la predicción de la serie	XGBoosting LSTM GRU BI-LSTM CNN-LSTM
Métodos basados en la reconstrucción de la serie	Autoencoders LSTM-Autoencoders

4.3. Conjunto de Datos

Para poder evaluar los distintos algoritmos descritos, así como sus modificaciones, se han utilizado conjuntos de datos de series temporales **univariantes** con anomalías etiquetadas. Estos conjuntos de datos han sido usados como pruebas para algoritmos de detección de anomalías en muchos artículos como por ejemplo [37] y [9].

En concreto, se han utilizado los datos proporcionados y publicados por Yahoo [38]. Estos datos pueden dividirse en 4 conjuntos:

- **A1 – Real Yahoo Services Network traffic**
  Este conjunto de datos contiene el tráfico de datos de los diferentes servicios de Yahoo, siendo las anomalías etiquetadas por humanos. En concreto, contiene 67 series temporales diferentes, cada una con entorno a 1400 instancias temporales, grabadas con una frecuencia horaria. En promedio, el 1.9 % de los datos son anómalos.
- **A2 – Synthetic Yahoo Services Network traffic** En este caso consiste en 100 series temporales de datos sinteticos conteniendo anomalías, en las que las anomalías se han insertado de manera aleatoria. En promedio, cada serie temporal contiene un 0.3 % de datos anómalos.
- **A3 – Synthetic Yahoo Services with Seasonality** Este conjunto de datos contiene 100 series temporales cada una con un promedio de 1680 instancias temporales y un 0.3 % de puntos anómalos.
- **A4 – Synthetic Yahoo Services with Seasonality** Este conjunto de datos contiene 100 series temporales cada una con un promedio de 1680 instancias temporales y un 0.5 % de puntos anómalos, además de contar con puntos en los que la tendencia de la serie cambia.

Estos conjuntos de datos difieren mucho respecto a la tipología de sus anomalías. El conjunto más sencillo es el conjunto A2, que representa series temporales sintéticas a las que se le ha añadido ruido en valores arbitrarios. Sus anomalías son anomalías contextuales similares a las anomalías contextuales vistas en la Figura 2.4. Estas anomalías están normalmente formadas por parejas del mismo valor y rompen claramente con el comportamiento temporal que sigue la serie sintética.

Los conjuntos A3 y A4 son los más parecidos. Consisten en series temporales sintéticas con tendencias claras y diferentes componentes estacionales a lo largo del tiempo. Sus anomalías representan anomalías contextuales y puntuales difíciles de detectar, pues su comportamiento anómalo no es anómalo respecto a la serie entera, sino respecto al periodo en el que se encuentra dicha anomalía. Son las series temporales más complejas debido a su gran variabilidad, tanto en los datos como en las dependencias temporales que siguen su media y desviación estándar. Esta gran variabilidad, junto al hecho de que las anomalías son contextuales y del mismo orden que los puntos no anómalos de la serie, hace que se encuentren 'escondidas' entre datos normales.

Finalmente, el conjunto A1 está formado por series temporales reales con anomalías reales y etiquetadas manualmente. Estas anomalías son, en su gran mayoría, anomalías colectivas y

aparecen en una sub-secuencia de gran tamaño que representa un comportamiento anómalo prolongado en el tiempo. Estas regiones anómalas difieren en gran medida de la región no anómala (son valores extremos de la serie) por lo que no es necesario comprender las dependencias temporales de la serie, simplemente separar las regiones anómalas de las regiones normales, como si de datos no temporales se tratase.

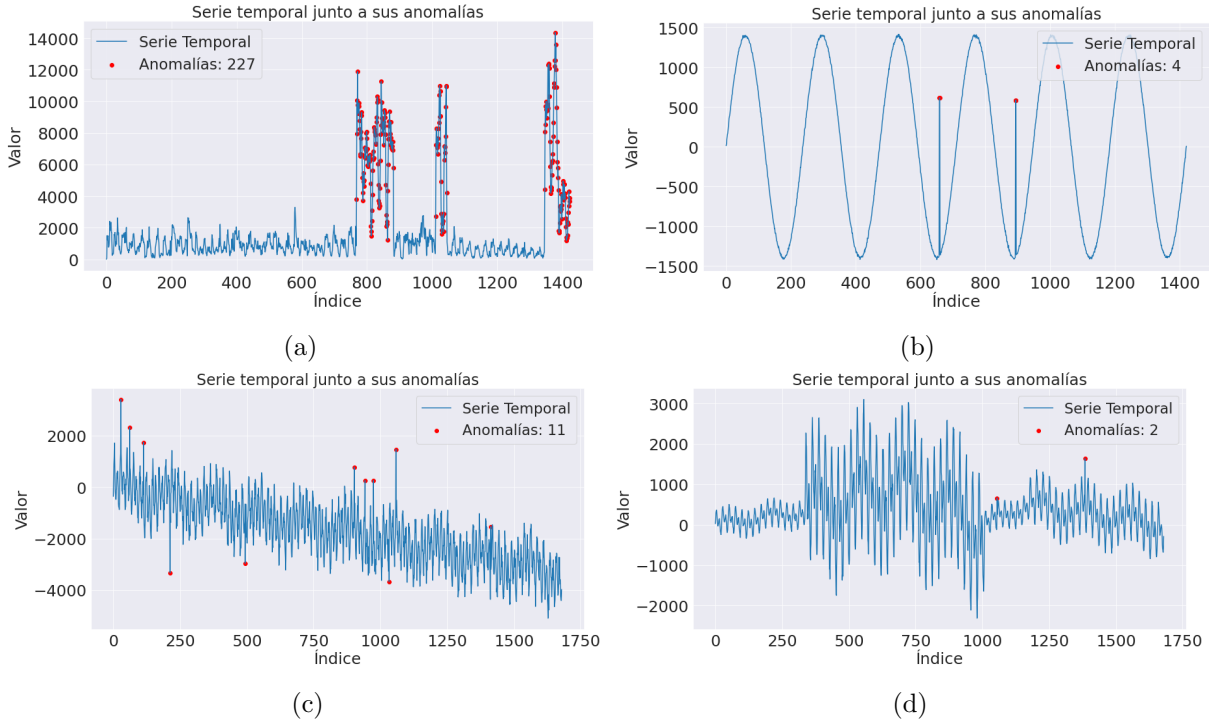


Figura 4.1: (a) Serie temporal número 17 del conjunto A1 junto a sus anomalías. (b) Serie temporal número 1 del conjunto A2 junto a sus anomalías. (c) Serie temporal número 1 del conjunto A3 junto a sus anomalías. (d) Serie temporal número 26 del conjunto A4 junto a sus anomalías.

Capítulo 5

Análisis y desarrollo de los métodos

En este capítulo se describen los métodos mostrados en la Tabla 4.1 para el cálculo de la puntuación de anomalía, así como su comportamiento experimental al aplicar dichos métodos. Con ello, se pretende mostrar el funcionamiento de estos algoritmos y cómo asignan la puntuación de anomalía. Además, la visualización de las series temporales estudiadas y el análisis de los diferentes métodos dependiendo de los hiperparámetros escogidos se realizó utilizando la aplicación web desarrollada. Para mostrar dicha aplicación web interactiva sin alargar excesivamente el trabajo, se muestran imágenes del funcionamiento y estilo de web en el apéndice C.

5.1. Métodos estadísticos

Los algoritmos pertenecientes a estos métodos se basan en técnicas estadísticas clásicas, simples pero poderosas, que pueden ser usadas fácilmente para la detección inicial de valores atípicos. Si bien en muchos casos es necesario el uso de algoritmos más complejos capaces de capturar el comportamiento anómalo de un punto, a veces las técnicas estadísticas simples son suficientes para la detección de anomalías, especialmente para series con una dependencia temporal baja o con anomalías puntuales claras.

5.1.1. Puntuación-Z absoluta

El *Z-score* o *standard score* es un concepto importante en estadística que representa cuantas desviaciones estándar se aleja un valor de la media. Esta técnica tan simple servirá para poder explicar los conceptos previos de ϕ, γ y θ.
Sea $\mathcal{D}$ una serie de datos unidimensional $\mathcal{D} = \{x_t \mid x_t \in \mathbb{R},\ t = \{1, \ldots, T\}\}$ con media μ y desviación estándar $\sigma = \sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(x_i - \mu)^2}$. Dado un punto $x_t \in \mathcal{D}$, su puntuación-Z o *Z-score* es:

$$Z(x_t) = \frac{x_t - \mu}{\sigma}. \tag{5.1}$$

El valor absoluto de esta puntuación-Z $|Z(x_t)|$ representa el número de desviaciones estándar que se separa el punto $x_t \in \mathcal{D}$ de la media de la distribución de $\mathcal{D}$.
Es posible utilizar el valor absoluto de esta puntuación-Z como puntuación de anomalía, siendo aquellos puntos con una mayor puntuación-Z los más probables de ser anómalos.
Así, escogiendo un umbral θ apropiado, la detección de anomalías mediante las puntuaciones-Z se formaliza como:

$$\Phi_Z(x_t, \theta) = \begin{cases} 1 & \text{Si } \phi_Z(x_t) = |Z(x_t)| = \left|\frac{x_t - \mu}{\sigma}\right| > \theta \\ 0 & \text{En otro caso} \end{cases} \tag{5.2}$$

Esta técnica de detección de anomalías es capaz de detectar aquellos valores extremos que se desvían en gran medida del comportamiento de la serie de datos. Además, es un método simple desde un punto de vista computacional, lo que lo hace ideal como método preliminar de detección de anomalías.
En la Figura 5.1 se representa la serie temporal número 3 del conjunto de series temporales reales de Yahoo número 1 *A1 – Real Yahoo Services Network traffic*. En ella se puede observar la presencia de anomalías, en especial la última región donde tenemos un conjunto de anomalías colectivas.

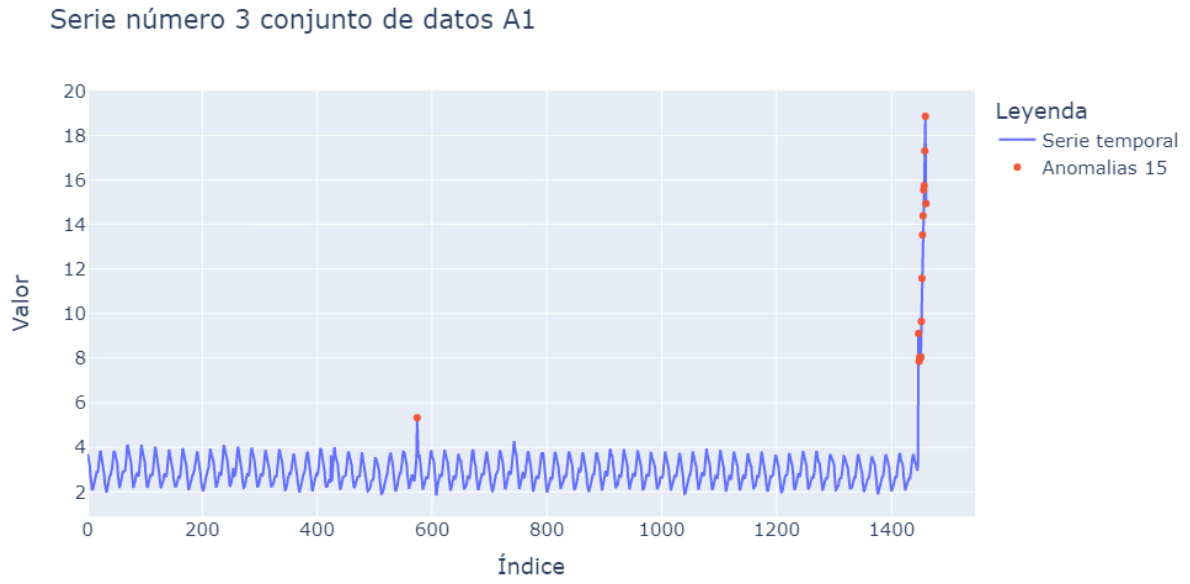
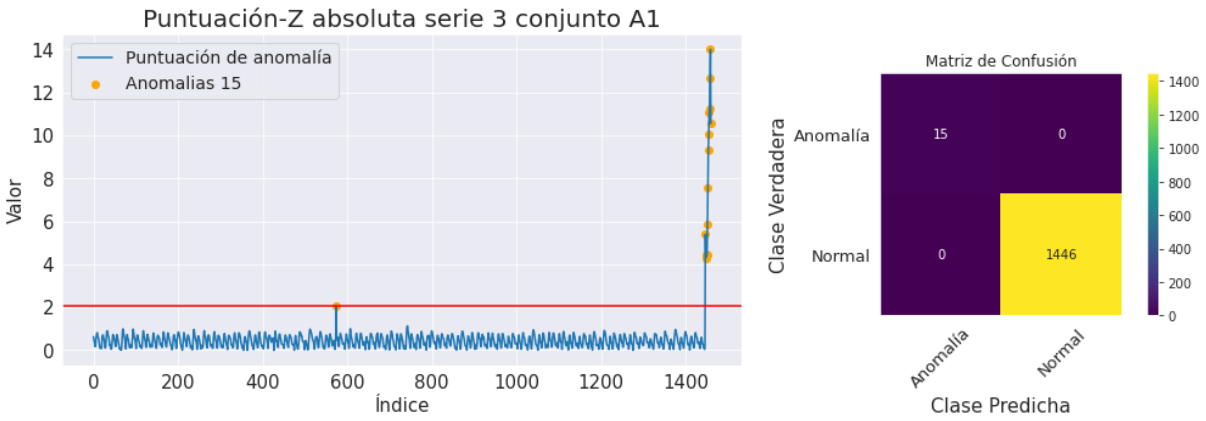


Figura 5.1: Serie temporal número 3 del conjunto de series A1 junto a sus anomalías.

En la Figura 5.2 se representa el valor de la puntuación de anomalía obtenida a través de la función $\phi_z(x_i)$ para cada punto x_i perteneciente la serie temporal 3 del conjunto A1, junto al mejor umbral obtenido para separar los puntos anómalos y no anómalos con su respectiva matriz de confusión.



(a) (b)

Figura 5.2: (a) Puntuación de anomalía obtenida por el método de las puntuaciones-Z absolutas, junto a las anomalías reales y el *umbral ideal* (rojo), aquel umbral que mejor métrica *F1* obtenía. (b) Matriz de confusión obtenida usando como umbral θ el umbral ideal.

Como se observa, la puntuación de anomalía calculada por el método de las puntuaciones-Z absolutas es suficientemente válida para esta serie temporal, pues aquellos puntos con una mayor puntuación corresponden con las anomalías reales.

Sin embargo, al trabajar con series temporales, hay que tener en cuenta las variaciones temporales de la media y desviación estándar de la serie.
En la Figura 5.3, se representa un ejemplo de una serie temporal con tendencia y componente estacional. Los datos corresponden al número de pasajeros que registro una aerolínea americana desde 1949 hasta 1960 [39]

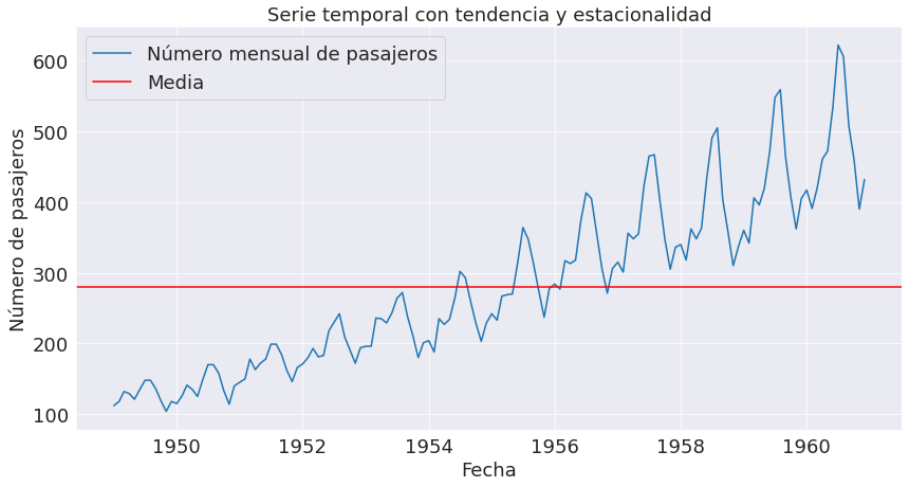


Figura 5.3: Ejemplo de serie con tendencia y componente estacional. El eje y representa el número de pasajeros registrados por la aerolínea, mientras que el eje x representa el instante temporal. Fuente: propia

Se observa como la media y desviación estándar del conjunto de datos no aportan gran información, pues se trata de una serie que evoluciona en el tiempo. Concretamente, se trata de

una serie no estacionaria en media y desviación estándar, por lo que utilizar las puntuaciones-Z para medir la desviación de un punto respecto a la media de la serie total no es la mejor manera de puntuar como de anómalo es un punto, pues esta media se encuentra en constante evolución. Por ello, en este trabajo se han desarrollado y formulado matemáticamente dos variaciones simples pero efectivas en la detección de anomalías mediante puntuaciones-Z absolutas para lidiar con las dependencias temporales de la serie. Estas formulaciones, aunque se basan en conceptos conocidos como diferenciación de la serie y ventanas móviles, no se han encontrado formuladas en la literatura revisada sobre detección de anomalías en series temporales.

5.1.2. Puntuación-Z absoluta sobre la serie diferenciada

Con el método de las puntuaciones-Z absolutas se utilizan la media y desviación estándar del conjunto de observaciones para puntuar cuánto se desvía una observación de la media. Sin embargo, al trabajar con series temporales, es frecuente que la media y la desviación estándar de la serie temporal evolucionen a lo largo del tiempo (series temporales no estacionarias). En este tipo de series puede resultar inefectivo el uso de las puntuaciones-Z absolutas sobre la serie original.
Una manera de estacionalizar la serie es el uso del operador ∇ para diferenciar la serie, como se comentó en la sección 2.2.2, consiguiendo una serie estacionaria en media y varianza; siendo apta para utilizar el método de las puntuaciones-Z absolutas.
Como ejemplo para ilustrar este caso puede utilizarse la serie temporal número 89 del conjunto de datos de Yahoo número 4 *A4 – Synthetic Yahoo Services with Seasonality.*

En la Figura 5.4 se representa dicha serie de datos junto a las anomalías correspondientes. Se trata de una serie con una tendencia descendente al principio, que más adelante se vuelve ascendente; y que además cuenta con una componente estacional.

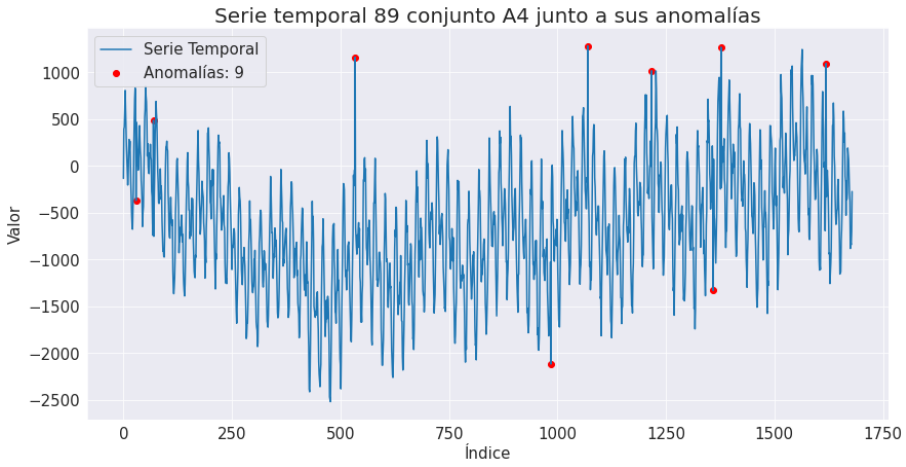


Figura 5.4: Serie temporal número 89 del conjunto de series A4. Cuenta con tendencia y estacionalidad, por lo que no es una serie estacionaria

En la Figura 5.5 se representa el valor de la puntuación-Z absoluta calculada como en (5.2) sobre la serie de datos anterior; junto con el umbral ideal. La Figura muestra como **no** es posible

encontrar un umbral capaz de separar bien los datos anómalos de los que no lo son, por lo que la puntuación de anomalía calculada por el modelo no asigna las puntuaciones más altas a los puntos anómalos.

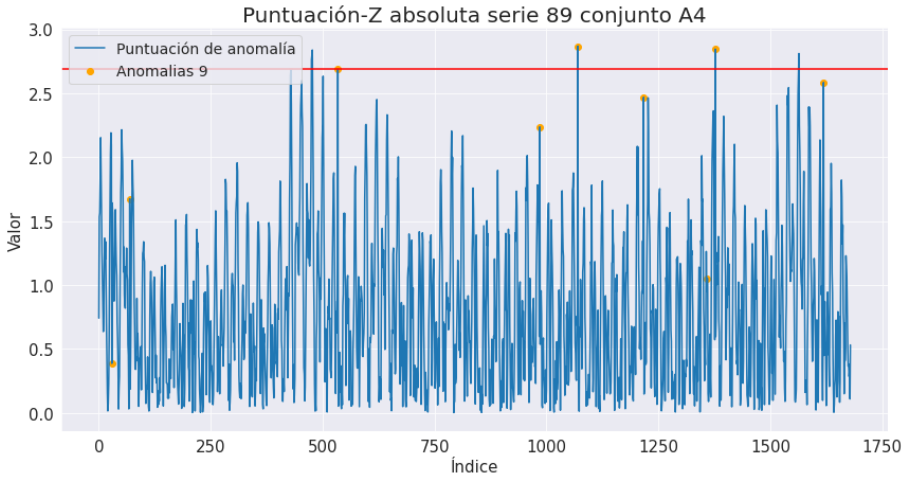


Figura 5.5: Puntuación de anomalía obtenida con el método de las puntuaciones-Z absolutas, junto a las anomalías reales y el *umbral ideal* (rojo).

En cambio, al transformar la serie temporal diferenciándola, esta se vuelve estacionaria en media y varianza, siendo efectivo para esta nueva serie transformada el uso de las puntuaciones-Z absolutas como puntuación de anomalía, como se verá más adelante.

La detección de anomalías con series temporales diferenciadas tiene dos inconvenientes: no se puede realizar esta detección sobre el primer punto de la serie temporal, pues no tiene ningún valor con el que establece la diferencia; y el hecho de que al trabajar con las diferencias existe la posibilidad de detectar puntos no anómalos como anómalos. Supongamos que se tiene la siguiente secuencia: $X = \{1, 2, 3, 75, 4\}$; claramente el punto $X_4 = 75$ es una anomalía. Si se aplica el operador ∇ a la serie $\nabla X = \{1, 1, 72, -71\}$, su puntuación de anomalía según la función $\phi_z(x)$ es $(\nabla X)_Z = \{0.005, 0.005, 1.41, 1.41\}$. Al elegir un umbral para separar las anomalías de la serie diferenciada; se detectarían los últimos dos puntos $(X_4 - X_3)$ y $(X_5 - X_4)$ debido a que la diferencia en valor absoluto de 75 a 3 es del mismo orden que de 75 a 4.
Por ello, en este trabajo se ha exigido que una vez la serie sea diferenciada, si dos puntos consecutivos son etiquetados como anomalías y sus valores diferenciados son de signo distinto, solo se considerará anómalo el primero de ellos, puesto que el segundo podría ser perfectamente un punto no anómalo. De esta manera se está priorizando que los puntos que se clasifican como no anómalos lo sean verdaderamente. Esto es importante, pues en muchos casos de la vida real la detección de anomalías se aplica a productos cuya eliminación supone pérdidas económicas para la empresa.

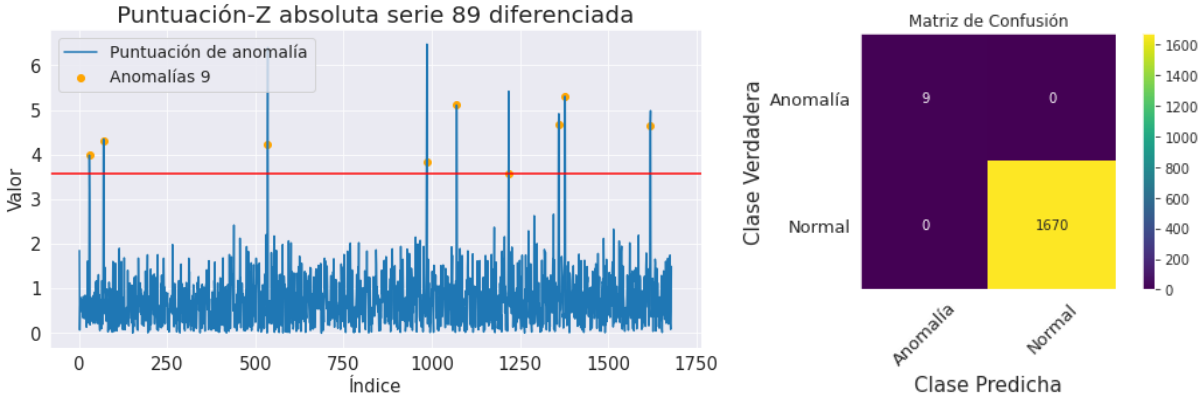
Sea $\mathcal{D}' = \{x'_t \mid x'_t \in \mathbb{R},\ t \in \{2, ..., T\}\}$ la serie temporal diferenciada $\nabla\mathcal{D}$ tal que $x'_i = x_i - x_{i-1}$, $i \in \{2, ..., T\}$ con media μ' y desviación estándar de la distribución σ', es posible definir el proceso de detección de anomalías de puntuaciones-Z absolutas sobre la serie diferenciada a

través de:

$$\Phi_{Zdiff}(x'_t,\theta) = \begin{cases} 1 & \text{Si } \phi_Z(x'_t) = |Z(x'_t)| = |\frac{x'_t-\mu'}{\sigma'}| > \theta \text{ y } \frac{x'_{t-1}\cdot x'_t}{|x'_t||x'_{t-1}|} + \Phi_{Zdiff}(x'_{t-1},\theta) \neq 0 \\ 0 & \text{En otro caso} \end{cases} \tag{5.3}$$

donde el proceso se realiza de forma secuencial siguiendo el orden natural de la serie temporal (orden temporal).
Para el ejemplo anterior, al elegir un umbral $\theta \in (0.005, 1.41)$ con esta nueva restricción, solo se detectaría como anomalía el punto correspondiente al valor $X_4 = 75$ pues $X'_4 = X_4 - X_3$ y $X'_5 = X_5 - X_4$ poseen signo distinto, y X'_4 ya ha sido detectado como anomalía, $\Phi_{Zdiff}(\hat{X}_4,\theta) = 1$ por lo que $\frac{X'_4\cdot X'_5}{|X'_5||X'_4|} + \Phi_{Zdiff}(X'_4,\theta) = 0$.

Este método detecta con facilidad las anomalías que no siguen la tendencia de sus puntos anteriores, pero va a ser mucho más complicado detectar secuencias de puntos anómalos debido a que la diferencia entre estos valores no tiene por qué ser una anomalía; aunque ambos puntos lo sean.
Como se verá a continuación, la condición expuesta en (5.3) para la detección de anomalías sobre series temporales diferenciadas, **desarrollada en este trabajo** y no encontrada en la literatura estudiada, asegura que al trabajar con series temporales diferenciadas se asocian los puntos anómalos al índice de la serie original correspondiente y no adicionalmente al punto siguiente.
Volviendo a la serie temporal anterior, en la Figura 5.6 se representa el valor de las puntuaciones Z absolutas obtenidas sobre la serie temporal representada en la Figura 5.4, pero **diferenciada**; junto al umbral ideal y la matriz de confusión correspondiente a dicha clasificación, teniendo en cuenta la ecuación (5.3). Ahora la puntuación de anomalía calculada es válida para la detección de anomalías, pues es posible encontrar un umbral que divida los puntos anómalos de los no anómalos, lo que quiere decir que se están asociando las puntuaciones de anomalía más altas a los puntos anómalos.



(a) (b)

Figura 5.6: (a) puntuación de anomalía obtenida por el método de las puntuaciones-Z absolutas aplicado a la serie diferenciada, junto a las anomalías reales y el *umbral ideal* (rojo). (b) Matriz de confusión obtenida usando como umbral θ el umbral ideal y la ecuación (5.3)

Puede observarse en la Figura 5.6 (a) como hay puntos no anómalos cuya puntuación de anomalía es superior al umbral ideal; sin embargo, gracias a las condiciones impuestas en la ecuación (5.3), estos puntos no son detectados como anomalías a pesar de superar el umbral, por lo que no existen anomalías mal clasificadas en la Figura 5.6 (b). Queda patente, por tanto, la importancia de la ecuación (5.3) para garantizar que al trabajar con series diferenciadas, la detección de anomalías se realiza de manera correcta. Sin embargo, como se ha comentado en 2.2.2, muchas veces la diferenciación de orden uno de la serie no es suficiente para transformar la serie en una serie estacionaria.
Para ello, en este trabajo se ha estudiado con gran éxito una variante del uso de la puntuación-Z como puntuación de anomalía.

5.1.3. Puntuación-Z con ventanas móviles

Esta variante consiste en utilizar ventanas móviles como las que se explican en 2.2.3. Esto es equivalente a estudiar la puntuación-Z de cada punto con la media y desviación estándar de su respectiva ventana. En estas ventanas, la media y desviación estándar de la serie poseen una variación menor a lo largo del tiempo en comparación con la serie completa. De esta manera, se obtienen puntuaciones de anomalías altas para aquellos puntos cuyo comportamiento se desvía mucho del comportamiento **en su ventana**.
Denominando como $\mu(w)_t$ y $\sigma(w)_t$ a las respectivas medias y desviaciones estándar de las distribuciones de la ventana t-ésima de tamaño w, es posible definir una nueva puntuación-Z dependiente del tamaño de la ventana elegida w como:

$$Z_w(x_i) = \frac{x_i - \mu(w)_i}{\sigma(w)_i} \tag{5.4}$$

Por lo que pueden definirse las funciones $\phi_Z^w(x_i)$ y $\Phi_Z^w(x_i;\theta)$ como

$$\Phi_Z^w(x_i,\theta) = \begin{cases} 1 & \text{Si } \phi_Z^w(x_i) = |Z_w(x_i)| = |\frac{x_i-\mu(w)_i}{\sigma(w)_i}| > \theta \\ \\ 0 & \text{En otro caso} \end{cases} \tag{5.5}$$

En concreto, de los dos tipos de ventanas móviles que se introdujeron en 2.2.3, se han escogido las ventanas móviles centradas. Estas ventanas móviles se han implementado a través de la función *rolling* de la librería *Pandas* del lenguaje de programación de *Python*. En estas ventanas móviles, si el tamaño de la ventana es impar, se asigna como puntuación de anomalía la puntuación correspondiente al valor central de la ventana. Sí, por el contrario, el tamaño de la ventana es par, se asigna la puntuación al valor central de la ventana de tamaño $w-1$, siendo w el tamaño de la ventana original (se toma el centro de la ventana compuesta por todos los elementos de la ventana original, excepto el primero).
Supóngase la ventana móvil de tamaño $w = 3$ y centrada en i: $W = \{x_{i-1}, x_i, x_{i+1}\} = \{4, 5, 6\}$ con $i > 1$, en este caso, $\mu(3)_i = 5$, $\sigma(3)_i = 1$ por lo que la puntuación de anomalía para el elemento i-ésimo de la serie sería $|\frac{5-5}{1}| = 0$. Por el contrario, supóngase la ventana móvil de tamaño $w = 4$ y centrada en i: $W = \{x_{i-2}, x_{i-1}, x_i, x_{i+1}\} = \{3, 4, 5, 6\}$ con $i > 1$, en este caso, $\mu(4)_i = 4.5$, $\sigma(4)_i = 1.29$ por lo que la puntuación de anomalía para el elemento i-ésimo de la serie sería $|\frac{5-4.5}{1.29}| = 0.39$.

En la Figura 5.7 se representa la serie temporal número 22 del conjunto de datos A4 de Yahoo. Como puede observarse, esta serie temporal posee cambios en la tendencia y estacionalidad de la serie. Además, cuenta con anomalías puntuales (valores extremos) y sobre todo con anomalías contextuales a lo largo de la serie. Son anomalías contextuales, puesto que los valores que alcanzan no son valores extremos ni claramente *outliers*; sin embargo, bajo el contexto de los puntos de su alrededor, sí que son puntos anómalos con un comportamiento diferente al resto.

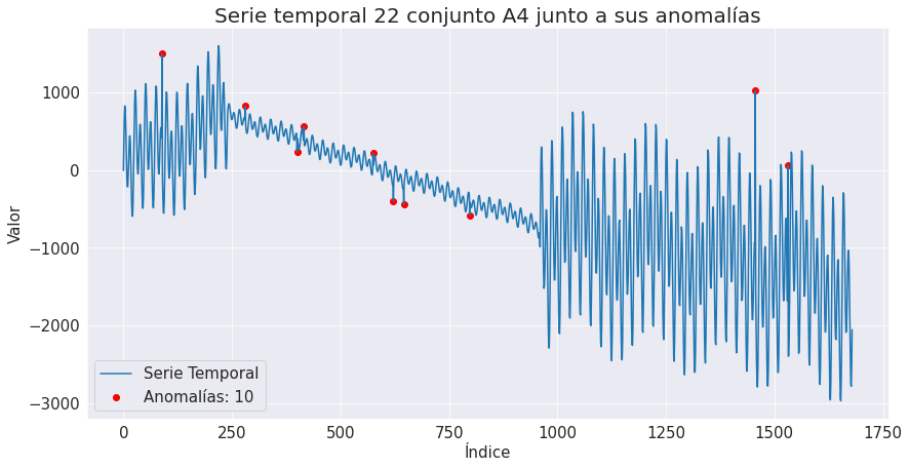
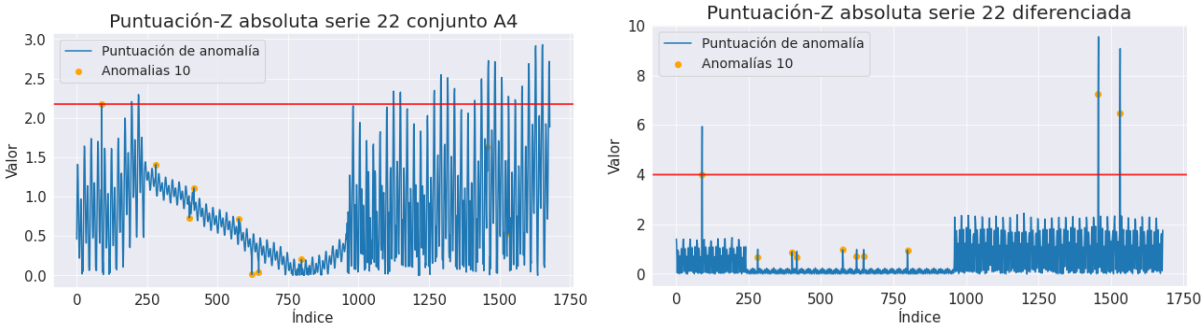


Figura 5.7: Serie temporal número 22 del conjunto A4 de Yahoo junto a sus 10 anomalías.

En la Figura 5.8 se representa la puntuación de anomalía obtenida por el método de la puntuación-Z sobre la serie original y sobre la serie diferenciada

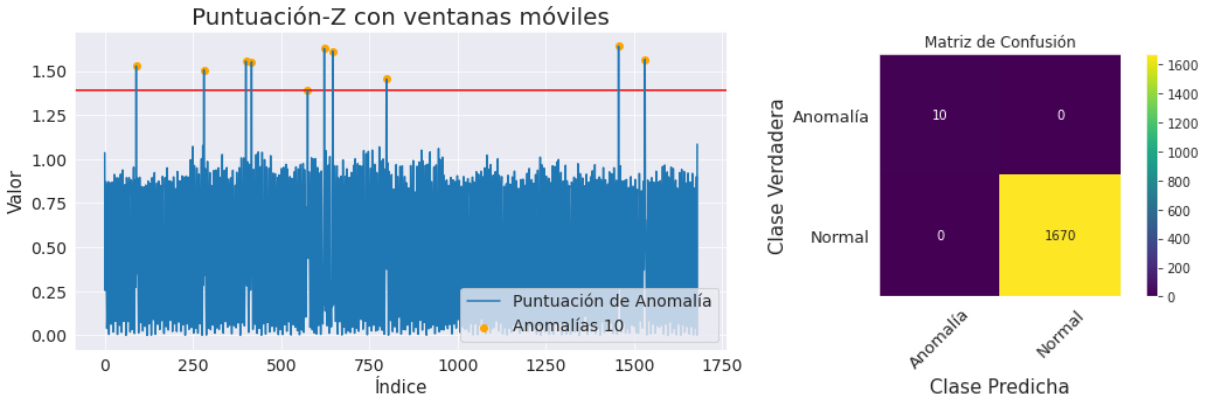


(a) (b)

Figura 5.8: (a) puntuación de anomalía utilizando el método de las puntuaciones-Z absolutas sobre la serie 22 del conjunto A4, junto a las anomalías reales y el *umbral ideal*(rojo). (b) puntuación de anomalía utilizando el método de las puntuaciones-Z absolutas sobre la serie diferenciada número 22 del conjunto A4, junto a las anomalías reales y el *umbral ideal* (rojo).

Como puede observarse, la puntuación de anomalía no es lo suficientemente buena, pues no asigna a las anomalías las puntuaciones de anomalía más altas. De hecho, las puntuaciones de anomalía más altas no se corresponden con los puntos anómalos de la serie; lo que significa que la forma de asignar la puntuación de anomalía no es la correcta.
Si se utiliza ahora la puntuación-Z con ventanas móviles definida en (5.5), se está estudiando las puntuaciones de anomalía de cada punto para su respectiva ventana y no para la serie completa;

en esta ventana de tiempo la serie puede considerarse estacionaria de manera que los puntos más anómalos de cada ventana tendrán una puntuación similar independientemente de valores muy anteriores (o posteriores) en el tiempo; al revés que lo que pasaba en las ocasiones anteriores donde estos valores anteriores y posteriores afectaban a todas las puntuaciones. En la Figura 5.9 se representa la puntuación de anomalía cuando se estudia su puntuación-Z con ventanas móviles; junto al umbral que mejor detecta las anomalías y a su matriz de confusión



(a) (b)

Figura 5.9: (a) Puntuación de anomalía calculada para la serie 22 del conjunto A4 a través del método de las puntuaciones-Z con una ventana móvil centrada de tamaño $w = 5$ junto con las anomalías reales y el *umbral ideal* (rojo) para la detección de anomalías. (b) Matriz de confusión obtenida usando como umbral θ el umbral ideal.

Efectivamente, se puede encontrar fácilmente un umbral que separe aquellos puntos anómalos del resto.

5.2. Métodos de aprendizaje automático no supervisado

Los tres algoritmos que se han incluido en esta categoría se han implementado gracias a la librería *Sklearn* del lenguaje *Python*. Gracias a ella, con tan solo introducir una serie de hiperparámetros es posible obtener las puntuaciones de anomalía asignadas por el método para la serie seleccionada. Para estudiar al completo estos algoritmos se ha trabajado sobre la serie original normalizada, así como sobre la serie normalizada diferenciada; en la que se ha exigido la misma condición que en la ecuación (5.3), es decir, que dos puntos diferenciados consecutivos no pueden ser anomalías y poseer signo distinto.

5.2.1. Árboles de Aislamiento-*iForests*

Los parámetros previos a introducir, son el número de árboles, conocido como *n_estimators* y el número de puntos de la serie de datos que se escogen para construir los árboles *max_samples*. La librería *Sklearn*, a través de la función *decision_function* devuelve el inverso en signo de la puntuación de anomalía media transformada al rango $[-0.5, 0.5]$. Se ha elegido como puntuación de anomalía el inverso en signo de este valor, de manera que cuanto más alto sea el valor mayor

será la puntuación de anomalía, asegurando así que la puntuación representa cómo de posible es que un punto sea anómalo.

5.2.2. *One Class Support Vector Machine* (OC-SVM)

Para el algoritmo de *One Class Support Vector Machine (OC-SVM)* explicado en 3.3.2 hay que seleccionar previamente los siguientes parámetros:
ν: cota superior para la fracción de puntos anómalos y cota inferior para la fracción de número de puntos utilizados como vectores soporte, γ: coeficiente kernel necesario para algunos tipos de kernel, $coef$: término independiente para la función Kernel escogida grado d: grado de la función polinómica en caso de que la escojamos como función *Kernel*, y finalmente kernel: Especifica el tipo de función kernel que se va a utilizar en el algoritmo. Puede ser

1. lineal: $\langle x, x' \rangle$

2. polinómica: $(\gamma \langle x, x' \rangle + coef)^d$

3. base radial: $e^{-\gamma \|x - x'\|}$

4. sigmoidal: $tanh\,(\gamma \langle x, x' \rangle + coef)$

La función de dicha librería *decision_function* nos devuelve la distancia ortogonal al hiperplano de separación; lo que utilizamos como puntuación de anomalía. Para el algoritmo de *Sklearn*, una distancia positiva corresponde con un punto no-anómalo, mientras que una distancia negativa corresponde con un punto anómalo. En este trabajo se utiliza también esta distancia ortogonal como puntuación de anomalía, pero multiplicada por -1 para así que sea coherente con las definiciones previas de puntuación de anomalías, donde se representaba el grado que tenía un punto de ser un punto anómalo.

5.2.3. Factor Anómalo Local-*LOF*

Los parámetros a introducir previamente son el número de vecinos *n_neighbors* y el tipo métrica para calcular las distancias *metric*. De nuevo, la función *decision_function* devuelve el inverso de la puntuación de anomalía, en este caso el valor LOF. Se utilizará el inverso en signo de este valor como puntuación de anomalía para representar que cuanto mayor es la puntuación, más probable es que el punto sea anómalo.

La puntuación de anomalía asignada por estos métodos difiere de la serie original. En la Figura 5.10 se representa la serie número 95 del conjunto de datos A2, así como las tres puntuaciones de anomalía asignadas por los métodos.

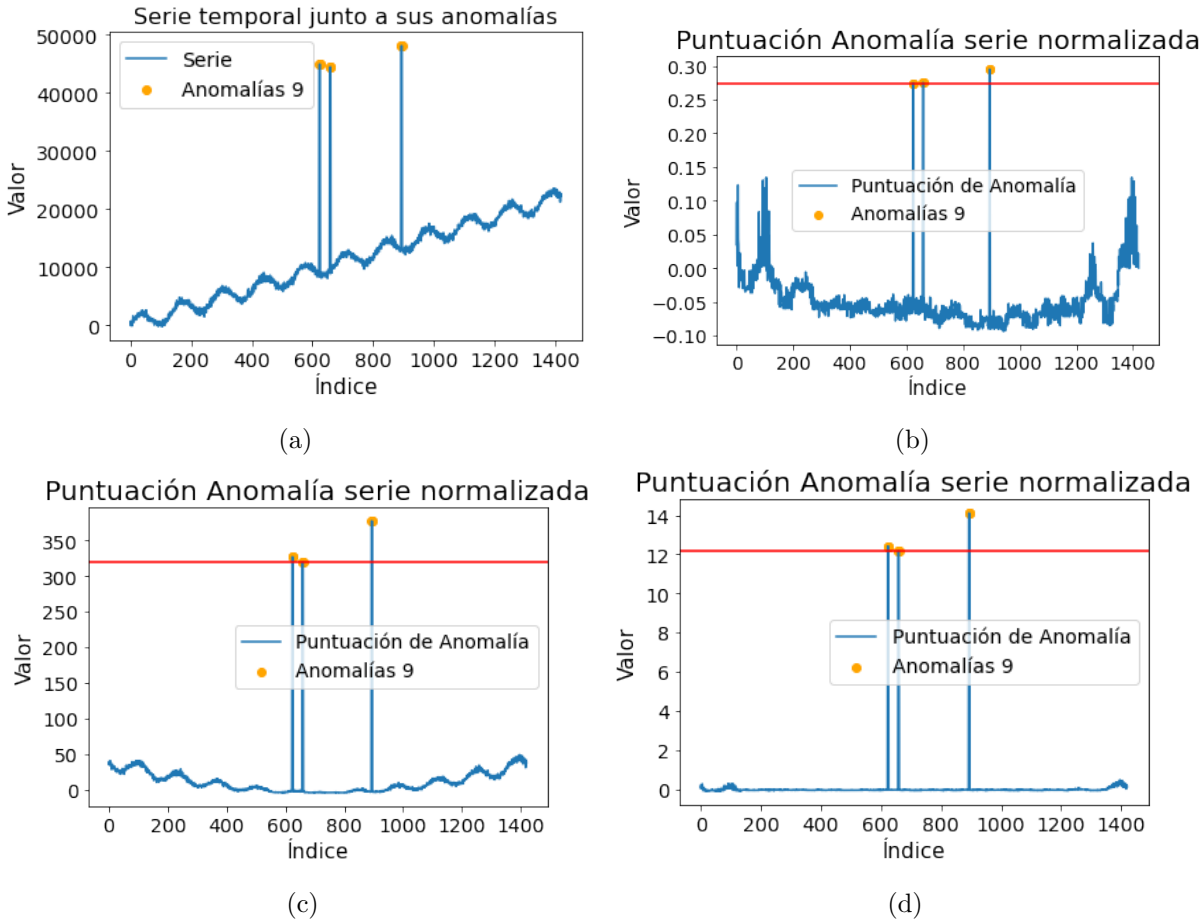
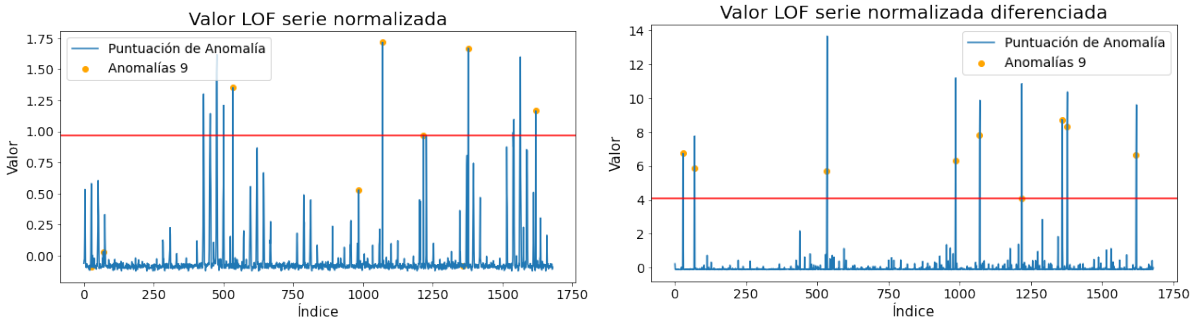


Figura 5.10: (a) Serie temporal número 95 del conjunto A2 junto a sus anomalías. (b) Puntuación de anomalía obtenida por el algoritmo *Isolation Forest* junto a las anomalías reales y el *umbral ideal* (rojo). (c) Puntuación de anomalía obtenida por el algoritmo *One class SVM* junto a las anomalías reales y el *umbral ideal* (rojo). (d) Puntuación de anomalía obtenida por el algoritmo *Local Outlier Factor* junto a las anomalías reales y el *umbral ideal* (rojo).

Sin embargo, estos métodos de detección de anomalías no se desarrollaron para series temporales en los que los datos se encuentren correlados con el pasado; por ello, cuando las dependencias temporales de la serie son muy marcadas, los algoritmos no asignan de manera efectiva la puntuación de anomalía. Un ejemplo es la serie número 89 del conjunto A4 que se representó en la Figura 5.4. En ella los tres métodos fallan al calcular la puntuación de anomalía para cada punto debido al carácter temporal de la serie. Es por ello que es necesario aplicar métodos que transformen la serie temporal haciéndola estacionaria. En la Figura 5.11 se representa la puntuación de anomalía del conjunto A4 calculada por el método LOF para la serie temporal número 89 y la serie temporal diferenciada; junto a sus anomalías y el umbral ideal. Claramente, el cálculo de la puntuación de anomalía es más óptima sobre la serie diferenciada, pues el umbral óptimo separa de manera absoluta los puntos anómalos, a diferencia de la serie no diferenciada, donde la puntuación de anomalía calculada no es válida para separar las anomalías. Esto último es clave, pues indica que no se está asociando las puntuaciones más altas a las anomalías. Además, aunque en la Figura 5.11 (b) se observan puntos no-anómalos por encima del umbral, estos puntos no van a ser clasificados como anomalías, pues se impone la condición

desarrollada en este trabajo y explicada en la ecuación (5.3) para tratar la detección de anomalías con series diferenciadas, en la que si dos puntos consecutivos de la serie diferenciada son de signo distinto y superan el umbral θ, solamente será clasificado como punto anómalo el primero de ellos.



(a) (b)

Figura 5.11: (a) Puntuación de Anomalía para la serie 89 del conjunto A4 junto a las anomalías y el *umbral ideal*. (b) Puntuación de Anomalía para la serie 89 del conjunto A4 diferenciada junto a las anomalías y el *umbral ideal*. Hiperparámetros: *metric*: euclidean, *n_neighbors*:100

5.3. Métodos basados en la predicción de la serie

La acción de predecir valores en series temporales, conocido como *forecasting* es un problema ampliamente tratado en la literatura. En [40] se explican y revisan diferentes algoritmos para la predicción de series temporales.

Debido a la autocorrelación de las series temporales, no es posible utilizar modelos clásicos de regresión como regresión lineal. Es necesario utilizar métodos capaces de modelar y parametrizar las diferentes relaciones temporales entre los datos. Algunos ejemplos sencillos son los procesos estudiados en 2.2; sin embargo, cada problema necesita ser estudiado independientemente, por lo que a menudo son necesarios modelos paramétricos mucho más complejos, con un mayor número de parámetros, frecuentemente no lineales.

La llegada del aprendizaje automático y aprendizaje profundo supone una revolución en la predicción de series temporales, debido a que estos modelos son capaces de tratar con datos temporales y sus correlaciones. Esto supone la construcción de modelos complejos de forma automática a través de los datos, evitando tener que comprender a fondo todas las dependencias temporales de la serie y modelarlas matemáticamente. Este tipo de modelos se conocen vulgarmente *modelos de caja negra*, pues se conocen los datos de entrada y los datos de salida, pero los mecanismos para llegar de uno a otro son complicados de interpretar debido al gran número de operaciones internas que suceden en ellos, haciendo muy complicado entender en qué medida afecta cada variable a la decisión final

Para que los modelos sean capaces de aprender de manera efectiva, es necesario utilizar los

datos pasados como variables predictoras del siguiente instante. Para ello, se utilizan las ventanas móviles explicadas en 2.2.3. En concreto, ventanas asimétricas: supongamos que disponemos de una serie temporal univariante de longitud T: $\mathcal{D} = \{x_t \mid x_t \in \mathbb{R}, t \in \{1, ..., T\}\}$ y escogemos un tamaño de ventana $w \in \mathbb{R}$ con $0 < w < T$; el algoritmo de predicción utilizaría los w valores de la ventana $\{x_{t-w+1}, x_{t-w+2}, ..., x_t\}$ como variables para predecir el valor siguiente $\hat{x}_{t+1} = f(x_{t-w+1}, x_{t-w+2}, ..., x_t)$ siendo f la función utilizada por el modelo para predecir un instante temporal a partir de sus w instantes anteriores.
Los algoritmos de aprendizaje automático y aprendizaje profundo se entrenan de manera supervisada para minimizar el error de predicción:

$$L = \sum_{i=w}^{T-1} (e_{i+1})^2 = \sum_{i=w}^{T-1} (\hat{x}_{i+1} - x_{i+1})^2 = \sum_{i=w}^{T-1} (f(x_{i-w+1}, x_{i-w+2}, ..., x_i) - x_{i+1})^2 \tag{5.6}$$

si se ha escogido como función de pérdida el error cuadrático medio; o

$$L = \sum_{i=w}^{T-1} \|e_{i+1}\| = \sum_{i=w}^{T-1} \|\hat{x}_{i+1} - x_{i+1}\| = \sum_{i=w}^{T-1} \|f(x_{i-w+1}, x_{i-w+2}, ..., x_i) - x_{i+1}\| \tag{5.7}$$

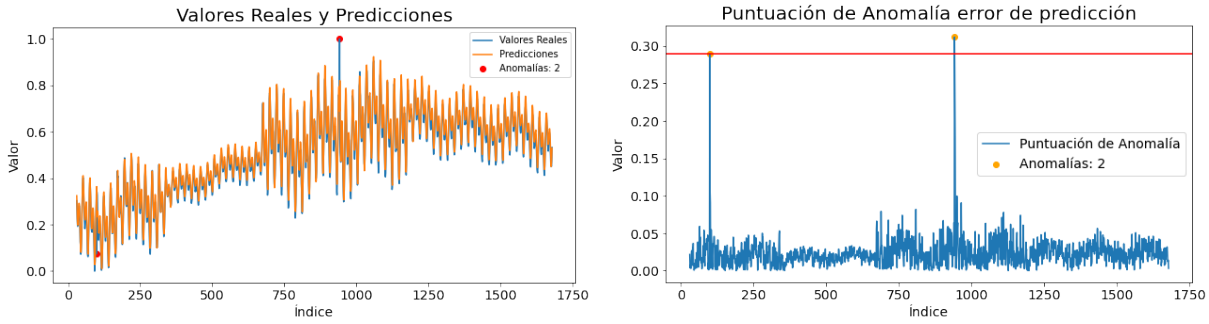
si se ha escogido como función de pérdida el error medio absoluto (no aparecen divididos por el tamaño de la muestra, pues es independiente para su minimización).
De esta manera, volviendo a la ventana de tamaño w, el modelo trataría de ajustar el valor x_{w+1} a partir de $\{x_1, x_2, ..., x_{w-1}, x_w\}$. A continuación, haría lo mismo con x_{w+2} a partir de $\{x_2, x_3, ..., x_w, x_{w+1}\}$ corrigiendo el ajuste tras comprobar el error en la predicción; y así de manera sucesiva hasta que la ventana móvil llegase a T, es decir, cuando predijese x_T a partir de $\{x_{T-w+1}, x_{T-w+2}, ..., x_T\}$.

El problema puede ser visto como un problema de regresión. Para llevar a cabo la detección de anomalías, asociamos el error en la predicción $e_i = \|\hat{x}_i - x_i\|$ a la puntuación de anomalía bajo el supuesto de que el modelo ha sido capaz de capturar las complicadas dependencias temporales de la serie y, por tanto, aquellos puntos con un mayor error de predicción son aquellos que se desvían en mayor medida del comportamiento normal predicho por el modelo.

En este trabajo se estudian cinco algoritmos lo suficientemente robustos como para ser capaces de construir un modelo de regresión válido para series temporales. En concreto, se ha estudiado un modelo de aprendizaje profundo XGBoost 3.3.4 basado en el ensamblaje de regresores de tipo árbol, y cuatro modelos de aprendizaje profundo basados en arquitecturas de redes recurrentes descritas en 3.4.2

Para explicar el funcionamiento experimental se utiliza como ejemplo la serie número 96 del conjunto de datos $A4$. Esta serie temporal posee dos anomalías solamente. La primera representa una anomalía contextual, la cual necesita de los puntos cercanos a ella para entender su comportamiento anómalo. La segunda representa una anomalía puntual cuyo valor es anómalo, pues se aleja en gran medida de la media de la distribución de la serie, así como de la distribución de una ventana centrada en él.
Al reconstruir la serie a través de los valores predichos se deberían obtener dos puntuaciones de

anomalías muy superiores al resto, correspondientes con dichos punto anómalos. Esto significaría que es en estos puntos donde el algoritmo falla en mayor medida al predecir los valores y, por tanto, más se alejan del comportamiento 'normal' aprendido por dicho algoritmo.
En la Figura 5.12 (a) se representa el valor original de la serie normalizada número 96 del conjunto A4, junto al valor predicho a través de redes LSTM. En la Figura 5.12 (b) se representa la puntuación de anomalía (valor absoluto de la diferencia entre predicciones y observaciones). Efectivamente, los dos puntos con mayor puntuación corresponden con las dos anomalías.



(a) (b)

Figura 5.12: (a) Serie temporal número 96 del conjunto A4 junto a la serie predicha por las redes LSTM y las anomalías reales. (b) Puntuación de anomalía obtenida como la diferencia en valor absoluto entre el valor predicho y el valor real, junto a las anomalías reales y el *umbral ideal* para la puntuación calculada.

5.4. Métodos basados en la reconstrucción de la serie

Existen diferentes técnicas para conseguir reducir la dimensionalidad de los datos, mapeando los datos originales a un nuevo espacio de variables menos correladas que el espacio original.
La asunción principal de este tipo de técnicas de detección de anomalías es que la distribución de los datos no anómalos y los datos anómalos difieren en gran medida en este nuevo espacio de variables, de manera que al proyectar de nuevo la serie al espacio original, aquellos puntos que se diferencien más con los originales representan aquellos datos anómalos.
Como se ha mencionado anteriormente, los datos pertenecientes a una serie temporal están correlados temporalmente entre ellos. Esto significa que pueden extraerse variables numéricas que expliquen y compriman estas dependencias temporales, como puede ser las componentes de tendencia o las de estacionalidad.

Igual que en el caso de la detección de anomalías a través de la predicción de la serie, este método requiere transformar el conjunto de datos para utilizar sus valores pasados como variables. Esta transformación se hace igualmente a través de ventanas móviles asimétricas.
En este caso el error de reconstrucción para cada una de las ventanas es la suma de los errores individuales al cuadrado. Supóngase de nuevo una serie temporal unidimensional de longitud T: $\mathcal{D} = \{x_t \mid x_t \in \mathbb{R}, t \in \{1, ..., T\}\}$, y un tamaño de ventana $w \in \mathbb{R}$ con $0 < w \leq T$. Sea f la función que comprime la serie en el nuevo espacio reducido y la reconstruye de nuevo al espacio original; sea $\hat{\mathbf{x}}_\mathbf{t}$ el vector que representa la ventana asimétrica en el instante t reconstruida:

$\hat{\mathbf{x}}_t = (\hat{x}_{t-w+1}, \hat{x}_{t-w+2}, ..., \hat{x}_{t-1}, \hat{x}_t)$ y $\mathbf{x_t}$ el vector de la ventana asimétrica en el instante t original $(x_{t-w+1}, x_{t-w+2}, ..., x_{t-1}, x_t)$; el error al cuadrado asociado a esa ventana viene dado por

$$e_t^2 = \frac{1}{w} \sum_{i=t-w+1}^{t} (x_i - \hat{x}_i)^2 \tag{5.8}$$

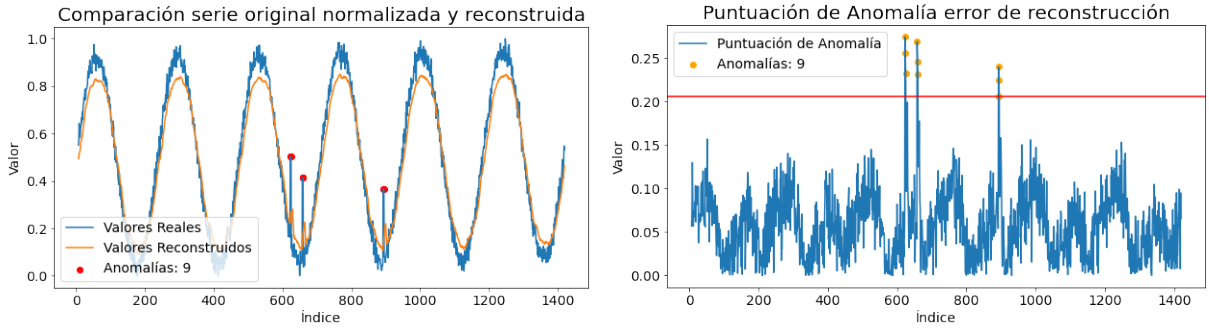
Este error que representa el promedio de las diferencias al cuadrado entre la ventana y la ventana reconstruida es el que se asocia como puntuación de anomalía, tal y como se hace en trabajos similares [41]. Esta manera de computar el error de anomalía tiene ventajas e inconvenientes. La ventaja más importante es que resulta mucho más fácil detectar las anomalías colectivas, pues el error será muy grande (al haber más de una anomalía en la ventana el error total aumenta). La desventaja es que para las anomalías individuales, este error contribuye igual para todas las ventanas en las que se encuentre. Esto significa que cuando el último punto de la ventana no sea anómalo, pero algún otro punto de la ventana sí que lo sea, el error será grande igualmente al sumarse todos los puntos de esta ventana.

Por ello, y a pesar de que en la literatura revisada se asocia el error de toda la ventana asimétrica $W_t = \{x_{t-w+1}, \ldots, x_t\}$ a t, en este trabajo se ha decidido calcular de la misma manera la ventana reconstruida, pero asociar la puntuación de anomalía del punto t solamente al error de reconstrucción entre el punto t y su valor reconstruido en la ventana. De esta manera, los métodos comprimen el vector de entrada original (la ventana) a un espacio reducido y reconstruyen la ventana. Finalmente, la puntuación de anomalía se asocia como la diferencia en valor absoluto entre el último punto de la ventana y el último punto de la ventana reconstruida.

$$e_t = |\mathbf{x}_t^w - \hat{\mathbf{x}}_t^w|, \tag{5.9}$$

donde $\mathbf{x}_t^w$ representa el último elemento del vector $\mathbf{x}_t$ (de longitud w).
Un ejemplo para entender esta manera de calcular la puntuación de anomalía: se dispone de la ventana de tamaño 3 $X_t = \{1, 2, 3\}$ de una serie de datos arbitraria, donde el valor de la serie original en el instante t es $x_t = 3$. El valor de la ventana reconstruida es $\hat{X}_t = \{1.01, 1.99, 3.04\}$. En este trabajo la puntuación de anomalía correspondiente al punto $x_t = 3$ se ha calculado como $\mid 3 - 3.04 \mid$ en vez de como el error cuadrático medio de la ventana. Además, los resultados apoyaron esta decisión obteniendo una mejoría considerable.
Se han estudiado dos algoritmos de aprendizaje profundo como son el *Autoencoder* y el *Autoencoder*-LSTM explicados en la sección 3.4.
Existen numerosos trabajos que utilizan *Autoencoders* para reconstruir la serie de datos y detectar anomalías basándose en ese error de reconstrucción. En [42] utilizan un *Autoencoder* entrenado con datos normales para detectar datos anormales en un subconjunto desconocido a través del error de reconstrucción de la serie. En [43] se utilizó un *Autoencoder* con arquitectura LSTM entrenado solo con datos normales de manera que comprenda los patrones relacionados con un comportamiento normal, y finalmente se reconstruyo la serie con instancias normales y anómalas utilizando el error de reconstrucción MSE como puntuación de anomalía. Una aproximación similar fue realizada en [44] donde se utilizó un conjunto de entrenamiento para establecer el umbral de detección de anomalías.
Un ejemplo del buen funcionamiento de estos métodos es la aplicación del *Autoencoder*-LSTM a la serie número 8 del conjunto de datos A2. En esta serie existen nueve anomalías (agrupadas

de dos en dos) que, a pesar de no contar con valores extremos al comparar todo el rango de valores posibles de la serie, 'rompen' con la estructura sinusoidal de la serie. En la Figura 5.13 (a) se representa la serie temporal original normalizada junto al último valor de cada una de las ventanas reconstruidas ($\hat{\mathbf{x}}^w$) de manera que coinciden siempre los índices temporales. Además, se representan las anomalías asociadas a la serie, donde puede observarse como estas anomalías corresponden con aquellos puntos donde se alejan en mayor medida los valores reconstruidos de los valores reales. En la Figura 5.13 (b) se representa la puntuación de anomalía asociada (error en la reconstrucción del último elemento de la ventana) junto a las anomalías de la serie. De esta manera se comprueba como efectivamente los puntos con una mayor puntuación de anomalía calculada por el *Autoencoder*-LSTM son aquellos que corresponden con las anomalías; por lo que el cálculo de la puntuación de anomalía que realiza el algoritmo es válida. De hecho, es posible encontrar un umbral que separe completamente las anomalías de los puntos no anómalos con la puntuación de anomalía calculada.



(a) (b)

Figura 5.13: (a) Serie temporal número 8 del conjunto A2 junto a la serie reconstruida por el *Autoencoder* y las anomalías reales. (b) Puntuación de anomalía obtenida como el error en valor absoluto de reconstrucción, junto a las anomalías reales y el *umbral ideal* para la puntuación calculada.

Ambas técnicas ajustan los pesos de sus redes neuronales según la función de perdida escogida. En este trabajo se ha optado por utilizar la función de perdida *MSE* en consonancia con el resto de algoritmos de aprendizaje profundo utilizados en el trabajo.

Capítulo 6

Resultados

En los capítulos anteriores se explicaba al lector los conceptos teóricos necesarios para llevar a cabo un problema de detección de anomalías en series temporales, mostrando ejemplos concretos del funcionamiento de la metodología desarrollada y algoritmos utilizados en el trabajo. Finalmente, en este capítulo, los distintos algoritmos explicados, así como sus variantes, se han aplicado a todas las series de cada uno de los cuatro conjuntos de datos con la intención de poder comparar con resultados numéricos el desempeño de cada método y algoritmo en los cuatro escenarios posibles que representan los conjuntos de datos utilizados en este trabajo, explicados en 4.3.

Para obtener el valor de la puntuación *F1* de cada serie temporal, una vez calculada su puntuación de anomalía, se ha utilizado el umbral ideal como umbral θ para clasificar cada punto como anómalo o no anómalo. Se valora entonces, cómo de bien asigna cada algoritmo la puntuación de anomalía. Un valor alto significa que existe un umbral capaz de separar con éxito los datos anómalos de los datos no anómalos; mientras que un valor bajo significa que el mejor umbral posible para esa puntuación de anomalía no es capaz de separar correctamente los puntos anómalos y, por tanto, el desempeño del modelo para calcular cómo de anómalo es un punto es peor que en el caso anterior.
Para llevar a cabo el análisis de los resultados en cada conjunto de datos se promedian todas las puntuaciones *F1* obtenidas para cada conjunto.

Debido a que se trata de conjuntos de datos para evaluar diferentes algoritmos, el trabajo no se ha centrado en la selección de hiperparámetros para cada una de las 367 series diferentes. En concreto, se han escogido los mismos hiperparámetros y arquitecturas para los cuatro conjuntos; excepto para la selección de ventanas móviles en el método de puntuaciones-Z con ventanas móviles, y para la tasa de aprendizaje y el número de retardos en el algoritmo de XGBoost, donde se han utilizado unos valores para los conjuntos $A1$ y $A2$, y otros para los conjuntos $A3$ y $A4$.
La lista de hiperparámetros y arquitecturas utilizadas para cada método se describe en el apéndice A.

6.1. Métodos estadísticos

En esta sección se analizan los resultados para los tres algoritmos pertenecientes al grupo de métodos estadísticos, denominados: *Puntuación-Z* (valor absoluto del *Z-score*), *Puntuación-Z diferenciada* (valor absoluto del *Z-score* de la serie diferenciada) y *Puntuación-Z móvil* (valor absoluto del *Z-score* utilizando ventanas móviles).
En la Tabla 6.1 se muestran los resultados para los tres algoritmos pertenecientes a métodos estadísticos.

Tabla 6.1: Puntuaciones *F1* promedio obtenidas para los tres algoritmos estadísticos explicados. Se utiliza el *umbral ideal* como umbral θ para la detección de anomalías.

Conjunto de datos	**Puntuación-Z**	**Puntuación-Z diferenciada**	**Puntuación-Z móvil**
A1	**0.761**	0.464	0.622
A2	0.764	0.634	**0.975**
A3	0.348	**0.942**	0.894
A4	0.273	0.778	**0.855**

Se observa como para el conjunto de datos $A1$ la mejor técnica obtenida es utilizar el valor absoluto de sus puntuaciones-Z como puntuación de anomalía. Esto se debe a la gran magnitud que tienen estás anomalías, de manera que su desviación respecto de la media es muy alta y son fácilmente identificables. No es el caso para los conjuntos A3 y A4, en los que las anomalías son anomalías contextuales. En estas series, en las que las anomalías poseen unos valores no extremos y del mismo orden de magnitud que los valores no anómalos, no es eficiente utilizar las puntuaciones-Z sobre la serie original, pues la variación de la media y la desviación estándar a lo largo del tiempo deben ser tenidas en cuenta para cuantificar el carácter anómalo de un punto.
Sin embargo, para el conjunto de datos A3 y A4, los métodos de las puntuaciones-Z absolutas aplicadas sobre la serie diferenciada, así como las puntuaciones-Z absolutas obtenidas con ventanas móviles, superan con creces al método de las puntuaciones-Z sobre la serie original. Esto se debe a que las anomalías de estos conjuntos de datos no son valores extremos para la serie global, sino que lo son en comparación con sus puntos vecinos (anomalías contextuales). De esta manera, al diferenciar la serie se encuentran las mayores diferencias con respecto a su vecino anterior, al igual que al estudiar su puntuación-Z por ventanas se asigna una puntuación de anomalía en torno a ese vecindario que recoge la ventana; y no en torno a la serie completa en la que las anomalías no son valores extremos. Destaca el uso de la ecuación (5.3) al utilizar las puntuaciones-Z absolutas sobre la serie diferenciada, donde el cambio en las métricas era notable para los conjuntos A2, A3 y A4 al exigir la condición de que dos puntos consecutivos de la serie diferenciada de signo contrario no podían ser anómalos los dos. La comparación entre las puntuaciones *F1* obtenidas al aplicar dicha condición y no aplicarla pueden encontrarse en el apéndice B.1.1.
Por ello, se observa como para las series temporales cuyas anomalías se correspondan con valores extremos de una magnitud may

6.2. Métodos de aprendizaje automático no supervisado

En esta sección se analizan los resultados obtenidos por los métodos de aprendizaje automático no supervisado denominados: *iForest* (árboles de aislamiento), *OC-SVM* (*One Class SVM*) y *LOF* (Factor anómalo local). Estos algoritmos se han aplicado tanto sobre la serie original normalizada, como sobre la serie normalizada y diferenciada.
Los resultados pueden observarse en Tabla 6.2:

Tabla 6.2: Puntuación *F1* promedio obtenida para los tres métodos de aprendizaje automático no supervisado, utilizando el *umbral ideal* como umbral θ. Los algoritmos se han aplicado sobre las series originales y las series diferenciadas.

Conjunto de datos	**iForest**	**iForest diferenciado**	**OC-SVM**	**OC-SVM diferenciado**	**LOF**	**LOF diferenciado**
A1	0.720	0.461	**0.759**	0.464	0.679	0.444
A2	0.742	0.627	**0.764**	0.634	0.758	0.630
A3	0.346	0.926	0.348	**0.942**	0.345	0.916
A4	0.288	**0.787**	0.273	0.777	0.283	0.774

Los tres modelos tienen un desempeño similar. Para el conjunto de datos A1 funcionan mejor los algoritmos aplicados sobre la serie sin diferenciar, pues se trata de anomalías colectivas de largos periodos, tal que al diferenciar la serie pierden su carácter anómalo debido a que las diferencias entre dos puntos anómalos de estas regiones son de un orden parecido a las diferencias entre dos puntos no anómalos.

Además, como se ha explicado anteriormente, en las series temporales del conjunto A1, las regiones anómalas poseen unos órdenes de magnitud superiores a las regiones no anómalas, por lo que estas anomalías son independientes de las variaciones temporales de la serie temporal, es decir, el cambio a lo largo del tiempo de la media y desviación estándar no son comparables a los valores pertenecientes a las regiones anómalas. Por este motivo funcionan tan bien los algoritmos de aprendizaje automático no supervisado sobre la serie original normalizada.
El buen funcionamiento de los algoritmos aplicados sobre la serie diferenciada se hace especialmente patente en los conjuntos A3 y A4. En ellos, las anomalías son anomalías contextuales cuyo valor numérico es del mismo orden que el de los puntos no anómalos; por lo que es necesario tratar de estacionalizar la serie, reduciendo sus dependencias con valores anteriores. Al diferenciar y eliminar la autocorrelación de los datos, los tres algoritmos de detección de anomalías obtienen puntuaciones *F1* muy buenas, especialmente en el caso del conjunto A3, más aún al comparar estas métricas con las obtenidas por los métodos aplicados a la serie sin diferenciar. De nuevo, al trabajar con la serie diferenciada se ha impuesto la condición explicada en (5.3) donde si dos puntos consecutivos de la serie diferenciada y de signo contrario son clasificados como anómalos, solamente se clasifica como anómalo el primero de ellos. La mejora de los resultados al imponer esta condición para los conjuntos A2, A3 y A4 puede encontrarse en el apéndice B.1.2 donde se comparan los resultados al aplicar los algoritmos de aprendizaje automático no supervisados sobre las series diferenciadas cuando se

impone dicha condición y cuando no.

Estos algoritmos se han utilizado sobre la serie univariante, sin embargo, durante la revisión de la literatura se encontraron artículos ([45],[9]) en los que se ampliaba el número de variables, utilizando los datos anteriores como variables adicionales. De esta manera, a través de una ventana móvil asimétrica de longitud w, la serie univariante de longitud T $(x_1, x_2, x_3, ..., x_T)$ se convierte en una serie con w variables $((x_1, x_2, ..., x_w), (x_2, x_3, ..., x_{w+1}), ..., (x_{T-w+1}, x_{T-w+2}, ..., x_T))$. En este trabajo se ha estudiado también esta manera de aplicar los tres algoritmos; sin embargo, esta manera de realizar la detección de anomalías ofrecía peores resultados debido a que los algoritmos no eran lo suficientemente complejos para tratar con series con dependencias temporales en los datos, ni estaban diseñados para trabajar con variables que no eran independientes entre ellas, como es el caso. Estos resultados se muestran en el apéndice B.2

6.3. Métodos basados en la predicción de la serie

En esta sección se analiza el resultado de los algoritmos pertenecientes al método de asignación de la puntuación de anomalía como el error de predicción. Estos algoritmos son: *XGBoost* (*Extreme Gradient Boosting*), *LSTM* (*Redes Long Short Term Memory*), *GRU* (*Redes Gated Recurrent Unit*), *BI-LSTM* (*Redes Bidirectional LSTM*) y *CNN-LSTM* (*Convolutional Neural Networks LSTM*).
Los resultados obtenidos para la detección de anomalías a través del error de predicción de los diferentes algoritmos de regresión explicados pueden observarse en la Tabla 6.3:

Tabla 6.3: Puntuaciones *F1* promedio obtenidas para los diferentes algoritmos de predicción. Se usa el error de predicción como puntuación de anomalía, y el *umbral ideal* como umbral θ.

Conjunto de datos	**XGBoost**	**LSTM**	**GRU**	**BI-LSTM**	**CNN-LSTM**
A1	**0.704**	0.694	0.604	0.688	0.675
A2	0.899	**0.984**	0.873	0.952	0.922
A3	0.835	0.909	0.832	**0.913**	0.784
A4	0.749	0.738	0.702	**0.811**	0.675

Todos los modelos de detección de anomalías por métodos de predicción de la serie se desempeñan peor en el conjunto de datos A1. Esto es debido a que las anomalías correspondientes a este conjunto de datos eran anomalías colectivas, extendiéndose en numerosas ocasiones por largos periodos de tiempo. Los modelos de regresión tienen un mayor error de predicción en el principio y final de estos periodos anómalos. Sin embargo, dentro de estos periodos los modelos utilizan datos anómalos pertenecientes a ese periodo para predecir el siguiente, lo que supone que el modelo se adapta a este periodo anómalo obteniendo unos errores de reconstrucción del mismo orden que para los periodos no anómalos.

Para los conjuntos A2 y A3, se observa como los modelos de regresión tienen un desempeño

muy bueno, especialmente las arquitecturas LSTM y LSTM bidireccionales. El hecho de obtener puntuaciones *F1* alrededor de 0.9 significa que el mejor umbral θ para las puntuaciones de anomalía asignadas por los algoritmos pertenecientes a este método es capaz de separar de manera precisa los puntos anómalos de los no anómalos, y, por tanto, el modelo asocia correctamente las puntuaciones de anomalía más altas a las anomalías reales.

Para el conjunto A4 se obtienen peores resultados que para las series temporales de A2 y A3. Esto se debe a que las series temporales de este conjunto cuentan con cambios abruptos de tendencia, que dificultan la modelización del comportamiento normal de la serie temporal. Sin embargo, la arquitectura *BI-LSTM* es capaz de obtener buenos resultados debido al carácter bidireccional de su arquitectura.

Destaca el buen funcionamiento del algoritmo *XGBoost*, el único algoritmo que no pertenece al marco del aprendizaje profundo y es un modelo más simple que el resto, tanto en selección de arquitectura y parámetros como en tiempo computacional.

Para cada serie temporal, los modelos se entrenaron con la serie completa de datos. Una práctica común es dividir la serie en dos de manera que los modelos se entrenan sobre un conjunto de entrenamiento, y la detección de anomalías se lleva a cabo sobre el conjunto de datos restante. En este trabajo se ha considerado que esta manera es contraproducente, pues las series temporales proporcionadas tienen sus anomalías distribuidas de maneras muy diversas y cuentan con comportamientos distintos a lo largo del tiempo. Esto supone que muchas veces el entrenamiento se realizaría sobre conjuntos anómalos y significativamente distintos al conjunto restante, resultando en reconstrucciones mucho menos precisas y donde no se podría asegurar si los mayores errores de reconstrucción estaban asociados a anomalías o a un pobre desempeño del algoritmo de predicción.

Por ello, en este trabajo se ha realizado el entrenamiento de los modelos de regresión y reconstrucción sobre las series completas, sin introducir ningún tipo de información acerca de las anomalías, solo el valor completo de la serie para su reconstrucción. De todas formas, se realizó también la evaluación de los modelos de regresión y reconstrucción de la serie, dividiendo las series en conjuntos de entrenamiento y validación. Los resultados obtenidos se muestran en B.3 son similares, aunque ligeramente peores en muchas ocasiones; sin embargo, muchas de las métricas obtenidas para el conjunto de validación cuentan con escasas anomalías, por lo que se considera que la manera más robusta, por las métricas obtenidas y por la cantidad de datos disponibles es utilizar la serie completa para el entrenamiento y reconstrucción de los modelos.

6.4. Métodos basados en la reconstrucción de la serie

En esta sección se analiza el resultado de los algoritmos pertenecientes al método de asignación de la puntuación de anomalía como el error de reconstrucción de la ventana. Estos

algoritmos son: *Autoencoder* y *LSTM-Autoencoder*.
Los resultados obtenidos para los dos métodos basados en la reconstrucción de la serie se muestran en la Tabla 6.4:

Tabla 6.4: Puntuaciones *F1* obtenidas para los diferentes algoritmos de reconstrucción de la serie. Se usa el error de reconstrucción como puntuación de anomalía, y el *umbral ideal* como umbral θ.

Conjunto de datos	**Autoencoder**	**LSTM-Autoencoder**
A1	**0.727**	0.612
A2	0.864	**0.972**
A3	0.374	**0.469**
A4	0.327	**0.379**

De los resultados anteriores pueden obtenerse varias conclusiones: como se ha comentado a lo largo del trabajo, en el conjunto A1 las anomalías son de una magnitud tan grande que no es necesario conocer las dependencias temporales de la serie, de hecho, se ha comprobado en los resultados anteriores como para el conjunto A1 utilizar algoritmos concebidos para series temporales resulta en un desempeño menor que en algoritmos para conjuntos de datos independientes (la puntuación-Z absoluta, los árboles de aislamiento y el algoritmo de *One Class SVM* obtienen mejores resultados a pesar de que no tienen en cuenta el carácter temporal de la serie). Es por ello por lo que es el único conjunto de datos en el que la puntuación *F1* promedio es superior para el algoritmo del *Autoencoder* clásico que para su variante con redes LSTM.
Para el conjunto A2, las series son series temporales sintéticas con una clara evolución sinusoidal cuyas anomalías 'rompen' con el patrón sinusoidal de estas. En este caso donde sí que son relevantes las dependencias temporales de la serie, es donde se ve más claro la superioridad de la variante LSTM del Autoencoder, alcanzando puntuaciones *F1* en muchos casos perfectas debido a la combinación entre las celdas de memoria LSTM y las capas ocultas del *Autoencoder* que comprimen el espacio extrayendo el comportamiento normal de la serie.
Los conjuntos A3 y A4 es donde peores resultados se obtienen para los métodos basados en la reconstrucción de la serie. Estas series temporales tienen una mayor variabilidad y cuentan con diferentes tendencias y comportamientos periódicos a lo largo del tiempo, además sus anomalías son anomalías contextuales cuyas magnitudes son del mismo orden que los valores normales de la serie.
Debido a estas razones, el algoritmo no es capaz de entender las dependencias temporales entre los datos y sus cambios, por lo que no aprende un comportamiento 'normal' para la serie, fallando de la misma manera a la hora de reconstruir puntos anómalos que puntos no-anómalos. Por ello, la puntuación de anomalía calculada no es válida, pues no asigna las puntuaciones más altas a los puntos anómalos, siendo imposible encontrar un umbral para estas puntuaciones que separe correctamente las anomalías.

En la Tabla 6.5 se muestran los cinco algoritmos con mejor puntuación *F1* promedio para cada conjunto de datos, ordenados de mayor a menor puntuación.

Tabla 6.5: Los cinco algoritmos con mejor puntuación *F1* promedio para cada conjunto de datos.

A1	Puntuación-Z	OC-SVM	Autoencoder	iForest	XGB
A2	LSTM	Puntuación-Z móvil	LSTM-Autoencoder	BI-LSTM	CNN-LSTM
A3	Puntuación-Z diferenciada	OC-SVM diferenciado	iForest diferenciadado	LOF diferenciado	BI-LSTM
A4	Puntuación-Z móvil	BI-LSTM	iForest diferenciado	Puntuación-Z diferenciada	OC-SVM diferenciado

Puede observarse de la tabla anterior como para los cuatro conjuntos de datos, uno de los cinco mejores algoritmos es un algoritmo perteneciente a los métodos basado en la predicción de la serie; y para tres de los cuatro conjuntos de datos, el mejor algoritmo es un algoritmo perteneciente a los métodos estadísticos, mientras que para el conjunto restante, su segundo algoritmo es un algoritmo perteneciente a un método estadístico.

Además, los algoritmos clásicos de aprendizaje automático aplicados sobre la serie diferenciada ocupan también posiciones altas en la mayoría de los casos. Esto prueba que, efectivamente, utilizar la serie temporal diferenciada resulta en una mejoría sustancial de los algoritmos clásicos de aprendizaje automático no supervisado para series temporales con anomalías contextuales y marcadas dependencias temporales.

Esto puede considerarse una de las conclusiones fundamentales de este trabajo: en aquellas series temporales en las que las anomalías poseen unos valores de magnitud similar a los valores no anómalos, los algoritmos de puntuaciones-Z absolutas sobre la serie diferenciada y puntuaciones-Z con ventanas móviles (puntuación-Z móvil) obtienen resultados muy buenos comparado con el resto de algoritmos de los diferentes métodos. Además, para el caso en el que las anomalías poseen unos valores extremos, las puntuaciones-Z absolutas originales obtienen también resultados superiores al resto de los modelos. Estos modelos estadísticos, de los cuales dos se han desarrollado en este trabajo íntegramente, se basan en métodos más sencillos y rápidos a nivel computacional que los algoritmos de aprendizaje profundo utilizados, los más complejos y usados en la actualidad.
De esta manera, el plantear un problema desde un punto de vista estadístico, en vez de tratarlo de resolver con inteligencia artificial, puede devolver resultados igualmente buenos sin necesitar grandes recursos computacionales.

Respecto a los métodos que asignan la puntuación de anomalía como el error absoluto entre el valor predicho y el valor real, algoritmos complejos como las redes LSTM obtienen buenos resultados, siendo capaces de entender y modelar el comportamiento normal de la serie temporal; de manera que las anomalías coinciden con aquellos puntos donde los valores predichos se desvían en mayor medida de los valores reales. Sin embargo, para aquellas series

temporales que cuentan con anomalías colectivas que se prolongan durante grandes periodos de tiempo, estos algoritmos se adaptan al comportamiento anómalo de la serie, disminuyendo su efectividad a la hora de asignar la puntuación de anomalía como el error de predicción.

A su vez, los métodos basados en la reconstrucción de la serie ofrecen resultados negativos para su complejidad y las necesidades computacionales que requieren, siendo buenas opciones para trabajar con series cuyas anomalías son anomalías colectivas, pero una alternativa inadecuada para anomalías contextuales que no aparecen de forma colectiva.

Capítulo 7

Conclusiones

El objetivo de este trabajo consistía en evaluar diferentes métodos de detección de anomalías en diferentes series temporales univariantes. Para ello, se estudió la literatura existente, recogiendo métodos pertenecientes a técnicas estadísticas, técnicas de aprendizaje automático y técnicas de aprendizaje profundo. A su vez, la literatura referente a la minería de series temporales permitió introducir transformaciones en la serie. Estas transformaciones permitían estacionalizar las series, eliminando las dependencias entre los datos y sus valores pasados, facilitando la aplicación de técnicas sencillas de detección de anomalías.
Tras ello, se formuló matemáticamente el problema de la detección de anomalías, introduciendo los conceptos de puntuación de anomalía o *anomaly score* y umbral θ, adaptando los diferentes métodos y transformaciones de la serie a esta formulación.
Una vez establecido el problema de la detección de anomalías, y escogidos los diferentes algoritmos a utilizar, así como las variaciones desarrolladas en el trabajo, se escogió un conjunto de series temporales utilizadas en el estado del arte para probar diversos algoritmos. Este conjunto contenía diversas series temporales, reales y sintéticas, con componentes periódicas y diferentes tendencias a lo largo del tiempo; así como anomalías procedentes de los tres tipos: contextuales, puntuales y colectivas.
Gracias a este conjunto de datos, se ha podido aumentar el estado del arte actual, comparando algoritmos estadísticos, algoritmos de aprendizaje automático y algoritmos de aprendizaje profundo, pertenecientes a diferentes métodos para calcular la puntuación de anomalía.

Del análisis de los resultados puede concluirse como:

- Para series temporales en las que las anomalías poseen valores extremos para la serie, funcionan mejor los algoritmos clásicos como los métodos clásicos de aprendizaje automático no supervisado o las puntuaciones-Z absolutas.
- Cuando se aplica a series temporales con anomalías contextuales en las que las variaciones de la media y desviación estándar de la serie a lo largo del tiempo no pueden obviarse, la transformación de la serie temporal a través de la diferenciación supone una mejora del desempeño de los algoritmos de detección de anomalías clásicos como las puntuaciones-Z o los métodos de aprendizaje automático no supervisados; siendo conveniente adaptar la formulación de la detección de anomalías a la serie diferenciada para que las anomalías

individuales no supongan un aumento en el número de falsos positivos.

- El uso de ventanas móviles resulta en un aumento considerable del algoritmo referente a las puntuaciones-Z absolutas cuando se aplica sobre series cuyas anomalías son contextuales y necesitan de un tratamiento de las dependencias temporales de la serie. Sin embargo, el uso de estas ventanas móviles o retardos como variables para los algoritmos de aprendizaje automático no supervisado disminuyen el rendimiento de estos.

- Los métodos de detección de anomalías basados en la predicción de la serie obtienen grandes resultados para los cuatro conjuntos de datos debido a que son capaces de utilizar instancias pasadas para trabajar con las complicadas dependencias temporales de la serie a través de métodos no lineales. Esto implica que son capaces de ofrecer buenos resultados en los diferentes contextos explicados. Además, a diferencia de la literatura estudiada, no es necesario separar la serie temporal en un conjunto de *entrenamiento* y un conjunto de *prueba* o *validación*, ya que se utiliza la serie completa para buscar aquellos puntos en los que el comportamiento normal aprendido por el algoritmo se desvía en mayor grado del comportamiento original de la serie. De los algoritmos que calculan la puntuación de anomalía como el error de predicción, destacan las redes LSTM y las redes LSTM bidireccionales, las más costosas computacionalmente.

- Los métodos basados en la reconstrucción de la serie presentan resultados óptimos para series temporales que poseen anomalías colectivas, obteniendo un peor desempeño para series con anomalías puntuales y contextuales en los que la proyección de la serie a un espacio de dimensionalidad reducida no basta para captar el comportamiento no anómalo de la serie.

- Con una compresión adecuada de la tipología de la serie temporal, así como de las anomalías presentes en ella, el uso de métodos estadísticos desarrollados en este trabajo puede suponer resultados parecidos a métodos más complejos de aprendizaje profundo que requieren de una mayor cantidad de tiempo y recursos.

Finalmente, el trabajo se ha completado mediante el desarrollo de una aplicación web de detección de anomalías con los algoritmos y métodos explicados, válida para cualquier serie temporal univariante y que provee al usuario de buenas herramientas interactivas para la etapa de preselección y visualización de algoritmos.

Respecto a trabajos futuros, este trabajo, podría completarse utilizando arquitecturas de aprendizaje profundo más complicadas como *Autoencoders* que combinen redes convolucionales y redes LSTM; o incluso haciendo uso de los recientes modelos *transformers* [46].
Respecto a la temática, sería muy interesante ampliar el estudio de este trabajo al campo de detección de anomalías en imágenes y vídeos, mezclando datos secuenciales con datos que se encuentren correlados espacialmente, como es el caso de los píxeles de una imagen.

Bibliografía

[1] Carlos A. M., David S. F., and Ronny P. Detección automática de microcalcificaciones en una mamografía digital, usando técnicas de inteligencia artificial. *TecnoLógicas*, pages 743–756, 2013.

[2] Jie Zheng, Hongyan Cui, Xiaoqiu Li, Lingge Meng, and Tian Wang. The clustering for clients in a bank based on big data. In *2018 4th International Conference on Universal Village (UV)*, pages 1–5, 2018.

[3] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[5] Konstantin Pogorelov, Michael Riegler, Sigrun Eskeland, Thomas de Lange, Dag Johansen, Carsten Griwodz, Peter Schmidt, and Pål Halvorsen. Efficient disease detection in gastrointestinal videos – global features versus neural networks. *Multimedia Tools and Applications*, 76, 11 2017.

[6] Waleed Hilal, S. Andrew Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41, 07 2009.

[8] Keith Ord. Outliers in statistical data: V. barnett and t. lewis, 1994, 3rd edition, (john wiley & sons, chichester), 584 pp., [uk pound]55.00, isbn 0-471-93094-6. *International Journal of Forecasting*, 12:175–176, 1996.

[9] Mohammad Braei and Dr.-Ing Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. 04 2020.

[10] Colin O'Reilly, Alexander D. Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. Anomaly detection in wireless sensor networks in a non-stationary environment. *IEEE Communications Surveys & Tutorials*, 16:1413–1432, 2014.

[11] Francisco Javier López y Gerardo Sanz. Departamento de Métodos Estadísticos, Universidad de Zaragoza, marzo 2022. Material extraído de los apuntes de Cadenas de Markov en Tiempo Continuo, de la asignatura de Procesos Estocásticos y Probabilidad correspondiente al Máster en Modelización e Investigación Matemática, Estadística y Computación.

[12] Ana Carmen Cebrián. Departamento de Métodos Estadísticos, Universidad de Zaragoza, octubre 2021. Material extraído de los apuntes de la asignatura de Series Temporales correspondiente al Máster en Modelización e Investigación Matemática, Estadística y Computación.

[13] Tendencias de población en el mundo, datos y gráficos [en linea]. `https://www.epdata.es/datos/tendencias-poblacion-mundo-datos-graficos/411/`, 2022. Última vez accedido en 2022-15-11.

[14] Simon Jensen, Thomas Moeslund, and Søren Andreasen. Deep learning-based anomaly detection on x-ray images of fuel cell electrodes. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2022.

[15] Dr. Carlos E.Obero M. Curva operador-receptor (roc) – módulo de bioestadística [en linea]. `https://modulodeestadistica.wordpress.com/curva-operador-receptor-roc/`. Última vez accedido en 2022-09-01.

[16] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, 03 2015.

[17] H.D. Nguyen, K.P. Tran, S. Thomassey, and M. Hamad. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, 2021.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[19] Wo-Ruo Chen, Yong-Huan Yun, Ming Wen, Hong-Mei Lu, Zhi-Min Zhang, and Yi-Zeng Liang. Representative subset selection and outlier detection via isolation forest. *Anal. Methods*, 8:7225–7231, 2016.

[20] Nello Cristianini and John Shawe-Taylor. *References*, page 173–186. Cambridge University Press, 2000.

[21] Roemer Vlasveld. Introduction to one-class support vector machines [en linea]. `https://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines`, julio 2013. Última vez accedido en 2022-10-25.

[22] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[23] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA, 2000. Association for Computing Machinery.

[24] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[25] Federico Divina, Miguel García Torres, Francisco A. Goméz Vela, and José Luis Vázquez Noguera. A comparative study of time series forecasting methods for short term electric energy consumption prediction in smart buildings. *Energies*, 12(10), 2019.

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, page 13, 12 2014.

[27] Claude Sammut and Geoffrey I. Webb, editors. *Sequential Data*, pages 902–902. Springer US, Boston, MA, 2010.

[28] Understanding lstm networks – colah's blog [en linea]. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, agosto 2015. Última vez accedido en 2022-08-11.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[31] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[32] Anis Yazidi, Rinkaj Goyal, Aline Paes, Nicole Gruber, Nicole Gruber@ur De, and Alfred Jockisch. Are gru cells more specific and lstm cells more sensitive in motive classification of text? *Frontiers in Artificial Intelligence — www.frontiersin.org*, 1:40, 2020.

[33] Illarion Khlestov. Convolutional layers [en linea]. `https://ikhlestov.github.io/pages/machine-learning/convolutional-layers/`, 2019. Última vez accedido en 2022-10-29.

[34] Tolga Ergen and Suleyman Serdar Kozat. Unsupervised anomaly detection with lstm neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):3127–3141, 2020.

[35] Chunyong Yin, Sun Zhang, Jin Wang, and Neal N. Xiong. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(1):112–122, 2022.

[36] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications*, 91:464–471, 2018.

[37] Tanja Hagemann and Katerina Katsarou. Reconstruction-based anomaly detection for the cloud: A comparison on the yahoo! webscope s5 dataset. In *Proceedings of the 2020 4th*

International Conference on Cloud and Big Data Computing, ICCBDC '20, page 68–75, New York, NY, USA, 2020. Association for Computing Machinery.

[38] Yahoo! webscope dataset ydata-labeled-time-series-anomalies-v1_0. `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`.

[39] Usuario RAKANNIMER. Air passengers. `https://www.kaggle.com/datasets/rakannimer/air-passengers`, 2016. Última vez accedido en 2022-09-13.

[40] Yacine Ben Baccar. *Comparative Study on Time Series Forecasting Models*. PhD thesis, 10 2019.

[41] Tsatsral Amarbayasgalan, Van Huy Pham, Nipon Theera-Umpon, and Keun Ho Ryu. Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error. *Symmetry*, 12(8), 2020.

[42] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, page 4–11, New York, NY, USA, 2014. Association for Computing Machinery.

[43] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '20, page 37–45, New York, NY, USA, 2020. Association for Computing Machinery.

[44] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. Lstm-autoencoder based anomaly detection for indoor air quality time series data, 2022.

[45] Zhang Rui, Zhang Shaoyan, Lan Yang, and Jiang Jianmin. Network anomaly detection using one class support vector machine. *Lecture Notes in Engineering and Computer Science*, 2168, 03 2008.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[47] Peter Grabusts, Arkady Borisov, and Ludmila Aleksejeva. Ontology-based classification system development methodology. *Information Technology and Management Science*, 18, 12 2015.

Apéndice A

Hiperparámetros seleccionados

En este apéndice se muestra la lista de hiperparámetros seleccionados para la aplicación de cada algoritmo a las series temporales de los cuatro conjuntos de datos estudiados.

A.1. Métodos estadísticos

Solo existe un hiperparámetro para los métodos basados en puntuaciones-Z y es solo para el algoritmo de las puntuaciones-Z móviles. El parámetro es el tamaño de la ventana móvil centrada en t. Se ha escogido la ventana del conjunto $\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ que mejor puntuación *F1* obtenía (para el mejor umbral θ posible)

A.2. Métodos Aprendizaje Automático no supervisado

Tabla A.1: Tabla con los hiperparámetros seleccionados para los tres algoritmos de aprendizaje automático no supervisado

Árboles de aislamiento	*n_estimators*	200
	max_samples	0.7
	random_state	4
	contamination	auto
OC-SVM	*kernel*	rbf
	ν	0.7
	γ	0.9
LOF	*n_neighbors*	100
	algorithm	auto
	contamination	auto

A.3. Métodos basados en la predicción de la serie

:

Tabla A.2: Hiperparámetros para los métodos basados en la predicción de la serie

XGBoosting	Número de retardos	5 (A1, A2) 10 (A3, A4)
	Tasa de aprendizaje	0.002 (A1, A2) 0.01 (A3, A4)
	Métrica de evaluación	MSE
	Parámetro γ	1
	n_estimators	1000
LSTM	Función de Pérdida	MSE
	Tasa de aprendizaje	0.00009
	Número de retardos	30
	Optimizador	*Adam*
	Número de neuronas	300,300
	Tamaño del *batch*	150
	Número de *epochs*	100
GRU	Función de Pérdida	MSE
	Tasa de aprendizaje	0.00009
	Número de retardos	30
	Optimizador	*Adam*
	Número de neuronas	300,300
	Tamaño del *batch*	150
	Número de *epochs*	100
BI-LSTM	Función de Pérdida	MSE
	Tasa de aprendizaje	0.00009
	Número de retardos	30
	Optimizador	Adam
	Número de neuronas	300,300
	Tamaño del *batch*	150
	Número de *epochs*	100
CNN-LSTM	Función de Pérdida	MSE
	Tasa de aprendizaje	0.00009
	Número de retardos	16
	Optimizador	Adam
	Número de neuronas LSTM	200
	Tamaño del *batch*	150
	Número de *epochs*	100
	Filtros	128

A.4. Métodos basados en la reconstrucción de la serie

Tabla A.3: Tabla con los hiperparámetros seleccionados para los tres algoritmos basados en la reconstrucción de la serie

Autoencoder	Función de Perdida	MSE
	Número de retardos como variables	5
	Número de *epochs*	200
	Tamaño del *batch*	100
	Neuronas del estado latente	3
	Neuronas de codificación	10
	Neuronas de decodificación	10
	Función de activación para Codificación y Decodificación	*ReLu*
	Optimizador	*Adam*
	Tasa de aprendizaje	0.002
LSTM-Autoencoder	Función de Perdida	MSE
	Número de retardos como variables	10
	Número de *epochs*	500
	Tamaño del *batch*	100
	Dimensión del estado latente	8
	Neuronas de codificación	16,8
	Neuronas de decodificación	8,16
	Función de activación para Codificación y Decodificación	*ReLu*
	Optimizador	*Adam*
	Tasa de aprendizaje	0.002

Apéndice B

Resultados Adicionales

En este apéndice se muestran aquellos resultados obtenidos a lo largo del trabajo que, a pesar de ser importantes para decidir cómo aplicar los algoritmos, ya sea para imponer las condiciones de la ecuación (5.3) para las puntuaciones-Z o los algoritmos de aprendizaje automático no supervisado, si utilizar valores anteriores como variables adicionales en el caso de los algoritmos de aprendizaje automático no supervisado, o si es necesario dividir la serie en un conjunto de *train* y *test* para los métodos de detección de anomalías basados en la predicción de la serie.

B.1. Detección de anomalías en series diferenciadas

En esta sección se muestran las puntuaciones *F1* promedias obtenidas al aplicar las puntuaciones-Z y los algoritmos de aprendizaje automático no supervisado sobre las series diferenciadas, cuando se impone la condición de que dos puntos consecutivos de la serie diferenciada con signo contrario no pueden ser anómalos a la vez, considerándose solo en ese supuesto el primero de los dos puntos como punto anómalo (condición explicada en (5.3)) y cuando no se impone dicha condición.

B.1.1. Resultados para las puntuaciones-Z sobre la serie diferenciada

Tabla B.1: Puntuaciones *F1* obtenidas al utilizar el método de las puntuaciones-Z sobre la serie diferencia sin usar la condición impuesta en (5.3) (original) y al utilizarla (trabajo). Se utiliza el *umbral ideal* como umbral θ para la detección de anomalías.

Conjunto de datos	**Puntuación-Z diferenciada original**	**Puntuación-Z diferenciada trabajo**
A1	**0.476**	0.464
A2	0.537	**0.634**
A3	0.700	**0.942**
A4	0.617	**0.778**

Como puede observarse de la Tabla B.1, excepto para el conjunto A1 en el que las anomalías eran anomalías colectivas por lo que la condición impuesta en (5.3) resulta en una peor puntuación *F1* promedio, para el resto de los conjuntos la condición desarrollada en este trabajo mejora los resultados.

B.1.2. Resultados para los métodos de aprendizaje automático no supervisado sobre la serie diferenciada

Tabla B.2: Puntuaciones *F1* promedio obtenidas al utilizar los métodos de aprendizaje automático no supervisado sobre la serie diferencia sin usar la condición impuesta en (5.3) (original) y al utilizarla (trabajo). Se utiliza el *umbral ideal* como umbral θ para la detección de anomalías.

Conjunto de datos	**iForest original**	**iForest trabajo**	**OC-SVM original**	**OC-SVM trabajo**	**LOF original**	**LOF trabajo**
A1	0.470	0.461	0.476	0.464	0.459	0.444
A2	0.544	0.627	0.537	0.634	0.537	0.630
A3	0.701	0.926	0.701	0.942	0.694	0.916
A4	0.629	0.787	0.616	0.777	0.626	0.774

La conclusión es la misma que en la sección anterior: imponer la condición de que si dos puntos consecutivos de la serie diferenciada son de signo contrario, solo puede considerarse como punto anómalo el primero de ellos, significa en una mejora sustancial de las puntuaciones *F1* promedio para los conjuntos A2, A3, y A4.

B.2. Métodos de Aprendizaje Automático con retardos

En esta sección se muestran las puntuaciones *F1* promedio para cada conjunto de datos cuando se aplican los algoritmos de aprendizaje automático no supervisado con valores anteriores como variables adicionales. En este caso se muestran los resultados con 5 valores como variables adicionales.

Tabla B.3: Puntuaciones *F1* promedio obtenidas por el mejor umbral para los métodos de árboles de aislamiento (iForest), One Class-Support Vector Machine (ocsvm) y Factor Anómalo Local (LOF) al usar valores anteriores como variables adicionales

Conjunto de datos	**iForest 5 retardos**	**OC-SVM 5 retardos**	**LOF 5 retardos**
A1	0.536	0.553	0.393
A2	0.358	0.289	0.454
A3	0.141	0.191	0.308
A4	0.119	0.191	0.256

Al comparar estos resultados con los mostrados en la Tabla 6.2 se observa el peor rendimiento

de estos algoritmos al utilizar valores anteriores como variables adicionales.

Existe un problema al tratar estos tres algoritmos usando los retardos como variables adicionales: los algoritmos no son capaces de comprender las relaciones temporales entre las variables. Imaginemos el caso del árbol de aislamiento. Este elige una variable y un valor para separar las muestras según superen o no dicho valor. Sin embargo, en este caso se están utilizando los valores anteriores como variables y, por tanto, la variable número 2 de la ventana asimétrica para el instante t será la variable número 3 para la ventana asimétrica para el instante $t+1$. De esta manera, las variables no son independientes entre ellas, lo que supone en un peor desempeño de los algoritmos.

B.3. Métodos de detección de anomalías a través de la predicción de la serie. Resultados con separación en conjunto *train* y *test*

En esta sección se muestran los resultados de las puntuaciones *F1* promedio para los cuatro conjuntos de datos cuando se aplican los algoritmos pertenecientes a los métodos basados en la predicción de la serie, utilizando el 30 % de los datos para entrenar el modelo (conjunto *train*) y el resto para evaluarlo (conjunto *test*).

Tabla B.4: Puntuaciones *F1* para los métodos basados en la predicción de la serie utilizando el 30 % de la serie como entrenamiento o *train* y el resto como *test*

Conjunto de datos	**XGBoost**	**LSTM**	**GRU**	**BI-LSTM**	**CNN-LSTM**
A1	0.767	0.746	0.679	0.713	0.737
A2	0.820	0.995	0.865	0.985	0.902
A3	0.400	0.860	0.776	0.918	0.608
A4	0.418	0.746	0.700	0.791	0.554

Al comparar con la Tabla 6.3, puede observarse como los resultados son similares, especialmente para los conjuntos A1 y A2, sin embargo, para métodos como XGB, las puntuaciones *F1* para los conjuntos A3 y A4 empeoran considerablemente.

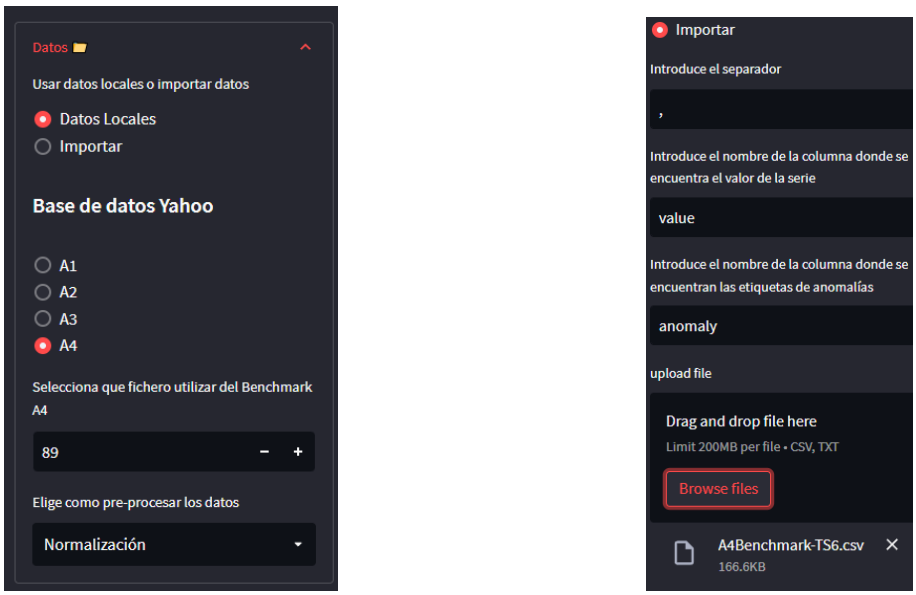
Apéndice C

Imágenes de la aplicación web

En este apéndice se explica las distintas partes que componen la aplicación web interactiva que se ha diseñado y utilizado en este trabajo.

El diseño de la aplicación consta de dos partes principales:

En el margen izquierdo se encuentran tres menús: el primero sirve para seleccionar la serie temporal a analizar (ya sea importada por el usuario o la que se encuentra por defecto) y cómo procesar sus datos, como se puede ver en la Figura C.1.



(a) (b)

Figura C.1: Desplegable *Datos*. (a) Elección de los datos por defecto. Se puede elegir cualquier serie de los cuatro conjuntos de Yahoo, junto al tipo de procesado de datos a aplicar *Normalización* o *Estandarización*. (b) Elección de una serie temporal que se encuentre en el equipo del usuario. Es necesario especificar una serie de parámetros como el nombre de las columnas donde se encuentran los valores y las anomalías, así como el separador utilizado para escribir el archivo. Una vez escogida la serie se elige qué tipo de procesado aplicar a ella.

El segundo sirve para seleccionar qué tipo de método de los cuatro explicados en el trabajo utilizar para calcular la puntuación de anomalía de la serie. Finalmente, el tercero sirve para seleccionar qué algoritmo, perteneciente al método seleccionado previamente, utilizar. Ambos desplegables pueden verse en la Figura C.2:

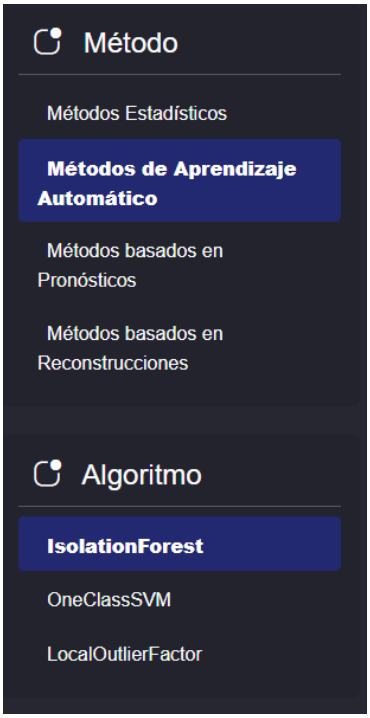
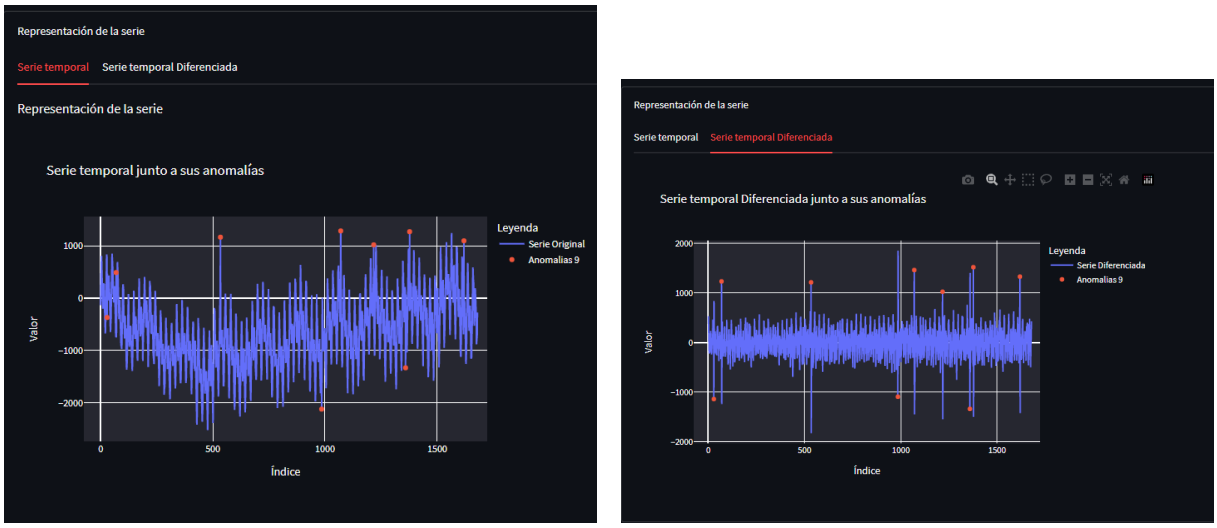


Figura C.2: Selección de método y algoritmo

La segunda parte engloba toda la página menos el margen izquierdo y se encuentra dividida en tres desplegables como puede verse en la Figura C.3:

Figura C.3: Parte de representación de la aplicación

El primero de estos desplegables sirve para la visualización de la serie original junto a sus anomalías; conteniendo dos pestañas: la primera para visualizar la serie original y la segunda para visualizar la serie diferenciada, de manera que sirve como primer paso para decidir si es mejor llevar a cabo la detección de anomalías sobre la serie original o sobre la serie diferenciada. Puede observarse en la Figura C.4:



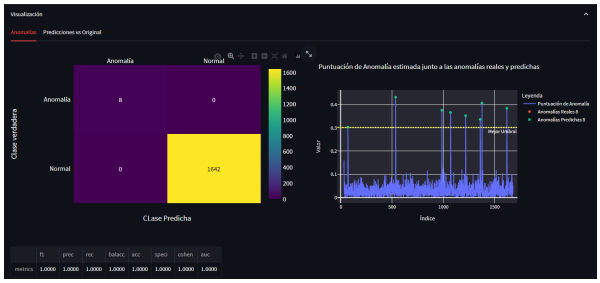
(a) (b)

Figura C.4: Desplegable *Representación de la serie*. (a) Representación de la serie temporal escogida junto a sus anomalías. (b) Representación de la serie temporal diferenciada junto a sus anomalías

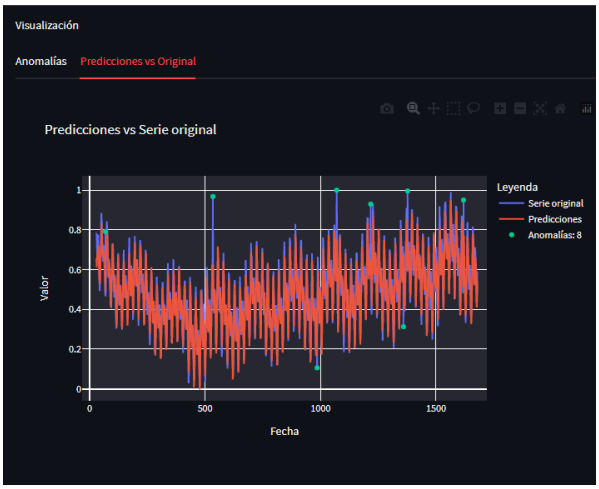
El segundo desplegable contiene los hiperparámetros del algoritmo a utilizar, siendo estos modificables por el usuario. Puede verse en la Figura C.5:

Figura C.5: Desplegable de hiperparametros para el algoritmo LSTM dentro de los métodos basados en la predicción de la serie

El tercer desplegable representa la visualización del desempeño del algoritmo.
En concreto se muestra la matriz de confusión, las métricas obtenidas para el umbral ideal (según se escoja este en el desplegable de hiperparámetros como el que maximiza la puntuación *F1* o la diferencia entre la tasa de verdaderos positivos y falsos positivos) y la representación de la puntuación de anomalía calculada por el algoritmo, junto a las anomalías reales, las anomalías predichas y el umbral seleccionado. Adicionalmente, para los métodos basados en la predicción y reconstrucción de la serie, es posible visualizar los valores originales (normalizados o estandarizados, según se hubiese elegido en el desplegable de datos) junto a las predicciones o reconstrucciones, para comparar la fiabilidad del modelo. El segundo desplegable y el tercero están relacionados, de manera que cambiar el valor de uno de los hiperparámetros modifica en tiempo real el desplegable 3 de acuerdo a los nuevos hiperparámetros. Este tercer desplegable puede verse en la Figura C.6:



(a)



(b)

Figura C.6: Desplegable *Vsiualización*. (a) Métricas para la puntuación de anomalía calculada según el mejor umbral. (b) Representación de la serie temporal junto a la serie predicha

Finalmente, en la Figura C.7, se representa una captura de pantalla de la aplicación en general.

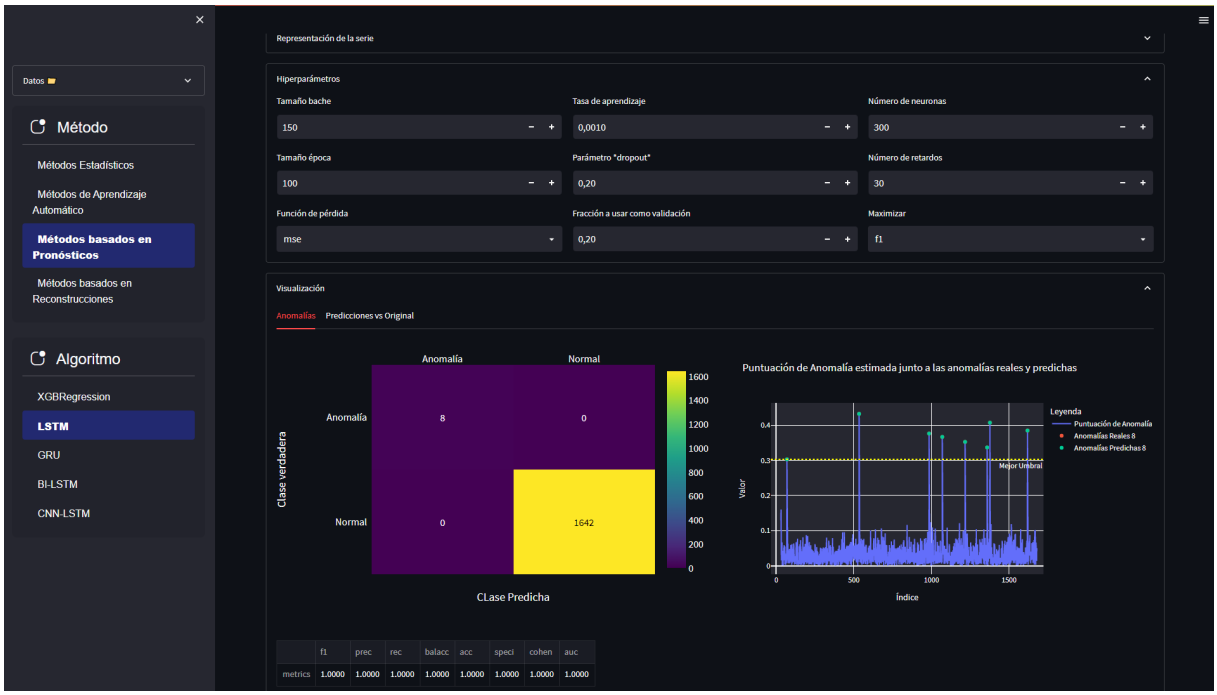


Figura C.7: Captura de la aplicación.

Apéndice D

Árboles de Regresión y clasificación

Los árboles de regresión y clasificación toman su nombre del aspecto del grafo dirigido que representa un conjunto de reglas. La idea principal es sencilla: dividir el espacio de posibles predicciones en un conjunto de regiones más simples en las que se agrupan las observaciones con valores similares. Si la variable a predecir es continua, el árbol es un árbol de regresión; mientras que si es una variable cualitativa, el árbol es un árbol de clasificación
Sea Y la variable que se quiere predecir a partir de n variables predictoras, $x_1, x_2, \ldots, x_n$ se busca establecer la relación entre Y y las variables predictoras para poder predecir Y a partir de $x_1, x_2, \ldots, x_n$. En concreto, se estudia el valor medio de la variable aleatoria Y condicionado a $x_1, \ldots, x_n$, $E[Y \mid x_1, x_2, \ldots, x_n]$, para los árboles de regresión; y la probabilidad condicional de y dado los predictores $\{x_1, x_2, \ldots, x_n\}$, $P(Y = y \mid x_1, x_2, \ldots, x_n)$.

Supongamos que tenemos un conjunto de entrenamiento $\mathcal{D} = \{\{X_i, y_i\} \mid X_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}, i \in \{1, \ldots, m\}\}$ con X_i el conjunto de variables predictoras para la i-ésima observación y_i.

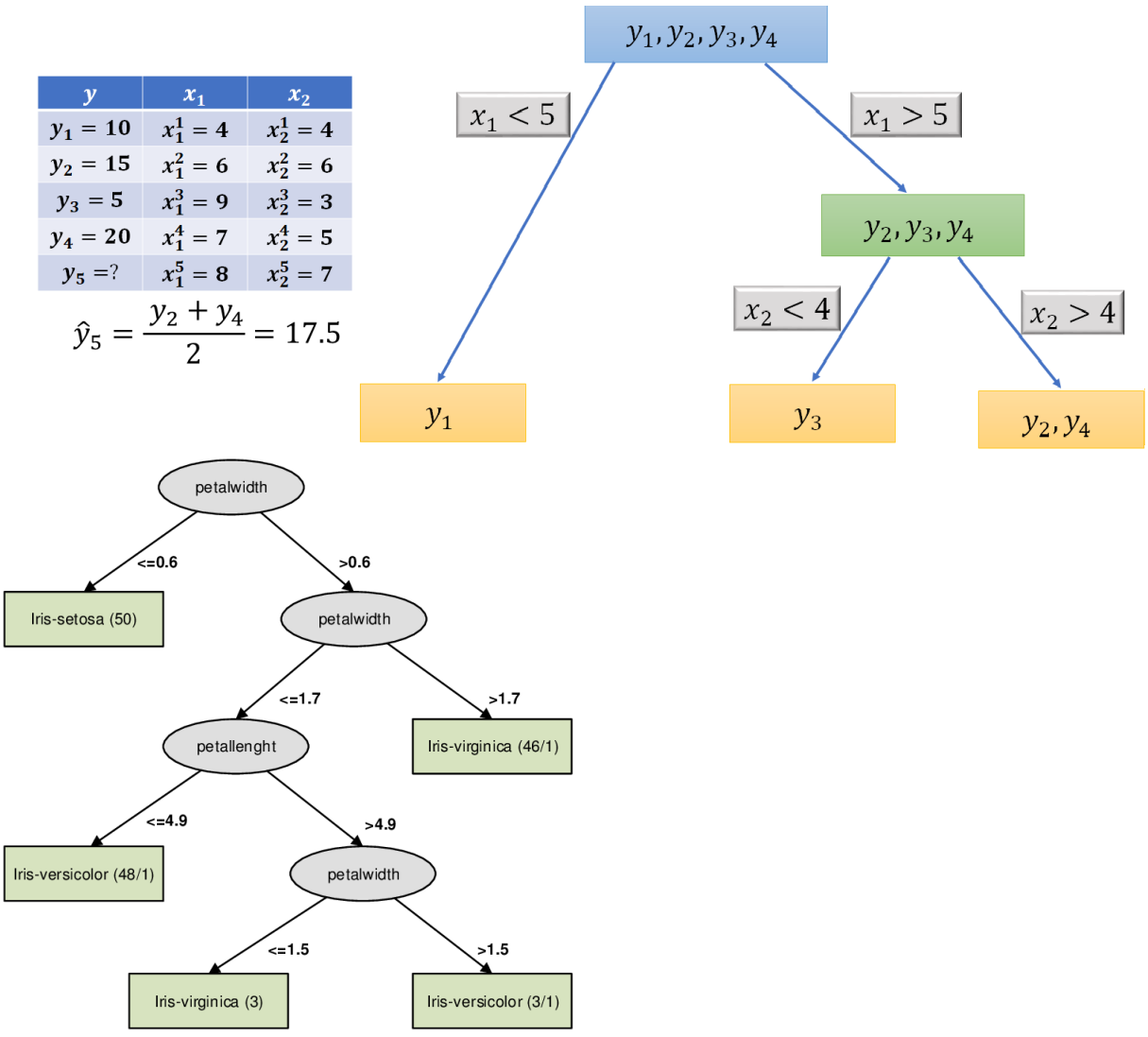
Para hacer la predicción, se realizan un conjunto de proposiciones lógicas basadas en las diferentes variables predictoras. Estas proposiciones lógicas funcionan a modo de reglas que van dividiendo el conjunto de entrenamiento en regiones cuyas observaciones poseen unos valores similares. De esta manera, dado una variable aleatoria nueva y' a predecir utilizando sus variables predictoras, se encontrará la región a la que pertenece la observación y' dentro del árbol construido de acuerdo a sus n variables predictoras. Para los árboles de regresión se utilizará la media de las observaciones del conjunto de entrenamiento que se encuentren en la misma región como predicción $\hat{y'}$. Para los árboles de clasificación, la clase asignada será la clase predominante para esa región.

Por tato, los árboles de regresión representan un conjunto de reglas sucesivas que ayudan a tomar una decisión. A esta estructura se le denomina q e indica los pasos que debe seguir un punto de la muestra para llegar a su nodo terminal. Se parte de un nodo inicial o nodo *raíz* que se va bifurcando en nodos sucesivos, llamados nodos hijos, llegando hasta los nodos terminales u *hojas* (aquellos en los que no se realizan más divisiones) el camino desde el nodo raíz hasta una hoja es lo que se denomina *rama*. Cada nodo no terminal representa una de las diferentes

proposiciones lógicas o reglas que separan las observaciones. Los nodos terminales representan regiones con observaciones de valores similares.

Estas divisiones se realizan en relación a un valor determinado para una de las variables predictoras. Existen diferentes criterios, por ejemplo, en los árboles de decisión se busca, entre los rangos de todas las variables predictoras, el valor de la división que divida de forma más homogénea el nodo. En los árboles aleatorios o *random forest*, se escoge una variable aleatoria y un valor aleatorio para esa variable, de manera que se evita el sobre-ajuste del modelo.

En la Figura D.1 se representa un ejemplo de un árbol de regresión y en la Figura D.1(b) un ejemplo de un árbol de clasificación.



y	x_1	x_2
$y_1 = 10$	$x_1^1 = 4$	$x_2^1 = 4$
$y_2 = 15$	$x_1^2 = 6$	$x_2^2 = 6$
$y_3 = 5$	$x_1^3 = 9$	$x_2^3 = 3$
$y_4 = 20$	$x_1^4 = 7$	$x_2^4 = 5$
$y_5 = ?$	$x_1^5 = 8$	$x_2^5 = 7$

Figura D.1: (a) Funcionamiento de un árbol de regresión básico (árbol de decisión). Fuente: propia. (b) Funcionamiento de un árbol de clasificación. Fuente: [47]