Trabajo Fin de Máster

Predicción de anomalías en series temporales multivariantes

Martina Basdediós Tilve

Máster en Técnicas Estadísticas

Curso 2023-2024

Propuesta de Trabajo Fin de Máster

Título en galego: Predición de anomalías en series temporais multivariantes
Título en español: Predicción de anomalías en series temporales multivariantes
English title: Anomaly prediction in multivariate time series
Modalidad: Modalidad B
Autor/a: Martina Basdediós Tilve, Universidade de Santiago de Compostela
Director/a: José Antonio Vilar Fernández, Universidade da Coruña
Tutor/a: José Jorge García Romarís, ABANCA CORPORACIÓN BANCARIA S.A.
Breve resumen del trabajo: A partir de distintas series históricas de datos relacionados con la infraestructura tecnológica de ABANCA, se busca detectar anomalías que permitan anticipar las caídas de la aplicación Banca Móvil, empleando la metodología de la predicción de anomalías en series temporales multivariantes.
Recomendaciones: Conocimientos de Python y SQL.
Otras observaciones: El alumno/a se incorporará a la Entidad en calidad de Estudiante en Prácticas. Estas prácticas están remuneradas.

Don José Antonio Vilar Fernández, Catedrático de la Universidade da Coruña, y don José Jorge García Romarís, Director de Data, Analytics y RTech de ABANCA CORPORACIÓN BANCARIA S.A., informan que el Trabajo Fin de Máster titulado

Predicción de anomalías en series temporales multivariantes

fue realizado bajo su dirección por doña Martina Basdediós Tilve para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 22 de julio de 2024.

VILAR FERNANDEZ JOSE ANTONIO - 32639957J
Firmado digitalmente por VILAR FERNANDEZ JOSE ANTONIO - 32639957J
Fecha: 2024.07.22 18:50:17 +02'00'

El director:
Don José Antonio Vilar Fernández

El tutor:
Don José Jorge García Romarís

La autora:
Doña Martina Basdediós Tilve

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Agradecimientos

En primer lugar, me gustaría agradecer a la entidad financiera ABANCA por la oportunidad de realizar mi Trabajo Fin de Máster en la empresa. En especial, agradecer a mi tutor, José Jorge García Romarís, por sus consejos y la orientación recibida durante estos meses, así como a Roberto López Rodríguez por su implicación y ayuda a lo largo de este proyecto. Agradecer también a mis compañeros, Aida, Álex, Diego, Julia, Patricia y demás miembros del equipo, por hacer de esta una experiencia aún más enriquecedora.

De igual modo, agradezco al director del proyecto, José Antonio Vilar Fernández, su apoyo y asesoramiento para la realización de este trabajo.

Finalmente, me gustaría agradecer a mi familia y, sobre todo, a mis padres y a mi hermana, por su cariño y por ser un pilar fundamental en el día a día. Dar las gracias también a mis amigos, en especial a Isabel y Álvaro, por su compañía y apoyo incondicional en este proceso.

Índice general

Resumen xi

1. Introducción 1

2. Revisión teórica y metodología 3
2.1. Series temporales multivariantes . . . 3
2.1.1. Definiciones y conceptos básicos . . . 3
2.1.2. Proceso estacionario . . . 4
2.1.3. Proceso de ruido blanco . . . 4
2.2. Tipos de anomalías . . . 5
2.3. Métodos para la detección de anomalías . . . 6
2.3.1. *Scoring*, umbral y etiquetado . . . 7
2.3.2. Clasificación de los modelos . . . 7
2.3.3. Selección del umbral . . . 10
2.4. Métricas de evaluación . . . 11
2.4.1. Clasificación . . . 11
2.4.2. *PATE* . . . 12
2.4.3. Métricas adaptadas . . . 13
2.5. Modelos empleados . . . 14
2.5.1. *Isolation Forest* . . . 14
2.5.2. *k Nearest Neighbors* . . . 16
2.5.3. *LSTM-AD* . . . 18
2.5.4. *EncDec-AD* . . . 24

3. Tratamiento de datos 29
3.1. Obtención y descripción de los datos . . . 29
3.2. Análisis exploratorio . . . 32
3.3. Preprocesamiento . . . 42

4. Resultados de los modelos empleados 43
4.1. Consideraciones generales . . . 43
4.2. Primera prueba . . . 44
4.3. Segunda prueba . . . 51

5. Conclusiones 53
5.1. Comentarios finales . . . 54

A. Conceptos complementarios 55
A.1. Análisis de componentes principales . . . 55
A.2. Árboles de decisión . . . 55
A.3. Distancia de Mahalanobis . . . 55

A.4. Distancia euclídea 56
A.5. Distancia Manhattan 56
A.6. Distribución normal multivariante 56
A.7. Error absoluto medio 56
A.8. Error cuadrático medio 56
A.9. Estimación por máxima verosimilitud 57
A.10.Maldición de la dimensionalidad 57

Bibliografía **59**

Resumen

Resumen en español

La predicción de anomalías en series temporales multivariantes es un tema que goza de gran relevancia en la actualidad, especialmente en sectores industriales y tecnológicos, en los que es crucial identificar comportamientos anómalos que permitan anticipar problemas en los sistemas monitorizados. En este trabajo, se aborda la tarea de detectar anomalías en distintas series históricas de la infraestructura tecnológica de ABANCA, con el objetivo de intentar anticipar las caídas de la aplicación Banca Móvil. En primer lugar, se realiza una revisión teórica de la metodología y los modelos existentes para la detección de anomalías en series temporales multivariantes. A continuación, se describen las distintas series históricas de datos relacionados con Banca Móvil que se van a considerar, explicando cómo han sido recopiladas. Finalmente, se presentan los resultados obtenidos para los modelos seleccionados y las conclusiones que se pueden extraer a partir del estudio realizado.

English abstract

Anomaly prediction in multivariate time series is a topic of great relevance nowadays, especially in industrial and technological sectors, where it is crucial to identify anomalous behaviour in order to anticipate problems in the monitored systems. In this work, the task of detecting anomalies in different historical series of ABANCA's technological infrastructure is addressed, with the aim of trying to anticipate crashes in the Banca Móvil app. First, a theoretical review of the existing methodology and models for the detection of anomalies in multivariate time series is carried out. Next, we describe the different time series of data related to Banca Móvil that will be considered, explaining how they have been collected. Finally, the results obtained for the selected models and the conclusions that can be drawn from the study are presented.

Capítulo 1

Introducción

Los grandes avances tecnológicos de los últimos años permiten el almacenamiento de grandes cantidades de datos provenientes de distintas áreas, como la economía, las telecomunicaciones, la medicina o la industria, entre otras. A menudo, estos datos se corresponden con observaciones de variables numéricas registradas ordenadamente a lo largo del tiempo, constituyendo así múltiples *series temporales*. A modo de ejemplo, se puede citar la evolución diaria del precio de la luz (serie *univariante*, formada por una única variable) o la monitorización continua de las constantes vitales de un paciente (serie *multivariante*, formada por varias variables). Todos aquellos métodos destinados a analizar este tipo de datos se engloban en el denominado *análisis de series temporales* (Shumway y Stoffer, 2017), cuyo principal objetivo es descubrir patrones o tendencias en los datos que permitan extraer conocimiento de los mismos. Este tipo de análisis resulta de gran utilidad en tareas como la predicción de valores futuros o la detección de anomalías en dichas series.

En un sentido abstracto, se entiende por *anomalía* a toda aquella observación o patrón en los datos que no se ajusta a una noción bien definida de comportamiento normal. Por lo tanto, la *detección de anomalías* se define como la tarea de encontrar aquellos patrones que no se ajustan al comportamiento esperado (Chandola, 2009). Sus aplicaciones prácticas son incontables: detección de fraude en el sector bancario (Thimonier *et al.*, 2023), mantenimiento predictivo en el sector industrial (Tian *et al.*, 2022), diagnóstico de enfermedades en medicina (Chauhan y Vig, 2015), detección de intrusiones en ciberseguridad (Shone *et al.*, 2018)... Existen numerosos y variados métodos de detección de anomalías dependiendo del tipo de datos que se estén analizando, del contexto o área en el que hayan sido generados y del objetivo concreto que se persiga. Por esta razón, seleccionar la mejor técnica en cada caso puede resultar una tarea compleja.

El presente trabajo se centra en el estudio de los métodos propios de la *detección de anomalías en series temporales multivariantes*. Este es un tema de gran relevancia en la actualidad, debido, entre otras cosas, al rápido desarrollo de industrias relacionadas con el Internet de las Cosas (*IoT*, por sus siglas en inglés) o el sector TI (Tecnología de la Información). En estas áreas es crucial monitorizar múltiples variables registradas desde elementos como sensores, servidores o sistemas que permitan identificar fallos internos o posibles ataques externos, con el objetivo de atajarlos a tiempo y evitar que evolucionen a problemas mayores.

El análisis de series temporales multivariantes presenta dos grandes retos o dificultades: por una parte, la dependencia secuencial inherente a los datos temporales y, por otra, la naturaleza multivariante de las series, considerando posibles correlaciones entre variables. Aunque la detección de anomalías en series multivariantes ya se había estudiado antes, muchos de los métodos existentes se han desarrollado en los últimos años gracias a los crecientes avances en técnicas de inteligencia artificial y *deep learning*, que permiten abordar este problema con elevada precisión.

La elaboración de este trabajo viene motivada por el problema de detección de anomalías propuesto por la entidad financiera ABANCA, que se explica a continuación.

Dentro de la infraestructura tecnológica de ABANCA se encuentra la aplicación *Banca Móvil*, que permite a los usuarios acceder a los servicios de la banca online de la entidad a través de sus dispositivos móviles. Para que la experiencia de los clientes sea satisfactoria, es deseable minimizar las incidencias que puedan producirse en el uso de la aplicación. Uno de los principales problemas a evitar es la caída o cese de funcionamiento, pues esto impide a los usuarios acceder a la *app* y realizar sus gestiones con normalidad. En muchos casos, estas caídas vienen provocadas por fallos en alguno o varios de los sistemas que componen la infraestructura que da soporte a la aplicación, como pueden ser los servidores y sus réplicas, las *API* o las bases de datos, entre otros. Sin embargo, debe considerarse también la posibilidad de ataques maliciosos como el *ataque de denegación de servicio*, que tiene como objetivo sobrecargar de tráfico un sistema web para interrumpir su funcionamiento normal.

El principal objetivo de este proyecto es hacer una revisión de la metodología de detección de anomalías en series temporales multivariantes y emplearla para detectar irregularidades en distintas series históricas de datos relacionados con Banca Móvil. De esta forma, sería factible identificar el mal funcionamiento de la aplicación e intentar anticipar las caídas con el mayor margen posible, siendo una herramienta de utilidad en un escenario en tiempo real. Así como no todas las anomalías detectadas se relacionan necesariamente con una caída (pueden deberse a otros factores, como se verá más adelante), no todas las caídas podrán anticiparse con un cierto margen, pues pueden ocurrir de forma casi repentina (por ejemplo, si se pierde la conexión con alguno de los servidores o bases de datos).

Con el propósito de hacer un seguimiento lo más preciso posible de la disponibilidad de Banca Móvil, se realizará un tratamiento y selección de variables de distintos canales tecnológicos del banco, con datos a nivel de minutos, que permitirán desarrollar distintos modelos de detección de anomalías. La intención es determinar, minuto a minuto, si se está produciendo una anomalía en los datos o no, de forma que se pueda prever una posible caída. Para referirse a esta tarea también se empleará la terminología *predicción de anomalías*, enfatizando así la idea de predecir o anticipar las posibles caídas de la aplicación, de ahí el título del trabajo.

Esta memoria se organiza como sigue. En el Capítulo 2, se realiza una revisión de los conceptos teóricos relacionados con las series temporales multivariantes y la detección de anomalías, así como de los modelos más utilizados dentro de este contexto. En el Capítulo 3 se presentan los datos de Banca Móvil disponibles, explicando cómo han sido recopilados y preprocesados, y proporcionando además un primer análisis de los mismos. Los principales modelos presentados en el Capítulo 2 se aplican a los datos de Banca Móvil en el Capítulo 4, comentando en detalle los resultados obtenidos y las diferencias más notables entre ellos. Finalmente, en el Capítulo 5, se establecen las principales conclusiones que se derivan de la aplicación de estos métodos, así como los avances más recientes en el campo de la detección de anomalías en series temporales multivariantes.

Capítulo 2

Revisión teórica y metodología

En este capítulo se exponen todos aquellos fundamentos teóricos que se requerirán a lo largo de este trabajo. Tras presentar brevemente los conceptos y elementos clave en el análisis de series de tiempo multivariantes, se introducen definiciones y conceptos propios del problema de detección de anomalías en este contexto. Finalmente, se revisan algunos de los métodos más utilizados para la detección de anomalías en series temporales multivariantes.

2.1. Series temporales multivariantes

En esta sección se desarrollan los elementos teóricos más relevantes del análisis de series temporales multivariantes. Para profundizar en este tema, puede consultarse Wei (2019).

2.1.1. Definiciones y conceptos básicos

Un *proceso estocástico* es un conjunto de variables aleatorias, $\{\boldsymbol{X}_t\}_{t\in C}$, definidas sobre el mismo espacio de probabilidad, donde el subíndice t hace referencia al instante en el que se observa cada variable aleatoria. De ahora en adelante, consideramos que $t \in C = \mathbb{Z}$, siendo $\mathbb{Z}$ el conjunto de los números enteros. Llamaremos *realización* de un proceso estocástico a los valores observados del mismo.

Una *serie de tiempo* o *serie temporal* puede definirse como una colección de observaciones de una variable numérica X tomadas secuencialmente a lo largo del tiempo. Más formalmente, una serie de tiempo es una realización de un cierto proceso estocástico. En adelante se asume que estas observaciones han sido tomadas en instantes de tiempo equiespaciados. Dependiendo de la dimensión de la variable X, se distingue entre series de tiempo *univariantes*, si X es una variable unidimensional, o series de tiempo *multivariantes*, si X es una variable multidimensional (i.e., un vector de variables).

A lo largo de la memoria, una serie de tiempo m-dimensional se denotará $\boldsymbol{X}_t = [X_{1t}, X_{2t}, \ldots, X_{mt}]'$, $m \in \mathbb{N},\ t \in \mathbb{Z}$, donde $X_{it},\ i = 1, \ldots, m$, es la componente i-ésima en el tiempo t. Se entenderá que la serie es multivariante cuando $m \geq 2$. Para cada i y cada t, X_{it} es una variable aleatoria, por lo que cada componente de la serie es, a su vez, una serie de tiempo univariante.

Una de las principales características de las series temporales multivariantes es que sus observaciones dependen tanto de la componente i-ésima como del tiempo t. Esto quiere decir que las observaciones entre dos variables X_{it} y X_{js} pueden estar correladas cuando $i \neq j$, haciendo referencia t y s al mismo instante de tiempo o no. Al no poder asumir la independencia entre observaciones, muchos de los métodos estadísticos clásicos no serán directamente aplicables en este contexto, debiendo recurrir, por tanto, a la teoría desarrollada en el análisis de series temporales.

Un proceso estocástico m-dimensional $\{\boldsymbol{X}_t\}_{t\in\mathbb{Z}}$ está caracterizado por:

- El *vector de medias* en el instante t, denotado por $\mathbb{E}(\boldsymbol{X}_t) = \boldsymbol{\mu}_t = [\mu_{1t}, \ldots, \mu_{mt}]'$, que recoge la media para cada variable X_{it} del proceso.

- La *matriz de covarianzas*, que se denota por $\boldsymbol{\Gamma}(t,s) = \mathrm{Cov}(\boldsymbol{X}_t, \boldsymbol{X}_s) = \mathbb{E}[(\boldsymbol{X}_t - \boldsymbol{\mu}_t)(\boldsymbol{X}_s - \boldsymbol{\mu}_s)']$. Así, cada elemento de la matriz puede escribirse como $\gamma_{ij}(t,s) = \mathbb{E}[(X_{it} - \mu_{it})(X_{js} - \mu_{js})]$, con $i = 1, \ldots, m$ y $j = 1, \ldots, m$. Los elementos de la diagonal, donde $i = j$, se corresponden con las *autocovarianzas* para cada una de las componentes i-ésimas del proceso. Fuera de la diagonal, encontramos las *covarianzas cruzadas* de las variables X_{it} y X_{js}. Estas covarianzas son una medida de la dependencia lineal entre dichas variables. Se tiene que $\boldsymbol{\Gamma}(t,t)$ es la matriz de varianzas-covarianzas del proceso en el instante t.

- La *matriz de correlaciones*, que se define como $\boldsymbol{\rho}(t,s) = \boldsymbol{D}(s)^{-1/2}\boldsymbol{\Gamma}(t,s)\boldsymbol{D}(t)^{-1/2} = [\rho_{ij}(t,s)]$, con $i = 1, \ldots, m$ y $j = 1, \ldots, m$, siendo $\boldsymbol{D}(t) = diag[\gamma_{11}(t,t), \ldots, \gamma_{mm}(t,t)]$ la matriz diagonal de varianzas de las componentes i-ésimas del proceso en el instante t, y $\boldsymbol{D}(s)$ lo análogo para el tiempo s. De forma similar a lo que ocurría en la matriz de covarianzas, los elementos de la diagonal serán en este caso las *autocorrelaciones* de las componentes i-ésimas, mientras que fuera de la diagonal se tienen las *correlaciones cruzadas* entre las variables X_{it} y X_{js}. Estas correlaciones son también una medida de la dependencia lineal entre las distintas variables, pero toman valores únicamente entre -1 y 1. Esto último permite una sencilla interpretación: un valor próximo a 1 indica una fuerte correlación positiva, mientras que un valor próximo a -1 indica una fuerte correlación negativa; por otra parte, una correlación cercana a 0 denota la ausencia de relación lineal entre las variables implicadas.

2.1.2. Proceso estacionario

Un proceso estocástico m-dimensional $\{\boldsymbol{X}_t\}$ se dice *débilmente estacionario* o simplemente *estacionario* si se cumplen las siguientes condiciones:

1. $\mathbb{E}(X_{it}) = \mu_i, \ \forall t$, es decir, la media es constante a lo largo del tiempo para cada $i = 1, \ldots, m$.

2. Las funciones que componen la matriz de covarianzas dependen únicamente de la diferencia de tiempos $s - t = k$, a la que llamaremos *retardo*. De esta forma, $\Gamma(t, t+k) = \Gamma(k), \ \forall t, k$.

Esto implica, además, que la varianza de cada componente i-ésima es invariante en el tiempo. La matriz de varianzas-covarianzas en el instante t se denotará ahora por $\Gamma(0)$. Las correlaciones podrán escribirse como $\boldsymbol{\rho}(k) = \boldsymbol{D}^{-1/2}\boldsymbol{\Gamma}(k)\boldsymbol{D}^{-1/2} = [\rho_{ij}(k)]$, siendo $\boldsymbol{D} = diag[\gamma_{11}(0), \ldots, \gamma_{mm}(0)]$.

Estas condiciones son equivalentes a decir que un proceso m-dimensional $\{\boldsymbol{X}_t\}$ es estacionario si cada una de las series que lo componen proviene de un proceso estacionario univariante y sus dos primeros momentos son invariantes en el tiempo.

2.1.3. Proceso de ruido blanco

Un proceso m-dimensional $\{\boldsymbol{a}_t\}$ será un *proceso de ruido blanco* con vector de medias $\mathbf{0}$ y matriz de covarianzas $\boldsymbol{\Sigma}$ si:

$$\mathbb{E}[\boldsymbol{a}_t\boldsymbol{a}_{t+k}] = \begin{cases} \boldsymbol{\Sigma}, \text{ si } k = 0, \\ \mathbf{0}, \text{ si } k \neq 0, \end{cases}$$

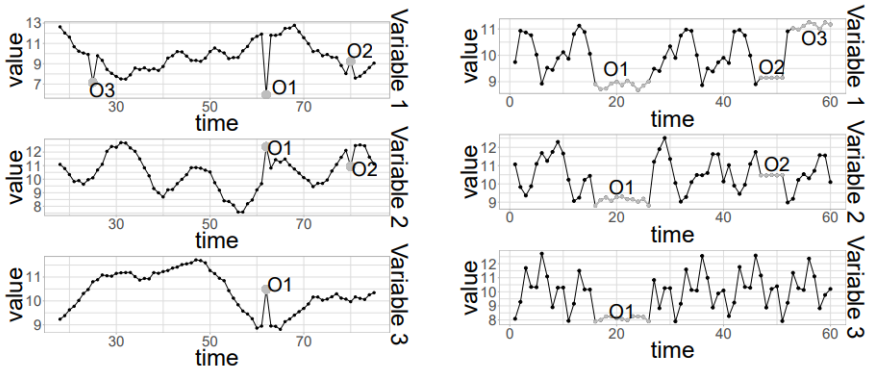
donde $\boldsymbol{\Sigma}$ es una matriz $m \times m$, simétrica y definida positiva.

2.2. Tipos de anomalías

Como ya se ha comentado, en un sentido amplio, todas aquellas observaciones o patrones que no se ajusten al comportamiento esperado de los datos se denominarán anomalías. En el contexto de series temporales multivariantes, se pueden distinguir los siguientes tipos de anomalías, tal como se propone en Blázquez-García *et al.* (2020) y Li y Jung (2023):

1. **Instante de tiempo anómalo.** Se define como un instante de tiempo t en el que se observan valores inusuales respecto a otros valores de la serie (anomalía global) o respecto a observaciones de instantes vecinos (anomalía local o contextual). Un instante de tiempo anómalo puede ser univariante o multivariante dependiendo de si se ven afectadas una o más variables de la serie, respectivamente. En la Figura 2.1(a) podemos ver algunos ejemplos de instantes de tiempo anómalos en una serie temporal multivariante.

2. **Intervalo de tiempo anómalo.** Se corresponde con un cierto número de instantes de tiempo consecutivos en los que la serie no se comporta según el patrón esperado. Cabe destacar que los instantes de tiempo que componen el intervalo anómalo no tienen por qué ser anómalos por sí mismos. Los intervalos de tiempo anómalos también pueden ser globales o contextuales, así como univariantes o multivariantes. En la Figura 2.1(b) se ilustran algunos ejemplos.



(a) Instantes de tiempo anómalos.

(b) Intervalos de tiempo anómalos.

Figura 2.1: Ejemplos de distintos tipos de anomalías en series de tiempo multivariantes. En el panel (a) (izquierda) podemos ver dos instantes de tiempo anómalos multivariantes (O_1 y O_2) y uno univariante (O_3), mientras que en el panel (b) (derecha) se observan dos intervalos de tiempo anómalos multivariantes (O_1 y O_2) y uno univariante (O_3). Imágenes obtenidas de Blázquez-García *et al.* (2020).

3. **Serie de tiempo anómala.** En este caso, alguna de las componentes de la serie temporal multivariante es considerada anómala por seguir un patrón sustancialmente distinto del resto de componentes. En la Figura 2.2 se muestra un caso de serie de tiempo anómala en una serie temporal multivariante.

Un caso especial de anomalía en el contexto de series temporales multivariantes es la denominada *anomalía de correlación* (Wang *et al.*, 2018), que consiste en detectar relaciones anómalas entre algunas variables en un instante o intervalo de tiempo determinado, a pesar de observar un comportamiento normal en cada una de ellas por separado. Puede verse un ejemplo en la Figura 2.3.

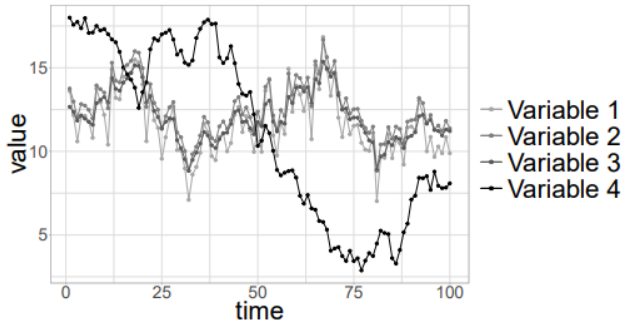


Figura 2.2: Serie de tiempo anómala (Variable 4) en una serie temporal multivariante compuesta por cuatro variables. Imagen obtenida de Blázquez-García *et al.* (2020).

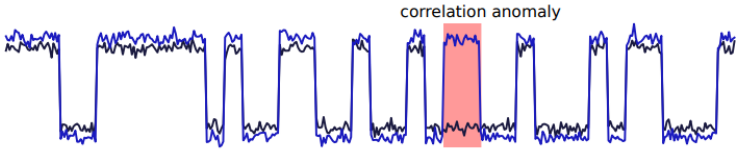


Figura 2.3: Ejemplo de anomalía de correlación en una serie de tiempo multivariante compuesta por dos variables. Imagen obtenida de Schmidl *et al.* (2022).

2.3. Métodos para la detección de anomalías

Debido a la complejidad intrínseca de la detección de anomalías en series temporales multivariantes, los primeros enfoques para abordar esta tarea consisten en obviar la dependencia secuencial de los datos, de forma que se pierde la información temporal de los mismos. Un ejemplo son los gráficos de control multivariantes propios del control estadístico de procesos, como los gráficos *MCUSUM* o *MEWMA*, que se basan en la hipótesis de que los datos son independientes e idénticamente distribuidos. Su objetivo es utilizar técnicas estadísticas que permitan hacer un seguimiento de la media y la varianza de la serie para detectar cambios abruptos en su comportamiento (Audibert *et al.*, 2022).

Como las técnicas de detección de anomalías en series temporales univariantes están mucho más estudiadas y desarrolladas, otro de los primeros enfoques que se plantean consiste en estudiar cada una de las variables que componen la serie temporal multivariante por separado. Así, habría que aplicar métodos de detección de anomalías en cada una de las series y se perdería toda la información conjunta. Esta vía impediría, por ejemplo, detectar posibles anomalías de correlación. Además, si la dimensión de la serie es relativamente alta, habría que ajustar demasiados modelos y no sería un método eficiente.

Con el objetivo de reducir la dimensión de la serie multivariante, se pueden aplicar técnicas derivadas del *análisis de componentes principales* (A.1) para obtener un menor número de series univariantes independientes que sí se podrían estudiar por separado sin perder la información de las correlaciones entre variables (Blázquez-García *et al.*, 2020). La desventaja sería no poder identificar en qué variable o variables tienen lugar las anomalías. En los últimos años, se han desarrollado métodos con técnicas de *machine learning* y *deep learning* que permiten monitorizar todas las variables de una serie de tiempo multivariante a la vez utilizando un único modelo, lo que facilita su puesta en producción en los casos de uso de distintas áreas (González, 2020).

2.3.1. *Scoring*, umbral y etiquetado

Motivado por el caso práctico de Banca Móvil abordado en los siguientes capítulos, de aquí en adelante se pondrá el foco en las técnicas de detección de instantes anómalos en series temporales multivariantes. Muchos de estos métodos adoptan el siguiente procedimiento general:

1. Primero, se construye un modelo que permita obtener una puntuación de anomalía para cada observación (instante) de la serie.
2. A continuación, se establece un límite o umbral de forma que aquellas observaciones con una puntuación superior se consideren anómalas.
3. Finalmente, teniendo en cuenta el umbral establecido, se asigna una etiqueta a cada observación indicando si es normal o anómala.

Dada una serie de longitud T y un modelo de detección de anomalías específico, se denominará *scoring* de la serie a la secuencia $S = \{s_1, s_2, \ldots, s_T\}$, donde $s_t \in \mathbb{R}$ denota la puntuación de anomalía o *anomaly score* que el citado método asigna al instante t, siendo $t = 1, \ldots, T$. Si dos puntuaciones s_{t_1} y s_{t_2} cumplen que $s_{t_1} > s_{t_2}$, entonces la observación en el instante t_1 es más anómala que la del instante t_2 en el contexto que corresponda (Schmidl *et al.*, 2022).

A partir de los *anomaly scores* obtenidos por el modelo y del umbral seleccionado (también llamado *threshold* y habitualmente denotado por θ), se asigna a cada observación una etiqueta binaria y_t para indicar si pertenece a la clase normal ($s_t < \theta,\ y_t = 0$) o anómala ($s_t > \theta,\ y_t = 1$). En el caso de disponer de un *etiquetado* de los datos, es decir, las verdaderas etiquetas para cada instante de tiempo (*Ground Truth*), se puede comparar este etiquetado con el obtenido por el modelo (a través de métricas adecuadas) para evaluar así su desempeño.

2.3.2. Clasificación de los modelos

Los modelos de detección de anomalías más utilizados en el contexto de series temporales multivariantes pueden agruparse en diversas categorías según el modo de entrenamiento, la metodología empleada o el enfoque del modelo. Nos hemos basado principalmente en las clasificaciones propuestas por Audibert *et al.* (2022) y Schmidl *et al.* (2022).

1. **Modo de entrenamiento.** En los modelos de *machine learning* y *deep learning* es habitual separar los datos de la serie en dos conjuntos: por una parte, los datos de entrenamiento, que nos permiten entrenar el modelo, y por otra parte, los datos de test, para evaluar su desempeño. Según el tipo de aprendizaje que lleva a cabo el modelo, se establecen las siguientes categorías:
   - **Métodos supervisados.** Todas las observaciones de la serie deben estar debidamente etiquetadas, de forma que pueda tratarse como un problema de clasificación binaria. El modelo se entrena para distinguir entre comportamientos normales y anómalos, asignando posteriormente las correspondientes etiquetas a los datos de test según lo que ha aprendido, sin necesidad de establecer un umbral. En general, al tratarse de eventos raros, las anomalías representan solo una pequeña parte de las observaciones, y este desequilibrio puede generar un sesgo hacia la clase normal. Además, el modelo podría sobreajustarse a un determinado tipo de anomalía o patrón anómalo. Por esta razón, las técnicas supervisadas solo se recomiendan en contextos en los que la naturaleza de las anomalías sea similar a lo largo de toda la serie, o en los que todos los posibles tipos de anomalías estén bien representados en los datos de entrenamiento, escenario que no suele darse en la práctica.
   - **Métodos semisupervisados.** Su objetivo es entrenar un modelo que aprenda el comportamiento normal de la serie, por lo que es preciso disponer de un subconjunto de los datos en el que se pueda asumir que todas las observaciones son normales (o, al menos, que no

presenten anomalías significativas) para llevar a cabo el entrenamiento de manera eficaz. Así, la detección de anomalías en los datos de test se basaría en determinar si estos se ajustan a ese comportamiento normal aprendido o no. Este enfoque es uno de los más utilizados en el contexto de detección de anomalías en series temporales multivariantes, debido a la propia definición de anomalía como observación o patrón discordante con el comportamiento normal de los datos.

- **Métodos no supervisados.** En este modo de aprendizaje, los modelos no disponen de ningún tipo de etiquetado o información previa sobre la serie, por lo que no tienen referencias de cuál es la clasificación correcta de los datos a la hora de realizar el entrenamiento. Los métodos englobados en esta categoría asumen que las observaciones normales son mucho más frecuentes que las anómalas, y que estas pueden separarse del comportamiento normal por seguir patrones o distribuciones diferentes. Un problema que puede surgir es la detección de anomalías que no resulten de interés para identificar problemas reales, o dicho de otra forma, la detección de ruido o potenciales falsos positivos.

2. **Metodología.** Los modelos existentes para la detección de anomalías en series temporales multivariantes pueden agruparse en tres grandes categorías según la metodología que emplean:

   - **Métodos tradicionales.** En muchos de estos métodos se asume que la serie ha sido generada por un proceso estocástico estacionario. Uno de los más conocidos dentro de este grupo es el modelo *VAR* (*Vector Autoregressive*) (Wei, 2019), en el que cada componente de la serie se modeliza como una combinación lineal de sus propios valores pasados y de los valores pasados de las demás componentes. El objetivo es estimar los parámetros del modelo a partir de los datos, y utilizarlo para predecir en tiempo t el valor de las distintas variables de la serie en tiempo $t+1$. De esta forma, una vez conocido el valor real, el *anomaly score* en $t+1$ se obtiene a partir del error de predicción correspondiente. Para poder aplicar este modelo, debemos realizar un análisis previo sobre la serie temporal multivariante para comprobar su estacionariedad y hacer las transformaciones necesarias si fuese el caso.

   - ***Machine Learning.*** En los métodos de esta categoría no se asume ninguna hipótesis de estacionariedad ni de linealidad. En general, se modeliza la dependencia temporal de la serie o bien creando variables adicionales (derivadas de los datos) que recojan esta información o bien utilizando la técnica de *ventanas deslizantes*. Esta técnica consiste en transformar la serie en una secuencia de ventanas, que denotaremos por $\mathcal{W} = \{W_1, \ldots, W_T\}$, donde W_t es la ventana de longitud K para el tiempo t, esto es, $W_t = \{\boldsymbol{x}_{t-K+1}, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t\}$ (Audibert *et al.*, 2022). Habitualmente, estas ventanas se construyen "deslizándose" hacia adelante en la serie un instante de tiempo cada vez (o lo que es lo mismo, con paso 1), por lo que están superpuestas. El *anomaly score* para un instante t se obtiene a partir del contexto proporcionado por su ventana W_t correspondiente. Dentro de esta categoría se incluyen los métodos basados en árboles de decisión, distribuciones o distancias, entre otros.

   - ***Deep Learning.*** A este grupo pertenecen la gran mayoría de métodos desarrollados en los últimos años, debido a su buen desempeño en el problema de detección de anomalías en series de tiempo multivariantes. Al igual que en los métodos de *machine learning*, no se asumen hipótesis sobre los datos y lo más común es utilizar la técnica de ventanas deslizantes para modelizar la dependencia temporal de la serie. Estos modelos están compuestos principalmente por redes neuronales como las redes *LSTM* (*Long Short-Term Memory*), las *GAN* (*Generative Adversarial Networks*) o los *autoencoders*. Son de especial interés los llamados modelos *sequence-to-sequence* (o *seq2seq*), que se componen de una arquitectura de *encoder-decoder* y están especialmente diseñados para tratar datos secuenciales, como las series de tiempo. En la Sección 2.5 se verá más en detalle en qué consisten algunas de estas redes y arquitecturas.

3. **Enfoque.** Podemos caracterizar los distintos métodos por la manera en la que determinan si una observación es anómala o no.

   - **Métodos predictivos.** Utilizan un modelo previamente estimado o entrenado (a menudo de forma semisupervisada) para predecir valores futuros de la serie a partir de una secuencia de valores pasados que dotan de contexto al instante actual (utilizando la técnica de ventanas deslizantes con paso 1). Muchos de estos modelos predicen únicamente la observación siguiente en cada instante, y estas predicciones se comparan con los valores observados de la serie utilizando alguna métrica de distancia adecuada. Los *anomaly scores* que devuelve el modelo se calculan entonces a partir de los errores de predicción, y su distribución puede emplearse para fijar el umbral que determina las observaciones anómalas. Como ejemplos de esta categoría, podemos mencionar el modelo *LSTM-AD* (Malhotra *et al.*, 2015) o el modelo *DeepAnT*, con una arquitectura más compleja (Munir *et al.*, 2019).
   - **Métodos de reconstrucción.** Su objetivo es construir un modelo del comportamiento normal de la serie codificando subsecuencias de los datos de entrenamiento en un espacio latente de dimensión menor. Luego, los datos de test se reconstruyen desde el espacio latente, utilizando como contexto en cada instante la correspondiente ventana. Se asume que las subsecuencias anómalas no se van a reconstruir correctamente, por lo que los errores de reconstrucción y su distribución nos permiten determinar los *anomaly scores* y el umbral del método, respectivamente. Como ejemplos, citar el modelo *EncDec-AD* (Malhotra *et al.*, 2016) o el modelo *LSTM-VAE* (Park *et al.*, 2017).
   - **Métodos de codificación.** Al igual que los métodos de reconstrucción, codifican las subsecuencias de la serie en un espacio latente de dimensión menor, pero calculan los *anomaly scores* directamente desde las representaciones en el espacio latente. En nuestro contexto, apenas se utilizan este tipo de modelos.
   - **Métodos basados en distancia.** Muchos son métodos no supervisados que asumen que las observaciones anómalas se distancian más del comportamiento normal de la serie que aquellas que se consideran normales. Se utiliza la técnica de ventanas deslizantes con paso 1 y se emplean métricas de distancia adecuadas para comparar las distintas observaciones entre ellas, siendo estas distancias los *anomaly scores* obtenidos por el modelo. Algunos métodos de esta categoría solo consideran las observaciones o secuencias más "cercanas" para hacer la comparación, como *k Nearest Neighbors* (Ramaswany *et al.*, 2000) o *Local Outlier Factor* (Breunig *et al.*, 2000).
   - **Métodos basados en distribución.** Estos métodos estiman la distribución de los datos o ajustan un modelo de distribución sobre las observaciones de la serie, obteniendo los *anomaly scores* a partir de probabilidades, verosimilitudes o distancias respecto de la distribución establecida. Se asume que las anomalías se encuentran en los extremos o colas de la distribución. Un ejemplo con enfoque semisupervisado es el modelo *Fast-MCD* (Rousseeuw y Driessen, 1999), que estima los parámetros de una distribución normal multivariante (A.6) a partir de los datos de la serie y calcula los *anomaly scores* como la distancia de Mahalanobis (A.3) entre cada observación y la media de la distribución considerada.
   - **Métodos basados en árboles de decisión** (A.2). La mayoría son métodos no supervisados en los que se establece un conjunto de árboles de decisión aleatorios cuyo propósito es "separar" los datos de test. Cada árbol se construye seleccionando de forma recursiva y aleatoria las variables y los puntos de división hasta llegar a las hojas del árbol. Se asume que las anomalías tienen caminos más cortos desde la raíz del árbol hasta la hoja en la que se encuentran, pues en teoría son más fáciles de aislar que los valores normales. Así, para cada observación, el *anomaly score* será el inverso de la longitud promedio de todos sus caminos en los árboles del conjunto (cuanta menor longitud, mayor puntuación de anomalía). El modelo más representativo de esta categoría y que sirve de base para todos los demás es el método de *Isolation Forest* (Liu *et al.*, 2008).

Para concluir esta sección, mencionar que los métodos semisupervisados de *deep learning* basados en predicción o reconstrucción son los más utilizados a día de hoy para el problema de detección de anomalías en series temporales multivariantes. Aún así, los métodos no supervisados de *machine learning* basados en distancias o árboles de decisión pueden ser de gran utilidad en casos en los que las anomalías se identifiquen principalmente con valores atípicos o *outliers*. Los métodos supervisados se restringen a escenarios muy concretos en los que todas las anomalías están correcta y exhaustivamente etiquetadas. Con todo, hay que tener en cuenta que no hay un modelo que funcione mejor que todos los demás en todas las situaciones (Audibert *et al.*, 2022), y el desempeño de cualquiera de ellos va a estar condicionado en última instancia al caso concreto en el que se apliquen.

2.3.3. Selección del umbral

Como se puede intuir, el umbral juega un papel muy importante en el rendimiento de los modelos, pues seleccionar un valor u otro puede llevar a métricas mejores o peores para los modelos considerados (Li y Jung, 2023). Además, este umbral debe ser lo suficientemente robusto frente a la detección de ruido para poder identificar las anomalías significativas. Existen diversas maneras de elegir el umbral que determina el etiquetado del modelo y, a menudo, su elección va a depender del contexto en el que se trabaje. Por una parte, puede ocurrir que en un área concreta se tenga un conocimiento muy específico del comportamiento de las anomalías, de forma que se pueda establecer un valor fijo para el umbral simplemente a partir de este conocimiento. Por otra parte, habrá modelos que contemplen un método propio para la selección del umbral. Sin embargo, en muchos otros casos, hay que recurrir a otro tipo de técnicas, como las que se mencionan a lo largo de este apartado.

A la hora de realizar el entrenamiento de un modelo, es habitual reservar un cierto porcentaje de los datos de entrenamiento (sobre un 20 o un 30 %) para llevar a cabo la validación. Para estos datos de validación se obtiene un *scoring* que puede ser utilizado para seleccionar un umbral adecuado. Se puede considerar, por ejemplo, el máximo de los *anomaly scores* de validación (Assendorp, 2017), o la media de estos *anomaly scores* más un cierto número k de desviaciones típicas, donde $k \in \mathbb{R}^+$ controlaría la sensibilidad de la detección. En este último caso, se estaría asumiendo que los *anomaly scores* siguen una distribución normal y, utilizando la conocida como "Regla de las tres sigma", es habitual seleccionar $k = 3$ desviaciones típicas (Laptev *et al.*, 2015).

En un escenario en el que se conoce o se puede estimar el porcentaje esperado de observaciones anómalas en el conjunto de datos de test, llamémosle α, una opción sería calcular el cuantil $1 - \alpha$ de la distribución de estos datos de forma que el $\alpha\,\%$ de las observaciones queden por encima del umbral y se etiqueten como anómalas. Esta forma de seleccionar el umbral se emplea, por ejemplo, en el modelo de *Isolation Forest*, donde α recibe el nombre de *contamination*.

Otra opción sería elegir el umbral de forma que se maximice alguna de las posibles métricas de evaluación, como el *Recall* o el F_1-*Score* (Assendorp, 2017), métricas que se revisarán en la siguiente sección. El buen funcionamiento de esta vía depende en gran medida de la calidad del etiquetado de verdaderas anomalías para los datos de test. En relación con esta técnica, también se podría considerar una lista de candidatos a umbrales y seleccionar de entre todos ellos el que mejores resultados devuelva respecto a una métrica concreta. Este es el método empleado por Audibert *et al.* (2022), donde se hacen pruebas con 1000 posibles valores del umbral y se elige el que proporciona una mejor puntuación de la métrica F_1-*Score*.

Las técnicas mencionadas son solo algunos ejemplos de las múltiples opciones para determinar un umbral adecuado en los modelos de detección de anomalías en series temporales multivariantes. En la práctica, siempre que sea oportuno, pueden probarse varias de estas técnicas y comparar los diferentes resultados obtenidos, tanto en el etiquetado como en las métricas de interés.

2.4. Métricas de evaluación

Es importante disponer de métricas apropiadas que permitan evaluar la efectividad y rendimiento de un modelo. En esta sección, se revisan distintas alternativas de métricas utilizadas para los modelos de detección de anomalías en series temporales multivariantes.

2.4.1. Clasificación

Dado que el etiquetado consiste en declarar cada instante temporal como de clase normal o anómalo, parece razonable usar algunas de las métricas propias de los problemas de clasificación binaria. Para esta revisión se toman en consideración principalmente el trabajo de Belay *et al.* (2023) y los apuntes de Fernández-Casal *et al.* (2021).

Para definir y calcular estas métricas, se tienen en cuenta los siguientes parámetros: el número de anomalías correctamente detectadas (*True Positives, TP*), el número de anomalías erróneamente detectadas (falsas alarmas o *False Positives, FP*), el número de instantes normales correctamente identificados (*True Negatives, TN*) y el número de instantes normales erróneamente identificados (*False Negatives, FN*). Así, el número total de anomalías detectadas por el modelo (positivos) se obtiene como $TP + FN$, mientras que el número total de instantes normales (negativos) será $FP + TN$. Estos datos se organizan en una tabla de contingencia llamada *matriz de confusión*, donde se representan las predicciones del modelo frente a los valores reales, como puede verse en la Tabla 2.1.

	Clase positiva	**Clase negativa**
Predicción positiva	TP	FP
Predicción negativa	FN	TN

Tabla 2.1: Estructura de la matriz de confusión.

Aunque existen numerosas métricas de evaluación para los modelos de clasificación, se pondrá el foco en cuatro de las más representativas en el contexto que nos ocupa.

- ***Precision.*** También llamada *True Positive Rate*, representa la proporción de anomalías detectadas por el modelo que son verdaderamente anomalías.
- ***Recall.*** Esta métrica representa la proporción de verdaderas anomalías que el modelo detecta correctamente.
- ***AUC-PR.*** *Precision* y *Recall* dependen del umbral establecido para el modelo y están relacionadas de forma que al aumentar una de ellas disminuirá la otra. La métrica *AUC-PR* no es más que el resultado de integrar la *curva Precision-Recall (PR)*, una representación gráfica de los valores de *Precision* para los posibles valores de *Recall*.
- ***F_1-Score.*** Se define como la media armónica de *Precision* y *Recall*.

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}, \qquad F_1\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Al ponderar los valores de *Precision* y *Recall*, el F_1-*Score* es particularmente útil en escenarios como el de detección de anomalías donde las clases están muy desbalanceadas. Cuanto más próximo a 1 sea el valor de todas estas métricas, mejor será el rendimiento del modelo considerado a la hora de detectar instantes anómalos en la serie.

2.4.2. *PATE*

Las métricas clásicas introducidas en el apartado anterior asumen que los datos son independientes e idénticamente distribuidos, hipótesis que no se cumple en el caso de las series temporales. En el reciente artículo de Ghorbani *et al.* (2024), se introduce una nueva métrica de evaluación especialmente diseñada para este tipo de modelos, denominada *Proximity-Aware Time series anomaly Evaluation* o *PATE*, cuyo objetivo es incorporar la relación temporal inherente a las series de tiempo y el hecho de que, a menudo, las anomalías se corresponden con sucesivos instantes anómalos, constituyendo así diferentes intervalos anómalos. Además, también se busca tener en cuenta la detección temprana (anticipación) o tardía de las anomalías, así como el rango de cobertura en cuanto a la duración, ya que no hacerlo puede llevar a conclusiones erróneas sobre el verdadero desempeño del modelo.

Al objeto de ilustrar diferentes escenarios en el problema de detección de anomalías, tómese en consideración la Figura 2.4. En el escenario de la izquierda, la predicción 1 anticipa la anomalía antes de que esta se produzca, mientras que la predicción 2 la detecta justo después de haber finalizado. En el escenario central, ambas predicciones se solapan con el verdadero intervalo anómalo, pero la predicción 1 ofrece una detección más rápida. En el escenario de la derecha, aunque ambas la detectan con retraso y al mismo tiempo, de nuevo la predicción 1 es mejor por cubrir un mayor rango de la duración de la anomalía.

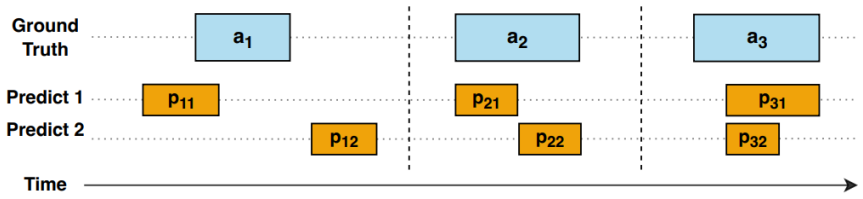


Figura 2.4: Diferentes escenarios en un problema de detección de anomalías. Los recuadros azules denotados por a representan las verdaderas anomalías, mientras que los recuadros naranjas denotados por p son las predicciones que devuelven ciertos modelos. La longitud de los distintos recuadros a y p indica la duración del evento anómalo. Los tramos en los que se solapan corresponden a las anomalías correctamente detectadas por el modelo. Imagen obtenida de Ghorbani *et al.* (2024).

Una vez introducidas estas ideas, veamos cómo se define esta nueva métrica. En el etiquetado obtenido en los modelos de detección de anomalías tras aplicar un umbral θ a los *anomaly scores*, se define un *evento de predicción*, $\boldsymbol{p}_l(\theta)$, como un subsegmento de instantes sucesivos considerados anómalos en ese etiquetado. El conjunto de todos los eventos de predicción se denotará por $\boldsymbol{P} = \{\boldsymbol{p}_l(\theta)\}_{l=1}^{M}$, con M el número de eventos identificados por el modelo. El rendimiento del modelo se determina entonces por cómo de bien se ajustan estos eventos de predicción a los verdaderos eventos anómalos de la serie.

Para calcular esta métrica, se distinguen primero cuatro categorías de detección: *True Detection* (solapamiento del valor real y la predicción), *Post-Buffer Detection* (detección tardía de la anomalía), *Pre-Buffer Detection* (capacidad de anticipación) y *Outside* (si la predicción no está asociada a ninguna anomalía real, por lo que sería un falso positivo). Así, un evento anómalo puede ser completamente detectado, parcialmente detectado o directamente no detectado por el modelo. La idea es asignar a cada instante de tiempo t unos ciertos pesos $w^{\mathrm{TP}}(t)$, $w^{\mathrm{FP}}(t)$ y $w^{\mathrm{FN}}(t)$ para determinar su contribución a los parámetros de *True Positive*, *False Positive* y *False Negative*, respectivamente. Estos pesos toman valores entre 0 y 1 y su definición va a depender de la zona de detección en la que se encuentre el instante de tiempo correspondiente (la explicación detallada puede verse en Ghorbani *et al.*, 2024).

Ahora, se establece un rango de posibles tamaños para las zonas de detección temprana (e) o detección tardía (d) de las anomalías y un rango de posibles umbrales θ. Así, es posible definir el *Precision* y el *Recall* para cada combinación de e, d y θ como sigue:

$$Precision_{e,d}(\theta) = \frac{\sum_{t=1}^{T} w^{\mathrm{TP}}(t)}{\sum_{t=1}^{T} w^{\mathrm{TP}}(t) + \sum_{t=1}^{T} w^{\mathrm{FP}}(t)}, \qquad Recall_{e,d}(\theta) = \frac{\sum_{t=1}^{T} w^{\mathrm{TP}}(t)}{\sum_{t=1}^{T} w^{\mathrm{TP}}(t) + \sum_{t=1}^{T} w^{\mathrm{FN}}(t)}$$

Finalmente, la métrica *PATE* se obtiene como un promedio de todos los valores de *AUC-PR* para las distintas combinaciones de e y d ($|E|$ y $|D|$ representan el número de valores distintos de e y d):

$$PATE = \frac{1}{|E| \times |D|} \sum_{e \in E,\ d \in D} AUC\text{-}PR_{e,d}$$

2.4.3. Métricas adaptadas

Como se ha explicado en la descripción del problema en el Capítulo 1, el principal objetivo de este trabajo es utilizar los modelos de detección de anomalías en series temporales multivariantes para anticipar o predecir las posibles caídas de la *app* de Banca Móvil, idealmente a nivel de minutos en tiempo real. Las métricas descritas hasta ahora evalúan el rendimiento del modelo a la hora de identificar anomalías, pero podrían adaptarse ligeramente para que se ajusten mejor al propósito perseguido.

Aplicar directamente las métricas clásicas de clasificación generaría una gran cantidad de falsos positivos. La razón es que, en muchos casos, en los instantes previos a la caída el modelo devuelve una predicción positiva mientras que el verdadero etiquetado indica una clase negativa. Además, como se mostrará en el Capítulo 4, frecuentemente el modelo dejará de devolver una predicción positiva instantes antes de que la caída se solucione completamente y la verdadera clase vuelva a ser negativa. Por lo tanto, una opción para adaptar las métricas de clasificación sería desplazar hacia delante el etiquetado que devuelve el modelo un total de n retardos o *lags*, para compararlo así con el verdadero etiquetado sin desplazar. De esta forma, escogiendo un *lag* adecuado, las métricas reflejarían mejor el desempeño del modelo a la hora de detectar y anticipar las caídas. La Figura 2.5 ilustra de forma gráfica un ejemplo de esta idea.

G.T.	0	0	0	1	1	1	1	0	...
Pred.	0	1	1	1	1	0	0	0	...

$\longrightarrow$

G.T.	0	0	0	1	1	1	1	0	...
Pred.			0	1	1	1	1	0	...

Figura 2.5: Representación de un desplazamiento de 2 *lags* para la predicción (Pred.) frente a la *Ground Truth* (*G.T.*).

Nótese que la métrica PATE evalúa todos los escenarios posibles de detección de un evento anómalo. Sin embargo, el interés principal aquí es evaluar la capacidad de anticipación del modelo. En efecto, en un escenario en tiempo real, no resulta de mucha utilidad una detección tardía del modelo ya que se sabe que la caída se ha producido. Asumiendo las caídas como eventos anómalos, en vez de calcular las métricas teniendo en cuenta todos los instantes temporales, se podría calcular: (i) la proporción de verdaderas caídas correctamente anticipadas con un número específico de minutos de antelación (una especie de *Recall* para esos minutos de antelación), y (ii) la proporción de caídas verdaderas de entre todas las que detecta el modelo (una especie de *Precision*). Se volverá sobre estas ideas en el Capítulo 4, cuando se presenten los resultados obtenidos por los distintos modelos.

2.5. Modelos empleados

En la Sección 2.3 se presentaron una gran variedad de métodos para la tarea de detección de anomalías en series temporales multivariantes. En esta sección se profundiza en cuatro de estos modelos, que serán aplicados en el Capítulo 4 a los datos de Banca Móvil: *Isolation Forest*, *k Nearest Neighbors*, *LSTM-AD* y *EncDec-AD*. Los dos primeros son métodos no supervisados de *machine learning* basados en árboles de decisión y distancias, respectivamente. Estos modelos se utilizan principalmente en contextos con datos independientes, pero pueden adaptarse convenientemente para ser utilizados con series de tiempo. Los dos últimos son métodos semisupervisados de *deep learning* basados en predicción y reconstrucción, respectivamente. Estos modelos sí han sido específicamente diseñados para trabajar con series temporales, por lo que se espera que tengan un rendimiento mejor en nuestro contexto. Como se verá en el Capítulo 3, no se dispondrá de un etiquetado exhaustivo de las anomalías de Banca Móvil, por lo que no se emplearán métodos supervisados.

También es relevante enfatizar que el modelo *VAR*, uno de los principales representantes de los métodos tradicionales, no será considerado en el presente estudio por las razones que siguen. Para un proceso estocástico multivariante estacionario $\boldsymbol{X}_t$ se define el modelo *VAR* de orden p como sigue:

$$\boldsymbol{X}_t = \boldsymbol{A}_1\boldsymbol{X}_{t-1} + \boldsymbol{A}_2\boldsymbol{X}_{t-2} + \cdots + \boldsymbol{A}_p\boldsymbol{X}_{t-p} + \boldsymbol{\varepsilon}_t,\ t \in \mathbb{Z},$$

donde cada $\boldsymbol{A}_j,\ j = 1,\ \ldots,\ p$, es una matriz de coeficientes constantes (los parámetros del modelo) y $\boldsymbol{\varepsilon}_t$ es un proceso de ruido blanco multivariante (Belay *et al.*, 2023). Este modelo es costoso computacionalmente, debido al crecimiento cuadrático de los parámetros a medida que aumenta la dimensión de la serie. Además, requiere una nueva estimación de estos parámetros por cada instante que se desea predecir, resultando ser un método ineficiente para el propósito de detección de anomalías.

A continuación, se describen más en detalle cada uno de los cuatro modelos que se han seleccionado para tratar posteriormente el caso práctico de detección de anomalías en series temporales multivariantes.

2.5.1. *Isolation Forest*

El método de *Isolation Forest* o *iForest*, introducido por Liu *et al.* (2008), es un modelo no supervisado de detección de anomalías basado en árboles de decisión (A.2) aleatorios. Su objetivo es "aislar" los datos anómalos, en contraposición al enfoque mayoritario de crear un perfil de comportamiento normal y detectar las desviaciones respecto al mismo. Este modelo aprovecha dos características propias de las anomalías: representan una minoría de las observaciones y los valores de las variables en ese instante son "diferentes" a los de los instantes normales. Al construir árboles que separan las observaciones de forma aleatoria, las anomalías se situarán más cerca de la raíz, pues son más susceptibles a ser aisladas y tendrán caminos más cortos hasta la hoja correspondiente.

Se procede a definir la estructura de los árboles de decisión utilizados en este método, conocidos como *Isolation Trees* o *iTrees*. Supóngase que tenemos un conjunto de datos de dimensión m. Un cierto nodo T de un *Isolation Tree* puede ser un nodo terminal o un nodo interno con una condición asociada. En este último caso, la condición relativa al nodo se define a partir de una variable X_q, $q \in \{1, \ldots, m\}$, y un valor de división p (en el rango de X_q) de forma que $X_q < p$ va a dividir los datos en dos conjuntos disjuntos, que denominaremos T_l y T_r. La idea del método consiste en aplicar este proceso recursivamente para separar los datos, seleccionando X_q y p de manera aleatoria en cada división. El proceso continúa hasta que todos los nodos terminales contengan una única observación (o varias en el caso de que tengan los mismos valores) o hasta que se alcance una profundidad límite establecida (un número máximo de niveles de división).

La *longitud del camino* para un punto $\boldsymbol{x} \in \mathbb{R}^m$, denotada por $h(\boldsymbol{x})$, se define como el número de nodos que $\boldsymbol{x}$ tiene que atravesar desde la raíz del árbol hasta su correspondiente nodo terminal. Cuánto más corto sea este camino, mayor será el carácter anómalo de $\boldsymbol{x}$. Por lo tanto, si se construyen varios *Isolation Trees* sobre un mismo conjunto de datos, las anomalías serán aquellas observaciones que tienen una longitud promedio corta para sus caminos en los distintos árboles.

Para mejorar la eficiencia del método, cada *iTree* se construye con una submuestra diferente de los datos (a través de una selección aleatoria sin reemplazamiento), de forma que no se utilizan todas las observaciones en todos los árboles. Esto permite además que los *Isolation Trees* sean especializados, pues la submuetra con la que se construye cada uno de ellos puede contener distintos tipos de anomalías o incluso ninguna. Sin embargo, el hecho de que las submuestras sean diferentes hace que no sea posible derivar un *anomaly score* directamente a partir de la longitud promedio de los caminos: es necesario normalizarla. Si n es el tamaño de la submuestra utilizada para construir los *iTrees*, se define el *anomaly score* s de un punto $\boldsymbol{x}$ como:

$$s(\boldsymbol{x}, n) \;=\; 2^{-E(h(\boldsymbol{x}))/c(n)}, \tag{2.1}$$

siendo $E(h(\boldsymbol{x}))$ el promedio de $h(\boldsymbol{x})$ para una colección de *iTrees* y $c(n)$ el promedio de $h(\boldsymbol{x})$ dado n. La expresión para calcular $c(n)$ se deriva del estudio de los *Binary Search Trees* (Preiss, 1999) y viene dada por $c(n) \;=\; 2H(n-1) - (2(n-1)/n)$, con $H(i)$ el número armónico i-ésimo, que se estima como $\ln(i) + 0.577215$ (constante de Euler).

Los *anomaly scores* obtenidos a partir de la Ecuación (2.1) toman valores en el intervalo $(0, 1]$. Suponiendo que todos los datos tienen distintos valores, el número de nodos externos de un *iTree* es igual a n y el número de nodos internos es igual a $n-1$, por lo que $0 < h(\boldsymbol{x}) \leq n-1$. Así,

$$E(h(\boldsymbol{x})) \to n-1 \;\Rightarrow\; s \to 0 \;\;;\;\; E(h(\boldsymbol{x})) \to c(n) \;\Rightarrow\; s \to 0.5 \;\;;\;\; E(h(\boldsymbol{x})) \to 0 \;\Rightarrow\; s \to 1.$$

Si el *anomaly score* de un punto $\boldsymbol{x}$ está próximo a 0, podemos considerar que es un dato normal; sin embargo, un *anomaly score* próximo a 1 indica que el punto tiene un carácter claramente anómalo.

Se describe a continuación cómo utilizar este método para detectar anomalías. En la fase de entrenamiento, se construyen t *Isolation Trees* utilizando distintas submuestras de tamaño n de los datos en cada uno de ellos. La construcción de un *iTree* termina cuando todos los datos han sido aislados o cuando se alcanza una profundidad máxima, establecida como $l \;=\; ceiling(\log_2 n)$ (valor que aproxima la profundidad promedio de un árbol para un tamaño n de la submuestra). Este límite contribuye a mejorar la eficiencia del modelo al evitar cálculos innecesarios, pues para detectar anomalías solo nos interesan los datos con caminos más cortos que el promedio. Por lo tanto, el número t de *iTrees* y el tamaño n de las submuestras empleadas son los parámetros del modelo, a elegir por el usuario. En el artículo de referencia (Liu *et al.*, 2008) se proponen los valores $t = 100$ y n una potencia de 2 no demasiado grande, por ejemplo, $n = 256$. Aunque el método es en esencia no supervisado, puede realizarse el entrenamiento utilizando solo datos normales, es decir, llevándolo a un enfoque semisupervisado. En este caso, es conveniente utilizar un tamaño de submuestra más grande, como por ejemplo $n = 8192$.

En la fase de evaluación, cada dato $\boldsymbol{x}$ de test se pasa por cada uno de los *iTrees* construidos en el entrenamiento para calcular $E(h(\boldsymbol{x}))$ y obtener así su *anomaly score* correspondiente. Una vez se tiene el *scoring* completo del modelo, debe fijarse un umbral a partir del cual se marcarán las anomalías. Si se conoce la proporción aproximada de anomalías que se espera encontrar en los datos de test, a la que se llamará *contamination*, se utilizaría esta proporción para establecer el umbral, de forma que ese porcentaje de los datos con los *anomaly scores* más altos se identifiquen como anomalías.

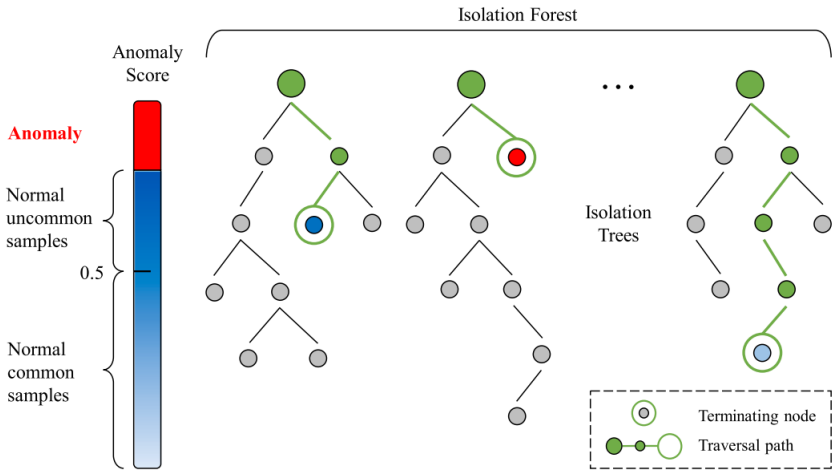


Figura 2.6: Detección de anomalías con *Isolation Forest*. Imagen obtenida de Chen *et al.* (2020).

En la Figura 2.6 se puede ver una representación gráfica del método que hemos descrito de *Isolation Forest* para detección de anomalías. Lo único que se debe tener en cuenta a la hora de aplicar este método en series de tiempo multivariantes es que no se está considerando en ningún momento la dependencia secuencial. Por lo tanto, cabe definir variables adicionales a partir de los datos disponibles que recojan la información temporal o bien utilizar la técnica de ventanas deslizantes explicada en la Sección 2.3.2.

Este método presenta numerosas ventajas. Es eficiente, pues el tiempo de computación se reduce significativamente gracias al uso de submuestras para la construcción de los *iTrees* y al hecho de fijar una profundidad máxima que evite cálculos innecesarios. Es eficaz en conjuntos de datos de alta dimensión, al no verse afectado por la *maldición de la dimensionalidad* (A.10) ni por las distintas escalas que puedan tener las variables. Además, si las anomalías son muy escasas en los datos, el modelo puede entrenarse únicamente con datos normales (enfoque semisupervisado), pues en general sigue ofreciendo un buen rendimiento si se aumenta el tamaño de las submuestras. Como principales desventajas citar la falta de interpretabilidad del modelo, debido a la construcción aleatoria de los *iTrees* en la fase de entrenamiento, y la propensión a detectar ruido o numerosos falsos positivos, al tratarse de un método no supervisado de detección de anomalías.

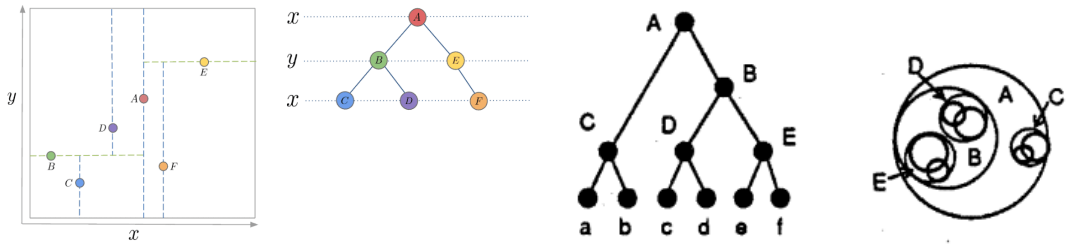
2.5.2. *k Nearest Neighbors*

El método de *k Nearest Neighbors* (*KNN*) o k vecinos más cercanos para detección de anomalías, propuesto por Ramaswamy *et al.* (2000), consiste en un modelo no supervisado basado en distancias que está especialmente diseñado para la detección de datos atípicos o *outliers*, un tipo de anomalía que se caracteriza por presentar valores considerablemente distantes o diferentes del resto de los datos.

Supóngase que se dispone de un conjunto de datos de dimensión m. Para un número $k \in \mathbb{N}$ y un punto $\boldsymbol{p} \in \mathbb{R}^m$, se define $D^k(\boldsymbol{p})$ como la distancia entre $\boldsymbol{p}$ y su k-ésimo vecino más cercano. Por lo tanto, $D^k(\boldsymbol{p})$ será una medida del carácter anómalo de $\boldsymbol{p}$, y se podrán ordenar los datos de menor a mayor según la distancia a su k-ésimo vecino más cercano. Los *outliers* serán aquellas observaciones con valores muy elevados de D^k en relación a los demás puntos. Para calcular las correspondientes distancias D^k asociadas a los datos, se puede utilizar cualquier métrica adecuada, como la distancia Manhattan (A.5) o la distancia euclídea (A.4).

Denotando por n al número de datos anómalos que se espera encontrar, se considera la definición de *outlier* basada en distancias: *Dados k y n, un punto $\boldsymbol{p}$ es un outlier si no más de $n-1$ puntos de los datos tienen un valor de D^k más grande que $\boldsymbol{p}$* (Ramaswamy *et al.*, 2000). Es decir, los n puntos con mayor distancia D^k serán considerados *outliers*. De esta forma, no es preciso conocer información previa alguna sobre la distribución o la clasificación de los datos.

Existen diversos métodos para obtener los k vecinos más cercanos a un cierto punto. Una primera forma es la fuerza bruta, calculando las distancias de ese punto a todos los demás y seleccionando las k distancias más pequeñas (los k vecinos más cercanos), lo que resulta claramente ineficiente. Otra opción es utilizar *KDTrees* o *BallTrees*, estructuras basadas en árboles que particionan el espacio en regiones o hiperesferas de forma que la búsqueda de los k vecinos más cercanos para un punto se restringe a las zonas definidas por esas estructuras. En un *KDTree*, cada nodo representa una región del espacio delimitada por hiperplanos, y se selecciona una dimensión de los datos para hacer una nueva división. En un *BallTree*, cada nodo representa una hiperesfera (una región definida por un punto central y un radio) que contiene un determinado subconjunto de los datos, y las divisiones se realizan buscando las dos hiperesferas que minimizan la suma de sus volúmenes. Los *KDTrees* están más afectados por la maldición de la dimensionalidad, mientras que los *BallTrees* tienen una estructura compleja y son más costosos computacionalmente. Para un mayor detalle sobre la construcción de estos árboles y el funcionamiento de estos algoritmos, puede consultarse Omohundro (1989). En la Figura 2.7 se muestran un par de ejemplos gráficos de la construcción de estos árboles.



(a) *KDTree* (b) *BallTree*

Figura 2.7: Ejemplos sencillos de construcción de un *KDTree* y un *BallTree* en $\mathbb{R}^2$. Imágenes obtenidas de Hristov (2023) y Omohundro (1989), respectivamente.

El único parámetro del modelo es, por tanto, el número k de vecinos más cercanos, pudiendo elegir a mayores la métrica utilizada para el cálculo de las distancias y el algoritmo de búsqueda de los k vecinos más cercanos de un punto. En la práctica, es habitual utilizar como $D^k(\boldsymbol{p})$ el promedio de todas las distancias del punto $\boldsymbol{p}$ a cada uno de sus k vecinos más próximos en los datos de entrenamiento, en lugar de simplemente la distancia al k-ésimo vecino más cercano. Estas distancias promedio serán los *anomaly scores* obtenidos por el modelo para los datos de test. Además, al igual que en el método de *Isolation Forest*, se emplea la proporción de anomalías esperadas en los datos (*contamination*) para determinar el umbral que marcará las observaciones anómalas. Nótese que, de nuevo, no se considera la dependencia secuencial en el caso de aplicar el método a series temporales multivariantes, por lo que deberá usarse alguna de las técnicas mencionadas anteriormente.

Las principales ventajas del método de k *Nearest Neighbors* son su sencillez y su capacidad de capturar relaciones complejas entre las variables. Sin embargo, al ser un método basado en distancias, se ve muy afectado por la maldición de la dimensionalidad y es sensible a las distintas escalas de las variables. Además, al ser no supervisado, puede ser propenso a la detección de falsos positivos.

2.5.3. *LSTM-AD*

El modelo *LSTM-AD* o *Long Short Term Memory for Anomaly Detection* es un método semisupervisado basado en predicción propuesto por Malhotra *et al.* (2015) para detectar anomalías en series temporales multivariantes. La idea consiste en entrenar un modelo compuesto por una red neuronal con capas *LSTM* apiladas para que aprenda el comportamiento normal de la serie, de forma que se realicen las predicciones de los valores futuros basándose en los patrones aprendidos. Posteriormente, se utilizan los errores de predicción para identificar comportamientos anómalos en los datos de test.

Una red *LSTM* es un tipo de *red neuronal recurrente* especialmente diseñada para tratar datos secuenciales como las series de tiempo debido a su capacidad para aprender y recordar información a corto y largo plazo. Para entender cómo funcionan este tipo de redes, se procede a explicar en primer lugar en qué consisten las redes neuronales artificiales (*Artificial Neural Networks, ANN*) y las redes neuronales recurrentes (*Recurrent Neural Networks, RNN*). Como principales referencias mencionar las monografías de James *et al.* (2023), Goodfellow *et al.* (2016), Torres (2020) y los apuntes de Fernández-Casal *et al.* (2021).

Redes neuronales artificiales

Las *redes neuronales artificiales*, o simplemente *redes neuronales*, son una metodología de aprendizaje inspirada en la estructura de las conexiones neuronales del cerebro humano que ha ganado popularidad en los últimos años, gracias a los grandes avances en el campo de la computación. Estos modelos constan de un elevado número de parámetros que permiten abordar problemas con estructuras muy complejas, aunque requieren conjuntos de datos con tamaños muestrales grandes para conseguir un buen rendimiento, lo que eleva a su vez el tiempo de computación.

Los modelos de redes neuronales están compuestos por nodos o unidades llamadas *neuronas*, conectadas entre sí a través de enlaces y organizadas en capas, habitualmente con la siguiente estructura: una capa de entrada o *input layer*, donde se encuentran las variables predictoras que entran al modelo; una o más capas ocultas o *hidden layers*, cada una con un cierto número de neuronas; y una capa de salida o *output layer*, con las predicciones finales que devuelve el modelo. En el caso de tener un número elevado de capas ocultas, también reciben el nombre de *redes neuronales profundas*.

De forma general, el funcionamiento de las redes neuronales se resume como sigue. En cada enlace de la red se multiplica el valor de salida de la neurona anterior por un cierto peso, y cada neurona tiene asociada una *función de activación*, que transforma la información recibida de forma no lineal y la envía a las neuronas de la siguiente capa. Si todas las neuronas de una capa están conectadas con cada una de las neuronas de la siguiente, se dice que es una *red apilada*. El entrenamiento del modelo consiste en ir actualizando todos los pesos implicados en los distintos pasos del proceso, con el objetivo de minimizar una cierta *función de pérdida*, que actúa como medida de calidad de las predicciones obtenidas.

En la Figura 2.8 se muestra una representación gráfica de una red neuronal apilada compuesta por dos capas ocultas. La capa de entrada está formada por p neuronas, una por cada variable predictora $X_1, \ldots, X_p$. En los enlaces a la primera y segunda capa oculta (con K_1 y K_2 neuronas, respectivamente) se sitúan las matrices de pesos $\boldsymbol{W}_1$ y $\boldsymbol{W}_2$. Las neuronas de las capas ocultas cuentan con funciones de activación denotadas por $A_1^{(1)}, \ldots, A_{K_1}^{(1)}$ y $A_1^{(2)}, \ldots, A_{K_2}^{(2)}$, que no son más que transformaciones no lineales de combinaciones lineales de las variables de entrada (dadas por los pesos $\boldsymbol{W}_1$ y $\boldsymbol{W}_2$, según corresponda). En los enlaces a la capa de salida (compuesta por 10 neuronas) se sitúa el vector de pesos $\boldsymbol{B}$, y finalmente las transformaciones $f_0(X), \ldots, f_9(X)$ dan lugar a las predicciones $Y_0, \ldots, Y_9$.

Se proporciona a continuación una descripción más detallada de cómo se definen las distintas operaciones implicadas en la construcción de las redes neuronales.

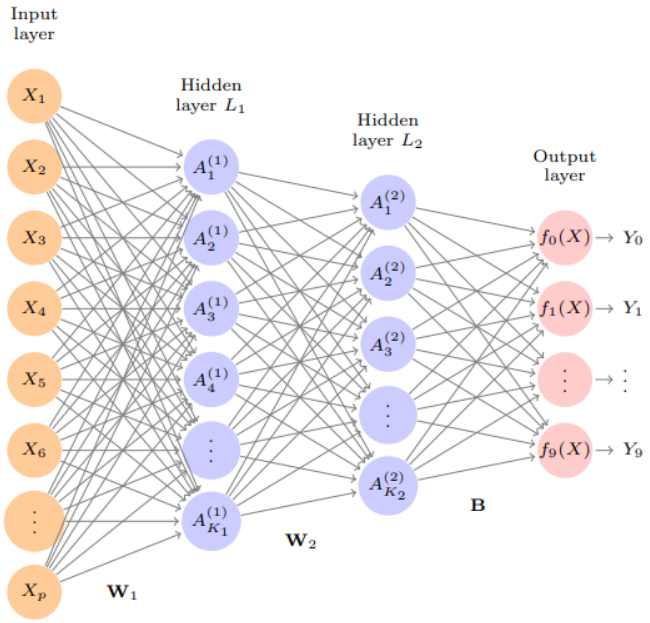


Figura 2.8: Ejemplo de red neuronal apilada con 2 capas ocultas. Imagen de James *et al.* (2023).

Para simplificar, nos centraremos en la estructura de red más simple, conocida como *single-hidden-layer feedforward network*, o también *single layer perceptron*. Es una red neuronal *feedforward*, es decir, la información fluye desde la capa de entrada hasta la de salida, sin bucles ni enlaces hacia atrás, y está compuesta por una única capa oculta. Sean $\boldsymbol{X} = (X_1, \ldots, X_p)$ el vector de las p variables predictoras, por lo que habrá p neuronas en la capa de entrada, el número de neuronas K de la capa oculta, sus funciones de activación asociadas A_k, $k = 1, \ldots, K$, la predicción Y obtenida por el modelo (la capa de salida tendrá entonces una única neurona) y los pesos para los enlaces de la red, $\boldsymbol{W} = \{w_{kj}\}$, $k = 1, \ldots, K$, $j = 0, \ldots, p$, para los enlaces a la capa oculta y $\boldsymbol{B} = (\beta_0, \beta_1, \ldots, \beta_K)$ para los enlaces a la capa de salida. El modelo se construye en dos etapas:

- Primero, se definen las variables ocultas h_k, que son combinaciones lineales de las variables predictoras dadas por los parámetros $\{w_{kj}\}$ y transformadas por una función no lineal $g(z)$ (la función de activación). Así, las activaciones A_k para las neuronas de la capa oculta serán:

$$A_k \;=\; h_k(\boldsymbol{X}) \;=\; g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right), \quad k = 1 \ldots, K. \tag{2.2}$$

- Las activaciones obtenidas en la capa oculta son los valores de entrada para la neurona de la capa de salida, obteniendo una combinación lineal de las mismas a partir de los pesos $\boldsymbol{B}$:

$$Y \;=\; f(\boldsymbol{X}) \;=\; \beta_0 + \sum_{k=1}^{K} \beta_k A_k. \tag{2.3}$$

Las funciones de activación no lineales de la capa oculta son esenciales, pues permiten capturar relaciones complejas entre las variables. Aunque existen diversos tipos, las más populares actualmente son la *función sigmoide* (que toma valores en $(0, 1)$) y la *función tangente hiperbólica* (que toma valores en $(-1, 1)$), definidas como:

$$\phi(z) = \frac{1}{1+e^{-z}}\ ,\ z \in \mathbb{R}\ ; \quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\ ,\ z \in \mathbb{R}. \tag{2.4}$$

Todos los parámetros implicados en las Ecuaciones 2.2 y 2.3 se estiman a partir de los datos, y se van actualizando a medida que se entrena la red. El objetivo es minimizar una cierta función de pérdida, como el *error absoluto medio* (*MAE*) (A.7) o el *error cuadrático medio* (*MSE*) (A.8), a través de un algoritmo de descenso de gradientes llamado *backpropagation*. Este método utiliza las derivadas parciales de los errores de predicción con respecto a los pesos y converge la mayoría de las veces a un óptimo local. Con este algoritmo se entrenan habitualmente las redes neuronales, empleando la siguiente metodología: se agrupan los datos en lotes llamados *batches*, con un tamaño que debería ser preferentemente una potencia de 2, y se denomina *epoch* a cada una de las veces que el algoritmo procesa todos los lotes (la totalidad de los datos). En cada iteración, el parámetro *learning rate* controla el ratio de aprendizaje. Para una explicación más detallada puede consultarse James *et al.* (2023).

Debido a la gran cantidad de parámetros involucrados, las redes neuronales son modelos que tienden al sobreajuste. Para reducirlo, en el entrenamiento se emplean técnicas como el *dropout learning* o el *early stopping*. En el primer caso, cada vez que se procesan los datos (es decir, en cada *epoch*) se elimina aleatoriamente una fracción ϕ de las neuronas de una capa, reescalando los pesos de las demás neuronas convenientemente. Esta fracción se conoce como *dropout rate* y evita que los nodos se vuelvan "sobreespecializados". En la práctica, se asigna un peso cero a las neuronas eliminadas para no tener que modificar la arquitectura del modelo en cada iteración (James *et al.*, 2023). Por otra parte, la técnica de *early stopping* consiste en parar el entrenamiento cuando no se produzca una mejora significativa de la función de pérdida para los datos de validación después de un cierto número de *epochs* (Torres, 2020). Es necesario fijar tanto la métrica que se desea monitorizar como como el número de *epochs* para verificar si se produce una mejora o no (*patience*).

Tras introducir los principales conceptos y técnicas relacionadas con las redes neuronales artificiales, se explica ahora en qué consisten las redes neuronales recurrentes.

Redes neuronales recurrentes

Las *redes neuronales recurrentes* son un tipo de red neuronal artificial especialmente diseñadas para tratar datos secuenciales, como textos, audios o series temporales. Dicho de otra forma, son una generalización de las redes *feedforward* al caso en el que los datos de entrada son secuencias. Así, una entrada a la red es una secuencia de longitud L, $X = \{X_1, \ldots, X_L\}$, mientras que la salida puede ser también una secuencia de una determinada longitud o un escalar. Una característica distintiva frente a las redes *feedforward* es que los nodos de las capas ocultas no solo se conectan a los nodos de las siguientes capas, sino que pueden interactuar entre sí, por ejemplo, permitiendo bucles o "apuntando hacia atrás en el tiempo" (Torres, 2020). En el caso de las series temporales, esta interacción permite modelizar la dependencia temporal de los datos.

En la Figura 2.9 se puede ver una representación gráfica de la red neuronal recurrente más sencilla, formada por una sola neurona recurrente en la capa oculta, donde los datos de entrada son una secuencia temporal formada por las observaciones $\{X_l\}_{l=1}^{L}$ para L instantes de tiempo y la predicción de salida es un número. El modelo procesa esta secuencia de forma que cada vector X_l va pasando a la neurona recurrente de forma secuencial, por lo que las activaciones A_l se van actualizando tomando como información de entrada tanto el vector X_l del instante correspondiente como la activación A_{l-1} calculada en el instante anterior. Cada A_l se utiliza en la capa de salida para obtener la predicción O_l. En general, para calcular Y solo nos va a interesar la última predicción O_L. Los pesos $\boldsymbol{W}$, $\boldsymbol{U}$ y $\boldsymbol{B}$ son los mismos cada vez que se procesa un vector de la serie y no dependen del tiempo, lo que se conoce como *weight sharing*.

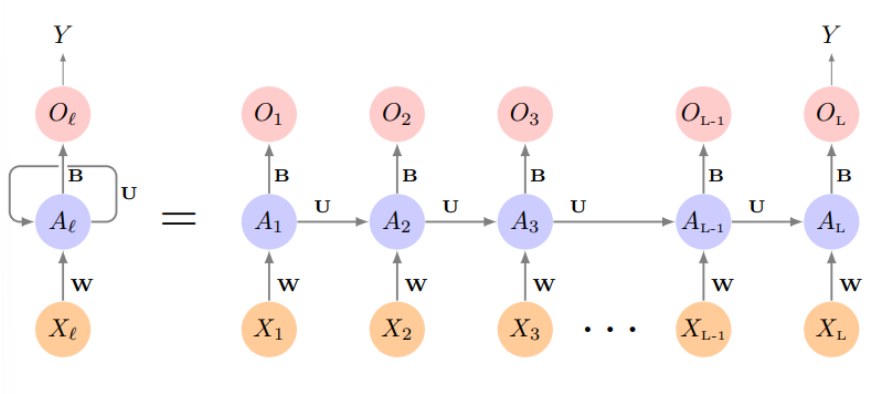


Figura 2.9: Esquema de una red neuronal recurrente con una capa oculta compuesta por una sola neurona recurrente. A la izquierda, una representación compacta de la red; a la derecha, una representación más explícita. Imagen obtenida de James *et al.* (2023).

Más formalmente, supóngase que se tiene una secuencia temporal de dimensión p, por lo que cada instante l tiene la forma $X_l = (X_{l1}, \ldots, X_{lp})$. La capa oculta está compuesta por K neuronas recurrentes con funciones de activación $A_l = (A_{l1}, \ldots, A_{lK})$ para cada instante l. La matriz $\boldsymbol{W} = \{w_{kj}\}$, $k = 1, \ldots, K$, $j = 1, \ldots, p$, representa los pesos para los enlaces que salen de la capa de entrada, la matriz $U = \{u_{ks}\}$, $k, s = 1, \ldots, K$, contiene los pesos para los enlaces de la capa oculta y el vector $\boldsymbol{B} = (\beta_0, \ldots, \beta_K)$ representa los pesos para los enlaces que van a la capa de salida. De esta forma, las funciones de activación para cada instante de tiempo l y cada neurona k se obtienen como:

$$A_{lk} = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{lj} + \sum_{s=1}^{K} u_{ks} A_{l-1,s}\right), \tag{2.5}$$

con $g(z)$ una función no lineal como las de la Ecuación 2.4. Por otra parte, las predicciones que se obtienen en cada paso del proceso secuencial son de la forma:

$$O_l = \beta_0 + \sum_{k=1}^{K} \beta_k A_{lk}. \tag{2.6}$$

Como ya se ha mencionado, los pesos no varían según el instante de tiempo l. Al ser una función de las entradas obtenidas en instantes de tiempo anteriores, las activaciones A_l van acumulando la información que se ha visto previamente. De esta forma, se puede decir que una neurona recurrente "tiene memoria". Se denomina *celda de memoria* a la parte de una neurona que preserva un estado o una cierta información a través del tiempo.

Al igual que en el caso de las redes neuronales artificiales, el proceso de entrenamiento de una red neuronal recurrente se realiza a través del método de *backpropagation*, que en este contexto se modifica ligeramente y recibe el nombre de *backpropagation through time*. Para una explicación más detallada, puede consultarse Williams y Zipser (1995).

Redes *LSTM*

Las redes *Long-Short Term Memory* son un tipo de red neuronal recurrente que busca ampliar la memoria de las neuronas para recordar la información recibida durante un período de tiempo más largo. Se puede decir que las neuronas de las capas ocultas de las redes *LSTM* tienen celdas de memoria

"bloqueadas", de forma que estas deciden cuándo abrir sus puertas a la nueva información recibida en función de la importancia que le otorgan, que viene dada por unos ciertos pesos. En cada neurona *LSTM* hay tres puertas que dan paso a las celdas de memoria: la puerta de entrada o *input gate*, la puerta de olvido o *forget gate* y la puerta de salida o *output gate*. Cada una de estas puertas determina si permite o no una nueva entrada, si deja que la información recibida intervenga en la salida del instante actual o si elimina la información que ya no es relevante, respectivamente.

La Figura 2.10 muestra un esquema de una celda de memoria *LSTM*. Esta arquitectura permite el almacenamiento de información a corto y largo plazo, lo que resulta muy eficaz a la hora de modelizar la dependencia secuencial de las series de tiempo. Se presentan ahora las funciones de activación para las distintas puertas de la celda de memoria, tal como se definen en Graves (2013):

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\ c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ h_t &= o_t \tanh(c_t) \end{aligned} \tag{2.7}$$

Las funciones i_t, f_t y o_t son las activaciones de las puertas de entrada, olvido y salida para el instante t, respectivamente, mientras que c_t es la activación de la celda de memoria. La función σ es habitualmente la función sigmoide, y tanh es la función tangente hiperbólica (2.4). La variable oculta que representa la activación de la neurona es h_t. Los pesos implicados en las Ecuaciones 2.7 (representados por W y b con sus correspondientes subíndices) no varían según el instante de tiempo, y son los parámetros que se van actualizando en el entrenamiento de la red, llevado a cabo por el método de *backpropagation through time*.

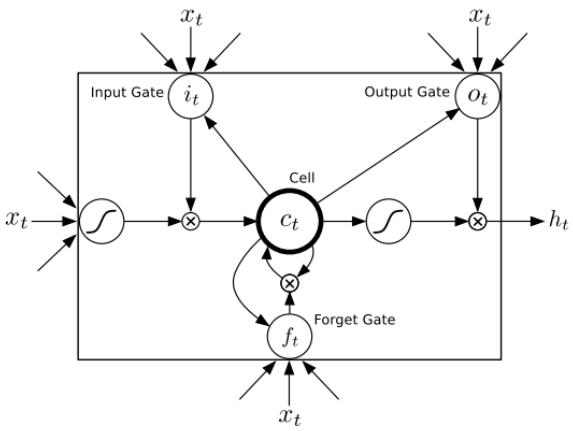


Figura 2.10: Celda de memoria *LSTM*. Imagen obtenida de Graves (2013).

Modelo *LSTM-AD*

Presentados brevemente los conceptos necesarios para entender cómo funcionan las redes *LSTM*, se procede a explicar en qué consiste el método *LSTM-AD* de detección de anomalías. Considérese una serie de tiempo multivariante de dimensión m, en la que cada observación en el instante t se denota por $\boldsymbol{x}_t = (x_{1t}, \ldots, x_{mt}) \in \mathbb{R}^m$.

El objetivo de la fase de predicción es predecir un cierto número l de valores futuros para d variables de la serie, siendo $1 \leq d \leq m$. En este caso el interés se centra en predecir los valores de todas las variables en el instante siguiente al actual, por lo que $l = 1$ y $d = m$. Se utilizará la técnica de ventanas deslizantes con paso 1 para construir secuencias de una cierta longitud K establecida, que serán las secuencias de entrenamiento del modelo. Si estamos en el instante t, utilizaremos su ventana asociada para predecir los valores de las variables en el instante $t + 1$. Los instantes que componen estas secuencias reciben el nombre de retardos o *lags*.

Al tratar con un modelo semisupervisado, los datos de entrenamiento no deben contener anomalías en orden a modelar únicamente el comportamiento normal. En el modelo original (Malhotra *et al.*, 2015), las secuencias de entrenamiento se dividen en cuatro subconjuntos independientes: *entrenamiento normal* (s_N), *validación normal 1* (v_{N1}), *validación normal 2* (v_{N2}) y *test normal* (t_N). A su vez, las secuencias de test, que sí deben contener anomalías, se dividen en dos subconjuntos: *validación anómala* (v_A) y *test anómalo* (t_A). En la primera etapa del método, se entrena un modelo compuesto por redes *LSTM* apiladas con el objetivo de minimizar una cierta función de pérdida que mida las diferencias entre los valores reales y sus correspondientes predicciones, es decir, la calidad de las predicciones. Como función de pérdida se utilizará el *error cuadrático medio* o *MSE* (A.8). En el entrenamiento, se emplean las secuencias s_N para aprender el comportamiento normal de la serie y las secuencias v_{N1} para la validación y la técnica de *early stopping*.

En cuanto a la estructura de las capas de la red: la capa de entrada está compuesta por m neuronas, una por cada una de las m variables; la capa de salida tiene $d \times l$ neuronas, una por cada uno de los l valores a predecir para cada dimesión d y, como habíamos supuesto que $l = 1$ y $d = m$, la capa de salida tendrá también m neuronas. Además, habrá dos capas ocultas con un cierto tamaño o número de neuronas, todas ellas conectadas por conexiones recurrentes. Como función de activación para las capas ocultas se considera la función sigmoide. Las capas de la red *LSTM* están apiladas, de forma que cada neurona de la capa oculta inferior está conectada con todas las neuronas de la capa oculta superior a través de conexiones *feedforward* (dirección única hacia adelante).

En la segunda etapa del método se lleva a cabo la fase de detección de anomalías. Los errores de predicción $\boldsymbol{e}_t$ se pueden calcular como la diferencia en valor absoluto entre el valor real $\boldsymbol{x}_t$ y su predicción $\hat{\boldsymbol{x}}_t$, de forma que $\boldsymbol{e}_t = |\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t|$. Los errores con los datos formando las secuencias de validación v_{N1} se utilizan para estimar los parámetros $\boldsymbol{\mu}$ y $\boldsymbol{\Sigma}$ de una distribución normal multivariante $\mathcal{N}_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$(A.6) por el método de máxima verosimilitud (A.9) (asumiendo, por tanto, la independencia de los errores).

La verosimilitud $p(t)$ de observar un error $\boldsymbol{e}_t$ en los datos de test viene dada por el valor de la distribución normal estimada previamente en $\boldsymbol{e}_t$. En la práctica, se utiliza el logaritmo de la verosimilitud como *anomaly score* a_t para un punto $\boldsymbol{x}_t$. Así, dado un cierto umbral θ, un punto $\boldsymbol{x}_t$ se clasifica como anómalo si:

$$a_t \;=\; -\log(p(t)) \;>\; \theta. \tag{2.8}$$

Los subconjuntos de validación v_{N2} y v_A son usados para seleccionar θ de forma que se maximice el F_β-*Score*, una métrica definida como:

$$F_\beta\text{-}Score = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \tag{2.9}$$

El F_1-*Score* corresponde al caso particular $\beta = 1$. En Malhotra *et al.* (2015) se selecciona $\beta << 1$, en concreto $\beta = 0.1$, para dar una mayor importancia al *Precision* frente al *Recall*, pues interesa penalizar la detección de falsos positivos, sobre todo si los datos de test contienen pocas anomalías.

El modelo *LSTM-AD* para detección de anomalías en series temporales multivariantes pertenece al grupo de los modelos de *deep learning*, por lo que su estructura es mucho más compleja que la de los métodos de *Isolation Forest* y *k Nearest Neighbors*, resultando también en un tiempo de computación mayor. Sin embargo, presenta otras ventajas: al ser un método semisupervisado predictivo, tiene la capacidad de identificar relaciones complejas entre las variables al mismo tiempo que considera la dependencia secuencial, aprendiendo los patrones normales de la serie y apoyándose en la periodicidad o estacionalidad de los datos. De esta forma, se obtiene un método robusto frente al ruido y las anomalías espúreas (Schmidl *et al.*, 2022). Además, es un método adecuado para trabajar con series de alta dimensión, aunque teniendo en cuenta que es sensible a diferentes en escala de las variables. Al igual que *Isolation Forest*, una desventaja de este método es la falta de interpretabilidad del modelo debido a la gran cantidad de parámetros involucrados en el mismo.

2.5.4. *EncDec-AD*

El modelo *EncDec-AD* o *LSTM-based Encoder-Decoder for Anomaly Detection* es un método semisupervisado basado en reconstrucción propuesto por Malhotra *et al.* (2016) para detectar anomalías en series temporales multivariantes. Los modelos *LSTM Encoder-Decoder* son ampliamente utilizados en el aprendizaje *sequence-to-sequence* (o *seq2seq*), introducido por Cho *et al.* (2014) y Sutskever *et al.* (2014). El objetivo de este tipo de aprendizaje es asignar a una secuencia de entrada una cierta secuencia de salida, que no tiene por qué ser de la misma longitud, y se usa principalmente en el ámbito del *Procesamiento del lenguaje natural*[1], en tareas como la traducción automática o el reconocimiento de voz.

El método consiste en entrenar un modelo con una arquitectura de red neuronal *encoder-decoder*, de forma que el *encoder* utiliza una red *LSTM* para asignar una representación vectorial de dimensionalidad fija a una secuencia de entrada, mientras que el *decoder* usa también una red *LSTM* para obtener la secuencia de salida a partir de la representación vectorial obtenida por el *encoder*. Se dice que la secuencia de salida es una *reconstrucción* de la secuencia de entrada. El modelo se entrena únicamente con secuencias normales de la serie, asumiendo que, una vez entrenado, no será capaz de reconstruir una secuencia anómala correctamente. De esta forma, los datos anómalos tendrán errores de reconstrucción mayores que los datos normales, pudiendo usar estos errores para identificar anomalías en los datos de test.

Antes de explicar este método con más detalle, veamos cómo funciona la red neuronal con estructura de *encoder-decoder*.

Arquitectura *Encoder-Decoder*

La arquitectura *RNN Encoder-Decoder* se introduce en Cho *et al.* (2014) para la tarea de traducción automática, mientras que en Sutskever et al. (2014) se adapta para utilizar redes *LSTM*. Se usarán ambas referencias para el desarrollo de este apartado, así como Goodfellow *et al.* (2016).

Con redes neuronales recurrentes, la idea del *encoder-decoder* es como sigue. El *encoder* procesa la secuencia de entrada $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_1}\}$ y emite una representación vectorial $\boldsymbol{c}$ asociada, llamada *contexto*. El *decoder* utiliza esta representación para generar la secuencia de salida $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T_2}\}$. En el contexto de series temporales, las secuencias de entrada y salida tendrán la misma longitud, por lo que $T_1 = T_2$.

[1] El Procesamiento del Lenguaje Natural o *NLP* es la rama de la Inteligencia Artificial que combina la lingüística computacional con los modelos estadísticos de *machine learning* para que las máquinas y ordenadores puedan reconocer, comprender y generar texto o voz. https://www.ibm.com/es-es/topics/natural-language-processing

Tanto el *encoder* como el *decoder* están compuestos por redes neuronales recurrentes, entrenadas conjuntamente para aprender la distribución de probabilidad condicionada de la secuencia de salida dada una secuencia de entrada, esto es:

$$P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T_2} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_1}),$$

para todas las secuencias $\boldsymbol{x}$ e $\boldsymbol{y}$ de los datos de entrenamiento. El *encoder* procesa la secuencia de entrada elemento a elemento a través de la red neuronal recurrente y, a medida que procesa cada instante, el estado oculto o *hidden state* se va actualizando según la siguiente ecuación:

$$\boldsymbol{h}_E^t = f(\boldsymbol{h}_E^{t-1}, \boldsymbol{x}_t) \ , \quad t = 1, \ldots, T_1,$$

siendo f una función de activación no lineal, habitualmente la función sigmoide (2.4). El último estado oculto del *encoder*, $\boldsymbol{h}_E^{T_1}$, es la representación $\boldsymbol{c}$ de la secuencia de entrada que se introduce al *decoder*. El *decoder* se entrena ahora para predecir cada instante $\boldsymbol{y}_t$ de la secuencia de salida dado el estado oculto $\boldsymbol{h}_D^t$, así como su probabilidad condicionada, tal como se indica a continuación:

$$\begin{aligned} \boldsymbol{h}_D^t &= f(\boldsymbol{h}_D^{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}) \ , \quad t = 1, \ldots, T_2, \\ P(\boldsymbol{y}_t | \boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1, \boldsymbol{c}) &= g(\boldsymbol{h}_D^t, \boldsymbol{y}_{t-1}, \boldsymbol{c}) \ , \quad t = 1, \ldots, T_2, \end{aligned}$$

siendo f y g funciones de activación no lineales. La función seleccionada g tiene que ser válida para obtener probabilidades, por lo que se suele emplear la función *softmax* (también llamada *función exponencial normalizada*). Esta función permite pasar de un vector $\boldsymbol{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$ a un vector $\sigma(\boldsymbol{z}) = (\sigma(\boldsymbol{z})_1, \ldots, \sigma(\boldsymbol{z})_K) \in [0,1]^K$ y se define como:

$$\sigma(\boldsymbol{z})_k = \frac{e^{z_k}}{\sum_{i=1}^K e^{z_i}} \ , \quad \boldsymbol{z} \in \mathbb{R}^K \ , \quad k = 1, \ldots, K.$$

Teniendo en cuenta todo lo anterior, puede demostrarse la siguiente igualdad:

$$P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T_2} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_1}) = \prod_{t=1}^{T_2} P(\boldsymbol{y}_t | \boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1, \boldsymbol{c}).$$

El *encoder* y el *decoder* se entrenan para maximizar la siguiente log-verosimilitud condicionada:

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log(p_{\boldsymbol{\theta}}(\boldsymbol{y}_n | \boldsymbol{x}_n)),$$

siendo $\boldsymbol{\theta}$ los parámetros involucrados en el cálculo de todos los estados ocultos de las redes neuronales recurrentes del modelo, y N el número de pares de secuencias de entrada y salida $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ del conjunto de entrenamiento. El modelo puede entrenarse con un algoritmo de descenso de gradientes, al estilo del *backpropagation through time* que mencionábamos antes para las redes neuronales recurrentes.

La secuencia de salida que devuelve el modelo, es decir, la reconstrucción de la secuencia de entrada, es aquella que maximiza la verosimilitud condicionada:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} | \boldsymbol{x}).$$

Esta arquitectura puede utilizarse con redes *LSTM* en lugar de redes neuronales recurrentes, introduciendo los cambios correspondientes. La red *LSTM* para el *encoder* y el *decoder* suele ser distinta (distinto número de capas ocultas, por ejemplo). Además, invirtiendo el orden de la secuencia de entrada, se mejora el rendimiento del modelo, tal como se explica en Sutskever *et al.* (2014).

Modelo *EncDec-AD*

Se sigue el método propuesto en Malhotra *et al.* (2016). Supóngase que se tiene una serie de tiempo m-dimensional de longitud T y que esta puede dividirse en secuencias de longitud L empleando la técnica de ventanas deslizantes. El método está compuesto por dos etapas: primero, se entrena el modelo para reconstruir las secuencias de comportamiento normal; posteriormente, se utilizan los errores de reconstrucción para obtener un *anomaly score* para cada punto x_t de las secuencias de test, e identificar así las anomalías a partir del umbral seleccionado.

La arquitectura del modelo es una red *LSTM Encoder-Decoder* que funciona tal como se expuso en el apartado anterior. El único cambio es que se invierte el orden en el que se reconstruye la secuencia de entrada, pues se genera desde el final hasta el inicio. En la Figura 2.11 se muestra una representación del flujo de la red para una secuencia de longitud $L = 3$.

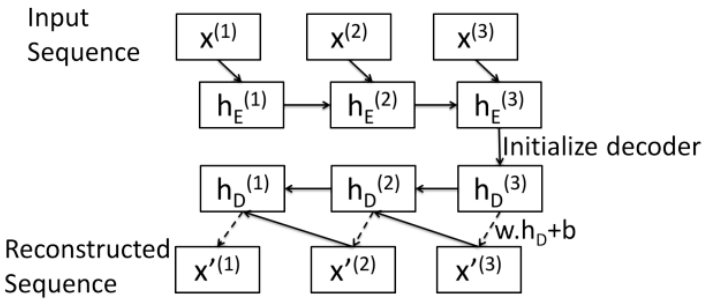


Figura 2.11: Representación del proceso de reconstrucción del *LSTM Encoder Decoder* para una secuencia de longitud $L = 3$. A partir de la secuencia de entrada $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, se obtienen las reconstrucciones $\{\boldsymbol{x}'_1, \boldsymbol{x}'_2, \boldsymbol{x}'_3\}$. Imagen obtenida de Malhotra *et al.* (2016).

El estado oculto del *encoder* en tiempo l se representa como $\boldsymbol{h}_E^l \in \mathbb{R}^c$, siendo c el número de neuronas *LSTM* en la capa oculta del *encoder*. Para este modelo, se consideran siempre arquitecturas en las que el *encoder* y el *decoder* tienen una sola capa oculta con c unidades *LSTM* cada uno. El *encoder* y el *decoder* se entrenan entonces para reconstruir la secuencia en orden inverso, es decir, la secuencia objetivo será $\{\boldsymbol{x}_L, \boldsymbol{x}_{L-1}, \ldots, \boldsymbol{x}_1\}$. Además, el *decoder* tiene una capa con una transformación lineal asociada con matriz de pesos $\boldsymbol{W}_{c\times m}$ y vector $\boldsymbol{b} \in \mathbb{R}^m$, utilizada para obtener las predicciones. Para el tiempo i, se tendría la predicción:

$$\boldsymbol{x}'_i \;=\; \boldsymbol{W}'\boldsymbol{h}_D^i + \boldsymbol{b}.$$

El objetivo del entrenamiento es minimizar una cierta función de pérdida, que será el sumatorio del error cuadrático medio a lo largo del conjunto X de todas las secuencias de entrenamiento:

$$\sum_{\boldsymbol{x}\in X}\sum_{i=1}^{L} \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|^2.$$

En la segunda etapa del método, se obtienen los *anomaly scores* para los datos de test. Tal como se hacía en el modelo *LSTM-AD*, las secuencias normales se dividen en los subconjuntos s_N, v_{N1}, v_{N2} y t_N, mientras que las secuencias de test (que deberían contener anomalías) se dividen en los subconjuntos v_A y t_A. El entrenamiento del modelo se realiza con las secuencias s_N, mientras que el conjunto v_{N1} se utiliza para la técnica de *early stopping*. El error de reconstrucción para un instante i se obtiene como la diferencia en valor absoluto entre el valor real y su reconstrucción, es decir, $\boldsymbol{e}_i = |\boldsymbol{x}_i - \boldsymbol{x}'_i|$.

Los errores de reconstrucción de los datos de las secuencias v_{N1} se utilizan para estimar por máxima verosimilitud (A.9) los parámetros $\boldsymbol{\mu}$ y $\boldsymbol{\Sigma}$ de una distribución normal multivariante $\mathcal{N}_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (A.6). De esta forma, el *anomaly score* a_i para un cierto punto $\boldsymbol{x}_i$ será:

$$a_i \;=\; (\boldsymbol{e}_i - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{e}_i - \boldsymbol{\mu}), \tag{2.10}$$

que no es más que la distancia de Mahalanobis al cuadrado (A.3) entre el vector de error correspondiente y la media de la distribución normal multivariante ajustada. Si $a_i > \theta$ para un umbral θ fijado, el dato $\boldsymbol{x}_i$ se etiqueta como anomalía, y en otro caso, como dato normal. El número c de neuronas *LSTM* en la capa oculta del *encoder* y el *decoder* y el umbral θ pueden seleccionarse de forma que maximicen el F_β-*Score* en las secuencias de validación v_{N2} y v_A, tal como se indica en la Ecuación 2.9.

Al igual que el modelo *LSTM-AD*, el *EncDec-AD* es un método semisupervisado de *deep learning*, en este caso basado en reconstrucción, que es adecuado para trabajar con datos de alta dimensión, teniendo en cuenta de nuevo la sensibilidad a las distintas escalas de las variables. Además, es capaz de aprender las relaciones entre las variables a la vez que considera la dependencia secuencial. Por otra parte, el tiempo de computación es mayor que para los modelos de *machine learning*, y la falta de interpretabilidad es también una desventaja.

$\sim$

A lo largo de este capítulo, se ha hecho una revisión de todos los aspectos teóricos y metodológicos relacionados con la detección de anomalías en series temporales multivariantes. En particular, se han seleccionado cuatro modelos dentro de las distintas categorías posibles para explicar en detalle su funcionamiento. En el Capítulo 4 se aplicará cada uno de estos modelos a los datos de Banca Móvil, que se presentan a continuación en el Capítulo 3.

Capítulo 3

Tratamiento de datos

Como ya se ha explicado en el Capítulo 1, el principal objetivo de este trabajo es emplear los métodos de detección de anomalías en series temporales multivariantes para intentar anticipar posibles caídas de la *app* de Banca Móvil. En este capítulo se presentan los datos disponibles para llevar a cabo esta tarea, obtenidos a partir de distintos canales tecnológicos del banco. Además, se realiza un primer análisis de los mismos y el preprocesamiento oportuno. Se emplea el lenguaje de programación *Python* (Rossum, 1995) para desarrollar todo el código necesario, utilizando principalmente la librería *pandas* (McKinney, 2010) para la manipulación y análisis de datos y la librería *matplotlib* (Hunter, 2007) para la creación y visualización de gráficos.

3.1. Obtención y descripción de los datos

En el Departamento de Monitorización y Operaciones de ABANCA se dispone de una herramienta de monitorización TI llamada *Zabbix* que permite llevar a cabo un seguimiento en tiempo real del estado de los sistemas que componen la infraestructura tecnológica del banco, como pueden ser servidores, bases de datos o aplicaciones, entre otros. En el departamento hay un especial interés por poder anticipar con la mayor antelación posible cuándo se va a producir una caída de Banca Móvil pues, actualmente, no disponen de ningún tipo de método o modelo que permita detectar problemas en la aplicación y generar alertas previas a la caída.

Lo que sí se monitoriza es una métrica llamada `Disponibilidad-Neta-Minuto`, que mide la disponibilidad de la aplicación minuto a minuto en una escala de 0 a 1: un valor de 1 representa un 100 % de disponibilidad, un valor entre 0 y 1 representa un servicio degradado y un valor de 0 indica la indisponibilidad o caída de la aplicación. Según se ha explicado desde el departamento, el servicio degradado no suele tener un valor de disponibilidad menor de 0.6 y no supone un impacto notable para los clientes, pues a menudo solo provoca que la aplicación funcione un poco más lenta y el problema se soluciona relativamente rápido. El interés va a estar, por tanto, en las indisponibilidades, ya que son las que afectan directamente al uso de la aplicación.

Los registros de las distintas métricas en *Zabbix* se van borrando por antigüedad. Se han podido recopilar los datos de disponibilidad de Banca Móvil minuto a minuto desde el 5 de enero de 2024 hasta el 10 de junio de 2024, ambos incluidos. En este período se observan caídas con una duración muy diversa, desde unos pocos minutos hasta 2 horas en el caso más grave. En el registro se descartan aquellos valores que corresponden a intervenciones planificadas, es decir, las relacionadas con el mantenimiento de los servicios de la aplicación. Estas indisponibilidades están programadas y no generan las mismas trazas en los otros sistemas que las verdaderas caídas, por lo que es mejor no tenerlas en cuenta en el análisis del problema. Por esta razón, se imputarán los datos faltantes con el valor 1.

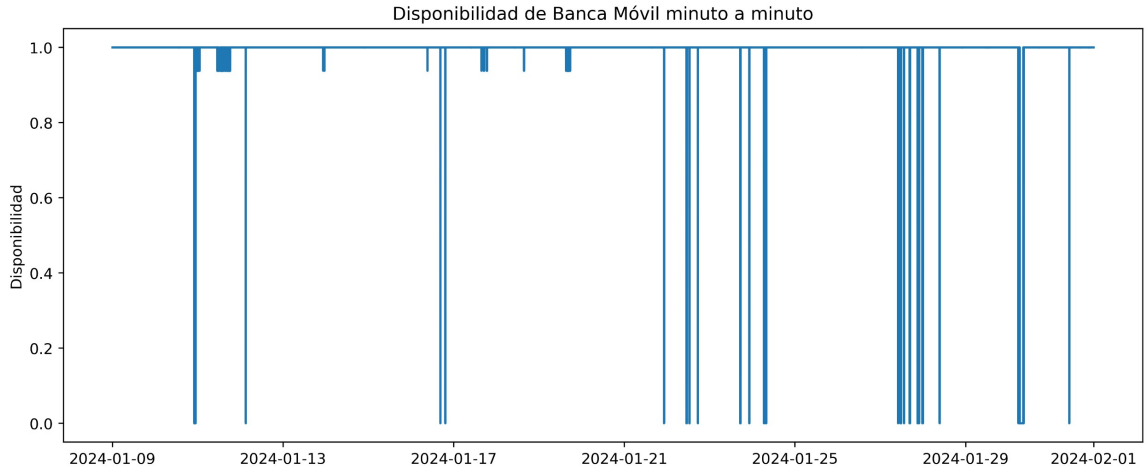


Figura 3.1: Serie de disponibilidad de Banca Móvil minuto a minuto entre el 9 de enero de 2024 y el 31 de enero de 2024, ambos incluidos.

La Figura 3.1 muestra un fragmento de la serie de disponibilidad de Banca Móvil, comprendido entre el 9 y el 31 de enero de 2024. Entre los días 9 y 21 se aprecian muy pocas caídas y varios momentos de servicio degradado, mientras que en los 10 últimos días del mes se observa un mayor número de caídas de la aplicación.

El cálculo de la disponibilidad se obtiene a partir del estado de los diferentes sistemas que dan soporte a la aplicación, como servidores, bases de datos o *API*s. En el *Mapa de canales tecnológicos* del banco se detallan todos los elementos relacionados con Banca Móvil, que a su vez están monitorizados con distintas métricas a través de *Zabbix*. Estos elementos se agrupan en tres categorías: Infraestructura, Operaciones y Sonda. Muchas de las caídas importantes (de mayor duración) están relacionadas con problemas en alguno de los elementos de Infraestructura, como por ejemplo, la sobrecarga de un servidor o la pérdida de conexión con una base de datos. La parte de Operaciones controla las distintas tareas que pueden hacer los clientes dentro de Banca Móvil, como bloquear una tarjeta, realizar un pago o consultar el saldo de sus cuentas. Los fallos en alguna de estas operaciones pueden acabar provocando la indisponibilidad de la aplicación. Por último se considera la Sonda, empleada para comprobar minuto a minuto si es posible establecer conexión con la aplicación o no, en cuyo caso el valor de disponibilidad pasaría a ser directamente cero. Esta sonda puede resultar bastante sensible, registrando caídas de un solo minuto de duración cuando en realidad todo funciona correctamente. Por este motivo, a la hora de aplicar los modelos en el Capítulo 4 se aplicará una limpieza para considerar solo las indisponibilidades con una duración de al menos dos minutos, siendo de especial interés aquellas caídas que se prolongan por más de cinco minutos.

A partir de la serie de disponibilidad es posible generar un etiquetado de las caídas de Banca Móvil. Basta asignar un 1 a todos los instantes de tiempo con un valor de disponibilidad distinto de cero y un 0 a todos los instantes correspondientes a las indisponibilidades, es decir, aquellos con un valor nulo. Las caídas serán entonces los eventos anómalos que interesa detectar y predecir. Nótese que, aunque lo más habitual es asociar el valor 1 a la clase anómala, en el contexto que nos ocupa parece más adecuado e intuitivo representar las caídas con un 0, respetando así la escala original de la disponibilidad. Este etiquetado se almacena en una variable que recibe el nombre de `disp_bmov` y que identifica la *Ground Truth* con la que se deben comparar las predicciones obtenidas por los diferentes modelos empleados.

Con el propósito de aplicar los modelos de detección de anomalías en este contexto, es necesario construir una serie de tiempo multivariante que contenga datos relacionados con Banca Móvil a nivel de minutos. En el *Mapa de canales tecnológicos* se puede comprobar la gran cantidad de elementos que dan soporte a la aplicación, con relaciones a menudo complejas entre ellos. De hecho, algunos servidores y bases de datos no son exclusivos de la *app*. En el caso de los servidores, es habitual que tengan varias réplicas con el fin de reducir el impacto que pueda tener en uno de ellos un problema en el funcionamiento de Banca Móvil. Además, muchas de las métricas que se registran en *Zabbix* para estos sistemas no están recogidas a nivel de minutos y, como ya se ha comentado, se van borrando por antigüedad. Por todo esto y tras identificar estas dificultades, se decidió que la búsqueda y selección de las variables para construir la serie multivariante se debería centrar en otro tipo de datos.

En ABANCA se utiliza *Teradata* (entre otras plataformas) como sistema de gestión de bases de datos relacionales, es decir, aquellas en las que se estructuran los datos en tablas entre las que es posible establecer vínculos y recuperar información a través de consultas en lenguaje *SQL* (*Structured Query Language*) (Chamberlin y Boyce, 1974). Se han revisado algunas de estas bases de datos para tratar de obtener distintas variables relacionadas con Banca Móvil, todas con observaciones minuto a minuto. A continuación, se exponen estas bases de datos y las variables finalmente consideradas:

- *BMOV*: recoge información de operaciones directamente relacionadas con Banca Móvil.
  - `num_accesos_ok`: recuento de accesos válidos a la *app* (mediante PIN o huella).
  - `num_error_host`: número de errores en el *host* de Banca Móvil.
- *Celebrus*: está compuesta principalmente por datos de las distintas interacciones individuales de los clientes con los servicios web del banco, en particular, con Banca Móvil.
  - `num_logins`: número de veces que los usuarios están en la página de acceso o *login*.
  - `promedio_segundos`: tiempo medio en segundos que tardan los usuarios en entrar a la *app*, es decir, desde que están en la página de *login* hasta que acceden a la página principal.
  - `num_account_activity`: número de veces que los usuarios entran en una de sus cuentas bancarias.
  - `num_success`: número de operaciones de envío de dinero que se realizan correctamente a través de la aplicación.
- *Viewsinq*: es la principal base de datos del banco.
  - `num_val_pin`: número de validaciones correctas de PIN que dan acceso a Banca Móvil.
  - `num_bizum`: recuento del número de *bizums* totales, incluidos los que se realizan a través de la aplicación.

Se han recopilado los datos para el mismo período que el etiquetado de las caídas, es decir, del 5 de enero al 10 de junio de 2024, ambos incluidos, y con la misma frecuencia (minuto a minuto). En total son 8 variables relacionadas principalmente con la navegación de los usuarios a través de Banca Móvil, lo cual permite considerar distintos escenarios previos a una caída y ver cómo afectan al uso de la aplicación. Por ejemplo, puede que se valide correctamente el PIN pero no se consiga acceder a la página principal, aumentando el tiempo de acceso y generando errores en el *host*, o que esté caída alguna de las operaciones pero no haya problemas en el acceso a la página principal de la *app*, entre otras muchas casuísticas.

Para no descartar completamente las métricas registradas en *Zabbix*, también se ha seleccionado la *CPU* media, minuto a minuto, de las réplicas de dos de los principales servidores de la aplicación, a saber `ECCOREBMOV` y `ECRESOBMOV`. Estos porcentajes se encuentran como *CPU(User+System)*, pues tienen en cuenta tanto el consumo debido a los usuarios como al propio sistema. Para estas variables solo estarán disponibles los datos desde el 1 de marzo hasta el 10 de junio de 2024, ambos incluidos.

- `cpu_media_eccorebmov`: porcentaje de CPU media para las 4 réplicas principales de `ECCOREBMOV`.
- `cpu_media_ecresobmov`: porcentaje de CPU media para las 3 réplicas principales de `ECRESOBMOV`.

Considerando todas las variables anteriores, se crean dos series temporales multivariantes equiespaciadas a nivel de minutos: una comprendida entre el 5 de enero y el 10 de junio de 2024 con las ocho variables obtenidas en *Teradata* (en adelante referida como "primera serie"), y otra comprendida entre el 1 de marzo y el 10 de junio de 2024 en la que se incluyen, además, estas dos últimas métricas de los servidores (denominada "segunda serie"). Cada una de las series se almacena en un *DataFrame* de *pandas*, donde las filas corresponden a las observaciones minuto a minuto ordenadas cronológicamente y las columnas hacen referencia a cada una de las variables seleccionadas, ordenadas como sigue: `num_logins` (`V1`), `num_val_pin` (`V2`), `num_accesos_ok` (`V3`), `promedio_segundos` (`V4`), `num_error_host` (`V5`), `num_account_activity` (`V6`), `num_bizum` (`V7`) y `num_success` (`V8`). Para la segunda serie, se añaden `cpu_media_eccorebmov` (`V9`) y `cpu_media_ecresobmov` (`V10`).

3.2. Análisis exploratorio

En primer lugar, se trata la imputación de datos faltantes. En el caso de las variables obtenidas a través de *SQL*, los datos faltantes se relacionan con minutos en los que no ha habido registros de los eventos medidos, siendo el recuento nulo en las correspondientes componentes. Por lo tanto, se imputarán estos datos con el valor 0. Para las dos métricas de los servidores obtenidas en *Zabbix*, los datos faltantes están relacionados principalmente con problemas en el registro, que pueden venir ocasionados por las caídas de Banca Móvil. Para que quede constancia de estas anomalías, se imputarán los datos faltantes también con el valor 0.

Para una primera visualización de los patrones que siguen las variables seleccionadas, se muestra en la Figura 3.2 un fragmento del gráfico secuencial de cada una de las componentes de la segunda serie, correspondiente a la semana del 2 de marzo (sábado) al 8 de marzo (viernes) de 2024. En este período no se registraron caídas, por lo que sería un ejemplo del comportamiento normal de la serie temporal multivariante. Se distinguen tres tipos de variables: las que miden un porcentaje (las dos variables de *CPU*), una variable que mide el tiempo en segundos (`promedio_segundos`) y las que hacen referencia a un recuento de eventos (todas las demás). Así, se puede concluir que es necesario trasladar las distintas variables a la misma escala para poder aplicar los métodos de detección de anomalías, problema que se tratará en la Sección 3.3.

Continuando con la Figura 3.2, se comentan los patrones observados en cada una de las variables. Todas las componentes de la segunda serie, excepto `promedio_segundos` y `num_error_host`, tienen una clara componente estacional diaria, es decir, un patrón similar que se repite cada 24 horas. En este caso, los valores comienzan a descender notablemente en las últimas horas del día para alcanzar registros mínimos durante la madrugada, volviendo a subir por la mañana temprano, donde habitualmente llegan a los valores máximos del día. En el caso de la variable `num_bizum`, los valores más altos parecen alcanzarse más bien en las últimas horas de la tarde.

Para las variables mencionadas también parece intuirse una cierta componente estacional semanal, pues los patrones observados durante el fin de semana (sábado y domingo) son diferentes a los del resto de días de la semana, registrándose valores más bajos en general durante las horas centrales del día, con diferencias también entre sábado y domingo. De nuevo, la variable `num_bizum` se distingue de las demás por no sufrir el efecto del fin de semana (se mantienen valores y patrones similares a los del resto de días de la semana). Estas estacionalidades son lógicas debido a los horarios habituales de trabajo y descanso de la mayoría de los usuarios de la aplicación.

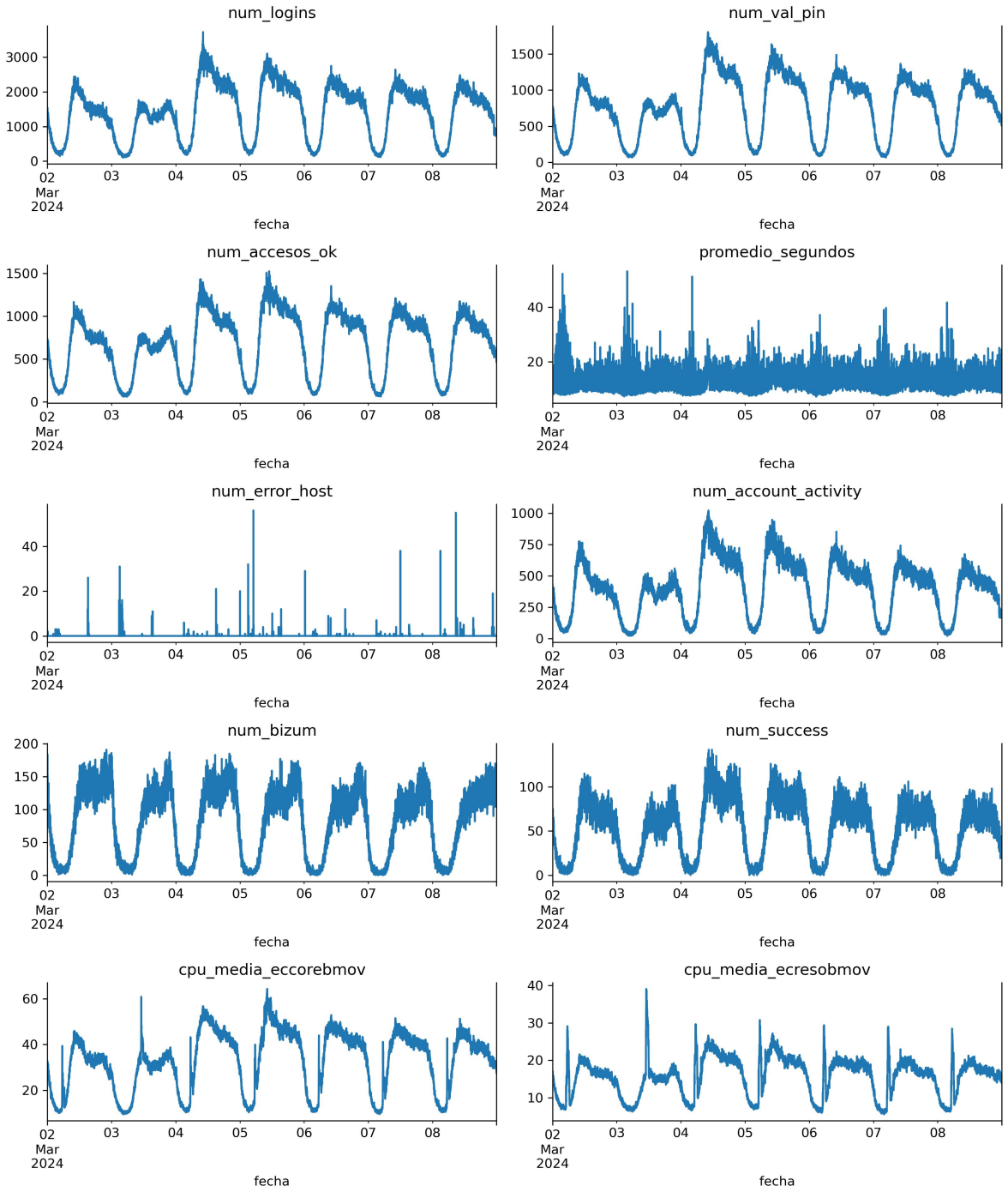


Figura 3.2: Representación gráfica de la serie temporal multivariante con 10 variables para la semana del 2 de marzo (sábado) al 8 de marzo (viernes) de 2024, período en el que no se registraron caídas.

En la Figura 3.2 también se aprecia que las dos variables relacionadas con la *CPU* de los servidores sufren un repunte cada día durante la madrugada. Para `cpu_media_eccorebmov` se produce sobre las 05:30 de la mañana, mientras que para `cpu_media_ecresobmov` ocurre sobre las 05:00, y en ambos casos este evento tiene una duración aproximada de una hora. La excepción se da el domingo, donde este repunte coincide a las 11:00 para ambas variables. Estas subidas se deben al reinicio automático de los servidores, elevando el porcentaje de *CPU* por la parte que corresponde al sistema.

Finalmente, se analizan las variables `promedio_segundos` y `num_error_host`. En el primer caso, al tratarse de valores promedios, las mayores diferencias se aprecian en las horas de madrugada, donde la variabilidad es mayor debido a tener un menor número de observaciones (hay menos personas accediendo a Banca Móvil). En el segundo caso, los errores del *host* no siguen aparentemente ningún patrón por estar sujetos a los fallos que se producen en el uso de la aplicación, los cuales pueden ocurrir a cualquier hora y cualquier día de la semana. De hecho, la mayoría de los valores observados para esta variable son iguales a 0, indicando que no hay errores en el *host* para los minutos correspondientes.

	V1	V2	V3	V4	V5	V6	V7	V8
count	227520.00	227520.00	227520.00	227520.00	227520.00	227520.00	227520.00	227520.00
mean	1386.53	718.61	660.53	14.65	2.86	358.05	76.00	46.80
std	800.21	410.53	373.36	41.99	76.04	220.07	51.05	31.68
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	687.00	340.00	318.00	10.26	0.00	174.00	19.00	14.00
50%	1441.00	747.00	693.00	12.31	0.00	363.00	89.00	51.00
75%	1941.00	1017.00	929.00	14.94	0.00	493.00	117.00	68.00
max	12587.00	8215.00	2647.00	3498.46	4757.00	1464.00	295.00	203.00

Tabla 3.1: Medidas descriptivas para las ocho variables de la primera serie.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
count	146880.00	146880.00	146880.00	146880.00	146880.00	146880.00	146880.00	146880.00	146880.00	146880.00
mean	1419.21	728.07	669.51	14.68	2.59	365.79	78.37	48.47	32.65	16.21
std	810.07	412.63	375.64	44.71	75.54	223.23	52.22	32.43	13.25	5.51
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	720.00	353.00	330.00	10.20	0.00	181.00	20.00	15.00	21.78	12.22
50%	1477.00	758.00	704.00	12.21	0.00	372.00	92.00	53.00	34.37	16.80
75%	1979.00	1027.00	939.00	14.76	0.00	503.00	120.00	71.00	42.50	19.86
max	12587.00	8215.00	2647.00	3498.46	4757.00	1464.00	282.00	203.00	79.28	46.09

Tabla 3.2: Medidas descriptivas para las diez variables de la segunda serie.

Las Tablas 3.1 y 3.2 proporcionan un grupo de estadísticos descriptivos para las variables de la primera y segunda, respectivamente. Específicamente, el recuento de observaciones disponibles, la media, la desviación típica, el mínimo, los cuartiles y el máximo, con valores redondeados a dos decimales, todos ellos obtenidos con el método `describe` de *pandas*. En la primera serie se tienen 227520 observaciones, mientras que la segunda está compuesta por 146880 registros. En ambas tablas se observan las diferentes escalas y rangos (diferencias entre el máximo y el mínimo) de las variables. Además, por tratarse de recuentos, tiempos y porcentajes, no hay valores negativos y el mínimo es 0 para todas ellas. Destacar que la media de `num_error_host` (`V5`) no supera los 3 errores por minuto, poniendo de manifiesto que la cantidad esperada de errores del *host* es muy baja cuando se tiene un comportamiento normal de la aplicación.

A continuación, en las Figuras 3.3 y 3.4 se muestran las correlaciones entre las variables de las dos series, organizadas en un mapa de calor y redondeadas a tres números decimales. Cuanto más intenso sea el color azul en una celda, mayor será la correlación entre las variables correspondientes. De cualquiera de las dos figuras se concluye la ausencia de relación lineal entre las variables `V4` y `V5` (`promedio_segundos` y `num_error_host`) y el resto, con correlaciones muy próximas a cero. Esta ausencia de correlación es congruente con los perfiles de las series componente en la Figura 3.2, donde `V4` y `V5` son las únicas que exhiben patrones sustancialmente distintos. La correlación entre ellas es superior (0.391 en la primera serie y 0.217 en la segunda), aunque muy inferior a las correlaciones entre `V1`, `V2`, `V3` y `V6`, todas ellas por encima de 0.93 en ambas figuras, evidenciando una fuerte asociación lineal. Esto es lógico por tratarse de variables muy relacionadas con el acceso a la aplicación. En particular, `V2` y `V3` (`num_val_pin` y `num_accesos_ok`) tienen la correlación más alta, con valores de 0.990 y 0.989 en las dos series, respectivamente.

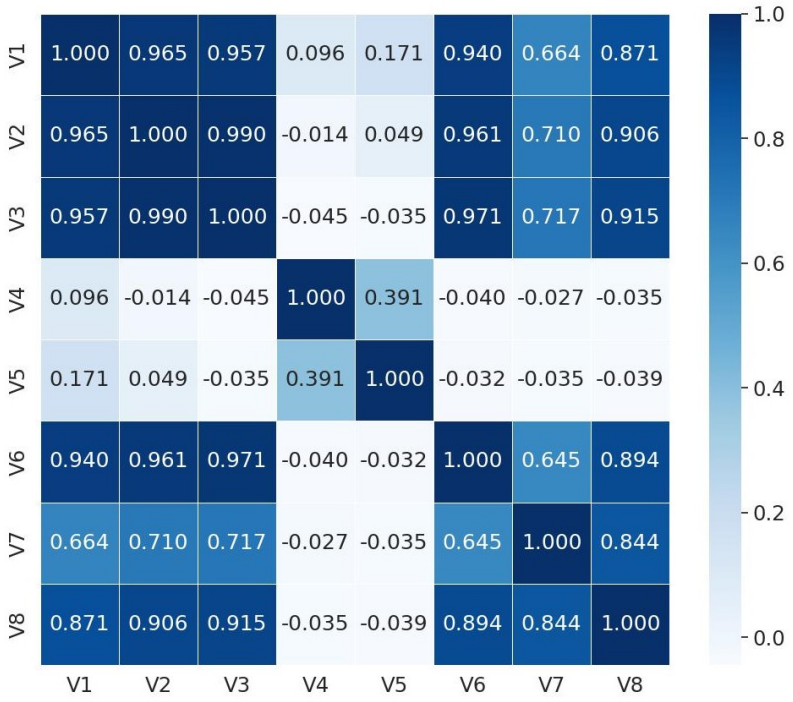


Figura 3.3: Mapa de calor con las correlaciones para las ocho variables de la primera serie. Una mayor intensidad del color azul indica una correlación más fuerte entre las variables correspondientes.

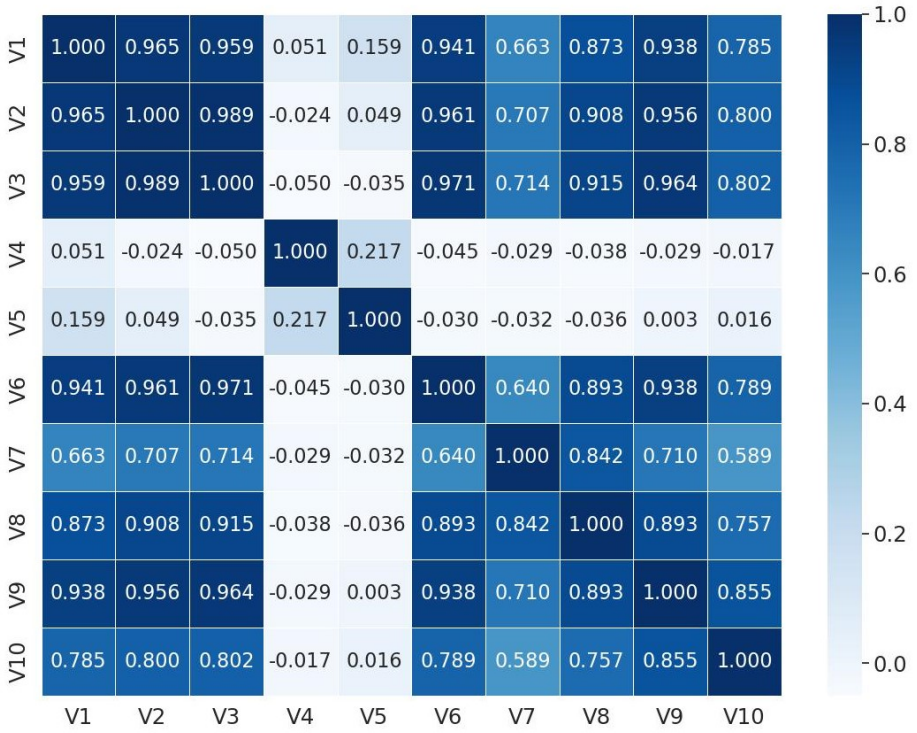


Figura 3.4: Mapa de calor con las correlaciones para las diez variables de la segunda serie. Una mayor intensidad del color azul indica una correlación más fuerte entre las variables correspondientes.

La variable `num_bizum` (`V7`) presenta correlaciones algo inferiores, moviéndose entre 0.58 y 0.85 en ambas series, lo que está en consonancia con las diferencias en el patrón estacional que esta variable muestra en la Figura 3.2. Su mayor correlación, aproximadamente 0.84, se da con `num_success` (`V8`), variable que también hace referencia a operaciones de envío de dinero.

Finalmente, se consideran las variables de *CPU* (`V9` y `V10`). Sus correlaciones con las demás pueden verse en la Figura 3.4, apreciando que el servidor `ECCOREBMOV` (`V9`) presenta correlaciones altas con las variables que tienen componentes estacionales (entre 0.7 y 0.97), mientras que para el servidor `ECRESOBMOV` (`V10`) los valores son más moderados (entre 0.58 y 0.86), siendo precisamente `V9` la variable con la que está más relacionada.

Se examina a continuación la estructura de autocorrelación para las componentes de la segunda serie (para la primera se obtienen las mismas conclusiones). Las autocorrelaciones para cada una de las variables se representan utilizando la función `plot_acf`[1] de la librería *tsaplots* de *Python*. En la Figura 3.5 se muestran dichas autocorrelaciones para un total de 60 retardos o *lags*. Como se puede observar, toman valores muy altos y decrecen muy lentamente a 0, sobre todo las que corresponden a componentes con patrones periódicos, manteniéndose por encima de 0.75 (a excepción de `cpu_media_ecresobmov`). Se concluye así que todas estas componentes son series no estacionarias y, por tanto, la serie temporal multivariante tampoco es estacionaria. Como ninguno de los métodos aplicados en el Capítulo 4 asume la estacionariedad de la serie, no es necesario realizar ninguna transformación.

[1] https://www.statsmodels.org/dev/generated/statsmodels.graphics.tsaplots.plot_acf.html

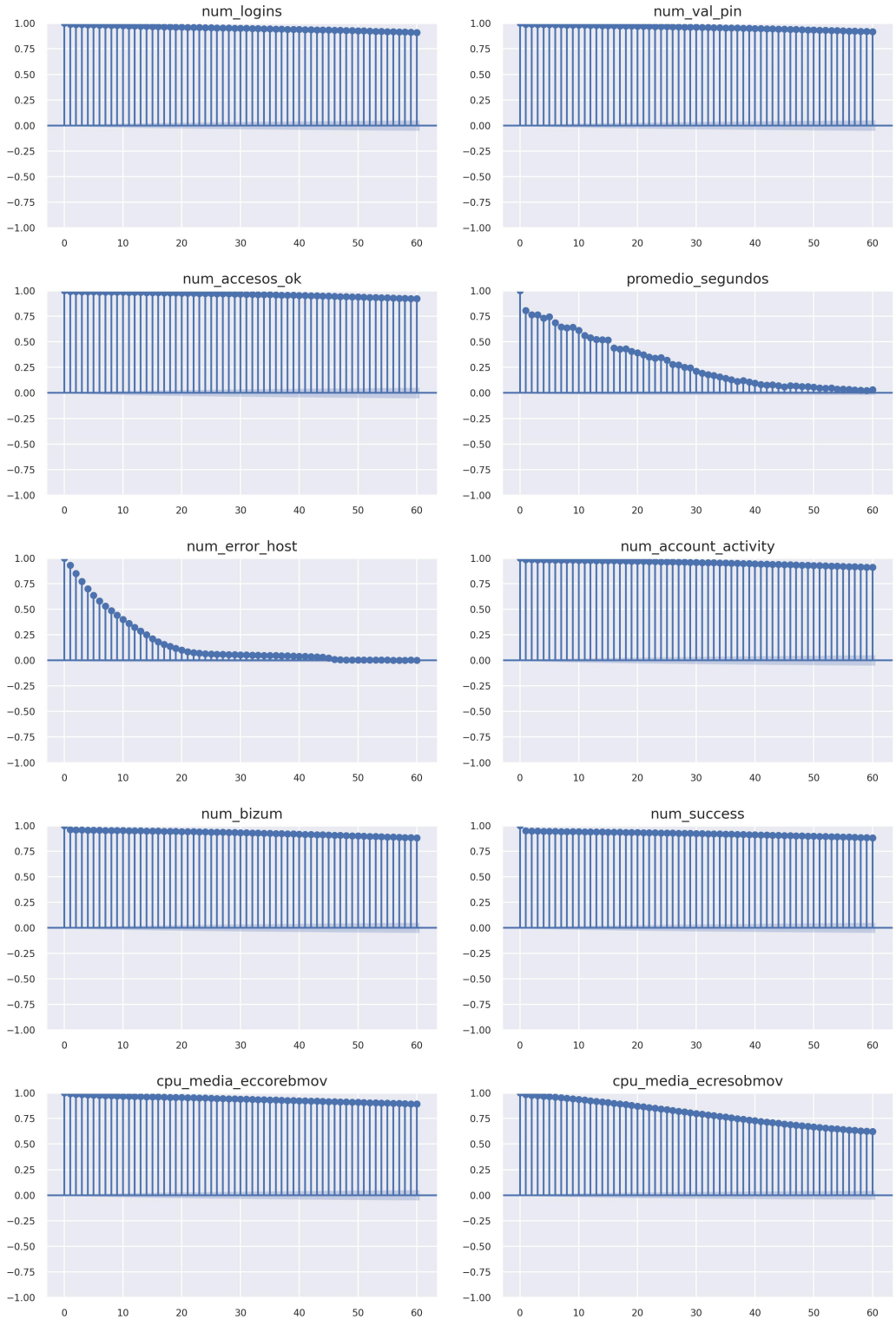


Figura 3.5: Representación gráfica de las autocorrelaciones para 60 *lags* de las diez componentes de la serie temporal multivariante.

A continuación, se pone el foco en analizar la evolución de las caídas de Banca Móvil, para lo que se calcula la proporción de instantes en los que la aplicación estuvo caída sobre diferentes períodos de tiempo. Así, en el mes de enero (del día 5 al 31) el porcentaje de instantes de caída fue 0.692 %. El mes de febrero resultó muy tranquilo y sin apenas caídas, con un porcentaje muy bajo de 0.077 %. Mientras marzo y mayo presentaron porcentajes similares del 0.352 % y 0.361 %, respectivamente, en abril el porcentaje de instantes de caída se incrementó hasta el 0.734 %. Por último, en los diez primeros días de junio se tuvo un valor de 1.167 %. Es relevante mencionar que entre el 8 y el 10 de ese mes se produjeron varios fallos en la aplicación de Banca Móvil, de ahí el incremento de la proporción de instantes de caída para esos primeros días de junio. En cualquier caso, puede observarse que en los meses comentados el porcentaje de caídas no supera el 0.8 %, lo que supone un número muy pequeño de observaciones y, como es habitual en un contexto de detección de anomalías, las clases se encuentran considerablemente desbalanceadas.

Para concluir esta sección, se incluyen tres figuras que permiten visualizar el impacto sobre las series de caídas producidas a lo largo de los meses estudiados. En la Figura 3.6 se muestran los valores de las componentes de la primera serie (en naranja) durante las caídas ocurridas entre las 23:30 y las 00:00 del día 27 de enero de 2024 (disponibilidad escalada de Banca Móvil en azul). Son dos caídas con una duración de 3 y 9 minutos, respectivamente. Se observa que las variables más afectadas en los minutos previos a la primera caída son aquellas relacionadas con operaciones, es decir, `num_bizum` y `num_success`. También se aprecian anomalías en las variables `num_error_host` y `promedio_segundos`, con incrementos de sus registros en los instantes anteriores a la caída. Sin embargo, en las demás variables no se producen variaciones tan notorias en esos instantes previos. De hecho, para `num_account_activity` parece que las caídas pasan desapercibidas. Por lo tanto, podría deducirse que estas caídas tuvieron algún tipo de relación con los elementos de la categoría de Operaciones.

En la Figura 3.7 se muestran las variables de la segunda serie durante las caídas del 21 de marzo de 2024 entre las 10:30 y las 12:30, con hasta cuatro caídas de corta duración previas a una caída más grande que se prolonga por 15 minutos. Se sabe que este cese de funcionamiento estuvo directamente relacionado con la caída de las *API*s que utiliza Banca Móvil. Las caídas de corta duración no parecen afectar al funcionamiento de la aplicación (podrían deberse, quizás, a la sonda), mientras que todas las variables están claramente afectadas por la caída más larga. En concreto, `num_logins` y `num_val_pin` tienen un repunte de sus valores, mientras que `num_accesos` sufre un descenso notable hasta alcanzar valores próximos o iguales a 0, lo que puede traducirse en que los usuarios están intentando una y otra vez entrar a la aplicación, consiguiendo validar el *PIN* pero encontrándose con un error en el acceso. Como consecuencia, también aumentan los errores del *host* y el tiempo de acceso. La *CPU* de los dos servidores parece tener un comportamiento contrario entre ellos: para `ECCOREBMOV` se produce una bajada del porcentaje, mientras que los valores suben para `ECRESOBMOV`. Esto podría deberse a que dan soporte a partes distintas de la aplicación y no tienen por qué verse afectados de la misma manera.

Por último, la Figura 3.8 representa el impacto de una caída de 21 minutos de duración. De nuevo, todas las variables se ven afectadas, pero no se aprecia un margen de antelación mayor a uno o dos minutos. En este caso, aumentan los *logins* y no se están validando los *PIN*, por lo que las variables relacionadas con el acceso y las operaciones descienden prácticamente a 0. Al contrario que en el ejemplo anterior, la *CPU* de los servidores sigue el mismo patrón en ambos casos, con un descenso de los porcentajes.

El análisis visual de estas y otras caídas permite concluir que prácticamente todas las variables consideradas sufren variaciones o desviaciones en su comportamiento normal cuando se produce una caída, comenzando estas anomalías habitualmente entre 1 y 5 minutos antes de obtener la detección efectiva con la disponibilidad y concluyendo también uno o dos minutos antes de que la disponibilidad vuelva a tener un valor de 1.

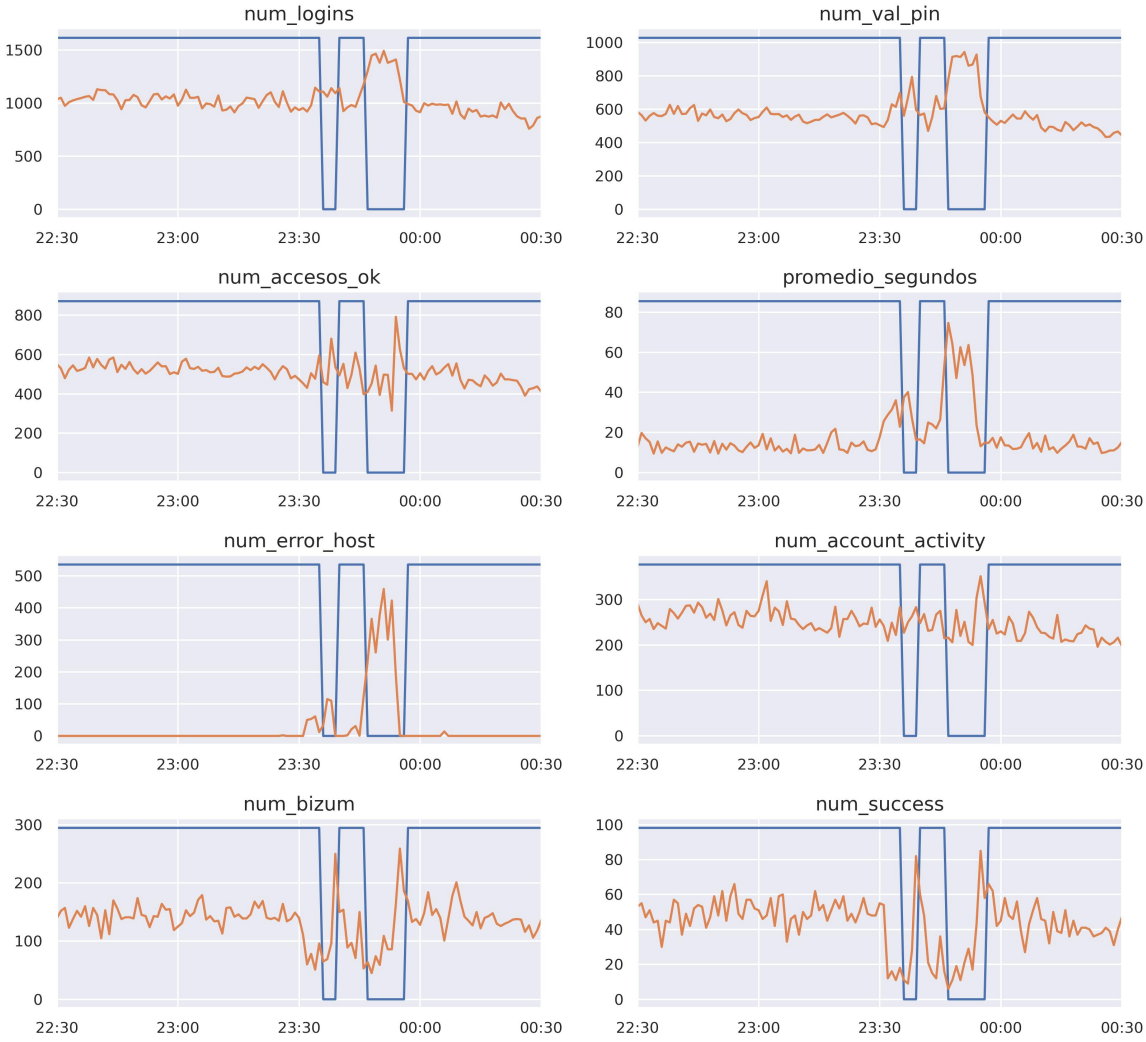


Figura 3.6: Representación de dos caídas consecutivas de la aplicación ocurridas entre las 22:30 del 27 de enero de 2024 y las 00:30 del día 28. En azul, la disponibilidad escalada de Banca Móvil, donde el 0 indica la caída. En naranja, los valores de las distintas variables que componen la serie temporal multivariante.

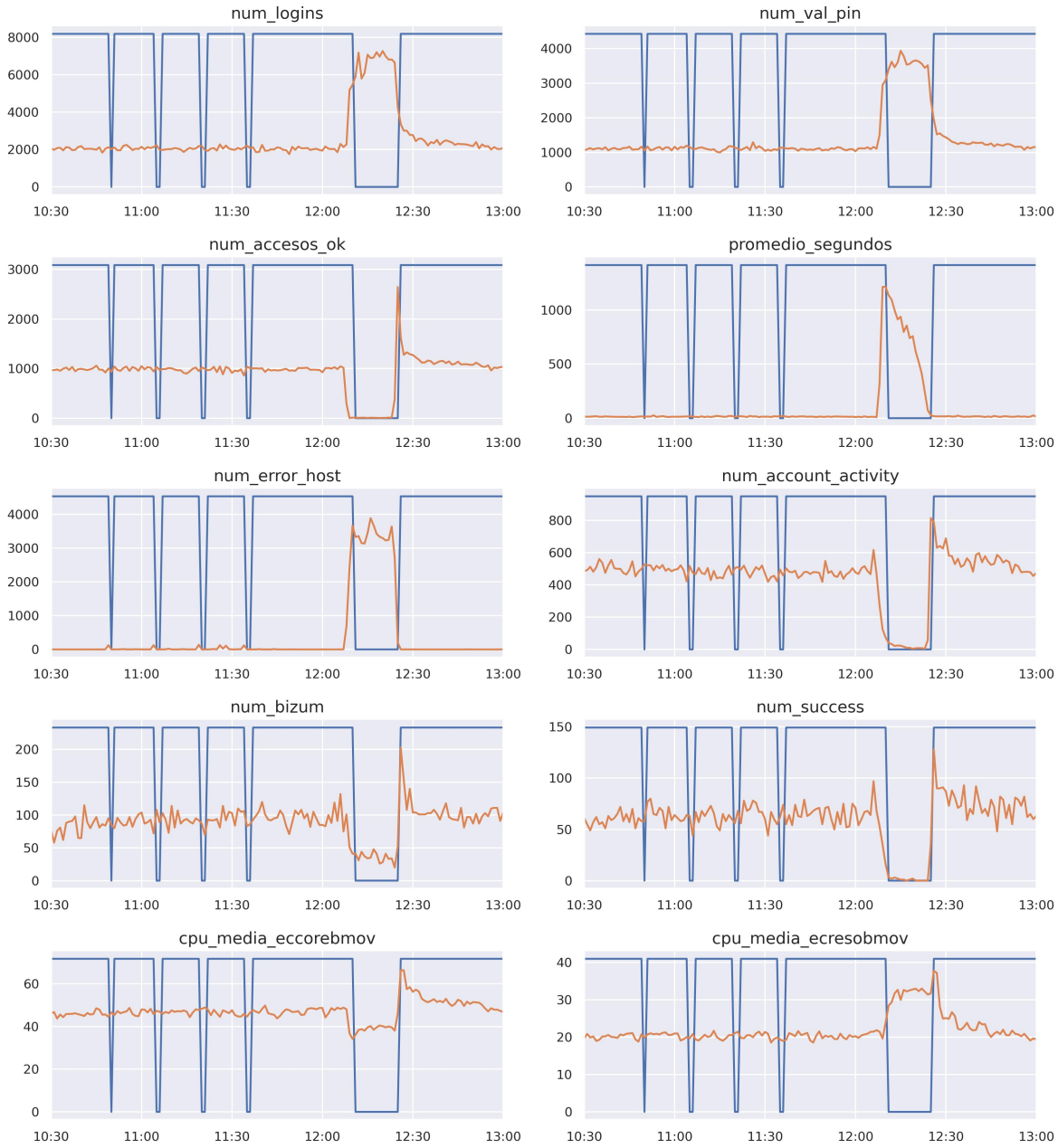


Figura 3.7: Representación de las caídas de Banca Móvil producidas el 21 de marzo de 2024 entre las 10:30 y las 13:00. En azul, la disponibilidad escalada de Banca Móvil, donde el 0 indica la caída. En naranja, los valores de las distintas variables que componen la serie temporal multivariante.

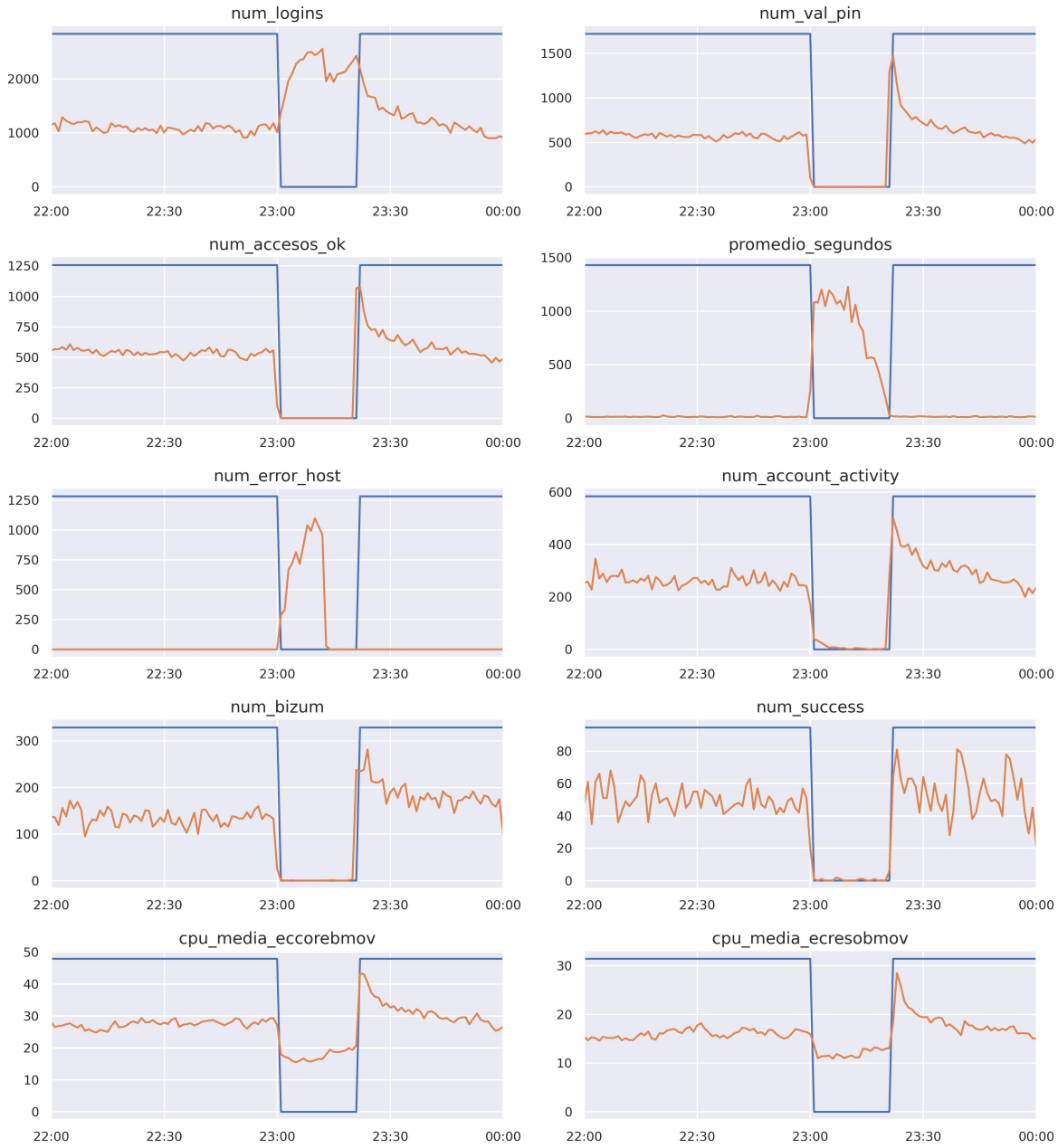


Figura 3.8: Representación de la caída de Banca Móvil ocurrida el 18 de mayo de 2024 entre las 22:00 y las 00:00. En azul, la disponibilidad escalada de Banca Móvil, donde el 0 indica la caída. En naranja, los valores de las distintas variables que componen la serie temporal multivariante.

3.3. Preprocesamiento

En este apartado se aborda el problema de las distintas escalas de las variables. Como ya se ha comentado en la Sección 2.5, el desempeño de los métodos *kNN*, *LSTM-AD* y *EncDecAD* puede verse afectado por estas diferencias en escala, mientras que *Isolation Forest* es indiferente a esta condición. Dos de las principales transformaciones utilizadas para trasladar los datos de la serie temporal multivariante a la misma escala son la *normalización Min-Max* (Belay *et al.*, 2023) y la *estandarización* (Braei y Wagner, 2020).

- **Normalización Min-Max**: consiste en reescalar los datos de cada una de las variables por separado para que todos los valores estén entre 0 y 1. Para una serie $\boldsymbol{X}_t$ con m variables, si $\text{mín}(X_{it})$ y $\text{máx}(X_{it})$ son los valores mínimo y máximo de la variable X_{it}, respectivamente, la transformación se define como:

$$X'_{it} = \frac{X_{it} - \text{mín}(X_{it})}{\text{máx}(X_{it}) - \text{mín}(X_{it})} \in [0,1] \ , \quad i = 1, \ldots, m. \tag{3.1}$$

- **Estandarización**: el objetivo es centrar las variables de la serie para que cada una de ellas tenga media igual a 0 y desviación típica igual a 1. Si $\mu_{X_{it}}$ denota la media de la variable X_{it} y $\sigma_{X_{it}}$ hace referencia a su desviación típica, la transformación para cada variable será:

$$X'_{it} = \frac{X_{it} - \mu_{X_{it}}}{\sigma_{X_{it}}} \ , \quad i = 1, \ldots, m. \tag{3.2}$$

Es muy importante tener en cuenta que estas transformaciones no pueden aplicarse directamente a todo el conjunto de datos. En efecto, esto llevaría a un problema de *Data Leakage*, es decir, la filtración de información de los datos de test en los datos de entrenamiento, o lo que es lo mismo en el contexto de series temporales, la filtración de datos futuros en el pasado (Chaba, 2023). Por esta razón, tanto la normalización Min-Max como la estandarización deben realizarse primero en el conjunto de datos de entrenamiento y, posteriormente, utilizando los estadísticos obtenidos a partir de estos datos (mínimo y máximo o media y desviación típica de cada variable, según la transformación utilizada), se reescalan los valores de los datos de test. En principio, no es posible saber cuál de las transformaciones será más efectiva a la hora de aplicar los distintos modelos. Con todo, las redes neuronales suelen funcionar mejor cuando los datos están estandarizados (Shanker *et al.*, 1996), beneficiándose también el método *kNN* de la estandarización (Braei y Wagner, 2020). En este trabajo se ha optado por aplicar ambas transformaciones para comprobar con cuál de ellas se obtienen mejores resultados.

~

A lo largo de este capítulo se han descrito y analizado las distintas variables disponibles relacionadas con Banca Móvil. Una vez concluido que pueden resultar de utilidad para el propósito de anticipar las caídas de la aplicación con unos ciertos minutos de margen, se procede a aplicar los métodos de detección de anomalías en el Capítulo 4, teniendo en cuenta el preprocesado necesario para los datos explicado en la sección anterior.

Capítulo 4

Resultados de los modelos empleados

En este capítulo se aplican los distintos modelos de detección de anomalías presentados en la Sección 2.5, a saber, *Isolation Forest*, *kNN*, *LSTM-AD* y *EncDec-AD*, a los datos disponibles de Banca Móvil. En particular, con el fin de analizar y evaluar el desempeño de estos modelos para anticipar las caídas de la aplicación, se han llevado a cabo dos pruebas diferentes, una con los datos de la primera serie y otra con los de la segunda. Además, las pruebas se han realizado de la forma más homogénea posible en orden a poder establecer una comparación entre ellos.

4.1. Consideraciones generales

En cada una de las pruebas, el primer paso consiste en establecer los datos de entrenamiento y los datos de test, que serán los mismos para todos los modelos. Como *LSTM-AD* y *EncDec-AD* son métodos semisupervisados, se debe buscar un período de tiempo en el que no se hayan producido caídas o, al menos, caídas no muy significativas, para definir un conjunto de entrenamiento válido para los cuatro modelos. Además, el conjunto de test sí debe contener anomalías con el propósito de evaluar la detección y anticipación de las caídas. Siguiendo la idea propuesta en la Sección 3.1, se procedió a modificar adecuadamente el etiquetado de *Ground Truth* de la disponibilidad para considerar solo las caídas con dos o más minutos de duración, pues son las que realmente tienen interés.

Posteriormente, se realiza el preprocesamiento de los datos según lo explicado en la Sección 3.3. El método de *Isolation Forest* es el único que no se ve afectado por las diferencias en escala de las variables, por lo que los resultados serán los mismos con o sin preprocesamiento, optando entonces por no hacerlo. Para los demás métodos, se consideran tanto la normalización como la estandarización para comprobar cuál de ellas ofrece mejores resultados en cada caso. Para hacer uso de estas transformaciones se utilizan las funciones `MinMaxScaler` y `StandardScaler` del módulo `preprocessing` de la librería *Scikit-learn* (Pedregosa *et al.*, 2011) de *Python*, desarrollada para modelos y tareas de *machine learning*.

En todos los modelos considerados se utiliza la técnica de ventanas deslizantes para modelar la dependencia secuencial de la serie. Un tamaño de ventana grande puede provocar que ciertas anomalías no sean detectadas, mientras que una longitud pequeña puede no capturar dependencias a largo plazo (Choi *et al.*, 2021). La selección para cada modelo se hace de forma empírica, al probar distintas longitudes y observar los resultados obtenidos.

Para evaluar el desempeño, se utilizan las métricas presentadas en la Sección 2.4. Por una parte, se consideran el F_1-*Score* clásico (función `f1_score` del módulo `metrics` de *Scikit-learn*) y sus versiones equivalentes cuando se desplaza el etiquetado de las predicciones hasta 1, 2 y 3 *lags*, para ver si esto mejora o no el valor de esta métrica. Se calcula también el *PATE* empleando la implementación del

repositorio oficial[1] de su artículo de referencia. Además, como el principal interés es anticipar las caídas, se mide la proporción de caídas correctamente anticipadas de entre las caídas verdaderas (similar al *Recall*) y la proporción de caídas verdaderas de entre todas las detectadas por el modelo (similar al *Precision*), métricas que permiten calcular una especie de F_1-*Score* de anticipación.

A la hora de seleccionar un umbral a partir del cual establecer las anomalías, es importante tener en cuenta que con un valor alto se reducirá el porcentaje de falsas alarmas, pero también pueden pasarse por alto las verdaderas anomalías, mientras que con un umbral bajo se detectarán más anomalías al mismo tiempo que aumentarán las falsas alarmas (Choi *et al.*, 2021). Como las anomalías representan un número muy pequeño de las observaciones y esto puede llevar a la detección de muchos falsos positivos, se decide emplear un umbral del tipo *contamination* o cuantil para todos los modelos. Así, la proporción de instantes anómalos en los datos de test se toma como estimación del porcentaje esperado de anomalías que deberían detectar los modelos, y esta estimación se emplea entonces para determinar el umbral. En otros términos, las anomalías serán ese porcentaje de instantes en los datos de test con los *anomaly scores* más altos. Aunque para *LSTM-AD* y *EncDec-AD* se proponía seleccionar el umbral maximizando el F_β-*Score* en los llamados conjuntos de validación v_{N2} y v_A, estos pueden no crearse si no hay suficientes anomalías o el verdadero etiquetado no es muy fiable, utilizando otro tipo de técnicas para la selección (Malhotra *et al.*, 2016). Además, como se busca poder anticipar las caídas, no conviene sobreajustar las predicciones al verdadero etiquetado.

Para elegir los *hiperparámetros* de los distintos modelos, es decir, los parámetros que controlan el entrenamiento y la estructura de cada uno de ellos, se utiliza el enfoque de la búsqueda en rejilla o *grid search*. Este método consiste en proponer varias combinaciones de hiperparámetros y seleccionar aquella con la que se obtienen los mejores resultados respecto a una métrica concreta. En el presente contexto, se elige aquella combinación con un mayor valor del F_1-*Score* de anticipación definido previamente (abreviado F_1-$Score_{\text{ant}}$).

Teniendo en cuenta todas estas consideraciones generales, se muestran a continuación los resultados obtenidos para los distintos modelos en las dos pruebas realizadas. Con fines de reproducibilidad, se emplea la semilla con un valor de 42 cuando sea necesario.

4.2. Primera prueba

Se utilizan los datos de la primera serie, con los siguientes conjuntos de entrenamiento y test:

- **Datos de entrenamiento**: desde las 00:00 del 07/02/2024 hasta las 23:59 del 07/03/2024, con 43200 observaciones de 8 variables.

- **Datos de test**: desde las 00:00 del 08/03/2024 hasta las 23:59 del 31/03/2024, con 34560 observaciones de 8 variables.

El porcentaje de instantes anómalos en el conjunto de test es del 0.391 %. Además, son 18 las caídas registradas con una duración de al menos 2 minutos en este período. En cada modelo, se añade a cada observación del *dataframe* de la serie el *anomaly score* que le corresponde y la etiqueta obtenida a partir del umbral para poder mostrar representaciones gráficas de los resultados.

A continuación, se presentan las implementaciones y los resultados obtenidos para cada uno de los modelos en esta primera prueba.

[1] https://github.com/Raminghorbanii/PATE

Isolation Forest

Para aplicar este método se utiliza la función `IsolationForest`[2] del módulo `ensemble` de la librería *Scikit-learn*. Además del número de *iTrees* (`n_estimators`) y el tamaño n de las submuestras (`max_samples`), que son los parámetros propios del modelo, esta implementación también permite seleccionar el número máximo de variables a considerar en la construcción de cada árbol (`max_features`). Además, incluye una opción para fijar el valor de `contamination` (en este caso, la proporción 0.00391) y establecer el umbral de forma automática, así como una opción para fijar la semilla (`random_state`=42).

Como ya se ha comentado, no se aplica ningún preprocesado a los datos. Para probar distintas combinaciones de hiperparámetros se consideran `n_estimators` $= [100, 500]$ y `max_samples` $= [16384, 32768]$, teniendo en cuenta lo sugerido en la Sección 2.5.1 y el tamaño del conjunto de entrenamiento. Se mantiene `max_features` con la configuración por defecto, en la que se permite usar todas las variables disponibles para construir cada árbol de entrenamiento. Las longitudes de ventana consideradas serán `window_size` $= [4, 6]$ pues, basándose en pruebas previas, este método parece no funcionar bien con longitudes mayores. De los ocho modelos resultantes, el mejor resultado en base al F_1-*Score* de anticipación se tiene con `n_estimators` $= 500$, `max_samples` $= 32768$ y `window_size` $= 4$, aunque las diferencias no son demasiado sustanciales. Se presentan entonces las métricas obtenidas y un par de representaciones gráficas de los *anomaly scores* frente a la *Ground Truth*.

Métrica	**Valor**
F_1-*Score* clásico	0.608
F_1-*Score*, *lag*=1	0.621
F_1-*Score*, *lag*=2	0.582
F_1-*Score*, *lag*=3	0.516

Métrica	**Valor**
PATE	0.683
$Recall_{\text{ant}}$	0.722
$Precision_{\text{ant}}$	0.560
$\boldsymbol{F_1\text{-}Score_{\text{ant}}}$	**0.631**

Anticipación	**Caídas**
1 minuto	8
2 minutos	3
3 minutos	2
4 minutos	0

Tabla 4.1: Resultados obtenidos para el modelo seleccionado de *Isolation Forest*.

En la Tabla 4.1 se puede ver que el F_1-Score con *lag*=1 tiene un valor mayor que el F_1-Score clásico, mientras que para los *lags* 2 y 3 se empeora el resultado. En la métrica *PATE*, que evalúa todos los posibles escenarios de detección de las caídas, se obtiene un valor también mayor que el F_1-Score clásico, el cual evalúa la predicción instante a instante. En cuanto a las métricas relacionadas con la anticipación, además de los valores de $Recall_{\text{ant}}$, $Precision_{\text{ant}}$ y $F_1\text{-}Score_{\text{ant}}$ mostrados en la tabla, se calculan los minutos de antelación con los que se detecta cada una de las 18 caídas de al menos dos minutos de duración registradas en el período de los datos de test. Se obtienen así el número de caídas detectadas con hasta 1, 2, 3 y 4 minutos de antelación, mostradas en el panel derecho de la Tabla 4.1. Se deduce entonces que la mayoría de las caídas anticipadas lo fueron con 1 minuto de antelación, y no se anticiparon o no se detectaron 5 de las 18 caídas.

En las Figuras 4.1 y 4.2 pueden verse los *anomaly scores* obtenidos por el modelo frente a la *Ground Truth* para dos caídas en los datos de test: el 10 de marzo de 2024 a las 22:04, anticipada con 2 minutos de antelación, y el 21 de marzo de 2024 a las 12:11, anticipada con 3 minutos de antelación, respectivamente. En la implementación utilizada, los *anomaly scores* se escalan de forma automática para que el umbral (en las figuras, *threshold*) quede establecido en 0.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

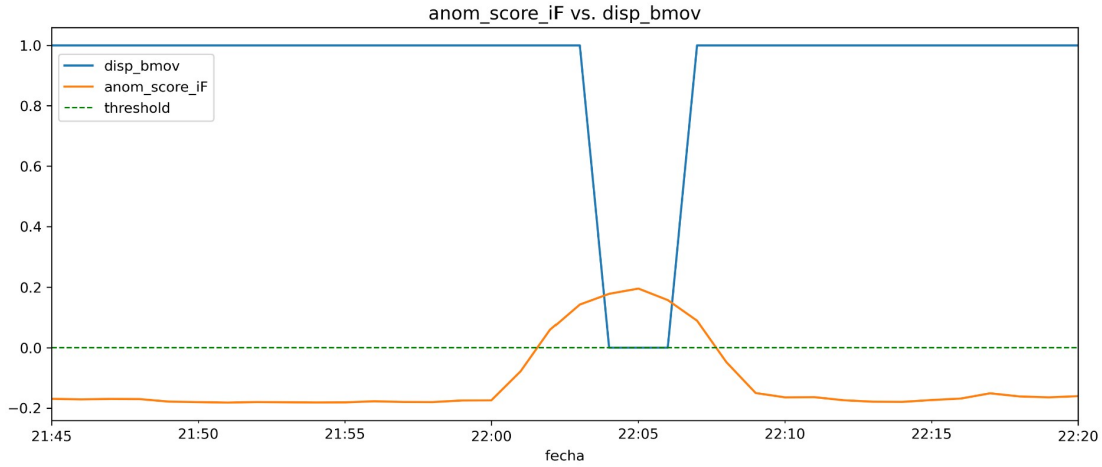


Figura 4.1: Representación de los *anomaly scores* obtenidos por *Isolation Forest* para la caída del 10 de marzo de 2024 a las 22:04. El umbral (*threshold*) determina las observaciones consideradas anómalas.

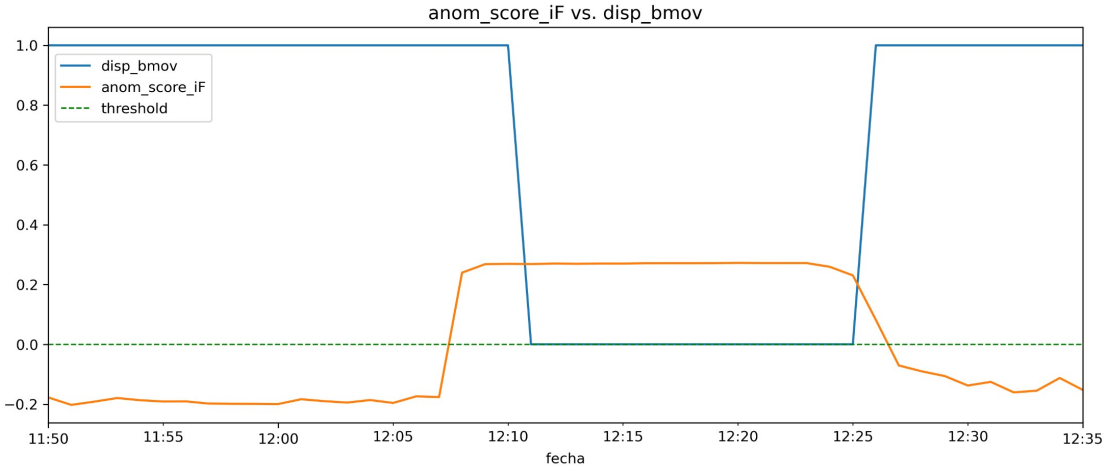


Figura 4.2: Representación de los *anomaly scores* obtenidos por *Isolation Forest* para la caída del 21 de marzo de 2024 a las 12:11. El umbral (*threshold*) determina las observaciones consideradas anómalas.

k Nearest Neighbors

Se utiliza la función `NearestNeighbors`[3] del módulo `neighbors` de *Scikit-learn*. Esta función permite seleccionar el número k de vecinos más cercanos (`n_neighbors`), el método utilizado para el cálculo de estos vecinos (`algorithm`) y la métrica empleada para el cálculo de las distancias (`metric`), entre otras opciones. Se mantiene la opción por defecto `algorithm` = *'auto'*, de forma que la función selecciona el mejor método de búsqueda de los vecinos (*KDTree*, *BallTree* o fuerza bruta) según los datos introducidos, y como métrica de distancia se emplea la distancia euclídea. Además, se establece

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html

la semilla igual a 42 con la función `seed` del paquete *random* de *Python*. En las pruebas realizadas, se considera `n_neighbors`= $[7, 10]$, `window_size` = $[3, 4]$ (al igual que *Isolation Forest*, parece no funcionar bien con valores mayores) y la normalización y la estandarización de los datos, pues este modelo sí es sensible a la escala de las variables. Se establece el umbral de forma que el 0.391 % de los *anomaly scores* más altos se etiqueten como anómalos. De nuevo, tampoco se aprecian diferencias sustanciales en los ocho modelos considerados, obteniéndose el mejor resultado de F_1-$Score_{\text{ant}}$ con `n_neighbors`= 7, `window_size` = 3 y la estandarización de los datos. Esto último concuerda con lo expuesto en la Sección 3.3.

Métrica	Valor
F_1-*Score* clásico	0.659
F_1-*Score*, *lag*=1	0.658
F_1-*Score*, *lag*=2	0.615
F_1-*Score*, *lag*=3	0.556

Métrica	Valor
PATE	0.726
$Recall_{\text{ant}}$	0.611
$Precision_{\text{ant}}$	0.632
$\boldsymbol{F_1\text{-}Score_{\text{ant}}}$	**0.621**

Anticipación	Caídas
1 min	6
2 min	3
3 min	2
4 min	0

Tabla 4.2: Resultados obtenidos para el modelo seleccionado de *k Nearest Neighbors*.

En la Tabla 4.2 se observan las métricas resultantes para el modelo seleccionado. En este caso, no se observa mejora en el F_1-*Score* clásico al considerar los *lags*, y este método tiene mayor F_1-*Score* clásico y mayor *PATE* que el modelo considerado de *Isolation Forest*. Sin embargo, en comparación tiene menor valor de $Recall_{\text{ant}}$ y de F_1-$Score_{\text{ant}}$, aunque mayor $Precision_{\text{ant}}$. Además, anticipa dos caídas menos que *Isolation Forest*.

En las Figuras 4.3 y 4.4 pueden verse los *anomaly scores* para las mismas caídas del 10 y el 21 de marzo consideradas para *Isolation Forest*, anticipándolas también con 2 y 3 minutos de antelación, respectivamente.

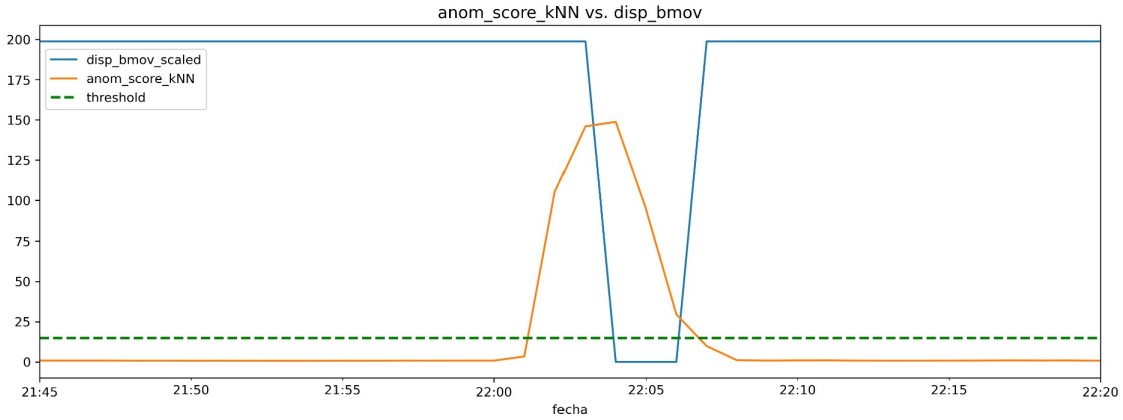


Figura 4.3: Representación de los *anomaly scores* obtenidos por *k Nearest Neighbors* para la caída del 10 de marzo de 2024 a las 22:04. El umbral (*threshold*) determina las observaciones anómalas.

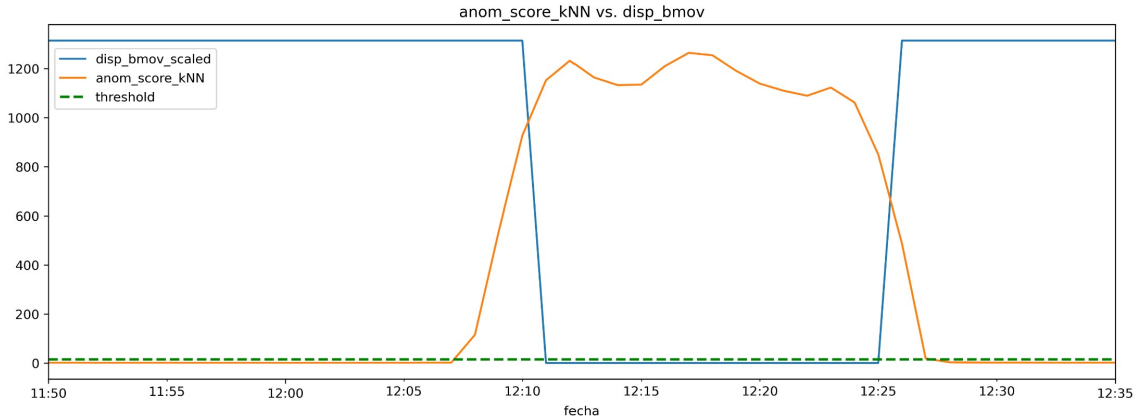


Figura 4.4: Representación de los *anomaly scores* obtenidos por *k Nearest Neighbors* para la caída del 21 de marzo de 2024 a las 12:11. El umbral determina las observaciones consideradas anómalas.

LSTM-AD

Para implementar este método, se utiliza principalmente el código para el modelo *LSTM-AD* del repositorio oficial[4] de la librería *TimeEval* (Wenig *et al.*, 2022) de *Python*. En este caso se tienen muchos más hiperparámetros que para los métodos anteriores y el entrenamiento también es más lento, por lo que es más tedioso probar diferentes combinaciones.

Las pruebas se realizan con `window_size` = [15, 30] (longitud de la ventana), `num_layers` = [2] (número de capas ocultas de la red), `hidden_size` = [128, 256] (neuronas en cada capa oculta), `dropout_rate` = [0.15] (para evitar sobreajuste), `learning_rate` = [0.001] (tasa de aprendizaje), `batch_size` = [128] (tamaño de los lotes), `epochs` = 30 (número de *epochs*), `patience` = [5] (para el método de *early stopping*), `activation` =‘tanh’ (función de activación, que es la tangente hiperbólica), `loss` =‘mse’ (función de pérdida, que es el error cuadrático medio) y la estandarización y normalización de los datos. Como ya se había comentado, no se crean los conjuntos v_{N2} y v_A, obteniéndose v_{N1} al seleccionar el 20 % de los datos de entrenamiento para la validación. En la implementación se emplea además el optimizador *Adam*, uno de los más utilizados para el entrenamiento de redes neuronales (Kingma y Ba, 2015). El mejor modelo en base al F_1-*Score* de anticipación se obtiene con `window_size` = 30, `hidden_size` = 256 y la estandarización de los datos (coincidiendo de nuevo esto último con lo expuesto en la Sección 3.3).

Métrica	Valor
F_1-*Score* clásico	0.568
F_1-*Score*, *lag*=1	0.669
F_1-*Score*, *lag*=2	0.667
F_1-*Score*, *lag*=3	0.576

Métrica	Valor
PATE	0.620
$Recall_{\text{ant}}$	0.778
$Precision_{\text{ant}}$	0.654
$\boldsymbol{F_1\text{-}Score_{\text{ant}}}$	**0.711**

Anticipación	Caídas
1 min	9
2 min	4
3 min	1
4 min	0

Tabla 4.3: Resultados obtenidos para el modelo seleccionado de *LSTM-AD*.

[4] https://github.com/TimeEval/TimeEval

En la Tabla 4.3 se aprecia que el F_1-*Score* clásico sí mejora para 1, 2 y 3 *lags*, sugiriendo que además de anticipar la caída también deja de detectarla antes. Se obtienen mejores valores para las métricas de anticipación que para los modelos anteriores, llegando el $Recall_{\text{ant}}$ hasta el 0.778, anticipando así un mayor número de caídas, aunque muchas de ellas con un solo minuto de antelación.

En las Figuras 4.5 y 4.6 se pueden observar los *anomaly scores* para las mismas caídas que en los modelos anteriores, detectadas también con 2 y 3 minutos de antelación, respectivamente. Se observa cómo efectivamente las caídas dejan de detectarse en cuanto terminan o un poco antes. En particular, en la Figura 4.6 los *anomaly scores* alcanzan valores muy elevados, haciendo que se aprecie peor el umbral visualmente.

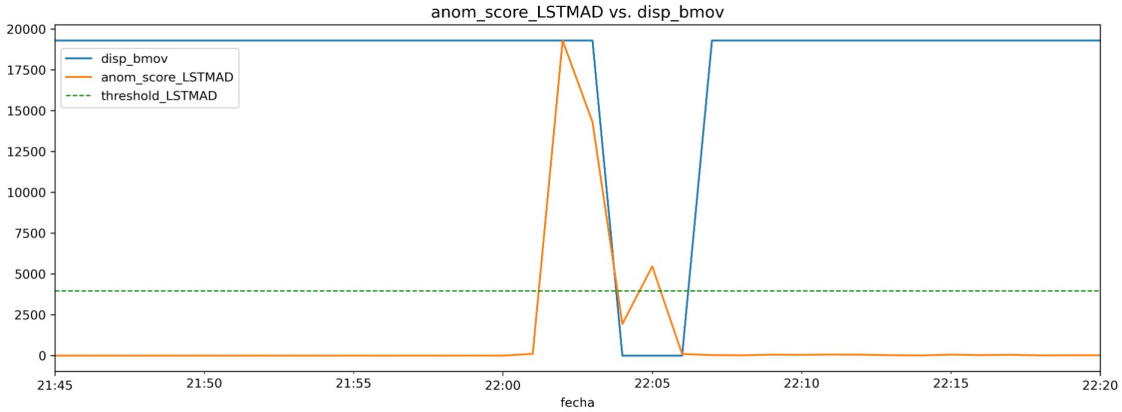


Figura 4.5: Representación de los *anomaly scores* obtenidos por *LSTM-AD* para la caída del 10 de marzo de 2024 a las 22:04. El umbral (*threshold*) determina las observaciones anómalas.

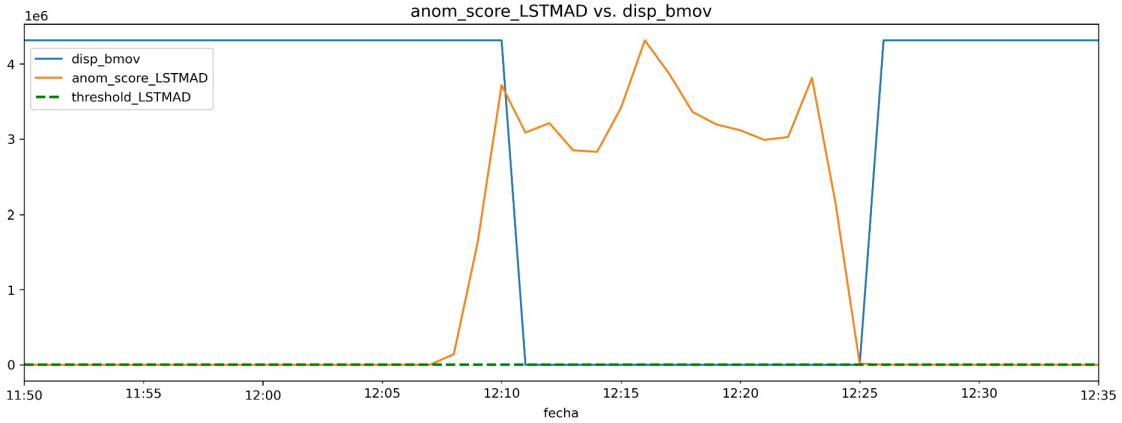


Figura 4.6: Representación de los *anomaly scores* obtenidos por *LSTM-AD* para la caída del 21 de marzo de 2024 a las 12:11. El umbral determina las observaciones consideradas anómalas.

EncDec-AD

La implementación de este método se hace siguiendo el código que se puede encontrar en el repositorio Giammarino (2022). Al igual que para el modelo *LSTM-AD*, se tienen varios hiperparámetros

y el entrenamiento es incluso un poco más lento. Se ha probado con `window_size` = [15, 30] (longitud de la ventana), `c` = [64, 128] (neuronas en la oculta de la red *LSTM*), `dropout_rate` = [0.15] (para evitar sobreajuste), `learning_rate` = [0.001] (tasa de aprendizaje), `batch_size` = [128] (tamaño de los lotes), `epochs` = 30 (número de *epochs*), `patience` = [3] (para el método de *early stopping*) y la estandarización y la normalización de los datos. De nuevo, se eliminan los conjuntos v_{N2} y v_A y se seleccionan el 30 % de los datos de entrenamiento para la validación. Se emplea también el optimizador *Adam* para el entrenamiento de la red (Kingma y Ba, 2015).

El mejor modelo se obtiene con `window_size` = 15, `c` = 64 y la estandarización de los datos. En la Tabla 4.4 se proporcionan los resultados. El F_1-*Score* clásico mejora para 1 y 2 *lags*, pero no para 3. La métrica de $Recall_{\text{ant}}$ es la mejor de todos los modelos, siendo este, por tanto, el método que más caídas anticipa. De nuevo, la mayoría se predicen con 1 minuto de antelación, y parece tener un poco más de capacidad de anticipación que el *LSTM-AD* al detectar dos caídas tres minutos antes, cuando *LSTM-AD* solo detectaba una. Sin embargo, se tiene un resultado de $Precision_{\text{ant}}$ similar al de *Isolation Forest*.

Métrica	**Valor**
F_1-*Score* clásico	0.553
F_1-*Score*, *lag*=1	0.656
F_1-*Score*, *lag*=2	0.638
F_1-*Score*, *lag*=3	0.531

Métrica	**Valor**
PATE	0.620
$Recall_{\text{ant}}$	0.889
$Precision_{\text{ant}}$	0.562
$\boldsymbol{F_1\text{-}Score_{\text{ant}}}$	**0.689**

Anticipación	**Caídas**
1 min	10
2 min	4
3 min	2
4 min	0

Tabla 4.4: Resultados obtenidos para el modelo seleccionado de *EncDec-AD*.

En las Figuras 4.7 y 4.8 se observan los *anomaly scores* obtenidos para las mismas caídas consideradas en los demás modelos, detectadas con 2 y 3 minutos de antelación, respectivamente. Al igual que en el *LSTM-AD*, las caídas dejan de detectarse justo al terminar o antes. De nuevo, los valores elevados de los *anomaly scores* en la Figura 4.8 impiden apreciar bien el umbral.

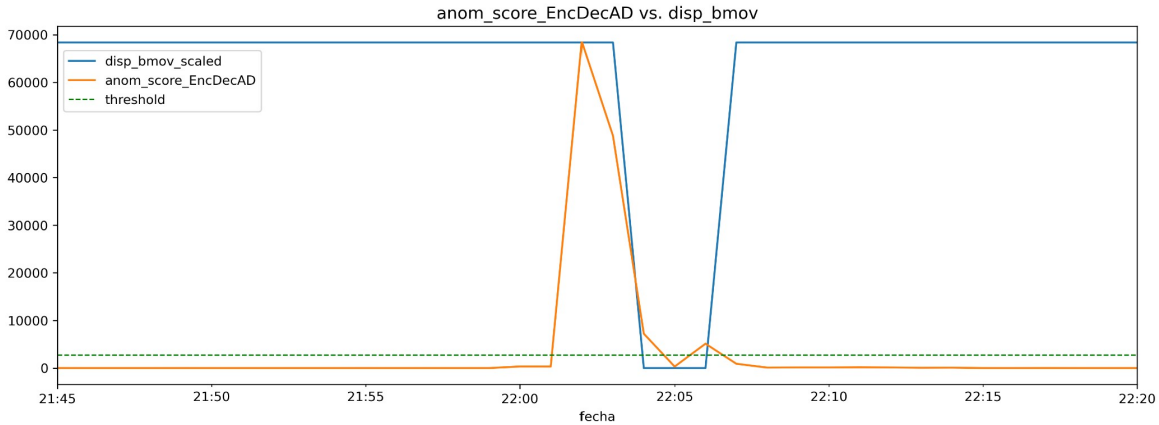


Figura 4.7: Representación de los *anomaly scores* obtenidos por *EncDec-AD* para la caída del 10 de marzo de 2024 a las 22:04. El umbral (*threshold*) determina las observaciones anómalas.

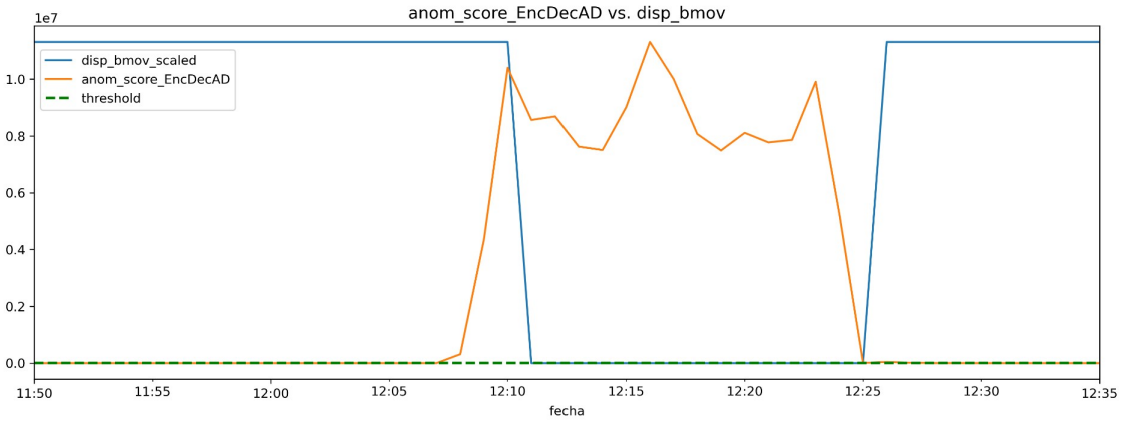


Figura 4.8: Representación de los *anomaly scores* obtenidos por *EncDec-AD* para la caída del 21 de marzo de 2024 a las 12:11. El umbral determina las observaciones consideradas anómalas.

Por último, en la Tabla 4.5 se muestran de forma conjunta las principales medidas de anticipación obtenidas para los 4 modelos. Se resaltan en negrita los mejores valores de cada métrica.

Modelo	$\boldsymbol{Recall_{ant}}$	$\boldsymbol{Precision_{ant}}$	$\boldsymbol{F_1\text{-}Score_{ant}}$
iForest	0.722	0.560	0.631
kNN	0.611	0.632	0.621
LSTM-AD	0.778	**0.654**	**0.711**
EncDec-AD	**0.889**	0.562	0.689

Tabla 4.5: Valores de las métricas de anticipación para los 4 modelos estudiados.

4.3. Segunda prueba

Se utilizan ahora los datos de la segunda serie, donde se incluyen las dos variables relacionadas con los servidores. Los conjuntos de entrenamiento y test serán los siguientes:

- **Datos de entrenamiento**: desde las 00:00 del 17/04/2024 hasta las 23:59 del 17/05/2024, con 44640 observaciones de 10 variables.
- **Datos de test**: desde las 00:00 del 18/05/2024 hasta las 23:59 del 10/06/2024, con 34560 observaciones de 10 variables.

El porcentaje de instantes anómalos en el conjunto de test es del 0.718 %. Además, son 24 las caídas registradas con una duración de al menos 2 minutos en este período. Para cada modelo se emplean los mismos hiperparámetros con los que se obtuvieron los mejores resultados en la prueba anterior. El análisis pone ahora el foco simplemente en el efecto que tiene en los modelos el hecho de incluir las dos variables de los servidores. En la Tabla 4.6 se muestran los valores obtenidos en este caso para las métricas adaptadas de anticipación.

Modelo	$\boldsymbol{Recall_{ant}}$	$\boldsymbol{Precision_{ant}}$	$\boldsymbol{F_1\text{-}Score_{ant}}$
iForest	0.667	0.325	0.437
kNN	0.625	0.429	0.509
LSTM-AD	0.792	**0.647**	**0.712**
EncDec-AD	**0.833**	0.513	0.625

Tabla 4.6: Valores de las métricas de anticipación en la segunda prueba para los 4 modelos estudiados.

En la Figura 3.2 se apreciaba el repunte del porcentaje de *CPU* para los servidores en la madrugada, debido al reinicio automático de los mismos. Parece que esto es lo que puede estar afectando a la precisión de *Isolation Forest* y *k Nearest Neighbors*, pues es más baja en comparación con lo obtenido para *LSTM-AD* y *EncDec-AD* y con los resultados de la primera prueba. En la Figura 4.9 se muestran los *anomaly scores* de *Isolation Forest* entre el 7 de junio a las 00:00 y el 9 de junio a las 04:00. En efecto, se observa cómo los *anomaly scores* correspondientes a las horas en las que se produce el reinicio de los servidores (sobre las 5:00 o 5:30) sobrepasan el umbral, con puntuaciones similares a las que se obtienen para los instantes de las caídas verdaderas que se producen en la medianoche del día 9.

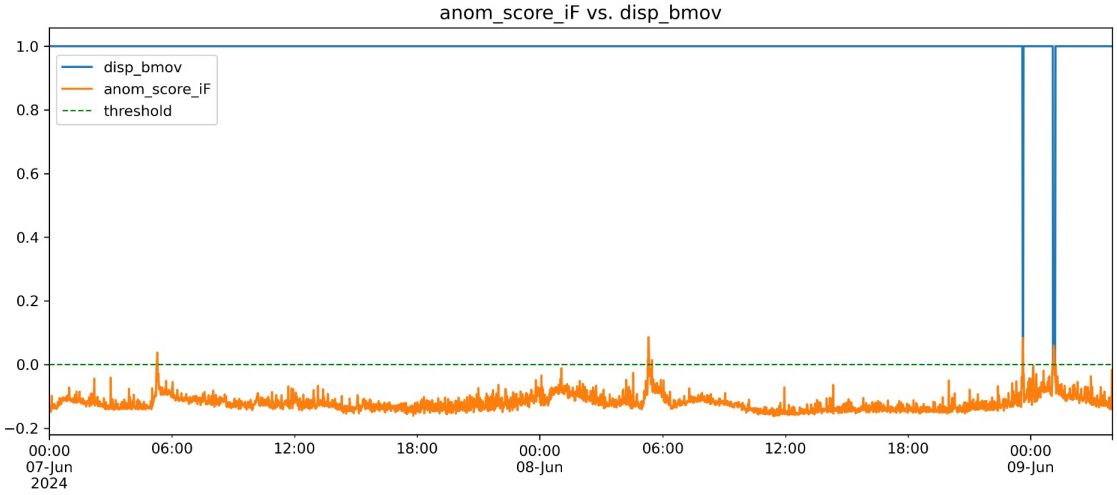


Figura 4.9: Representación de los *anomaly scores* obtenidos por *Isolation Forest* desde las 00:00 del 7 de junio a las 04:00 del 9 de junio. El umbral determina las observaciones consideradas anómalas.

En vista de los resultados obtenidos para ambas pruebas, se puede decir que los modelos *LSTM-AD* y *EncDec-AD* tienen un mejor desempeño en la tarea de detectar y anticipar las caídas de Banca Móvil. Como *Isolation Forest* y *kNN* no son modelos propios de las series temporales, tienen más dificultad para tratar la dependencia secuencial y aprender patrones cíclicos, como en el caso que se muestra en la segunda prueba con los servidores. La elección entre *LSTM-AD* y *EncDec-AD* depende principalmente de si se prefiere tener una mayor precisión o una mayor cantidad de caídas anticipadas.

Capítulo 5

Conclusiones

Del estudio realizado se derivan algunas conclusiones que se exponen a continuación.

Por una parte, se puede afirmar que los modelos empleados son válidos para la detección de anomalías en series temporales multivariantes. Más aún, son útiles para la tarea de anticipar eventos anómalos, como son las caídas de Banca Móvil. Sin embargo, con las series de datos utilizadas, el tiempo de anticipación para las caídas es de 3 minutos en el mejor de los casos. Tras las dificultades afrontadas para encontrar variables en las que se apreciasen las caídas con cierta antelación, se concluye que estos métodos serían mucho más efectivos si estuviesen disponibles otro tipo de datos. Por ejemplo, se podrían considerar métricas más intrínsecamente relacionadas con la infraestructura y los sistemas que dan soporte a la aplicación, como la telemetría de los servidores (temperatura, vibración y otras medidas), datos que en ABANCA, en el momento de escribir esta memoria, no se encuentran monitorizados.

Los modelos de *deep learning* semisupervisados como *LSTM-AD* y *EncDec-AD* presentan un mejor desempeño a la hora de predecir o reconstruir los datos cuanto mayor sea el número de observaciones de los datos de entrenamiento. Por lo tanto, también sería útil disponer de un período más amplio de datos históricos con comportamiento normal y sin caídas para entrenar estos modelos.

Por otra parte, se consiguen anticipar con más o menos minutos de antelación un número considerable de las caídas, en especial al utilizar el modelo *EncDec-AD*, con un valor de más del 0.8 para el *Recall* de anticipación en las pruebas realizadas. Con todo, en general también se aprecian bastantes falsas alarmas. El modelo más preciso es *LSTM-AD*, rondando un valor de 0.65 en ambas pruebas para la proporción de caídas verdaderas sobre el total de caídas detectadas. En este trabajo se ha optado por emplear una selección del umbral del tipo *contamination* o cuantil en base al número esperado de instantes anómalos en los datos de test, pero se podrían probar otro tipo de técnicas para ver si se consigue reducir el número de falsos positivos y mejorar los resultados. Además, aunque las verdaderas anomalías suelen tener *anomaly scores* altos, en muchos contextos estas puntuaciones pueden no ser determinantes por sí mismas, generando estas falsas alarmas, y sería el analista el encargado de distinguir entre ruido y anomalía basándose en su conocimiento del problema (Laptev *et al.*, 2015).

En ABANCA se dispone de varios modelos de detección de anomalías en series univariantes basados precisamente en redes neuronales *LSTM*, con una metodología similar al modelo *LSTM-AD* considerado en este trabajo. El objetivo es generar alertas del mal funcionamiento de un sistema, de forma que las personas a cargo puedan revisar más detalladamente si realmente se está produciendo un problema. Por esta razón, en el caso particular de las caídas de Banca Móvil, el mayor interés está en anticipar las caídas, es decir, en la generación de alertas. Así, se concluye que sería preferible utilizar el *EncDec-AD* por tener una mayor capacidad de anticipación, pues el beneficio es mayor que el coste de las posibles falsas alarmas.

5.1. Comentarios finales

Dentro del ámbito del *deep learning* se están desarrollando modelos con arquitecturas cada vez más complejas e innovadoras para tratar de abordar los dos problemas más comunes en la predicción de anomalías en series temporales multivariantes, que son las falsas alarmas y los eventos anómalos no detectados (Choi *et al.*, 2021). En particular, se está apostando por los modelos *sequence-to-sequence* basados en reconstrucción (categoría a la que pertenece *EncDec-AD*), introduciendo arquitecturas propias de otras áreas, como la arquitectura *Transformer* (Vaswani *et al.*, 2017). Esta es un tipo de red neuronal muy popular hoy en día gracias al ámbito del Procesamiento del Lenguaje Natural. Por ejemplo, el modelo *TranAD* (Tuli *et al.*, 2022) utiliza este tipo de redes para la detección de anomalías en series temporales multivariantes. Con todo, estos nuevos modelos que van surgiendo todavía no consiguen superar de forma clara a los modelos ya existentes, evidenciando las dificultades intrínsecas al tratar el problema de predicción de anomalías en series temporales multivariantes.

Apéndice A

Conceptos complementarios

En este apéndice se presentan algunas definiciones adicionales, ordenadas alfabéticamente, que pueden resultar de utilidad para una mayor comprensión de los contenidos teóricos de este trabajo. Se utilizan como referencia los apuntes de Fernández-Casal *et al.* (2021), Pateiro y Sánchez (2022) y García y Vélez (2012).

A.1. Análisis de componentes principales

El *análisis de componentes principales* es una técnica empleada para reducir la dimensión de los datos, pasando de una gran cantidad de variables relacionadas entre sí a unas pocas componentes independientes. La idea es encontrar unas pocas combinaciones lineales de estas variables que representen lo mejor posible la variabilidad de los datos. Estas combinaciones lineales reciben el nombre de *componentes principales* y permiten mantener la información de la correlación entre las variables.

A.2. Árboles de decisión

Los *árboles de decisión* constituyen una de las técnicas básicas del aprendizaje automático debido a su facilidad de interpretación a la hora de realizar predicciones tanto en problemas de clasificación como de regresión. La idea del método consiste en ir segmentando todos los posibles valores de las variables a partir de un *nodo* inicial que representa a toda la muestra de entrenamiento. De este nodo inicial salen dos *ramas* que dividen los datos en dos subconjuntos (atendiendo a una cierta condición), representado cada uno de ellos por un nuevo nodo. Este proceso se repite de forma recursiva hasta llegar a los nodos terminales, también llamados *hojas* del árbol, que son los que se utilizan para realizar las predicciones.

A.3. Distancia de Mahalanobis

La *distancia de Mahalanobis* es una medida de distancia que permite determinar la similitud entre dos variables aleatorias m-dimensionales. Si $\boldsymbol{x}$ e $\boldsymbol{y}$ son dos vectores aleatorios con la misma distribución de probabilidad y matriz de covarianzas $\boldsymbol{\Sigma}$, la distancia de Mahalanobis se define como:

$$d_M(\boldsymbol{x}, \boldsymbol{y}) \;=\; \sqrt{(\boldsymbol{x}-\boldsymbol{y})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{y})}.$$

En un contexto multivariante, esta medida es adecuada debido a que es invariante ante cambios de escala y tiene en cuenta las correlaciones entre las distintas variables.

A.4. Distancia euclídea

La *distancia euclídea* entre dos puntos $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ se define como:

$$d_2(\boldsymbol{x}, \boldsymbol{y}) \;=\; \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}.$$

Esta medida tiene algunas desventajas en el contexto multivariante, pues no es invariante ante cambios de escala y asume que las variables no están correlacionadas.

A.5. Distancia Manhattan

La *distancia Manhattan* entre dos puntos $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ se calcula como:

$$d_1(\boldsymbol{x}, \boldsymbol{y}) \;=\; \sum_{i=1}^{m}|x_i - y_i|.$$

Por definición, esta distancia siempre es mayor que la distancia euclídea.

A.6. Distribución normal multivariante

Se dice que una variable aleatoria $\boldsymbol{X}$ m-dimensional sigue una *distribución normal multivariante* con vector de medias $\boldsymbol{\mu}$ y matriz de covarianzas $\boldsymbol{\Sigma}$ si su función de densidad viene dada por:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) \;=\; \frac{1}{(\sqrt{2\pi})^m|\boldsymbol{\Sigma}|^{1/2}} \cdot exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\,\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \quad \boldsymbol{x} \in \mathbb{R}^m.$$

Se denotará como $\boldsymbol{X} \sim \mathcal{N}_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

A.7. Error absoluto medio

El *error absoluto medio* o *MAE* es una medida utilizada para evaluar la calidad de las predicciones de un modelo de regresión. Se usa de manera frecuente en los modelos relacionados con series de tiempo. Si n denota el tamaño de la muestra y m la dimensión de los datos, $\hat{\boldsymbol{y}} \in \mathbb{R}^m$ representa las predicciones obtenidas por el modelo e $\boldsymbol{y} \in \mathbb{R}^m$ representa los valores reales, el *MAE* se define como:

$$MAE \;=\; \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|.$$

Cuanto menor sea el valor del *MAE*, mejor será el rendimiento del modelo considerado.

A.8. Error cuadrático medio

El *error cuadrático medio* o *MSE* es una medida empleada para evaluar la calidad de las predicciones realizadas por un modelo de regresión. Si n denota el tamaño de la muestra y m la dimensión de los datos, $\hat{\boldsymbol{y}} \in \mathbb{R}^m$ representa las predicciones obtenidas por el modelo e $\boldsymbol{y} \in \mathbb{R}^m$ representa los valores reales, el *MSE* se define como:

$$MSE \;=\; \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|^2.$$

Cuanto menor sea el valor del *MSE*, mejor desempeño tendrá el modelo considerado.

A.9. Estimación por máxima verosimilitud

La *estimación por máxima verosimilitud* es un método que consiste en seleccionar como estimaciones de los parámetros de interés aquellos valores que tienen una mayor probabilidad de ocurrir según los datos observados. Sea $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ una muestra aleatoria simple con función de densidad conjunta $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\boldsymbol{\theta}) = f(\boldsymbol{x}_1|\boldsymbol{\theta}) \ldots f(\boldsymbol{x}_n|\boldsymbol{\theta})$ que depende de un vector de parámetros $\boldsymbol{\theta}$. El método está basado en la *función de verosimilitud*:

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{i=1}^{n} f(\boldsymbol{x}_i|\theta),$$

donde $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ representa el vector de parámetros a estimar y $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ son los datos observados. En la práctica, se suele utilizar el logaritmo de la función de verosimilitud:

$$l(\boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{i=1}^{n} \log(f(\boldsymbol{x}_i|\boldsymbol{\theta})).$$

De esta forma, la estimación del verdadero vector $\boldsymbol{\theta}_0$ será el $\boldsymbol{\theta}$ que maximice el logaritmo de la función de verosimilitud, es decir:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} l(\boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n).$$

Se puede comprobar que los estimadores de máxima verosimilitud para el vector de medias y la matriz de covarianzas de una distribución normal multivariante son el vector de medias muestral y la matriz de covarianzas muestrales, respectivamente.

A.10. Maldición de la dimensionalidad

La *maldición de la dimensionalidad* hace referencia a los problemas que surgen en numerosos modelos de *machine learning* al aumentar el número de dimensiones de los datos de entrenamiento. Afecta especialmente a los modelos basados en distancias, pues los datos se encuentran muy dispersos en el espacio de alta dimensión y esto agrava considerablemente el desempeño de los modelos, aumentando además el tiempo de computación.

Bibliografía

[1] Assendorp, J.P. (2017). *Deep learning for anomaly detection in multivariate time series data.* Master Thesis, Hamburg University of Applied Sciences.

[2] Audibert, J., Michiardi, P., Guyard, F., Marti, S., Zuluaga, M.A. (2022). Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, 132, 108945. https://doi.org/10.1016/j.patcog.2022.108945

[3] Belay, M.A., Blakseth, S.S., Rasheed, A., Rossi, P.S. (2023). Unsupervised Anomaly Detection for IoT-Based Multivariate Time Series: Existing Solutions, Performance Analysis and Future Directions. *Sensors*, 23, 2844. https://doi.org/10.3390/s23052844

[4] Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A. (2020). A review on outlier/anomaly detection in time series data. *CoRR*, abs/2002.04236. https://doi.org/10.48550/arXiv.2002.04236

[5] Braei M., Wagner, S. (2020). Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *arXiv preprint arXiv:2004.00433*. https://doi.org/10.48550/arXiv.2004.00433

[6] Breunig, M. M., Kriegel, H.P., Ng, R.T., Sander, J. (2000). LOF: identifying density-based local outliers. En *Proceedings of the International Conference on Management of Data (SIGMOD)*, 93-104. https://doi.org/10.1145/335191.335388

[7] Chaba, V. (8 de mayo de 2023). *Data Leakage in Machine Learning.* Medium. https://medium.com/@chabavictor7/data-leakage-in-machine-learning-d2ae0b3cd6ca

[8] Chamberlin, D.D., Boyce, R.F. (1974). SEQUEL: A structured English query language. En *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, 249-264, ACM. https://doi.org/10.1145/800296.8115

[9] Chandola, V. (2009). *Anomaly Detection for Symbolic Sequences and Time Series Data.* PhD Thesis, University of Minnesota.

[10] Chauhan, S., Vig, L. (2015). Anomaly detection in ECG time signals via deep long short-term memory networks. En *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 1-7. https://doi.org/10.1109/DSAA.2015.7344872

[11] Chen, H., Ma, H., Chu, X., Xue, D. (2020). Anomaly detection and critical attributes identification for products with multiple operating conditions based on isolation forest. *Advanced Engineering Informatics*, 46, 101139. https://doi.org/10.1016/j.aei.2020.101139

[12] Cho, K., Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. http://dx.doi.org/10.3115/v1/D14-1179

[13] Choi, K., Yi, J., Park, C., Yoon, S. (2021). Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines. *IEEE Access*, 9, 120043-120065. https://doi.org/10.1109/ACCESS.2021.3107975

[14] Fernández-Casal, R., Costa, J., Oviedo, M. (2021). *Aprendizaje Estadístico*. Apuntes de la materia, Universidade da Coruña. https://rubenfcasal.github.io/aprendizaje_estadistico/

[15] García, A., Vélez, R. (2012). *Principios de Inferencia Estadística*. Universidad Nacional de Educación a Distancia.

[16] Ghorbani, R., Reinders, M., Tax, D. (2024). PATE: Proximity-Aware Time series anomaly Evaluation. Accepted by *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2024)*. https://doi.org/10.48550/arXiv.2405.12096

[17] Giammarino, F. (2022). *TensorFlow implementation of EncDec-AD model for multivariate time series anomaly detection with LSTM-based encoder-decoder*. GitHub. https://github.com/flaviagiammarino/encdec-ad-tensorflow

[18] González, G.G. (2020). *Detección de anomalıas en series multivariable con modelos generativos*. Doctoral dissertation, Universidad de la República Montevideo.

[19] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.

[20] Graves, A. (2013). Generating Sequences with Recurrent Neural Networks. https://doi.org/10.48550/arXiv.1308.0850

[21] Hristov, H. (29 de marzo de 2023). *Introduction to K-D Trees*. Baeldung. https://www.baeldung.com/cs/k-d-trees

[22] Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

[23] James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J. (2023). *An Introduction to Statistical Learning with Applications in Python*. Springer Texts in Statistics, Springer Cham.

[24] Kingma, D.P., Ba, J. (2015). Adam: A Method for Stochastic Optimization. En *International Conference on Learning Representations (ICLR). arXiv preprint arXiv:1412.6980*.

[25] Laptev, N., Amizadeh, S., Flint, I. (2015). Generic and Scalable Framework for Automated Time-series Anomaly Detection. En *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1939-1947. https://doi.org/10.1145/2783258.2788611

[26] Li, G., Jung, J.J. (2023). Deep learning for anomaly detection in multivariate time series: Approaches, applications and challenges. *Information Fusion*, 91, 93-102. https://doi.org/10.1016/j.inffus.2022.10.008

[27] Liu, F.T., Ting, K.M., Zhou, Z.-H. (2008). Isolation Forest. En *Proceedings of the International Conference on Data Mining (ICDM)*, 413-422. https://doi.org/10.1109/ICDM.2008.17

[28] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *CoRR*, abs/1607.00148. https://doi.org/10.48550/arXiv.1607.00148

[29] Malhotra, P., Vig, L., Shroff, G., Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. En *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.

[30] McKinney, W. (2010). Data Structures for Statistical Computing in Python. En S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference*, 51-56.

[31] Munir, M., Siddiqui, S.A., Dengel, A., Ahmed, S. (2019). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, 7, 1991-2005. https://doi.org/10.1109/ACCESS.2018.2886457

[32] Omohundro, S. M. (1989). *Five Balltree Construction Algorithms* (TR-89-063). International Computer Science Institute.

[33] Park, D., Hoshi, Y., Kemp, C.C. (2017). A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder. *CoRR*, abs/1711.00614. https://doi.org/10.48550/arXiv.1711.00614

[34] Pateiro, B., Sánchez, C. (2022). *Análisis Multivariante*. Apuntes de la materia, Universidad de Santiago de Compostela.

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[36] Preiss, B.R. (1999). *Data Structures and Algorithms with Object Oriented Design Patterns in Java*. John Wiley & Sons.

[37] Ramaswamy, S., Rastogi, R., Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. En *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00)*, 427-438. https://doi.org/10.1145/342009.335437

[38] Rousseeuw, P., Driessen, K. (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics*, 41(3), 212-223. https://doi.org/10.1080/00401706.1999.10485670

[39] Schmidl, S., Wenig, P., Papenbrock, T. (2022). Anomaly Detection in Time Series: A Comprehensive Evaluation. *PVLDB*, 15(9), 1779-1797. https://doi.org/10.14778/3538598.3538602

[40] Shanker, M., Hu, M.Y., Hung, M.S. (1996). Effect of data standardization on neural network training. *Omega*, 24(4), 385-397. https://doi.org/10.1016/0305-0483(96)00010-2

[41] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. En *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), 41-50. https://doi.org/10.1109/TETCI.2017.2772792

[42] Shumway, R.H., Stoffer, D.S. (2017). *Time Series Analysis and Its Applications With R Examples*. Springer Texts in Statistics, Springer Cham.

[43] Sutskever, I., Vinyals, O., Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. En Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (Eds.), *Advances in Neural Information Processing Systems*, 27, 3104-3112. Curran Associates, Inc. https://doi.org/10.48550/arXiv.1409.3215

[44] Thimonier, H., Popineau, F., Rimmel, A., Doan, B.L., Daniel, F. (2023). Comparative Evaluation of Anomaly Detection Methods for Fraud Detection in Online Credit Card Payments. *arXiv preprint arXiv:2312.13896*. https://doi.org/10.48550/arXiv.2312.13896

[45] Tian, R., Liboni, L., Capretz, M. (2022). Anomaly Detection with Convolutional Autoencoder for Predictive Maintenance. En *2022 9th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 241-245. https://doi.org/10.1109/ISCMI56532.2022.10068441

[46] Torres, J. (2020). *Python Deep Learning: Introducción práctica con Keras y TensorFlow 2*. Marcombo.

[47] Tuli, S., Casale, G., Jennings, N.R. (2022). TranAD: Deep Tranformer Networks for Anomaly Detection in Multivariate Time Series Data. *arXiv preprint arXiv:2201.07284*. https://doi.org/10.48550/arXiv.2201.07284

[48] Van Rossum, G., Drake Jr, F.L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

[49] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention Is All You Need. *arXiv preprint arXiv:1706.03762*. https://doi.org/10.48550/arXiv.1706.03762

[50] Wang, X., Lin, J., Patel, N., Braun, M. (2018). Exact variable-length anomaly detection algorithm for univariate and multivariate time series. *Data Mining and Knowledge Discovery*, 32, 1806-1844. https://doi.org/10.1007/s10618-018-0569-7

[51] Wei, W. (2019). *Multivariate Time Series Analysis and Applications*. John Wiley & Sons Ltd, Oxford.

[52] Wenig, P., Schmidl, S., Papenbrock, T. (2022). TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms. *PVLDB*, 15(12): 3678 - 3681. https://doi.org/10.14778/3554821.3554873

[53] Williams, R., Zipser, D. (1995). Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. En *Back-propagation: Theory, Architectures and Applications*, 433-486.