ISSN 1688-2806


Universidad de la República
Facultad de Ingeniería


# Detección de Anomalías en Series Esparsas

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Juan Ignacio Fernández, Facundo Guillén

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutores

Sergio Martínez ..... Universidad de la República
Gabriel Gomez ..... Universidad de la República

Tribunal

José Acuña ..... Universidad de la República
Germán Capdehourat ..... Universidad de la República
Pablo Massaferro ..... Universidad de la República

Montevideo
miércoles 18 octubre, 2023

*Detección de Anomalías en Series Esparsas*, Juan Ignacio Fernández, Facundo Guillén.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 82 páginas.
Compilada el miércoles 18 octubre, 2023.
http://iie.fing.edu.uy/

# Agradecimientos

A nuestras familias, por su apoyo incondicional, paciencia y comprensión durante todo el proceso de desarrollo de este proyecto, que fueron fundamentales para nuestra motivación y bienestar emocional.

A nuestros tutores, Sergio y Gabriel. Su experiencia, conocimientos y disposición para resolver nuestras dudas y brindarnos dirección fueron clave para el éxito de nuestro proyecto.

Al Instituto de Ingeniería Eléctrica (IIE), específicamente al Grupo de Detección de Anomalías, por su valiosa contribución a nuestro trabajo. Las publicaciones, referencias y consejos proporcionados por este grupo de investigación fueron de gran ayuda para comprender el estado del arte en el campo y nos permitieron mejorar nuestra propuesta.

Por último, pero no menos importante, a la empresa Telefónica por brindarnos la oportunidad de trabajar en un proyecto tan interesante y por proporcionarnos los datos necesarios para llevar a cabo nuestra investigación.

Esta página ha sido intencionalmente dejada en blanco.

# Resumen

Las series temporales son una de las formas más extendidas de capturar información en la industria. Cualquiera sea el dominio, existen cantidades variables en el tiempo, cuantificables y medibles que proveen información valiosa cuando se analizan en forma correcta. La prevalencia de las series temporales en el sector productivo ha generado una creciente demanda de sistemas capaces de resolver diversos problemas que las involucran, incluyendo la detección de anomalías o valores atípicos.

El objetivo principal de este proyecto fue investigar y desarrollar algoritmos para la detección de anomalías en series de tiempo esparsas, las cuales se caracterizan por presentar una proporción significativa de valores nulos durante largos períodos de tiempo.

Se investigaron modelos de *clustering*, basados en estados, de *forecasting* y se desarrollaron modelos a medida basados en redes neuronales bayesianas. Algunos de los modelos desarrollados se integraron en un entorno de producción para su funcionamiento en un ambiente realista.

El proyecto también involucra el desarrollo de una metodología para la generación de anomalías sintéticas de distintos tipos, necesaria para desarrollar y evaluar los modelos de detección.

Esta página ha sido intencionalmente dejada en blanco.

# Tabla de contenidos

Tabla de contenidos

# Capítulo 1

# Introducción

Las series temporales son una de las formas más extendidas de capturar información en la industria. Cualquiera sea el dominio, existen cantidades variables en el tiempo, cuantificables y medibles que proveen información valiosa cuando se analizan en forma correcta. La prevalencia de las series temporales en el sector productivo ha generado una creciente demanda de sistemas capaces de resolver diversos problemas que las involucran, incluyendo la detección de anomalías o valores atípicos. Este último campo es de gran interés en el análisis de este tipo de series, ya que resulta central para comprender el comportamiento normal de los flujos de datos y detectar situaciones que se desvían de lo esperado.

El objetivo principal de este proyecto fue investigar y desarrollar algoritmos para la detección de anomalías en series de tiempo esparsas, las cuales se caracterizan por presentar una proporción significativa de valores nulos durante largos períodos de tiempo. Se implementaron y evaluaron modelos de aprendizaje automático de diversos tipos, integrando algunos de ellos en un entorno de producción, para su funcionamiento en un ambiente realista. El propósito es integrar la solución en una aplicación que permita la detección de anomalías en tiempo real, lo que contribuirá a mejorar la eficiencia y robustez de diversos sistemas de monitoreo y análisis, frecuentes en una amplia variedad de campos de aplicación.

## 1.1. Antecedentes

Este proyecto se enmarca en el trabajo del grupo de Detección de Anomalías del Instituto de Ingeniería Eléctrica (IIE), integrado por docentes de los departamentos de Procesamiento de Señales y de Telecomunicaciones. El grupo comenzó su trabajo en 2016 a partir de un primer acuerdo con Telefónica Móviles (Movistar) orientado al análisis y propuesta de algoritmos para detección de anomalías, identificación de fraudes, ataques o alteraciones en el funcionamiento de los servicios. Este tipo de análisis se enmarca en el área conocida como *Big Data Analytics* y se trata de un tema con importante actividad a nivel académico e interés del sector productivo.

En los años siguientes se ha continuado trabajando conjuntamente en subsi-

guientes etapas que ampliaron y profundizaron los aspectos abordados, valorándose la conformación de un equipo de trabajo conjunto entre el equipo universitario y el equipo de profesionales de Telefónica. Esto ha facilitado la comprensión del problema por parte del personal académico y la entrega del producto desarrollado a Telefónica, lográndose una destacada transferencia de conocimiento entre ambos equipos. A partir de este trabajo conjunto se ha contribuido a la formación de recursos humanos en la temática, con impacto tanto en el ámbito académico como profesional.

En las etapas anteriores se identificaron, implementaron y evaluaron diferentes algoritmos para la detección de anomalías. Se trabajó inicialmente con modelos SARIMA y filtrado de Kalman, implementándose una primera maqueta que se encuentra funcionando en producción. Se generaron además herramientas para la validación por un experto humano de las detecciones, lo cual es importante para la correcta evaluación del desempeño de los algoritmos y técnicas desarrolladas.

En el marco de estas actividades se evaluaron nuevas técnicas basadas en Redes Neuronales y Variational Autoencoders (VAEs) y se incorporó el análisis multivariado usando técnicas basadas en GANs (Generative Adversal Neural Networks). Se realizaron evaluaciones primarias de desempeño para el escenario de interés. Estos desarrollos se incluyen en una etapa de trabajo en curso donde se están agregando funcionalidades y se está migrando la implementación a una plataforma más moderna.

Desde el punto de vista académico, el grupo ha publicado varios trabajos en congresos arbitrados [18–24, 41] y ha obtenido financiación para el desarrollo de proyectos de investigación en la temática tanto del fondo María Viñas de la ANII como de CSIC-I+D.

También se realizó un análisis primario de algunas técnicas apropiadas para las series de comportamiento discreto o esparsas que serán objeto de análisis con más profundidad en este proyecto.

## 1.2. Motivación

La detección de anomalías en series temporales es una tarea importante en diversos sectores de la industria. En particular, resulta crucial para el monitoreo de procesos de manufactura, de la infraestructura en centros de datos, en análisis de datos médicos y financieros, entre otros. Identificar una anomalía puede mejorar el tiempo de reacción ante un evento relevante, como una potencial falla en una línea de producción o un sistema informático, ciberataques o enfermedades. Por lo tanto, la detección temprana es de interés central. Debido a esto, varios algoritmos de detección en series de tiempo han sido desarrollados y evaluados a lo largo de los años. También se han publicado varias investigaciones con recopilaciones sobre el tema, por ejemplo [8, 54].

Las series esparsas son también conocidas como “de demanda intermitente” en la literatura y son frecuentes en la industria del *retail*, donde existen varias investigaciones y recopilaciones [33, 34], y de donde proviene la mayor parte de la nomenclatura usada para describirlas. Sin embargo, la literatura específicamente

sobre detección de anomalías en este tipo de series es escasa.

Por otro lado, la empresa Telefónica, en el marco del acuerdo de trabajo existente con el IIE ha demostrado interés en extender los desarrollos ya realizados por el grupo de Detección de Anomalías a series esparsas. En particular, la empresa cuenta con series provenientes de transacciones de comercio electrónico que se pueden caracterizar como esparsas, donde se han identificado comportamientos anómalos indicativos de falla de sistemas informáticos o de ciberataques.

## 1.3. Objetivos

### 1.3.1. Objetivos Generales

Este proyecto tiene como objetivo principal relevar, analizar, implementar y evaluar métodos de detección de anomalías que puedan aplicarse específicamente a series de tiempo esparsas.

El tipo de datos y de anomalías a detectar son los provistos por la empresa Telefónica. Fue indicado por el cliente que los datos solo contienen valores normales, es decir, que no debe considerarse la existencia de anomalías en los mismos. Por lo tanto, un objetivo intermedio es producir de forma sintética los valores anómalos. Las anomalías generadas deberán ser una representación de los comportamientos anómalos que se esperan en estas series, descritos a alto nivel.

Además, se busca integrar uno o mas de estos métodos en una aplicación desarrollada por el grupo de trabajo del IIE, donde ya se encuentran en el entorno de producción de Telefónica algunos algoritmos de detección para otro tipo de series. La aplicación se encarga de la comunicación entre los modelos de detección y una base de datos para series temporales InfluxDB [29], creando un sistema que permite la detección de anomalías en tiempo real.

### 1.3.2. Objetivos Específicos

Consisten en la investigación, selección, implementación y evaluación de modelos de aprendizaje automático:

1. Realizar una exhaustiva investigación de la literatura y los métodos disponibles para abordar el problema de detección de anomalías en series esparsas.

2. Definir de manera clara y precisa el problema, los tipos de datos y de anomalías a detectar.

3. Generar anomalías sintéticas basado en descripciones de alto nivel provistas.

4. Implementar al menos tres modelos de detección. Evaluar y comparar su desempeño.

5. Poner al menos uno de los modelos implementados en el entorno de producción de Telefónica.

### 1.3.3. Alcance

En primer lugar, se define el problema de detección de anomalías en series esparsas, proporcionando una descripción de los distintos tipos de anomalías que se consideran en este trabajo. Posteriormente, se lleva a cabo una revisión de los métodos disponibles para la detección de estas anomalías.

Entre los distintos tipos de modelos que se consideran para abordar este problema, se encuentran los modelos de pronóstico basados en aprendizaje profundo, modelos de estados, técnicas de *clustering* y modelos de factorización. También se consideran otras variantes de estos o incluso otros tipos de modelos. Se seleccionaron algunas de las técnicas investigadas para evaluar su funcionamiento y desempeño.

Se utilizaron los datos facilitados por Telefónica y la evaluación se realizó con anomalías simuladas.

Finalmente se integró alguno de los algoritmos en un entorno de producción y se evalúa la factibilidad de su funcionamiento para detección en tiempo real.

# Capítulo 2

# Marco Teórico

## 2.1. Definición y Tipos de Anomalías

Se define una anomalía en un conjunto de datos como un patrón u observación que presenta un comportamiento fuera de lo esperado. La detección de anomalías se enfoca en identificar muestras, eventos o subconjuntos de datos que presentan este tipo de comportamientos. El propósito de la detección es exclusivamente la identificación en el tiempo y no buscar explicaciones para dichos sucesos.

Es importante conocer la naturaleza de la anomalía a la hora de seleccionar la técnica a utilizar para la detección, ya que cada tipo de anomalía puede requerir enfoques y métodos diferentes. La clasificación de los tipos de anomalías es fundamental para comprender su diversidad, y poder abordarlas de manera efectiva. Es usual clasificar las anomalías en tres diferentes tipos [8]:

1. *Anomalías puntuales*: Una anomalía puntual es el tipo más simple que se puede definir y consiste en una única instancia o muestra anómala en la serie de datos. En este caso el comportamiento anormal se refleja en una única muestra, es decir, no se considera que un conjunto consecutivo de anomalías puntuales compongan una anomalía colectiva.

2. *Anomalía contextual*: Cuando lo identificado como anormal lo es solo en un determinado contexto se le denomina anomalía contextual o condicional. El contexto quedará definido por el problema particular y por el conjunto de datos con el que se trabaje, y es usual definirlo como una ventana de tiempo en el pasado inmediatamente anterior a la observación que se quiere clasificar.

3. *Anomalías colectivas*. Una anomalía colectiva se da cuando un conjunto contiguo de observaciones es anómalo con respecto a todo el conjunto de datos. En este caso el comportamiento anormal se extiende en el tiempo más allá de una muestra. Se considera todo el rango de tiempo como una única anomalía.

En la figura 2.1, se presentan ejemplos de tres tipos de anomalías. Notar que en la clasificación anterior las clases no son mutuamente excluyentes. Por ejemplo, en

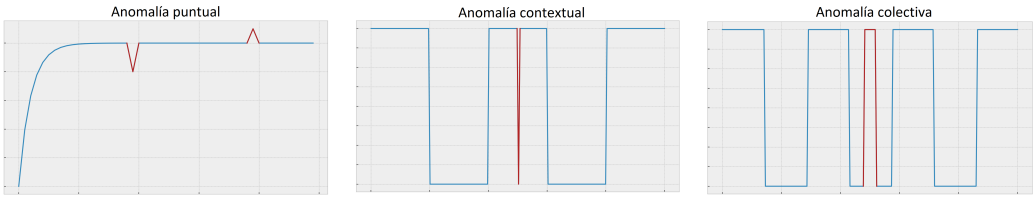



Figura 2.1: Tipos de anomalías.

la figura 2.1b, se puede observar que la anomalía identificada es tanto contextual como puntual, ya que se desvía significativamente del contexto en el que se presenta (el valor superior de la onda cuadrada periódica) y a su vez es una observación individual anormal única.

No debe confundirse la detección de anomalías con el problema de *novelty detection* en donde el objetivo es encontrar patrones emergentes o poco frecuentes en los datos que pueden ser inicialmente confundidos con comportamientos anómalos pero que son conceptualmente normales (ver [47] para una descripción general sobre este tema).

## 2.2. Propiedades de series temporales

En esta sección se describen algunas de las propiedades que caracterizan a las series de tiempo. No pretende ser una lista exhaustiva, pues se entiende que las propiedades descritas son suficientes para caracterizar las series que se usaron en la etapa de análisis. Para más detalles sobre análisis, modelado y propiedades de series temporales en general se puede consultar, por ejemplo [11].

### 2.2.1. Esparsidad

El concepto de esparsidad se encuentra comúnmente en el contexto del análisis numérico y se refiere generalmente a vectores y matrices cuyos componentes tienen una proporción muy alta de valores nulos. En el contexto de series temporales, puede definirse según la proporción de valores nulos en el total de muestras en un intervalo de tiempo dado. En este sentido, se puede clasificar a una serie como esparsa si su esparsidad supera cierto umbral. Una comparación entre una serie esparsa y no esparsa puede observarse en la figura 2.2

En ocasiones, las series pueden no resultar esparsas al aplicar directamente la anterior definición pero se pueden transformar llevándolas a representaciones de este tipo. Por ejemplo, es usual encontrar series con valor medio aproximadamente constante, pudiendo ser la serie formada por los datos que se aparten de dicho valor medio, una serie esparsa.

Por otro lado, si los datos son ruidosos (por ejemplo, si provienen de algún tipo de sensor eléctrico), prácticamente no existirán valores nulos. Si el *Signal to Noise Ratio* (SNR) es lo suficientemente alto, las muestras que solo contienen

ruido pueden truncarse a cero mediante una umbralización, permitiendo aplicar la definición de esparsidad a la serie resultante.

Un problema de la definición anterior es que necesita una cantidad fija de muestras y una serie de tiempo, por definición, no tiene largo fijo e incluso podría considerarse infinita. Es por esto que, en general, el concepto de *esparsidad* no es usado con frecuencia en la literatura cuando se refiere a series de tiempo y se recurre a otras caracterizaciones más apropiadas, que se describen en la sección siguiente.

### 2.2.2. Caracterización de series esparsas

Debido a la naturaleza particular de las series esparsas, se han desarrollado métodos para clasificarlas y simplificar la elección de modelos de pronóstico [33]. Si bien el problema de pronosticar no es equivalente al de detección de anomalías, las estadísticas utilizadas para caracterizar las series son generales y pueden aportar información relevante acerca de los datos disponibles.

Syntetos y Boylan proponen un esquema de clasificación basado en el *intervalo interdemanda medio* (ADI por sus siglas en inglés) [57]:

$$ADI = \frac{\sum_{i=1}^{N} t_i}{N} \tag{2.1}$$

donde $t_i$ es el tiempo entre dos muestras no nulas consecutivas. El $ADI$ representa una medida de intermitencia. También se define el coeficiente de variación $CV$:

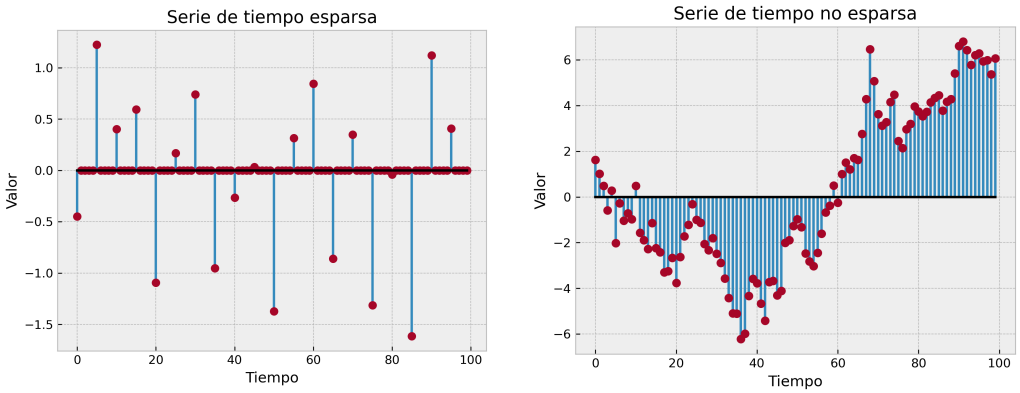
$$CV = \frac{\sqrt{\frac{\sum_{i=1}^{N}(\epsilon_i - \epsilon)^2}{N}}}{\epsilon} \tag{2.2}$$

$$\epsilon = \frac{\sum_{i=1}^{N} \epsilon_i}{N} \tag{2.3}$$

con $\epsilon_i$ siendo los valores no nulos de la serie y $\epsilon$ su promedio. Puesto de otra manera, se trata de la desviación estándar dividido la media de los datos no nulos.

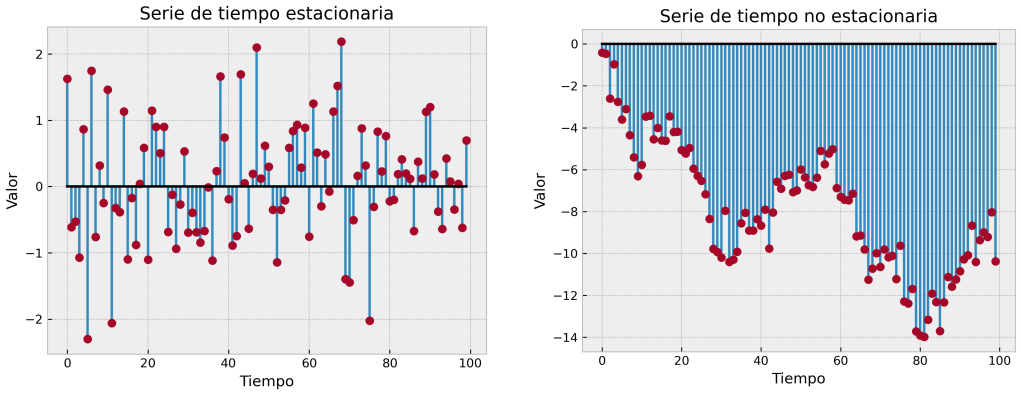
Basado en estos dos parámetros se definen en [57] cuatro categorías, donde los umbrales se establecieron en forma empírica según los desempeños de diferentes métodos de pronóstico, propuestos en [10, 57]:

- Suaves: $ADI < 1,32$ y $CV^2 < 0,49$. La serie presenta valores regulares tanto en magnitud como en frecuencia.
- Errática: $ADI < 1,32$ y $CV^2 > 0,49$. Hay regularidad en frecuencia pero no en magnitud.
- Intermitente: $ADI > 1,32$ y $CV^2 < 0,49$. La actividad es intermitente con poca variabilidad en magnitud.
- Desigual (*lumpy*): $ADI > 1,32$ y $CV^2 > 0,49$. Actividad intermitente y con mucha variabilidad en magnitud.




(a) (b)

Figura 2.2: Ejemplo de serie de tiempo esparsas 2.2a y no esparsa 2.2b.



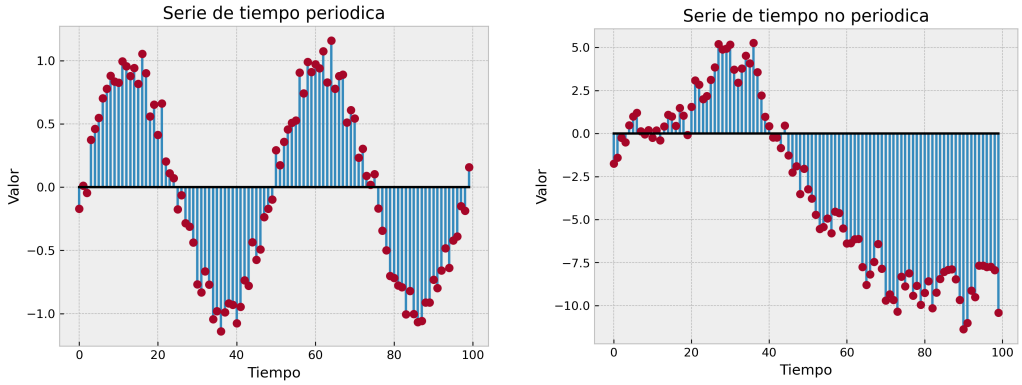

(a) (b)

Figura 2.3: Ejemplo de serie de tiempo estacionaria 2.3a y no estacionaria 2.3b.

### 2.2.3. Estacionariedad

La estacionariedad es una propiedad fundamental en los procesos estocásticos [9] (muchas veces asociados a series de tiempo) en la que se basan muchos de los modelos que buscan aprender el comportamiento de series temporales. De manera informal, la estacionariedad se puede definir como la propiedad de que la estadística de una instancia puntual de la serie es independiente del tiempo. En forma gráfica, las series estacionarias suelen oscilar en torno a un valor medio con varianza constante.

Es importante destacar que en muchas ocasiones las series de tiempo reales no son estacionarias, lo que puede dificultar su análisis y modelado. No obstante, es posible realizar ciertas transformaciones para volverla estacionaria, como por ejemplo la diferenciación. Esta técnica consiste en tomar la diferencia entre una observación y su observación anterior, lo que puede reducir o eliminar las tendencias y patrones de variabilidad no estacionarios en la serie. En la figura 2.3 puede visualizarse la propiedad de estacionariedad en una serie de tiempo.




(a) (b)

Figura 2.4: Ejemplo de serie de tiempo periódica 2.4a y no periódica 2.4b.

### 2.2.4. Periodicidad

La periodicidad es una propiedad de las series temporales que se refiere a la repetición de patrones en intervalos de tiempo regulares. Cuando se cuenta con un índice temporal sobre la serie, ya sea días, horas, minutos, segundos o cualquier otra unidad, es posible aprovechar esta información para analizar y comprender su comportamiento con mayor facilidad aprovechando la repetición de patrones en el tiempo.

La periodicidad puede tener diferentes frecuencias, lo que significa que los patrones pueden repetirse en diferentes intervalos de tiempo. Por ejemplo, una serie de ventas puede ser periódica con una frecuencia diaria, semanal, mensual o anual, dependiendo del tipo de negocio y del comportamiento de los clientes. Un claro ejemplo de una serie periódica en comparación a una no periódica puede observarse en la figura 2.4 .

## 2.3. Métricas

Si bien las métricas comúnmente usadas en problemas de clasificación, como *precision*, *recall* y *F1*, son útiles en el caso de anomalías puntuales, son incapaces de cuantificar conceptos importantes propios de las anomalías colectivas o de rango. Dichas métricas, en su forma clásica son definidas de la siguiente manera:

$$Recall = \frac{TP}{TP \ + \ FP} \tag{2.4}$$

$$Precision = \frac{TP}{TP \ + \ FN} \tag{2.5}$$

$$F1 = 2 \cdot \frac{Precision \ \cdot \ Recall}{Precision \ + \ Recall} \tag{2.6}$$

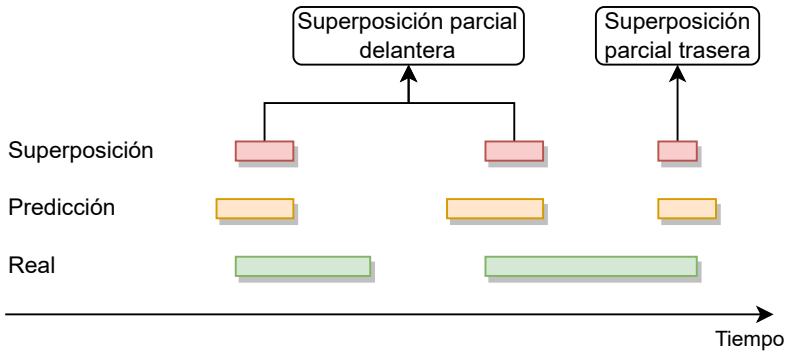



Figura 2.5: Ejemplos de superposición parcial en anomalías de rango.

donde TP, FP y FN son verdaderos positivos, falsos positivos y falsos negativos de las detecciones realizadas, respectivamente. En [58] se propone una extensión de estas métricas con modificaciones para obtener evaluaciones más efectivas y adaptables cuando el problema involucra anomalías colectivas. En este tipo de anomalías resulta de interés cuantificar conceptos como la cardinalidad y posición relativa de superposiciones parciales. Se ilustran en la figura 2.5 algunos ejemplos de superposición parcial de anomalías de rango.

Se consideran los siguientes factores a la hora de calcular estas métricas:

- *Existencia*: La anomalía debe ser encontrada, en el sentido de que al menos un punto del rango real debe ser clasificado como tal por el clasificador.
- *Tamaño*: La cantidad de puntos de solapamiento entre la anomalía real y la detectada.
- *Posición*: La posición relativa entre el rango real y la detección.
- *Cardinalidad*: La cantidad de rangos no continuos detectados que intersectan el rango de una anomalía real.

Matemáticamente, las fórmulas de *precision* y *recall* extendidas se expresan según el conjunto de anomalías de rango reales $R = \{R_1, \ldots, R_{N_r}\}$ con $R_i$ un conjunto de índices de tiempo consecutivos y $P = \{P_1, \ldots, P_{N_p}\}$ el conjunto de rangos de anomalías detectadas. Se definen además las funciones auxiliares $\omega(\cdot)$ *función de tamaño de superposición*, $\gamma(\cdot)$ *función de cardinalidad de superposición* y $\delta(\cdot)$ *función de sesgo posicional*, que deben cumplir las restricciones: $0 \leq \gamma \leq 1$, $0 \leq \omega \leq 1$ y $\delta \geq 1$. Las fórmulas de cálculo de *precision* y *recall* extendidas son las siguientes:

$$Recall_T(R, P) = \frac{\sum_{i=1}^{N_r} Recall_T(R_i, P)}{N_r} \tag{2.7}$$

$$\begin{aligned} Recall_T(R_i, P) =& \alpha \times ExistenceReward(R_i, P) \\ &+ (1 - \alpha) \times OverlapReward(R_i, P) \end{aligned} \tag{2.8}$$

$$\begin{aligned} OverlapReward(R_i, P) =& CardinalityFactor(R_i, P) \\ & \times \sum_{j=1}^{N_p} \omega(R_i, R_i \cap P_j, \delta) \end{aligned} \tag{2.9}$$

$$Precision_T(R, P) = \frac{\sum_{i=1}^{N_r} Precision_T(R, P_i)}{N_r} \tag{2.10}$$

$$\begin{aligned} Precision_T(R, P_i) =& CardinalityFactor(R, P_i) \\ & \times \sum_{j=1}^{N_r} \omega(P_i, P_i \cap R_j, \delta) \end{aligned} \tag{2.11}$$

con

$$ExistenceReward(R_i, P) = \begin{cases} 1, \text{ si } \sum_{j=1}^{N_p} |R_i \cap P_j| \geq 1 \\ 0, \text{ en otro caso} \end{cases} \tag{2.12}$$

$$CardinalityFactor(R_i, P) = \begin{cases} 1, \text{ si } R_i \text{ se intersecta con un solo } P_i \in P \\ \gamma(R_i, P), \text{ en otro caso} \end{cases} \tag{2.13}$$

El parámetro $\alpha$ controla el peso relativo que se le da a la existencia de la anomalía dentro del conjunto de predicciones. Los conceptos de tamaño, posición y cardinalidad se encuentran encapsulados dentro del término *OverlapReward*. Notar que $\alpha$ no es necesario al calcular *precision* ya que esta métrica intrínsecamente debe cuantificar la calidad de las predicciones y no su existencia.

Se definen cuatro tipos generales de estas métricas según como se definan las funciones auxiliares $\omega$, $\delta$ y $\gamma$ (o lo que es lo mismo, el término *OverlapReward*):

- *Sesgo plano*: Todas las muestras del intervalo tienen el mismo peso
- *Sesgo frontal*: Se le da más importancia a las predicciones en el inicio del intervalo.
- *Sesgo trasero*: Se le da más importancia a las predicciones en el final del intervalo.
- *Sesgo medio*: Se le da más importancia a las predicciones en el centro del intervalo.

Para más detalles se refiere a la publicación [58].

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 3

# Modelos

En este capítulo se describen los modelos investigados y su uso para detección de anomalías en diferentes tipos de series temporales, incluyendo las series de tiempo esparsas. Se provee un panorama general de la arquitectura y funcionamiento de cada modelo, y se detallan las modificaciones requeridas para su uso como detectores de anomalías.

## 3.1. Modelos de Markov

Los modelos de Markov [48] son una familia de modelos basados en el concepto de estado que se utilizan usualmente para modelar sistemas de comportamiento pseudoaleatorio. La hipótesis es que el comportamiento futuro de los estados del sistema depende únicamente del estado actual. Más formalmente, dada la secuencia de estados $\theta_0, \theta_1, \theta_2, \ldots, \theta_i$ observados desde el tiempo inicial hasta el presente, se asume que:

$$P(\theta_{i+1} = q|\theta_0, \theta_1, \cdots, \theta_i) = P(\theta_{i+1} = q|\theta_i) \tag{3.1}$$

donde $q$ representa un estado particular del sistema y $P(\cdot)$ es una función de probabilidad.

### 3.1.1. Cadenas de Markov

Dentro de la familia de los modelos de Markov, el más simple es la cadena de Markov. Este modelo está especificado por tres parámetros: el primero es un conjunto $Q = \{q_1, q_2, \ldots, q_n\}$ de los N estados posibles del sistema. El segundo es una matriz $A$ asociada al conjunto $Q$ de dimensiones $N \times N$ con las probabilidades de transición entre estados, es decir, $a_{ij}$ es la probabilidad de transición del estado $q_i$ hacia $q_j$. Finalmente, el vector $\Pi_o = (\pi_1, \pi_2, \ldots, \pi_n)$ cuyos elementos $\pi_i$ representan la probabilidad de que el estado inicial del sistema sea $q_i$. Se asume que se tiene acceso al estado del sistema en un tiempo dado y que el sistema es autónomo, es decir, no tiene estímulos externos.

Una representación esquemática de una cadena de Markov con tres estados $q_0 = HOT$, $q_1 = COLD$ y $q_2 = WARM$ se muestra en la figura 3.1. Aquí, las probabilidades $a_{ij}$ se indican junto a las flechas que marcan las transiciones entre estados.

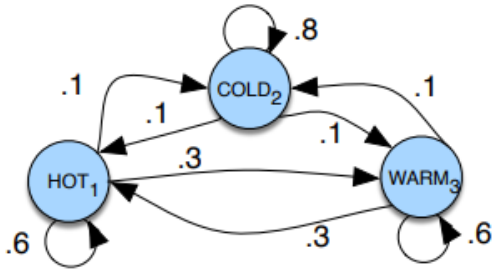



Figura 3.1: Ejemplo de Cadena de Markov (imagen extraída de [32]).

### 3.1.2. Modelos ocultos de Markov

Abreviados como HMM (por su nombre en inglés: *Hidden Markov Model*) los modelos ocultos de Markov son muy similares a las cadenas de Markov, con la diferencia de que en estos los estados se consideran ocultos o ruidosos, por lo que solo se puede acceder a los mismos a partir de observaciones ruidosas. Los HMM ya han sido estudiados y aplicados a detección de anomalías en series de tiempo con buenos resultados [26].

Son necesarias variables y parámetros adicionales a los de las cadenas de Markov para representar un HMM. Considerando el caso de un HMM en tiempo discreto y con un conjunto discreto de estados, si se tiene el conjunto de observaciones $O = (o_0, o_1, \ldots, o_i)$ hasta el tiempo $t_i$ y, para el caso de observaciones continuas, se tiene un conjunto de distribuciones $B = \{b_0, b_1, \ldots., b_n\}$ donde los elementos $b_i$ son la distribución supuesta para las observaciones ruidosas del estado $q_i$; puede obtenerse la verosimilitud de una observación $o_t$ en el estado $q_i$ calculando $b_i(o_t)$. El tipo de distribución que tendrá el conjunto $B$ es un hiperparámetro del modelo, pero típicamente se utilizan distribuciones Gaussianas.

Matemáticamente, un HMM como el que se describió anteriormente puede expresarse por un sistema de dos ecuaciones, una que describe la evolución del estado y otra que explica las observaciones:

$$\begin{cases} Y_t \sim p_t(y_t|\Theta_t) \\ \Theta_t \sim p_t(\Theta_t|\Theta_{t-1}) \end{cases}$$

donde $p_t(.)$ denota una distribución, $Y_t$ y $\Theta_t$ las variables aleatorias de las observaciones y el estado en el tiempo $t$.

Un ejemplo de un HMM puede verse en la figura 3.2 donde los estados $q_i$ son $q_0 = HOT$ y $q_1 = COLD$. También puede observarse el vector $\Pi$ de probabilidades iniciales y las distribuciones discretas de probabilidad $B_1$ y $B_2$ que forman el vector

$B$ de observaciones. Las probabilidades $a_{ij}$ se indican junto a las flechas que marcan las transiciones entre estados. Para una introducción más detallada de lo que es un Hidden Markoov Model puede consultarse [49].

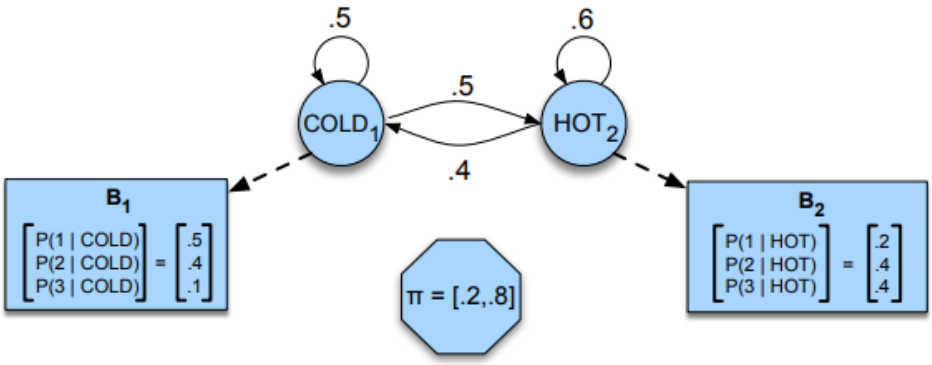



Figura 3.2: Ejemplo de modelo de Markov Oculto ( imagen extraída de [32]).

### Problemas fundamentales

Existen tres posibles problemas que pueden ser resueltos con HMM:

1. Problema 1: Verosimilitud. Dado un HMM caracterizado por sus parámetros $\lambda = (A, B)$ y una secuencia de observaciones $O$, determinar la verosimilitud de dicha secuencia dados los parámetros del modelo, es decir, calcular $P(O|\lambda)$.

2. Problema 2: Decodificación. Similar al problema anterior, para la decodificación se cuenta con un HMM que tiene sus parámetros $\lambda = (A, B)$ aprendidos y una secuencia de observaciones $O$. El interés en este caso es determinar la secuencia de estados ocultos $Q$ más probable dadas las observaciones $O$. Esto será de utilidad cuando se quiera hacer inferencia sobre un conjunto de datos.

3. Problema 3: Aprendizaje. El último problema consiste en aprender los parámetros $\lambda = (A, B)$ dada una secuencia de observaciones $O$ y un conjunto de estados $Q$ asumidos que mejor se ajusten a la secuencia de observaciones.

### Entrenamiento

Con el fin de solucionar el problema de aprendizaje se utiliza el algoritmo de Baum-Welch [5], que es un caso especial del algoritmo de máxima expectativa también conocido como EM (*Expectation Maximization*, en inglés) [12]. Este algoritmo permitirá calcular tanto $A$ como $B$ de forma iterativa.

El algoritmo EM consta de dos pasos llamados estimation-step y maximization-step, los cuales se iteran hasta un nivel deseado de convergencia. El estimation-step consiste en estimar variables latentes y el maximization-step en optimizar los

parámetros del modelo usando las variables estimadas en el estimation-step de modo que sean más verosímiles respecto a los datos observados.

Sea la variable de estado $Q_t$, variable aleatoria discreta con $N$ posibles valores en la que se asume $P(Q_t|Q_{t-1},\ldots,Q_1) = P(Q_t|Q_{t-1})$ independiente del tiempo $t$, se define la matriz de transición de estados como:

$$A = \{a_{ij}\} = P(Q_t = j|Q_{t-1} = i)$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad i = 1,\ldots,N$$

Por otro lado se considera que la distribución inicial de estados esté dada por el vector $\Pi_o = \{\pi_1,\ldots,\pi_N\}$ donde $\pi_i$ representa la probabilidad de que el estado inicial sea el $i$.

Particularmente, para este proyecto se utilizan emisiones Gaussianas, es decir, se asume una distribución Gaussiana para las observaciones de los estados, por lo tanto se definen el conjunto de distribuciones $B$ y el de observaciones $Y$ de la siguiente forma:

$$B = \{b_1,\ldots,b_N\} \qquad b_i = \mathcal{N}(\mu_i,\sigma_i) \quad i = i,\ldots,N$$

$$Y = \{y_1,\ldots,y_T\}$$

Sean $M = \{\mu_1,\ldots,\mu_N\}$ y $\Sigma = \{\sigma_1,\ldots,\sigma_N\}$ los parámetros de las distribuciones de las observaciones, se puede definir el conjunto de parámetros del modelo como $\Theta = (A, B, \Pi, M, \Sigma)$.

El primer paso del algoritmo es estimar $\alpha_i(t)$ y $\beta_i(t)$ que definen las siguientes probabilidades:

$$\alpha_i(t) = P(Y_1 = y_1,\ldots,Y_t = y_t, Q_t = i|\Theta)$$

$$\beta_i(t) = P(Y_{t+1} = y_{t+1},\ldots,Y_T = y_T|Q_t = i,\Theta)$$

Es decir, $\alpha$ es la probabilidad de que se de la secuencia observada $Y = \{y_1,\ldots,Y_t\}$ dado el parámetro $\theta$ y que el estado actual es i, por otro lado $\beta$ es la probabilidad de observar la secuencia $Y = \{y_{t+1},\ldots,Y_T\}$ dado el parámetro $\theta$ y que el estado anterior es i. Notar que tanto $\alpha$ como $\beta$ pueden obtenerse de forma iterativa:

$$\begin{cases} \alpha_i(1) = \pi_i b_i(y_1) \\ \alpha_i(t+1) = b_i(y_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji} \end{cases}$$

$$\begin{cases} \beta_i(T) = 1 \\ \beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(y_{t+1}) \end{cases}$$

Teniendo $\alpha$ y $\beta$ calculados para todos los estados, se procede a aplicar el teorema de Bayes para calcular las siguientes variables de utilidad:

$$\gamma_i(t) = P(Q_t = i|Y,\Theta) = \frac{P(Q_t, Y|\Theta)}{P(Y|\Theta)} = \frac{P(Q_t, Y_1 = y_1, \ldots, Y_T = y_T|\Theta)}{P(Y|\Theta)}$$
$$= \frac{P(Q_t, Y_1 = y_1, \ldots, Y_t = y_t|\Theta).P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|Q_t = i, \Theta)}{P(Y|\Theta)} =$$
$$\frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)} \quad (3.2)$$

$$\zeta_{ij}(t) = P(Q_t = i, Q_{t+1} = j|Y,\Theta) = \frac{P(Q_t = i, Q_{t+1} = j, Y|\Theta)}{P(Y|\theta)}$$
$$= \frac{P(Q_t = i, Y|\Theta).P(X_{t+1} = j|X_t = i).P(Y|Q_{t+1} = j, \Theta).P(Y_{t+1} = y_{t+1}|\Theta)}{P(Y|\Theta)}$$
$$= \frac{\alpha_i(t)a_{ij}b_j(t+i)b_j(y_{t+1})}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)} \quad (3.3)$$

Finalmente teniendo las variables $\alpha$, $\beta$, $\gamma$ y $\zeta$ para todos los estados se procede a realizar el maximization-step calculando los siguientes estimadores para la próxima iteración:

$$\begin{cases} \tilde{\pi_i} = \gamma_i(1) \\ \tilde{a_{ij}} = \frac{\sum_{t=1}^{T-1}\zeta_{ij}(t)}{\sum_{t=1}^{T-1}\gamma_i(t)} \\ \tilde{m_i} = \frac{\sum_{t=1}^{T}\alpha_i(t)\beta_i(t)y_t}{\sum_{t=1}^{T}\alpha_i(t)\beta_i(t)} \\ \tilde{\sigma_i^2} = \frac{\sum_{t=1}^{T}\alpha_i(t)\beta_i(t)(y_t - m_i)^2}{\sum_{t=1}^{T}\alpha_i(t)\beta_i(t)} \end{cases}$$

### HMM para detección de anomalías

El proceso de detección y los criterios de clasificación de datos anómalos se detallan a continuación.

En primer lugar, se entrena un modelo HMM con emisiones Gaussianas utilizando los datos de la serie como observaciones.

Para inferencia, se proporciona al modelo una secuencia de $T$ datos. A continuación, se procederá a calcular la secuencia de estados más probable dadas las observaciones utilizando el algoritmo de Viterbi [17]. Una vez obtenida la distribución de las observaciones del estado estimado para el dato recibido, se compara el valor observado con dicha distribución. En el caso específico de la distribución Gaussiana, se considerará que el dato es una anomalía si su valor se encuentra a una distancia mayor a tres desviaciones estándar de la media, lo que significa que el mismo se encuentra fuera del 99.7 % de los valores esperados para dicha observación.

## 3.2. LSTM Bayesiana

Las redes LSTM han demostrado ser capaces de captar de forma satisfactoria el comportamiento de datos secuenciales [28, 53], gracias a sus mecanismos de aprendizaje a corto y largo plazo. Este tipo de redes son capaces de procesar y analizar secuencias de datos de longitud variable, lo que resulta en un modelo con gran flexibilidad y adaptabilidad a diferentes tipos de datos.

Las redes LSTM ya han sido estudiadas para el problema de detección de anomalías con buenos resultados en [40, 61]. Sin embargo, con base en pruebas preliminares se encontró que los resultados mejoraban utilizando capas bayesianas en lugar de capas normales. En el enfoque bayesiano los parámetros desconocidos o latentes se tratan como variables aleatorias y la red aprende una distribución de estos parámetros condicionada a lo que podemos observar en los datos de entrenamiento. Para una estudio completo de las redes bayesianas se puede consultar [25, 31].

El uso de una red bayesiana implica que el modelo adquiere un enfoque probabilístico, lo que le permite establecer intervalos de confianza para sus predicciones, de forma similar a los modelos para *forecasting*.

### 3.2.1. Modelo

A continuación, se detalla cada uno de los componentes que conforman la arquitectura del modelo. Posteriormente, se presentará un diagrama completo del mismo con el objetivo de proporcionar una visión más clara y completa.

#### Celda LSTM

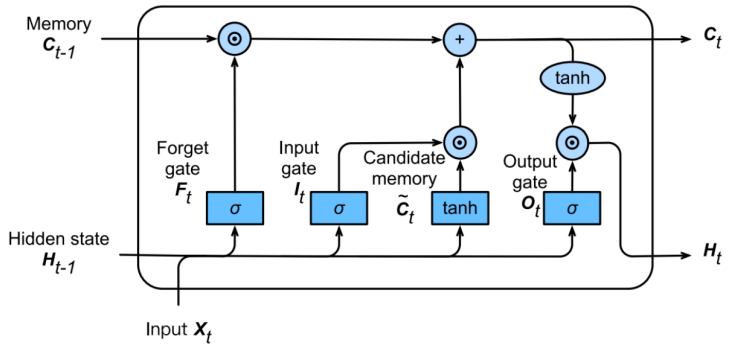



Figura 3.3: Arquitectura de celda LSTM (figura extraída de [55]).

Una celda LSTM [53] es el componente principal de las redes LSTM. En cada paso recurrente de la red, la celda se encargará de generar la salida correspondiente, utilizando tanto la entrada como su propio estado interno conocido como *memoria a largo plazo*. Este estado es generado por la celda LSTM anterior en la secuencia temporal, o en su defecto, por una inicialización predeterminada. Esta memoria a largo plazo es lo que diferencia a las redes LSTM de las clásicas redes

recurrentes ya que permite la retención de información relevante, evitando así el problema del desvanecimiento de gradientes.

Una vez determinada la cantidad de información que debe ser olvidada, el siguiente paso consiste en determinar cuánta nueva información se agregará al estado de la celda. Para ello, se utiliza un bloque llamado *puerta de entrada* o *input gate*, que determina qué porcentaje del nuevo estado estimado debe agregarse al estado anterior. La puerta de entrada utiliza una función sigmoide para calcular el porcentaje de información que debe ser agregado, y luego, una capa con una función de activación de tangente hiperbólica crea un nuevo vector de estado. Este vector es multiplicado por la salida de la puerta de entrada y luego sumado al estado anterior, generando así el estado actualizado de la celda.

La ultima etapa del proceso es puerta de salida u *output gate*. Esta puerta consiste en una capa sigmoide que utiliza únicamente los datos de entrada para determinar cuánto del estado actual de la celda debe utilizarse para generar la salida de la misma. El resultado de la puerta de salida es multiplicado por el estado actual de la celda pasado por una capa tangente hiperbólica para obtener la salida final de la celda LSTM. De esta forma, la puerta de salida controla la cantidad de información que se comparte con la siguiente celda en la secuencia.

Una visualización de una celda LSTM puede verse en la figura 3.3

#### Redes Bayesianas

Una red bayesiana posee la particularidad de considerar los parámetros entrenables y datos de entrada como no deterministas [25]. A diferencia de una red convencional donde en el entrenamiento se optimizan los pesos del modelo de forma de obtener un valor que minimiza una determinada función de costo, en una red bayesiana los pesos son considerados distribuciones y se optimizan de forma que sus parámetros expliquen de la mejor forma los datos de entrenamiento. El utilizar un enfoque basado en redes bayesianas tiene dos grandes ventajas: primero, permite que el modelo sea más robusto ya que al aprender una distribución para los pesos se está de alguna manera regularizando. Por otro lado, permite abordar el problema desde un punto de vista probabilístico, lo cual facilita el método de detección de anomalías.

#### Arquitectura

La arquitectura propuesta para el modelo es una red LSTM de una capa con un largo de secuencia fijo (que será considerado un hiperparámetro) cuya salida se une a una capa densa que irá reduciendo su tamaño hasta llegar a una salida con el valor de la predicción del modelo (ver figura 3.4). Tanto la LSTM como la capa densa serán bayesianas.

### 3.2.2. LSTM para detección de anomalías

En lo que respecta a la detección de anomalías, el proceso se lleva a cabo de la siguiente manera: con el modelo ya entrenado se recibe una secuencia de datos

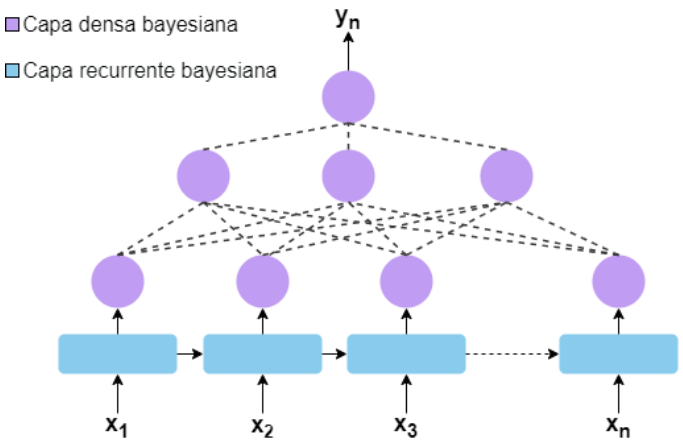



Figura 3.4: Arquitectura del modelo completo.

cuya longitud coincida con la establecida para el bloque LSTM. A continuación, se obtiene una muestra aleatoria de la salida de la red, la cual permite calcular un intervalo de confianza para la predicción. Se compara el dato real con el intervalo de confianza calculado y en el caso de que el dato a clasificar no se encuentre dentro, se considerará que se trata de una anomalía.

## 3.3. Modelos de pronóstico probabilístico

Si bien los modelos de pronóstico (*forecasting*) no se diseñan específicamente para tratar el problema de detección de anomalías se han usado anteriormente con este fin en series de tiempo [34, 40, 43, 56], obtenéendo buenos resultados.

El problema del *forecasting* en el contexto de la detección de anomalías de este proyecto es del tipo no supervisado, donde se intenta predecir una distribución de probabilidad para los valores de muestras futuras basado en información de muestras pasadas y posiblemente otros metadatos. El procedimiento de detección usual consiste en comparar los valores reales observados con alguna estadística de las distribuciones pronosticadas, típicamente un cuantil. El umbral de clasificación así como la estadística utilizada deben considerarse como hiperparámetros y dependen del problema particular, del modelo utilizado, de las características de los datos disponibles y del tipo de anomalías a detectar.

Hay disponibles gran variedad de implementaciones este tipo de modelos que están, en general, bien documentadas y testeadas. Las ventajas de usar estos métodos es que con el esquema de entrenamiento no supervisado no se necesitan datos etiquetados y además permiten el uso de otro tipo de variables que potencialmente pueden mejorar el desempeño, como metadatos estáticos y variables no estáticas pero conocidas a futuro. Por otro lado, se puede ajustar la regla de detección para adaptarla a distintos tipos de datos y anomalías lo cual otorga otro grado de flexibilidad al diseño. Se investigaron dos modelos de este tipo que se describen en las secciones siguientes.

### 3.3.1. DeepAR

DeepAR [16] es un modelo supervisado para predicción probabilística de series temporales basado en una red neuronal profunda, autorregresiva y recurrente (RNN). Admite varios modelos de ruido, permitiendo elegir el más apropiado según las características estadísticas de los datos.

#### Modelo

Dadas $n$ series univariadas objetivo denotadas por $z_{i,t}$, se quiere modelar la distribución condicional

$$P(\mathbf{z}_{i,t_o:T}|\mathbf{z}_{i,1:t_o-1}, \mathbf{x}_{i,1:T}), \tag{3.4}$$

donde $\mathbf{z}_{i,t_o:T} = [z_{i,t_o}, z_{i,t_o+1}, \ldots, z_{i,T}]$ y $t_o$ es el primer punto donde se asume que el valor de la serie es desconocido. $x_{i,1:T}$ son covariantes (variables continuas e independientes que pueden tener poder predictivo de la variable dependiente) que se asumen conocidas tanto en el pasado como en el futuro.

Se asume también que la distribución de salida del modelo es

$$Q_\Theta(\mathbf{z}_{i,t_o:T}|\mathbf{z}_{i,1:t_o-1}, \mathbf{x}_{i,1:T}),$$

y consiste en el producto de funciones de verosimilitud

$$\begin{aligned} Q_\Theta(\mathbf{z}_{i,t_o:T}|\mathbf{z}_{i,1:t_o-1}, \mathbf{x}_{i,1:T}) &= \prod_{t=t_o}^{T} Q_\Theta(z_{i,t_o:T}|\mathbf{z}_{i,1:t_o-1}, \mathbf{x}_{i,1:T}) \\ &= \prod_{t=t_o}^{T} \ell(z_{i,t_o:T}|\theta(\mathbf{h}_{i,t}, \Theta)) \end{aligned}$$

parametrizadas por $\mathbf{h}_{i,t}$, que es la salida de una red multicapa con celdas del tipo LSTM que a su vez depende del parámetro $\Theta$. La verosimilitud $\ell(z_{i,t_o:T}|\theta(\mathbf{h}_{i,t}, \Theta))$ es una distribución fija y sus parámetros están dados por $\theta(\mathbf{h}_{i,t}, \Theta)$, función de la salida de la red.

La función de verosimilitud define el modelo de ruido utilizado. Los más comunes son los modelos gaussianos para datos reales y binomiales negativos para datos enteros positivos. Es posible usar otras funciones de verosimilitud, siempre que se puedan obtener muestras a partir de la distribución de forma eficiente y se pueda evaluar el logaritmo de la función y su gradiente con respecto a los parámetros.

Para ruido gaussiano, la función de verosimilitud es

$$\ell_G(z|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{(2\sigma)^2}} \tag{3.5}$$

Los parámetros de la distribución del siguiente punto $t$ en la serie temporal $i$, se predicen a partir de la salida de la red neuronal $\mathbf{h}_{i,t}$. La media es simplemente una función afín de esta, y la desviación estándar es una función afín compuesta con la función *softplus*, la cual es una versión suavizada de la función de activación $ReLU$. En efecto,

$$\mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu \tag{3.6}$$

$$\sigma(\mathbf{h}_{i,t}) = \log(1 + \exp \mathbf{w}_\sigma^T \mathbf{h}_{i,t} + b_\sigma) \tag{3.7}$$

Para modelar datos enteros positivos se utiliza la distribución binomial negativa. Esta es paramétrica en la media $\mu \in \mathbb{R}^+$ y el parámetro de forma $\alpha \in \mathbb{R}^+$,

$$\ell_B N(z|\mu, \alpha) = \frac{\Gamma(z + 1/\alpha)}{\Gamma(z+1)\Gamma(\frac{1}{\alpha})}(\frac{1}{1+\alpha\mu})^{1/\alpha}(\frac{\alpha\mu}{1+\alpha\mu})^z \tag{3.8}$$

$$\mu(\mathbf{h}_{i,t}) = \log(1 + e^{\mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu}) \tag{3.9}$$

$$\alpha(\mathbf{h}_{i,t}) = \log(1 + e^{\mathbf{w}_\alpha^T \mathbf{h}_{i,t} + b_\alpha}). \tag{3.10}$$

El parámetro $\alpha$ relaciona la varianza de la variable con su media, según la ecuación $var(z) = \mu + \mu^2\alpha$.

### Entrenamiento

A diferencia de los métodos tradicionales en los que el entrenamiento consiste en encontrar los parámetros que minimizan una función de costo, aquí se maximiza el logaritmo de la función de verosimilitud respecto de los pesos de la RNN y de los pesos de las funciones de estimación de parámetros. Matemáticamente:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=t_o}^{T} \log \ell(z_{i,t}|\theta(\mathbf{h}_{i,t})) \tag{3.11}$$

La ecuación 3.11 puede optimizarse directamente utilizando alguna variante de descenso por gradiente.

## 3.3.2. Temporal Fusion Transformer

Similar a DeepAR, el Temporal Fusion Transformer (TFT) [39] es un modelo de predicción probabilística multi-horizonte para series temporales multi-variadas, con la particularidad de que la predicción se basa en cuantiles. Posee una arquitectura moderna del tipo *transformer*, y reporta mejores métricas que DeepAR en varios conjuntos de datos de referencia.

### Modelo

La ecuación que modela la salida del TFT es

$$\hat{y}_i(q, t, \tau) = f_q(\tau, y_{i,t-k:t}, z_{i,t-k:t}, x_{i,t-k:t+\tau}, s_i), \tag{3.12}$$

donde $q$ es el cuantil que se quiere predecir, $t$ es tiempo y $\tau$ es la cantidad de muestras posteriores a $t$ de la predicción. $\hat{y}_i(q, t, \tau)$ es, entonces, el valor predicho del q-ésimo cuantil en el tiempo $t + \tau$. Al igual que en *DeepAR*, las muestras de entrada dependientes del tiempo se dividen en dos tipos: $\mathbf{z}_{i,t}$ son los valores conocidos hasta el tiempo $t$, y $\mathbf{x}_{i,t}$ son los valores conocidos de antemano, tanto en el pasado como en el futuro, y $s_i$ representa las variables estáticas de la serie $i$.

### Arquitectura

La arquitectura simplificada del TFT se muestra en la figura 3.5. Los componentes fundamentales son los siguientes:

- *Mecanismos de compuerta* que descartan componentes no usados de la arquitectura y proveen complejidad a la red.
- *Redes de selección de variables* que seleccionan las variables de entrada relevantes en cada paso temporal.
- *Codificadores de covariantes estáticas* integran las entradas estáticas a la red que condicionan la dinámica del sistema.
- *Procesamiento temporal* para aprender los patrones a corto y largo plazo a partir de las observaciones y de las variables dinámicas conocidas a futuro.
- *Intervalos de predicción* basados en pronóstico probabilístico.

En forma resumida, los procesos llevados a cabo con los datos en la arquitectura del TFT son los siguientes: En primer lugar, se preprocesan los datos para mantener únicamente las *features* que sean relevantes para el aprendizaje. Esto se logra utilizando los bloques *Variable Selection* que se aplican tanto a los variables dinámicas, como a las estáticas. Los datos ingresados a la arquitectura consisten en una serie temporal, con $k$ muestras pasadas, $\tau$ muestras futuras conocidas y datos estáticos invariantes en el tiempo.

Una vez preprocesados estos datos, se envían a varias celdas LSTM que actúan como encoders, las cuales utilizan como entrada las *features* relevantes obtenidas de los datos temporales e incorporan en su estado las *features* obtenidas de los datos estáticos. El objetivo del encoder es el de capturar la información a largo y corto plazo de los datos para luego enviarla hacia el decoder.

El decoder, denominado *Temporal Fusion Decoder* recibe los datos del encoder y aprende las relaciones temporales entre las distintas muestras gracias a su mecanismo de atención. Finalmente, el decoder alimenta a una red densa que predice los cuantiles de los valores de la serie para las $\tau$ muestras futuras.

La descripción del funcionamiento interno de cada componente es extensa y está fuera del alcance de este texto, por lo que se refiere a [39] por más detalles.

### Entrenamiento

El entrenamiento consiste en la minimización del *quantile loss* [62]

$$\mathcal{L}(\Omega, \mathbf{W}) = \sum_{y_t \in \Omega} \sum_{q \in \mathcal{Q}} \sum_{\tau=1}^{\tau_{max}} \frac{QL(y_t, \hat{y}(q, t-\tau, \tau), q)}{M \tau_{max}} \tag{3.13}$$

$$QL(y, \hat{y}, q) = q \,\text{máx}(0, y - \hat{y}) + (1 - q)\,\text{máx}(0, \hat{y} - y) \tag{3.14}$$

donde $\Omega$ es el dominio de las $M$ muestras del conjunto de entrenamiento y $\mathcal{Q}$ es el conjunto de cuantiles de salida.

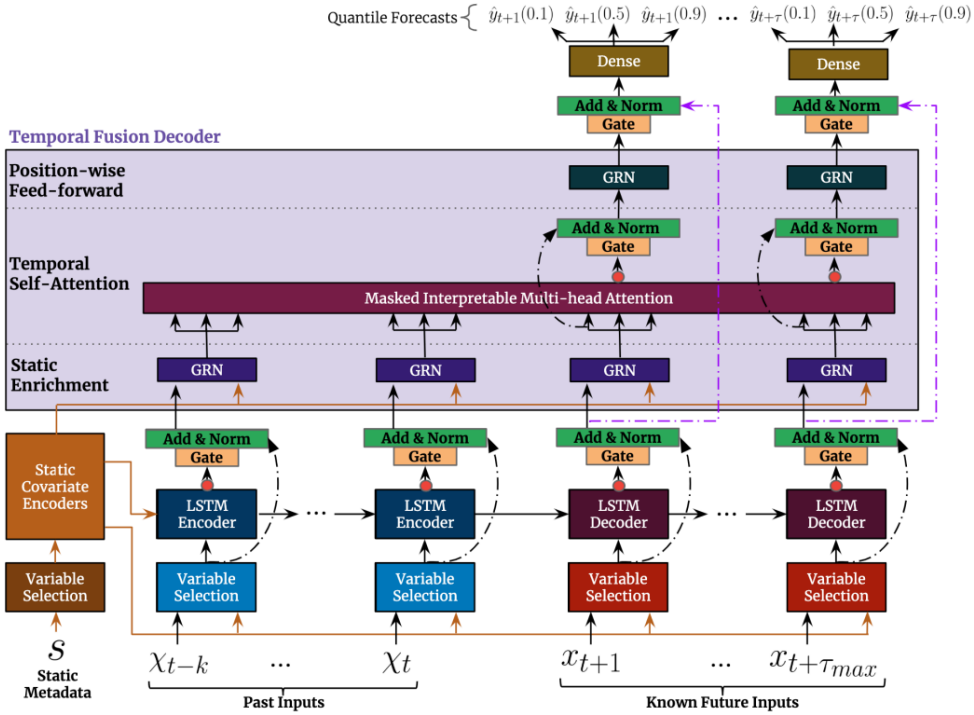



Figura 3.5: Arquitectura del TFT, extraído de [39].

### Intepretabilidad

Una de las características interesantes del TFT es que permite interpretar las relaciones aprendidas analizando algunos de sus componentes individuales. Existen estudios en los cuales se ha trabajado en la interpretación de los resultados del TFT [39], investigando métodos para: 1) evaluar el peso de cada variable de entrada, 2) visualizar patrones temporales, y 3) identificar eventos que potencialmente generaron cambios en la dinámica de las series. Este tipo de métodos agregan más valor al modelo, ya que permiten entender de mejor manera que es lo que aprende.

## 3.4. Máquinas de Factorización

Las Máquinas de Factorización (FM, por sus siglas en inglés) son un modelo supervisado estrechamente relacionado a las Máquinas de Vectores de Soporte (SVM), que han demostrado buenos resultados en datos con gran esparsidad [50]. El uso de FMs para detección de anomalías en datos esparsos con variables categóricas y numéricas ya ha sido estudiado con resultados prometedores [64]. Tienen la ventaja de ser modelos simples con relativamente pocos parámetros entrenables y computacionalmente eficientes, tanto en entrenamiento como en predicción. Se aplican directamente cuando las variables son del tipo categórico, pero pueden aplicarse también a variables numéricas mediante un procesamiento previo ade-

cuado.

### 3.4.1. Modelo

La ecuación que define un modelo FM de grado $d = 2$ es la siguiente:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < \mathbf{v}_i, \mathbf{v}_j > x_i x_j \tag{3.15}$$

donde $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^n$ y $\mathbf{V} \in \mathbb{R}^{n \times k}$. Aquí, se sigue la notación del artículo original y $< \cdot, \cdot >$ indica el producto escalar de dos vectores.

Cada fila $\mathbf{v}_i$ de la matriz $\mathbf{V}$ describe a la variable $i$ con $k$ factores. $k \in \mathbb{N}_0^+$ es un hiper-parámetro que define la dimensión de la factorización. $w_0$, $w_i$ y $\hat{w}_{i,j} =< \mathbf{v}_i, \mathbf{v}_j >$ son parámetros entrenables y modelan el sesgo, el peso de la variable $i$ y el peso de la interacción entre las variables $i$ y $j$, respectivamente.

### 3.4.2. Entrenamiento

El entrenamiento se puede realizar con cualquier variante de descenso por gradiente, y como se demuestra en [50], es $O(n)$ (lineal) en la cantidad de características de entrada.

### 3.4.3. FM para detección de anomalías

Para la detección de anomalías en series temporales se usan FM para regresión y se clasifica según la comparación con un umbral fijo. Los datos de entrada deben preprocesarse formando un vector de características de la siguiente manera: si el valor de la serie univariada a predecir es $y = x_i$, el vector de entrada es

$$\vec{x} = \begin{bmatrix} x_{i-n} & \cdots & x_{i-1} & h_i & d_i & f_i & m_i \end{bmatrix} \tag{3.16}$$

donde $n$ es el tamaño de la ventana temporal de predicción, que es un hiperparámetro del modelo. $h_i$, $d_i$, $f_i$ y $m_i$ son datos temporales de la muestra $i$; hora del día, día de la semana, día del mes y mes del año, respectivamente.

## 3.5. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise [15]) es un algoritmo de *clustering* no supervisado ampliamente utilizado para detectar anomalías en conjuntos de datos ruidosos. El algoritmo es capaz de encontrar grupos de puntos cercanos y separarlos de otros grupos mediante la definición de un umbral de distancia y un número mínimo de puntos necesarios para formar un grupo. De esta manera, los puntos que no pertenecen a ningún grupo y están lejos de los grupos existentes son considerados como anomalías. El algoritmo DBSCAN es especialmente útil para conjuntos de datos con formas y tamaños arbitrarios y

es menos sensible a los valores atípicos que otros algoritmos de clustering basados en centroides.

### 3.5.1. Modelo

Para explicar el modelo es importante definir los siguientes conceptos:

**Definición 1** *($\epsilon$-vecindad de un punto) La $\epsilon$-vecindad de un punto p, denotada por $N_\epsilon(p)$, es definida por $N_\epsilon(p) = \{q \in D | dist(p,q) \leq \epsilon\}$.*

La definición anterior sugiere un enfoque inicial para definir un clúster, que consiste en establecer que un punto pertenece a un clúster si existe un mínimo de MinPts puntos en su $\epsilon$-vecindad. Sin embargo, esta aproximación no tiene en cuenta la presencia de diferentes tipos de puntos que pueden darse dentro de un clúster como los puntos de borde y los puntos del centro (ver figura 3.6 ), que dada su distinta naturaleza dificultan la selección de un criterio para la definición del clúster.

Los puntos de borde en un clúster existen en menor cantidad en su $\epsilon$-vecindad en comparación con los puntos del centro, lo que sugiere la necesidad de establecer un valor de MinPts pequeño para incluir ambos tipos. Sin embargo, esta solución puede generar problemas en presencia de ruido, además de que no representa adecuadamente los puntos centrales del clúster. Para superar esta dificultad, se han propuesto las siguientes definiciones adicionales:

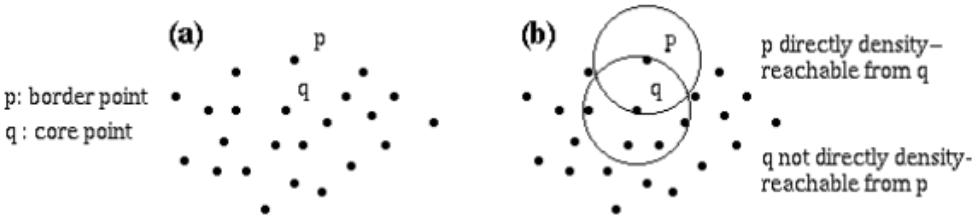



Figura 3.6: Puntos borde y puntos centrales [15].

**Definición 2** *(directamente accesible por densidad) Un punto p es directamente accesible por densidad desde un punto q con respecto a $\epsilon$, $MinPts$ si*

1. $p \in N_\epsilon(q)$
2. $|N_\epsilon(q)| \geq MinPts$ *(condición de punto núcleo)*

**Definición 3** *(Accesible por densidad) 3.7 Un punto p es accesible por densidad desde un punto q con respecto a $\epsilon$ y $MinPts$ si hay una cadena de puntos $p_1, \ldots, p_n$ con $p_1 = q$ y $p_n = q$ tal que $p_{i+1}$ es directamente accesible por densidad desde $p_i$.*

**Definición 4** *(Conectado por densidad) Un punto p está conectado por densidad a un punto q con respecto a $\epsilon$ y $MinPts$ si hay un punto o tal que ambos, p y q son accesibles por densidad desde o con respecto a $\epsilon$ y $MinPts$.*

De esta forma, al utilizar los conceptos de *conectado por densidad* y *accesible por densidad* , se puede establecer una relación entre todas las posibles interacciones que pueden ocurrir entre los puntos de un cluster (punto borde a punto central, punto central a punto borde, punto central a punto central y punto borde a punto borde) y agruparlos en un conjunto único al que se le denomina cluster. Esto permite tener una definición más precisa y completa de un cluster, considerando la presencia de distintos tipos de puntos y sus interacciones en el conjunto de datos.

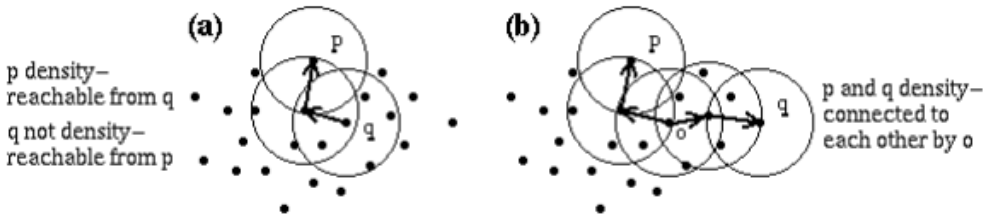



Figura 3.7: Representación accesibilidad por densidad (izquierda) y conectividad por densidad (derecha) [15]

**Definición 5** *(Clúster) Sea $D$ un conjunto de puntos. Un cluster $C$ con respecto a $\epsilon$ y $MinPts$ es un subconjunto no vació de $D$ que satisface las siguientes condiciones:*

1. *$\forall p, q$: si $p \in C$ y $q$ es accesible por densidad desde $p$ con respecto a $\epsilon$ y $MinPts$, entonces $q \in C$. (maximalidad)*

2. *$\forall p, q \in C$: $p$ es conectado por densidad a $q$ con respecto a $\epsilon$ y $MinPts$. (Conectividad)*

**Definición 6** *(ruido) Sean $C_1, \ldots, C_k$ los clústers del conjunto de datos $D$ con respecto a los parámetros $\epsilon_i$ y $MinPts_i$, $i = 1, \ldots, k$. Entonces se define el ruido como el conjunto de puntos en el conjunto de datos $D$ que no pertenece a ningún cluster $C_i$. Eso es, $ruido = \{p \in D | \forall i : p \notin C_i\}$*

Habiendo definido un cluster y el concepto de puntos ruidosos se obtienen los siguientes lemas que serán de utilidad en el entrenamiento del algoritmo:

**Lema 1** *Sea $p$ un punto en $D$ y $|N_\epsilon| \geq MinPts$. Entonces el conjunto $O = \{o \mid o \in D$ y $o$ es accesible por densidad desde $p$ con respecto a $\epsilon$ y $MinPts\}$ es un cluster con respecto a $\epsilon$ y MinPts.*

**Lema 2** *Sea $C$ un clúster con respecto a $\epsilon$ y MinPts y sea $p$ cualquier punto en $C$ que cumple $|N_\epsilon(p)| \geq MinPts$. Entonces $C$ es igual al conjunto $O = \{o \mid o$ es accesible por densidad desde $p$ con respecto a $\epsilon$ y $MinPts\}$*

### 3.5.2. Entrenamiento

El proceso de entrenamiento del algoritmo DBSCAN se lleva a cabo siguiendo una serie de pasos ordenados, los cuales se detallarán a continuación. En primer lugar, se procede a seleccionar un valor para los hiperparámetros del modelo, es decir, el valor de $\epsilon$ (distancia máxima entre puntos para considerarlos vecinos) y MinPts (número mínimo de puntos requeridos para formar un clúster). A continuación, se asigna un identificador al primer clúster a considerar y se inicia el proceso de expansión de clústers.

Para cada punto no clasificado, se lleva a cabo la expansión de un clúster alrededor de dicho punto. En esta etapa, se determinan los vecinos del punto seleccionado y, si se encuentran a una distancia menor a $\epsilon$, se les asigna el mismo identificador del clúster. Por el contrario, si los vecinos están a una distancia mayor a $\epsilon$, se etiquetan como puntos ruidosos, es decir, no pertenecientes a ningún clúster.

Una vez que se ha expandido un clúster, se crea un nuevo identificador y se repite el proceso anteriormente mencionado hasta que todos los puntos hayan sido clasificados, es decir, asignados a un clúster o etiquetados como ruidosos.

### 3.5.3. DBSCAN para detección de anomalías

La detección de anomalías con DBSCAN es directa, se consideran anomalías todos los puntos que sean clasificados como ruidosos, es decir, que no pertenecen a ningún cluster. Para su aplicación a series de tiempo esparsas es necesario realizar un preprocesamiento de los datos, el cual se describe en detalle en el capítulo 5.

## 3.6. Autoencoder

Cuando se hace referencia a modelos que utilizan reducción dimensional y el error de reconstrucción de los datos para detectar anomalías, es inevitable pensar en los autoencoders [51]. Estos modelos, mediante un enfoque de entrenamiento no supervisado, proporcionan una reducción dimensional no lineal y son muy efectivos para la detección de anomalías y generación de datos.

### 3.6.1. Modelo

En términos generales, un autoencoder consta de dos funciones: el *encoder* $E_\theta(x) : X \rightarrow Z$ y el *decoder* $D_\phi(z) : Z \rightarrow X$. El encoder recibe un elemento del conjunto de datos $X = \{x_i \in \mathbb{R}^n,\ i = 1, \ldots, k\}$, donde $n$ es el número de características o *features* y devuelve su código correspondiente $Z = \{z_i \in \mathbb{R}^m,\ i = 1, \ldots, k\}$, donde $m < n$ en el espacio latente, mientras que el decoder intenta reconstruir el dato original del conjunto de datos que se introdujo. En la figura 3.8 se muestra un esquema de la arquitectura del autoencoder.

El modelo del autoencoder se construye concatenando la salida del encoder con la entrada del decoder. El modelo sigue la siguiente expresión matemática:

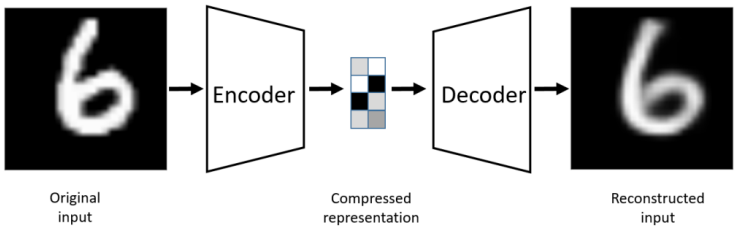



Figura 3.8: Ejemplo de un autoencoder para imágenes. Extraído de [4].

$$x' = D_\phi(E_\theta(x)), \quad x \in X$$

### 3.6.2. Entrenamiento

Es necesario definir una métrica para poder optimizar sus parámetros en la tarea específica de la reconstrucción de los datos de entrada. En este caso, se utilizará la distancia euclídea como medida de desempeño, es decir, la diferencia entre el dato de entrada y la salida del modelo, evaluada a través de la distancia euclídea.

La función de costo a optimizar se define de de la siguiente manera:

$$L(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_\theta(E_\phi(x_i))\|_2^2$$

donde $x_i$ es el dato de entrada, $D_\phi$ es la función de decodificación, $E_\theta$ es la función de codificación, y $\theta$ y $\phi$ son los parámetros del modelo que se optimizan durante el entrenamiento.

### 3.6.3. Autoencoder para detección de anomalías

La clasificación de anomalías se realiza simplemente umbralizando el error de reconstrucción. Los datos cuyo error supere el umbral son considerados anómalos. Este umbral se puede optimizar para maximizar alguna métrica de clasificación (en general, *F1* o *accuracy*) en un conjunto de validación, siempre que se cuente con datos con anomalías etiquetadas.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 4

# Implementación

Dentro de las motivaciones principales para la realización de este proyecto se encuentra el interés de Telefónica de obtener una herramienta capaz de detectar de forma automática anomalías en series de tiempo esparsas en tiempo real. Con este fin, cada modelo se debe implementar respetando una interfaz de software específica que permita su integración rápida a una aplicación existente que ya está siendo utilizada por el cliente en otro tipo de series. La aplicación fue desarrollada por el grupo de Detección de Anomalías del Instituto de Ingeniería Eléctrica y el código fuente se encuentra disponible.

En este capítulo se describe en primer lugar la aplicación disponible y su funcionamiento básico, así como la integración de los detectores desarrollados. Luego se explican brevemente algunos detalles de la implementación de los modelos de detección. Todas las implementaciones, tanto de la aplicación base como de los detectores se realiza en el lenguaje de programación *python*.

## 4.1. Aplicación

La aplicación se encarga de la ejecución de los modelos de detección de anomalías y de la comunicación con la base de datos *InfluxDB* para la extracción e inyección de datos. La misma ya se encuentra corriendo en el entorno de producción de Telefónica. Implementa una API del tipo REST con la cual se pueden manejar los modelos disponibles, entrenarlos y generar detecciones.

Los datos son recolectados y enviados a *InfluxDB* utilizando el agente *Telegraf* [30], parte del ecosistema de *Influx*, que permite la recolección de datos de diversos tipos de sistemas como sensores o bases de datos.

La comunicación entre la aplicación y la base de datos es realizada mediante la API de *InfluxDB* [29]. Cuando las detecciones están listas, son enviadas a la base de datos para su almacenamiento y posterior visualización con la herramienta de monitoreo y analítica *Grafana* [36].

### 4.1.1. Funcionamiento

El funcionamiento de la aplicación tiene como eje central un objeto único (o *singleton*) de la clase `Detectors`, que controla el conjunto de detectores que se utilizaran para la detección de anomalías en tiempo real. Posee métodos que permiten agregar, quitar, entrenar y hacer predicciones con un detector en específico.

Las instancias de los detectores, definidos en la clase `Detector`, están relacionados uno a uno con instancias de la clase `DataFrame`. La clase `DataFrame` hereda de la clase `Thread` que en python representa un hilo del CPU, de modo que cada detector podrá ejecutarse en un hilo diferente haciendo mejor uso de los recursos de hardware disponibles.

De esta forma, al recibir instrucciones, la aplicación se comunica con el objeto instancia de `Detectors`, donde en caso de querer crear un detector nuevo, crea la instancia correspondiente de la clase `DataFrame` y le asocia un objeto de la clase `Detector` según corresponda.

La aplicación corre en segundo plano como servicio web, recibiendo instrucciones en forma de peticiones `http` a diferentes *endpoints* implementados en una API del tipo REST. Estos se comunican con la instancia de `Detectors` que se encarga de manejar los detectores. Los *endpoints* disponibles son los siguientes:

- `/newTS`: Inicializa un nuevo `DataFrame`. Recibe como parámetro un `id` que lo identifica.
- `/setAD`: Define un detector y lo asocia a un `DataFrame` que debe ser creado previamente.
- `/startAD`: Inicializa el detector asociado a un `DataFrame`.
- `/removeAD`: Borra el detector.
- `/listAD`: Muestra todos los detectores activos.
- `/fit`: Entrena un detector.
- `/detect`: Realiza detecciones en un conjunto de datos.

Finalmente las peticiones HTTP son enviadas a la aplicación desde la base de datos InfluxDB mediante la creación de scripts programados en el lenguaje de consultas flux [13]. Estos scripts pueden ser ejecutados de forma manual o programada, y dentro de los mismos se realiza la consulta de los datos a la base, el filtrado y las transformaciones necesarias y el envío de las peticiones HTTP a la aplicación. La aplicación devuelve las detecciones a la base de datos comunicándose con la API de Influx, que se abstrae en un objeto cliente parte del SDK de Influx para *python*.

### 4.1.2. Integración de modelos

Los modelos de detección se pueden integrar de forma sencilla implementándolos como clases y siguiendo las interfaces definidas por la aplicación. Cada detector debe implementar obligatoriamente los siguientes métodos:

- `fit(df)`: Ajusta el modelo a los datos provistos.
- `detect(observations)`: Clasifica los datos provistos como anómalos o no anómalos.

En la figura 4.1 se muestra un diagrama de clases usando un detector basado en modelos de Markov a modo de ejemplo. El namespace `pd` hace referencia a la librería *pandas* [44] que es una herramienta flexible y poderosa para la manipulación y procesamiento de datos y que se usó extensivamente en el desarrollo de los modelos.

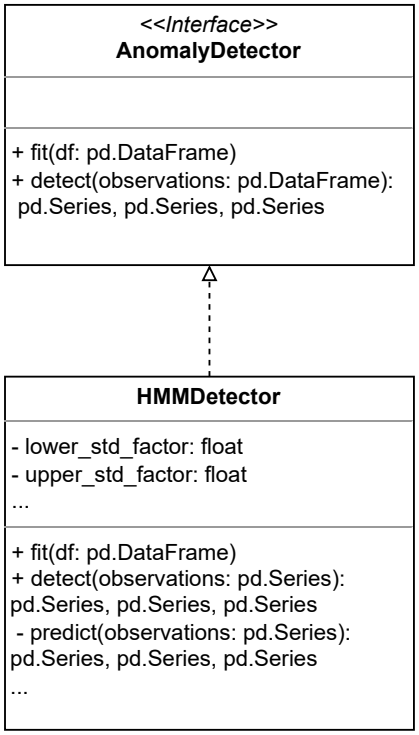



Figura 4.1: Ejemplo de implementación de detector.

## 4.2. Modelos

Para la implementación de los algoritmos de los diferentes modelos se utilizaron librerías de terceros siempre que fuera posible. En los casos donde el uso de librerías no fue posible se implementaron los algoritmos desde cero, utilizando librerías y *frameworks* adecuados pero de más bajo nivel.

En el caso del HMM se utilizó la librería `hmmlearn` [27] que implementa varios tipos de *Hidden Markov Models* con diferentes distribuciones de emisión. Para FM se utilizó una librería `fastFM` [6], que tiene la ventaja de implementar las interfaces de forma idéntica al conocido *framework* para ciencia de datos `scikit-learn` [46], con el que además se implementó el detector basado en DBSCAN. Los modelos de pronóstico están todos implementados en la librería `gluonTS` [2, 3], la cual ofrece un gran repertorio de modelos para *forecasting* de series de tiempo. Finalmente, las redes neuronales de los detectores basados en Autoencoder y LSTM fueron programadas a más bajo nivel utilizando el *framework* `Pytorch` [45]. En el caso del LSTM se utilizó además la librería `blitz` [14] para las capas bayesianas.

## 4.3. Métricas

Existe una implementación no oficial de las métricas para series de tiempo [52, 58]. La misma provee una interfaz sencilla que permite seleccionar fácilmente entre las definiciones en 2.3 y ajustar el parámetro $\alpha$. Si bien tiene disponibles más opciones de personalización, se entiende que con las definiciones básicas es suficiente para el alcance de este proyecto.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 5

# Metodología

El entrenamiento de los modelos es del tipo no supervisado, donde se pretende aprender el comportamiento y patrones normales de la serie, por lo que no se necesitan etiquetas para entrenar. Sin embargo, estas son necesarias en la etapa de evaluación, por lo que se requiere desarrollar algún método para evaluar el funcionamiento de los detectores.

En este capítulo se explica la metodología usada para el entrenamiento y evaluación de los detectores presentados en 3. Se comienza por describir y analizar las características de los datos disponibles y el preprocesamiento involucrado. Después se explica el método de generación e inyección de anomalías en los datos, seguido de los detalles del proceso de entrenamiento y evaluación.

## 5.1. Datos

Los datos que se utilizaron para este proyecto son de carácter privado, por lo que no es posible dejar referencias para su uso ni tampoco brindar descripciones más allá de las que se exponen aquí.

Se cuenta con 14 series provenientes de datos de transacciones de distintas entidades comerciales, cuya duración comprende desde el día 12/9/2022 al 12/10/2022, inclusive. Las muestras no son equiespaciadas y no necesariamente existen datos en todas las series para todos los instantes de tiempo, como puede apreciarse en la figura 5.1.

Con el fin de entender las características de las series temporales de las que se dispone, se llevó a cabo un análisis de su esparsidad, intervalo de interdemanda medio y coeficiente de variación. Para calcular dichas propiedades se periodiza cada una de forma independiente realizando agregaciones de sus valores (en general, suma o promedio) en intervalos regulares.

En principio se utilizan intervalos de cinco minutos y la suma como función de agregación para esta etapa, pudiendo utilizar una configuración distinta en etapas posteriores si fuera necesario. La frecuencia para el muestreo de las series se eligió considerando las características de los datos originales y los requerimientos en cuanto a velocidades de detección.

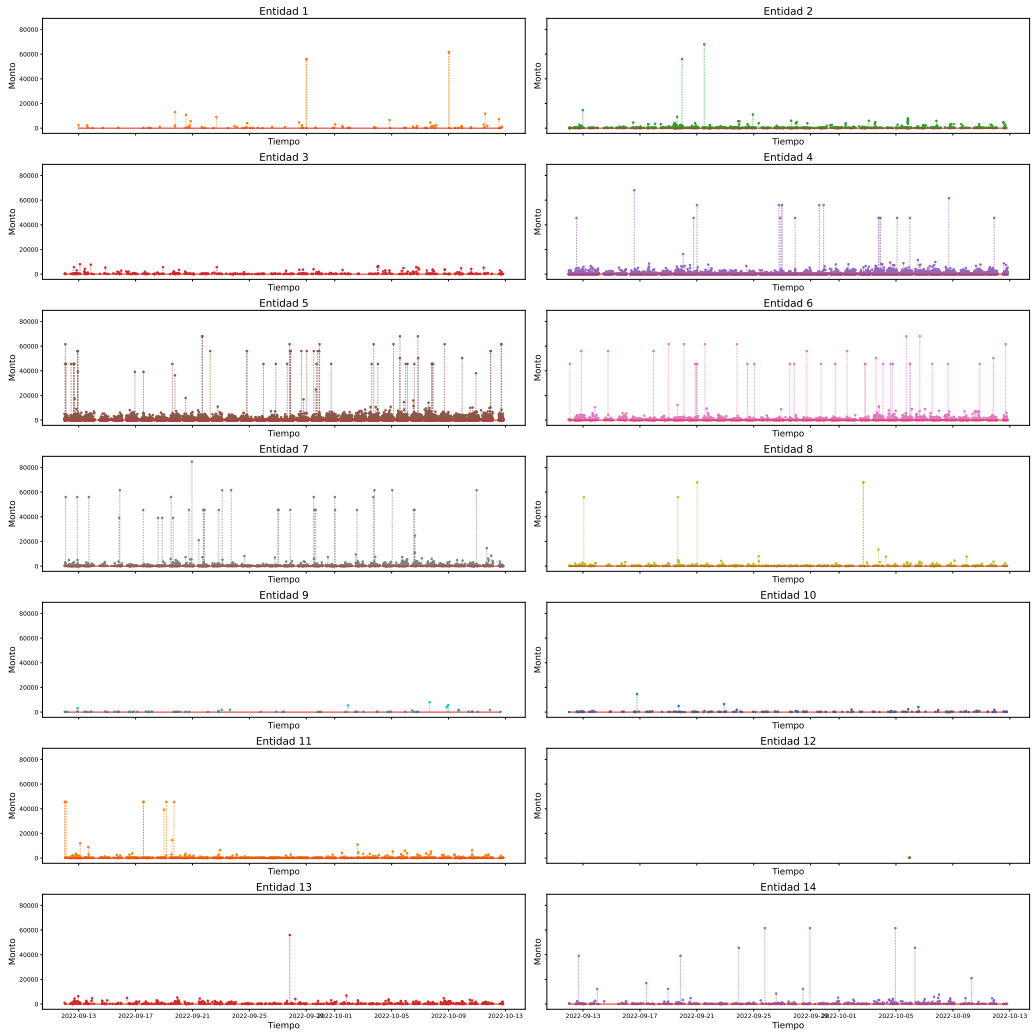



Figura 5.1: Datos originales.

| Entidad | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Esparsidad ( %) | 98.93 | 89.00 | 95.14 | 25.97 | 26.00 | 67.46 | 77.87 |
| Entidad | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Esparsidad ( %) | 93.53 | 99.19 | 97.84 | 88.75 | 99.97 | 93.64 | 93.73 |

Tabla 5.1: Esparsidad por serie.

En la tabla 5.1 se muestra el grado de esparsidad global de cada serie periodizada. Se puede ver que, a menos de las series de las entidades 4 y 5, todas las series tienen esparcidad $> 65\,\%$. Las series de las entidades número 1, 9, 10 y 12 poseen más de $99\,\%$ de datos nulos, lo cual hace prácticamente imposible el entrenamiento de modelos y se descartan del análisis.

Los tiempos de interdemanda medios y el coeficiente de variación en valores

| Entidad | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $ADI$ | 91.17 | 9.07 | 20.60 | 1.35 | 1.35 | 3.07 | 4.52 |
| Entidad | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| $ADI$ | 15.45 | 124.25 | 46.39 | 8.88 | 4.50 | 15.74 | 15.95 |

Tabla 5.2: Intervalo interdemanda medio por serie.

| Entidad | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $CV^2$ | 8.79 | 21.59 | 3.34 | 7.82 | 5.89 | 17.80 | 21.23 |
| Entidad | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| $CV^2$ | 43.53 | 3.13 | 7.79 | 22.78 | 0.01 | 11.62 | 20.63 |

Tabla 5.3: Coeficiente de variación por serie.

no nulos de cada serie se muestran en las tablas 5.2 y 5.3, respectivamente, y reportados en intervalos de 5 minutos. De acuerdo al esquema de clasificación descrito en la sección 2.2.2, todas las series son del tipo desigual (o *lumpy*). Este tipo de series han demostrado ser las más difíciles de pronosticar [33].

## 5.2. Preprocesamiento

Además del proceso de periodización ya descrito, algunos algoritmos necesitan un preprocesamiento especial de los datos de entrada. En particular, para los métodos que no se basan en pronóstico cada serie se proyecta a un espacio bidimensional donde una de las dimensiones es el intervalo interdemanda y la otra es el valor de demanda propiamente dicho. Esta metodología está inspirada en investigaciones previas en el contexto del pronóstico en series de demanda intermitente [10, 33, 57, 59].

### 5.2.1. Serie de demanda

Para el caso de series esparsas ideales (sin presencia de ruido) la serie de demanda puede definirse de la siguiente forma:

**Definición 7** *(serie de demanda ideal): Dada una serie de tiempo esparsa definida como una secuencia $X(n)$, donde $n$ varía en el intervalo discreto $1, 2, \ldots, N$; se define la serie de demanda ideal $D$ como la serie que cumple $D = X(k)$, con $k \in \{n | X(n) > 0\}$.*

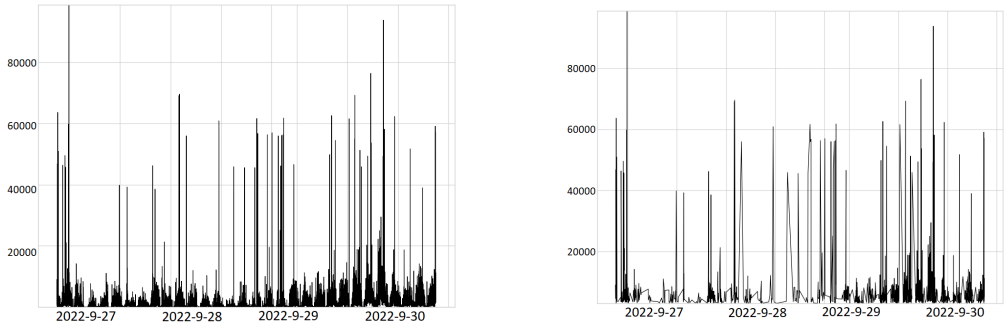
Si bien la definición anterior de serie de demanda es válida al trabajar con series de tiempo esparsas en condiciones ideales, puede presentar dificultades al tratar con series de tiempo que contienen ruido. Esto se debe a que los valores originalmente nulos de la serie pueden ser alterados por el ruido agregado. Por lo tanto, es necesario realizar una modificación en la definición de la serie de demanda, incorporando un umbral a partir del cual los datos se consideran no nulos.

Eligiendo el parámetro $m$ para establecer este umbral, la serie de demanda real, a la que se llamará simplemente serie de demanda puede definirse de la siguiente forma:

**Definición 8** *(serie de demanda): Dada una serie de tiempo esparsa definida como una secuencia $X(n)$, donde $n$ varía en el intervalo discreto $1, 2, \ldots, N$ y $m$ un numero real; se define la serie de demanda $D$ como la serie que cumple $D = X(k)$, con $k \in \{n | X(n) > m\}$.*

Esta nueva definición permite adaptarse adecuadamente a las series de tiempo con presencia de ruido, al establecer un criterio claro para determinar qué valores se consideran no nulos en la serie de demanda.

En la figura 5.2 se muestra un ejemplo de construcción de la serie de demanda. Notar que por definición, la serie de demanda de una serie esparsa, no es esparsa, ya que no posee valores nulos. Además, se pierde algo de información temporal debido a que las muestras no están necesariamente equiespaciadas, aunque mantienen el orden original.




(a) Serie original

(b) Serie de demanda

Figura 5.2: Extracción de serie de demanda.

## 5.2.2. Serie de inactividad

Con el objetivo de codificar los períodos de inactividad en las series de tiempo esparsas y proporcionar información relevante para los modelos, se llevó a cabo un preprocesamiento para obtener una serie auxiliar conocida como serie de inactividad. Al igual que en el caso de la serie de demanda, se parte de una definición útil para series de tiempo esparsas ideales para luego adaptarla a series de tiempo con ruido.

**Definición 9** *(serie de inactividad ideal) : Dada una serie de tiempo esparsa, definida como una secuencia $X(n)$, donde $n$ varía en el intervalo discreto $1, 2, \ldots, N$ y sea $Y$ una serie formada por los índices de $X$ que pertenecen al conjunto $\{k \mid X(k) > 0\}$ , la serie de inactividad ideal $I$ se define como $I(l) = Y(l) - Y(l-1)$ para todo $l$ perteneciente al dominio de $Y$.*

La serie de inactividad entonces, asocia a cada intervalo de inactividad un punto con una magnitud proporcional a su duración. Sin embargo, al trabajar con series ruidosas los períodos de inactividad pueden verse alterados y presentar pequeñas perturbaciones a lo largo de su duración. Con la definición anterior, estas perturbaciones son vistas como períodos de inactividad independientes volviendo poco efectivo el preprocesamiento. Por esta razón, se modifica la definición de serie de inactividad agregando un umbral en forma de parámetro al igual que en el caso de las series de demanda.

**Definición 10** *(serie de inactividad) : Dada una serie de tiempo esparsa definida como una secuencia $X(n)$, donde $n$ varía en el intervalo discreto $1, 2, \ldots, N$; sea $m$ un número real y sea $Y$ una serie formada por los índices de $X$ que pertenecen al conjunto $\{k \mid X(k) > m\}$, la serie de inactividad $I$ se define como $I(l) = Y(l) - Y(l-1)$ para todo $l$ perteneciente al dominio de $Y$*

Eligiendo el parámetro $m$ de forma adecuada las series de inactividad no se ven afectadas por las pequeñas perturbaciones del ruido permitiendo obtener modelos más robustos. Un ejemplo de la serie de inactividad de una serie se muestra en la figura 5.3.

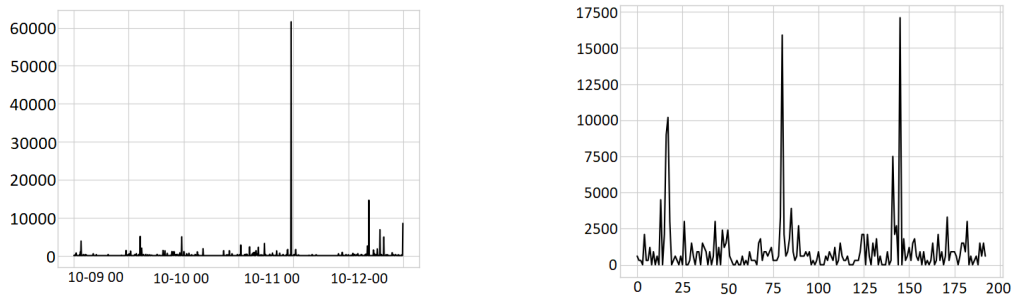



Figura 5.3: Extracción de serie de inactividad.

## 5.3. Simulación de anomalías

Se asume que los datos con los que trabajaremos son en su totalidad normales, es decir, no contienen anomalías. Si bien esta hipótesis puede ser cuestionable, en una inspección general no se encontraron anomalías evidentes. Hay que tener en cuenta el poco conocimiento del dominio que se tiene, pero creemos que si efectivamente existieran anomalías en los datos, la cantidad sería mínima y la hipótesis seguiría siendo válida.

La hipótesis anterior implica que necesariamente se deberán generar anomalías sintéticas si se quiere contar con métricas de rendimiento de los algoritmos de detección implementados. La generación no es estrictamente necesaria en el conjunto

de entrenamiento ya que en todos los modelos planteados se usa aprendizaje no supervisado, aunque igualmente puede ser útil para comparar el rendimiento de los detectores en distintos escenarios.

La generación de anomalías sintéticas fiables y realistas es un problema difícil, ya que depende en gran medida del conocimiento que se tenga del problema particular. La mayoría de los métodos se centran en la generación de series completas con datos anómalos usando VAEs [37, 60] que en pruebas preliminares con los datos disponibles no dieron buenos resultados. Debido a esto y al espacio acotado de tiempo que se tiene para esta parte del proyecto se adopta un enfoque más simple, donde se modifican los valores de la serie original en muestras seleccionadas aleatoriamente (a veces llamado inyección de anomalías [7]). Los valores de las muestras anómalas inyectadas se asignan según reglas que dependen de estadísticas extraídas de los datos normales.

Principalmente se quieren detectar tres tipos de anomalías, sugeridos por el personal de la empresa Telefónica. No se dispuso de descripciones detalladas de cada uno de los casos, sino que se obtuvieron explicaciones generales de los eventos que las producen y de su manifestación en los datos. Estos tipos son los siguientes:

- *Anomalías de demanda*: puntuales con valores extremos atípicos.
- *Anomalías de inactividad*: de rango con valores nulos o muy bajos durante períodos de tiempo anormalmente extensos.
- *Anomalías de actividad*: de rango y contextuales, cuyos valores presentan una tendencia marcadamente creciente.

El número de anomalías generadas debe ser tal que la densidad de anomalías sea verosímil, es decir la cantidad de muestras anómalas debe ser mucho menor a la cantidad total de datos. Por otro lado la cantidad de anomalías debe ser suficiente para que exista variabilidad y para que las métricas obtenidas sean representativas del verdadero rendimiento de los detectores. Además, en cada ventana o contexto de predicción debería existir una, y solo una, anomalía [63].

Se muestra un ejemplo de serie con anomalías simuladas en la figura 5.4.

### 5.3.1. Simulación de anomalías de demanda

La simulación de este tipo de anomalías puede justificarse en la necesidad del cliente de detectar cuando de forma repentina se presenta un gran incremento en la magnitud de la serie, lo que puede significar un movimiento fraudulento o una sobrecarga en algún sistema. El lograr detectar este tipo de anomalías es de vital importancia para proteger los recursos de los usuarios y amortiguar posibles pérdidas económicas.

Las anomalías en la demanda se generan de manera aleatoria y puntual. Se asume que se pueden dar en cualquier momento con igual probabilidad, por lo que se utiliza una distribución uniforme para muestrear puntos en todo el intervalo de tiempo. En cuanto a los posibles valores que toman, si se trabaja con series de tiempo ideales, se entiende que están acotadas inferiormente por el máximo de la

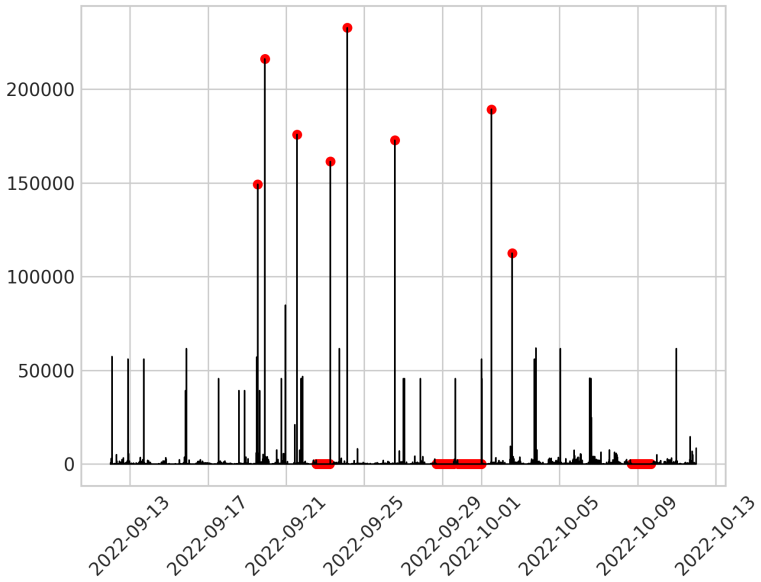



Figura 5.4: Serie con anomalías de demanda e inactividad.

serie. En la realidad se asume que se trabaja con series ruidosas, por lo tanto en la generación se deja un margen al límite inferior de un 20 % y por temas de facilidad a la hora de la visualización y análisis se acotan superiormente por un 300 % del valor del máximo.

El rango de valores y las distribuciones elegidas son discutibles, ya que es posible que existan anomalías de demanda que globalmente son menores al máximo, pero que localmente tienen valores demasiado altos. Sin embargo, las series provistas presentan valores normales extremadamente altos en tiempos aparentemente aleatorios cuya cantidad de muestras (del orden de las decenas) no permite extraer estadísticas fiables que justifiquen una elección de distribución y rango distinta.

En la figura 5.5 se muestra un ejemplo de serie con anomalías de demanda. Para el entrenamiento y evaluación se generarán 0,5 % del total de muestras de anomalías de demanda, teniendo especial cuidado en que no se superpongan más de una en el contexto de predicción utilizado más largo, que es igual a 24 muestras cuando los intervalos son de 5 minutos. Con esta configuración se tienen $0,005 \times 8885 \approx 44$ anomalías de demanda en total, por serie y para todo el largo del intervalo.

### 5.3.2. Simulación de anomalías de inactividad

Las anomalías de inactividad se dan en situaciones en las cuales una serie experimenta un período de inactividad más prolongado de lo normal, lo cual puede deberse, por ejemplo, a la interrupción de un servicio proporcionado por terceros o a fallas en un sistema interno. Identificar este comportamiento desde la perspectiva del cliente es crucial para poder ofrecer una mejor experiencia de usuario.

En este caso, la anomalía es generada asignando valores nulos a puntos consecu-

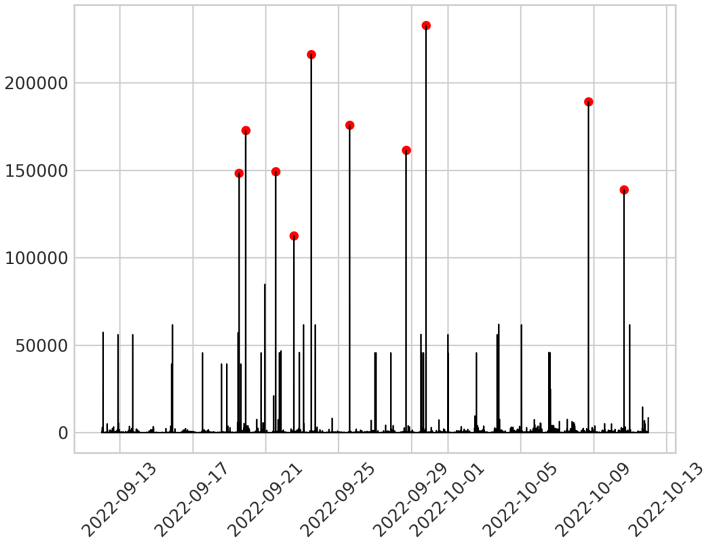



Figura 5.5: Serie con anomalías de demanda.

tivos de la serie. Como los problemas que generarían estas anomalías se pueden dar en cualquier momento y a falta de mayor información, la posición de la anomalía es determinada aleatoriamente siguiendo una distribución uniforme. La longitud del período de inactividad sigue el mismo razonamiento que en el caso de las anomalías de demanda. Se deja un margen para el límite inferior de la duración del período de un 20 % del máximo período de inactividad en la serie y se acota superiormente por el 250 %, un poco menor que en el caso de las anomalías de demanda ya que para las anomalías de inactividad una magnitud mayor equivale a modificar una mayor cantidad de puntos de la serie. De igual forma, al hacer esto no se pierde generalización, ya que de aparecer un período de inactividad más prolongado también será detectado como anomalía.

Al comprender rangos relativamente grandes de muestras consecutivas, la cantidad de anomalías de este tipo debe ser mucho menor que las de demanda, como puede apreciarse en el ejemplo de la figura 5.6. Se generan en total 10 anomalías de inactividad en total.

### 5.3.3. Simulación de anomalías de actividad

La necesidad de generación de este tipo de anomalías reside en representar escenarios en los cuales, a causa de un error o una circunstancia sospechosa, se produce un incremento considerable en la frecuencia de actividad de una serie que, por lo general, presentaría una actividad significativamente más baja. Un ejemplo sería la aparición repentina de un elevado volumen de llamadas internacionales hacia destinos poco habituales. Identificar este tipo de anomalías es importante para encontrar posibles vulnerabilidades en sistemas que puedan estar siendo explotadas.

Las anomalías de actividad se caracterizan por un aumento significativo de la

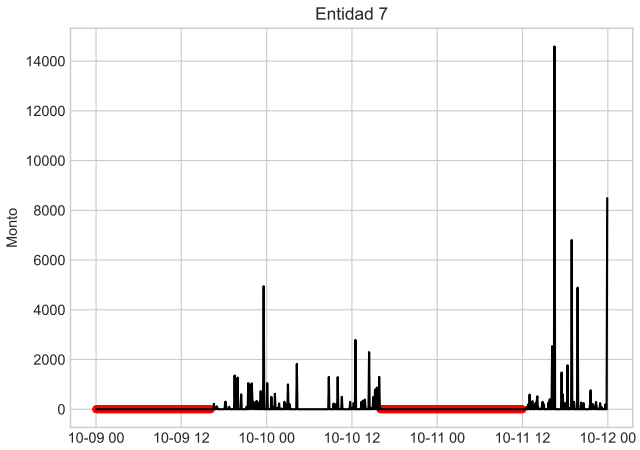



Figura 5.6: Serie con anomalías de inactividad.

actividad con múltiples datos consecutivos no nulos. Durante este período, también se puede observar un incremento en la magnitud de los datos a medida que aumenta el tiempo. Al igual que en los casos anteriores una anomalía de actividad puede suceder en cualquier momento, por lo cual su posición sera determinada en forma aleatoria siguiendo una distribución uniforme. El valor inicial de dicha anomalía quedará determinada por el valor del punto que sea elegido por esta distribución.

Respecto a la duración del período de actividad, no se dispone de información al respecto, pero se asumirá de entre el 0.5 % y el 1.5 % del largo total de la serie. Esto es para no modificar demasiado la serie y poder simular varias anomalías en una misma serie en lugar de una sola de gran tamaño. Además, en cuanto a la evolución en el tiempo de las mismas se eligió un crecimiento que sigue una progresión geométrica ya que este se aproxima mejor que un crecimiento lineal a las situaciones que en principio generan este tipo de anomalía. Finalmente para simular una anomalía más realista, a la progresión geométrica se le suma en cada instante de tiempo ruido de distribución uniforme de entre el 90 % y 110 % del valor de la serie en ese instante, estos límites fueron seleccionados de forma empírica para que el ruido modifique los datos pero no pueda llegar a alterar en gran medida la tendencia marcada por la progresión geométrica. Un ejemplo de varias anomalías de actividad sobre una serie puede observarse en 5.7.

### 5.3.4. Limitaciones

El uso de anomalías sintéticas, aún en los casos sencillos que se plantearon, impone limitaciones claras a la hora de interpretar los resultados. Evidentemente, conociendo exactamente el método de generación de las anomalías es posible diseñar fácilmente un detector sencillo casi perfecto. Por ejemplo, para las anomalías de actividad se puede definir un detector que clasifique como anomalías todos los valores por encima del valor máximo de la serie de entrenamiento. Sin embargo,

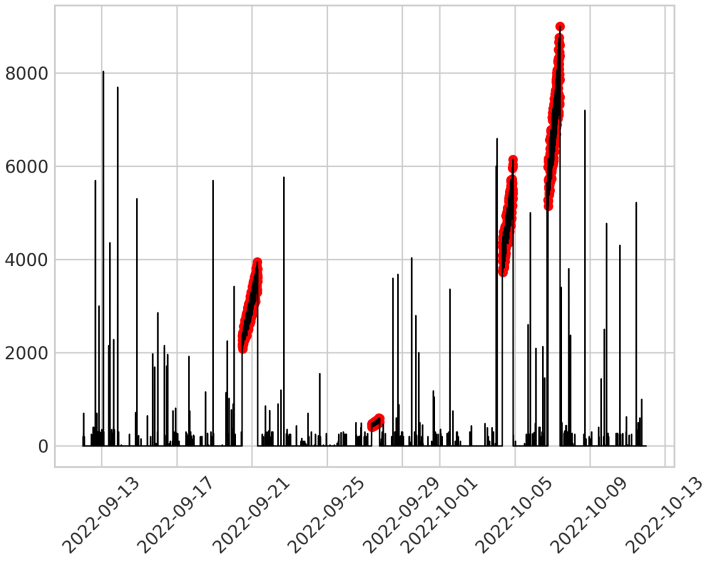



Figura 5.7: Serie con anomalías de actividad.

esto no implica que en la realidad dicho detector obtenga un mejor desempeño que los detectores implementados. De hecho, si se dejara de lado la hipótesis de que los datos no contienen anomalías, algo completamente factible, el detector sencillo vería su desempeño afectado en gran medida. Otro factor a considerar es la calidad de la detección en las anomalías de rango, las cuales se puntúan mejor cuando la detección es temprana. Se espera que los modelos implementados sean más robustos y provean algún incremento en el desempeño cuando se comparan con detectores sencillos.

Al no disponer de datos etiquetados con anomalías reales es difícil saber con seguridad si los resultados obtenidos con las anomalías simuladas son representativos de la realidad. No obstante debido a que se contó con descripciones de alto nivel de los tipos de anomalías más frecuentes y las precauciones tomadas en la simulación, se espera que los resultados sean en alguna medida transferibles a escenarios realistas. De todas maneras, los puntos anteriores deben tenerse presentes al interpretar los resultados de la sección 6.

## 5.4. Entrenamiento

En todos los casos se reserva el 40 % final de los datos para test, quedando del 60 % restante 20 % para validación y 40 % para entrenamiento. La elección del tamaño del conjunto de test que, en principio, puede parecer exagerada, se justifica por las condiciones que se deben cumplir en la generación sintética, como se describe en la sección 5.3 de simulación de anomalías.

En caso de requerirse, la normalización será del tipo mínimo-máximo. Este tipo de normalización se caracteriza por ajustar los valores de la serie a un rango comprendido en el intervalo $[0, 1]$ de forma lineal, transformando el máximo en 1

y el mínimo en 0. La normalización se efectúa de forma independiente para cada serie.

Todos los modelos se entrenan usando datos originales, es decir, antes de generar anomalías. Los conjuntos de test y validación con anomalías sintéticas se reservan para optimizar hiperparámetros y para la evaluación final.

## 5.5. Ajuste de hiperparámetros

Dada la naturaleza única de cada serie y la diversidad en los tipos de anomalías que se consideran, es poco probable que los modelos implementados funcionen de forma óptima con los mismos hiperparámetros. Por este motivo, se desarrolló una metodología para optimizar los hiperparámetros de los diferentes modelos para cada serie y tipo de anomalía en particular.

Existen diversas técnicas para optimizar hiperparámetros en modelos de aprendizaje automático, como por ejemplo *Grid search* o *Random search* [38]. Sin embargo, dada la cantidad de modelos, series y tipos de anomalías a analizar, se optó por utilizar el método de optimización bayesiana, la cual utiliza un enfoque más inteligente que los métodos anteriormente mencionados logrando un compromiso entre resultados y costo computacional más apropiado [42].

La optimización bayesiana es un método iterativo utilizado para encontrar puntos máximos en funciones. Se debe definir una función objetivo única la cual se busca optimizar. El proceso de optimización consiste en crear un modelo probabilístico que representa la función objetivo y que se actualiza iterativamente a medida que se obtienen nuevos datos. En cada iteración, se selecciona una nueva ubicación para evaluar la función siguiendo una estrategia de adquisición, que busca un equilibrio entre explorar áreas desconocidas y explotar áreas prometedoras.

Una vez que se evalúa la función objetivo en la nueva ubicación, los datos se utilizan para actualizar el modelo probabilístico, lo que permite ajustar las predicciones y reducir la incertidumbre sobre la ubicación del máximo. Este proceso se repite tantas veces como sea necesario hasta llegar a un máximo de iteraciones o a un valor de la función objetivo deseado.

Aplicando la optimización bayesiana a la búsqueda de hiperparámetros, la función objetivo es aquella que utilizando los hiperparámetros muestreados como argumentos, devuelve una métrica de interés a optimizar. En este caso se utilizó la métrica *F1* calculada con el *precision* y *recall* extendidos como se definieron en 2.3.

Para implementar la optimización bayesiana se utilizó el *framework* optuna disponible en python [1], el cual dispone de varios métodos para la optimización de hiperparámetros. Los hiperparámetros de los modelos fueron optimizados utilizando las particiones descritas en 5.4 y maximizando la métrica elegida en el conjunto de validación.

## 5.6. Evaluación

Se reportan métricas tanto en el conjunto de entrenamiento como en el conjunto de test sobre todas las series seleccionadas para el análisis (ver 5.1) y para cada tipo de anomalía. La inyección de anomalías se aplica una sola vez a cada una y se analiza la variabilidad de los resultados en las distintas series.

Se realiza también un análisis general del rendimiento de los detectores usando un conjunto único de hiperparámetros para cada modelo en la sección 6.4.

Se evaluará el rendimiento utilizando las métricas para series de tiempo descritas en la sección 2.3 con $\alpha = 0{,}7$ y *sesgo frontal*, que significa que el 70 % del puntaje se debe a la existencia de la detección, y *sesgo frontal* que pondera positivamente las detecciones tempranas en anomalías de rango.

# Capítulo 6

# Resultados

En el este capítulo se exponen los resultados correspondientes a la evaluación de los modelos mencionados en el capítulo 3. El análisis es independiente para cada tipo específico de anomalía para obtener una visión detallada del desempeño de los modelos en cada caso. Cabe mencionar que algunos modelos no resultan adecuados para determinados tipos de anomalías, por lo que serán excluidos del análisis correspondiente. Se presentan los resultados de la evaluación de todas las series según se describió en 5.

Siguiendo el análisis por tipo de anomalía se hace un análisis general, donde se comparan las métricas promedio en todas las series para todos los modelos y se discuten los posibles factores que pueden haber influido en el rendimiento de los detectores.

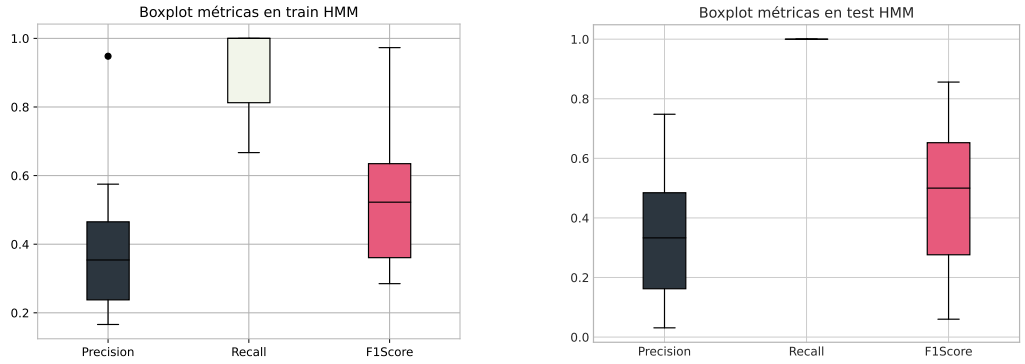
Por último, se discute la robustez de los modelos. Se analizará su capacidad para mantener un rendimiento aceptable incluso en escenarios fuera de lo esperado.

Los hiperparámetros de todos los modelos analizados en esta sección fueron ajustados utilizando optimización bayesiana para cada serie y cada tipo de anomalía como se explica en el capítulo de metodología 5.5.

## 6.1. Anomalías de inactividad

Los modelos seleccionados para detectar las anomalías de inactividad fueron HMM y DBSCAN. Ambos modelos hacen uso de la descomposición en serie de demanda y de inactividad, lo cual, en principio, favorecería la detección de este tipo de anomalías.

En la figura 6.1 se presentan los boxplots de los resultados del modelo HMM en las distintas series. Se puede apreciar que los modelos de Markov muestran un desempeño general decente, teniendo una varianza muy pequeña de *recall* y una mediana de *F1* en el conjunto de prueba cercano a 0,6. La poca variabilidad de *recall* es esperable, dada la baja cantidad de anomalías en el conjunto de prueba. La alta variabilidad de *precision* parece indicar que la selección de hiperparámetros de este modelo es bastante sensible a las características de la serie. Esto puede observarse en las detecciones de la figura 6.2, donde se evidencia el alto *recall* del




(a) (b)

Figura 6.1: Boxplots de resultados de HMM sobre todas las series con anomalías de inactividad simuladas.

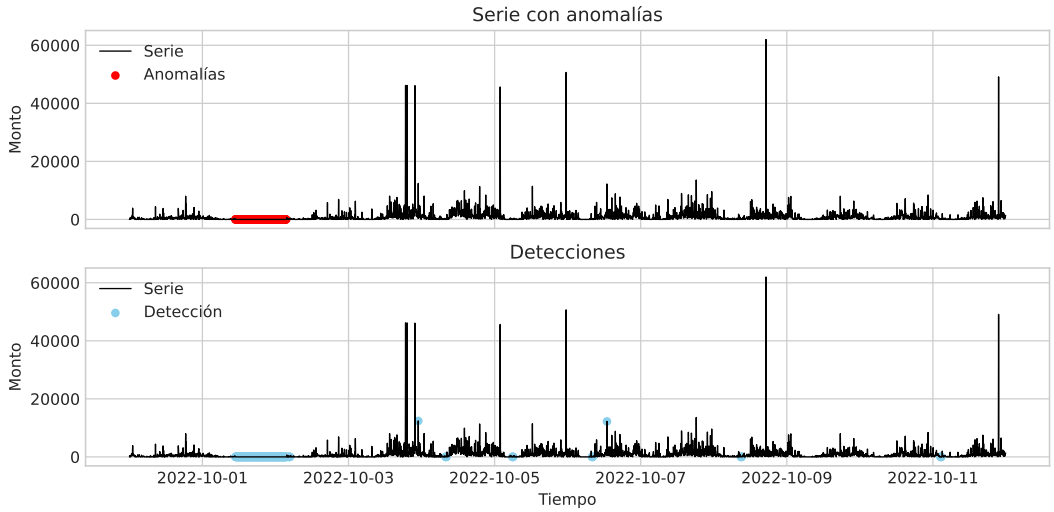
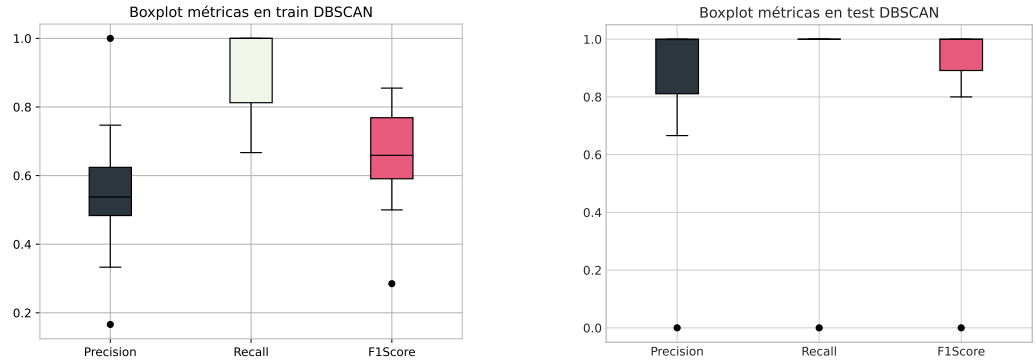



Figura 6.2: Resultados de la detección de anomalías de inactividad con HMM en la serie numero 4.

modelo.

Los resultados obtenidos usando el modelo DBSCAN se muestran en 6.3. Se puede notar que los resultados de este método son mucho mejores a los obtenidos con HMM. Más aún, DBSCAN obtuvo una *precision* muy superior a la del HMM, cubriendo así el principal defecto del mismo. Esto queda claro al analizar la figura 6.4, donde se ve que la tasa de falsos positivos de DBSCAN en la serie 4 es mucho menor que la de HMM.




(a) (b)

Figura 6.3: Boxplots de resultados de DBSCAN sobre todas las series con anomalías de inactividad simuladas.

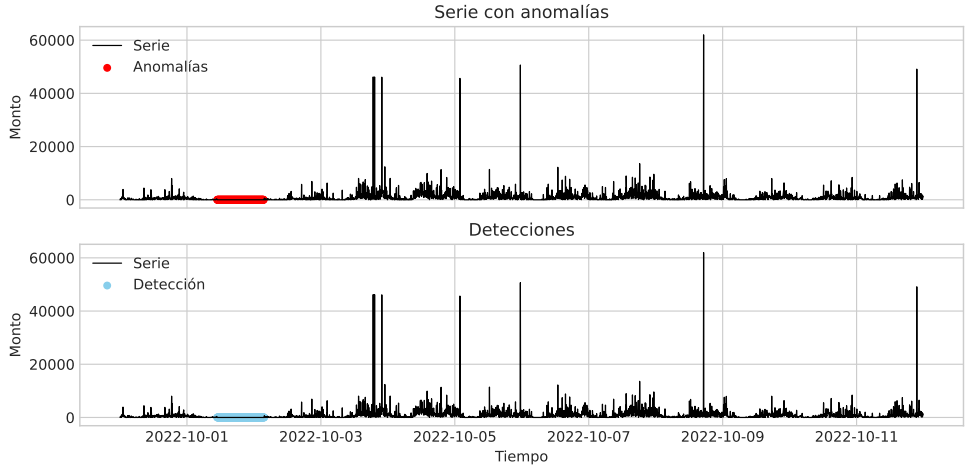
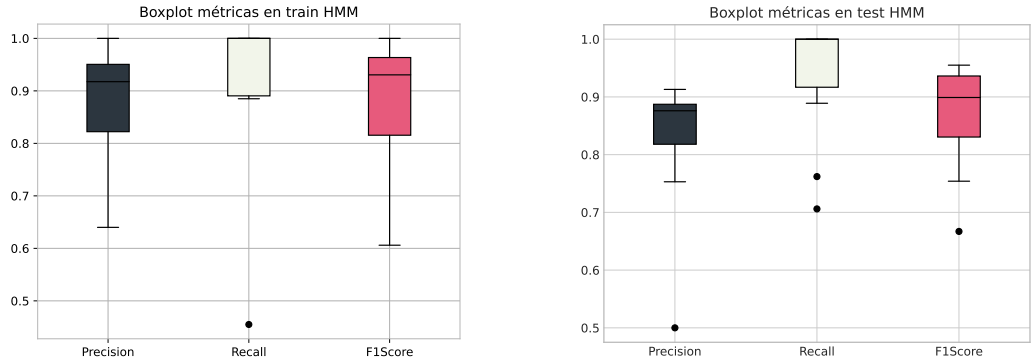



Figura 6.4: Detecciones en test de DBSCAN para anomalías de inactividad en la serie número 4.

## 6.2. Anomalías de demanda

Al evaluar anomalías de demanda, además de los modelos HMM y DBSCAN se incluyen en el análisis los modelos DeepAR y FM. Al ser estos últimos del tipo auto-regresivo, se espera que tengan un desempeño razonable en la detección de este tipo de anomalías ya que estas son básicamente valores extremos muy alejados en magnitud a los valores inmediatos pasados. Además, el modelo DeepAR, en particular, ofrece un enfoque de predicción probabilístico, lo que facilita la interpretación de los resultados.

Comenzando con los modelos utilizados también en la detección de anomalías de inactividad, tanto el HMM como el DBSCAN muestran resultados satisfactorios, como se evidencia en sus respectivos boxplots en las figuras 6.5 y 6.7. HMM, de




(a) (b)

Figura 6.5: Boxplots de resultados de HMM sobre todas las series con anomalías de demanda simuladas.

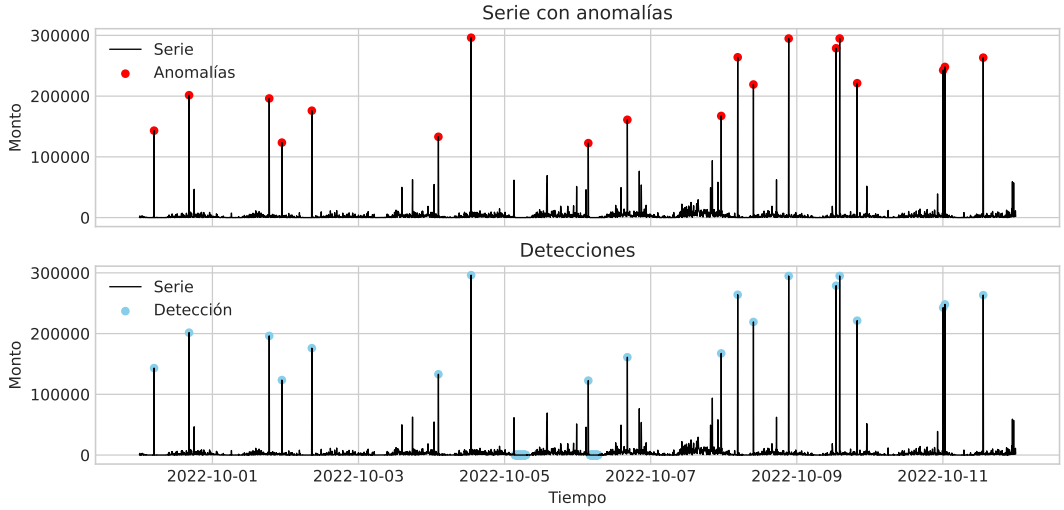
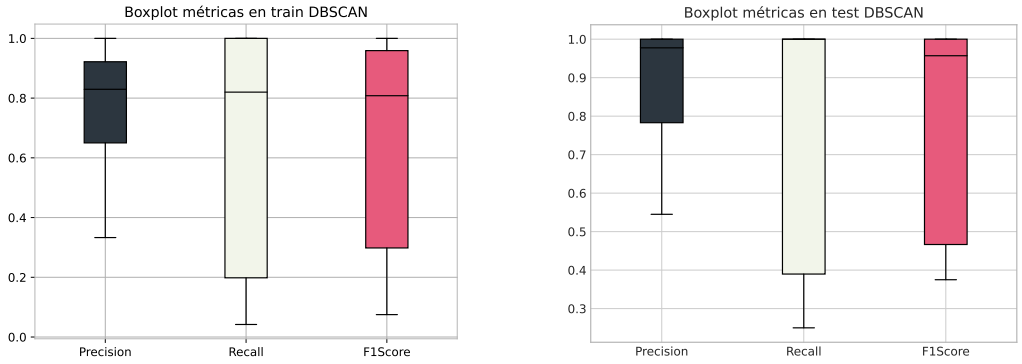



Figura 6.6: Resultados de HMM en la detección de anomalías de demanda en la serie número 5.

igual forma que con las anomalías de inactividad, al utilizar la serie de demanda puede aprender estados sobre los datos del período activo de la serie, sin sesgar sus resultados por los períodos de inactividad. Por otro lado DBSCAN podrá agrupar en clusters los períodos activos de la serie e identificar los datos de una magnitud considerable como anómalos.

Sin embargo, se observa una reducción considerable en la varianza de las métricas en el modelo HMM en comparación con las anomalías de inactividad, sobre todo en la *precision*, como se observa en 6.6, lo que evidencia que el modelo funciona de forma similar en varias series. Por otro lado, si bien DBSCAN exhibe una mediana de *F1* muy alta, su varianza es demasiado alta, lo que indica que su desempeño no es consistente en las diferentes series aunque muy bueno en algunas,




(a) (b)

Figura 6.7: Boxplots de resultados de DBSCAN sobre todas las series con anomalías de demanda simuladas.

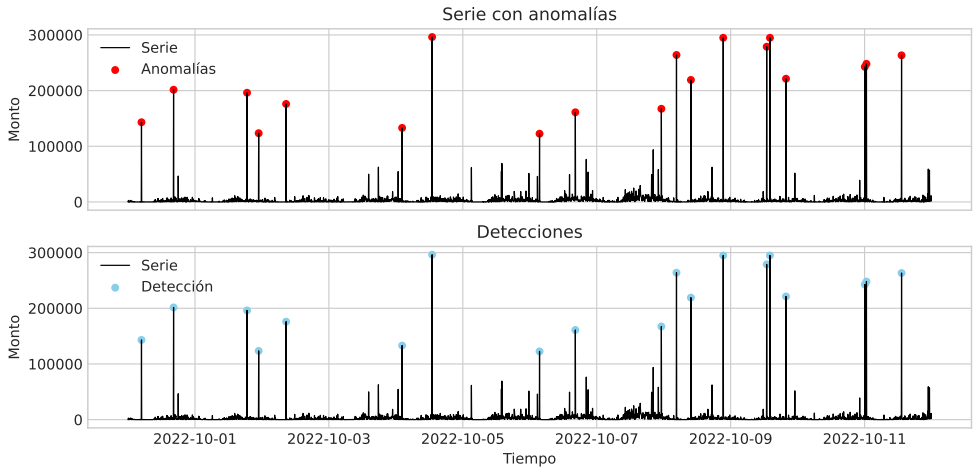
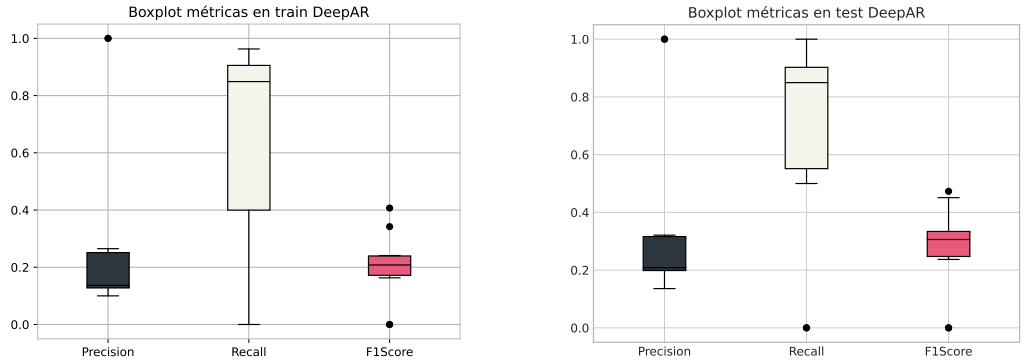



Figura 6.8: Resultados de DBSCAN en la detección de anomalías de demanda en la serie número 5.

como se muestra en 6.8.

En contraste con DBSCAN y HMM, DeepAR muestra una menor varianza en sus métricas de *precision* y *F1*, pero su desempeño en comparación con los modelos anteriores es muy inferior como se ve en su boxplot 6.9. En particular, la *precision* baja indica una alta tasa de falsos positivos y más aún, la gran varianza de su *recall* muestra que el modelo no es capaz de identificar gran parte de las anomalías en varias series, teniendo un desempeño general regular. Sin embargo, donde DeepAR destaca es en la visualización de las detecciones. Al ser un modelo probabilístico, proporciona un umbral para sus predicciones basado en cuantiles, lo cual resulta muy útil a la hora de interpretar qué está aprendiendo el modelo. Esto se puede apreciar en la figura 6.10, donde se muestra el umbral usado en la detección.

Finalmente, es esperable que el modelo basado en *Factorization Machines*




(a) (b)

Figura 6.9: Boxplots de resultados de DeepAR sobre todas las series con anomalías de demanda simuladas.

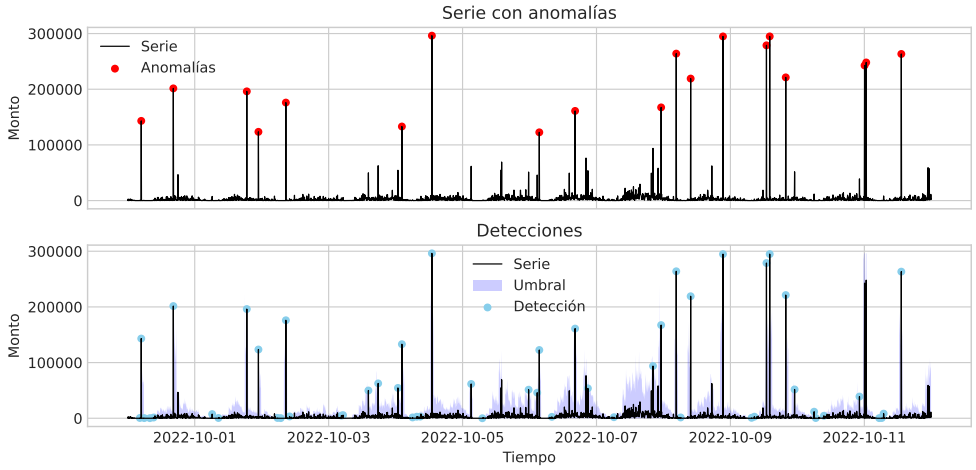
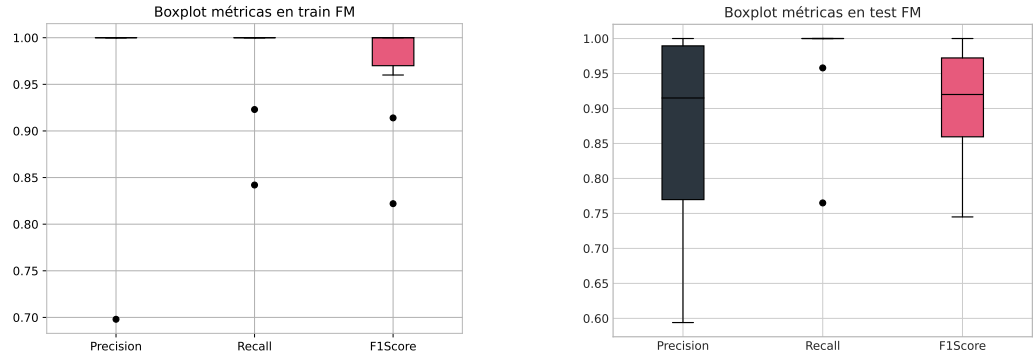



Figura 6.10: Resultados de DeepAR en la detección de anomalías de demanda en la serie número 5.

(FM), también auto-regresivo, arroje buenos resultados en la detección de demandas muy altas, ya que la predicción de estas demandas seguirá una tendencia que se verá poco afectada por valores anómalos de las mismas. Además, son modelos adecuados para aprender datos en condiciones de esparsidad debido a la factorización de los datos de entrada. En este proceso se obtienen vectores que resumen información característica de las *features* de los datos y permiten tener en cuenta el peso de la interacción entre *features* en el aprendizaje mediante el producto escalar de sus correspondientes vectores de factorización [50, 64]. Debido a esto es probable que el modelo basado en FM pueda captar el comportamiento normal de la serie.

En cuanto a los resultados, se puede observar en la figura 6.11 que se obtuvieron los mejores desempeños en el conjunto de prueba, logrando un de *F1* mayor a 0,9




(a) (b)

Figura 6.11: Boxplots de resultados de FM sobre todas las series con anomalías de demanda simuladas.

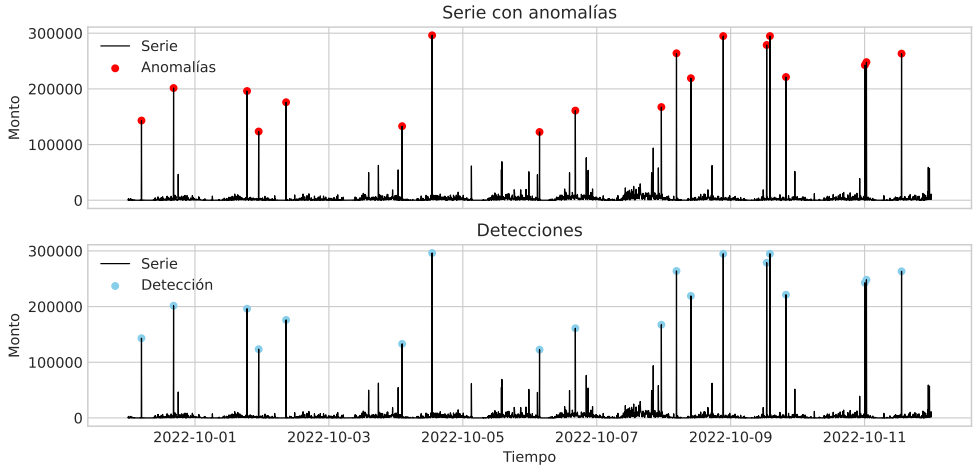



Figura 6.12: Resultados de FM en la detección de anomalías de demanda en la serie número 5.

en la mayoría de las series. Se muestra un ejemplo de las detecciones en la figura 6.12.

## 6.3. Anomalías de actividad

Para finalizar, se procederá a evaluar el desempeño en la detección de anomalías de actividad. Estas anomalías presentan características distintivas en comparación con las analizadas en secciones anteriores, ya que son de naturaleza colectiva y además no estacionarias debido al incremento en promedio de la actividad de sus puntos a lo largo de su duración. Dado que no se dispone de un preprocesamiento de la serie de datos original que en principio facilite su distinción, se tomó la decisión de abordar el problema mediante el uso de modelos de pronóstico. Los

modelos elegidos para esta sección son TFT, LSTM y Autoencoder.

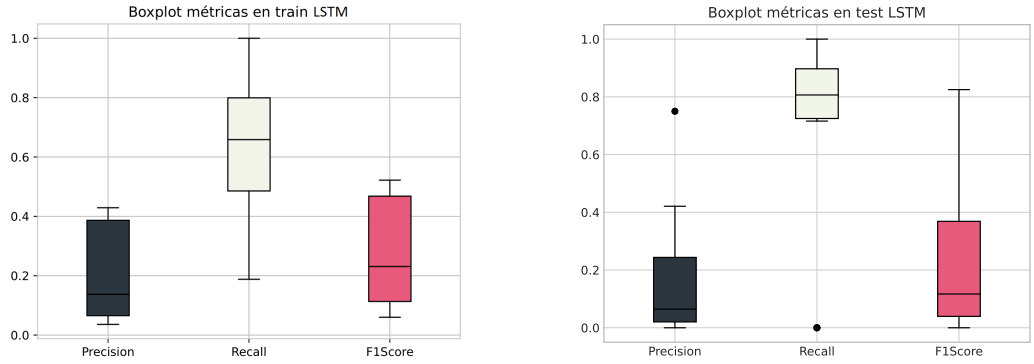



Figura 6.13: Boxplots de resultados de LSTM sobre todas las series con anomalías de actividad simuladas.

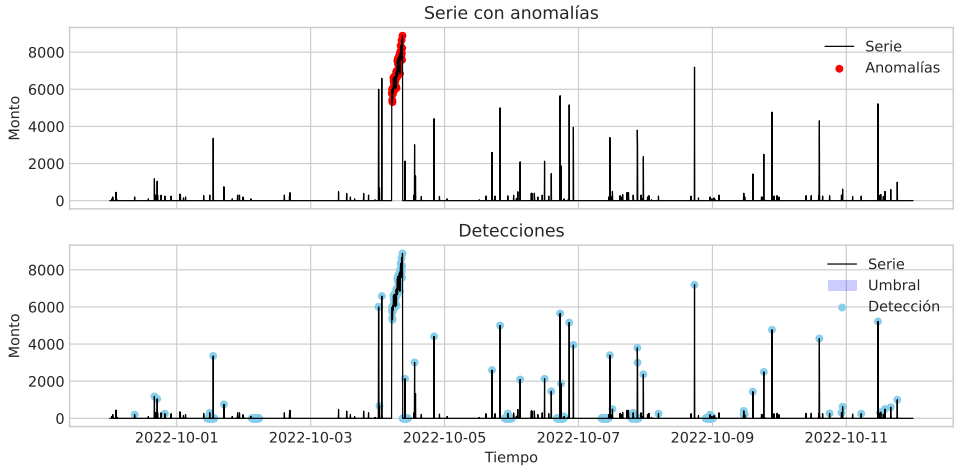
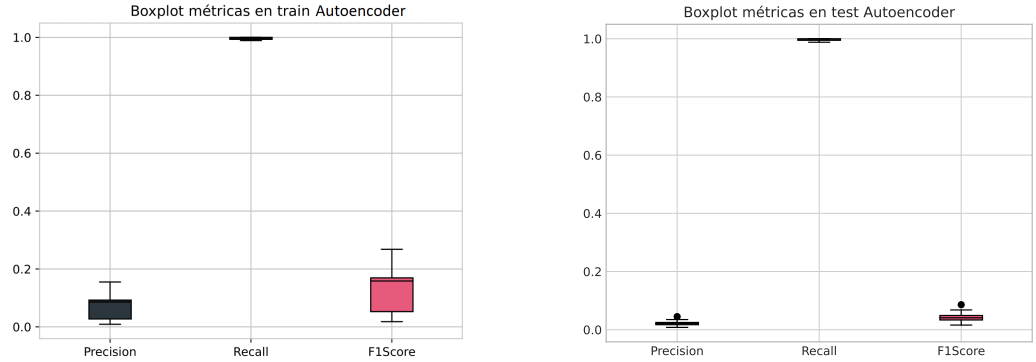



Figura 6.14: Resultados de LSTM en la detección de anomalías de actividad en la serie número 3.

LSTM presenta los mejores resultados de los tres como puede observarse en 6.13,6.15 y 6.17. Teniendo en cuenta que las métricas utilizadas premian la detección de anomalías tempranas y en intervalos continuos, y que la naturaleza de las anomalías dificulta dicha tarea, las detecciones realizadas por el LSTM son muy buenas como se muestra en 6.14. Este resultado puede deberse al enfoque probabilístico que adopta el LSTM. Para cada predicción, las redes bayesianas que conforman el modelo permiten sortear sus pesos según la distribución de probabilidad aprendida y por consiguiente variar ligeramente la salida del modelo. Tomando una gran cantidad de estas realizaciones para cada punto a inferir, es posible estimar una media y desviación estándar para la salida del modelo (la cual

es asumida Gaussiana) y detectar anomalías considerando a cada observación que se encuentre a más de tres desviaciones estándar de la media de la distribución de salida como una anomalía.




(a) (b)

Figura 6.15: Boxplots de resultados de Autoencoder sobre todas las series con anomalías de actividad simuladas.

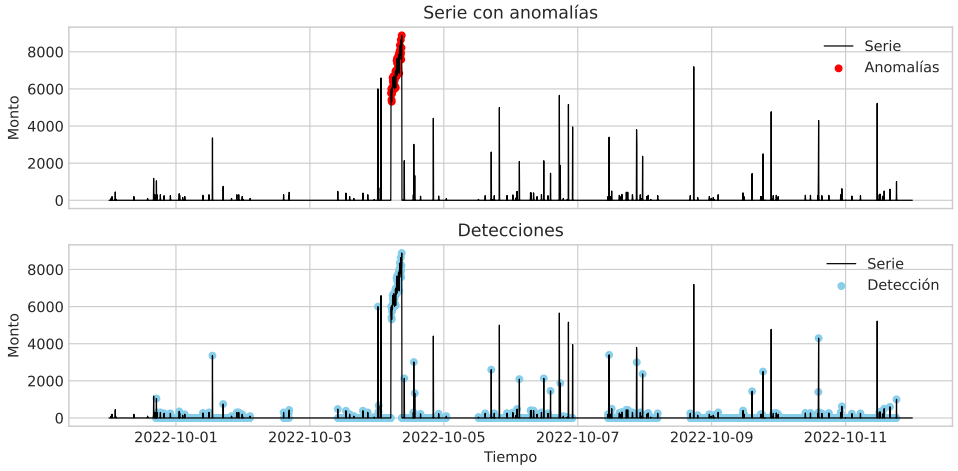



Figura 6.16: Resultados de Autoencoder en la detección de anomalías de actividad en la serie número 3.

Siguiendo una línea similar al LSTM, el Autoencoder esta compuesto por capas recurrentes, aunque en este caso no son bayesianas. Mientras el LSTM busca predecir la observación siguiente dados los datos anteriores a la misma, el Autoencoder busca aprender la dinámica y patrones de la serie en sí, lo que sugeriría una aplicación directa a la detección de anomalías. Sin embargo, como se muestra en la figura 6.15 parece que el modelo no fue capaz de aprender dichos patrones, tal vez por la naturaleza esparsa de las series. No obstante, las detecciones realizadas por el Autoencoder en algunas series son bastante buenas, aunque como sus intervalos

de predicción son discontinuos, la métrica penaliza su desempeño en gran medida, un ejemplo de estas detecciones puede verse en la figura 6.16.

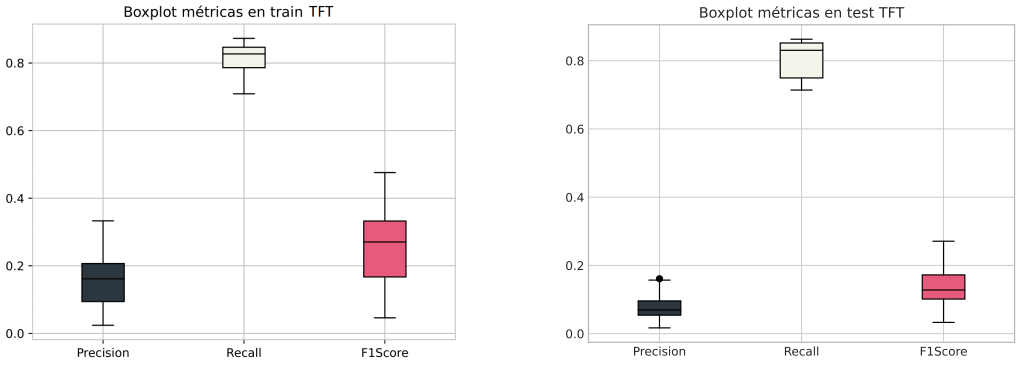



Figura 6.17: Boxplots de resultados de TFT sobre todas las series con anomalías de actividad simuladas.

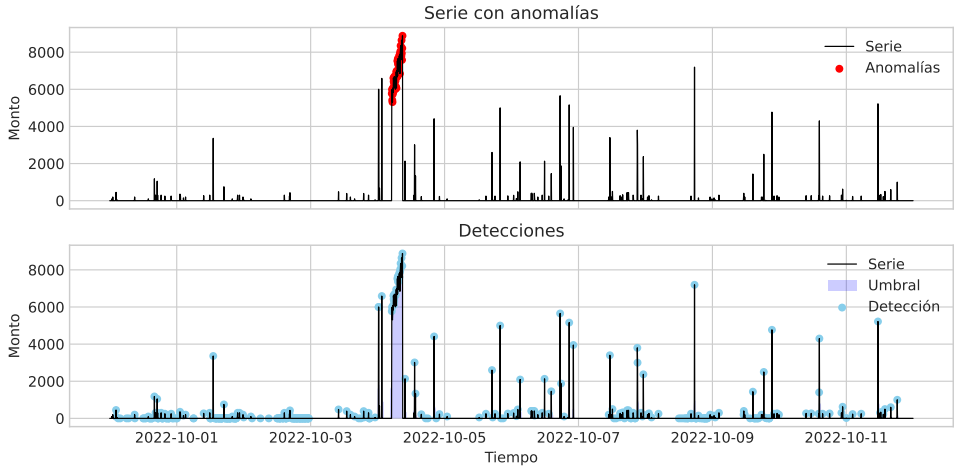



Figura 6.18: Resultados de TFT en la detección de anomalías de actividad en la serie número 3.

Para finalizar con el análisis, a diferencia del Autoencoder y LSTM, el modelo basado en TFT clasifica las anomalías utilizando un cuantil de la distribución de probabilidad predicha, que en este caso se fijó en 98 %. Aunque los resultados que se muestran en la figura 6.18 parecen peores a los del autoencoder, sus métricas son superiores en comparación, como puede verse en la figura 6.17. Esto se debe a que las detecciones que realiza, pese a tener una mayor cantidad de falsos positivos, se realizan en intervalos continuos en la sección donde están presentes las anomalías de actividad. En contraposición, el autoencoder tiene menos porcentaje de falsos positivos, pero sus detecciones sobre la anomalía, aunque no se aprecie en la gráfica, son discontinuas.

## 6.4. Resumen

En esta sección se resumen todos los resultados de las secciones anteriores. Las métricas promedio *precision*, *recall* y *F1* en el conjunto de prueba se muestran en las tablas 6.1, 6.2 y 6.3, respectivamente. Los modelos fueron entrenados cada uno con un conjunto único de hiperparámetros y evaluados en todas las series, pues se pretende comparar el desempeño general de los detectores en todos los datos disponibles.

Observando en la tabla 6.3, se pueden destacar los resultados de los modelos DBSCAN, HMM, FM y LSTM en la detección de anomalías de demanda, siendo FM el mejor de los cuatro. En el caso de las anomalías de inactividad, menos modelos son capaces de detectarlas eficientemente. De hecho, solo pueden destacarse los resultados de DBSCAN, LSTM y HMM, con este último siendo el de mejor desempeño. Por último, en lo que respecta a anomalías de actividad, pueden destacarse únicamente HMM por sus buenos resultados, aunque DBSCAN, FM y LSTM presentan un desempeño regular.

Si bien todos los resultados anteriores deben interpretarse tomando en cuenta las limitaciones presentadas en 5.3.4, se puede concluir que generalmente, el detector basado en HMM es el que muestra mayor potencial. Por otro lado, los detectores que usan la descomposición demanda-intervalo mostraron en general resultados mejores, lo que puede indicar que ese preprocesamiento es ventajoso.

Los detectores basados en modelos de *forecasting* (DeepAR, TFT) mostraron desempeño pobre. Ninguno logra predecir correctamente intervalos de confianza razonables, lo que se traduce en malas detecciones. Esto puede deberse a varias razones, pero creemos que la cantidad de datos puede no ser suficiente para que el modelo aprenda los patrones normales de las series ya que en general, los modelos basados en redes neuronales y en *transformers* los requieren en gran cantidad y variabilidad. En particular, todos tienen dificultades con los picos de demanda que, como se puede ver en la figura 5.1, son bastante usuales pero que aparecen en todo el rango de tiempo en el orden de las decenas de veces.

Tabla 6.1: Resultados *precision*.

| | DBSCAN | HMM | FM | TFT | DeepAR | Autoencoder | LSTM | Naive |
|---|---|---|---|---|---|---|---|---|
| Demanda | 0.89 | 0.83 | 0.88 | 0.08 | 0.16 | 0.14 | 0.57 | 0.99 |
| Inactividad | 0.84 | 0.35 | 0.00 | 0.06 | 0.00 | 0.48 | 0.11 | 0.93 |
| Actividad | 0.18 | 0.39 | 0.14 | 0.10 | 0.00 | 0.08 | 0.14 | 0.04 |

Tabla 6.2: Resultados *recall*.

| | DBSCAN | HMM | FM | TFT | DeepAR | Autoencoder | LSTM | Naive |
|---|---|---|---|---|---|---|---|---|
| Demanda | 0.74 | 0.94 | 0.97 | 0.96 | 0.71 | 0.98 | 1.00 | 1.00 |
| Inactividad | 0.90 | 1.00 | 0.00 | 0.52 | 0.00 | 0.67 | 0.80 | 0.95 |
| Actividad | 0.92 | 0.59 | 0.91 | 0.78 | 0.11 | 1.00 | 0.86 | 0.99 |

Tabla 6.3: Resultados *F1*.

| | DBSCAN | HMM | FM | TFT | DeepAR | Autoencoder | LSTM | Naive |
|---|---|---|---|---|---|---|---|---|
| Demanda | 0.78 | 0.87 | 0.91 | 0.14 | 0.25 | 0.23 | 0.69 | 0.99 |
| Inactividad | 0.87 | 0.48 | 0.00 | 0.10 | 0.00 | 0.51 | 0.18 | 0.92 |
| Actividad | 0.24 | 0.41 | 0.22 | 0.17 | 0.00 | 0.13 | 0.21 | 0.07 |

## 6.5. Robustez de los modelos

La robustez es una propiedad de gran importancia en el proceso de diseño de soluciones dentro del ámbito de la ingeniería. Este principio también se aplica al campo del aprendizaje automático, en el que se dispone de varias herramientas para mejorar y evaluar dicha propiedad en los diferentes tipos de modelos [35].

Con el objetivo de tener un modelo con el cuál comparar el desempeño de los modelos implementados, se diseñó un modelo simple o *naive* el cual realiza detecciones ingenuas para cada tipo de anomalías. Cómo las anomalías de demanda se caracterizan por su gran magnitud en comparación con el resto de los datos, el criterio utilizado para su detección en dicho modelo será considerar todo lo que se encuentre por encima del valor máximo de la serie como una anomalía de demanda.

Por otro lado, en este modelo los períodos de inactividad se definen como los intervalos de la serie con ceros consecutivos. De este modo, el modelo detecta una anomalía de inactividad cuando se encuentra con un período de inactividad cuyo largo es más grande al mayor período de inactividad detectado en la serie.

Finalmente, como las anomalías de actividad se caracterizan principalmente por un incremento en la frecuencia de actividad, es decir, de valores no nulos, el modelo simple detecta como anomalía de actividad todos los datos consecutivos cuyo valor sea distinto de cero.

### 6.5.1. Anomalías en presencia de ruido

En lo que va del análisis, se ha tratado de fortalecer la robustez de los modelos mediante la optimización de sus hiperparámetros. Sin embargo, también resulta interesante explorar el concepto de robustez en relación al problema específico que se enfrenta. En este sentido, se plantea la posibilidad de introducir ruido en las anomalías más susceptibles al mismo, las cuales corresponden a las anomalías de inactividad. Al añadir ruido la mayoría de los datos que anteriormente mostraban valores nulos durante el período de inactividad ahora mostraran valores potencialmente muy pequeños pero no nulos, lo cual dificultará su detección. Dado que las series disponibles tienen únicamente valores no negativos, se utiliza ruido uniforme, siendo cero el limite inferior de la misma y el tercer valor no nulo más chico de cada serie como limite superior. Esto se define así con el objetivo de que el ruido no tenga una magnitud demasiado grande y la anomalía pueda seguir siendo considerada como de inactividad.

En la tabla 6.4 se observa el resultado de experimentar con anomalías de inactividad ruidosas en las detecciones. El cambio en la estructura básica de la anomalía resulta afectar completamente el funcionamiento de los tres detectores puestos a

prueba. Sin embargo, como se observa en la ultima fila de la tabla, al modificar el hiperpárametro $m$, el cual proviene de las definiciones de serie de demanda y serie de inactividad en 5.2 que son utilizados en los modelos HMM y DBSCAN se pudo amortiguar hasta cierto punto la caída del desempeño. Al hiperparámetro $m$ se le asignó el valor del cuantil de la serie correspondiente al porcentaje de esparsidad. La justificación de lo anterior es que al ingresar ruido a la serie, los valores nulos serán trasladados a los cuantiles más bajos de la serie, aumentando su proporción.

Tabla 6.4: F1 con anomalías de inactividad ruidosas.

| | NAIVE | DBSCAN | HMM |
|---|---|---|---|
| Inactividad | 0.93 | 0.86 | 0.87 |
| Inactividad ruidosa | 0.0 | 0.0 | 0.04 |
| Inactividad ruidosa tratada | 0.0 | 0.16 | 0.18 |

### 6.5.2. Anomalías en el conjunto de entrenamiento

Otro escenario interesante a considerar en el análisis de robustez consiste en introducir una cantidad reducida de anomalías en el conjunto de entrenamiento. En la figura 6.3 se presenta el promedio del puntaje *F1* de todos los modelos evaluados para diferentes tipos de anomalías en todas las series de datos, a excepción de las series correspondientes a las entidades 1,9,10 y 12. Es notable que, a excepción de las anomalías de actividad, el modelo simple (*naive*) muestra un rendimiento superior en muchos casos en comparación con los demás modelos para los otros tipos de anomalías. Este resultado era en cierta medida predecible, dado que al conocer el método de generación de anomalías, es fácil desarrollar un detector que se adapte a las mismas.

No obstante, al introducir anomalías en el entrenamiento, se espera que el modelo simple presente un rendimiento peor. Se espera que en un modelo robusto entrenado en estas condiciones, su rendimiento en el conjunto de prueba no se vea significativamente afectado.

Los resultados de este experimento se presentan en la figura 6.5, donde se ha inyectado en el conjunto de entrenamiento una anomalía del tipo correspondiente, para cada serie. De los modelos de la tabla, destacan positivamente cuatro.

Tabla 6.5: Experimento con anomalías en conjunto de train, resultados *F1*.

| | DBSCAN | HMM | FM | TFT | DeepAR | Autoencoder | LSTM | Naive |
|---|---|---|---|---|---|---|---|---|
| Demanda | 0.74 | 0.27 | 0.93 | 0.22 | 0.19 | 0.05 | 0.60 | 0.53 |
| Inactividad | 0.38 | 0.22 | 0.0 | 0.04 | 0.02 | 0.0 | 0.13 | 0.33 |
| Actividad | 0.04 | 0.18 | 0.0 | 0.04 | 0.0 | 0.02 | 0.14 | 0.05 |

Por un lado, FM obtuvo excelentes resultados en la detección de anomalías de demanda, aumentando incluso su puntaje en la misma y superando ampliamente al modelo simple que vio su *F1* reducido desde 0.99 a 0.53. En lo que respecta a las anomalías de actividad, HMM logró mantenerse por encima del modelo simple

aunque eso sí, reduciendo drásticamente su desempeño. DBSCAN fue capaz de superar al modelo simple tanto en anomalías de demanda como en anomalías de inactividad, mostrando muy buenos resultados en ambas.

El modelo basado en LSTM mostró resultados sorprendentes en la detección de anomalías de demanda, viéndose favorecido por la introducción de anomalías en el entrenamiento y superando ampliamente al modelo simple. A su vez, luego de HMM es el mejor modelo en la detección de anomalías de actividad.

En lo que respecta al Autoencoder, DeepAR y TFT, no lograron superar el desempeño mostrado por el modelo simple. Estos modelos tampoco dieron buenos resultados en el primer análisis, por lo que, de algún modo, es esperable que el modelo simple funcione mejor.

# Capítulo 7

# Conclusiones

A lo largo de este proyecto, se ha logrado abordar de manera relativamente satisfactoria el problema de detección en tiempo real de anomalías en series de tiempo esparsas. Se llevó a cabo una exhaustiva investigación de siete modelos diferentes, aprovechando sus capacidades en sus áreas de mayor fortaleza.

Algunos de los modelos implementados se integraron a una herramienta específica del cliente, lo cual permitió realizar detecciones en un ambiente de producción real, tal como se requería. Además, se desarrolló una metodología que permitió evaluar el desempeño de los modelos entrenados, incluso en ausencia de datos etiquetados, mediante la simulación sintética de anomalías.

La generación sintética de anomalías probó ser un problema difícil, pero entendemos que se logró un método que, aunque simple proporciona una base sólida para la evaluación de modelos de detección. Se logró obtener varios modelos con alto desempeño para la detección de tres tipos específicos de anomalías simuladas. Si bien la validez de los resultados está acotada por las limitaciones del enfoque de generación de anomalías sintéticas, se puede extraer información importante para desarrollos futuros y aplicaciones en ambientes reales.

La realización de este proyecto ha brindado al equipo una experiencia enriquecedora, fortaleciendo nuestras habilidades técnicas, de trabajo en equipo y de gestión del tiempo. Esto ha contribuido a nuestro crecimiento profesional y marca el cierre de una etapa, preparándonos para adentrarnos en el ámbito laboral con mayor solidez y confianza.

En cuanto a futuras investigaciones, sería interesante considerar la extensión de modelos como DBSCAN, HMM, Autoencoder y LSTM. Aunque actualmente no logran buen desempeño en los tres tipos de anomalías propuestos, creemos que poseen la flexibilidad y el potencial suficiente para obtener buenos resultados.

La generación de anomalías sintéticas es un tema amplio que podría ser materia de un proyecto independiente. Como ya se mencionó, este es un problema difícil que se abordó de forma relativamente simple dado el tiempo limitado con el que se contó para esta parte del proyecto.

Esta página ha sido intencionalmente dejada en blanco.

# Referencias


[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[2] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. Gluonts: Probabilistic time series modeling in python. *arXiv preprint arXiv:1906.05264*, 2019.

[3] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[4] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[5] Leonard E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

[6] Immanuel Bayer. fastfm: A library for factorization machines. *Journal of Machine Learning Research*, 17(184):1–5, 2016.

[7] Chris U. Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. Neural contextual anomaly detection for time series. *CoRR*, abs/2107.07702, 2021.

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[9] Alexandre Chorin and Ole Hald. *Stochastic Tools in Mathematics and Science*, pages 109–132. Springer New York, NY, 3 edition, 06 2015.

[10] J. D. Croston. Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society*, 23:289–303, 1972.

[11] Fatoumata Dama and Christine Sinoquet. Analysis and modeling to forecast in time series: a systematic review. *CoRR*, abs/2104.00164, 2021.

[12] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[13] Influx development team. Influxdb query language. Accedido: 28 de junio de 2023.

[14] Piero Esposito. Blitz - bayesian layers in torch zoo (a bayesian deep learing library for torch). `https://github.com/piEsposito/blitz-bayesian-deep-learning/`, 2020.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[16] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017.

[17] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[18] Gastón García González. Detección de anomalías en series multivariable con modelos generativos. Master's thesis, Universidad de la República (Uruguay). Facultad de Ingeniería. IIE, sep 2020.

[19] Gastón García González, Pedro Casas, Alicia Fernández, and Gabriel Gómez. Net-gan : Recurrent generative adversarial networks for network anomaly detection in multivariate time-series. In *TMA Conference 2020, Network Traffic Measurement and Analysis Conference, Berlin, Germany, 8-12 jun*, page 1, 2020.

[20] Gastón García González, Pedro Casas, Alicia Fernández, and Gabriel Gómez. Network anomaly detection with net-gan, a generative adversarial network for analysis of multivariate time-series. In *ACM Special Interest Group on Data Communication (SIGCOMM '20 Demos and Posters), Nueva York, NY, USA, 10-14 aug*, pages 1–3. ACM, 2020.

[21] Gastón García González, Pedro Casas, Alicia Fernández, and Gabriel Gómez. On the usage of generative models for network anomaly detection in multivariate time-series. In *WAIN 2020 : Workshop on AI in Networks and Distributed Systems, Milan, Italy, 2-6 nov*, pages 1–5, 2020.

[22] Gastón García González, Pedro Casas, Alicia Fernández, and Gabriel Gómez. Steps towards continual learning in multivariate time-series anomaly detection using variational autoencoders. In *IMC 22 : Proceedings of the*

*22nd ACM Internet Measurement Conference, Nice, France, 25-27 oct. https://doi.org/10.1145/3517745.3563033*, pages 774–775. ACM, 2022.

[23] Gastón García González, Sergio Martínez Tagliafico, Alicia Fernández, Gabriel Gómez, José Acuña, and Pedro Casas. Dc-vae, fine-grained anomaly detection in multivariate time-series with dilated convolutions and variational auto encoders. In *7th International Workshop on Traffic Measurements for Cybersecurity (WTMC 2022), Genoa, Italy Monday, jun 6*, pages 1–7. IEEE, 2022.

[24] Gastón García González, Sergio Martínez Tagliafico, Alicia Fernández, Gabriel Gómez, José Acuña, and Pedro Casas. Mining multivariate time-series for anomaly detection in mobile networks on the usage of variational auto encoders and dilated convolutions. In *8th SIGKDD International Workshop on Mining and Learning from Time Series – Deep Forecasting : Models, Interpretability, and Applications, Washington, DC, USA, aug 15*, pages 1–7. ACM, 2022.

[25] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science*, pages 45–87, 2020.

[26] Nico Goernitz, Mikio Braun, and Marius Kloft. Hidden markov anomaly detection. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1833–1842, Lille, France, 07–09 Jul 2015. PMLR.

[27] hmmlearn Contributors. hmmlearn: Hidden markov models in python. Accedido: 29 de junio de 2023.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[29] Inc. InfluxData. InfluxDB OSS 2.6 documentation, 2023. Accedido: 29 de junio de 2023.

[30] Inc. InfluxData. Telegraf documentation, 2023. Accedido: 29 de junio de 2023.

[31] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, may 2022.

[32] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000.

[33] Gamze Kaya, Merve Sahin, and Omer Demirel. Intermittent demand forecasting: a guideline for method selection. *Sādhanā*, 45, 12 2020.

[34] Danielle Kiefer, Florian Grimm, Markus Bauer, and Clemens van Dinther. Demand forecasting intermittent and lumpy time series: Comparing statistical, machine learning and deep learning methods. In *Hawaii International Conference on System Sciences*, 2021.

[35] Jan Kukacka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *CoRR*, abs/1710.10686, 2017.

[36] Grafana labs. Grafana documentation, 2023. Accedido: 29 de junio de 2023.

[37] Nikolay Pavlovich Laptev. Anogen : Deep anomaly generator nikolay laptev. Disponible en linea: `https://research.facebook.com/publications/anogen-deep-anomaly-generator/`, accedido el 29 de junio de 2023, 2018.

[38] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for NAS. *CoRR*, abs/1912.06059, 2019.

[39] Bryan Lim, Sercan O. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[40] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[41] Sergio Martínez Tagliafico, Gastón García González, Alicia Fernández, Gabriel Gómez, and José Acuña. Real time anomaly detection in network traffic time series. In *ITISE 2018 : International conference on Time Series and Forecasting, Granada, Spain, 19-21 sep*, pages 1–12, 2018.

[42] J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[43] Mohsin Munir, Shoaib Siddiqui, Andreas Dengel, and Sheraz Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, PP:1–1, 12 2018.

[44] Inc. NumFOCUS. Pandas documentation. Accedido: 29 de junio de 2023.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[47] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[48] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[49] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[50] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation.* MIT Press, Cambridge, MA, USA, 1986.

[52] Masanari Kimura Ryohei Izawa, Ryosuke Sato. Prts: Python library for time series metrics, 2021. Accedido: 29 de junio de 2023.

[53] H. Sak, Andrew Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 338–342, 01 2014.

[54] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, jul 2022.

[55] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *CoRR*, abs/1912.05911, 2019.

[56] Dominique T. Shipmon, Jason M. Gurevitch, Paolo M. Piselli, and Stephen T. Edwards. Time Series Anomaly Detection; Detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. *arXiv e-prints*, page arXiv:1708.03665, August 2017.

[57] A.A Syntetos and J.E Boylan. On the bias of intermittent demand estimates. *International Journal of Production Economics*, 71(1):457–466, 2001. Tenth International Symposium on Inventories.

[58] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, and Justin Gottschlich. A new model for evaluating range-based anomaly detection algorithms. *CoRR*, abs/1803.03639, 2018.

[59] Ali Caner Turkmen, Yuyang Wang, and Tim Januschowski. Intermittent demand forecasting with deep renewal processes, 2019.

[60] Chengyu Wang, Kui Wu, Tongqing Zhou, Guang Yu, and Zhiping Cai. Tsagen: Synthetic time series generation for kpi anomaly detection. *IEEE Transactions on Network and Service Management*, 19(1):130–145, 2022.

[61] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. Lstm-autoencoder based anomaly detection for indoor air quality time series data. *IEEE Sensors Journal*, 2023.

[62] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[63] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *CoRR*, abs/2009.13807, 2020.

[64] Mengxiao Zhu, Charu C. Aggarwal, Shuai Ma, Hui Zhang, and Jinpeng Huai. Outlier detection in sparse data with factorization machines. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 817–826, New York, NY, USA, 2017. Association for Computing Machinery.

# Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

# Índice de figuras

Esta es la última página.
Compilado el miércoles 18 octubre, 2023.
http://iie.fing.edu.uy/