Trabajo Fin de Máster

Estudio de la aplicación de modelos basados en Transformers a la predicción de series temporales

Xabier Suárez Dios

Máster en Técnicas Estadísticas

Curso 2024-2025

Propuesta del Trabajo Fin de Máster

Título en galego: Estudo da aplicación de modelos baseados en Transformers á predicción de series temporais
Título en español: Estudio de la aplicación de modelos basados en Transformers a la predicción de series temporales
English title: Study of the application of models based on Transformers to time series forecasting
Modalidad: Modalidad B
Autor: Xabier Suárez Dios, Universidade de Vigo
Directora: Beatriz Pateiro López, Universidade de Santiago de Compostela
Tutoras: Elena Doctor Bracho, SDG Consulting España, S.A.; Sara Gil Abad, SDG Consulting España, S.A.
Breve resumen del trabajo: Este trabajo presenta un estudio de la aplicación de los modelos Chronos, basados en arquitectura Transformer, en la predicción de series temporales. Además, previamente se exponen los fundamentos teóricos de la metodología empleada y de los modelos clásicos Box-Jenkins y de suavización exponencial.

Doña Beatriz Pateiro López, profesora titular del área de Estadística e Investigación Operativa de la Universidade de Santiago de Compostela, doña Elena Doctor Bracho, de SDG Consulting España, S.A., y doña Sara Gil Abad, de SDG Consulting España, S.A., informan que el Trabajo Fin de Máster titulado

Estudio de la aplicación de modelos basados en Transformers a la predicción de series temporales

fue realizado bajo su dirección por don Xabier Suárez Dios para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para la presentación y defensa ante un tribunal. Además, doña Beatriz Pateiro López y don Xabier Suárez Dios

√ si □ no

autorizan la publicación de la memoria en el repositorio de acceso público asociado al Máster en Técnicas Estadísticas.

En Vigo, a 2 de junio de 2025.

La directora:
Doña Beatriz Pateiro López

PATEIRO LOPEZ BEATRIZ - 46898344X
Firmado digitalmente por PATEIRO LOPEZ BEATRIZ - 46898344X
Fecha: 2025.06.02 09:11:46 +02'00'

La cotutora:
Doña Sara Gil Abad

El autor:
Don Xabier Suárez Dios

La tutora:
Doña Elena Doctor Bracho

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

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

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

Agradecimientos

A todos los profesionales que han contribuido a mi formación académica durante estos dos años de posgrado, tanto profesores de las Universidades de Vigo, Coruña y Santiago de Compostela, como trabajadores de *SDG Consulting España, S.A.*, sobre todo a mis tutoras Elena Doctor Bracho y Sara Gil Abad, quienes siempre se preocuparon por ayudarme durante mi estancia en la empresa.

En especial, a mi tutora académica de este trabajo, Beatriz Pateiro López, del Departamento de Estadística e Investigación Operativa de la Universidade de Santiago de Compostela, por su confianza y dedicación.

Y a mis seres cercanos y familia, sobre todo, a mis padres. Sin su apoyo, nunca hubiera llegado hasta este punto.

Índice general

Resumen

Resumen en español

La predicción de series temporales constituye una tarea ampliamente desarrollada hoy en día en multitud de ámbitos. Por ello, durante mi estancia en *SDG Consulting España, S.A.*, se me ofreció la posibilidad de evaluar el rendimiento predictivo de modelos avanzados de aprendizaje profundo aplicados a un conjunto de datos reales proporcionados por la empresa. Concretamente, el estudio se centró en los modelos Chronos, desarrollados por Amazon Science y basados en arquitectura Transformer, metodología de vanguardia en el ámbito del *deep learning*.

Como punto de partida, y con el fin de establecer un marco teórico sólido, se llevó a cabo una revisión bibliográfica exhaustiva acerca de las metodologías clásicas de predicción de series temporales: los modelos Box-Jenkins y las técnicas de suavización exponencial. Esta revisión permitió contextualizar y comparar los enfoques tradicionales con las metodologías más recientes basadas en redes neuronales, conocidas como técnicas de aprendizaje automático.

Posteriormente, se procedió a la implementación y evaluación empírica de los modelos Chronos, llevando a cabo un análisis estadístico a través del lenguaje de programación Python.

Finalmente, el trabajo concluye con la exposición de los resultados obtenidos en dicho análisis y de las conclusiones extraídas sobre los modelos Chronos, centradas en su aplicabilidad en entornos empresariales reales, destacando sus fortalezas, limitaciones y posibles líneas de mejora futuras.

English abstract

Time series forecasting is a widely developed task nowadays in many fields. Therefore, during my stay at *SDG Consulting España, S.A.*, I was offered the possibility to evaluate the predictive performance of advanced deep learning models applied to a real dataset provided by the company. Specifically, the study focused on Chronos models, developed by Amazon Science and based on Transformer architecture, a cutting-edge methodology in the field of deep learning.

As a starting point, and in order to establish a solid theoretical framework, an exhaustive literature review was carried out on the classical time series forecasting methodologies: Box-Jenkins models and exponential smoothing techniques. This review allowed us to contextualize and compare the traditional approaches with the most recent methodologies based on neural networks, known as machine learning techniques.

Subsequently, we proceeded to the implementation and empirical evaluation of the Chronos models, carrying out a statistical analysis using the Python programming language.

Finally, the paper concludes with the results obtained in this analysis and the conclusions drawn about the Chronos models, focusing on their applicability in real business environments, highlighting their strengths, limitations and possible lines of future improvement.

Capítulo 1

Introducción

En este primer apartado introductorio del trabajo, los principales objetivos se centran en dar a conocer el contexto sobre el que se desarrolló el mismo, así como los objetivos específicos planteados. Además, se tratarán los métodos y recursos empleados durante su desarrollo y se especificará la estructura seguida en su elaboración.

1.1. Contexto y justificación

Hoy en día, la sociedad se encuentra inmersa en una transformación digital continua. La gran mayoría de actividades que lleva a cabo el ser humano, ya sean personales, profesionales, científicas, académicas, etc., generan datos de forma constante: redes sociales, navegación por internet, comercio en línea, marketing, experimentos en laboratorios, encuestas sociales, entre otras. Un ejemplo de esto es el que se recoge en OECD (2024), donde se constata que la digitalización está transformando la economía global, creando nuevos desafíos y oportunidades para empresas y gobiernos en la gestión de datos e innovación digital. Esto evidencia que la ciencia de datos se encuentra en auge en todos los ámbitos.

En el ámbito académico y científico, una de las áreas más destacadas del análisis estadístico de datos es la obtención de predicciones de los valores de variables medidas a lo largo de un intervalo de tiempo determinado. Esto resulta realmente valioso en el entorno empresarial, ya que la capacidad de anticipar comportamientos futuros a partir de datos históricos permite obtener información estratégica sobre la toma de decisiones.

En esta línea, el presente Trabajo Fin de Máster se basó en la predicción de series temporales a partir de un conjunto de datos de ventas reales proporcionado por la empresa colaboradora con el mismo. En consonancia con los contenidos abordados durante el máster, se evaluó la capacidad predictiva de unos modelos novedosos de aprendizaje automático, basados en arquitectura Transformer, desarrollados por Amazon Science en 2024: los modelos Chronos (véase Ansari et al. (2024)), de manera que se pudiese así analizar su aplicabilidad a problemas reales con datos de carácter temporal.

1.2. Objetivos y alcance

El objetivo general del trabajo residió en analizar el rendimiento de los modelos Chronos en la predicción de series temporales, aplicándolos a un conjunto de datos reales sobre ventas de productos en los almacenes estadounidenses Walmart (véase Capítulo 4). De este modo, este análisis supondría un aporte notable tanto para mí, al permitirme conocer en detalle un tipo de modelos de machine learning desconocidos hasta el momento, como para *SDG Consulting España, S.A.*, ya que el estudio realizado junto con las correspondientes conclusiones extraídas añadió una nueva alternativa de trabajo a la empresa.

Desglosando este objetivo general, haciendo hincapié en los puntos más importantes, y atendiendo también a aquellos referentes a la primera parte del trabajo, dedicada a la revisión teórica de modelos clásicos y avanzados de aprendizaje automático, se detallan los siguientes objetivos específicos:

- Introducir los fundamentos teóricos de las series temporales.
- Explicar las principales características de los principales modelos clásicos de predicción de series temporales: modelos Box-Jenkins y suavizado exponencial.
- Analizar los modelos de aprendizaje automático en el contexto de predicción de series temporales, haciendo énfasis en aquellos basados en arquitectura Transformer.
- Implementar y documentar el proceso de evaluación de los modelos Chronos a la hora de predecir valores en series temporales.
- Evaluar el rendimiento de los modelos Chronos a través de diferentes métricas de error.
- Investigar sobre la utilidad de los *embeddings* asociados a las series temporales.
- Reflexionar acerca de las ventajas, limitaciones y posibles extensiones de la implementación de Chronos en situaciones reales.

1.3. Metodología

Para abordar los objetivos planteados en el subapartado anterior, se llevaron a cabo diferentes métodos de acuerdo con la naturaleza de cada uno de los pasos.

Inicialmente, se llevó a cabo una revisión bibliográfica exhaustiva, que permitió presentar en los Capítulos 2 y 3 una síntesis clara y fundamentada de las bases teóricas que sustentan los conceptos aplicados en la fase práctica del trabajo. Dicha revisión se basó principalmente en bibliografía especializada relacionada con los temas tratados, complementada con material docente impartido durante el máster.

En cuanto al Capítulo 4, la metodología definida por *SDG Consulting España, S.A.* consistió en la implementación de la librería Chronos a través del lenguaje de programación Python. La plataforma de edición de código recayó en mi libre elección, por lo que, debido a mi previa familiarización con ella, el editor empleado fue Visual Studio Code. Además, también se hizo uso de otra serie de paquetes de programación que ayudaron a realizar diversas tareas, como se detalla a continuación:

- **pandas:** manipulación del conjunto de datos en tareas como el manejo de DataFrames o la lectura de archivos CSV.
- **numpy:** cálculo numérico de las métricas de error e intervalos de predicción.
- **time:** medición del tiempo de entrenamiento de los modelos.
- **matplotlib, seaborn:** visualización de representaciones gráficas.
- **plotly:** creación de gráficos interactivos.
- **torch:** carga de los modelos.
- **scipy:** cálculo de distancias euclídeas.
- **sklearn:** escalado de embeddings y aplicación de técnicas de reducción de la dimensión y clustering.
- **hdbscan:** implementación del algoritmo de clusterización "HDBSCAN"

Por último, cabe destacar también el uso del programa de software de hojas de cálculo Excel para el almacenamiento de los datos, tanto los originales como los generados durante el desarrollo del trabajo. Y, todo esto, se pudo realizar gracias a un ordenador portátil, cedido por la empresa, con las siguientes características: marca Lenovo-Thinkpad, sistema operativo Windows 11, capacidad de 32 GB de memoria RAM y procesador Intel Core i7-1165G7.

1.4. Estructura

Para facilitar la comprensión del desarrollo de esta memoria, a continuación se describe la organización de los capítulos que suceden a esta introducción.

El Capítulo 2, titulado "Series temporales: modelos clásicos", comienza con la introducción del concepto de serie temporal en la Sección 2.1, en la que también se tratan las componentes más relevantes de las mismas. A continuación, se detallan las principales características de los procesos estocásticos en la Sección 2.2 y, por último, se analizan dos enfoques de predicción tradicionales: la metodología Box-Jenkins (Sección 2.3) y los modelos basados en suavización exponencial (Sección 2.4).

El Capítulo 3 está dedicado al aprendizaje automático (*machine learning*), que constituye el eje central de la aplicación práctica desarrollada posteriormente. La Sección 3.1 presenta una visión general del enfoque metodológico, mientras que la Sección 3.2 introduce los fundamentos del aprendizaje profundo (*deep learning*), como extensión avanzada del primero. A continuación, la Sección 3.3 aborda los modelos basados en arquitectura Transformer, que constituyen una de las múltiples alternativas que ofrece el aprendizaje profundo en este contexto, a través de la explicación de su definición, estructura interna y adecuación en la predicción de series temporales.

El Capítulo 4 expone los detalles de la implementación práctica. Primeramente, se introduce el conjunto de datos real empleado para el estudio, a través de un pequeño análisis exploratorio del mismo en la Sección 4.1, y en la Sección 4.2 se presentan los modelos de arquitectura Transformer implementados (Chronos). Finalmente, las Secciones 4.3 y 4.4 recogen la evaluación de su rendimiento en tareas de predicción temporal y un análisis de representación vectorial mediante embeddings, respectivamente.

Finalmente, en el Capítulo 5 se exponen las conclusiones extraídas tras la evaluación de los modelos Chronos, destacando los principales hallazgos obtenidos, así como posibles líneas futuras de trabajo.

Capítulo 2

Series temporales: modelos clásicos

Este capítulo está dedicado a la comprensión del concepto de serie temporal y a la introducción a los modelos clásicos que permiten realizar predicciones sobre las mismas.

Por ello, en el primer apartado se definirá este concepto y se detallarán sus principales componentes y características, para que, una vez contextualizado, se pueda dar paso a tratar los modelos Box-Jenkins y el método de suavización exponencial. Estos enfoques, aunque suponen una perspectiva tradicional de la metodología predictiva, están lo suficientemente reconocidos y se continúan empleando a día de hoy, por lo que conviene abordarlos antes de adentrarse en el siguiente nivel (Capítulo 3).

Cabe mencionar que la revisión bibliográfica llevada a cabo para la elaboración de este capítulo fue apoyada, principalmente, por Carrión-García (2001) y Brockwell y Davis (1991).

2.1. Definición y componentes

Una serie temporal se define como un conjunto de observaciones de una variable cuantitativa tomadas en puntos sucesivos en el tiempo o durante períodos sucesivos. Aunque es posible que las observaciones se recojan en intervalos irregulares de tiempo, lo más común, e ideal, es que sean medidas de forma regular. Así, denotando por X a la variable en cuestión, la serie de tiempo observada se expresa como: $x_1, x_2, ..., x_T$, siendo T el número de instantes de tiempo evaluados.

La herramienta que permite representar gráficamente una serie de tiempo es su gráfico secuencial, el cual se construye representando cada observación x_t frente al instante t en el que se observa. Así, si se repite este proceso T veces con cada par (t, x_t) y se unen los puntos resultantes, se obtiene el gráfico secuencial de la serie temporal en cuestión. A partir de este gráfico, que muestra la evolución de la serie a lo largo del tiempo en el que es registrada, se pueden observar ciertas componentes típicas de las series temporales:

- **Tendencia:** evolución de la serie a largo plazo. Su existencia se detecta ante la presencia de un patrón ascendente o descendente a lo largo del tiempo, es decir, en el caso de que el nivel de la serie no permanezca constante.
- **Estacionalidad:** comportamiento periódico de la serie. Aparecen patrones repetitivos en intervalos regulares con una frecuencia fija y conocida. Factores estacionales diarios, semanales o anuales son los más comunes.
- **Heterocedasticidad:** presente cuando la variabilidad de la serie no es constante. Los casos más comunes son aquellos en los que esta depende del nivel de la serie, y se suele corregir mediante transformaciones Box-Cox, que se estudiarán en el apartado 2.3.3.
- **Ruido:** se refiere a variaciones observadas en las series debidas a la aleatoriedad, por lo que no responden a ningún patrón sistemático.

2.2. Procesos estocásticos

Desde un punto de vista estadístico, para poder realizar inferencia (predicción, en este caso) sobre una serie temporal, esta debe entenderse como una muestra generada a partir de un proceso subyacente denominado proceso estocástico. Un proceso estocástico consiste en un conjunto de variables aleatorias definidas sobre un mismo espacio de probabilidad, y se define: $\{X_t\}_{t\in C}$, donde C se refiere al conjunto índice que representa el tiempo, por ejemplo, $C = \mathbb{Z}$ o $C = \mathbb{N}$ si t toma valores enteros (tiempo discreto), o $C = \mathbb{R}$ en el caso de que las observaciones temporales se puedan evaluar en cualquier punto de un intervalo (tiempo continuo); y el subíndice t al instante de tiempo en el que es observada cada variable aleatoria. De este modo, una serie de tiempo es, por tanto, una realización o trayectoria parcial de un proceso estocástico.

Antes de dar paso al apartado dedicado a los modelos Box-Jenkins que analizan las series con el objetivo de obtener predicciones temporales, es necesaria la introducción del término de estacionariedad, ya que dichos modelos se sustentan bajo su supuesto. Y, para ello, dado un proceso estocástico $\{X_t\}_t$, primero se definen las siguientes medidas:

- **Media:** la media del proceso estocástico en el instante t, también llamada función de medias, es una medida de tendencia central que describe el valor esperado de la variable aleatoria X_t. Se define como:
$$\mu_t = E(X_t) \tag{2.1}$$
- **Varianza:** medida del grado de variabilidad o dispersión de cada variable X_t con respecto a su media (μ_t). Se expresa como:
$$\sigma_t^2 = Var(X_t) = E((X_t - \mu_t)^2) \tag{2.2}$$
- **Autocovarianza:** medida del grado de dependencia lineal entre dos instantes de tiempo diferentes. Se define de la siguiente manera:
$$\gamma(s,t) = Cov(X_s, X_t) = E((X_s - \mu_s)(X_t - \mu_t)) \tag{2.3}$$

Así, un proceso estocástico se dice que es estacionario si se cumplen tres condiciones:

- El valor medio de la serie permanece constante a lo largo del tiempo: $\mu_t = \mu, \quad \forall t$
- La variabilidad de la serie no fluctúa en el tiempo: $\sigma_t^2 = \sigma^2, \quad \forall t$
- La función de autocovarianzas entre dos instantes depende únicamente de la diferencia entre los mismos: $\gamma(s,t) = \gamma_{|s-t|}, \quad \forall s,t$

Estas propiedades dotan a los procesos estocásticos de estabilidad en su media, varianza y autocovarianzas, permitiendo así estimar las diferentes características del mismo a partir de la serie de tiempo asociada.

Esto así, existen dos tipos de procesos estocásticos particulares de gran relevancia en el análisis de series temporales, conocidos como ruido blanco y paseo aleatorio. A continuación se exponen las principales características de cada uno de ellos, ya que su entendimiento es clave para la posterior introducción de los modelos Box-Jenkins:

El ruido blanco, que se denota por $\{a_t\}_t$, se define como una colección de variables aleatorias incorreladas con media nula y varianza finita. Es decir, se debe cumplir:

- $\mu_t = 0, \quad \forall t$
- $\sigma_t^2 = \sigma_a^2, \quad \forall t$
- $\gamma(s,t) = 0, \quad \forall t \neq s$

Por ello, el ruido blanco, además de ser un proceso estocástico, es estacionario. Como se verá en el siguiente capítulo, el cumplimiento de esta condición será necesario en el término de error (residuos) de los modelos ajustados mediante la metodología Box-Jenkins.

Por otro lado, habiendo descrito $\{a_t\}_t$ (ruido blanco), se define paseo aleatorio como aquel proceso estocástico expresado de la siguiente forma:

$$X_t = c + X_{t-1} + a_t \tag{2.4}$$

donde c es un parámetro.

Si se establece que $X_0 = 0$ y $t \geq 0$, se cumplen las siguientes propiedades:

- $\mu_t = ct, \quad \forall t$
- $\sigma_t^2 = \sigma_a^2 t, \quad \forall t$
- $\gamma(s,t) = \text{mín}(s,t)\sigma_a^2, \quad \forall s,t \geq 1$

Al contrario que $\{a_t\}_t$, en este caso el proceso estocástico resulta no estacionario, pues aparece una componente no determinista que afecta a su tendencia con respecto al valor inicial. Por ello, tal y como se explicará en profundidad en la siguiente sección, será necesario aplicar una diferenciación de primer orden a la serie temporal para lograr la estacionariedad.

Una vez abordados estos conceptos, se está en condiciones de profundizar sobre los modelos de predicción temporal Box-Jenkins.

2.3. Metodología Box-Jenkins

2.3.1. Definiciones previas

Recapitulando los contenidos anteriores, sea $\{a_t\}_t$ un proceso de ruido blanco y $\{X_t\}_t$ un proceso estocástico, se definen los siguientes tipos de procesos:

- **Proceso lineal:** se define como una combinación lineal de errores (ruido blanco), por lo que se expresa como:

$$X_t = c + \sum_{i=-\infty}^{\infty} \psi_i a_{t-i}, \tag{2.5}$$

  donde los coeficientes $\{\psi_i\}$ determinan la influencia de cada $\{a_{t-i}\}$. Este tipo de representación de los procesos es crucial, pues, como se verá a lo largo del capítulo, a partir de ellos es posible aplicar la metodología Box-Jenkins.

- **Proceso causal:** se puede entender como un caso particular de un proceso lineal en el que los valores de la variable dependen únicamente de errores pasados y presentes, nunca futuros. Esta característica resulta fundamental para predecir valores de series, pues en la práctica únicamente se dispone de información hasta la actualidad. Se expresa:

$$X_t = c + \sum_{i=0}^{\infty} \psi_i a_{t-i} \tag{2.6}$$

  Para garantizar que la varianza del proceso sea finita, se requiere que $\sum_{i=0}^{\infty} |\psi_i| < \infty$.

- **Proceso invertible:** Se expresa de la siguiente manera:

$$X_t = c + a_t + \sum_{i=1}^{\infty} \pi_i X_{t-i}, \tag{2.7}$$

por lo que, atendiendo a a_t, se observa que el ruido blanco se puede expresar como una combinación lineal de los valores no futuros de las series. En este caso también se parte de que $\sum_{i=1}^{\infty} |\pi_i| < \infty$ y, en el contexto de los modelos Box-Jenkins, esto juega un papel clave, debido a que permite expresar los errores como funciones de observaciones pasadas, lo que facilita su identificación.

Como resultado de esto, se presenta la descomposición de Wold:

$$X_t = \sum_{i=0}^{\infty} \psi_i a_{t-i}, \quad \psi_0 = 1, \quad \sum_{i=0}^{\infty} \psi_i^2 < \infty, \tag{2.8}$$

a partir de la cual se garantiza que todo proceso estacionario es o puede ser transformado en un proceso lineal.

De cara a la identificación de modelos en la práctica, conviene también especificar las siguientes funciones:

- **Función de Autocorrelaciones (ACF, por sus siglas en inglés):** medida relativa normalizada de la función de autocovarianzas, por lo que toma valores en el intervalo $[-1, 1]$ y se expresa:

$$\rho(s,t) = \frac{\gamma(s,t)}{\sigma(s)\sigma(t)}, \tag{2.9}$$

  siendo $\sigma(s)$ y $\sigma(t)$ las desviaciones estándar en los instantes s y t, respectivamente ($\sigma_i = \sqrt{\sigma_i^2}$).

- **Función de Autocorrelaciones Parciales (PACF, por sus siglas en inglés):** medida de la correlación entre dos variables distintas en la que se elimina el efecto lineal de las observaciones intermedias comprendidas entre s y t. Se define como:

$$\phi(s,t) = \frac{\text{Cov}\left(X_s - \hat{X}_s^{(s,t)}, X_t - \hat{X}_t^{(s,t)}\right)}{\sqrt{\text{Var}\left(X_s - \hat{X}_s^{(s,t)}\right) \text{Var}\left(X_t - \hat{X}_t^{(s,t)}\right)}}, \tag{2.10}$$

  donde $\hat{X}_j^{(s,t)}$ es el mejor predictor lineal de X_j, calculado utilizando las variables observadas en los instantes de tiempo entre s y t, pero sin considerar las observaciones en esos instantes específicos de tiempo. También toma valores en el intervalo $[-1, 1]$.

Estas medidas serán de gran importancia en el apartado de identificación de modelos, donde, atendiendo a la representación gráfica de las mismas a lo largo del retardo evaluado, será posible identificar el tipo de proceso que generan las series de tiempo a predecir.

2.3.2. Procesos estacionarios

Una vez establecidas las bases teóricas sobre las que se cimienta la metodología Box-Jenkins, se da paso a detallar cada uno de los posibles modelos estocásticos paramétricos generadores de series de tiempo.

En primer lugar, derivado de los procesos lineales invertibles, aparecen los modelos autorregresivos. Dado un proceso estacionario $\{X_t\}_t$ que admita una representación tal que:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + a_t, \tag{2.11}$$

donde c y $\{\phi_i\}$ son constantes, con $\phi_p \neq 0$, se dice que el modelo es autorregresivo de orden p, y se denota por AR(p).

Las características representativas de un proceso AR(p) son las siguientes:

- Es estacionario si se cumple que $1 - \sum_{i=1}^{p} \phi_i z^i \neq 0, \quad \forall z \text{ con } |z| = 1$
- Si es estacionario, es causal.
- Siempre es invertible.
- Tras los primeros retardos, los valores de la ACF convergen rápidamente a cero de forma exponencial y/o sinusoidal.
- El último valor observable de la PACF ocurre en el retardo p.

Por otro lado, se dice que un proceso es de medias móviles de orden q (MA(q)) si admite la siguiente representación:

$$X_t = c + \sum_{j=0}^{q} \theta_j a_{t-j}, \tag{2.12}$$

en la que c y $\{\theta_j\}$ también son constantes, con $\theta_q \neq 0$.

En este caso, las propiedades que verifican son las siguientes:

- Es invertible si se cumple que $1 + \sum_{j=1}^{q} \theta_j z^j \neq 0, \quad \forall z \text{ con } |z| \leq 1$
- Siempre es estacionario y causal.
- La ACF se anula a partir del retardo posterior a q.
- Igual que en el caso de la ACF en los modelos AR(p), los valores de la PACF se desvanecen lentamente.

A raíz de esto, se obtienen los procesos ARMA. Estos modelos integran tanto estructura AR como MA, por lo que se entienden como una combinación lineal de valores y errores pasados, respectivamente. Así, definidos los retardos p y q de cada subproceso, se obtiene la representación de un modelo ARMA(p, q):

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=0}^{q} \theta_j a_{t-j}, \tag{2.13}$$

siendo, de nuevo, constantes los parámetros y distintos de cero los correspondientes al último retardo, en cada caso.

Si se optimiza la ecuación, se obtiene su forma compacta:

$$\phi(B) X_t = c + \theta(B) a_t, \tag{2.14}$$

donde

$$\phi(B) = (1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p),$$
$$\theta(B) = (1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q)$$

y B denota al operador retardo definido como $BX_t = X_{t-1}$.

De este modo, estos modelos resultan estacionarios si el polinomio AR no presenta raíces de módulo unidad, lo que asegura también su causalidad. Además, son invertibles si las raíces del polinomio MA también están fuera del módulo unidad, lo que permite expresar los errores a partir de valores pasados de las series.

Como ampliación a estos procesos, cabe mencionar los modelos ARMA$(P, Q)_s$. Estos modelos están basados en la estructura hasta ahora detallada con una única diferenciación: la introducción de una

dependencia estacional. Esto quiere decir que, además de ser procesos estocásticos estacionarios de estructura mixta autorregresiva y de medias móviles, se ven afectados por una componente estacional s, que provoca la aparición de comportamientos repetitivos a lo largo del tiempo. Es decir, no son más que procesos ARMA(sP, sQ) con coeficientes nulos, excepto aquellos de múltiplo s. Su representación en forma compacta es la que sigue:

$$\Phi(B^s)X_t = c + \Theta(B^s)a_t, \tag{2.15}$$

donde $\Phi(B^s)$ y $\Theta(B^s)$ son polinomios estacionales y $B^sX_t = X_{t-s}$.

Ejemplos típicos son las series anuales que se ven afectadas por picos de subidas o bajadas en determinadas épocas del año, como por ejemplo datos sobre turismo, que alcanzan valores máximos en los meses de verano.

Por último, se resalta que, resultado de la combinación de los modelos ARMA(p, q) y ARMA$(P, Q)_s$, existen también los modelos ARMA$(p, q)\times(P, Q)_s$ estacionales multiplicativos, que permiten modelizar de forma conjunta la dependencia regular y estacional de las series, respectivamente. En forma compacta, la representación de estos procesos resulta:

$$\phi(B)\Phi(B^s)X_t = c + \theta(B)\Theta(B^s)a_t \tag{2.16}$$

2.3.3. Procesos no estacionarios

La metodología Box-Jenkins ofrece también recursos para modelizar procesos no estacionarios. Este tipo de procesos suele argumentar dicha falta de estacionariedad de acuerdo a varios motivos, como la presencia de tendencia en la serie, cambios en la variabilidad de la misma o aparición de dependencia temporal. Por ello, si una serie presenta alguna de estas características, ha de ser transformada de manera que se pueda modelar como un proceso estacionario de los ya vistos, siendo las transformaciones más comunes las siguientes:

- **Diferenciación:** se lleva a cabo cuando una serie de tiempo presenta tendencia a lo largo del tiempo. Para sustraer tal dependencia, se aplican d diferenciaciones regulares (normalmente $d \leq 3$) hasta que la serie logra que se estabilicen sus propiedades, convirtiéndose así en estacionaria. Una vez alcanzada la estacionariedad, esta puede ser modelizada como un proceso ARMA(p, q). Por ello, este tipo de procesos se denominan ARIMA(p, d, q) y se representan:

  $$\phi(B)(1 - B)^dX_t = c + \theta(B)a_t, \tag{2.17}$$

  donde d se refiere al número de diferenciaciones regulares aplicadas.

  Además, dado el caso en el que se detecte también que la serie presenta cierta componente estacional, es posible aplicar una diferenciación, en este caso, de tipo estacional, de orden D y periodo s, obteniéndose así los procesos denominados ARIMA$(p, d, q)\times(P, D, Q)_s$:

  $$\phi(B)\Phi(B^s)(1 - B)^d(1 - B^s)^DX_t = c + \theta(B)\Theta(B^s)a_t \tag{2.18}$$

- **Transformaciones Box-Cox:** por otro lado, puede ocurrir que la varianza de la serie dependa del nivel de la misma. En estos casos en los que la serie presenta heterocedasticidad, la herramienta más común para estabilizar la varianza y así lograr que la serie se vuelva estacionaria es aplicar a los valores de la serie una de las transformaciones Box-Cox:

  $$x_t = \begin{cases} \frac{x_t^\lambda - 1}{\lambda}, & \text{si } \lambda \neq 0, \\ \log(x_t), & \text{si } \lambda = 0, \end{cases}$$

  en los que el valor óptimo de λ se puede calcular a partir de los métodos definidos en Guerrero (1993): método de Guerrero o método de máxima verosimilitud.

Cabe destacar que estas transformaciones únicamente son aplicables a valores positivos, por lo que si una serie $\{x_t\}$ contiene valores nulos o negativos, estas transformaciones se llevarán a cabo sobre $\{y_t\} = \{x_t\} + c$, siendo c una constante que garantice la positividad de la misma. En concreto, la transformación logarítmica es de gran utilidad ante series en las que la varibilidad aumenta de forma directa con el nivel, ya que, al aplicar el logaritmo a los valores de la serie, la distancia entre puntos consecutivos disminuye según aumenta t. Además, es importante resaltar que este proceso será implementado siempre antes de proceder con cualquier tipo de diferenciación.

2.3.4. Identificación, estimación y diagnosis

Ahora bien, una vez entendida la naturaleza de los procesos vistos, ¿cómo se pueden identificar en la práctica? La respuesta radica en la interpretación de diferentes gráficos, que se comentará a continuación sin entrar demasiado en detalle, pues el tema central de este trabajo se enfoca en la evaluación de los modelos que se abordarán en el siguiente capítulo.

Primeramente, se debe comprobar la estacionariedad de la serie. Esto es, a partir de su gráfico secuencial, se debe observar si su nivel (valor medio) y variabilidad (varianza) permanecen constantes a lo largo del tiempo. Si esto no es así, se deben llevar a cabo las transformaciones oportunas detalladas previamente. Una vez cumplida esta premisa, el siguiente paso es identificar los órdenes del proceso a partir de los gráficos de la ACF y PACF frente al retardo.

En cuanto a los procesos no estacionarios, bastará con averiguar el número de diferenciaciones requeridas y, en caso de existir dependencia temporal, la periodicidad estacional.

A continuación, una vez identificado el posible proceso o procesos generadores de la serie, es necesario validar la elección de acuerdo a alguna métrica robusta, con el objetivo de garantizar la buena elección del mismo en términos de acierto y simplicidad del modelo. Para ello, previamente se definen r y T como el número de parámetros de cada modelo y el tamaño de la serie (número de observaciones), respectivamente. Además, $L(\hat{\psi})$ se corresponde con la función de verosimilitud que indica la probabilidad de que los datos de la misma hayan podido ser generados por el modelo en cuestión y $\hat{\psi}$ denota al conjunto de parámetros del modelo estimados. Así, se propone seleccionar el modelo que minimice alguna de las siguientes funciones:

- **Criterio de Información de Akaike:** AIC, por sus siglas en inglés. Se calcula:

$$AIC = -2\log(L(\hat{\psi})) + 2r \tag{2.19}$$

- **Criterio de Información de Akaike corregido:** AICc, por sus siglas en inglés. Su cálculo es el siguiente:

$$AICc = -2\log(L(\hat{\psi})) + 2(rT + r + 2)/(T - r - 2) \tag{2.20}$$

- **Criterio de Información Bayesiano:** BIC, por sus siglas en inglés. Calculado como:

$$BIC = -2\log(L(\hat{\psi})) + r\log(T) \tag{2.21}$$

Identificado correctamente el proceso generador de la serie, el siguiente paso consiste en estimar sus respectivos parámetros, tanto los coeficientes asociados a cada término como el valor de la varianza. Esta labor se puede llevar a cabo mediante métodos diferentes:

- **Mínimos cuadrados:** los valores estimados de los parámetros se obtienen de forma que minimicen la siguiente función:

$$S(\hat{\psi}) = \sum_{t=1}^{T} \hat{a}_t^2, \tag{2.22}$$

siendo $\hat{a}_t = x_t - \left(\tilde{c} + \tilde{\phi}_1 x_{t-1} + \cdots + \tilde{\phi}_p x_{t-p} + \tilde{\theta}_1 \hat{a}_{t-1} + \cdots + \tilde{\theta}_q \hat{a}_{t-q}\right)$.

- **Mínimos cuadrados condicionados:** es una corrección del método anterior, pues si $p > 0$, la obtención de las estimaciones de los valores de a_p depende de valores no observados de X_t. Por ello, se plantea la siguiente modificación de la función S:

$$S_C(\hat{\psi}) = \sum_{t=p+1}^{T} \hat{a}_t^2 \tag{2.23}$$

- **Máxima verosimilitud:** como se introdujo anteriormente, la función de verosimilitud indica la probabilidad de explicar los datos a partir de valores estimados de los parámetros del modelo. Por ello, este método consiste en estimar los valores de los parámetros de manera que se maximice dicha función:

$$L(\hat{\psi}) = \frac{1}{(2\pi\sigma_a^2)^{T/2}} \exp\left(-\frac{1}{2\sigma_a^2}\sum_{t=1}^{T} a_t^2\right), \tag{2.24}$$

  bajo el supuesto de que los residuos (errores) del proceso son normales e independientes.

Una vez seleccionado el modelo y estimados sus parámetros, la última parada antes de proceder con la predicción consiste en realizar un diagnóstico de validación del modelo, es decir, comprobar si el término de error corresponde a un proceso de ruido blanco (aleatoriedad). Esto es fundamental para poder aceptar el modelo sugerido, pues, en caso de no cumplirse, significaría que el modelo no habría capturado correctamente toda la estructura sistemática de la serie. Por ello, se realiza un análisis de residuos, en el que se verifican las hipótesis de media cero e incorrelación por contraste de hipótesis a través del t-test y Ljung-Box, respectivamente. Además, se suele evaluar también la normalidad de los residuos, tanto analíticamente mediante tests de normalidad como el de Shapiro-Wilk o Jarque-Bera, como gráficamente empleando gráficos Q-Q plot. Si bien las hipótesis de media cero e independencia son de obligado cumplimiento para poder realizar predicciones en base al modelo especificado, la ausencia de normalidad es más flexible, ya que los estimadores de máxima verosimilitud no perderían consistencia a pesar de que las innovaciones no fuesen gaussianas.

2.3.5. Valores atípicos y predicción

Otra casuística que puede invalidar el proceso estocástico seleccionado durante el análisis de residuos es la detección de valores atípicos (*outliers*). Un valor atípico se puede definir como aquella observación que se da en un instante de tiempo $t = h$ y que está lo suficientemente distante numéricamente de la estructura general de los datos como para influir en ella, por lo que, en la práctica, el objetivo recae en detectar estos valores y separarlos del modelo, de modo que esa división permita que el proceso sea validado como uno de los vistos a lo largo de la sección.

En cuanto a su naturaleza, se distinguen dos tipos de valores atípicos:

- **Aditivos:** el valor de la observación en dicho instante es generado de manera diferente al resto de la serie. Su presencia se puede modelizar de la siguiente manera:

$$Y_t = w_A I_t^{(h)} + \psi(B)a_t, \tag{2.25}$$

  donde $w_A I^{(h)}$ se refiere al peso asociado a la intervención y $\psi(B)a_t$ a X_t.

- **Innovativos:** en este caso, la perturbación no solo afecta a la observación en dicho instante, sino al proceso subyacente, por lo que dicho efecto se ve reflejado en el resto de la serie a partir del retardo en cuestión. Así, la representación del mismo resulta:

$$Y_t = \psi(B)w_I I_t^{(h)} + \psi(B)a_t = \psi(B)(w_I I_t^{(h)} + a_t) \tag{2.26}$$

Esto así, en la práctica, un método ampliamente utilizado para detectar este tipo de valores es el test de Bonferroni. Internamente, este test calcula un valor crítico como el cuantil $z_{1-\frac{\alpha}{2T}}$ dado un nivel de significación α (típicamente 0.05), y lo compara con los valores resultantes de un estadístico de prueba para cada observación de los datos. Así, si se dan valores que exceden dicho nivel, aquel que tome el mayor valor será catalogado como valor atípico, aditivo o innovativo, según afecte o no a la serie. De nuevo, se iteraría este procedimiento en busca de más valores atípicos hasta el punto en que ya no resulten valores significativos.

Tras todos estos pasos, una vez el modelo ha sido validado, sus parámetros han sido estimados y los valores atípicos, en caso de haberlos, han sido "eliminados", se da paso a la obtención de predicciones. Es habitual obtener predicciones puntuales sobre el valor esperado, es decir, sobre la media, a partir de la información pasada de la serie y de acuerdo con el modelo seleccionado. Para ello, se debe establecer un horizonte h sobre el que se llevará a cabo la predicción, para que, una vez realizada, las predicciones obtenidas puedan ser comparadas con los valores reales esperados. De este modo, se puede comparar la potencia predictiva del modelo de acuerdo a distintas métricas de error como las detalladas a continuación.

Si se define n como el tamaño muestral de los datos e y_t e $\hat{y}_t$ describen, respectivamente, los valores reales y predichos de los datos, se obtienen los denominados residuos:

$$e_t = y_t - \hat{y}_t, \tag{2.27}$$

que representan los errores de predicción. Así, se presentan las siguientes medidas de error:

- **No escaladas:** se expresan mediante las unidades originales de los datos y dependen de forma directa de la magnitud de la serie, por lo que no permiten comparabilidad entre diferentes contextos. Entre ellas destacan:

  - **MAE:** Error Absoluto Medio (MAE, por sus siglas en inglés). Representa el promedio del valor absoluto de los residuos. Se define como:

    $$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2.28}$$

    Destaca por su sencilla interpretación, aunque una de sus limitaciones es que no penaliza fuertemente errores grandes. El predictor que lo minimiza es la mediana.

  - **MSE:** Error Cuadrático Medio (MSE, por sus siglas en inglés). Representa el promedio de los residuos al cuadrado, por lo que se define como:

    $$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}e_t^2 \tag{2.29}$$

    Al contrario que el MAE, el MSE penaliza mejor los errores grandes. Sin embargo, es mucho más sensible que el Error Absoluto Medio frente a valores atípicos.

  - **RMSE:** Raíz del Error Cuadrático Medio (RMSE, por sus siglas en inglés). Se define como la raíz cuadrada del MSE:

    $$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2} \tag{2.30}$$

    Sus ventajas y desventajas son similares a las del MSE, ya que resulta de aplicarle la raíz cuadrada. Con respecto al MAE, su interpretación es más laboriosa, y en este caso, el predictor que lo minimiza es la media.

- **Escaladas:** normalizan el error respecto a un valor de referencia, permitiendo así ser comparadas entre series con distintas escalas. Las más comunes bajo este contexto son:

- **MAPE:** Error Porcentual Absoluto Medio (MAPE, por sus siglas en inglés). Representa un porcentaje del valor real, y se define como:

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{e_t}{y_t} \right| \tag{2.31}$$

  Al expresar el error como un porcentaje, facilita su comparación antre distintas escalas, pero no está definido cuando $y_t = 0$. También se puede entender en términos de proporción si no se multiplica por cien.

- **MASE:** Error Absoluto Medio Escalado (MASE). Representa el MAE escalado por el error de un modelo basado en el valor anterior, lo que se conoce como naïve. Se define como:

$$\text{MASE} = \frac{\frac{1}{n} \sum_{t=1}^{n} |e_t|}{\frac{1}{n-1} \sum_{t=2}^{n} |y_t - y_{t-1}|} \tag{2.32}$$

  Al estar escalado respecto a un modelo naïve resulta útil para la comparación entre series. Según su valor sea menor o mayor que 1, indica que el modelo es mejor o peor que el naïve, respectivamente.

- **NRMSE:** Raíz del Error Cuadrático Medio Normalizado (NRMSE, por sus siglas en inglés). Representa el RMSE normalizado por una medida de la escala de los datos, que en este caso fue la media de los mismos. Se define como:

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{n} \sum_{t=1}^{n} e_t^2}}{\frac{1}{n} \sum_{t=1}^{n} y_t} \tag{2.33}$$

  Sigue la línea del MSE y RMSE en cuanto a utilidades, pero su comparación se puede ver afectada por la medida de normalización empleada.

Al existir cierta incertidumbre derivada de dicha estimación, es habitual acompañar las predicciones de intervalos de predicción, comúnmente al 80 % o 95 %, con el objetivo de mostrar la variabilidad de los errores. El método clásico empleado para obtenerlos se basa en la distribución asintótica del error de predicción, únicamente aplicable bajo el supuesto de normalidad. Alternativamente, ante la ausencia de normalidad, otra manera de obtener los intervalos de predicción es mediante el método de remuestreo *bootstrap*, sobre el que se puede profundizar en Chong y Choo (2011), pero que, básicamente, se basa en estimar de forma empírica la distribución del error mediante repeticiones muestrales.

2.4. Suavización exponencial

Otro enfoque clásico ampliamente utilizado en tareas de análisis y predicción de series temporales es el método de suavización exponencial, cuyas bases teóricas se desarrollan en Gardner (1985). A diferencia de la metodología Box-Jenkins, en la que se debía identificar un proceso subyacente generador de la serie interpretando diversos factores, esta técnica resulta conceptualmente más sencilla, pues basta con determinar si la serie en cuestión presenta tendencia y/o estacionalidad para aplicar según qué tipo de modelo.

Estos modelos se basan en estimaciones suavizadas de los valores observados de la serie, los cuales son ponderados de manera exponencial según el instante en que se producen. Esto es lo que se conoce como suavización exponencial, y a continuación se trata dicho concepto según los diferentes modelos existentes producto de esta metodología.

2.4.1. Suavización Exponencial Simple

Este primer enfoque fue introducido en el ámbito académico a finales de los años cincuenta por Robert Goodell Brown. Esta modelización fue pensada para predecir valores de procesos que no presenten ni tendencia ni componente estacional, es decir, procesos estocásticos estacionarios, por lo que es la más sencilla de las que se van a comentar. Continuando con la notación anterior de los procesos estocásticos ($\{X_t\}$), la fórmula general del modelo es la que sigue:

$$\hat{X}_{t+h} = \alpha X_t + (1-\alpha)\hat{X}_{t+h-1}, \tag{2.34}$$

siendo $\alpha \in (0,1)$ el parámetro de suavización y h el horizonte de predicción.

Alternativamente, esta formulación también puede ser expresada introduciendo una ecuación de nivel, lo que facilitará la comprensión de cara a los modelos siguientes más complejos:

$$\begin{aligned} \text{Nivel: } \ell_t &= \alpha X_t + (1-\alpha)\ell_{t-1}, \\ \text{Predicción: } \hat{X}_{t+h} &= \ell_t, \end{aligned} \tag{2.35}$$

para todo valor de t y h positivo y no nulo (esto se deberá cumplir para todos los modelos).

Así, ℓ_t representa la estimación suavizada del nivel actual de la serie, y la predicción del próximo valor se corresponde directamente con dicho nivel.

Se observa que, a la hora de generar predicciones, estos modelos otorgan una mayor relevancia a las observaciones más recientes, ya que les asigna un peso mayor con respecto a valores referentes a instantes anteriores, causando así un decrecimiento exponencial según se retrocede en el tiempo. El peso se rige de acuerdo al parámetro de suavización y, aunque este se puede establecer manualmente, en la práctica, generalmente, se estima su valor óptimo de acuerdo a algún método, como por ejemplo el MSE. Su elección es crucial, pues según su valor, el modelo puede ser más o menos sensible a cambios recientes, si dicho valor es mayor o menor, respectivamente.

Como detalle, es destacable la conexión de estos modelos con la metodología Box-Jenkins, pues un modelo de Suavización Exponencial Simple (SES) puede ser interpretado como un ARIMA(0,1,1) aplicado a la serie original diferenciada. Esta relación implica que la metodología SES captura correctamente una estructura MA tras eliminar la tendencia de la serie mediante una diferenciación ($d = 1$).

Sin embargo, estos modelos quedan notablemente limitados cuando la serie presenta alguna de las dos componentes mencionadas. Por ello, se proponen las siguientes extensiones del método para cada uno de los casos.

2.4.2. Suavización Exponencial Doble: Método de Holt

Este método fue sugerido por Charles C. Holt en 1957 como extensión al modelo SES. En él, se introdujo adicionalmente el término de la tendencia lineal, ofreciendo así una nueva alternativa para modelizar series generadas a partir de procesos cuyo valor medio no es constante a lo largo del tiempo. Para ello, el autor incorporó una nueva componente a la formulación (tendencia de la serie), resultando las siguientes ecuaciones:

$$\begin{aligned} \text{Nivel: } \ell_t &= \alpha X_t + (1-\alpha)(\ell_{t-1} + b_{t-1}), \\ \text{Tendencia: } b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}, \\ \text{Predicción: } \hat{X}_{t+h} &= \ell_t + hb_t, \end{aligned} \tag{2.36}$$

siendo $\beta \in (0,1)$ el parámetro de suavización, en este caso, de la tendencia. Al igual que α (parámetro de suavización del nivel), el valor de β puede ajustarse manualmente u obtenerse de manera empírica mediante criterios de optimización.

En este caso, se observa que la ecuación de nivel se ve ajustada por el nuevo parámetro de la tendencia, que, por su parte, estima la pendiente de la serie. De este modo, las predicciones a horizonte

h se entienden como proyecciones lineales del nivel corregidas por la tendencia estimada. Es por esto que este método es conocido también como método lineal de Holt.

Estadísticamente, esta modelización se puede asociar también con la metodología Box-Jenkins. Concretamente, su comportamiento es similar al de un modelo ARIMA(0,2,2) que actúa sobre la serie tras haber aplicado una doble diferenciación ($d = 2$). Es decir, permite trabajar con series no estacionarias sin requerimiento de modelización explícita de ruido blanco o autorregresión, lo cual no era posible a través de SES.

No obstante, el método de Holt también presenta una limitación clave, y es que su rendimiento predictivo se puede ver comprometido ante la presencia de patrones estacionales. Para poder abordar esta carencia, surgió el método de Holt-Winters, que sí permite añadir ambas componentes (tendencia y estacionalidad), como se expondrá en el siguiente apartado.

2.4.3. Suavización Exponencial de Holt-Winters

Como se ha introducido, los modelos de SES y Holt resultan insuficientes a la hora de capturar correctamente la estructura de los procesos si estos presentan, además de tendencia, estacionalidad. Por ello, Charles C. Holt desarrolló, a comienzos de la década de los años sesenta, una nueva extensión junto a Peter R. Winters: el método de suavización exponencial de Holt-Winters, en honor a sus nombres.

Esta modelización incorpora una nueva componente explícitamente estacional, pudiendo abarcar así un mayor abanico de series temporales, sobre todo aquellas cuya estacionalidad se muestra estable en el tiempo. Aún así, dependiendo de cómo sea dicha componente, se distinguen dos versiones diferentes del modelo:

- **Suavización Exponencial Aditiva:** a periodicidad de la componente estacional (m) permanece constante a lo largo del tiempo. Continuando con la notación hasta el momento, se presenta la formulación general del modelo:

$$\begin{aligned} \text{Nivel: } \ell_t &= \alpha(X_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}), \\ \text{Tendencia: } b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}, \\ \text{Estacionalidad: } s_t &= \gamma(X_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \\ \text{Predicción: } \hat{X}_{t+h} &= \ell_t + hb_t + s_{t+h-mk}, \quad k = \left\lfloor \frac{h-1}{m} \right\rfloor, \end{aligned} \tag{2.37}$$

  donde aparece un nuevo término para la nueva componente (estacional), con su respectivo parámetro de suavización $\gamma \in (0,1)$. Además, el índice del periodo estacional se ajusta según el número de ciclos completos transcurridos mediante la función suelo $\lfloor \cdot \rfloor$, que redondea hacia abajo al entero más próximo.

- **Suavización Exponencial Multiplicativa:** la estacionalidad es proporcional al nivel de la serie. En este caso, la formulación es análoga, pero intercambiando las combinaciones aditivas por productos (similar a lo que ocurría con los valores atípicos aditivos e innovativos en los modelos Box-Jenkins):

$$\begin{aligned} \text{Nivel: } \ell_t &= \alpha\left(\frac{X_t}{s_{t-m}}\right) + (1-\alpha)(\ell_{t-1} + b_{t-1}), \\ \text{Tendencia: } b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}, \\ \text{Estacionalidad: } s_t &= \gamma\left(\frac{X_t}{\ell_{t-1} + b_{t-1}}\right) + (1-\gamma)s_{t-m}, \\ \text{Predicción: } \hat{X}_{t+h} &= (\ell_t + hb_t)s_{t+h-mk}, \quad k = \left\lfloor \frac{h-1}{m} \right\rfloor, \end{aligned} \tag{2.38}$$

Como se puede observar, este último método ofrece un gran desarrollo que permite modelar series complejas. No obstante, su rendimiento puede verse deteriorado frente a cambios abruptos en la estructura de las series o, como ya se ha comentado, ante estacionalidades inestables. En esos casos, una alternativa más flexible puede ser la utilización de modelos ARIMA estacionales, especialmente si dicha estacionalidad es fuerte.

2.4.4. Estimación, validación y predicción

En la práctica, la manera de proceder para realizar un análisis predictivo de una serie temporal resulta más sencilla con respecto a lo expuesto acerca de la metodología Box-Jenkins. Primeramente, se debe seleccionar uno de los tres modelos detallados en función de la presencia (o no) de tendencia o patrones estacionales en la serie en cuestión.

A continuación, lógicamente, deben estimarse los parámetros necesarios según el modelo seleccionado. Al igual que en los modelos ARMA o ARIMA, dichas estimaciones no se realizan de forma arbitraria, sino a través de métodos de optimización sobre un subconjunto de entrenamiento. Los procedimientos empleados más comunes son métricas de error como el MSE o el MAE; máxima verosimilitud; o validación cruzada, que consiste en dividir la serie en un subconjunto de entrenamiento y otro de prueba, para posteriormente ajustar el modelo en el primero y evaluar su precisión en el siguiente. Para estimar el desempeño general del modelo se repite el proceso realizando diversas particiones del conjunto de datos.

Tras la estimación de parámetros, es fundamental inicializar las componentes del proceso (nivel, tendencia y/o estacionalidad). Esto se lleva a cabo, habitualmente, mediante procedimientos heurísticos. El nivel inicial (ℓ_0) se suele aproximar calculando la media de los primeros L valores de la serie; la tendencia inicial (b_0), como el promedio de las diferencias entre los valores medios de ciclos (conjuntos de instantes) consecutivos; y la estacionalidad inicial ($s_0, s_1, \ldots, s_{m-1}$), estimándola como la diferencia (modelo aditivo) o el cociente (modelo multiplicativo) entre cada observación y su nivel correspondiente en el primer ciclo. Además, se emplean medidas de error como el MAE, RMSE o MAPE para evaluar el ajuste del modelo.

Finalmente, una vez ajustado y validado el modelo, es posible dar paso a la última etapa: la predicción temporal, que se lleva a cabo a través de la formulación específica de cada modelo. Además, estas predicciones pueden acompañarse también de intervalos de predicción, suponiendo una distribución probabilística de los errores, lo cual permite cuantificar la incertidumbre derivada del pronóstico realizado.

Capítulo 3

Modelos basados en aprendizaje automático

Una vez estudiada y comprendida la metodología clásica de predicción de series temporales a través de los modelos Box-Jenkins y el método de suavización exponencial, conviene dar paso a una alternativa de inferencia ampliamente desarrollada desde principios del siglo XXI: el aprendizaje automático o *machine learning* (véase Hastie et al. (2009)).

Para ello, el capítulo comienza con una introducción general de dicha metodología en su primera sección, para posteriormente centrarse en uno de sus subcampos más relevantes: el aprendizaje profundo o *deep learning*, desarrollado en la Sección 3.2. Por último, en la última sección del capítulo se abordará un tipo particular de modelos de aprendizaje profundo: los modelos basados en arquitectura Transformer, foco del Capítulo 4.

3.1. Definición y metodología

A diferencia de los métodos tradicionales expuestos en el anterior capítulo, las diferentes técnicas de aprendizaje automático no asumen una estructura paramétrica específica de las series temporales, es decir, no dependen de hipótesis estrictas acerca de la forma funcional de los procesos que las generan. Por contra, como su propio nombre indica, "aprenden" de los propios datos disponibles. Esto quiere decir que, a partir de patrones capturados de los propios datos (relaciones, regularidades o subestructuras, entre otros), se construyen y ajustan modelos que permiten realizar predicciones mejoradas o tomar decisiones sin necesidad de intervención externa. Ahora bien, ¿cómo se lleva a cabo este proceso?

Primeramente, se debe dividir el conjunto de datos en dos subconjuntos: entrenamiento y test. Esta división debe realizarse con cautela, pues los datos de entrenamiento deben reflejar una correcta representación del comportamiento real del conjunto total, de manera que, tras desarrollar los modelos en cuestión, se pueda evaluar el rendimiento del aprendizaje modelado sobre la muestra de prueba. Por ello, a partir del tamaño muestral del que se parta, se debe encontrar un balance equilibrado y razonable en el que se disponga de suficientes datos para entrenar modelos robustos y, a la vez, disponer de un conjunto representativo sobre el que evaluar la eficacia del modelo. Lógicamente, en el caso de que los datos sean de tipo temporal, se debe respetar el orden cronológico.

Además, en ocasiones se añade un tercer subconjunto, denominado conjunto de validación, para evitar el sobreajuste de parámetros sin comprometer la evaluación final sobre los datos de prueba. Así, tras la división de los datos, se plantean modelos de aprendizaje que permiten obtener predicciones sobre el conjunto de entrenamiento. Estas predicciones son comparadas con los valores reales, y el fin del entrenamiento consiste en minimizar una función de pérdida que indique el grado de discrepancia entre los valores predichos y los reales, cuantificando el error cometido. En el contexto de series temporales, algunas de las funciones de pérdida más comunes son el MSE, MAE o MAPE. El objetivo del

entrenamiento es encontrar un conjunto de parámetros que minimice la función de pérdida, y para ello se suele emplear un método iterativo denominado "gradiente descendente" (detallado en IBM (s.f.)):

1. Se calcula el gradiente (vector de derivadas parciales) de la función de pérdida con respecto a cada parámetro del modelo, el cual indica la magnitud y dirección del reajuste de cada parámetro para que se reduzca el error.

2. Se actualiza el valor del parámetro de acuerdo a la fórmula siguiente:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta J(\theta^{(t)}), \tag{3.1}$$

   donde $\theta^{(t)}$ es el vector de parámetros en la iteración t, $\nabla_\theta J(\theta)$ denota al gradiente de la función de pérdida empleada y η (tasa de aprendizaje) acostumbra a ser un valor positivo y relativamente pequeño que cuantifica la convergencia de los parámetros en cada iteración.

3. Se repite el proceso hasta que la función de pérdida converge a un valor mínimo, se alcanza un número máximo de iteraciones o la mejora del error resulta insignificante, obteniéndose así los valores óptimos de los parámetros necesarios del modelo en cuestión.

Cabe señalar que el algoritmo iterativo utilizado depende, además de la estructura del modelo, del tipo de aprendizaje que se esté empleando. En el caso del aprendizaje profundo, epicentro del trabajo que se desarrollará en la siguiente sección, es habitual utilizar el descenso de gradiente, debido a que los modelos implican parámetros continuos diferenciables. Sin embargo, el ámbito del aprendizaje automático engloba una gran variedad de enfoques con características y objetivos distintos que requieren de estrategias de entrenamiento diferentes. A continuación, previo paso a adentrarse en el *deep learning*, se exponen brevemente los principales tipos de aprendizaje automático:

- **Aprendizaje Supervisado:** consiste en una técnica de *machine learning* en la que el entrenamiento de modelos se lleva a cabo a través de datos etiquetados, es decir, que para cada dato de entrada, se conoce su salida o respuesta esperada. Aquí se enmarcan tanto las tareas de predicción, donde se busca predecir valores numéricos continuos, como de clasificación, donde se asignan categorías a los datos.

- **Aprendizaje No Supervisado:** al contrario que el aprendizaje supervisado, este enfoque trata con datos no etiquetados. Esto quiere decir que los datos de entrada no disponen de etiquetas de salida, por lo que éstas no pueden ser predichas. Así, los modelos únicamente tratan de entender su estructura subyacente para poder identificar anomalías, grupos similares (*clustering*) o mejorar la visualización (representación) de los mismos mediante técnicas de reducción de la dimensión, como las que se desarrollarán en la Sección 4.4.3.

A su vez, dentro de los diferentes tipos de aprendizaje automático expuestos, existe también una amplia variedad de modelos aplicables a la predicción de series temporales: árboles de decisión, máquinas de soporte vectorial o modelos basados en redes neuronales, entre otros. De cara a la aplicación práctica desarrollada en el Capítulo 4, se hará hincapié en los modelos basados en redes neuronales profundas. Por ello, en la siguiente sección se introducen los fundamentos de las redes neuronales, base sobre la que se sustenta el *deep learning*.

3.2. Deep Learning: redes neuronales

Esta sección está dedicada al entendimiento del funcionamiento interno del aprendizaje profundo, con el objetivo de aclarar las bases teóricas de los modelos que se presentarán en el siguiente apartado, y para ello, resulta fundamental comenzar explicando qué son las redes neuronales.

Las redes neuronales son un tipo de modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano, -de ahí su nombre- pues replican la forma en que los humanos tomamos

decisiones gracias a nuestras neuronas. Toda red neuronal está formada por un conjunto de nodos denominados neuronas, y consta de tres capas: una primera capa de entrada que recibe los datos, una o varias capas ocultas que procesan la información mediante transformaciones matemáticas y una capa final de salida que devuelve el resultado esperado según la técnica aplicada. Por ejemplo, en un problema de clasificación, esta última capa devolvería una categoría asociada a un dato de entrada, mientras que si se está ante un problema de regresión, como será el caso, la capa de salida entregaría una predicción como un valor continuo. En la Figura 3.1 se muestra el esquema básico descrito:

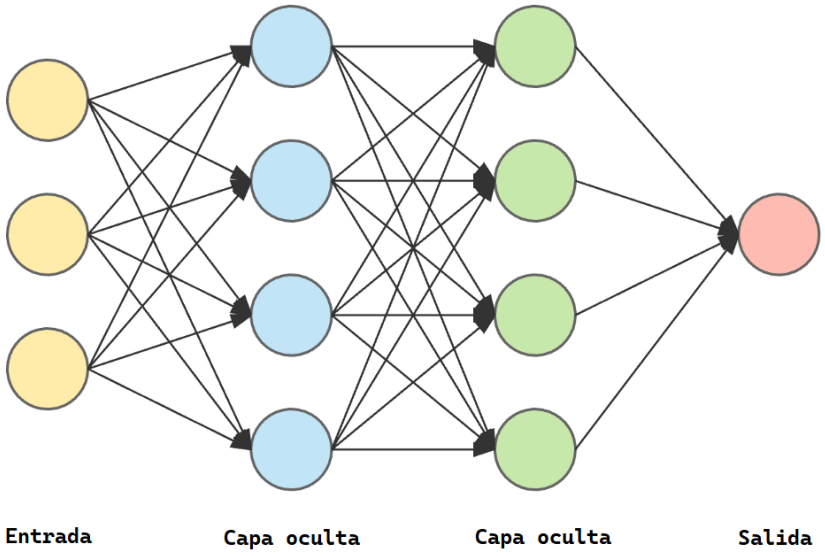

Figura 3.1: Estructura básica de una red neuronal. Imagen extraída de Huet (2023).

De este modo, partiendo de un conjunto de datos con n observaciones, estas son recibidas por n neuronas en la capa de entrada de la red, de manera que cada nodo recibe una unidad de dato. Todos los nodos de la capa de entrada están conectados a la primera, o única, capa oculta. Así, una vez los datos se encuentran en las neuronas de entrada, "viajan" a través de estas conexiones hasta la siguiente capa, que recibe la información tras haber sido multiplicada por un factor de peso que indica el grado de importancia de cada conexión. Tras ello, cada neurona de la capa oculta calcula la suma de la información recibida, de manera que , matemáticamente, el valor que cada neurona j recibe (z_j), se expresa de la siguiente forma:

$$z_j = \sum_{i}^{n} w_{ij} x_i + b_j, \tag{3.2}$$

donde x_i hace referencia a la información proveniente de la neurona i de entrada ponderada por un peso w_{ij}. Además, se añade una componente adicional de sesgo b_j, cuyo valor permite la activación de las neuronas ante cualquier valor de entrada, incluso nulos, aumentando así su flexibilidad.

Antes de volver a enviar la información obtenida y transformada a la siguiente capa, ya sea oculta o de salida, ha de tenerse en cuenta un factor: la linealidad. Si directamente se replicase el proceso, surgiría una gran limitación a la vista del aprendizaje de la red, pues esta resultaría básicamente una combinación lineal de funciones. De ser así, e independientemente del número de capas ocultas presentes, la red únicamente podría aprender patrones lineales, lo que, en la práctica, resulta realmente pobre. Por este motivo, para que la red neuronal pueda capturar todo tipo de relaciones complejas, se aplica una función de activación sobre z_j:

$$a_j = f(z_j), \tag{3.3}$$

siendo ahora a_j el valor que se envía a la siguiente capa. Existen diferentes tipos de funciones de activación aplicables en las capas ocultas; no obstante, la más recomendada es la función *Rectified Linear Unit*. ReLU, por sus siglas en inglés, se define:

$$f(z_j) = \text{máx}(0, z_j) \tag{3.4}$$

Mediante la sencilla operación de anulación de valores negativos sustituyéndolos por ceros, ReLU evita la linealidad del modelo. En ocasiones, también se hace uso de ella en los nodos de la capa final ante problemas de regresión. Por otro lado, si el objetivo es clasificar los datos, las funciones de activación típicas son la sigmoide ($f(z_j) = \frac{1}{1+e^{-z_j}}$) y *softmax* ($f(z_j) = \frac{e^{z_j}}{\sum_{j=1}^{n} e^{z_j}}$), en caso de que la clasificación sea binaria o multiclase, respectivamente.

Ahora bien, para determinar el valor de los pesos y sesgos que afectan a las conexiones neuronales, se debe iniciar el entrenamiento de la red. Primeramente, se asignan valores pequeños, no simétricos y aleatorios a dichos parámetros, y se propaga la red hacia adelante (*forward pass*), es decir, los datos atraviesan la red desde su capa de entrada hasta la capa de salida. La salida obtenida se compara con los valores reales de los datos, obteniéndose así una medida del error cometido (a partir de alguna de las métricas estudiadas). A continuación, se lleva a cabo el proceso inverso, denominado retropropagación (*backpropagation*), mediante el que se actualizan los valores estimados de los pesos y sesgos de acuerdo a mecanismos de optimización como el del gradiente descendente, reduciéndose así el error. Reproduciendo este proceso varias veces, el modelo consigue aprender de los datos, de manera que se completa su entrenamiento y es posible obtener resultados precisos.

Además, cabe mencionar que, al tratarse de modelos con un alto grado de complejidad, puede alcanzarse un punto de sobreajuste de parámetros, conocido como *overfitting*. Este fenómeno se produce cuando el modelo aprende en exceso de los datos de entrenamiento, por ejemplo, capturando ruido o detalles irrelevantes, lo que a posteriori se traduce en una reducción de la capacidad de generalización sobre nuevos datos. Para sortear esta problemática, es común introducir técnicas de regularización en el proceso de entrenamiento, como restricciones ante pesos elevados (regularización L1 y L2), la detención temprana del entrenamiento (*early stopping*) de forma que se pueda evaluar el rendimiento del modelo sobre un conjunto de validación antes de que se produzca sobreajuste, o mecanismos como el *dropout*, que desactiva aleatoriamente algunas neuronas en cada iteración, reduciendo así la dependencia entre ellas.

Esto así, el aprendizaje profundo se define como el área del *machine learning* basada en la modelización de datos mediante redes neuronales con múltiples capas ocultas, denominadas redes neuronales profundas. Atendiendo a la configuración de este tipo de redes, existe una gran variedad de modelos de aprendizaje profundo de gran utilidad en tareas de predicción de series temporales. A continuación se detallan brevemente algunos de ellos:

- **Redes neuronales convolucionales:** CNN, por sus siglas en inglés. Están especializadas en tareas de procesado de imágenes o señales, ya que emplean filtros que permiten detectar estructuras espaciales como bordes, formas o caras.

- **Redes neuronales recurrentes:** RNN, por sus siglas en inglés. Están diseñadas para procesar datos secuenciales como el lenguaje natural (cadenas de texto) o el reconocimiento de voz. Para ello, procesan los datos (secuencias) en orden, manteniendo un estado oculto que se actualiza con cada entrada nueva.

- ***Long Short-Term Memory*:** LSTM, por sus siglas en inglés. Constituyen una variante mejorada de las RNN, ya que incorporan celdas de memoria con puertas de entrada, salida y olvido que permiten retener la información durante intervalos más largos.

- **Redes Generativas Antagónicas:** GAN, por sus siglas en inglés. Son un tipo de red neuronal profunda compuesta por dos redes que compiten entre sí para generar nuevos datos. Una de ellas, la generadora, crea nuevos datos similares a los de entrenamiento, mientras que la otra, denominada discriminadora, intenta distinguir estos datos de los reales.
- **Autoencoders:** AE, por sus siglas en inglés. Están diseñados para realizar tareas de aprendizaje no supervisado, como reducción de la dimensión o detección de anomalías. Su funcionamiento se basa en la interacción de dos bloques principales: un codificador y un decodificador. El codificador recoge los datos y los transforma en una representación latente de menor dimensión, para que, posteriormente, el decodificador trate de reconstruir la entrada original a partir de dichas representaciones.

A mayores, se encuentran los modelos, o redes, basados en arquitectura Transformer, los cuales se detallarán en mayor profundidad en la siguiente sección, pues su metodología se verá implementada en la aplicación práctica del Capítulo 4.

3.3. Modelos basados en Transformers

3.3.1. Definición

En el análisis de series temporales, un factor clave reside en capturar correctamente las dependencias entre observaciones a lo largo del tiempo, especialmente cuando se trabaja con grandes volúmenes de datos. En este contexto, los modelos RNN definidos previamente presentan restricciones importantes, pues sus redes procesan la información de forma estrictamente secuencial, lo que impide una eficiente paralelización durante el proceso de entrenamiento que ralentiza el aprendizaje, en el que cada paso depende del anterior. Aunque los modelos LSTM mejoran esta limitación, también enfrentan ciertas dificultades a la hora de capturar dependencias a largo plazo, ya que, muchas veces, el gradiente pierde información pasada durante la fase de retropropagación (desvanecimiento del gradiente). Por ello, se propone una nueva arquitectura de redes neuronales: la arquitectura Transformer.

3.3.2. Estructura

La arquitectura original de los Transformers está compuesta por un codificador (*encoder*) y un decodificador (*decoder*). Sin embargo, en tareas como la predicción de series temporales, se suele utilizar únicamente un codificador para introducir los datos en la red, y no un decodificador, pues el objetivo no consiste en generar una nueva secuencia de datos, sino en obtener predicciones de una pequeña partición de los mismos. El bloque codificador está compuesto por varias capas idénticas y apiladas, y cada una de ellas consta de dos componentes principales: un mecanismo de atención y una red neuronal *feedforward*.

Por un lado, el mecanismo de atención, denominado como *Scaled Dot-Product Attention* (atención escalada por producto escalar) en Vaswani et al. (2023), se basa en el concepto de autoatención (*self-attention*), que permite a cada elemento de una secuencia asignar distintos pesos al resto de elementos (incluido él mismo) en función de su relevancia contextual. Para lograrlo, el mecanismo genera tres representaciones distintas para cada elemento de la secuencia:

- ***Query*(Q):** representa la consulta que formula cada elemento sobre los demás, es decir, qué busca dentro del contexto. Se calcula como:

$$Q = W_q \cdot X \tag{3.5}$$

- ***Key*(K):** se entiende como una etiqueta que describe a cada elemento y permite evaluar si responde a la consulta. Se calcula como:

$$K = W_k \cdot X \tag{3.6}$$

- ***Value*(V):** contiene la información que será transmitida si el elemento resulta relevante según la comparación entre Q y K. Se calcula como:

$$V = W_v \cdot X \tag{3.7}$$

Inicialmente, X representa una matriz de entrada que contiene los datos de la secuencia original, de manera que cada fila se corresponde con un instante de tiempo y, a través de estas operaciones, se transforma en vectores ponderados por matrices de pesos aprendidas durante el entrenamiento ($\{W_q, W_k, W_v\}$). De este modo, se obtiene la siguiente ecuación del mecanismo de atención:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{3.8}$$

donde d_k se refiere a la dimensión de los vectores K, clave para escalar la magnitud de los productos y así evitar valores extremadamente grandes en la función.

Este cálculo devuelve una matriz de atención en la que los *values* resultantes se ven ponderados según la similitud entre las consultas y las claves, dando lugar a un modelo que se centra mayoritariamente en aquellos elementos con mayor relevancia. De esta manera, se consigue solucionar el problema del desvanecimiento del gradiente típico de los modelos LSTM.

Además, es necesario considerar el orden de los elementos en la secuencia, por lo que se introduce una codificación posicional que permite codificar temporalmente cada dato. Uno de los métodos más comunes en este contexto es la codificación sinusoidal, y en Vaswani et al. (2023) se detallan las siguientes funciones:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d}}\right), \end{aligned} \tag{3.9}$$

donde pos, i y d representan, respectivamente, la posición del dato en la secuencia, la dimensión dentro del vector de codificación posicional y la dimensión del modelo.

Además de esta mejora, los Transformers también permiten aplicar la paralelización en su aprendizaje, lo que, computacionalmente, resulta realmente beneficioso. Esto se logra gracias a la atención multi-cabeza (*Multi-Head Attention*), típica en los codificadores, que, básicamente, consiste en aplicar h mecanismos de autoatención en paralelo. En cada uno de ellos se proyecta X en un subespacio diferente, de modo que cada cabeza puede enfocarse en distintos aspectos de la secuencia:

$$\begin{aligned} head_i &= Attention(Q_i, K_i, V_i), \\ Q_i = W_q^{(i)}X, \quad K_i &= W_k^{(i)}X, \quad V_i = W_v^{(i)}X, \end{aligned} \tag{3.10}$$

y, tras ello, las salidas correspondientes se concatenan y se proyectan nuevamente a través de una capa lineal, sobre la que también se aplica una matriz de pesos aprendible para poder combinar la salida de todas las cabezas. En la Figura 3.2 se muestra el esquema correspondiente:

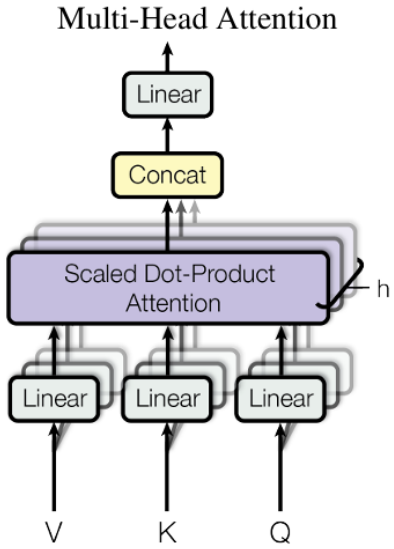

Figura 3.2: Representación básica del mecanismo de atención multi-cabeza de un Transformer. Imagen extraída de Vaswani et al. (2023).

Por otro lado, las redes neuronales *feedforward* se conectan de forma independiente a cada posición de la secuencia. Su objetivo radica en aplicar operaciones no lineales a las representaciones obtenidas por la capa de atención, como funciones ReLU, evitando así la linealidad y permitiendo que el modelo capture patrones complejos. De acuerdo a la documentación mencionada, el flujo de estas redes se formula:

$$FFN(X) = \max(0, XW_1 + b_1)W_2 + b_2, \tag{3.11}$$

siendo $\{W_1, W_2\}$ y $\{b_1, b_2\}$ matrices de pesos aprendibles y vectores de sesgo, respectivamente.

Adicionalmente, para evitar el *overfitting* durante el proceso de entrenamiento, es habitual incorporar técnicas de regularización como el *dropout*.

Para facilitar la fase de entrenamiento, cada subcapa del Transformer está conectada mediante una conexión residual, seguida de una normalización conocida como *Layer Normalization*. En Ba et al. (2016) se puede profundizar sobre ella, pero cabe destacar que su implementación mejora tanto la propagación del gradiente como su velocidad de convergencia.

Sobre el proceso de optimización, en este contexto aparece un optimizador hasta ahora desconocido, conocido como optimizador Adam (desarrollado en Kingma y Ba (2017)). Este mecanismo ajusta de manera dinámica las tasas de aprendizaje para cada parámetro y contribuye a alcanzar estabilidad y una alta velocidad de convergencia. En adición, se suele emplear como función de pérdida el MSE para minimizar el error de predicción del modelo.

Esta técnica supuso un hito en el modelado de series temporales. Al ofrecer la posibilidad de paralelizar el proceso, se logra procesar grandes cantidades de información a mayor rapidez computacional y evitando a su vez la pérdida de detalles, pues cada subproceso trabaja con un número reducido de dimensiones. Además, esto se traduce también en un mayor rendimiento del modelo, ya que cada cabeza, al aplicar ponderaciones de pesos distintas, consigue captar y aprender patrones complejos diferentes.

En definitiva, este mecanismo permite que cada entrada tenga una visión global de la secuencia completa, lo que supone una notable ventaja frente a los modelos RNN o LSTM, en los que la información fluye paso a paso.

3.3.3. Predicción de series temporales

Una vez comprendidas las ventajas de aplicar modelos basados en arquitectura Transformer a tareas de inferencia, en este último apartado teórico se exponen las principales características de esta metodología en el ámbito de la predicción de series temporales, como antesala a la presentación de un ejemplo de aplicación práctica desarrollado sobre un conjunto real de datos temporales.

En primer lugar, es importante señalar que la predicción de series temporales se puede abordar desde un enfoque univariante o multivariante, según el objetivo que se plantee.

El objetivo del enfoque univariado radica en predecir valores futuros de una única variable. En este contexto, las series se modelan única y exclusivamente a partir de su propio comportamiento histórico, es decir, no se tiene en cuenta información externa proveniente de variables exógenas u otras series temporales. Este tipo de configuración del modelo se denomina local, de manera que cada serie se modela con un Transformer independiente empleando distintos parámetros. Como posible desventaja, las predicciones obtenidas podrían no ser lo suficientemente robustas si el tamaño de las series es demasiado pequeño. Sin embargo, bajo un contexto adecuado, la modelización resulta directa, sencilla, especializada y no costosa computacionalmente.

En cambio, se puede tratar de predecir más de una variable o serie de forma simultánea, es decir, aplicar un enfoque multivariado. En este caso, los modelos se denominan globales, ya que entrenan un único Transformer que procesa varias series a la vez, de modo que todas ellas comparten los mismos pesos y parámetros. A pesar de que computacionalmente esto resulta más costoso que si se entrenan modelos locales y existe cierto riesgo de sobreajuste, la principal ventaja es que resulta posible capturar relaciones cruzadas entre series o efectos significativos de información externa.

En cuanto al proceso de entrenamiento, existen también dos planteamientos, en función de cómo se estructuren los datos de entrenamiento.

Por un lado, cuando un modelo es entrenado desde cero, el método más común consiste en aplicar ventanas deslizantes. Para ello, primeramente se define un tamaño de ventana fijo v y un horizonte de predicción h, y una vez definidos, se recorre la serie punto a punto o en varios pasos extrayendo dichas secuencias, de manera que la repetición del proceso da lugar a múltiples muestras de entrenamiento. Así, dada una serie temporal de longitud n, se podrían generar $n-v-h+1$ ejemplos de entrenamiento. Este tipo de entrenamiento resulta especialmente útil cuando se trabaja con series que se ven afectadas por distintos patrones en distintas fases temporales, y es ampliamente utilizado por modelos *DeepAR* y *TemporalFusionTransformer* (TFT).

Por otro lado, el entrenamiento se puede llevar a cabo de manera más sencilla, de forma que el modelo recibe como entrada el conjunto de entrenamiento completo y genera predicciones de los siguientes h puntos (conjunto de prueba). Por ello, en estos casos es fundamental una correcta partición del conjunto de datos, de modo que el comportamiento de las series entrenadas sea extrapolable de cara a la obtención de los valores predichos. Como ventaja, se destaca la simplicidad, rapidez y consistencia temporal para toda la secuencia de datos, y su implementación es típica de modelos preentrenados, como los modelos Chronos, que se evaluarán en el siguiente capítulo.

Con respecto a la obtención de predicciones, en muchas ocasiones, este tipo de modelos genera sus salidas en forma de distribuciones, en lugar de una única predicción puntual por cada instante temporal. Por ello, es habitual hacer uso del cálculo de cuantiles para devolver una única predicción por punto. De esta manera, se suele proporcionar una predicción puntual representativa calculada como la mediana, y, además, esta suele ser acompañada por intervalos de predicción que indican la incertidumbre inherente al proceso. Así, especificando un nivel de confianza $1-\alpha$, donde α representa la probabilidad de que la predicción quede excluida del intervalo, una forma típica de representar estas predicciones consiste en ofrecer un valor puntual acompañado por un intervalo de predicción que marca el rango dentro del que se espera encontrar el valor futuro, con una probabilidad de $1-\alpha$. Dicho esto, la expresión correspondiente se formaliza del siguiente modo:

$$\begin{aligned} &\text{Predicción puntual: } q_{0,5}, \\ &\text{Intervalo de predicción al } (1-\alpha)\,\% : (q_{\frac{\alpha}{2}}, q_{1-\frac{\alpha}{2}}), \end{aligned} \tag{3.12}$$

donde q_p hace referencia al cuantil de orden p de la distribución de valores predichos por el modelo.

Por último, cabe mencionar que, al igual que todos los tipos de modelos expuestos hasta el momento, en este contexto también resulta efectivo representar el error de predicción cometido de acuerdo a diferentes métricas, como las definidas en el apartado 2.3.5.

Con esto, tras haber expuesto en detalle los fundamentos teóricos sobre los diferentes modelos de predicción de series temporales, se procede a presentar los principales resultados obtenidos en el estudio práctico llevado a cabo en *SDG Consulting España, S.A.*

Capítulo 4

Aplicación práctica en Python

Tras haber profundizado sobre la naturaleza de las series temporales, los diferentes modelos clásicos y tecnologías más modernas de aprendizaje que permiten su análisis y predicción, en este capítulo se exponen todos los detalles del trabajo elaborado durante mi estancia en *SDG Consulting España, S.A.*

El estudio llevado a cabo consistió en la valoración de los modelos preentrenados de la librería Chronos, basados en arquitectura Transformer, a través del software Python. Para ello, el trabajo se dividió en tres etapas diferenciadas: en primer lugar, se realizó un análisis exploratorio inicial del conjunto de datos utilizado para poder conocer sus características más relevantes. A continuación, a partir de los modelos Bolt, los cuales se detallarán a lo largo del capítulo, se realizaron predicciones de los valores de la variable objetivo para su posterior evaluación en comparación con los valores reales, de acuerdo a diferentes métricas. Finalmente, se calcularon representaciones vectoriales de las series mediante *embeddings*, con el fin de identificar patrones repetitivos y diferencias entre las mismas.

Esto así, el objetivo principal de este capítulo se resume en comprobar la utilidad y rendimiento de los modelos Chronos en el marco de las series temporales, identificando sus ventajas e inconvenientes en comparación con los modelos clásicos.

4.1. Análisis exploratorio inicial

4.1.1. Conjunto de datos

El conjunto de datos empleado en este trabajo, proporcionado por la empresa colaboradora, consiste en información relacionada con el volumen de ventas de productos en tiendas Walmart, efectuadas en el marco temporal comprendido entre el 29 de enero de 2011 y el 22 de mayo de 2016 (1.941 días). Según se recoge en Statista Research Department (2024), Walmart, fundada en Estados Unidos, "es la mayor corporación multinacional de supermercados y almacenes de descuento en el mundo".

La información recogida en dicho conjunto de datos se muestra a través de las siguientes siete variables:

- **date:** fecha en formato AAAA-MM-DD.
- **ID:** identificador único de cada serie.
- **Num_Pedidos:** número de pedidos realizados. Es la variable objetivo sobre la que se llevaron a cabo las tareas de predicción. Como excepción, en la Tabla 4.1 se denota como "N_Ped" para reducir el ancho de la tabla y respetar los márgenes del documento.
- **Nivel:** jerarquía de la serie. Existen cuatro niveles jerárquicos: **total** (1), **state** (estado: 2), **store** (tienda: 3) y **dept** (departamento: 4).

- **state_id:** indica el estado en el que se ubica cada tienda (Nivel 2): **CA** (California), **TX** (Texas) o **WI** (Wisconsin).
- **store_id:** denota el número de tienda dentro de cada estado (Nivel 3). En el estado CA existen cuatro tiendas, mientras que TX y WI cuentan con tres cada uno.
- **dept_id:** se refiere al tipo de departamento dentro de cada tienda, según el tipo de producto que venden (Nivel 4). Existen tres departamentos para la categoría **FOODS** (alimentación), dos para **HOBBIES** (pasatiempos) y otros dos para **HOUSEHOLD** (hogar).

Así, dependiendo del nivel jerárquico elegido, se puede representar la información de varias maneras: desde una única serie total (nivel 1), hasta 3, 10 o 70 series diferentes, correspondientes al segundo, tercer y cuarto nivel, respectivamente, por lo que el conjunto de datos cuenta con 84 series temporales.

Para ilustrar esto y comprenderlo mejor, en la Tabla 4.1 se muestra la información asociada a una observación particular de una serie temporal de nivel 4:

date	**ID**	**N_Ped**	**Nivel**	**state_id**	**store_id**	**dept_id**
2016-05-20	CA_CA_2_FOODS_1	584	4	CA	CA_2	FOODS_1

Tabla 4.1: Ejemplo de observación temporal del conjunto de datos.

4.1.2. Distribuciones: representación gráfica

Una vez entendida la estructura y características de los datos, se ahondó sobre su distribución. En concreto, se graficó la distribución de las ventas, es decir, del número de pedidos, según el estado y, posteriormente, por tienda. Esto se llevó a cabo mediante los siguientes diagramas de caja:

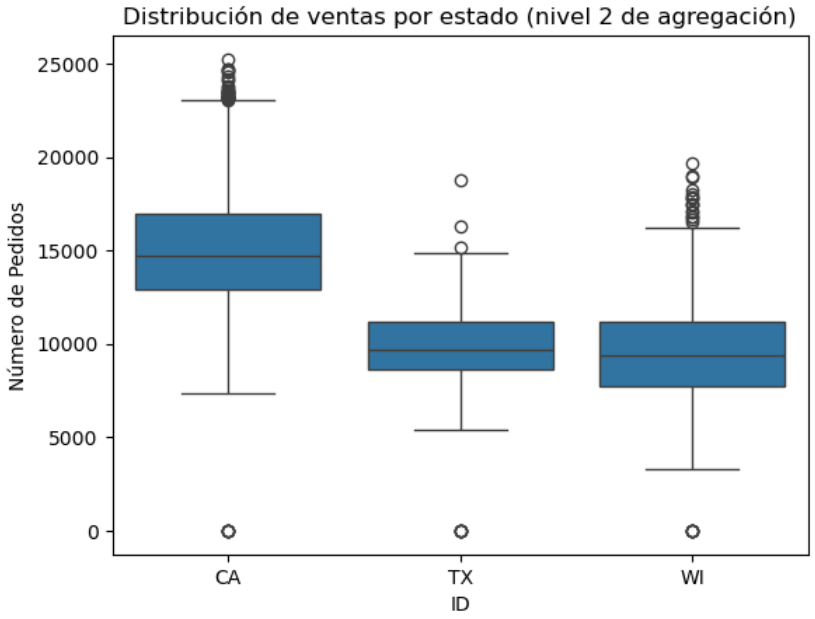

Figura 4.1: Distribución del número de pedidos en los 3 estados.

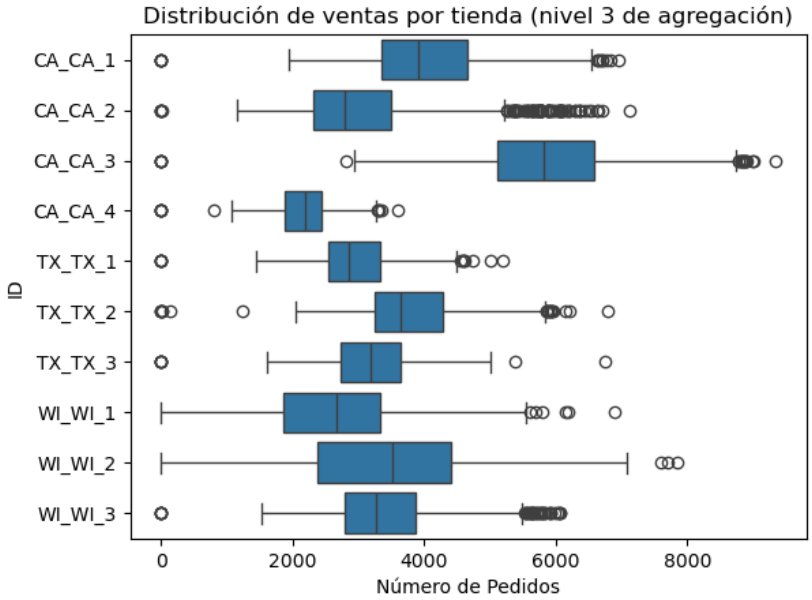

Figura 4.2: Distribución del número de pedidos por tienda.

A la vista de los *boxplots*, se pueden extraer ciertas conclusiones sobre los datos.

En primer lugar, atendiendo a la Figura 4.1, se observa una clara diferencia entre la distribución de las ventas en el estado de California con respecto a Texas y Wisconsin, que indica que en este primer estado se produce un mayor volumen de las mismas.

Además, aparecen valores atípicos en ambos extremos de las distribuciones. Aquellos que indican un número de pedidos igual a 0, observable también en todos los casos de la Figura 4.2, se corresponden, como se ha podido comprobar, a los días de Navidad, lo que se atribuye a que Walmart no opera todos los años el día 25 de diciembre.

Por otro lado, en cuanto a los *outliers* correspondientes al extremo superior de las distribuciones, no se ha observado ningún patrón que los explique, pues se supuso que vienen dados por diferentes motivos que dependen de la ubicación de las tiendas, tipo de productos o días festivos, entre otros.

4.1.3. Análisis temporal

Por último, antes de adentrarse en el *grosso* del trabajo, se previsualizó también el comportamiento de las series, representándolas en función del número de pedidos a lo largo de los años. Además, al igual que en el primero de los gráficos anteriores, esta información se mostró desglosada a nivel 2 (por estado).

Como se puede observar en la Figura 4.3, la evolución temporal del número de pedidos sigue un patrón estacional anual con una ligera pendiente positiva, lo que indica un buen rendimiento de la cadena. En adición, se corrobora que en el estado de California es donde se realiza una mayor cantidad de pedidos, y se observa claramente el cierre de la empresa al final de cada año.

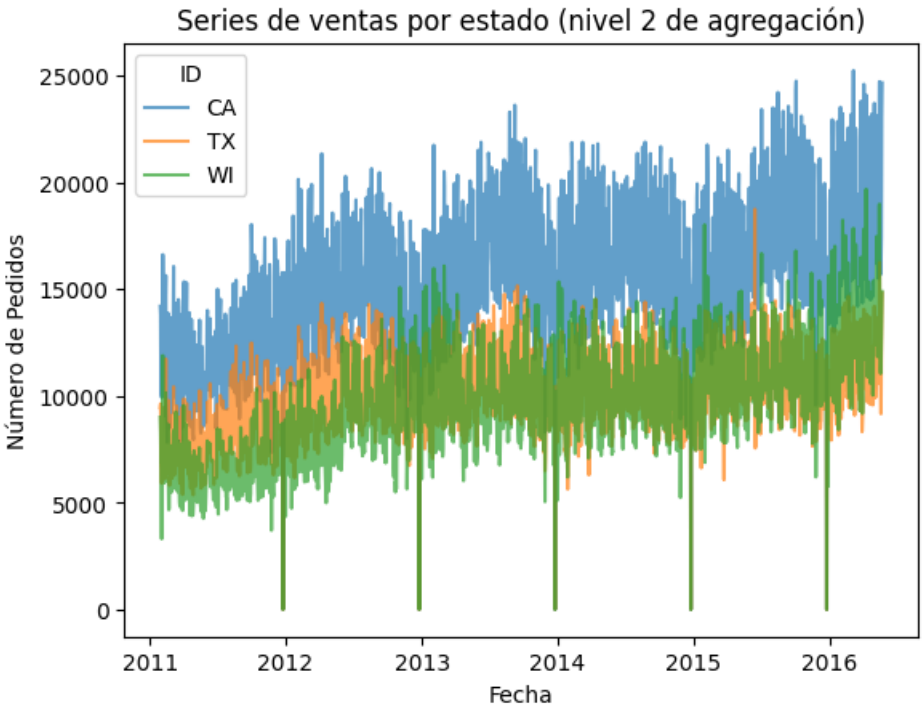

Figura 4.3: Evolución del número de pedidos en los 3 estados.

4.2. Chronos

Una vez introducido y comprendido el conjunto de datos empleado en este trabajo, conviene dedicar otro apartado a los modelos utilizados en el mismo.

Los modelos seleccionados para el desarrollo de este proyecto, y para la evaluación de los mismos, fueron los modelos Chronos de Amazon Science. Chronos es una familia de modelos preentrenados de predicción de series temporales basados en arquitecturas de modelos de lenguaje. Concretamente, los modelos de Chronos están basados en la arquitectura Transformer T5 (Text-To-Text Transfer Transformer) desarrollada por Google Research. T5 unifica múltiples tareas de procesamiento del lenguaje natural (PLN) bajo un mismo enfoque de conversión de texto a texto, lo que le permite adaptarse a distintos escenarios como la predicción de series temporales. Para su desarrollo, T5 se preentrena en un gran corpus de datos textuales (por lo que no requiere ajuste de hiperparámetros): C4 (Colossal Clean Crawled Corpus), que consiste en una gran colección de texto extraída de la web, filtrada y limpiada para garantizar la calidad de los datos.

4.2.1. Arquitectura

El proceso que siguen estos modelos para obtener predicciones de las series temporales es el que se observa en la Figura 4.4:

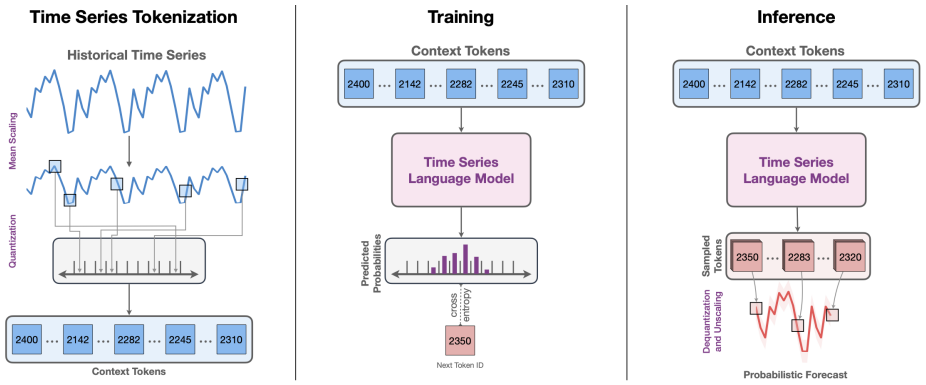

Figura 4.4: Representación de alto nivel de la arquitectura de los modelos Chronos. Imagen extraída de Ansari et al. (2024).

Primeramente, la serie de tiempo histórica es sometida a dos procesos de preprocesamiento: escalado y cuantificación. La primera fase de escalado se lleva a cabo calculando la media de los valores absolutos de cada serie, normalizando las observaciones para así mejorar la estabilidad numérica y preservar la estructura de la misma, incluso ante valores nulos o próximos a cero. A continuación, estos valores escalados son cuantificados: se transforman en categorías discretas mediante su asignación a intervalos definidos, denominados *bins*, mediante cuantiles, lo que permite representar la serie como una secuencia de *tokens* (unidades mínimas de texto). Una vez realizada esta transformación, los *tokens* se introducen en el modelo, que es entrenado mediante la pérdida de entropía cruzada, empleando un algoritmo de optimización que ajusta los parámetros minimizando una función de pérdida basada en la diferencia entre la distribución de probabilidad real y la predicha por el modelo. Tras ello, se obtienen nuevos *tokens* a partir de muestras autorregresivas del modelo, a los que posteriormente se les asignan valores numéricos. Finalmente, se muestrean múltiples trayectorias para obtener la distribución predictiva de las series.

4.2.2. Modelos

En cuanto a la implementación de Chronos en este trabajo, se utilizaron dos tipos de modelos según sus características y tareas a realizar, que se detallan y diferencian a continuación:

Desde un principio, se hizo uso de los modelos Chronos "originales", basados en la arquitectura T5 ya mencionada. Según el número (en millones) de parámetros, se diferencian cinco modelos:

- **Tiny:** 8M
- **Mini:** 20M
- **Small:** 46M
- **Base:** 200M
- **Large:** 710M

resultando la única diferencia entre ellos el tiempo de ejecución necesario para entrenarlos, derivado del número de parámetros en cada caso (lógicamente, a mayor cantidad de parámetros, mayor tiempo de entrenamiento).

Además de estos modelos, Amazon Science desarrolló unos nuevos denominados Bolt, publicados el 26 de noviembre de 2024, un mes después del inicio de mi estancia en la empresa. Los modelos Bolt, al igual que los originales de Chronos, están optimizados específicamente de cara a la predicción de series temporales, motivo por el que superan a otros modelos más genéricos basados también en arquitectura

Transformer y adaptados a esta tarea. Su diseño permite capturar mejor patrones temporales complejos y, además, han demostrado un rendimiento robusto ante limitaciones comunes en la práctica, como series de poca longitud o la presencia de ruido.

Otra de las claves de estos modelos es la optimización del mecanismo de atención del Transformer en términos de reducción del coste computacional evitando la pérdida de la capacidad de representación. Por este motivo, los modelos Bolt ofrecieron notables mejoras en el desarrollo del trabajo, pues, con los mismos tamaños que los originales (salvo el modelo Large, que no existe como modelo Bolt), se logran reducir significativamente los tiempos de ejecución sin pérdida de precisión. Es más, estos modelos resultaron ser más precisos que los originales, pues reducen el error un 5 %. Estas mejoras se pueden observar en la Figura 4.5, donde se comparan los tiempos de ejecución de los modelos Bolt frente a los originales, y en la Figura 4.6, que muestra la Pérdida de Cuantiles Ponderada (WQL, por sus siglas en inglés) y el Error Medio Absoluto Escalado (MASE, por sus siglas en inglés), ambos en términos relativos acumulados, de catorce modelos distintos, entre los que se encuentran los Chronos-Bolt:

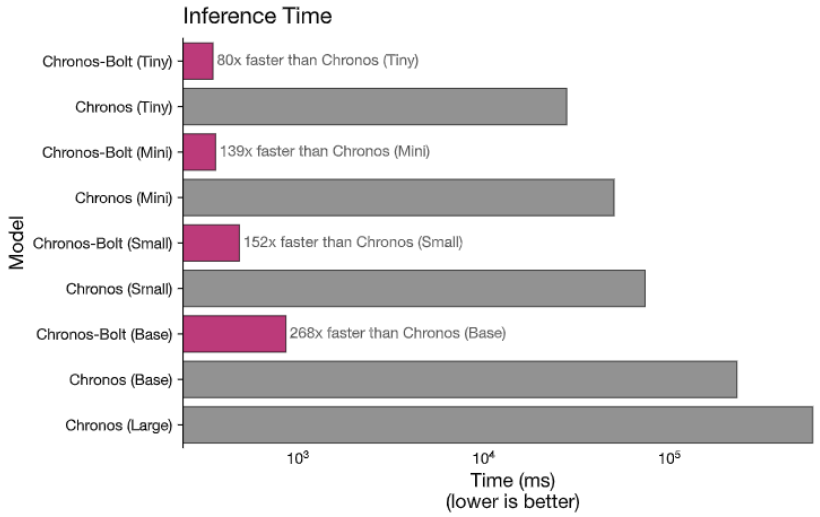

Figura 4.5: Tiempos de ejecución de los modelos Bolt frente a los originales de Chronos. Imagen extraída de Amazon Web Services (2024).

Figura 4.6: WQL y MASE relativo acumulado de los modelos Chronos-Bolt frente a otros. Imagen extraída de Amazon Web Services (2024).

Atendiendo a la imagen de la izquierda de la Figura 4.6, salta a la vista que los modelos Bolt se imponen de manera relevante frente al resto en cuanto al WQL relativo acumulado. El WQL consiste en una métrica de evaluación de la precisión de la capacidad predictiva en series temporales mediante la estimación de cuantiles, calculada como el cociente entre la suma total de errores de cuantiles, que penalizan de forma asimétrica según el nivel del cuantil, entre la suma de los valores absolutos reales de la serie en el horizonte de predicción, por lo que resulta dependiente de la escala. Además, se incluye un vector de pesos que pondera el horizonte de predicción, lo que permite ajustar la importancia relativa de las observaciones en el tiempo. Por ello, la familia Bolt de Chronos queda ensalzada en cuanto a rendimiento en la predicción de series temporales, objetivo central de este estudio.

En cuanto al gráfico de la derecha de la figura, se representa la misma comparativa pero en términos de MASE, métrica que se detallará más adelante, concretamente, en el apartado 4.3.4. Pero, aún así, ya se puede entrever una mejora observable de los modelos Bolt también en este apartado.

En adición, cabe decir que, debido a que estos modelos Bolt también están preentrenados bajo un enfoque "texto a texto", su uso no solo destaca en tareas de predicción, sino que también suponen una gran alternativa para clasificación de series o detección de agrupamiento, como se verá en la Sección 4.4. Todo esto los convierte en herramientas altamente versátiles y adaptables en escenarios reales.

4.3. Predicción de series temporales

Analizado el conjunto de datos y los modelos a utilizar, la siguiente fase consistió en evaluar el rendimiento de los mismos a la hora de generar predicciones de las series temporales del conjunto de datos.

4.3.1. División de los datos

Para ello, el primer paso consistió en dividir la muestra de datos en dos subconjuntos: uno de entrenamiento con la mayoría de los datos (primeros 1.926 días) y otro correspondiente a los últimos 15 días de la muestra. La elección de quince días se fundamentó en el *modus operandi* de la empresa en este tipo de pruebas para así, de cara a futuro, poder disponer de los resultados y realizar comparaciones entre modelos. Esto así, se crearon los siguientes intervalos temporales:

- **Entrenamiento:** 2011-01-29 a 2016-05-06
- **Test:** 2016-05-07 a 2016-05-22

4.3.2. Entrenamiento y obtención de las predicciones

Una vez dividido el conjunto de datos, se entrenaron los cuatro modelos Bolt existentes, uno a uno, sobre los datos de entrenamiento. Para ello, el procedimiento fue el siguiente: en primer lugar, se cargó en la CPU el modelo correspondiente desde *HuggingFace*, plataforma de código abierto que aloja una gran variedad de modelos de inteligencia artificial preentrenados, entre ellos los Chronos-Bolt. A continuación, se creó un bucle para todos los ID en el que, a partir de un contexto de tipo tensor que almacenaba todos los valores de la variable "Num_Pedidos" en cada serie, se generaron las predicciones de dicha variable sobre los datos de la muestra test. La función de la librería Chronos empleada para realizar las predicciones, por defecto, devuelve un abanico de nueve valores distintos por punto, por lo que, para lograr un único valor predicho para cada día, se calculó la mediana de esos nueve datos resultantes. Además, se calcularon también los valores correspondientes a los extremos del intervalo para un 80 % de confianza, a petición de la empresa, es decir, los percentiles 10 y 90, y los tiempos de entrenamiento resultantes. En la Tabla 4.2 se muestra un ejemplo, cuya observación temporal es la misma que la de la Tabla 4.1:

$x_{0,10}$	Predicción	$x_{0,90}$	Modelo	Tiempo (s)
501,9389	573,8492	659,3643	Bolt-Tiny	0,1181
464,0824	526,5287	602,2416	Bolt-Mini	0,2473
456,3083	531,5988	621,5079	Bolt-Small	0,6611
453,9423	535,8238	623,8738	Bolt-Base	3,3723

Tabla 4.2: Predicciones del número de pedidos a partir de los modelos Bolt (observación de la Tabla 4.1).

A la vista de la tabla, se pueden extraer varios indicios. En cuanto a la calidad de la predicción, que posteriormente se evaluará mejor según diferentes métricas, se observa un resultado muy similar al valor real en dicho punto (584 pedidos), lo que podría sugerir un buen rendimiento de los modelos a primera vista, que, sin embargo, tendría que ser profundamente estudiado para descartar que se trate de una "coincidencia" o caso aislado. Además, los tiempos de ejecución (entrenamiento) resultantes no superan los cuatro segundos en ningún caso, lo que termina de corroborar lo ya enunciado previamente sobre la rapidez de estos nuevos modelos Bolt mejorados. Como detalle, cabe destacar que, lógicamente, el tiempo aumenta de manera proporcional con el número de parámetros existentes en los diferentes modelos.

4.3.3. Representaciones gráficas

Antes de avanzar hacia dicha evaluación mediante métricas más robustas, se realizó una parada intermedia en la que se representaron gráficamente todas las series temporales junto con las predicciones resultantes de los modelos Bolt.

Como ya es sabido, el conjunto de datos cuenta con 84 series temporales y se trabajó con cuatro modelos diferentes. Esta combinación resultaría en una cifra demasiado elevada para presentar en este documento cada uno de los gráficos, por lo que, siguiendo con el mismo ejemplo que anteriormente, se mostrarán las gráficas correspondientes a la observación CA_CA_2_FOODS_1 , que se pueden observar en las Figuras 4.7, 4.8, 4.9 y 4.10:

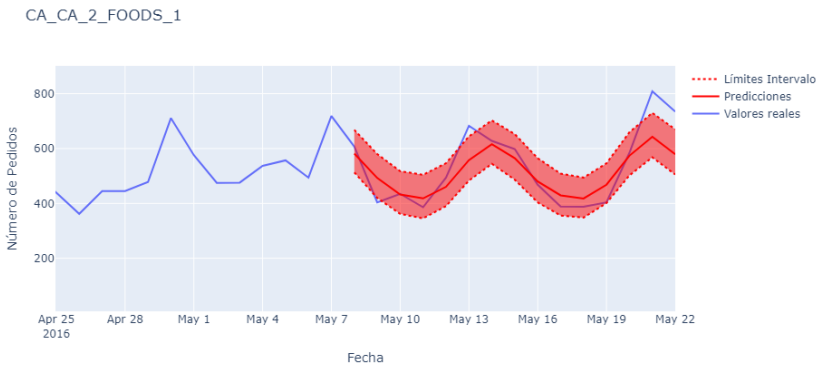

Figura 4.7: Ejemplo de representación gráfica de las predicciones del número de pedidos a partir del modelo Bolt-Tiny.

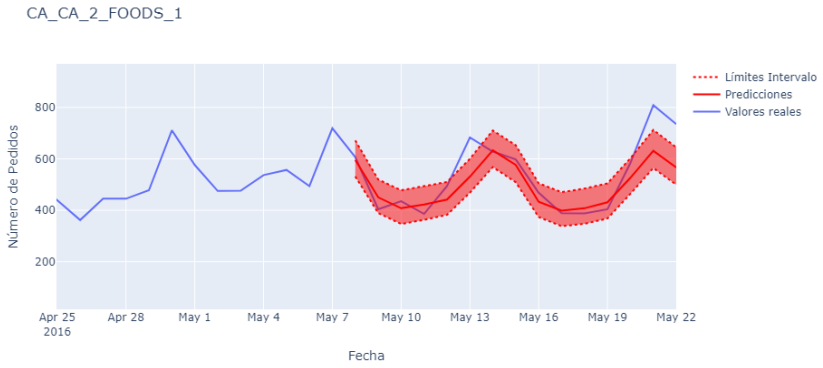

Figura 4.8: Ejemplo de representación gráfica de las predicciones del número de pedidos a partir del modelo Bolt-Mini.

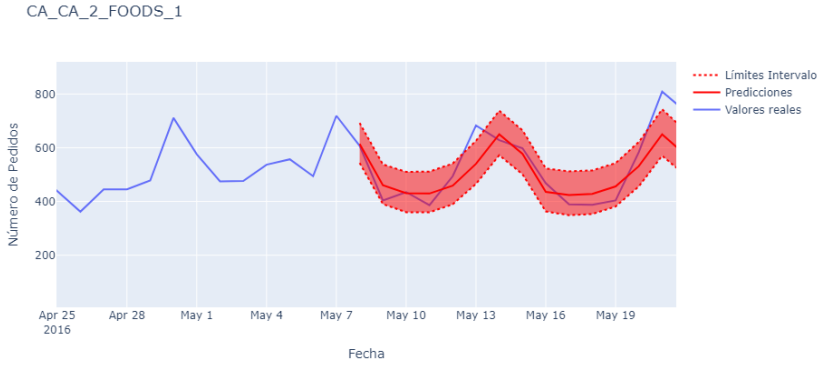

Figura 4.9: Ejemplo de representación gráfica de las predicciones del número de pedidos a partir del modelo Bolt-Small.

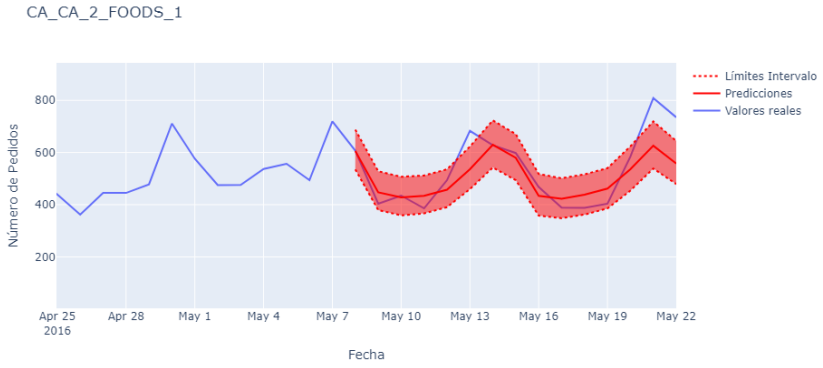

Figura 4.10: Ejemplo de representación gráfica de las predicciones del número de pedidos a partir del modelo Bolt-Base.

A la vista de los gráficos, se puede observar un aparente notable rendimiento de los modelos a la

hora de predecir los valores de las series temporales. Sin embargo, las figuras presentadas simplemente representan una pequeña muestra, pues no se está atendiendo a las 83 series restantes. Por lo tanto, para poder extrapolar dicha conclusión, en el apartado siguiente se mostrarán resultados referentes a métricas más precisas evaluadas sobre el conjunto de datos completo.

A la vista de los gráficos, se puede observar un aparente notable rendimiento de los modelos a la hora de predecir los valores de las series temporales. Sin embargo, las figuras presentadas simplemente representan una pequeña muestra, pues no se está atendiendo a las 83 series restantes. Por ello, es importante destacar que una evaluación de rendimiento basada única y exclusivamente en valores puntuales como estos puede resultar engañosa. Realmente, la precisión de la predicción debería evaluarse teniendo en cuenta otros factores como la variabilidad de las series, pues la magnitud del error cometido depende de la dispersión de los datos. Por lo tanto, para poder extrapolar una conclusión firme, en el siguiente apartado se presentarán resultados más detallados sustentados por métricas de error más precisas.

4.3.4. Rendimiento: métricas

Una vez calculadas las predicciones a quince días de horizonte de las 84 series temporales a través de los cuatro modelos, además de haber generado también todos los gráficos correspondientes, se procedió a evaluar la capacidad predictiva de Chronos de acuerdo con el error cometido en la misma.

Para ello, se realizó un estudio comparativo frente a otros quince modelos diferentes evaluados por la empresa interna y previamente bajo las mismas condiciones. Estos se dividían en seis versiones de modelos *DeepAR* (modelos autorregresivos desarrollados por Amazon basados en redes neuronales recurrentes), un modelo *N-BEATS* (desarrollado por Facebook y basado en una red neuronal profunda), un modelo *Prophet*, también desarrollado por Facebook, pero de tipo aditivo, y siete versiones de modelos TFT (de arquitectura Transformer y desarrollados por Google DeepMind), en los que las diferentes versiones diferían entre sí según la selección de hiperparámetros.

Dicho estudio se llevó a cabo mediante la comparación de siete métricas de error diferentes, sobre las que se puede profundizar en el apartado 3.4 de Hyndman y Athanasopoulos (2018). Seis de ellas ya fueron detalladas en la Sección 2.3.5, y, a mayores, se presenta otra utilizada para esta tarea:

- **MSIS:** Error Medio de Intervalo Escalado (MSIS, por sus siglas en inglés). Representa el error en la calidad de intervalos de predicción, combinando el ancho del intervalo, cuyos extremos inferior y superior son L_t y U_t, respectivamente, con otras dos penalizaciones por predicciones que no contienen el valor real. Estas penalizaciones se realizan según un nivel de confianza de los intervalos, por lo que es necesaria la especificación de un nivel de significación α. En este caso, el valor elegido fue de 0,05. Finalmente, este término es escalado, al igual que en el caso del MASE, según un modelo de referencia naïve:

$$\text{MSIS} = \frac{1}{n}\sum_{t=1}^{n}\left((U_t - L_t) + \frac{2}{\alpha}(L_t - y_t)\cdot \mathbf{1}_{\{y_t<L_t\}} + \frac{2}{\alpha}(y_t - U_t)\cdot \mathbf{1}_{\{y_t>U_t\}}\right) \Bigg/ \frac{1}{n-1}\sum_{t=2}^{n}|y_t - y_{t-1}| \tag{4.1}$$

  Evalúa la calidad de los intervalos de predicción según su amplitud y precisión, pero depende de un buen ajuste del modelo naïve y de la correcta calibración de los intervalos.

Una vez entendidas las métricas de error empleadas y sus características y utilidades, se presentan los resultados obtenidos del estudio comparativo entre los diecinueve modelos. Para su correcto entendimiento, cabe destacar que el estudio se realizó distinguiendo por nivel jerárquico de los datos y, además, el valor de cada error se calculó como la media de los errores obtenidos para las diferentes series dentro de cada nivel. Esto se contempla en las Figuras 4.11, 4.12, 4.13 y 4.14:

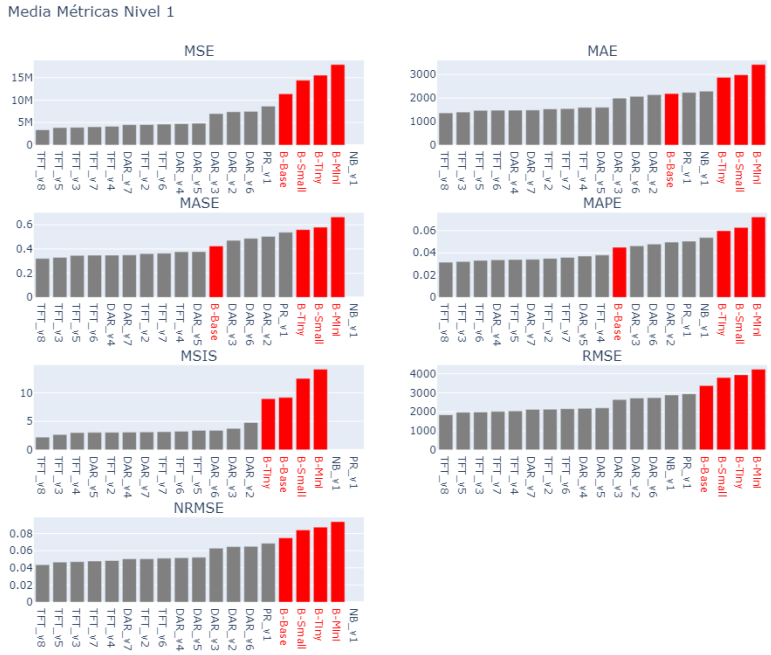

Figura 4.11: Comparativa de las métricas de error resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 1.

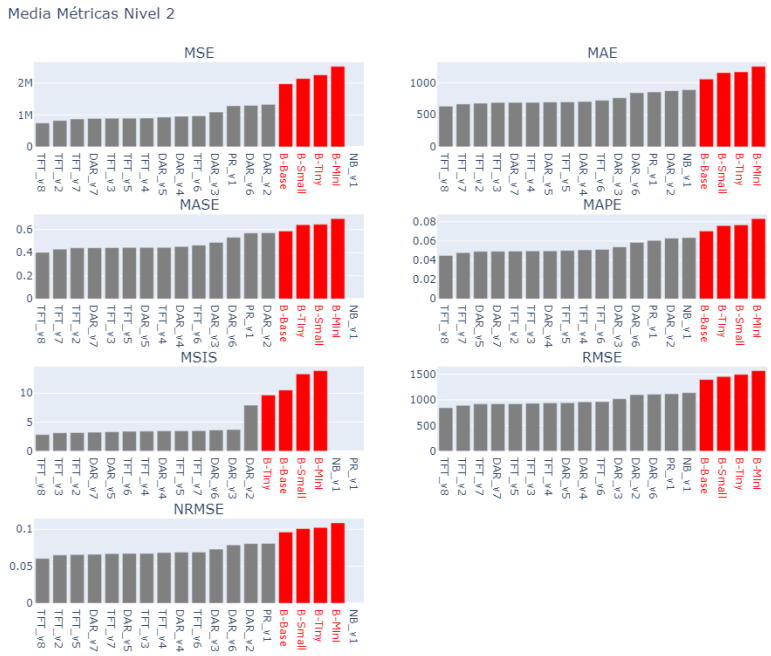

Figura 4.12: Comparativa de las métricas de error resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 2.

Figura 4.13: Comparativa de las métricas de error resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 3.

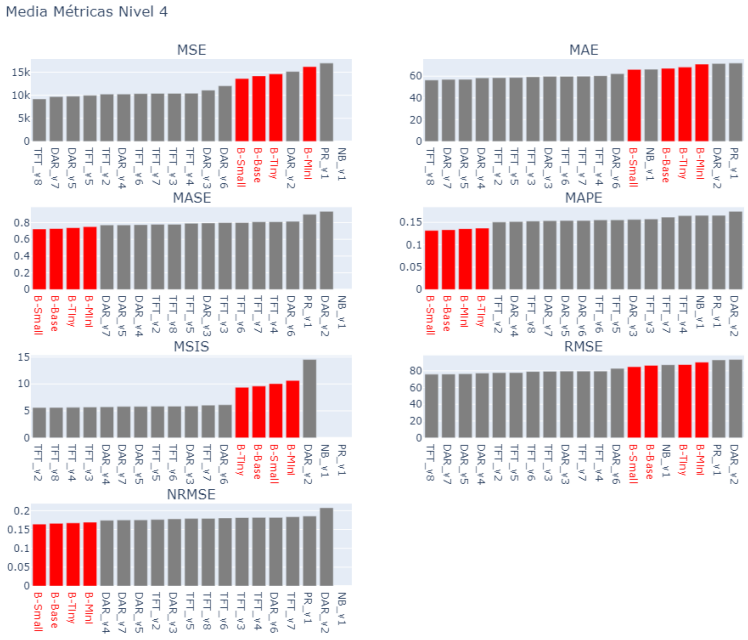

Figura 4.14: Comparativa de las métricas de error resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 4.

A la vista de los gráficos de barras, en los que las correspondientes a los modelos Chronos se muestran destacadas en color rojo, se pueden extraer conclusiones muy relevantes sobre el rendimiento de los modelos Bolt en este ámbito de predicción de series temporales.

En primer lugar, se observa que para la serie TOTAL del nivel 1, las métricas de error obtenidas se sitúan como las "peores" en comparación con los quince modelos restantes, exceptuando algún valor ligeramente mejor como en el caso del MAE, MASE o MAPE. De este modo, se concluye que, al nivel mínimo de profundidad de los datos, los modelos Chronos-Bolt no aportan ningún tipo de mejora en términos de rendimiento en la capacidad predictiva con respecto a los modelos ya conocidos y evaluados. Esto parece lógico, pues estos modelos son realmente sencillos (no requieren de ajuste de hiperparámetros), por lo que, si evaluamos todo el conjunto de datos como una única serie total, la gran cantidad de información sin desglosar limita fuertemente las virtudes de estos modelos.

En los niveles 2 y 3, en los que las series temporales se diferencian según el estado al que pertenecen y el tipo de tienda al que se refieren, respectivamente, los resultados son prácticamente idénticos a lo comentado para el primer nivel, debido a los mismos motivos.

Sin embargo, cuando los datos son desglosados completamente en el último nivel de profundidad, donde el número de series pasa de 10 (nivel 3) a 70 (nivel 4), los resultados cambian drásticamente para bien. En este caso, los modelos Bolt pasan de ser los "peores" a los "mejores" en algunas métricas, como por ejemplo el MASE, donde, además de obtener un valor menor que 1 para los cuatro modelos, se obtienen valores menores que para los quince modelos restantes. En cuanto al MAPE, la situación es la misma. Atendiendo al gráfico, en el que los valores del error se expresan no como porcentaje, sino como proporción, se observa que se consigue bajar del 15 % gracias a los cuatro modelos Bolt. Además, también se logra reducir notablemente los valores del NRMSE.

Por otro lado, también es destacable la enorme diferencia observable en los tiempos de ejecución de Chronos frente al resto de modelos, sobre todo en estos modelos Bolt mejorados, como ya se había introducido con la Figura 4.5. En las Figuras 4.15, 4.16, 4.17 y 4.18 se deja claro:

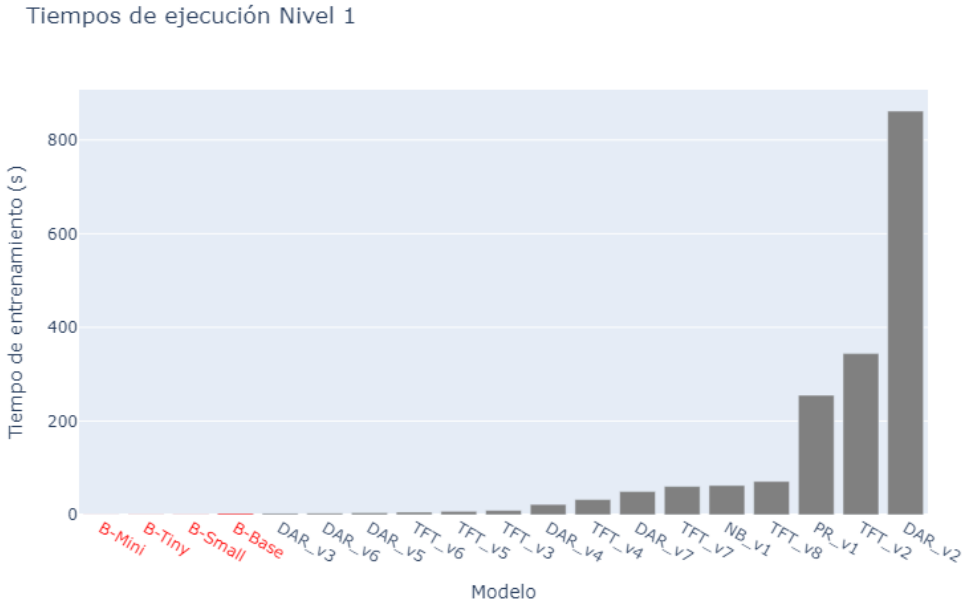

Figura 4.15: Comparativa de los tiempos de ejecución resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 1.

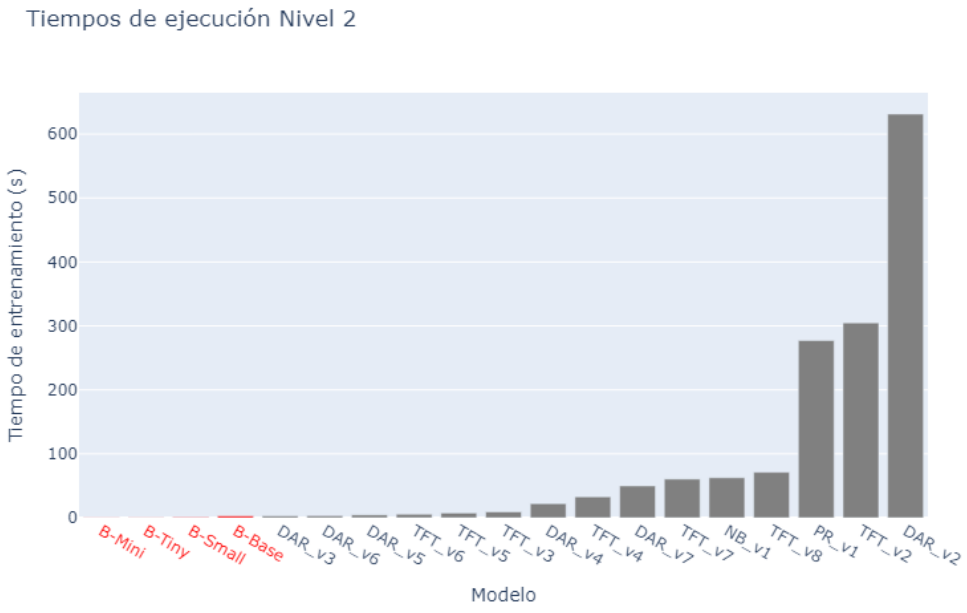

Figura 4.16: Comparativa de los tiempos de ejecución resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 2.

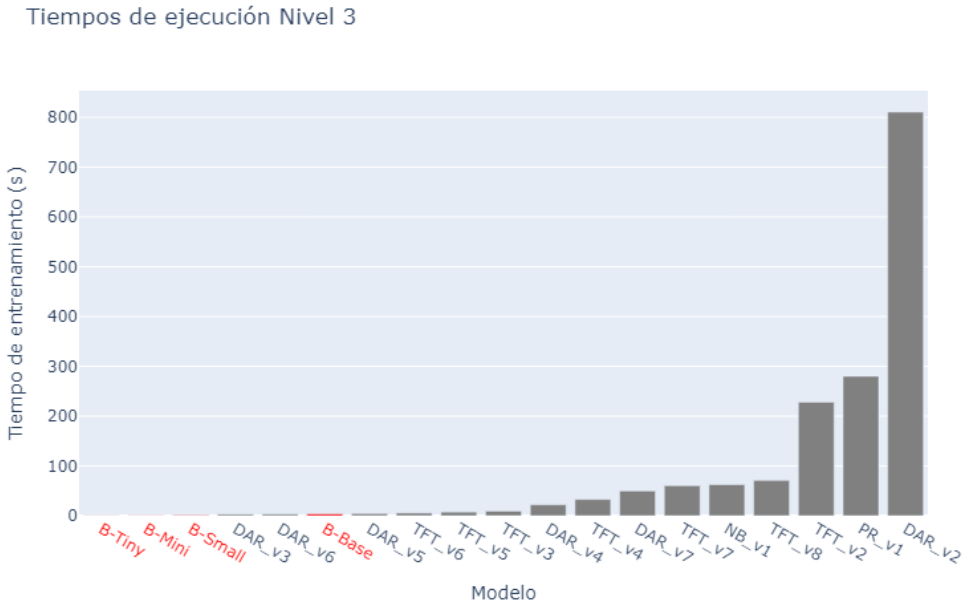

Figura 4.17: Comparativa de los tiempos de ejecución resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 3.

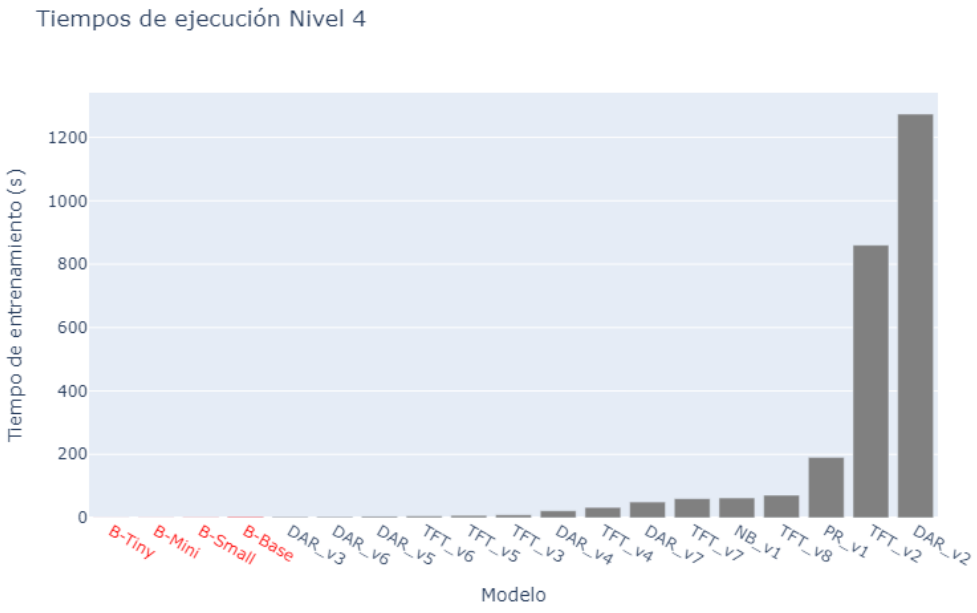

Figura 4.18: Comparativa de los tiempos de ejecución resultantes de los modelos Bolt frente a 15 modelos diferentes en el nivel jerárquico 4.

4.4. Representaciones vectoriales: *embeddings*

Por último, tras haber evaluado profundamente la capacidad predictiva de los modelos Bolt de Chronos, se llevó a cabo un segundo estudio con un objetivo diferente: descubrir los *embeddings* en el contexto de las series temporales.

4.4.1. Definición

Según Hessani (2025), los *embeddings* se definen como el proceso de conversión de datos, de cualquier tipo, en vectores numéricos, con el objetivo de encontrar relaciones entre los mismos. En este caso, los datos a convertir fueron las series temporales, de manera que, transformadas en vectores, permitieron identificar patrones temporales similares, pues, como se verá más adelante, series temporales con patrones semejantes provocan que sus correspondientes vectores o *embeddings* estén muy próximos entre sí.

4.4.2. Conversión vectorial

Para esta tarea, los modelos Chronos utilizados no fueron los Bolt, pues, a día de hoy, la función de Chronos que permite calcular *embeddings* no está disponible para estos nuevos modelos, por lo que se llevó a cabo a partir de los originales: Tiny, Mini, Small, Base y Large. Concretamente, los resultados que se presentan en este documento resultan de los *embeddings* generados a partir del modelo Tiny, para así no sobrecargar el documento con multitud de gráficos similares, pues, salvo los tiempos de ejecución, los resultados obtenidos a partir de los cinco modelos no mostraron diferencias significativas.

De esta manera, el procedimiento para la conversión de las series temporales a *embeddings* fue el siguiente: igual que en el caso de las predicciones, se creó un bucle para trabajar con los 84 ID diferentes de los datos. Para cada uno de ellos, se creó nuevamente un contexto tipo tensor en el que almacenar los valores de la variable "Num_Pedidos", en este caso sin dividir el conjunto de datos en una muestra

de entrenamiento y test, sino seleccionando todo el rango de datos. Para ello, se precisó configurar la variable tensor para seleccionar los 1.941 puntos, pues, por defecto, Chronos toma únicamente los últimos 512. Una vez reconfigurada la longitud, se inicializó la función correspondiente al cálculo de *embeddings*, modificando también ciertos detalles, pues existen varias posibilidades a la hora de generar los vectores. Los *embeddings*, por defecto, añaden un último valor, además de los 1.941 puntos, al final de su secuencia, conocido como *end-of-sequence* (EOS), cuyo objetivo es marcar el fin de la misma, por lo que es aconsejable eliminarlo, ya que para posteriores análisis afectaría negativamente si se tratase como un *embedding* representativo. Tras su eliminación, se realizó el promedio de los 1.941 valores restantes, de manera que, para cada punto, se obtuvo un único *embedding* representativo del mismo.

4.4.3. Aplicaciones

Una vez transformadas las series temporales en vectores, se llevaron a cabo distintas técnicas estadísticas conocidas con el objetivo de evaluar la precisión de los *embeddings* en Chronos.

Distancias euclídeas

La primera aplicación se centró en el cálculo de distancias euclídeas entre vectores (*embeddings*). Esta métrica se corresponde con la distancia ordinaria en línea recta existente entre dos puntos en el espacio euclidiano, definida:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{4.2}$$

de manera que, denotando como x_i a los valores de los *embeddings* de una serie e y_i a los de otra serie, se podría calcular la distancia euclídea existente entre ambas.

De este modo, se procedió a calcular una matriz con las distancias euclídeas entre todos los pares posibles de series diferentes. Así, se pudo comprobar si las representaciones vectoriales a partir de los modelos resultaban precisas, comparando los valores resultantes de las distancias euclídeas con gráficos de doble eje que representaron las dos series temporales en cuestión para cada caso. Esto se contempla realmente bien en los siguientes dos casos: en la Figura 4.19 se representan las dos series temporales cuya distancia euclídea es máxima en la matriz generada, junto a su respectivo valor. En la Figura 4.20, se presenta el caso opuesto, donde la distancia euclídea es la menor observada:

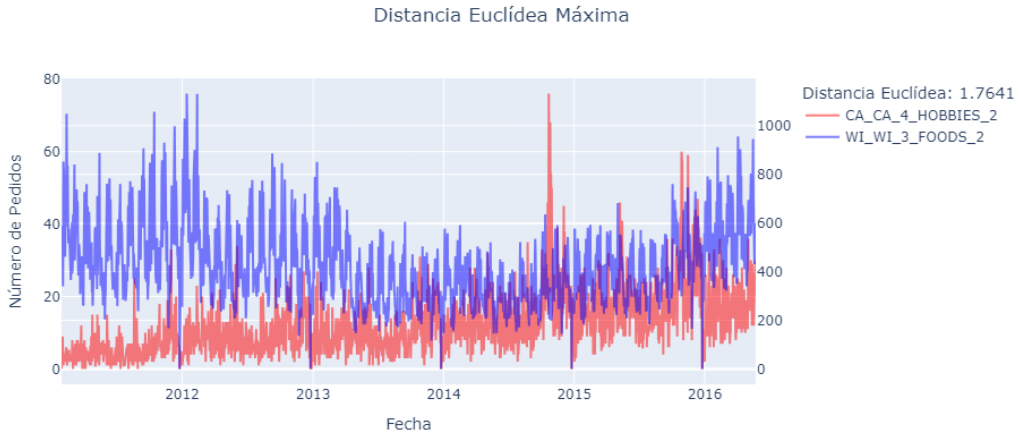

Figura 4.19: Representación gráfica de las series temporales cuya distancia euclídea es mayor.

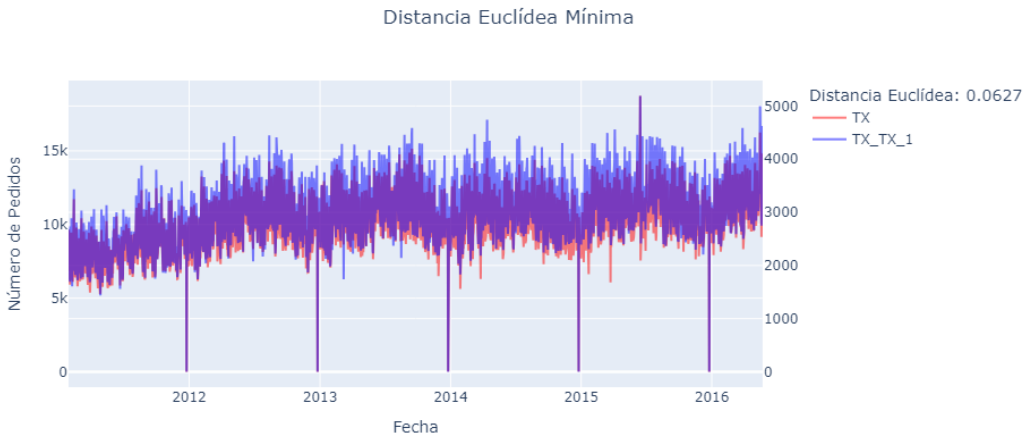

Figura 4.20: Representación gráfica de las series temporales cuya distancia euclídea es menor.

A la vista de las imágenes, se comprueba el correcto funcionamiento de la función de Chronos que calcula *embeddings*, pues, en el caso de las series con un valor elevado (el mayor observado) en cuanto a distancia euclídea entre ambas, se observa que las representaciones gráficas de las mismas difieren mucho entre sí, por lo que los *embeddings* están capturando correctamente esas diferencias. Y, en el segundo caso, también se corrobora que los vectores de ambas series identifican de forma correcta la similitud entre ellas, ya que pertenecen al mismo estado (Texas).

Reducción de la dimensión

Tras conocer los *embeddings* y haber evaluado su precisión a la hora de tratar las series temporales como vectores numéricos, se incidió en un detalle: su tamaño. Explorando la función ".embed()" que transforma los datos de las series en *embeddings*, se descubrió que Chronos, independientemente de la longitud del contexto, genera *embeddings* con una dimensión de 256. Es decir, las series, de longitud T, se representan como vectores de dimensión 256. Esto puede resultar insignificante para ciertas tareas como el cálculo de distancias euclídeas; sin embargo, supone una limitación en otros ámbitos como las representaciones gráficas, pues no es posible graficar vectores de tal dimensión.

Por ello, se llevaron a cabo técnicas de reducción de la dimensión. Primeramente, se trató de reducirla a través de la técnica *t-SNE*, cuyas siglas, en inglés, significan: "t-distributed Stochastic Neighbor Embedding". Esta técnica es ampliamente utilizada para la visualización de datos de alta dimensión en espacios bidimensionales y tridimensionales. Para un profundo entendimiento de la técnica, convendría revisar la información recogida en Van der Maaten y Hinton (2008), pero, como este trabajo se centra sobre todo en los resultados logrados con la librería Chronos, simplemente cabe destacar que el objetivo de la misma radica en mantener la estructura local de los datos, de manera que los puntos que eran similares en el espacio original se situarán próximos en el nuevo espacio. Esto se logra transformando las distancias entre puntos en probabilidades de similitud y, posteriormente, ajustando las posiciones en el espacio reducido con el objetivo de que esas probabilidades se mantengan lo más parecidas posible.

t-SNE resulta realmente útil a la hora de explorar patrones, agrupaciones o relaciones ocultas en datos complejos, como en el caso de los *embeddings* generados por modelos de aprendizaje automático. Por ello, se empleó esta técnica para reducir la dimensión 256 de los *embeddings* a únicamente 3 dimensiones, para tratar así de descubrir relaciones hasta ahora no identificadas.

Al aplicar *t-SNE*, los resultados dependen de varios parámetros prefijados en la función. Estos son, entre otros, *perplexity*, que se encarga de regular el equilibrio entre la atención al entorno local y

global según el número de vecinos considerados para cada punto; *n_iter*, que indica el número efectivo de iteraciones que el algoritmo debe ejecutar para optimizar la representación en baja dimensión; o *learning_rate* (tasa de aprendizaje). Para esta prueba, se seleccionaron los valores de los parámetros establecidos por defecto por la función ($perplexity = 30$, $n\_iter = 1000$ y $learning\_rate = 200$), pues el objetivo fue llevar a cabo una primera visualización general para posteriormente aplicar un análisis por componentes principales, en el que se obtuvieron resultados interesantes.

Así, primeramente los *embeddings* fueron escalados de forma que todas las dimensiones pasaron a tener una media igual a cero y desviación estándar igual a uno. Así, se evitó que dimensiones con valores mucho más elevados que otras dominasen las distancias que el algoritmo utiliza para calcular similitudes entre puntos, y todas contribuyesen de manera justa a la reducción, pues *t-SNE* es sensible a escalas. Una vez escalados, se llevó a cabo la reducción de la dimensión, obteniéndose también las matrices de dispersión de las Figuras 4.21, 4.22, 4.23, 4.24 y 4.25:

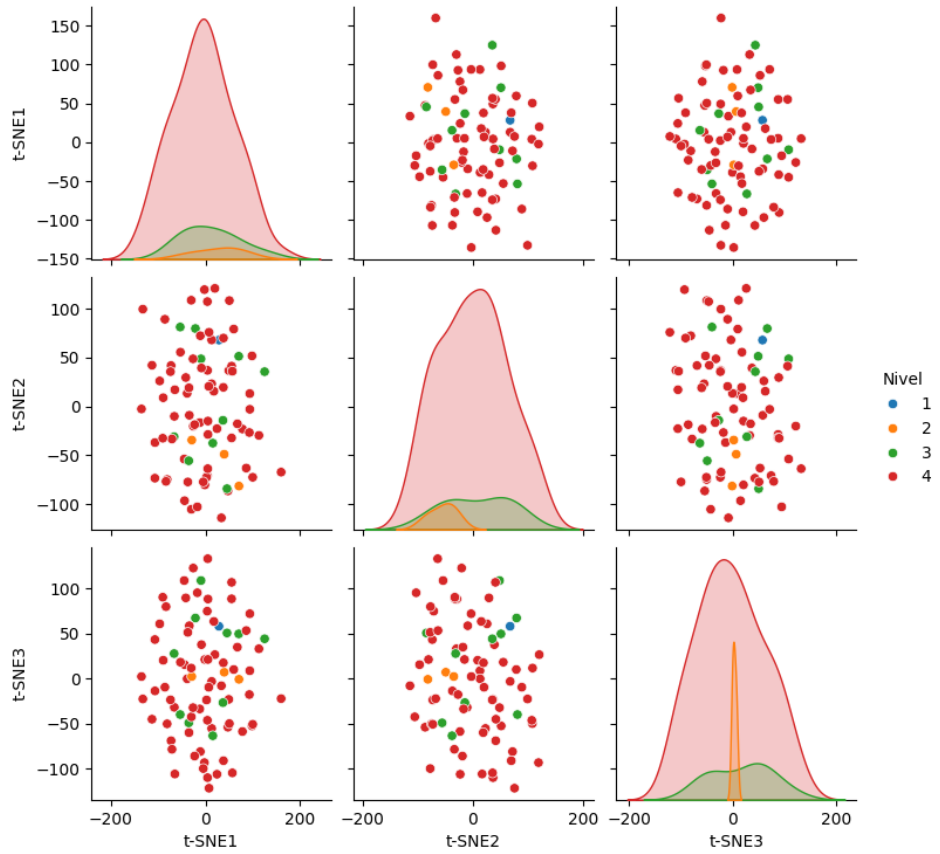

Figura 4.21: Matriz de dispersión de los datos distinguidos según su nivel jerárquico generada mediante *t-SNE*.

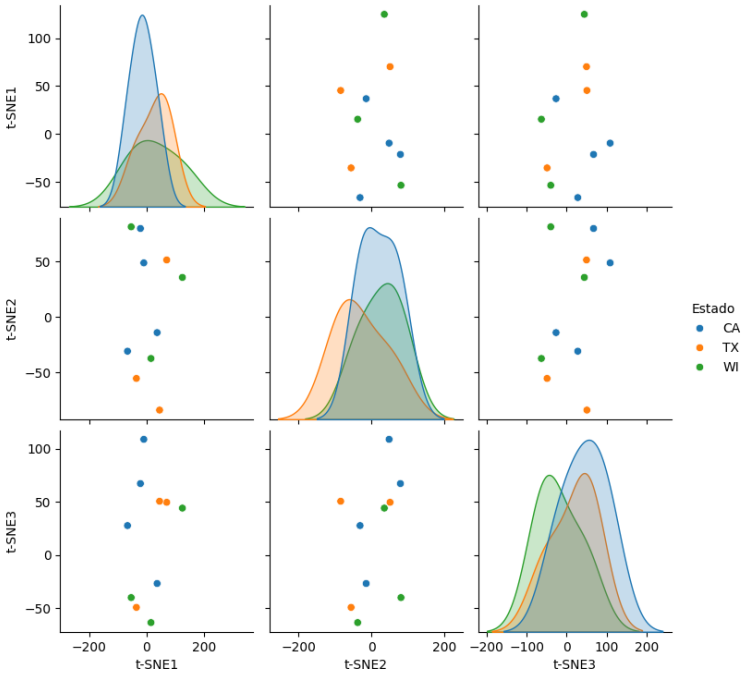

Figura 4.22: Matriz de dispersión de las diferentes tiendas distinguidas según el estado en el que se encuentran generada mediante *t-SNE*.

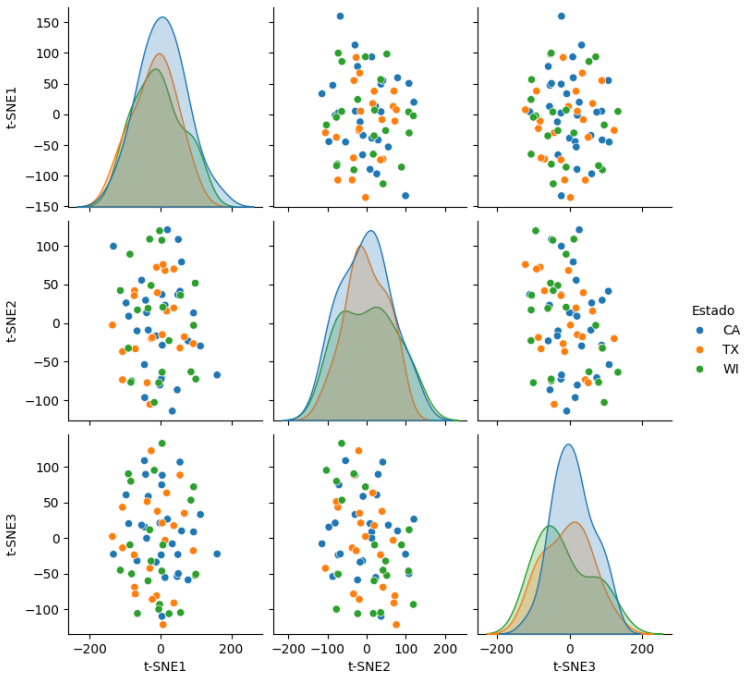

Figura 4.23: Matriz de dispersión de los diferentes departamentos distinguidos según el estado en el que se encuentran generada mediante *t-SNE*.

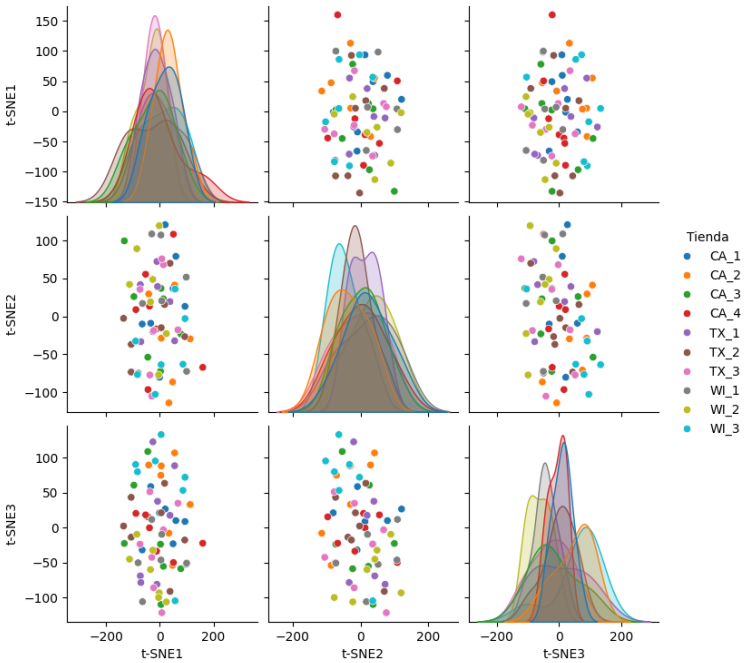

Figura 4.24: Matriz de dispersión de los diferentes departamentos distinguidos según la tienda a la que pertenecen generada mediante *t-SNE*.

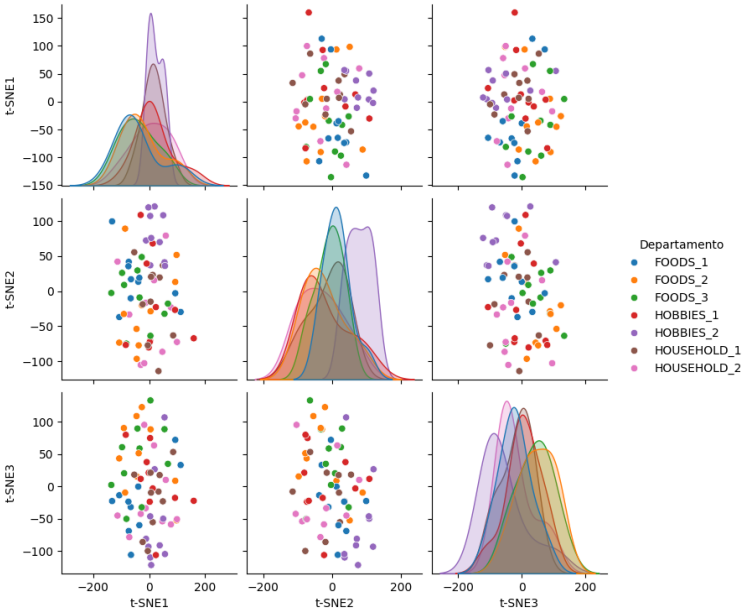

Figura 4.25: Matriz de dispersión de los diferentes departamentos distinguidos según el tipo de departamento al que pertenecen generada mediante *t-SNE*.

En las matrices de dispersión adjuntas se muestra la siguiente información: en cada una de ellas se presentan los diferentes diagramas de dispersión resultantes entre todas las posibles combinaciones por pares entre las tres dimensiones, y en la diagonal, aparece otro tipo de gráfico, que se corresponde con representaciones de la estimación de la densidad, en los que se muestran cómo se distribuyen los valores de cada una de las dimensiones por separado. Además, los puntos y densidades aparecen coloreados de acuerdo a una leyenda, también especificada en el pie de las figuras.

Esto así, se observa que, a pesar de haber generado diferentes gráficos con distintas combinaciones de los datos (*embeddings*) reducidos, no se identifica ningún tipo de patrón o relación oculta en ellos, por lo que se descartó esta técnica en cuanto a aportación de valor al trabajo.

Entonces, tras haber "fracasado" empleando *t-SNE*, se contempló otra técnica de reducción de la dimensión: Análisis de Componentes Principales (*PCA*, por sus siglas en inglés), sobre la que se puede ahondar en Jolliffe (2002). Esta técnica, cuyo objetivo final es idéntico al de *t-SNE*, presenta ciertas diferencias durante el proceso de reducir las dimensiones. Los datos, en este caso *embeddings*, se ven reducidos desde una alta dimensión hasta unas pocas componentes, denominadas "componentes principales". Estas componentes son ordenadas según el porcentaje de varianza de los datos originales que pueden explicar, lo que permite representar la información esencial del conjunto original en menos dimensiones.

Así, en una primera instancia se calculó el número de componentes necesarias para explicar un porcentaje de los *embeddings* de al menos un noventa por ciento, mínima cifra que se consideró suficiente para poder extraer conclusiones con cierto grado de veracidad. Se observó que, a través de 11 componentes, se lograba explicar un 90,07 % de la variabilidad, lo que resultaba un número elevado para poder representarlas. Por ello, se siguió el mismo patrón que mediante *t-SNE*, representando diferentes combinaciones de los datos reducidos a tres dimensiones, como se muestra en las Figuras 4.26, 4.27, 4.28, 4.29 y 4.30:

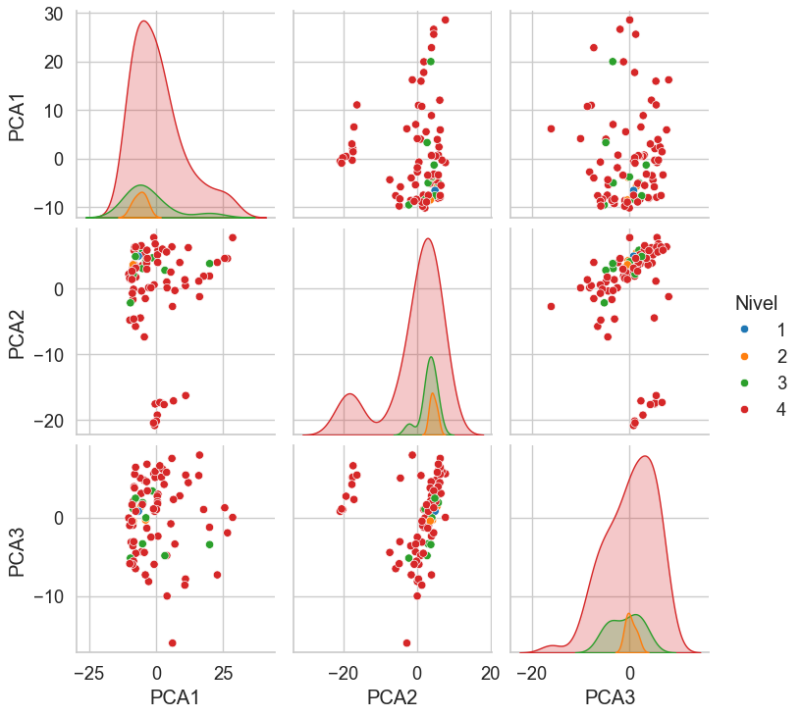

Figura 4.26: Matriz de dispersión de los datos distinguidos según su nivel jerárquico generada mediante *PCA*.

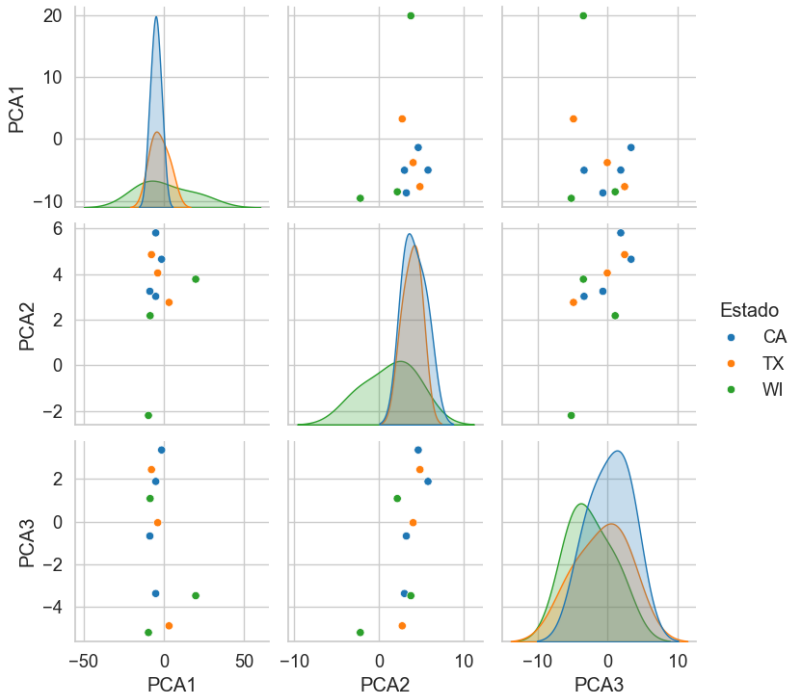

Figura 4.27: Matriz de dispersión de las diferentes tiendas distinguidas según el estado en el que se encuentran generada mediante *PCA*.

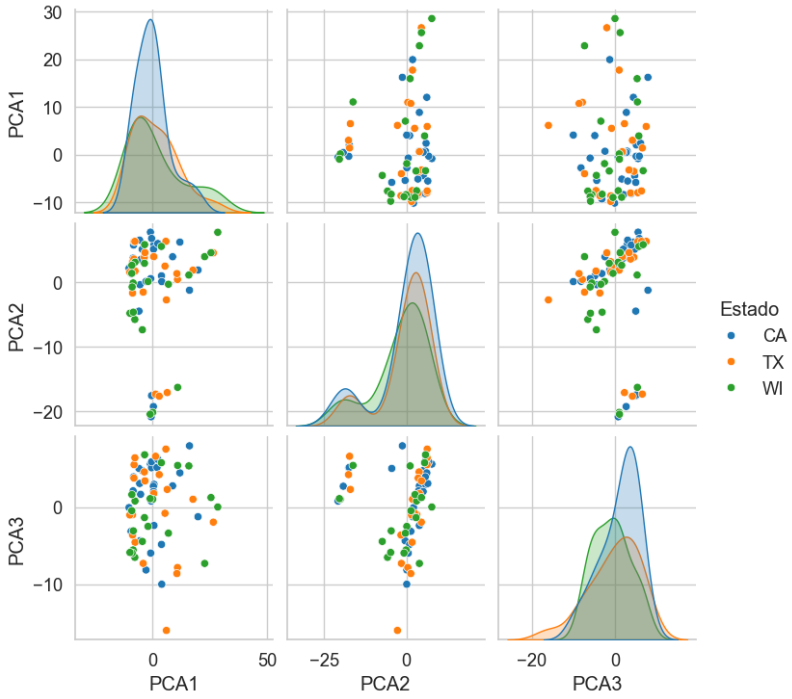

Figura 4.28: Matriz de dispersión de los diferentes departamentos distinguidos según el estado en el que se encuentran generada mediante *PCA*.

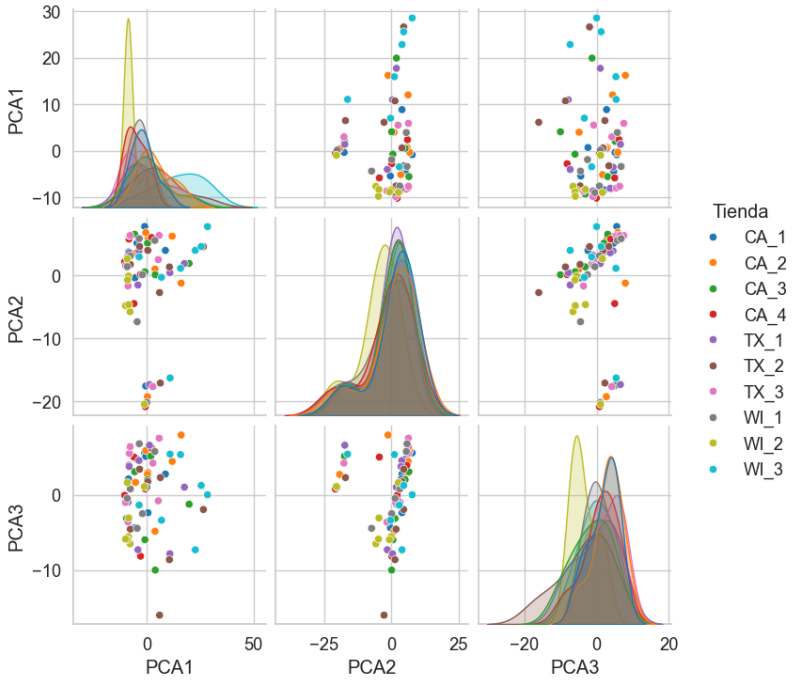

Figura 4.29: Matriz de dispersión de los diferentes departamentos distinguidos según la tienda a la que pertenecen generada mediante *PCA*.

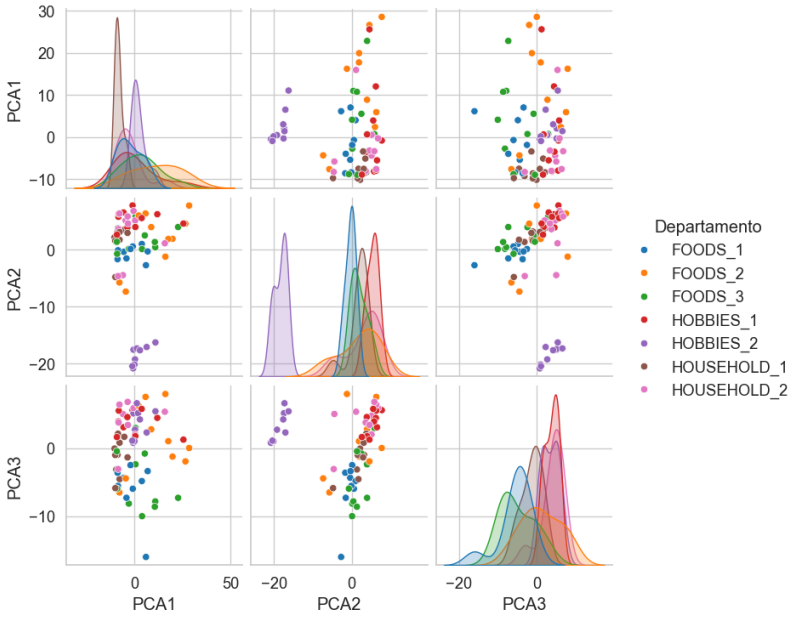

Figura 4.30: Matriz de dispersión de los diferentes departamentos distinguidos según el tipo de departamento al que pertenecen generada mediante *PCA*.

Atendiendo a las diferentes matrices de dispersión, se observa que, después de evaluar las mismas combinaciones que en el paso anterior con *t-SNE*, ahora sí se logra identificar una agrupación no identificada hasta el momento. En la Figura 4.30, en la que los *embeddings* correspondientes a las series temporales de nivel 4 son representados mediante reducción de su dimensión, se contempla que, al distinguir los datos por colores según el tipo de departamento, los *embeddings* referentes al tipo de departamento "HOBBIES_2" se sitúan claramente diferenciados del resto. A raíz de esto, se llevó a cabo un pequeño estudio de indagación con el fin de descubrir el motivo de esta diferenciación, pero sin éxito. Por ello, tras tratar el tema en cuestión con los responsables de la empresa, se entendió que se debía, seguramente, a información exógena no proporcionada, como por ejemplo, afectaciones del calendario. Pero, a pesar de esta limitación, el resultado se consideró exitoso, pues se demostró que los *embeddings* generados con Chronos, además de resultar representaciones vectoriales de las series precisas, también permiten la identificación de información oculta en los datos mediante técnicas de reducción de la dimensión.

Clustering

Por último, también se llevó a cabo una tercera tarea con los *embeddings*: formación de grupos mediante *clustering*. El concepto *clustering* se define como una técnica de aprendizaje automático consistente en el agrupamiento de un conjunto de datos en subgrupos (*clusters*) de manera que los elementos de cada grupo presentan mayor similitud entre sí que con elementos de otros grupos. De este modo, a partir de los vectores numéricos de los *embeddings*, se desarrollaron dos técnicas de *clustering* diferentes: *K-Means* y *HDBSCAN*, para tratar de comprobar si los elementos referentes al departamento "HOBBIES_2" volverían a diferenciarse respecto al resto, evidenciando así aún más su peculiaridad. Para mayores detalles sobre estas técnicas, se aconseja consultar Madhulatha (2012).

Inicialmente, se realizó *clustering* mediante el algoritmo conocido como *K-Means*. *K-Means* consiste en elegir un número K de *clusters* a formar, de forma que dichos grupos se crean de la siguiente manera: se inicializan aleatoriamente K centros de grupo, denominados centroides. Una vez formados, los puntos de los datos se asignan a cada centroide según su proximidad a los mismos. A continuación, se recalculan los centroides como el promedio de los puntos de cada grupo y, tras ello, se repite el proceso hasta que los centroides convergen a un punto marcado, quedando así formados K *clusters*.

En este caso, el número K de *clusters* seleccionado se basó en el método del codo. El método del codo consiste en ejecutar el algoritmo *K-Means* con diferentes valores de K y, para cada caso, calcular la Suma de Errores al Cuadrado (SSE, por sus siglas en inglés). Una vez calculados todos los valores de error, se grafican con respecto a los valores de K, de manera que el valor óptimo de K elegido es aquel para el que la curva resultante al unir los puntos de la gráfica comienza a aplanarse paralelamente al eje x, formando así una especie de "codo". En este caso, como $K = 11$ era el valor que explicaba al menos un 90 % de la varianza de los datos en *PCA*, se desarrolló el método del codo hasta dicho valor, resultando el gráfico de la Figura 4.31.

A la vista de la imagen, se observa que la gráfica no se aplana de forma clara en ningún momento, por lo que, tras probar diferentes valores de K para elegir el número de *clusters*, se decidió que la cifra adecuada en relación complejidad-resultados fue $K = 5$. Los resultados del *clustering* se muestran en la Figura 4.32, donde se presentan con respecto a los siete tipos de departamentos existentes en el nivel 4 de los datos:

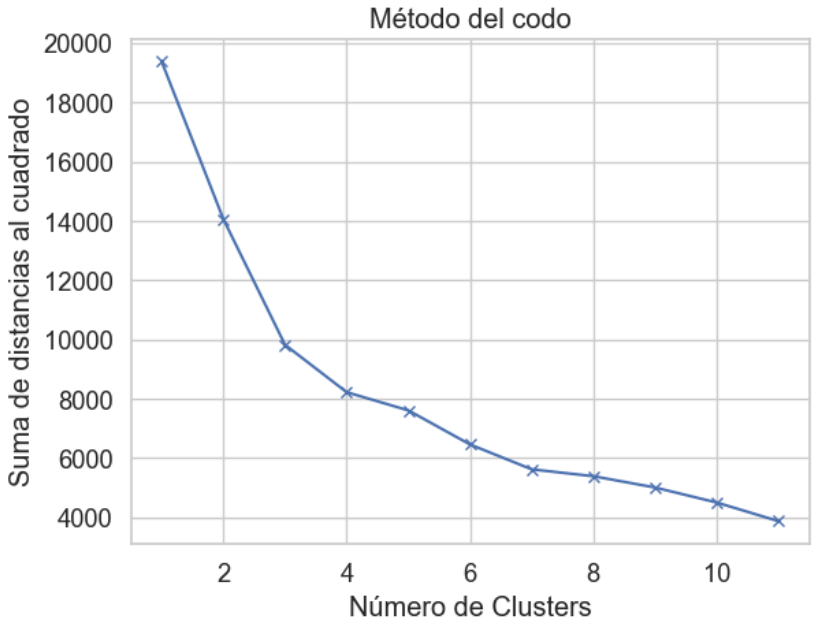

Figura 4.31: Método del codo aplicado a los *embeddings* del conjunto de datos en el algoritmo *K-Means* (*Clustering*).

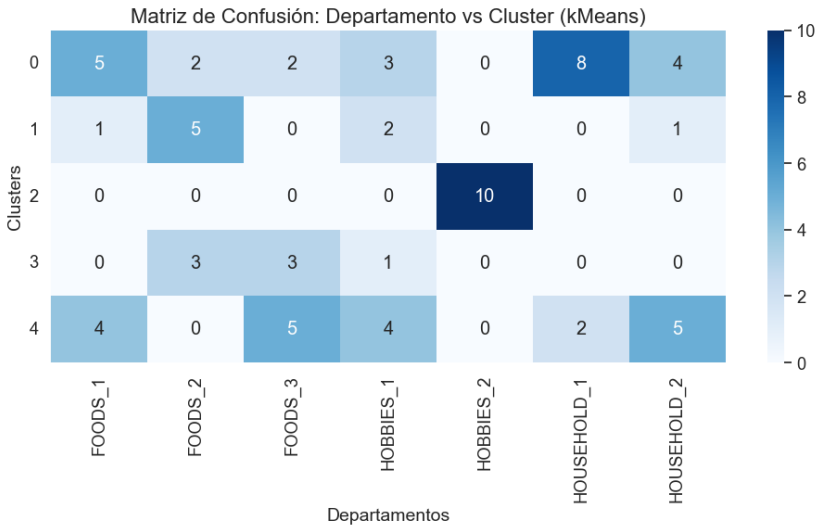

Figura 4.32: *Clusters* resultantes del método del codo frente al tipo de departamentos del conjunto de datos.

Atendiendo a la matriz mostrada, se comprueba que el departamento "HOBBIES_2" vuelve a ser correctamente identificado mediante *clustering*, en este primer caso, a través del algoritmo *K-Means*. Los diez valores correspondientes a dicho departamento se agrupan en un único *cluster* en la fila número dos, mientras que, en el resto de casos, surge bastante mayor confusión en cuanto a su agrupamiento, ya que, a partir de los *embeddings*, el método no reconoce grandes diferencias entre los datos, como ya había pasado en *t-SNE* y *PCA*.

Finalmente, se realizó el mismo proceso pero empleando otro algoritmo distinto al *K-Means*: *HDBSCAN*, cuyas siglas en inglés significan "Hierarchical Density-Based Spatial Clustering of Applications with Noise". Este algoritmo forma *clusters* a partir de la densidad de los datos. Mediante cálculo de distancias, mide el nivel de agrupación de los puntos y construye un grafo de expansión mínima de manera jerárquica según las densidades resultantes. A continuación, elimina los grupos más inestables o cuyas conexiones entre puntos son más débiles y mantiene aquellos que son más robustos. Y, tras ello, asigna a cada punto uno de estos *clusters*. Además, si un punto no se define de forma clara en ningún *cluster*, lo define como "ruido".

Continuando con los *embeddings*, se hizo uso de la librería "hdbscan" de Python para realizar este agrupamiento. La función correspondiente permitía modificar ciertos parámetros, por lo que se estableció un tamaño mínimo de diez puntos para poder formarse un *cluster* y la métrica euclídea para el cálculo de distancias. En la Figura 4.33 se observa la matriz resultante:

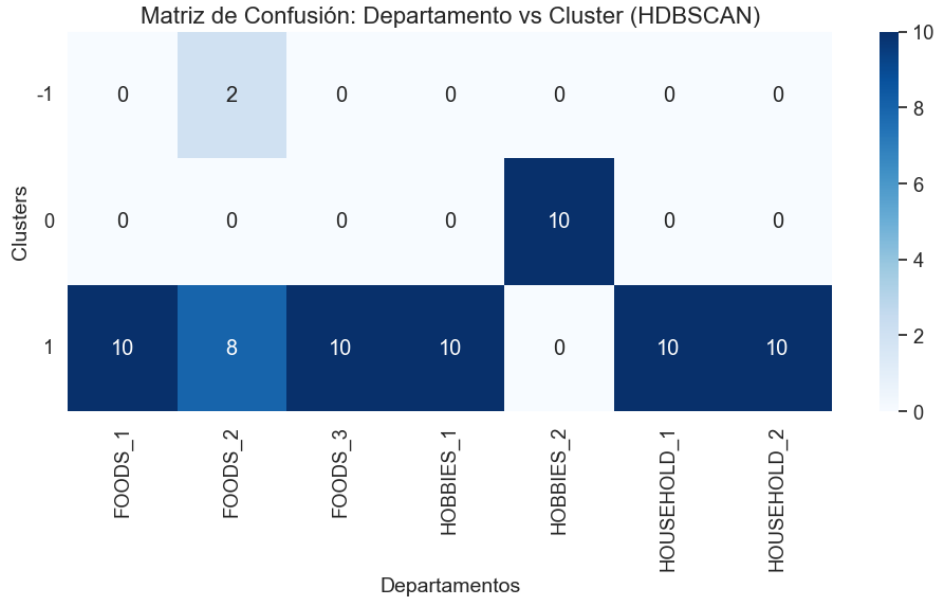

Figura 4.33: *Clusters* resultantes del algoritmo HDBSCAN frente al tipo de departamentos del conjunto de datos.

Mediante este método, se constata la particularidad de los datos del departamento de "HOBBIES_2" frente al resto, pues únicamente formando dos *clusters* (la fila categorizada como "-1" se refiere a los valores considerados como ruido) se consigue separar dichos *embeddings* de todos los demás, lo que refuerza nuevamente las dos conclusiones más relevantes de este apartado: Chronos permite transformar datos en *embeddings* logrando un notable rendimiento en su manejo y, dentro del conjunto de datos, aquellos pertenecientes al departamento "HOBBIES_2" se diferencian claramente del resto debido a información que escapa del alcance de este trabajo, pero que podría sugerir una línea de continuación de investigación sobre ello.

Capítulo 5

Conclusiones y extensiones

A la vista de los resultados expuestos en el Capítulo 4, a continuación se destacan, tanto a favor como en contra, las conclusiones extraídas sobre la implementación de los modelos Chronos, basados en arquitectura Transformer, en la predicción de series temporales, con respecto a las metodologías clásicas Box-Jenkins y suavización exponencial.

A nivel general, la primera característica que salta a la vista acerca de la librería Chronos es su sencilla implementación. Al tratarse de modelos preentrenados, su uso se reduce a la carga de los mismos en el entorno de trabajo. Una vez cargados, basta con crear una variable de tipo tensor y ejecutar la función *.predict* sobre ella, especificando el horizonte de predicción, para obtener los valores futuros deseados. Este proceso no requiere de otras subtareas, como el ajuste de hiperparámetros, la creación de ventanas deslizantes para llevar a cabo el entrenamiento del conjunto de datos o la inclusión de variables exógenas. Además, el rendimiento ofrecido, el cual se tratará en el párrafo contiguo, no se ve afectado por la limitación ligada a la longitud del contexto. Por ello, Chronos supone una alternativa realmente interesante en términos de nivel de programación y sencillez de uso.

A pesar de su simplicidad, el rendimiento observado fue notable. Atendiendo a las métricas de error obtenidas en la tarea de predicción temporal en comparación con las correspondientes a otros modelos de deep learning entrenados por la empresa sobre el mismo conjunto de datos, los resultados fueron favorables a los modelos Chronos, logrando valores mínimos de error en métricas como el MASE, MAPE o NRMSE (Figura 4.14). Eso sí, es necesario puntualizar que esta mejora ocurre bajo un contexto específico, en el que el conjunto de datos es ampliamente desglosado, concretamente, en setenta series temporales diferentes. Por ello, aunque se destaca el rendimiento de los modelos Chronos, cabe señalar que ello depende del contexto bajo el que se trabaje, de manera que se pueden dar casos en los que esta metodología no resulte la mejor opción para realizar predicción temporal, en especial, si los datos en cuestión presentan una elevada homogeneidad.

Además, el reciente lanzamiento de los modelos Bolt-Chronos en noviembre de 2024 supuso un importantísimo avance en la librería. Como se expuso en la Figura 4.5, esta mejora logró reducir drásticamente los tiempos de computación requeridos hasta el momento, alcanzando velocidades entre 80 y 268 veces mayores en comparación con sus modelos homólogos originales (por ejemplo, en el caso de Chronos-Bolt Tiny vs. Chronos Tiny y Chronos-Bolt Base vs. Chronos Base, respectivamente). Esto supuso un punto de inflexión en el caso particular de este trabajo, ya que, durante su fase de desarrollo, en la cual se precisó llevar a cabo una gran cantidad de simulaciones, agilizó notablemente la velocidad de ejecución de las mismas.

Por último, conviene resaltar también las conclusiones extraídas en la aplicación de representaciones vectoriales mediante *embeddings* de las series temporales, a través de la función *.embed*. Por un lado, se evidenció un buen rendimiento en la vectorización de las mismas, validado de acuerdo a la coherencia observada entre las distancias euclídeas y los patrones visuales de las series. Las distancias euclídeas calculadas entre los *embeddings* asociados a cada serie, cuyos valores indican el grado de similitud o diferencia entre ellas, se correspondieron razonablemente con los gráficos secuenciales

correspondientes, de manera que series temporales con patrones similares reflejaron valores bajos en la matriz de distancias euclídeas calculada, y viceversa.

Por otro lado, en referencia a los procedimientos de reducción de la dimensión desarrollados, se recalca que la técnica *t-SNE* no aportó ningún tipo de información relevante acerca de los diferentes tipos de departamentos existentes en los almacenes Walmart. No obstante, la reducción por componentes principales sí permitió diferenciar uno de ellos, catalogado como "HOBBIES_2". Posteriormente, esto se pudo lograr también mediante herramientas de aprendizaje no supervisado como la clusterización de los datos a través del método *K-Means* y *HDBSCAN*.

Sin embargo, a pesar de haber tratado de averiguar la razón de la desviación detectada en la estructura de los datos correspondientes al departamento "HOBBIES_2", no se logró identificar el motivo que la provocaba. Por ello, como posible extensión de este trabajo, se propone continuar investigando este fenómeno, ya sea mediante la aplicación de técnicas adicionales no consideradas en este estudio, o a través del entrenamiento del conjunto de datos con modelos distintos que permitan incorporar variables externas, como, por ejemplo, información relacionada con el calendario estadounidense.

Bibliografía

[1] Amazon Web Services (2024). Chronos-Bolt-Tiny: Pretrained Time Series Forecasting Model. https://huggingface.co/amazon/chronos-bolt-tiny.

[2] Ansari A.F., Stella L., Turkmen C., Zhang X., Mercado P., Shen H., Shchur O., Rangapuram S.S., Arango S.P., Kapoor S., Zschiegner J., Maddix D.C., Wang H., Mahoney M.W., Torkkola K., Wilson A.G., Bohlke-Schneider M., Wang Y. (2024). Chronos: Learning the Language of Time Series. https://arxiv.org/abs/2403.07815.

[3] Ba J.L., Kiros J.R., Hinton G.E. (2016). Layer Normalization. https://arxiv.org/abs/1607.06450.

[4] Brockwell P.J., Davis R.A. (1991). Time Series: Theory and Methods. Springer, New York.

[5] Carrión-García A. (2001). Análisis de series temporales, técnicas de previsión. ResearchGate.

[6] Chong S.F., Choo R. (2011). Introducing to Bootstrap. Proceedings of Singapore Healthcare 20:236-240.

[7] Gardner Jr. E.S. (1985). Exponential smoothing: The state of the art. Journal of Forecasting 4:1-28.

[8] Guerrero V.M. (1993). Time-series analysis supported by power transformations. Journal of Forecasting 12:37-48.

[9] Hastie T., Tibshirani R., Friedman J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

[10] Hessani H.S. (2025). LLM Embeddings Explained: A Visual and Intuitive Guide. https://huggingface.co/spaces/hesamation/primer-llm-embedding?section=references.

[11] Huet P. (2023). Qué son las redes neuronales y sus aplicaciones. https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/.

[12] Hyndman R.J., Athanasopoulos G. (2018). Forecasting: principles and practice. OTexts, Melbourne.

[13] IBM (s.f.). ¿Qué es el descenso de gradiente? https://www.ibm.com/es-es/think/topics/gradient-descent. Accedido 17 de abril de 2025.

[14] Jolliffe I.T. (2002). Principal Component Analysis. Springer, New York.

[15] Kingma D.P., Ba J. (2017). Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980.

[16] Madhulatha T.S. (2012). An Overview on Clustering Methods. https://arxiv.org/abs/1205.1117.

[17] OECD (2024). OECD Digital Economy Outlook 2024 (Volume 2): Strengthening Connectivity, Innovation and Trust. OECD Publishing, París.

[18] Statista Research Department (2024). Walmart - Datos estadísticos. https://es.statista.com/temas/8019/walmart/#topicOverview.

[19] Van der Maaten L., Hinton G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research 9:2579-2605.

[20] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I. (2023). Attention Iis All You Need. https://arxiv.org/abs/1706.03762.

Índice de figuras