Trabajo Fin de Máster

Modelo de detección de anomalías para el aseguramiento de la calidad de datos

María Teresa Villar Guerra

Máster en Técnicas Estadísticas

Curso 2024-2025

Propuesta de Trabajo Fin de Máster

Título en galego: Modelo de detección de anomalías para a garantía da calidade dos datos
Título en español: Modelo de detección de anomalías para el aseguramiento de la calidad de datos
English title: Anomaly detection model for data quality assurance
Modalidad: Modalidad B
Autor/a: María Teresa Villar Guerra, Universidad de Vigo
Director/a: Salvador Naya Fernández, Universidad de A Coruña; Javier Tarrío Saavedra, Universidad de A Coruña
Tutor/a: Jorge García Romarís, ABANCA CORPORACIÓN BANCARIA S.A.
Breve resumen del trabajo: El modelo a desarrollar utilizará tecnicas de aprendizaje automático y estadística avanzada para evaluar la calidad de los datos en diferentes dimensiones, como la comletitud, la unicidad, la consistencia, la validez y la precisión.
Recomendaciones: Conocimientos de Python y SQL
Otras observaciones: El alumno se incorporará a la Entidad en calidad de Estudiante en Prácticas. Estas prácticas están remuneradas

Don Salvador Naya Fernández, Catedrático de la Universidad de A Coruña, don Javier Tarrío Saavedra, Catedrático de la Universidad de A Coruña, y don Jorge García Romarís, Director de Data, Analytics & R^2Tech de ABANCA CORPORACIÓN BANCARIA S.A., informan que el Trabajo Fin de Máster titulado

Modelo de detección de anomalías para el aseguramiento de la calidad de datos

fue realizado bajo su dirección por doña María Teresa Villar Guerra para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal. Además, don Salvador Naya Fernández, don Javier Tarrío Saavedra y doña María Teresa Villar Guerra

√ sí ☐ no

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

En A Coruña, a 22 de julio de 2025.

El director:
Don Salvador Naya Fernández

NAYA FERNANDEZ SALVADOR - DNI 76353731X

Firmado digitalmente por NAYA FERNANDEZ SALVADOR - DNI 76353731X
Fecha: 2025.07.22 16:16:11 +02'00'

El director:
Don Javier Tarrío Saavedra

TARRIO SAAVEDRA JAVIER - DNI 53164914T

Firmado digitalmente por TARRIO SAAVEDRA JAVIER - DNI 53164914T
Fecha: 2025.07.22 18:21:56 +02'00'

El tutor:
Don Jorge García Romarís

La autora:
Doña María Teresa Villar Guerra

J Teresa

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

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

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

Agradecimientos

Me gustaría agradecer el apoyo a todas las personas que han estado a mi lado durante la realización de este trabajo, acompañándome tanto en los momentos buenos como en aquellos que no lo han sido tanto.

Quiero destacar por supuesto a mis padres y a mi pareja que siempre están cuando los necesito. Y hacer una mención especial al equipo de Inteligencia Artificial de Abanca. A mis compañeros Aida, Martina, Diego, Julia, Álex, Raquel y Álvaro que me han acogido desde el primer día haciéndome sentir como una más y ayudándome en todo lo que he necesitado. Y a mis jefes José Jorge García Romarís y Roberto López Rodriguez que han sido mi guía en todo este proceso.

Índice general

Resumen xi

1. Introducción 1

2. Tratamiento de datos nulos 3
2.1. Mecanismos de generación de datos faltantes . . . 3
2.1.1. Missing Completely at Random (MCAR) . . . 4
2.1.2. Missing at Random (MAR) . . . 4
2.1.3. Missing Not at Random (MNAR) . . . 5
2.2. Revisión teórica de métodos clásicos . . . 5
2.2.1. Reglas de integridad y validación estructural . . . 6
2.2.2. Diccionarios de referencia y listas de validación . . . 8
2.2.3. Dependencias funcionales y validación relacional . . . 9
2.2.4. Análisis comparativo y justificación de enfoques avanzados . . . 9

3. Modelo HoloClean 11
3.1. Introducción al método . . . 12
3.2. Formulación del problema . . . 12
3.3. Optimización del modelo . . . 13
3.3.1. Grafo de factores . . . 13
3.3.2. Relajación de restricciones duras . . . 14
3.3.3. Poda de dominios . . . 14
3.3.4. Agrupación de tuplas . . . 15
3.3.5. Muestreo de Gibbs . . . 15
3.3.6. Compilación a programas probabilísticos . . . 16
3.4. Consideraciones finales sobre el modelo . . . 16

4. Detección de anomalías en series temporales univariantes 19
4.1. Introducción y motivación . . . 19
4.2. Conceptos básicos . . . 20
4.2.1. Definición y características de una serie temporal . . . 20
4.2.2. Tipos de anomalías . . . 22
4.2.3. Métodos para detección de anomalías . . . 24
4.2.4. Métricas para la validación de las predicciones . . . 26
4.3. Presentación de modelos para series temporales univariantes . . . 27
4.4. Modelo ARIMA . . . 28
4.4.1. Formulación del modelo . . . 28
4.4.2. Hipótesis del modelo . . . 29
4.4.3. Ajuste del modelo . . . 29
4.5. Modelos basados en árboles de decisión: bagging y boosting . . . 32
4.5.1. Árboles de decisión . . . 32

4.5.2. Bagging: Random Forest . . . 33
4.5.3. Boosting . . . 34
4.5.4. Aplicación a series temporales univariantes . . . 35
4.5.5. Ajuste del modelo y validación . . . 36
4.6. Modelo Encoder-Decoder . . . 37
4.6.1. Fundamentos de redes neuronales y arquitectura LSTM . . . 37
4.6.2. Descripción teórica del modelo Encoder-Decoder . . . 40
4.6.3. Aplicación a series temporales univariantes . . . 42
4.7. Consideraciones finales . . . 43

5. Aplicación práctica de los métodos **45**
5.1. Imputación de datos nulos: Modelo HoloClean . . . 45
5.1.1. Presentación del conjunto de datos . . . 45
5.1.2. Preprocesado de los datos . . . 46
5.1.3. Aplicación del modelo HoloClean . . . 47
5.1.4. Resultados de la imputación . . . 49
5.1.5. Conclusiones . . . 51
5.2. Detección de anomalías en series temporales . . . 51
5.2.1. Presentación y tratamiento del conjunto de datos . . . 52
5.2.2. Metodología general . . . 53
5.2.3. Modelo ARIMA . . . 55
5.2.4. Modelo LightGBM . . . 61
5.2.5. Modelo Encoder-Decoder . . . 65
5.2.6. Comparación de resultados y elección del modelo . . . 68

6. Conclusiones finales y lineas futuras de investigación **71**

Bibliografía **73**

Resumen

Resumen en español

En este Trabajo de Fin de Máster se tratan dos cuestiones claves en el ámbito de la calidad de los datos; la detección y corrección, tanto de valores nulos como de anomalías en un conjunto de datos. En primer lugar se presentan algunos métodos clásicos para el tratamiento de valores faltantes como son el uso de diccionarios de referencia, las reglas de integridad o las dependencias funcionales. Sus limitaciones justifican la introducción de un modelo probabilístico mas reciente, denominado HoloClean, que permite una corrección mas precisa y automática de los errores integrando múltiples señales. Se demostrará la precisión de este método aplicándolo a un conjunto de datos proporcionado por la entidad financiera ABANCA, empresa con la cual se ha desarrollado el proyecto.

A continuación se estudiará la detección de anomalías en series temporales. Se desarrollarán los fundamentos teóricos de tres métodos: un modelo estadístico clásico (Arima), un algoritmo de aprendizaje automático basado en árboles de decisión y un modelo de aprendizaje profundo (Encoder-Decoder). Finalmente estos modelos se aplicarán a una serie temporal, cuyo estudio es fundamental para la empresa ABANCA, con el objetivo de evaluar su capacidad para identificar anomalías.

English abstract

This Master's Thesis addresses two key issues in the field of data quality: the detection and correction of both missing values and anomalies in a dataset. First, several classical methods for handling missing data are presented, including the use of reference dictionaries, integrity rules, and functional dependencies. The limitations of these approaches justify the introduction of a more recent probabilistic model, known as HoloClean, which enables more accurate and automated error correction by integrating multiple signals. The effectiveness of this model is demonstrated through its application to a dataset provided by the financial institution ABANCA, with whom the project was developed.

Subsequently, the focus shifts to the detection of anomalies in univariate time series. The theoretical foundations of three methods are developed: a classical statistical model (ARIMA), a machine learning algorithm based on decision trees, and a deep learning model (Encoder-Decoder). Finally, these models are applied to a time series that are of critical operational importance to ABANCA, with the aim of evaluating their ability to accurately identify anomalies.

Capítulo 1

Introducción

En la actualidad, la calidad de un conjunto de datos se ha consolidado como un factor crítico tanto en el desarrollo de modelos predictivos como en la toma de decisiones basadas en ellos. El aumento del número de fuentes de datos heterogéneas y el creciente volumen de información han intensificado los desafíos relacionados con las distintas características de los datos como la completitud, la unicidad, la precisión o la coherencia, entre otras (Batini et al. 2009; Pipino et al. 2002).

Este proyecto se ha realizado en colaboración con la entidad financiera ABANCA CORPORACIÓN BANCARIA S.A., el primer banco en Galicia por cuota de mercado. La entidad tiene presencia en 11 países de Europa y América, contando con alrededor de 6000 empleados al servicio de más de dos millones de clientes (ABANCA s.f.). Para la empresa asegurar la calidad de sus bases de datos es un proceso crítico. La toma de decisiones del banco se basa en la información que maneja, por lo que es fundamental que ésta sea lo más fiable posible. Muchas condiciones pueden influir en gran medida en la información almacenada por la empresa como continuas incorporaciones de nuevos clientes, los cambios en el mercado o las múltiples integraciones que la entidad ha realizado en los últimos años. Es por ello que contar con sistemas robustos para asegurar la calidad y validez de sus datos es fundamental.

Tras una consulta a los responsables del departamento de Calidad de Datos de la empresa, se determinó que en la actualidad contaban con dos cuestiones que requerían de su especial atención, las cuales se decidió abordar en este Trabajo de Fin de Máster. Dentro del campo de la calidad de datos, el proyecto se centra en dos factores claves relacionados con la misma: la detección y corrección, tanto de valores faltantes como de anomalías.

Esto se debe a que la primera de las necesidades de la empresa era depurar un dataset que cuenta con información de carácter catastral. Estos datos tienen especial importancia para ellos pues son críticos en el ámbito de la gestión hipotecaria. Las referencias catastrales que aparecen en dicho dataset son aquellas que identifican los bienes que los clientes aportan como garantía en una solicitud de financiación, uno de los procesos críticos del banco. Por lo que garantizar la calidad de este conjunto es crucial para la posterior toma de decisiones en el proceso.

El principal problema de estos datos era que contaban con un porcentaje muy elevado de valores faltantes por lo que la primera parte de este trabajo se centrará en la detección y corrección de dichos valores. Se darán argumentos sólidos que explican el motivo de tratar este problema y se expondrán diferentes métodos de detección e imputación centrados en un tipo de datos en específico, los datos alfanuméricos, pues son los que aparecen en el dataset a tratar. Se revisarán los tipos de errores que se pueden encontrar en los datos alfanuméricos, como tipográficos, de formato o de consistencia entre otros. Para a continuación, realizar una revisión de las técnicas clásicas para la detección y corrección de errores como el uso de diccionarios de referencia, la validación basada en reglas o las dependencias

funcionales. Finalmente se presentará un modelo probabilístico más reciente, denominado HoloClean, que permite una detección y corrección más precisa y automática de los errores.

El tratamiento de los datos faltantes es un paso fundamental previo a la aplicación de algoritmos de detección de anomalías ya que se ha demostrado que estos registros pueden influir de forma significativa a los análisis posteriores que se realicen sobre un dataset (Little y Rubin 2002). Es por ello que este primer campo tratado dentro de la calidad de datos nos lleva al siguiente problema planteado por la empresa, la detección de anomalías.

Cada día laborable del año se carga la información de la tabla, CONTRATO_POSICION_DIA, que contiene la información diaria de un contrato en términos de saldos, divisas, clasificación contable y otros atributos financieros, así mismo, en otra tabla llamada CONTRATO se carga información complementaria que se utiliza principalmente para distinguir la aplicación de la que proviene cada contrato. La empresa cuenta con más de 100 aplicaciones diferentes, sin embargo, hay varias de ellas que destacan por su importancia para la entidad pues son las que cuentan con un mayor volumen de contratos asociados. Para cada una de estas aplicaciones se tiene una serie temporal asociada que mide el número de registros cargados cada día. Se ha elegido una de estas series para centrar el estudio, pudiendo replicar más adelante los resultados a las demás.

El objetivo del estudio será encontrar un modelo que detecte cualquier tipo de anomalía que se presente en esta serie temporal. Dicho modelo será incorporado por la empresa a los múltiples controles de calidad diarios que ya están siendo aplicados en todo tipo de ámbitos.

Para abordar este reto se desarrollarán diversos enfoques que combinan técnicas estadísticas tradicionales, algoritmos de aprendizaje automático y arquitecturas de redes neuronales profundas. Se compararán tres modelos representativos y complementarios para la detección de anomalías en series temporales: Arima, árboles de decisión y Encoder-Decoder. Y se elegirá el mejor de ellos para el caso de estudio.

Capítulo 2

Tratamiento de datos nulos

Uno de los principales desafíos en el análisis de conjuntos de datos es la presencia de valores nulos. Este problema se ha estudiado ampliamente en la literatura (Little y Rubin 2002; Schafer 1997; Batista y Monard 2003) puesto que se ha demostrado que la presencia de valores ausentes puede afectar de forma significativa a la validez de los análisis estadísticos que se realicen sobre un conjunto de datos, así como a los modelos predictivos que se apliquen sobre el mismo (Little y Rubin 2002). Por tanto, la correcta identificación de dichos registros y su posterior tratamiento es un paso prioritario y fundamental en cualquier proceso de análisis de datos.

El objetivo de este capítulo será, en primer lugar, realizar una revisión de los mecanismos que explican la aparición de datos faltantes, para a continuación, presentar algunos de los métodos clásicos más utilizados para la detección y posterior corrección de dichos registros. Las debilidades que presentarán estos métodos dará una justificación para la adopción de enfoques más avanzados que se tratarán en secciones posteriores.

2.1. Mecanismos de generación de datos faltantes

En una base de datos relacional, un dato faltante o valor nulo se define como la ausencia de un valor observado para una variable (Little y Rubin 2002). Conceptualmente, ese valor existe, pero no ha sido registrado. Es importante destacar que un valor nulo no es equivalente a cero, una cadena vacía ni a ningún otro valor explícito; su significado está asociado a una falta de información.

En la Tabla 2.1 se puede observar una muestra de un conjunto de datos formado por cuatro variables: nombre, sexo, edad y número de hijos; el cual contiene datos faltantes. No se cuenta con información del sexo de Andrea, el número de hijos de Daniel y Ruth ni tampoco con la edad de esta última. Es importante destacar, tal y como se muestra en la tabla, que un valor nulo puede estar representado en un dataset de diversas formas, sin embargo todas ellas representan una falta de información. El cero que aparece en el número de hijos de Andrea no puede considerarse un valor nulo, pues en efecto está proporcionando información sobre la variable.

La comprensión de los mecanismos que explican por qué un valor está ausente en un conjunto de datos es esencial para tomar decisiones metodológicas adecuadas. Estos mecanismos, presentados formalmente en Rubin (1976), se clasifican en tres categorías principales: *Missing Completely at Random* (MCAR), *Missing at Random* (MAR) y *Missing Not at Random* (MNAR). Saber identificar cuál de estos mecanismos ha generado los datos faltantes en la práctica tiene importantes consecuencias, pues determina las estrategias de imputación que se deben seguir para no añadir sesgos significativos.

Nombre	Sexo	Edad	Número de hijos
Paula	Mujer	35	2
Andrea	-	27	0
Daniel	Hombre	32	
Ruth	Mujer		NULL

Tabla 2.1: Muestra de un dataset con presencia de datos nulos.

Se presentan a continuación las principales características de estos mecanismos, las cuales se pueden encontrar en Little y Rubin (2002).

2.1.1. Missing Completely at Random (MCAR)

Un conjunto de datos cumple el mecanismo MCAR cuando la probabilidad de que una observación esté ausente es completamente independiente tanto de los valores observados, como de los no observados.

Formalmente, sea $Y = (y_{ij})$ el conjunto de estudio y $M = (M_{ij})$ una matriz indicadora tal que $M_{ij} = 1$ cuando el valor y_{ij} está ausente y $M_{ij} = 0$ en cualquier otro caso. Se dice que los datos son MCAR si: $P(M \mid Y) = P(M)$.

Esto significa que la ausencia de información ocurre de forma completamente aleatoria, sin relación con ninguna característica del dataset. Por ejemplo, si al ingresar en el sistema la información recabada en una encuesta algunas respuestas se pierden debido a un fallo técnico aleatorio, estaríamos ante un caso MCAR.

Este es el escenario más favorable desde el punto de vista del análisis estadístico, ya que los análisis realizados sobre los datos observados siguen siendo representativos de la población completa. Métodos como la eliminación de casos incompletos o la imputación simple (por la media, moda...) no introducen sesgos bajo este supuesto (Little y Rubin 2002).

Existen diversos tests para evaluar la veracidad del supuesto MCAR en la práctica. El test de Little (Little 1988), por ejemplo, es una prueba estadística formal para conjuntos de datos multivariantes continuos que compara las medias y covarianzas de las variables observadas entre los distintos patrones de ausencia. Bajo la hipótesis nula de MCAR, se espera que estas estadísticas sean consistentes entre patrones, ya que la ausencia de datos no debería depender de las variables. Así mismo, para verificar el supuesto MCAR en datos categóricos, se pueden analizar tablas de contingencia entre el patrón de datos faltantes y las variables observadas, aplicando pruebas de independencia como el test chi-cuadrado. Se puede ampliar la información de este método en Enders (2010).

2.1.2. Missing at Random (MAR)

Se dice que los datos son MAR cuando la probabilidad de ausencia depende de los valores observados, pero no de los valores faltantes en sí. Matemáticamente: $P(M \mid Y) = P(M \mid Y_{observado})$.

Por ejemplo, supongamos que las personas con menos nivel educativo tienden a no reportar su

ingreso. Si el nivel educativo está disponible, entonces la ausencia del ingreso puede explicarse por una variable observada. En este caso, se pueden aplicar técnicas de imputación como regresión, imputación múltiple o modelos basados en inferencia bayesiana sin introducir sesgos, siempre y cuando se utilicen correctamente las variables explicativas disponibles (Schafer 1997; Rubin 1987).

En la práctica, MAR es un supuesto más realista que MCAR, aunque difícil de verificar empíricamente. A diferencia de MCAR, no existe un test formal para MAR porque su definición depende de los valores faltantes no observados. Se puede evaluar la plausibilidad de MAR explorando las relaciones entre el patrón de ausencia de los datos y las variables observadas, por ejemplo utilizando modelos logísticos, o también se puede recurrir a argumentos teóricos sobre el diseño del estudio (Little y Rubin 2002). Para que las técnicas bajo MAR sean válidas, es necesario que se incluyan todas las variables relacionadas con el mecanismo de omisión en el modelo de imputación.

2.1.3. Missing Not at Random (MNAR)

Los datos son MNAR cuando la probabilidad de que una observación esté ausente depende de su propio valor no observado o de otras variables también no observadas. Es decir:
$P(M \mid Y)$ depende de $Y_{faltante}$.

Este mecanismo es el más problemático, ya que implica que la falta de información está correlacionada con lo que se intenta medir. Por ejemplo, si las personas con ingresos muy altos deciden no declarar sus ingresos por motivos de privacidad, entonces la ausencia del dato está directamente relacionada con el valor que falta.

Al igual que en el caso MAR, no existe un test estadístico formal para verificar este supuesto en los datos, ya que de nuevo su definición depende de valores no observados.

Bajo el supuesto MNAR, no es posible imputar los datos sin hacer suposiciones adicionales sobre el proceso de generación de ausencias. Algunos de los métodos más utilizados en este caso son los modelos de selección, como el modelo de Heckman (Heckman 1979), o el análisis de sensibilidad que explora diferentes supuestos de ausencia no al azar (Molenberghs y Kenward 2007). Estos enfoques suelen ser complejos, menos robustos y requieren experiencia estadística avanzada para su correcta implementación.

La clasificación que se acaba de presentar es de vital importancia pues determina la viabilidad de aplicar distintos métodos de análisis y de imputación de valores faltantes. La correcta identificación del mecanismo subyacente resulta esencial para seleccionar una estrategia de tratamiento adecuada. Sin embargo, como ya se ha explicado, no siempre es posible determinar de manera concluyente en la práctica cuál de los mecanismos de generación de datos faltantes sigue el conjunto de estudio.

2.2. Revisión teórica de métodos clásicos

Una vez que se han presentado los mecanismos de generación de datos faltantes, en esta sección se van a describir en detalle varios métodos clásicos empleados en la práctica para detectar e imputar valores nulos en bases de datos alfanuméricos. Estos métodos han sido tradicionalmente utilizados debido a su simplicidad y facilidad de implementación, aunque se verá que presentan limitaciones importantes en escenarios complejos o con datos ruidosos.

Este trabajo se centrará principalmente en el desarrollo de métodos aplicables a datos alfanuméricos, pues son aquellos que aparecen en el dataset que la empresa ABANCA tenía la necesidad de depurar. Es por ello que en el presente documento no se expondrá ningún método que no sea aplicable

a este tipo de datos.

Los datos alfanuméricos consisten en secuencias de caracteres que pueden incluir letras (mayúsculas y minúsculas), dígitos y símbolos especiales. A diferencia de los datos numéricos, estos no están destinados a cálculos aritméticos, sino que suelen representar códigos, descripciones, identificadores u otra información textual. En minería de datos, se modelan habitualmente como atributos nominales, que representan categorías sin orden inherente (Han et al. 2012).

A continuación se presentan algunos ejemplos de este tipo de datos:

- Códigos postales: `36013`, `17011`
- Números de DNI: `99853284A`, `12245509X`
- Matrículas de vehículo: `3452AGB`, `0981CPL`
- Identificadores de producto: `REF-A23`, `PROD-001`

Los datos alfanuméricos son propensos a errores de diversa naturaleza que pueden comprometer la calidad de los mismos. Algunos de los tipos de errores más habituales que se pueden encontrar son (Rahm y Do 2000):

- **Errores tipográficos**: Fallos que se pueden producir al introducir los datos en el sistema de manera manual y que incluyen, sustituciones, adiciones, omisiones o trasposiciones de caracteres. Por ejemplo: `A3452GB` en lugar de `3452AGB` en una matrícula.
- **Errores de codificación o formato**: Los datos no cumplen con el formato marcado. Por ejemplo, el código postal `3601` no cumple con la longitud esperada de 5 dígitos.
- **Errores de consistencia**: Los valores no coinciden en diferentes registros de un dataset. Por ejemplo, en una entrada del conjunto de datos el identificador de producto aparece como `REF-A23` y en otra `REFA23`.

La naturaleza textual de estos datos complica el proceso de imputación, ya que no se pueden aplicar directamente técnicas como la imputación por la media o mediana ni tampoco la interpolación lineal. Por ello, es frecuente recurrir a métodos específicos como los diccionarios de referencia o la validación basada en reglas. Aunque presentan limitaciones frente a métodos estadísticos o de aprendizaje automático, siguen siendo fundamentales en contextos donde se requiere explicabilidad, bajo tiempo de ejecución o donde los datos son estructurados pero no complejos.

A continuación se presenta una amplia y detallada revisión de algunos de los métodos clásicos más utilizados para la detección y corrección de valores nulos o inconsistentes en conjuntos de datos alfanuméricos, los cuales han sido ampliamente documentados en la literatura sobre limpieza de datos y calidad de la información (Rahm y Do 2000; Han et al. 2012; Batini y Scannapieca 2006).

2.2.1. Reglas de integridad y validación estructural

Las reglas de integridad y validación estructural son ampliamente utilizadas en bases de datos relacionales, por lo que han sido tratadas en múltiples publicaciones técnicas (Date 2004; Elmasri y Navathe 2015).

Estas reglas son mecanismos declarativos que imponen restricciones lógicas sobre los valores admisibles en un conjunto de datos relacional. Su objetivo principal es prevenir la inserción de registros incompletos o inconsistentes.

Sea D el conjunto de datos relacional compuesto por tuplas $t \in D$, y sea A_i un atributo (o columna) específico del esquema de datos. Cada tupla t es una instancia que asigna un valor a cada atributo del esquema, de modo que $t[A_i]$ denota el valor de A_i en la tupla t.

Las reglas de integridad se clasifican típicamente en:

- **Restricciones de `NOT NULL`**: impiden la presencia de valores nulos en campos obligatorios. Formalmente, para un atributo A_i:

  $\forall t \in D,\ t[A_i] \neq \text{NULL}$.

- **Restricciones de formato**: validan que el valor de un atributo siga un patrón predefinido. Por ejemplo, para el atributo `CODIGO_POSTAL`, se puede exigir que sea una cadena de exactamente cinco dígitos:

  $t[\text{CODIGO_POSTAL}] \in \{s \in \mathbb{R}^5 \mid s_i \in \{0, \ldots, 9\}\ \forall i = 1, \ldots, 5\}$.

- **Restricciones de dominio**: limitan los valores posibles de un atributo a un conjunto predefinido V:

  $\forall t \in D,\ t[A_i] \in V$.

Estas validaciones son computacionalmente eficientes en la mayoría de los casos, y son fáciles de implementar en bases de datos relacionales mediante DDL (*Data Definition Language*).

El término DDL (*Data Definition Language*) se refiere al subconjunto del lenguaje SQL utilizado para definir y modificar la estructura de la base de datos (Date 2004; Elmasri y Navathe 2015). Se diferencia del DML (*Data Manipulation Language*) en que define la estructura y las restricciones del esquema, mientras que el DML se usa para consultar y modificar los datos almacenados. Incluye comandos como:

- `CREATE TABLE`: define nuevas tablas y sus columnas, incluyendo restricciones de integridad.
- `ALTER TABLE`: modifica la estructura existente de una tabla (agregar o eliminar columnas, cambiar restricciones).
- `DROP TABLE`: elimina una tabla completa de la base de datos.

Las restricciones de integridad se especifican directamente en las sentencias DDL al momento de definir el esquema de la base de datos, lo que permite que el propio sistema de gestión de bases de datos verifique automáticamente la validez de los datos durante inserciones y actualizaciones.

Por ejemplo, la definición de una tabla en SQL con restricciones de integridad podría escribirse así:

```
CREATE TABLE Direcciones (
CODIGO_POSTAL CHAR(5) NOT NULL,
PROVINCIA_ID INT NOT NULL,
CONSTRAINT chk_codigo_postal CHECK (
CODIGO_POSTAL ~ '^[0-9]{5}$'
)
);
```

En este ejemplo:

- `CREATE TABLE Direcciones`: crea una nueva tabla llamada `Direcciones`.
- `CODIGO_POSTAL CHAR(5) NOT NULL`: define una columna de texto de longitud fija 5, que no puede ser nula.

- `PROVINCIA_ID INT NOT NULL`: define una columna numérica entera, también obligatoria.
- `CONSTRAINT chk_codigo_postal CHECK (...)`: define una restricción llamada `chk_codigo_postal` que obliga a que los valores de `CODIGO_POSTAL` sean exactamente 5 dígitos del 0 al 9.

De este modo, el sistema de gestión de bases de datos rechaza automáticamente cualquier inserción o actualización que no cumpla con estas condiciones, asegurando la calidad estructural de los datos desde el origen.

A pesar de todo lo descrito anteriormente, las reglas de integridad tienen ciertas limitaciones significativas. Solo permiten detectar errores o valores faltantes, pero no proponen correcciones automáticas; así mismo, son reglas estáticas que requieren un mantenimiento manual y actualización frecuente. Además también podrían generar falsos positivos si se definen de forma demasiado rígida o no contemplan excepciones válidas.

2.2.2. Diccionarios de referencia y listas de validación

Los diccionarios de referencia son bases de datos auxiliares que contienen los valores válidos o autorizados de uno o varios atributos de un conjunto de datos. Se utilizan como fuentes externas de validación o para imputar valores faltantes o incorrectos.

Se define $R \subseteq \Sigma$ como el conjunto (o diccionario) de valores válidos para el atributo A_i, donde Σ es el conjunto de todas las cadenas de caracteres posibles.

La validación con diccionarios se expresa formalmente como: $\forall t \in D,\ t[A_i] \in R$. Es decir, para cada registro t, el valor del atributo A_i debe encontrarse dentro del conjunto de valores autorizados R.

Cuando se detecta un valor nulo o inválido, se puede realizar imputación seleccionando el valor más cercano en R según alguna función de distancia. Sea $\text{Dist}(\cdot, \cdot)$ una métrica de similitud, la imputación se formula como:

$$\hat{v} = \arg\min_{v \in R} \text{Dist}(t[A_i], v),$$

donde $\hat{v}$ es el valor imputado elegido del diccionario R que minimiza la distancia con el valor observado o faltante.

La distancia de edicion o distancia de Levenshtein, introducida por Levenshtein (1966), es uno de los algoritmos más utilizados a la hora de comparar cadenas alfanuméricas. Considerando s_1 y s_2 dos cadenas de caracteres de longitudes no necesariamente iguales, la distancia de Levenshtein entre s_1 y s_2 se define como el número mínimo de operaciones elementales necesarias para transformar una cadena en la otra. Dichas operaciones elementales son las siguientes:

- Eliminacion: se borra un caracter de la cadena reduciendo la longitud de la misma.
- Inserción: se añade un caracter a la cadena, en cualquier posición, aumentando la longitud de la misma.
- Substitución: se intercambia un caracter por otro cualquiera manteniéndose la longitud de la cadena.

Este método, a diferencia de las reglas de integridad, no solo permite detectar valores faltantes sino que también aporta un algoritmo para realizar imputaciones. Sin embargo, el uso de diccionarios de referencia cuenta con una limitación importante, y es que en este método hay una dependencia total de la información recogida en el diccionario por lo que es necesario un mantenimiento continuo del mismo para asegurar la calidad de los resultados.

2.2.3. Dependencias funcionales y validación relacional

Las dependencias funcionales son restricciones semánticas que expresan relaciones deterministas entre atributos de una base de datos relacional. Se definen formalmente de la siguiente forma:

$$X \to Y \iff \forall t_1, t_2 \in D,\ t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y],$$

donde D es el conjunto de datos relacional, t_1 y t_2 son tuplas (filas) de D, X e Y son conjuntos de atributos (columnas) del esquema de la tabla y $t[X]$ denota la proyección de la tupla t sobre los atributos X.

Por tanto, una dependencia funcional establece que el valor de X determina de forma única el valor de Y.

Como ejemplo, en un conjunto de direcciones, la dependencia funcional
CODIGO_POSTAL $\to$ PROVINCIA_ID indica que para cada código postal válido existe una única provincia asociada.

Las dependencias funcionales se utilizan para detectar errores en un dataset, de modo que si en el conjunto D se encuentran dos tuplas con el mismo valor de X pero distinto valor de Y (formalmente $\exists t_1, t_2 \in D\ :\ t_1[X] = t_2[X] \wedge t_1[Y] \neq t_2[Y]$), se tendría un error potencial.

Este método también se puede utilizar para imputar valores faltantes de modo que si $t[X]$ está presente y $t[Y] = \text{NULL}$, entonces $t[\hat{Y}] = f(X)$, donde f se define como la asignación más frecuente observada en el conjunto de datos: $f(X) = \arg\max_y \ \text{count}(X = x, Y = y)$.

De nuevo, este método proporciona mecanismos tanto para detectar como para corregir valores nulos; sin embargo, cuenta con limitaciones importantes. Las restricciones son estáticas y requieren ser definidas y mantenidas por expertos en el dominio. Además, Bohannon et al. (2005) demuestran que el problema de encontrar una reparación de costo mínimo mediante modificación de valores es NP-completo; es decir, no existe, hasta donde se sabe, un algoritmo eficiente (en tiempo polinomial) que resuelva el problema de forma exacta en todos los casos.

Por estas razones, aunque las dependencias funcionales son esenciales para garantizar integridad semántica, su uso exclusivo no es suficiente para resolver todos los problemas de calidad de datos.

2.2.4. Análisis comparativo y justificación de enfoques avanzados

Para evaluar las capacidades de los métodos clásicos en la detección y corrección de valores nulos o inconsistentes, es útil compararlos en términos de tres dimensiones clave: capacidad de detección, capacidad de imputación y adaptabilidad al contexto. La Tabla 2.2 resume estas características para los principales enfoques revisados.

Método	**Detección**	**Imputación**	**Adaptabilidad**
Reglas de integridad	Alta	No	Baja
Diccionarios externos	Alta	Sí	Baja
Dependencias funcionales	Alta	Sí	Baja

Tabla 2.2: Comparativa de métodos clásicos para la detección y corrección de datos faltantes.

La tabla evidencia que los métodos clásicos, aunque efectivos en la detección cuando las reglas están bien definidas, presentan limitaciones notables en otros aspectos.

- **Detección**: confiable cuando las reglas y catálogos están bien definidos, pero dependiente de su calidad y mantenimiento.
- **Imputación**: limitada y frecuentemente heurística o estática, sin considerar incertidumbre estadística.
- **Adaptabilidad**: métodos mayoritariamente estáticos, sensibles a excepciones, valores atípicos y cambios en el dominio.

Estas limitaciones motivan el desarrollo de enfoques avanzados basados en modelos probabilísticos y aprendizaje automático que puedan integrar múltiples fuentes de evidencia, permitan imputaciones con incertidumbre cuantificada o aprendan automáticamente patrones y relaciones a partir de los datos históricos.

En este contexto, modelos como HoloClean, el cual se presentará en la próxima sección, representan un avance significativo. Al formular la limpieza de datos como un problema de inferencia probabilística, HoloClean combina los métodos comentados en esta sección en un marco unificado y escalable, logrando una imputación más precisa, robusta y automatizada (Rekatsinas et al. 2017).

En síntesis, aunque los métodos clásicos resultan esenciales como primera línea de defensa en la calidad de los datos, sus limitaciones justifican la adopción de enfoques más avanzados y adaptativos. El próximo capítulo explorará en profundidad uno de estos enfoques innovadores.

Capítulo 3

Modelo HoloClean

En este capítulo se presenta el modelo HoloClean (Rekatsinas et al. 2017), un enfoque avanzado y holístico para la detección y corrección de valores erróneos o faltantes en bases de datos estructuradas.

El término holístico hace referencia a la capacidad del modelo de integrar de forma conjunta y coherente múltiples fuentes de evidencia heterogéneas. En lugar de aplicar reglas aisladas o procedimientos independientes, HoloClean combina restricciones de integridad, conocimiento externo (como diccionarios) y patrones estadísticos extraídos directamente de los datos para realizar imputaciones consistentes de forma automatizada. Esta visión permite capturar dependencias complejas y explotar toda la información disponible de manera unificada.

El método HoloClean surge para dar respuesta a las limitaciones de los métodos tradicionales de limpieza de datos, que se caracterizan por:

- Aplicar reglas estáticas, difíciles de mantener y que requieren intervención manual constante.
- Carecer de capacidad adaptativa, es decir, no aprender automáticamente de los datos históricos ni ajustarse a nuevas excepciones o patrones.
- Estrategias de imputación heurísticas o simplistas, que no consideran la incertidumbre estadística ni relaciones de dependencia complejas entre atributos.
- Abordar cada fuente de error de manera aislada, sin unificar la evidencia proveniente de restricciones, coocurrencias y conocimiento externo en un modelo único.

HoloClean nació en el contexto de la era del Big Data, donde las bases de datos relacionales y semiestructuradas crecen en volumen, heterogeneidad y complejidad. Gestionar la calidad de los datos de forma manual resulta cada vez más inviable, sobre todo en sectores como las finanzas, la salud o el comercio electrónico, donde se toman decisiones importantes basadas en datos que deberían ser fiables y precisos. Para afrontar estos problemas, es fundamental disponer de sistemas automáticos y escalables que sean capaces de:

- Representar de forma explícita la incertidumbre presente en datos incompletos o con ruido.
- Manejar conjuntos de datos con millones de registros y miles de atributos.
- Generar imputaciones que no solo sean precisas, sino también explicables y basadas en modelos probabilísticos bien fundamentados.

En este capítulo se desarrollarán las bases teóricas de HoloClean, describiendo su formulación matemática, su arquitectura de sistema y las técnicas de optimización que lo convierten en una herramienta potente para el aseguramiento de la calidad de datos en entornos reales.

3.1. Introducción al método

En los últimos años, la forma de tratar errores en bases de datos estructuradas ha pasado de usar únicamente reglas fijas o comparaciones con fuentes externas, como se explicó en el capítulo anterior, a adoptar enfoques más completos e integradores. HoloClean es un ejemplo destacado de este cambio, ya que propone una estrategia holística para la limpieza de datos y formula el problema de la corrección como uno de inferencia probabilística (Rekatsinas et al. 2017). Lo interesante de este enfoque es que no solo se limita a imputar un valor faltante o erróneo, sino que también permite estimar la incertidumbre asociada a cada predicción. Este aspecto es esencial en aplicaciones críticas, donde es necesario conocer la incertidumbre de una imputación para tomar decisiones informadas.

A diferencia de los métodos más tradicionales, que suelen basarse en un único tipo de señal (como restricciones de integridad o diccionarios externos), HoloClean combina diferentes fuentes de información en un único modelo probabilístico. Su diseño se basa en tres pilares fundamentales: restricciones de integridad, conocimiento externo y estadísticas cuantitativas derivadas de los propios datos. Esta combinación le permite ofrecer reparaciones más precisas, coherentes y adaptativas, incluso en contextos de datos ruidosos y heterogéneos.

3.2. Formulación del problema

Sea D un conjunto de datos definido sobre el conjunto de atributos $A = \{A_1, \ldots, A_N\}$. Cada tupla $t \in D$ contiene celdas de la forma $c = t[A_i]$. HoloClean parte de la premisa de que algunos valores observados v_c pueden ser incorrectos respecto de su valor verdadero v_c^*, que se busca estimar. El objetivo es obtener una estimación $\hat{v}_c$ lo más precisa posible para cada celda considerada sospechosa.

La Figura 3.1 ilustra el flujo de trabajo del modelo HoloClean (Rekatsinas et al. 2017), organizado en tres bloques principales: *Input*, *HoloClean Framework* y *Output*.

En la primera etapa, *Input*, se introduce el conjunto de datos D, que puede contener valores nulos o erróneos. Junto con los datos a limpiar, es posible incorporar diferentes fuentes de información adicionales, como restricciones de integridad, dependencias funcionales o diccionarios de referencia, representadas mediante el conjunto Σ.

El bloque central, *HoloClean Framework*, se subdivide en tres módulos: detección de errores, compilación y reparación.

- **Detección de errores**: En esta fase se aplican técnicas elegidas por el usuario para identificar registros con errores o valores faltantes. Normalmente se recurre a métodos tradicionales como los descritos en el capítulo anterior. El conjunto de datos D se divide entonces en D_n, que contiene los registros marcados como erróneos, y $D_c = D \backslash D_n$, compuesto por las entradas consideradas válidas.
- **Compilación**: Aquí se asigna a cada celda $c \in D$ una variable aleatoria T_c que toma valores en un dominio finito $dom(c)$. La relación entre estas variables aleatorias se modela mediante un grafo de factores, que describe su distribución conjunta.
- **Reparación**: En esta etapa se emplean técnicas de aprendizaje estadístico e inferencia para estimar la distribución conjunta de las variables $T_1, T_2, \ldots$, y así calcular la probabilidad marginal $P(T_c = d)$ para cada valor $d \in \text{dom}(c)$. Las celdas de D_c se utilizan como datos etiquetados para ajustar los parámetros del modelo, mientras que las celdas de D_n se tratan como variables de consulta cuyo valor debe inferirse. Para realizar la inferencia aproximada se utiliza muestreo de Gibbs.

Por último, el bloque *Output* presenta los resultados del proceso. Además del conjunto de datos con las celdas corregidas, se obtiene la distribución de probabilidad asociada a cada celda. Esta información proporciona no solo los valores imputados, sino también las probabilidades estimadas para cada posible valor, lo que permite evaluar la confianza en las reparaciones realizadas.

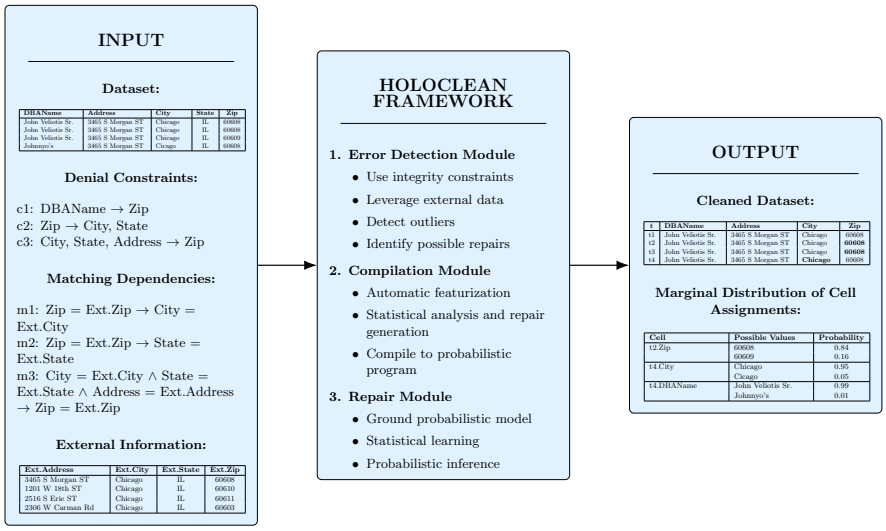


Figura 3.1: Flujo del modelo HoloClean.

3.3. Optimización del modelo

Una vez se ha presentado el flujo principal del modelo, esta sección explicará con detalle las técnicas más importantes que incorpora. Así mismo, se explorarán los retos fundamentales que han tenido que solventarse para contar con un modelo eficaz y robusto.

3.3.1. Grafo de factores

La complejidad del método radica en que los valores de las celdas del dataset no son independientes. Las restricciones de integridad, las coocurrencias estadísticas y la información externa introducen dependencias que deben modelarse explícitamente. Para capturar estas interacciones de forma rigurosa, HoloClean emplea un enfoque basado en grafos de factores, concepto introducido por Kschischang et al. (2001). Este modelo permite representar de manera compacta y eficiente las relaciones entre variables y la incertidumbre sobre los valores correctos.

Un grafo de factores (T, F, θ) está compuesto por un conjunto de nodos T (variables aleatorias) y otro de factores F. Cada factor $\phi \in F$ tiene asociada una función h_ϕ que relaciona subconjuntos de T mediante un peso θ_ϕ (Rekatsinas et al. 2017).

La distribución conjunta de las variables se modela como:

$$P(T) = \frac{1}{Z} \exp\left(\sum_{\phi \in F} \theta_\phi \cdot h_\phi(T_\phi)\right),$$

donde Z representa una constante que asegura la validez de la distribución y h_ϕ son las funciones que definen los factores. Estos factores permiten capturar relaciones como coocurrencias de valores, cumplimiento de reglas de integridad o coincidencias con diccionarios externos, integrando todas estas señales en un único marco probabilístico coherente.

Uno de los principales retos de HoloClean es la escalabilidad. La formulación probabilística detallada en el grafo de factores implica definir factores para todas las restricciones, coocurrencias y correspondencias externas, lo que conduce a una explosión combinatoria en datasets grandes. Si se modelan todas las interacciones posibles de manera explícita, el número de factores puede crecer de forma cuadrática o incluso cúbica con el número de tuplas, haciendo la inferencia intratable.

Para resolver este problema, HoloClean introduce un conjunto de técnicas de optimización destinadas a reducir el tamaño efectivo del espacio de búsqueda, la complejidad del grafo de factores y el coste de la inferencia, las cuales se tratan a continuación.

3.3.2. Relajación de restricciones duras

Todas las señales externas junto con las coocurrencias entre las distintas variables del dataset están modeladas como factores en el grafo de factores. Si estas señales se modelan como restricciones duras, es decir, el peso que tienen asociado es infinito, la inferencia se vuelve intratable, ya que el espacio de soluciones válidas puede ser muy reducido o incluso vacío por inconsistencias entre los datos.

Para superar este problema, HoloClean transforma estas restricciones en restricciones suaves, modeladas como factores con pesos finitos que penalizan pero no prohíben violaciones.

Esta relajación garantiza que la distribución posterior sea suave y continua.

3.3.3. Poda de dominios

Otra optimización importante que incluye el modelo es limitar el dominio de imputación de cada celda errónea a un conjunto reducido de valores plausibles.

Se presenta en primer lugar la forma en la que el modelo determina el dominio de las variables aleatorias T_c. Sea c una celda perteneciente a D_n, t la tupla a la que pertenece, c' una celda en t distinta de c, $v_{c'}$ su valor y $A_{c'}$ su atributo correspondiente. Se definen las reparaciones candidatas para c a todos los valores en el dominio del atributo de c, denotado A_c, que aparecen conjuntamente con el valor $v_{c'}$.

Sea $dom(c)$ el dominio original, se define un dominio podado $dom'(c) \subseteq dom(c)$ seleccionando solo aquellos valores que presentan coocurrencias significativas con los atributos observados (Rekatsinas et al. 2017).

Matemáticamente, la probabilidad condicional se estima mediante frecuencia empírica:

$$P(v \mid v') = \frac{\#(v, v')}{\#(v')},$$

donde $\#(v, v')$ es el número de veces que v y v' aparecen de forma conjunta en el dataset. Para elegir el dominio podado, se establece un umbral τ de probabilidad mínima tal que:

$$dom'(c) = \{v \in dom(c) \mid P(v \mid v') \geq \tau\}.$$

Esta poda reduce el número de valores candidatos por celda, disminuyendo así la complejidad de la inferencia. El valor de umbral τ, en la práctica, debe escogerse de modo que optimice los resultados del modelo. A mayores valores de τ, se consiguen dominios más pequeños junto con grafos de factores de menor tamaño lo que hace que la inferencia sea más eficiente, sin embargo, si el valor escogido es demasiado alto puede llevar a un modelo muy simple que no sea capaz de modelar correctamente toda la información del dataset junto con las señales externas.

3.3.4. Agrupación de tuplas

Uno de los principales desafíos en HoloClean es el tratamiento eficiente de las reglas lógicas derivadas de las restricciones de integridad. Al modelar el problema como un grafo de factores, cada celda potencialmente errónea se representa mediante una variable aleatoria, y las restricciones introducen dependencias que deben codificarse como factores en el grafo.

Aplicar todas las reglas sin restricciones requiere evaluar todos los pares de tuplas en el conjunto de datos D, lo que conlleva una alta complejidad. En la práctica, la mayoría de estas combinaciones son irrelevantes, ya que muchas tuplas no pueden violar las mismas restricciones debido a la naturaleza de sus atributos o valores. HoloClean resuelve este problema para hacer la generación de instancias de las reglas más escalable.

La idea central es restringir la generación de instancias de las reglas únicamente a grupos de tuplas que tengan una probabilidad significativa de violar alguna restricción de integridad. Para ello se construye un grafo de conflictos (Kolahi y Lakshmanan 2009) a partir de las violaciones detectadas en el dataset original.

Los nodos en el grafo de conflictos H corresponden a celdas que participan en violaciones detectadas, y las aristas conectan las celdas involucradas en la misma violación. Las aristas también se anotan con la restricción que generó la violación.

Para cada restricción específica, se considera el subgrafo de H que contiene únicamente las aristas correspondientes a violaciones de esa restricción. Sea H_φ este subgrafo inducido. Cada componente conexa en H_φ define un grupo de tuplas sobre el cual se materializará el factor correspondiente a la restricción.

En definitiva, este método permite reducir drásticamente el número de factores en el grafo por lo que posibilita la inferencia en datasets con un número elevado de registros.

3.3.5. Muestreo de Gibbs

El muestreo de Gibbs (Casella y George 1992) en HoloClean permite aproximar la distribución conjunta de todas las variables aleatorias que representan valores de celdas en el dataset. En cada paso, se selecciona una celda c y se actualiza su valor condicionada a todas las demás celdas:

$$P(T_c = v \mid T_{-c}) \propto \prod_{\phi \in F_c} \exp\left(\theta_\phi \cdot h_\phi(T'_\phi)\right),$$

donde F_c es el conjunto de factores que involucran a T_c y T'_ϕ representa la configuración al asignar $T_c = v$. Se calcula esta probabilidad para todos los valores posibles en el dominio de c, se normaliza y se muestrea un nuevo valor. Repetir este proceso para todas las celdas y durante múltiples iteraciones genera una cadena de Markov que converge a la distribución estacionaria del modelo, permitiendo estimar las distribuciones marginales y seleccionar las imputaciones más probables con una medida de

incertidumbre calibrada.

Como ya hemos visto, HoloClean implementa optimizaciones específicas para hacer el muestreo de Gibbs escalable a grandes datasets, como la descomposición del grafo de factores en componentes conexas más pequeñas (grafo de conflictos) y la relajación de restricciones para obtener modelos con independencia condicional entre variables. Estas optimizaciones reducen la complejidad de la inferencia y permiten realizar muestreo paralelo y distribuido en diferentes particiones del grafo.

3.3.6. Compilación a programas probabilísticos

Finalmente, HoloClean implementa toda esta especificación en un lenguaje declarativo llamado DDlog, compilando las restricciones y dependencias en un programa probabilístico que se ejecuta sobre el motor DeepDive (Zhang et al. 2017). Esto permite un diseño modular, reutilizable y optimizable, además de facilitar la paralelización y la ejecución en infraestructuras distribuidas.

En conjunto, estas técnicas de optimización convierten a HoloClean en un sistema capaz de escalar a millones de registros y miles de atributos, manteniendo la capacidad de ofrecer imputaciones precisas y probabilísticamente justificadas.

3.4. Consideraciones finales sobre el modelo

A lo largo de este capítulo se ha presentado HoloClean como un modelo de reparación de datos estructurados con un enfoque holístico y probabilístico. Su principal virtud radica en la capacidad de integrar distintas fuentes de evidencia, como restricciones de integridad, diccionarios externos y patrones de coocurrencia, dentro de un marco unificado. A diferencia de los métodos tradicionales, que tienden a aplicar estas señales por separado o a través de reglas fijas, HoloClean construye un grafo de factores que representa explícitamente las dependencias entre celdas, así como la incertidumbre asociada a cada posible valor.

Este modelo destaca por su versatilidad para combinar información heterogénea y generar imputaciones coherentes entre atributos y tuplas, lo cual permite mantener la consistencia global del conjunto de datos. Además, ofrece la posibilidad de cuantificar la incertidumbre de cada imputación mediante distribuciones de probabilidad, lo que resulta especialmente útil en contextos donde es necesario evaluar el grado de confianza en los datos corregidos. Su arquitectura modular y escalable, junto con el uso de técnicas como la poda de dominios, la partición del grafo de factores o la relajación de restricciones duras, facilita su aplicación en bases de datos reales de gran volumen y complejidad.

En comparación con los enfoques más clásicos, HoloClean evita depender exclusivamente de listas de validación cerradas o de reglas predefinidas. En su lugar, aprende directamente a partir de los datos las relaciones relevantes, permitiendo una mayor adaptabilidad al contexto específico de cada conjunto. Esta capacidad de combinar distintas señales en un único proceso de inferencia mejora sustancialmente la robustez de las imputaciones, incluso cuando los datos contienen ruido, valores atípicos o patrones excepcionales no previstos previamente.

No obstante, conviene señalar que el modelo no está exento de limitaciones. A pesar de las estrategias de optimización implementadas, tanto el proceso de entrenamiento como el de inferencia pueden resultar costosos desde el punto de vista computacional, especialmente en entornos con restricciones complejas o con bases de datos muy extensas. Además, la calidad de las reparaciones obtenidas depende en gran medida de la calidad de los datos disponibles y de una correcta formulación de las restricciones. Por último, aunque HoloClean ofrece mayor interpretabilidad que otros modelos basados

únicamente en aprendizaje automático de tipo caja negra, su entendimiento pleno requiere familiaridad con conceptos estadísticos avanzados, particularmente en lo relativo a grafos de factores y a la interpretación de sus parámetros aprendidos.

Capítulo 4

Detección de anomalías en series temporales univariantes

4.1. Introducción y motivación

El tratamiento de datos faltantes es uno de los primeros pasos a seguir en el aseguramiento de la calidad de un conjunto de datos pues, se ha demostrado que, estos pueden afectar en gran medida a los resultados de los análisis posteriores que se realicen sobre ellos.

En particular, Niako et al. (2024) realizaron un estudio experimental en el que se evaluó, para distintos niveles de valores faltantes (del 10 % al 35 %) en una serie temporal univariante, el impacto de distintas técnicas de imputación sobre los ajustes de los modelos ARIMA y LSTM. En el estudio se compararon desde técnicas de imputación sencillas como la sustitución por la media, hasta técnicas más avanzadas como pueden ser el suavizado de Kalman o la interpolación spline. Los resultados de este estudio mostraron que la imputación de los datos nulos afectaba directamente a la precisión de las predicciones. Los métodos de imputación más simples, como la sustitución por la media, llevaban a incrementos en el error de predicción, especialmente a mayor porcentaje de datos faltantes. Sin embargo, métodos más robustos como el suavizado de Kalman o la interpolación spline fueron capaces de preservar mejor la estructura de la serie y por tanto condujeron a predicciones más precisas de los modelos, incluso en los escenarios con mayor porcentaje de datos faltantes. Este ejemplo demuestra que un tratamiento cuidadoso de los datos nulos no solo es un paso previo necesario, sino que resulta crítico para garantizar la validez y fiabilidad de los modelos predictivos aplicados posteriormente.

El tratamiento de datos nulos nos lleva por tanto al siguiente eslabón natural en la cadena de gobernanza de la calidad del dato que consiste en la detección de valores anómalos. Las anomalías representan desviaciones significativas respecto de la dinámica histórica de los datos, su detección se utiliza ampliamente en una gran variedad de aplicaciones; como la detección de fraudes en tarjetas de crédito, seguros o atención médica, la detección de intrusiones en ciberseguridad, la detección de fallas en sistemas críticos para la seguridad y la vigilancia militar de actividades enemigas (Chandola et al. 2009).

El estudio que se realizará en el presente documento se centrará en la detección de anomalías en series temporales univariantes, un tema tratado ampliamente en la literatura científica (Blazquez-Garcia et al. 2021; Box et al. 2015). La decisión de centrar este capítulo en datos temporales, se debe a la necesidad de la entidad financiera ABANCA de incluir a los múltiples controles de calidad que la empresa aplica diariamente a sus datos, nuevos controles para asegurar que, el número de registros de una aplicación crítica para el banco, tenga entradas consistentes cada día.

La estructura de este capítulo será la siguiente. En primer lugar se presentarán los conceptos básicos necesarios para entender el problema como pueden ser la definición de una serie temporal junto con sus características o los tipos de anomalías que se pueden encontrar en ellas. Una vez explicado el marco teórico se pasará a la descripción de tres métodos para la modelización de series temporales y su posterior análisis de valores anómalos. Se revisarán las principales familias de métodos: estadísticos, de aprendizaje automático tradicional y basados en deep learning. Dentro de los métodos estadísticos tradicionales se estudiará el modelo ARIMA; los modelos basados en árboles de decisión serán el método de aprendizaje automático tradicional elegido, mientras que para representar los algoritmos de deep learning se tomará el modelo Encoder-Decoder.

Se desarrollarán las bases teóricas para cada uno de los citados métodos y se expondrá el modo de utilizarlos a la hora de detectar anomalías en una serie temporal.

4.2. Conceptos básicos

Antes de abordar los métodos específicos para la detección de anomalías en series temporales, es esencial definir formalmente los conceptos clave involucrados en esta tarea.

4.2.1. Definición y características de una serie temporal

Una serie temporal es una secuencia de observaciones ordenadas en el tiempo, registradas habitualmente a intervalos de tiempo regulares (diarios, mensuales, anuales, etc.). Desde un punto de vista estadístico, puede definirse como una realización de un proceso estocástico $(X_t)_{t\in T}$, donde T es un conjunto de índices temporales y cada X_t es una variable aleatoria definida sobre un espacio de probabilidad (Box et al. 2015). Una serie temporal se puede expresar de la siguiente forma:

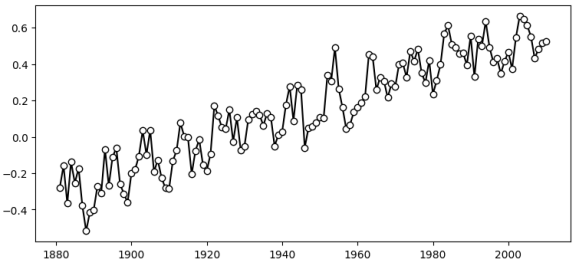
$$\{x_t : t = 1, 2, \ldots, n\}, \quad x_t \in \mathbb{R}, \quad \text{donde } x_t \sim X_t.$$

Es decir, la serie $\{x_t\}_{t=1}^{n}$ es una realización específica (una muestra) del proceso estocástico generador. En la Figura 4.1 se pueden ver algunos ejemplos de series temporales simuladas.

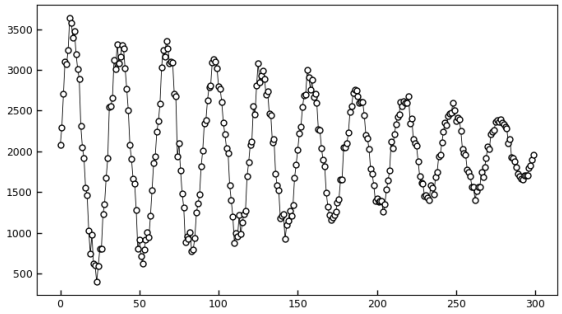
Es importante destacar que si cada observación x_t es un escalar, $x_t \in \mathbb{R}$, entonces se tiene una serie temporal univariante que representa la evolución de una única variable a lo largo del tiempo. Sin embargo si cada observación x_t es un vector, $x_t = (x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(p)})^\top$, se cuenta con una serie temporal multivariante que permite modelar la evolución conjunta de varias variables correlacionadas a lo largo del tiempo.

El análisis de series temporales busca modelar la dependencia temporal, entender la estructura subyacente del fenómeno y estimar valores futuros.

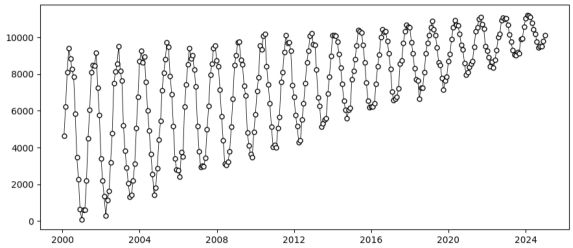
Una vez se ha definido formalmente el concepto de serie temporal, se describen a continuación algunas de las principales características a las que se debe prestar atención a la hora de analizar una serie temporal: tendencia, estacionalidad, heterocedastidad y estacionariedad.



(a) Serie temporal con tendencia.



(b) Serie temporal periódica y heterocedástica.



(c) Serie temporal con tendencia, periódica y heterocedástica.

Figura 4.1: Ejemplos de series temporales.

Tendencia

La tendencia en una serie temporal describe la evolución sistemática y prolongada de su valor medio a lo largo del tiempo. Formalmente, puede entenderse como la dependencia del valor medio de la serie respecto del tiempo:

$$\mu_t = \mathbb{E}[X_t],$$

donde μ_t no es constante, sino que varía suavemente con t (Box et al 2015). En procesos con tendencia, la media de la serie temporal muestra un patrón creciente, decreciente o más complejo, que puede capturarse mediante funciones polinómicas, exponenciales o modelos no paramétricos.

En la Figura 4.1 se observa que tanto la primera serie 4.1a como la tercera 4.1c tienen una tendencia creciente. Así mismo, la segunda 4.1b muestra un ejemplo de serie temporal con nivel medio constante.

Estacionalidad

Se dice que una serie tiene componente estacional si contiene patrones que se repiten de manera regular a intervalos constantes en el tiempo. Normalmente estos patrones se pueden vincular a factores calendario o efectos cíclicos previsibles. Formalmente, la serie presenta estacionalidad con periodo s si existe un patrón sistemático tal que:

$$X_t = X_{t+s} + \varepsilon_t,$$

con ε_t representando ruido aleatorio (Box et al 2015). La correcta identificación y modelización de la estacionalidad es esencial, ya que ignorarla puede inducir autocorrelación residual significativa y predicciones sesgadas. Las Figuras 4.1b y 4.1c ilustran un ejemplo de series con un patrón estacional.

Heterocedasticidad

Una serie temporal es heterocedástica si su varianza no se mantiene constante a lo largo del tiempo, es decir, la magnitud de la dispersión o volatilidad cambia sistemáticamente. Matemáticamente, se puede representar como:

$$Var(X_t) = \sigma_t^2,$$

donde σ_t^2 no es constante (Box et al 2015).

En las Figuras 4.1b y 4.1c se presenta un ejemplo de serie con heterocedasticidad. En ellas se aprecia cómo la variabilidad de la serie disminuye a medida que avanza el tiempo.

Estacionariedad

La estacionariedad es una propiedad fundamental en el análisis de series temporales, ya que muchos modelos estadísticos clásicos, como el ARIMA, requieren que la serie sea estacionaria para su correcta aplicación. Se tiene que una serie temporal es estacionaria si sus propiedades estadísticas; media, varianza y autocorrelación, son invariantes con respecto al tiempo.

$$\mathbb{E}[X_t] = \mu \quad \text{constante},$$
$$Var(X_t) = \sigma^2 \quad \text{constante},$$
$$Cov(X_t, X_{t+h}) = \gamma(h) \quad \text{dependiente solo de } h.$$

Es decir, la media y la varianza deben ser constantes en el tiempo y la autocovarianza solo depender del desfase h (Box et al 2015). Ninguna de la series de la Figura 4.1 es estacionaria.

Hacer notar que la correcta identificación de estas características resulta esencial para especificar modelos adecuados que capturen la dinámica subyacente de la serie temporal y permitan una detección eficaz de las anomalías.

4.2.2. Tipos de anomalías

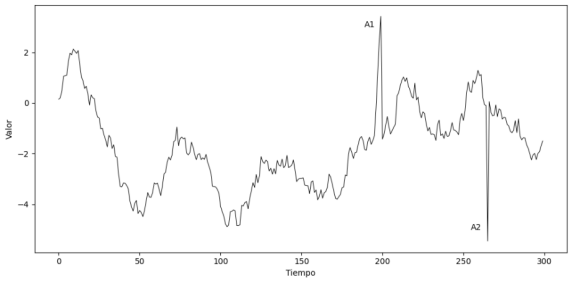
El análisis de series temporales no solo busca modelar su estructura para realizar predicciones fiables, sino que también trata de identificar observaciones que se desvían significativamente del comportamiento esperado. Dichas observaciones se denominan anomalías o *outliers*, que se definen como aquellos datos que no concuerdan con el patrón general del resto de las observaciones y cuya presencia puede deberse a errores de medición, eventos excepcionales o cambios en el régimen subyacente del proceso generador de datos (Chandola et al 2009).

Formalmente, la detección de anomalías se define como la tarea de identificar patrones en los datos que no se ajustan a un comportamiento esperado o normal Chandola et al. (2009). En el contexto de series temporales, esta definición adquiere una dimensión adicional: la noción de normalidad no solo depende del valor puntual observado, sino también de su localización temporal y de su relación con valores cercanos. Debido a ello, las técnicas de detección de anomalías en series temporales deben tener en cuenta la estructura temporal, la autocorrelación de la serie y los patrones recurrentes como la tendencia o la estacionalidad.

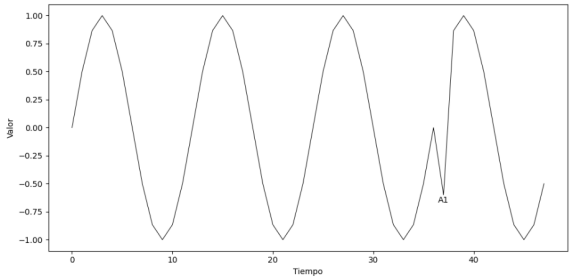
Detectar anomalías en series temporales es un problema con numerosas aplicaciones prácticas, como el control de calidad industrial, la monitorización de sistemas financieros, la detección de fraudes, la ciberseguridad o la vigilancia de infraestructuras críticas (Chandola et al. 2009; Blázquez-García et

al. 2021).

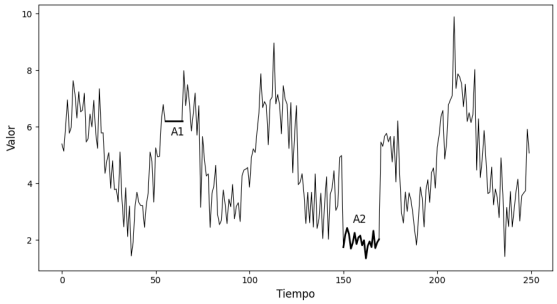
En series temporales univariantes, se distinguen tres tipos principales de anomalías, que se describen a continuación con mayor detalle.



(a) Serie temporal con dos anomalías puntuales.



(b) Serie temporal con una anomalía contextual.



(c) Serie temporal con dos anomalías colectivas.

Figura 4.2: Simulaciones de los distintos tipos de anomalías en series temporales.

Anomalías puntuales

Las anomalías puntuales son observaciones individuales que se alejan significativamente del patrón general de la serie. Matemáticamente, se produce una anomalía puntual en $x^* \in \{x_t\}_{t=1}^n$ si:

$$|x^* - \mu| > k,$$

donde μ representa el nivel medio de la serie y k es un umbral marcado (Chandola et al. 2009).

Anomalías contextuales o dependientes del contexto

Las anomalías contextuales son observaciones que pueden considerarse normales en ciertos contextos, pero anómalas en otros. En series temporales, el contexto suele definirse por la localización temporal o la estacionalidad. Formalmente, $x^* \in \{x_t\}_{t=1}^n$ se considera contextualemente anómalo si:

$$x^* \notin \mathcal{N}_t,$$

donde $\mathcal{N}_t$ representa el rango de valores esperados condicionado al contexto temporal (por ejemplo, la estación del año, el día de la semana o la hora del día) (Chandola et al. 2009).

Un ejemplo de este tipo de anomalía podría ser un pico de demanda energética que es normal en invierno pero inusual en verano.

Anomalías colectivas

Las anomalías colectivas se producen cuando un conjunto secuencial de observaciones es anómalo en su conjunto, aunque cada punto individual pueda no serlo. Se definen como:

$$\{x_t, x_{t+1}, \ldots, x_{t+k}\} \not\sim \mathcal{P},$$

donde $\mathcal{P}$ describe el patrón temporal esperado para secuencias de longitud $k+1$ (Chandola et al. 2009).

En la Figura 4.2 se tienen ejemplos gráficos de cada uno de los tipos de anomalías que se han mencionado. Es importante tener en cuenta que la identificación del tipo de anomalía presente en la serie temporal es esencial para seleccionar métodos de detección apropiados.

4.2.3. Métodos para detección de anomalías

Una estrategia ampliamente utilizada para la detección de anomalías en series temporales univariantes consiste en aprovechar la capacidad predictiva de un modelo ajustado a los datos históricos. Si se dispone de un modelo capaz de capturar la dinámica subyacente del proceso generador de datos, las predicciones realizadas para puntos futuros deben aproximarse a los valores reales observados bajo condiciones normales.

Cuando se produce una anomalía, el valor observado se desvía significativamente de la predicción del modelo, generando un error de predicción, o residuo, inusualmente grande. Por tanto, la tarea de detección de anomalías puede formularse como la identificación de valores atípicos en la distribución de los residuos del modelo (Box et al 2015).

Dada una serie temporal $\{x_t\}_{t=1}^n$ y un modelo que produce predicciones $\{\hat{x}_t\}_{t=1}^n$, se define el residuo como:

$$\hat{e}_t = x_t - \hat{x}_t \ \forall t \in \{1, ..., n\}.$$

Bajo el supuesto de normalidad en los residuos, la mayoría de los valores $\hat{e}_t$ con $t \in \{1, ..., n\}$ deberían situarse dentro de un rango de control preestablecido. Los puntos donde $\hat{e}_t$ supere este umbral pueden señalar la presencia de una anomalía.

Gráficos de control

Una herramienta clásica para visualizar este enfoque consiste en los gráficos de control, que permiten monitorizar la evolución de un proceso a lo largo del tiempo y detectar desviaciones significativas respecto a su comportamiento esperado. En el contexto de series temporales, este enfoque puede aplicarse directamente sobre los residuos de un modelo predictivo. Su análisis permite identificar observaciones que podrían considerarse anómalas (Montgomery 2013).

Como se puede observar en la Figura 4.3, un gráfico de control está compuesto por tres elementos fundamentales: una línea central que representa el valor medio estimado de los residuos, y dos límites de control (superior e inferior) que delimitan el rango esperado de variabilidad bajo condiciones normales del proceso. Estos límites se calculan típicamente a partir de la desviación estándar de los residuos, siguiendo la fórmula:

$$\hat{\mu} \pm k \cdot \hat{\sigma},$$

donde $\hat{\mu}$ es la media de los residuos, $\hat{\sigma}$ su desviación estándar, y k un coeficiente que determina el nivel de confianza deseado (comúnmente $k = 3$ para cubrir aproximadamente el 99,73 % de los valores bajo normalidad). Un residuo que se sitúe fuera de estos límites se interpreta como un indicio de que el

proceso podría estar fuera de control o que se ha producido una anomalía puntual.

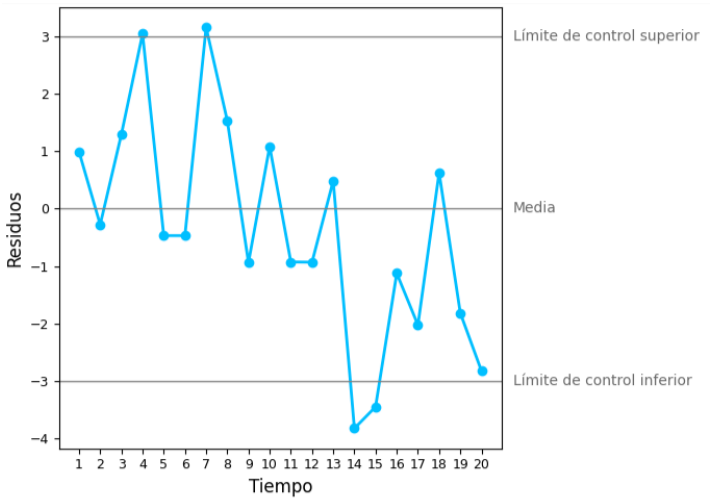


Figura 4.3: Ejemplo de gráfico de control.

Cuando se trabaja con modelos predictivos y conjuntos de datos separados en entrenamiento y test, la construcción del gráfico de control sigue una metodología estructurada. En primer lugar, se entrena el modelo utilizando únicamente el conjunto de entrenamiento, y se calculan los residuos correspondientes a dicho conjunto. A partir de estos residuos se estima la media $\hat{\mu}_{\text{train}}$ y la desviación estándar $\hat{\sigma}_{\text{train}}$, que se emplearán para definir los límites de control:

$$\text{Límite inferior} = \hat{\mu}_{\text{train}} - k \cdot \hat{\sigma}_{\text{train}},$$

$$\text{Límite superior} = \hat{\mu}_{\text{train}} + k \cdot \hat{\sigma}_{\text{train}}.$$

Estos límites reflejan el comportamiento normal aprendido durante el entrenamiento y permanecen fijos. A continuación, se evalúa el modelo sobre el conjunto de test, obteniendo los residuos de predicción para cada punto. Estos residuos se representan sobre el gráfico de control junto con los límites previamente definidos. Aquellas observaciones del test que caen fuera de los límites de control se consideran candidatas a anomalía.

En resumen, los gráficos de control constituyen una herramienta visual potente y flexible para la detección de anomalías, permitiendo identificar tanto errores puntuales como patrones de desviación sistemática en la calidad de las predicciones. Su construcción basada en la estimación de parámetros sobre el conjunto de entrenamiento asegura la validez estadística del procedimiento, y su aplicación sobre datos de test permite evaluar de forma objetiva el rendimiento del modelo ante datos no vistos.

Evaluación de la detección de anomalías

El rendimiento de un sistema de detección de anomalías se evalúa comparando sus predicciones con las etiquetas verdaderas (cuando estas están disponibles), mediante métricas clásicas del aprendizaje supervisado (Tatbul et al. 2018). Para ello, se utiliza la matriz de confusión, que permite clasificar las predicciones de acuerdo con los siguientes términos:

- **True Positives (TP):** Anomalías correctamente identificadas por el sistema.

- **False Positives (FP):** Observaciones normales clasificadas erróneamente como anomalías.
- **False Negatives (FN):** Anomalías reales no detectadas por el sistema.

A partir de estos valores, se definen las siguientes métricas de evaluación:

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F1\text{-}score = 2\ \frac{Precisión \times Recall}{Precisión + Recall}.$$

Precision indica la proporción de observaciones detectadas como anómalas que realmente lo eran, el *recall* refleja la capacidad del sistema para identificar todas las anomalías presentes y finalmente *F1-score* representa la media armónica entre *precision* y *recall*, proporcionando una medida única del rendimiento del sistema cuando se desea equilibrar ambas dimensiones.

Estas métricas, ampliamente utilizadas en clasificación binaria, resultan especialmente relevantes cuando se dispone de etiquetas fiables para las anomalías, permitiendo una evaluación cuantitativa y comparativa entre distintos modelos de detección.

4.2.4. Métricas para la validación de las predicciones

Además de detectar anomalías mediante el análisis de residuos, es fundamental evaluar la calidad global de las predicciones generadas por un modelo de series temporales. Para ello se emplean métricas de error que cuantifican la discrepancia entre los valores observados y los valores estimados. Estas métricas permiten comparar diferentes modelos y seleccionar aquel que ofrece el mejor ajuste a los datos históricos (Hyndman y Athanasopoulos 2018).

A continuación se describen tres métricas ampliamente utilizadas en la literatura: el error absoluto medio (MAE), la raíz del error cuadrático medio (RMSE) y el error porcentual absoluto medio (MAPE).

Error Absoluto Medio (MAE)

El Error Absoluto Medio (MAE) mide el valor promedio de los errores absolutos entre las predicciones y las observaciones reales. Se define como:

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^{n} |x_t - \hat{x}_t|,$$

donde x_t es el valor observado, $\hat{x}_t$ es el valor estimado, y n es el número de observaciones.

El MAE proporciona una medida interpretable en las mismas unidades que la variable de interés y penaliza de manera lineal todos los errores, lo que lo hace robusto ante la presencia de anomalías moderadas.

Raíz del Error Cuadrático Medio (RMSE)

La Raíz del Error Cuadrático Medio (RMSE) se define como la raíz cuadrada del promedio de los errores al cuadrado entre los valores ajustados y los valores observados:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_t - \hat{x}_t)^2}.$$

El RMSE proporciona una medida de la magnitud promedio del error, penalizando con mayor severidad los errores grandes debido al efecto cuadrático. Esta característica lo hace particularmente útil cuando se desea dar mayor importancia a las desviaciones significativas en el ajuste del modelo.

Error Porcentual Absoluto Medio (MAPE)

El Error Porcentual Absoluto Medio (MAPE) mide el error absoluto en porcentaje respecto al valor real, ofreciendo una interpretación intuitiva:

$$\text{MAPE} = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{x_t - \hat{x}_t}{x_t}\right|.$$

Es ampliamente utilizado por su interpretación directa, ya que indica cuánto se desvían las predicciones del modelo respecto a los valores reales, en términos porcentuales. Sin embargo, debe aplicarse con precaución cuando la serie contiene valores cercanos a cero, ya que en esos casos el MAPE puede volverse inestable o indefinido.

El uso conjunto de estas métricas permite evaluar de forma integral el rendimiento de un modelo predictivo.

4.3. Presentación de modelos para series temporales univariantes

Una vez tratados los conceptos previos necesarios, en las próximas secciones se explicará con detalle el funcionamiento de tres métodos ampliamente utilizados para la modelización y predicción de series temporales univariantes. En este trabajo se han seleccionado tres métodos representativos de distintas familias de modelos:

- **ARIMA**, como modelo estadístico clásico y paramétrico, basado en supuestos bien definidos sobre la estructura estocástica de la serie temporal. Es ampliamente utilizado por su interpretabilidad y por su eficacia en capturar dependencias lineales y tendencias tras procesos de diferenciación (Box et al. 2015).

- **Modelos basados en árboles de decisión**, como representante de los métodos de aprendizaje automático tradicionales. Permite modelar relaciones no lineales y complejas entre variables derivadas de retardos, a partir de un conjunto de árboles de decisión agregados para mejorar la estabilidad y precisión de las predicciones (Bontempi et al. 2012).

- **Encoder-Decoder**, como ejemplo de modelos basados en redes neuronales profundas, diseñados para capturar dependencias de largo plazo y patrones no lineales en secuencias temporales mediante arquitecturas de codificación y decodificación, ampliamente utilizadas en aplicaciones de secuencia a secuencia (Hewamalage et al. 2021).

Cada uno de estos métodos aporta perspectivas distintas sobre la estructura de la serie temporal y presenta ventajas y limitaciones específicas.

4.4. Modelo ARIMA

El primer método para modelar series temporales univariantes que se tratará será el modelo ARIMA, uno de los enfoques clásicos más utilizados. Se caracteriza por su capacidad para modelar dependencias lineales en datos estacionarios o transformados en estacionarios mediante diferenciación (Box et al. 2015).

4.4.1. Formulación del modelo

Sea $\{x_t\}_{t=1}^n$ una serie temporal observada. El modelo ARIMA(p, d, q) combina tres componentes:

- **Componente autorregresivo (AR) de orden** p: la variable depende linealmente de sus p valores pasados.
- **Componente de diferenciación (I) de grado** d: diferencia la serie d veces para hacerla estacionaria.
- **Componente de media móvil (MA) de orden** q: modela la dependencia de los errores pasados.

El modelo ARIMA(p, d, q) se expresa como:

$$\Phi(B)(1 - B)^d x_t = c + \Theta(B)\varepsilon_t,$$

donde:

- B es el operador de retardo: $Bx_t = x_{t-1}$.
- $c \in \mathbb{R}$ es una constante
- $(1 - B)^d$ representa la diferencia de orden d para transformar la serie en estacionaria. Aplicar $(1 - B)$ una vez corresponde a la diferencia primera: $(1 - B)x_t = x_t - x_{t-1}$ y de forma general, aplicar la diferencia de orden d consiste en repetir la operación d veces.
- $\Phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$ es el polinomio autorregresivo.
- $\Theta(B) = 1 + \theta_1 B + \ldots + \theta_q B^q$ es el polinomio de media móvil.
- ε_t se denomina ruido blanco. Se caracterizan por ser variables incorreladas con media cero y varianza constante.

El modelo busca explicar la dinámica de x_t a partir de sus valores pasados y de la estructura del error. La diferencia de orden d garantiza la estacionariedad en media, eliminando tendencias deterministas o estacionales de baja frecuencia.

Cuando la serie presenta patrones estacionales regulares, la extensión estacional del modelo, ARIMA$(p, d, q)(P, D, Q)_s$, permite capturar tanto componentes no estacionales como estacionales.

La forma general del modelo se expresa como:

$$\Phi(B^s)\phi(B)(1 - B)^d(1 - B^s)^D x_t = c + \Theta(B^s)\theta(B)\varepsilon_t,$$

donde:

- B es el operador de retardo: $Bx_t = x_{t-1}$.
- $c \in \mathbb{R}$ es una constante

- $(1-B)^d$ realiza diferencias no estacionales.
- $(1-B^s)^D$ aplica diferencias estacionales.
- $\phi(B)$ y $\theta(B)$ son los polinomios no estacionales:

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$$

$$\theta(B) = 1 + \theta_1 B + \ldots + \theta_q B^q$$

- $\Phi(B^s)$ y $\Theta(B^s)$ son los polinomios estacionales:

$$\Phi(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_P B^{Ps}$$

$$\Theta(B^s) = 1 + \Theta_1 B^s + \ldots + \Theta_Q B^{Qs}$$

- ε_t es ruido blanco.

4.4.2. Hipótesis del modelo

El modelo ARIMA impone varias hipótesis fundamentales:

- **Estacionariedad (tras diferenciación):** Se asume que la serie transformada mediante diferenciación es estacionaria, es decir, su media y varianza son constantes en el tiempo y las autocorrelaciones dependen solo del retardo.
- **Linealidad:** El modelo supone una relación lineal entre los valores actuales y pasados de la serie.
- **Errores ruido blanco:** El término de error ε_t es un proceso de ruido blanco, es decir; incorrelado, con media cero y varianza constante.

4.4.3. Ajuste del modelo

El ajuste de un modelo ARIMA consta de tres etapas principales: identificación, estimación y validación del modelo.

Identificación de órdenes

El primer paso es determinar los valores adecuados de los órdenes p, d y q (y P, D, Q, s en la versión estacional). Para ello, se realiza un análisis exploratorio de la serie temporal y sus transformaciones, por ejemplo, diferenciación para eliminar tendencia y estacionalidad.

Tras la diferenciación apropiada (de orden d), la serie transformada debe aproximarse a la estacionariedad. En ese estado, se emplean las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para sugerir los órdenes autorregresivo y de media móvil.

- La **función de autocorrelación (ACF)** mide el grado de dependencia lineal entre observaciones separadas por distintos instantes temporales. En su forma general, sin asumir estacionariedad, se define como:

$$\rho(s,t) = \frac{\mathbb{E}[(X_s - \mu_s)(X_t - \mu_t)]}{\sigma_s \sigma_t},$$

  donde $\mu_s = \mathbb{E}[X_s]$ y $\mu_t = \mathbb{E}[X_t]$ son las medias en los tiempos s y t, respectivamente, y σ_s, σ_t son sus desviaciones típicas.

En contextos prácticos, la ACF se representa en función de un retardo h y se estima mediante la autocorrelación muestral:

$$\hat{\rho}(h) = \frac{\sum_{t=1}^{n-h}(x_t - \bar{x})(x_{t+h} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}.$$

Un comportamiento característico de los procesos de medias móviles MA(q) es que su función de autocorrelación (ACF) se anula exactamente a partir del retardo $q+1$, es decir, $\rho(h) = 0$ para todo $h > q$. Por tanto, observar un corte abrupto en la ACF del proceso muestral, donde los coeficientes de autocorrelación caen a cero o no resultan estadísticamente significativos más allá de cierto retardo q, constituye una señal empírica de que el proceso subyacente podría ajustarse adecuadamente con un modelo MA(q).

- La **función de autocorrelación parcial (PACF)** mide la correlación entre X_t y X_{t-h} eliminando el efecto lineal de las variables intermedias $(X_{t-1}, \ldots, X_{t-h+1})$. En la práctica, se estima como el coeficiente asociado a x_{t-h} en la regresión lineal múltiple:

$$x_t = \alpha_1 x_{t-1} + \cdots + \alpha_h x_{t-h} + \varepsilon_t,$$

siendo:

$$\text{PACF}(h) = \hat{\alpha}_h.$$

En los procesos autorregresivos AR(p), la función de autocorrelación parcial (PACF) presenta un comportamiento distintivo: los coeficientes son diferentes de cero únicamente hasta el retardo p, anulándose o volviéndose no significativos a partir de $h > p$. Por tanto, cuando los coeficientes de autocorrelación parcial disminuyen rápidamente y se estabilizan en torno a cero a partir de cierto orden p, se interpreta como una evidencia de que un modelo AR(p) puede ser adecuado para representar la dinámica de la serie.

Es importante destacar que ambas funciones de autocorrelación toman valores en el intervalo $[-1, 1]$.

Estas funciones también desempeñan un papel fundamental en la identificación de los componentes estacionales del modelo ARIMA. En particular, cuando la serie presenta patrones que se repiten cada s periodos (por ejemplo, $s = 12$ para datos mensuales con estacionalidad anual), la función de autocorrelación (ACF) y la de autocorrelación parcial (PACF) pueden revelar dicha estructura.

En el caso estacional, se examinan las ACF y PACF en múltiplos del período s (es decir, en los retardos s, $2s$, $3s$, etc.) para detectar indicios de comportamiento estacional. Por ejemplo, un pico significativo en la ACF en el retardo s puede sugerir la presencia de una componente de media móvil estacional de orden $Q = 1$, mientras que un corte en la PACF en el mismo retardo puede indicar una componente autorregresiva estacional con $P = 1$. Así mismo, si se observa una tendencia o una periodicidad persistente en las funciones de autocorrelación, puede ser necesario aplicar diferenciación estacional, es decir, considerar un orden $D \geq 1$.

Este análisis gráfico se complementa con criterios de selección de modelos basados en medidas de ajuste como el criterio de información de Akaike (AIC) o el criterio bayesiano de Schwarz (BIC), que permiten comparar formalmente diferentes especificaciones de modelos ARIMA$(p, d, q)(P, D, Q)_s$ en función de su capacidad explicativa y penalización por complejidad (Box et al. 2015).

En la Figura 4.4 se tiene un ejemplo de la función de autocorrelación (ACF) y la función de autocorrelación parcial (PACF) asociadas a un modelo MA(2).

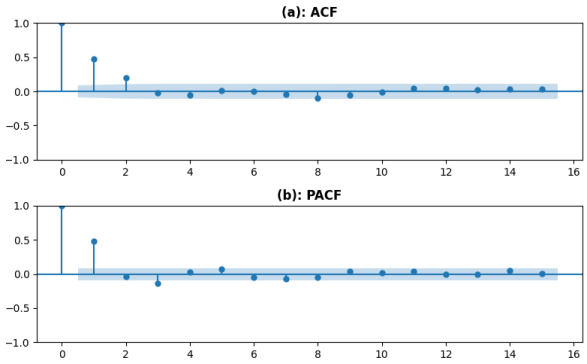


Figura 4.4: Función de autocorrelación y autocorrelación parcial de un proceso MA(2).

Estimación de parámetros

Una vez seleccionados los órdenes, se procede a estimar los parámetros del modelo, es decir, la constante c y los coeficientes de los polinomios autorregresivos y de medias móviles, así como la varianza del error σ^2. Las dos estrategias más utilizadas son:

- **Mínimos cuadrados condicionales:** consiste en minimizar la suma de los cuadrados de los residuos generados al aplicar el modelo a la serie. Sea $\hat{e}_t$ el residuo en el tiempo t, la función objetivo se expresa como:

$$\min_{\theta} \sum_{t=1}^{n} \hat{e}_t^2,$$

  donde θ representa el vector de parámetros del modelo.

- **Máxima verosimilitud:** busca maximizar la probabilidad de observar la serie dada la parametrización del modelo. La función de verosimilitud se define como:

$$L(\theta) = P(x_1, x_2, \ldots, x_T \mid \theta)$$

  El objetivo es encontrar $\hat{\theta}$ que maximice $L(\theta)$. (Box et al. 2015).

Validación del modelo

Una parte esencial del ajuste de modelos ARIMA consiste en analizar los residuos:

$$\hat{e}_t = x_t - \hat{x}_t.$$

Para que el modelo sea adecuado, los residuos deben comportarse como ruido blanco, es decir, deben cumplir las siguientes propiedades: tener media estadísticamente indistinguible de cero, autocorrelaciones no significativas en todos los retardos relevantes y varianza constante. Así mismo, aunque no es estrictamente necesario, la normalidad de los residuos es deseable para la validez de ciertos intervalos de predicción y tests.

El diagnóstico de residuos se lleva a cabo mediante diversas herramientas, tanto gráficas como estadísticas. Entre las primeras destacan la representación temporal de los residuos y los gráficos de autocorrelación (ACF) y autocorrelación parcial (PACF) aplicados sobre ellos. En cuanto a los métodos estadísticos, se emplean contrastes como el test de Ljung-Box para evaluar la independencia temporal, y pruebas de normalidad como el test de Shapiro-Wilk, complementadas habitualmente con gráficos

Q-Q (Box et al. 2015; Hyndman y Athanasopoulos 2018).

Este diagnóstico es fundamental para asegurar la validez del modelo y su capacidad de predicción.

4.5. Modelos basados en árboles de decisión: bagging y boosting

En esta sección se describen los métodos basados en árboles de decisión en el contexto general de problemas de regresión supervisada, con especial atención a dos estrategias de ensamblado ampliamente utilizadas: *bagging* y *boosting*. Estos métodos operan sobre conjuntos de datos formados por pares $(\mathbf{Z}, X)$, donde $\mathbf{Z} = [Z_1, Z_2, \ldots, Z_p]$ es un vector de p variables explicativas y X representa la variable respuesta. El objetivo es construir un modelo capaz de predecir X para nuevas observaciones $\mathbf{z}_*$, a partir de un conjunto de entrenamiento. La adaptación de estas técnicas al caso de series temporales se tratará en una sección posterior.

4.5.1. Árboles de decisión

Un árbol de decisión es un modelo predictivo que particiona recursivamente el espacio de las variables explicativas $\mathbf{Z}$ buscando minimizar la heterogeneidad de la variable respuesta X dentro de cada región terminal o hoja. En el contexto de regresión, el árbol asigna a cada observación un valor estimado correspondiente a la media de X en la hoja a la que pertenece.

Formalmente, un árbol define un conjunto de regiones disjuntas $\{R_m\}_{m=1}^{M}$ en el espacio de las variables predictoras $\mathbf{Z}$, y la predicción para una nueva observación $\mathbf{z}_*$ se expresa como:

$$\hat{x} = \sum_{m=1}^{M} c_m \cdot \mathbb{I}(\mathbf{z}_* \in R_m),$$

donde $\mathbb{I}(\cdot)$ es la función indicadora y c_m representa el valor estimado en la región R_m, calculado como:

$$c_m = \frac{1}{|R_m|} \sum_{i \in R_m} x_i,$$

es decir, la media de los valores de la variable respuesta correspondientes a las observaciones de entrenamiento cuyas variables predictoras $\mathbf{z}_i$ pertenecen a R_m.

El algoritmo de construcción del árbol sigue un enfoque de partición recursiva. En cada nodo, se evalúan todas las posibles divisiones del tipo $Z_j < s$, para cada variable Z_j y cada posible umbral s, se selecciona la partición (R_1, R_2) que minimiza la suma de los errores cuadráticos dentro de las regiones generadas:

$$\min_{j,\, s} \left[\sum_{i \in R_1(j,s)} (x_i - \bar{x}_{R_1})^2 + \sum_{i \in R_2(j,s)} (x_i - \bar{x}_{R_2})^2 \right],$$

donde $R_1(j,s) = \{\mathbf{Z} : Z_j < s\}$ y $R_2(j,s) = \{\mathbf{Z} : Z_j \geq s\}$ son las regiones definidas por la división en la variable Z_j y $\bar{x}_{R_k}$ representa la media de los x_i pertenecientes a la región R_k.

El proceso se detiene cuando se alcanza una profundidad máxima predefinida, cuando el número de observaciones en una hoja es inferior a un umbral mínimo, o cuando la mejora en la función objetivo no justifica una nueva partición.

Aunque los árboles de decisión son modelos interpretables y flexibles, capaces de capturar relaciones no lineales e interacciones entre variables sin necesidad de especificarlas explícitamente, presentan una alta varianza: pequeñas perturbaciones en los datos de entrenamiento pueden conducir a árboles estructuralmente distintos, lo que incrementa el riesgo de sobreajuste (Hastie et al. 2009).

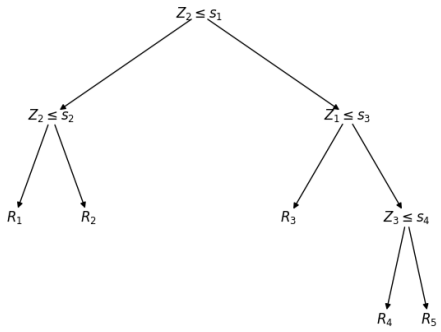


Figura 4.5: Ejemplo de la estructura de un árbol de decisión.

4.5.2. Bagging: Random Forest

El *bagging* es una técnica de ensamblado diseñada para reducir la varianza de predictores inestables, como los árboles de decisión individuales. El procedimiento consiste en generar múltiples subconjuntos de entrenamiento D_b mediante muestreo con reemplazo, *bootstrap*, a partir del conjunto original, entrenar un árbol de decisión independiente sobre cada uno de ellos, y promediar sus predicciones para obtener la salida final (Breiman 2001).

Sea $\hat{f}_b(\mathbf{z})$ la predicción del árbol número b, entrenado sobre la muestra D_b. La predicción agregada del modelo *Random Forest* se obtiene como el promedio de las predicciones individuales:

$$\hat{f}(\mathbf{z}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\mathbf{z}),$$

donde B es el número total de árboles del ensamblado.

Para introducir diversidad adicional entre los árboles y reducir la correlación entre ellos, en cada división de cada árbol se selecciona aleatoriamente un subconjunto de las variables explicativas $\mathbf{Z}$. Este procedimiento mejora la estabilidad y capacidad de generalización del modelo.

Una de las principales ventajas de los métodos de bagging como *Random Forest* es su capacidad para reducir la varianza del modelo respecto a un único árbol de decisión, manteniendo un sesgo bajo. Además, estos métodos son robustos frente al ruido, manejan adecuadamente relaciones no lineales complejas y son relativamente insensibles a la presencia de variables irrelevantes.

Sin embargo, este tipo de modelos presenta ciertas limitaciones. En particular, al combinar un gran número de árboles, se pierde interpretabilidad, lo que dificulta el análisis detallado del mecanismo de decisión subyacente.

Esquema del algoritmo Random Forest:

1. A partir del conjunto de entrenamiento original $\mathbf{z} = [z_1, z_2, \ldots, z_p]$, generar B muestras bootstrap independientes: $D_1, D_2, \ldots, D_B$.

2. Para cada muestra D_b, entrenar un árbol de decisión T_b:
   - En cada división del árbol, seleccionar aleatoriamente un subconjunto de $k < p$ variables explicativas.
   - Elegir la variable y el umbral que minimicen el error cuadrático dentro de las regiones resultantes.
3. Para realizar una predicción sobre una nueva observación $\mathbf{z}_*$, obtener la predicción de cada árbol $\hat{f}_b(\mathbf{z}_*)$.
4. Calcular la predicción final como el promedio de las predicciones individuales:

$$\hat{f}(\mathbf{z}_*) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(\mathbf{z}_*).$$

4.5.3. Boosting

El *boosting* es una técnica de ensamblado secuencial que combina múltiples modelos débiles (típicamente árboles de decisión poco profundos) para construir un modelo fuerte. A diferencia del *bagging*, en el que los modelos se entrenan de forma independiente, el *boosting* sigue un enfoque aditivo y secuencial: en cada iteración se entrena un nuevo árbol con el objetivo de corregir los errores del modelo acumulado hasta ese momento, prestando mayor atención a las observaciones mal estimadas en etapas anteriores (Friedman 2001).

El modelo final se construye como una suma ponderada de M modelos base:

$$\hat{f}(\mathbf{z}) = \sum_{m=1}^{M} \eta \cdot h_m(\mathbf{z}),$$

donde h_m es el modelo débil aprendido en la iteración m, η es la tasa de aprendizaje, *learning rate*, y $\mathbf{z}$ representa el vector de variables explicativas. La tasa de aprendizaje controla la contribución de cada nuevo modelo al ensamblado y actúa como mecanismo de regularización para prevenir el sobreajuste.

En el caso particular de regresión con error cuadrático, cada nuevo árbol h_m se ajusta para predecir los residuos del modelo actual, es decir:

$$r_i^{(m)} = x_i - \hat{f}^{(m-1)}(\mathbf{z}_i),$$

donde x_i es el valor observado de la variable respuesta y $\hat{f}^{(m-1)}(\mathbf{z}_i)$ es la predicción obtenida tras las primeras $m-1$ iteraciones.

De forma más general, en el algoritmo conocido como *gradient boosting*, se puede utilizar cualquier función de pérdida diferenciable. En este caso, cada nuevo modelo h_m se ajusta para aproximar el gradiente negativo de la función de pérdida respecto a las predicciones actuales, dirigiendo así el proceso de aprendizaje en la dirección de máximo descenso.

Este enfoque permite reducir simultáneamente el sesgo y la varianza del modelo, resultando en predictores muy potentes para capturar relaciones no lineales complejas. No obstante, el *boosting* es más sensible al sobreajuste que el *bagging*, especialmente si los árboles base son demasiado profundos o si la tasa de aprendizaje es elevada. Para mitigar estos riesgos, es habitual regular el modelo mediante una combinación de tasa de aprendizaje baja, árboles de pequeña profundidad y técnicas adicionales de regularización.

Esquema del algoritmo Gradient Boosting:

1. Inicializar el modelo con una predicción constante, por ejemplo la media de la variable respuesta:

$$\hat{f}^{(0)}(\mathbf{z}) = \bar{x}.$$

2. Para cada iteración $m = 1, 2, \ldots, M$:

   a) Calcular los residuos como el gradiente negativo de la función de pérdida con respecto a las predicciones actuales. En el caso de la pérdida cuadrática media:

$$\mathcal{L}(x_i, \hat{f}^{(m-1)}(\mathbf{z}_i)) = \frac{1}{2}(x_i - \hat{f}^{(m-1)}(\mathbf{z}_i))^2,$$

   el gradiente respecto a la predicción es:

$$\frac{\partial \mathcal{L}}{\partial \hat{f}^{(m-1)}(\mathbf{z}_i)} = \hat{f}^{(m-1)}(\mathbf{z}_i) - x_i,$$

   por lo que el gradiente negativo (dirección de descenso más pronunciado) es:

$$r_i^{(m)} = x_i - \hat{f}^{(m-1)}(\mathbf{z}_i).$$

   b) Ajustar un nuevo árbol de decisión $h_m(\mathbf{z})$ para predecir los residuos $r_i^{(m)}$ a partir de $\mathbf{z}_i$.

   c) Actualizar el modelo acumulado con una tasa de aprendizaje η:

$$\hat{f}^{(m)}(\mathbf{z}) = \hat{f}^{(m-1)}(\mathbf{z}) + \eta \cdot h_m(\mathbf{z}).$$

3. La predicción final tras M iteraciones es:

$$\hat{f}(\mathbf{z}) = \hat{f}^{(M)}(\mathbf{z}).$$

4.5.4. Aplicación a series temporales univariantes

Para emplear modelos basados en árboles de decisión en la predicción de series temporales univariantes, es necesario reformular el problema como una tarea de regresión supervisada. A diferencia de los modelos clásicos específicos para series temporales, como ARIMA, estos métodos no incorporan de forma inherente la estructura temporal de los datos, por lo que es fundamental construir variables explicativas que representen el historial de la serie.

Este enfoque consiste en generar un conjunto de variables predictoras a partir de los valores retardados (lags) de la propia serie. El número de retardos p puede determinarse mediante validación cruzada con particiones temporales o mediante análisis exploratorio con funciones de autocorrelación. Para un instante t, el vector de predictores $\mathbf{z}_t$ se define como:

$$\mathbf{z}_t = (x_{t-1}, x_{t-2}, \ldots, x_{t-p}),$$

donde x_t representa el valor de la serie temporal univariante. El objetivo es aprender una función f tal que:

$$\hat{x}_t = f(\mathbf{z}_t),$$

donde f es una función no paramétrica estimada mediante un modelo de ensamblado de árboles, capaz de capturar relaciones no lineales entre el pasado y el presente de la serie. Este planteamiento ha sido ampliamente documentado en la literatura como una estrategia eficaz para adaptar métodos de

aprendizaje supervisado al problema de predicción en series temporales (Bontempi et al. 2012).

En la práctica, se construye una matriz de datos desplazados a partir de la serie original, asegurando que las observaciones de entrenamiento utilicen únicamente información disponible hasta el instante t, con el fin de evitar fugas de información.

Para ilustrar este procedimiento, la Tabla 4.1 muestra un ejemplo de cómo se construye la matriz de predictores a partir de una serie temporal utilizando $p = 3$ retardos. Cada fila representa una observación en el instante t, formada por los tres valores pasados de la serie (x_{t-1}, x_{t-2}, x_{t-3}) como variables explicativas, y el valor actual x_t como variable respuesta.

Sea la serie temporal $x = \{x_1 = 100,\ x_2 = 105,\ x_3 = 102,\ x_4 = 107,\ x_5 = 110,\ x_6 = 108,\ x_7 = 111\}$, la tabla siguiente representa un subconjunto de la matriz de entrenamiento resultante:

$z_1 = x_{t-3}$	$z_2 = x_{t-2}$	$z_3 = x_{t-1}$	x_t
100	105	102	107
105	102	107	110
102	107	110	108
107	110	108	111

Tabla 4.1: Ejemplo de matriz de datos desplazados con $p = 3$ retardos.

El método *Random Forest* aborda esta tarea generando múltiples árboles entrenados sobre muestras bootstrap de la matriz de retardos, cuyas predicciones se agregan mediante promedio para reducir la varianza del modelo. Por su parte, el enfoque de *boosting*, como el implementado en *LightGBM*, ajusta secuencialmente árboles que corrigen los errores residuales de etapas anteriores, permitiendo modelar relaciones más complejas y adaptativas.

4.5.5. Ajuste del modelo y validación

El ajuste de modelos basados en árboles de decisión para series temporales univariantes se plantea como un problema de regresión supervisada y sigue una secuencia de etapas bien definidas. En primer lugar, se generan las variables explicativas a partir de retardos (lags) de la serie temporal. Así, cada observación en el instante t se representa como un vector $\mathbf{z}_t = [x_{t-1}, x_{t-2}, \ldots, x_{t-p}]$, que actúa como entrada del modelo para predecir el valor de x_t.

A continuación, la serie se divide en un conjunto de entrenamiento y otro de prueba, respetando estrictamente el orden temporal para evitar fugas de información del futuro al pasado. El modelo se entrena exclusivamente sobre el conjunto de entrenamiento, optimizando sus parámetros internos (como la estructura de los árboles) para minimizar una función de pérdida, típicamente el error cuadrático medio.

La selección de hiperparámetros del modelo, tales como el número de árboles, la profundidad máxima o la tasa de aprendizaje, puede realizarse mediante validación cruzada temporal o utilizando un conjunto de validación separado. En ambos casos, es fundamental preservar la naturaleza secuencial

de la serie durante el proceso de validación.

Una vez entrenado, el modelo se emplea para generar predicciones sobre el conjunto de prueba. Los residuos se definen como la diferencia entre los valores reales y las predicciones generadas por el modelo:

$$\hat{e}_t = x_t - \hat{x}_t,$$

y se analizan para evaluar la calidad del ajuste. Adicionalmente, se utilizan métricas cuantitativas como el error absoluto medio (MAE), el error cuadrático medio (RMSE) o el error porcentual absoluto medio (MAPE) para evaluar el rendimiento predictivo del modelo de forma objetiva.

4.6. Modelo Encoder-Decoder

La arquitectura Encoder-Decoder es un enfoque basado en redes neuronales recurrentes, inicialmente desarrollado para tareas de traducción automática bajo el paradigma *sequence-to-sequence* (Cho et al. 2014; Sutskever et al. 2014). Su versatilidad ha permitido su aplicación a otras tareas donde se requiere modelar secuencias de entrada y salida, como el resumen automático, el reconocimiento de voz o la predicción de series temporales.

El enfoque se basa en entrenar una red neuronal con arquitectura Encoder-Decoder, en la que tanto el encoder como el decoder están implementados mediante redes LSTM. El encoder transforma la secuencia de entrada en una representación vectorial de tamaño fijo que resume la información temporal relevante, y el decoder utiliza dicha representación para generar una secuencia de salida. Esta salida se interpreta como una reconstrucción de la secuencia original. El entrenamiento del modelo se realiza exclusivamente con datos considerados normales, bajo la hipótesis de que, al no haber visto ejemplos anómalos, no podrá reconstruir correctamente este tipo de patrones. En consecuencia, las observaciones anómalas generarán errores de reconstrucción significativamente más altos, lo que permite utilizar dichos errores como criterio para detectar anomalías en los datos de prueba.

4.6.1. Fundamentos de redes neuronales y arquitectura LSTM

Antes de introducir el modelo Encoder-Decoder, es fundamental comprender el funcionamiento de las redes neuronales artificiales, en particular las redes neuronales recurrentes (RNN) y las celdas *Long Short-Term Memory* (LSTM), ya que constituyen la base sobre la cual se construye dicha arquitectura.

Redes neuronales artificiales

Una red neuronal artificial (ANN) es un modelo computacional inspirado en la estructura y el funcionamiento del cerebro humano. Está compuesta por capas de nodos o neuronas artificiales, donde cada nodo realiza una operación matemática sobre sus entradas. La arquitectura típica de una red neuronal incluye tres tipos de capas: una capa de entrada que recibe los datos originales, una o más capas ocultas que procesan la información, y una capa de salida que genera la predicción final (Goodfellow et al. 2016).

En la Figura 4.6 se tiene un ejemplo de una red neuronal con una capa de entrada de 5 nodos, una capa oculta de 4 neuronas y una capa de salida de 2 nodos.

Cada nodo de una capa está conectado a los nodos de la capa anterior mediante un conjunto de pesos w_{ij}, y habitualmente incorpora un término de sesgo b_i. El valor de activación a_i de una neurona se obtiene aplicando una función de activación no lineal ϕ a la suma ponderada de sus entradas:

$$a_i = \phi\left(\sum_j w_{ij}x_j + b_i\right).$$

Estas funciones de activación, como la función sigmoide o la tangente hiperbólica, permiten al modelo aprender relaciones no lineales entre las variables. El flujo de información sigue una estructura de avance hacia adelante desde la capa de entrada hasta la de salida, donde cada nodo calcula su salida a partir de las salidas de la capa anterior.

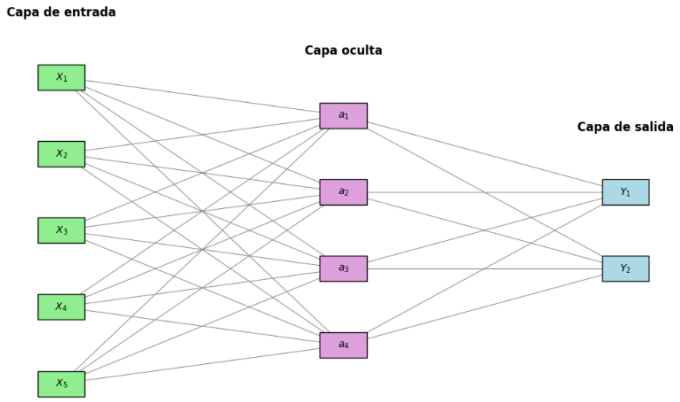


Figura 4.6: Ejemplo de red neuronal con una capa oculta.

De forma más general, si se denota por L el número total de capas, la salida final del modelo se puede representar como:

$$\hat{y} = \phi^{(L)}\left(W^{(L)} \cdot \phi^{(L-1)}\left(\cdots \phi^{(1)}\left(W^{(1)} \cdot \mathbf{x} + b^{(1)}\right)\cdots\right) + b^{(L-1)}\right) + b^{(L)},$$

donde $\mathbf{x}$ representa el vector de entrada, $W^{(l)}$ y $b^{(l)}$ son la matriz de pesos y el vector de sesgos de la capa l, y $\phi^{(l)}$ la función de activación correspondiente a dicha capa.

En problemas de regresión, como la predicción de valores continuos en series temporales, se emplea comúnmente una función de activación lineal en la capa de salida. En cambio, en tareas de clasificación se utilizan funciones como la sigmoide (para clasificación binaria) o la softmax (para clasificación multiclase) (Nielsen 2015).

A continuación se presentan dos de las funciones de activación más utilizadas:

- **Tangente hiperbólica**:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{con rango } [-1, 1].$$

- **Función sigmoide**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad \text{con rango } [0, 1].$$

El proceso de aprendizaje de una red neuronal consiste en ajustar los pesos y sesgos con el objetivo de minimizar una función de pérdida que mide la discrepancia entre las predicciones del modelo $\hat{y}$ y los valores reales y. Este ajuste se realiza mediante un algoritmo de optimización, como el descenso

del gradiente, que actualiza los parámetros en la dirección de mayor disminución de la pérdida. Para calcular los gradientes de forma eficiente se utiliza el algoritmo de retropropagación (*backpropagation*), que aplica la regla de la cadena para propagar el error desde la capa de salida hacia las capas anteriores (Bishop 2006).

Un inconveniente de las redes neuronales tradicionales es su incapacidad para modelar eficazmente datos secuenciales o temporales, debido a que no conservan memoria del contexto anterior. Esta limitación ha motivado el desarrollo de las redes neuronales recurrentes, especialmente diseñadas para procesar secuencias, lo que será abordado en la siguiente sección.

Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes (RNN) surgen como una evolución natural de las redes neuronales artificiales, pensadas específicamente para trabajar con datos secuenciales, como texto, audio o series temporales. A diferencia de las redes *feedforward* tradicionales, en las que la información circula en una única dirección desde la entrada hasta la salida, las RNN introducen conexiones recurrentes que les permiten mantener una especie de memoria dinámica del pasado (Goodfellow et al. 2016).

Esta memoria se consigue gracias a un mecanismo de retroalimentación. En cada instante temporal t, la red no solo procesa la entrada actual x_t, sino que también toma en cuenta el estado oculto anterior h_{t-1}. De este modo, el nuevo estado oculto h_t actúa como un resumen comprimido de toda la información que la red ha recibido hasta ese momento, lo que le permite capturar dependencias temporales a lo largo de la secuencia.

El funcionamiento interno de una RNN se describe matemáticamente con la siguiente expresión:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b),$$

donde x_t es el vector de entrada en el instante t, h_{t-1} representa el estado oculto del paso anterior, W_x y W_h son las matrices de pesos asociadas a la entrada y al estado oculto, b es un vector de sesgo, y ϕ una función de activación no lineal.

Una vez calculado el estado oculto h_t, este se utiliza para generar la salida correspondiente $\hat{y}_t$, que se obtiene mediante una transformación lineal, seguida de una función de activación:

$$\hat{y}_t = \psi(W_y h_t + c),$$

siendo W_y la matriz de pesos de salida, c un vector de sesgo, y ψ una función de activación adaptada a la tarea, como una función lineal para problemas de regresión o *softmax* para clasificación.

Este proceso se repite en cada paso temporal: el estado oculto se actualiza combinando la entrada actual con la memoria del paso anterior, y se genera una predicción que puede depender del contexto acumulado. Gracias a este flujo de información, las RNN están capacitadas para modelar secuencias de datos y extraer patrones temporales complejos.

Sin embargo, en la práctica, las RNN estándar enfrentan ciertas limitaciones. Una de las más importantes es el problema del desvanecimiento del gradiente durante el entrenamiento: a medida que se retropropaga el error a través del tiempo, la magnitud del gradiente puede volverse tan pequeña que impida la actualización efectiva de los pesos en las primeras capas. También puede producirse el fenómeno contrario, la explosión del gradiente, aunque es menos común (Goodfellow et al. 2016).

Para superar estas dificultades, se han propuesto arquitecturas recurrentes más sofisticadas, como las redes LSTM (*Long Short-Term Memory*) y las GRU (*Gated Recurrent Units*). Estas variantes incorporan mecanismos de compuertas internas que regulan el flujo de información, lo que les permite

preservar y actualizar de forma más eficiente la información relevante a lo largo de secuencias extensas.

En la siguiente sección se analizará con mayor detalle la arquitectura LSTM, que constituye la base del modelo Encoder-Decoder.

Celdas Long Short-Term Memory (LSTM)

Para superar las limitaciones de las RNN clásicas, Hochreiter y Schmidhuber (1997) propusieron la arquitectura *Long Short-Term Memory* (LSTM). Este tipo de red introduce un mecanismo interno más sofisticado que permite preservar y controlar el flujo de información a lo largo del tiempo.

La unidad básica de una red LSTM es la celda de memoria, que mantiene un estado interno c_t y un estado oculto h_t. A diferencia de las RNN estándar, las LSTM utilizan un conjunto de compuertas que regulan el acceso, modificación y salida de la información almacenada en la celda. Estas compuertas, implementadas como capas densas con funciones de activación específicas, permiten que la red decida qué información debe olvidar, qué información debe almacenar y qué debe utilizarse en la salida.

Cada celda LSTM consta de tres compuertas principales:

- **Puerta de olvido** (f_t): determina qué parte del estado de memoria anterior c_{t-1} debe descartarse.
- **Puerta de entrada** (i_t): controla qué nueva información se añade al estado de memoria actual.
- **Puerta de salida** (o_t): decide qué parte del estado de memoria se usa para generar el estado oculto h_t.

El flujo de información dentro de una celda LSTM se describe mediante las siguientes ecuaciones:

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{(puerta de olvido)} \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{(puerta de entrada)} \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) && \text{(nueva información candidata)} \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{(estado de memoria)} \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{(puerta de salida)} \\
h_t &= o_t \odot \tanh(c_t) && \text{(estado oculto)}
\end{aligned}
$$

donde $\odot$ representa el producto elemento a elemento (producto Hadamard), σ es la función sigmoide y tanh la tangente hiperbólica. Cada una de las variables tiene el siguiente significado: x_t vector de entrada en el instante t, h_{t-1} estado oculto del paso anterior, c_{t-1} estado de memoria del paso anterior, f_t vector de activaciones de la puerta de olvido, i_t vector de activaciones de la puerta de entrada, $\tilde{c}_t$ vector de nuevos candidatos a memoria, c_t estado de memoria actualizado, o_t vector de activaciones de la puerta de salida, h_t nuevo estado oculto del modelo y finalmente; W_*, U_*, b_* son matrices de pesos y vectores de sesgos que se aprenden durante el entrenamiento.

Gracias a este diseño, las celdas LSTM pueden preservar información útil durante largos intervalos de tiempo, controlar de forma precisa el flujo de información y mitigar los problemas típicos del aprendizaje en secuencias largas.

4.6.2. Descripción teórica del modelo Encoder-Decoder

La arquitectura *Encoder-Decoder* es un marco ampliamente utilizado en tareas secuenciales, caracterizado por su capacidad para transformar una secuencia de entrada en otra de salida, potencialmente

de distinta longitud. Esta arquitectura ha demostrado ser especialmente eficaz en aplicaciones como la traducción automática, el resumen de texto o la predicción de series temporales, gracias a su capacidad para modelar dependencias complejas a lo largo del tiempo (Sutskever et al. 2014; Cho et al. 2014).

En su formulación general, el modelo Encoder-Decoder está compuesto por dos módulos principales:

- **Encoder:** procesa la secuencia de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ y la transforma en una representación densa o vector de contexto $\mathbf{c}$ que resume la información relevante.
- **Decoder:** genera una secuencia de salida $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ a partir del vector de contexto $\mathbf{c}$, de forma autoregresiva.

El encoder actualiza su estado oculto a través de una función recurrente (habitualmente una celda LSTM o GRU), según la siguiente relación:

$$h_t^E = f_E(h_{t-1}^E, x_t), \quad t = 1, \ldots, n$$

donde h_t^E representa el estado oculto del encoder en el instante t. La representación de contexto $\mathbf{c}$ suele definirse como el último estado del encoder: $\mathbf{c} = h_n^E$, que actúa como un resumen comprimido de toda la secuencia de entrada.

A continuación, el decoder utiliza $\mathbf{c}$ como estado inicial y genera la secuencia de salida paso a paso. En el instante t, su estado oculto se actualiza de la siguiente forma:

$$h_t^D = f_D(h_{t-1}^D, \hat{y}_{t-1}, \mathbf{c}), \quad \hat{y}_t = g(h_t^D)$$

donde f_D es una función recurrente que incorpora el estado anterior y la predicción previa, y g es una transformación (normalmente una capa densa seguida de una función de activación) que convierte el estado oculto en una predicción.

La función g transforma el estado oculto en una predicción mediante una capa completamente conectada:

$$\hat{y}_t = \sigma(W h_t^D + b),$$

donde W es la matriz de pesos, b es el vector de sesgo, y σ es una función de activación. En tareas de regresión, como la predicción de series temporales, se suele emplear $\sigma(x) = x$ (función lineal); en cambio, en tareas de clasificación se utilizan funciones como la sigmoide o la *softmax*.

El modelo define la probabilidad conjunta de la secuencia de salida condicionada a la entrada como:

$$P(\hat{y}_1, \ldots, \hat{y}_m \mid x_1, \ldots, x_n) = \prod_{t=1}^{m} P(\hat{y}_t \mid \hat{y}_1, \ldots, \hat{y}_{t-1}, \mathbf{c}).$$

Durante el entrenamiento, se optimizan los parámetros del modelo θ para maximizar la log-verosimilitud condicional de las secuencias observadas:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{y}^{(n)} \mid \mathbf{x}^{(n)}),$$

donde $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ son pares de secuencias del conjunto de entrenamiento. En tareas de regresión, se emplea comúnmente el error cuadrático medio (MSE) como función de pérdida:

$$\mathcal{L} = \frac{1}{m} \sum_{t=1}^{m} (y_t - \hat{y}_t)^2.$$

Gracias a esta arquitectura modular, el modelo Encoder-Decoder puede aprender representaciones latentes que capturan patrones temporales complejos incluso en presencia de ruido, estacionalidad o no linealidades.

4.6.3. Aplicación a series temporales univariantes

La arquitectura *Encoder-Decoder* se adapta especialmente bien a la predicción multihorizonte en series temporales univariantes, es decir, a aquellas tareas en las que se desea estimar varios valores futuros a partir de observaciones pasadas. Este tipo de problema puede entenderse como una traducción de una secuencia histórica en una secuencia futura. En términos formales, el objetivo es aprender una función que realice la siguiente transformación:

$$(x_{t-p+1}, \ldots, x_t) \longrightarrow (\hat{x}_{t+1}, \ldots, \hat{x}_{t+h}),$$

donde p representa el tamaño de la ventana de entrada y h el número de pasos que se desea predecir. Para entrenar el modelo, se recorre la serie completa aplicando una ventana deslizante, lo que permite construir múltiples pares de entrada y salida $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ de la siguiente forma:

$$\mathbf{x}^{(n)} = (x_n, \ldots, x_{n+p-1}), \quad \mathbf{y}^{(n)} = (x_{n+p}, \ldots, x_{n+p+h-1}).$$

Durante el entrenamiento, el encoder recibe cada secuencia de entrada $\mathbf{x}^{(n)}$ y la convierte en un vector de contexto $\mathbf{c}$ que resume la información temporal contenida en dicha ventana. A partir de este vector, el decoder genera paso a paso la secuencia de salida $\hat{\mathbf{y}}^{(n)} = (\hat{x}_{t+1}, \ldots, \hat{x}_{t+h})$, utilizando en cada paso su propia predicción anterior como entrada para el siguiente.

Este enfoque ha demostrado ser especialmente útil para capturar patrones complejos en series temporales, incluyendo relaciones no lineales, estacionalidades y dependencias de largo plazo. Sin embargo, cuando se predicen múltiples pasos de forma autoregresiva puede producirse una acumulación de errores, sobre todo si el horizonte h es amplio. Para abordar este problema, se recurre a técnicas como el *teacher forcing* durante el entrenamiento, o al *scheduled sampling*, que introduce de forma gradual las predicciones del modelo como entradas durante el aprendizaje (Bengio et al. 2015).

La eficacia de esta arquitectura para la predicción multihorizonte ha sido ampliamente respaldada en estudios recientes y aplicaciones reales (Hewamalage et al. 2021; Lim et al. 2021).

Una ventaja adicional del modelo Encoder-Decoder es que, una vez entrenado para predecir valores futuros, puede utilizarse para detectar anomalías, ya que si el modelo ha aprendido a predecir correctamente el comportamiento normal de la serie, entonces los errores de predicción serán notablemente mayores en presencia de datos anómalos.

Este modelo ofrece una solución potente y flexible para la predicción en series temporales univariantes. Su capacidad para manejar relaciones complejas y extenderse a múltiples horizontes lo convierte en una herramienta valiosa, especialmente cuando se dispone de datos abundantes y se busca precisión en escenarios desafiantes. No obstante, su uso también implica ciertos retos: requiere una cantidad considerable de datos para entrenarse adecuadamente, y su alto número de parámetros incrementa el riesgo de sobreajuste si no se emplean técnicas de regularización. Además, su interpretación no siempre resulta intuitiva, lo que puede ser una desventaja frente a modelos más simples, como los lineales o los basados en árboles de decisión. En consecuencia, la elección de esta arquitectura debe realizarse valorando cuidadosamente el equilibrio entre rendimiento, disponibilidad de datos y necesidad de interpretabilidad, en función del problema específico que se desea abordar.

4.7. Consideraciones finales

A lo largo de este capítulo se han desarrollado los fundamentos teóricos y prácticos clave para llevar a cabo la detección de anomalías en series temporales univariantes.

En primer lugar, se han descrito las características fundamentales de una serie temporal, así como los distintos tipos de anomalías que pueden aparecer en su comportamiento. A partir de ello, se ha propuesto una estrategia general de detección basada en la capacidad de ciertos modelos para predecir el comportamiento futuro de la serie, y en el análisis de los residuos generados como indicador de posibles desviaciones anómalas.

Para llevar a cabo esta estrategia, se han revisado tres enfoques representativos desde distintas perspectivas: el modelo estadístico clásico ARIMA, los modelos de aprendizaje automático basados en árboles de decisión (como Random Forest y técnicas de *boosting*), y las arquitecturas neuronales del tipo Encoder-Decoder, especialmente diseñadas para capturar dependencias complejas en datos secuenciales.

Cada uno de estos modelos aporta un enfoque complementario para modelar la dinámica subyacente de la serie temporal. Presentan diferencias tanto en sus supuestos como en su capacidad de representación y en los requisitos de datos para su entrenamiento. Además, se han abordado las métricas más utilizadas para evaluar la calidad de las predicciones y la eficacia de los métodos de detección de anomalías.

Todo el contenido teórico desarrollado en este capítulo servirá de fundamento para el análisis práctico que se presenta a continuación. En el próximo capítulo se pondrán en práctica los modelos descritos, aplicándolos a una serie temporal real. Se evaluará su capacidad para anticipar el comportamiento de la serie y detectar correctamente la presencia de valores anómalos, utilizando los criterios y métricas definidos previamente.

Esta parte experimental permitirá validar la utilidad de los enfoques estudiados y comparar su rendimiento en condiciones reales, ofreciendo una visión más completa de sus fortalezas y limitaciones en contextos aplicados.

Capítulo 5

Aplicación práctica de los métodos

En este capítulo se presentan los resultados obtenidos al aplicar los modelos descritos en los capítulos anteriores a los problemas reales planteados por la entidad financiera ABANCA. Tal como se expuso en la introducción, la empresa identificó dos cuestiones prioritarias en el ámbito de la calidad de los datos: la corrección de valores faltantes en un conjunto de información catastral y la detección de anomalías en series temporales que reflejan el número de registros cargados diariamente en sus sistemas.

Para abordar la primera problemática, se emplea el modelo probabilístico HoloClean, capaz de integrar múltiples fuentes de evidencia para imputar valores nulos de forma precisa y automatizada. En cuanto a la segunda problemática, se construye un sistema de detección de anomalías aplicando tres enfoques complementarios: un modelo estadístico clásico (ARIMA), un algoritmo de aprendizaje automático basado en árboles de decisión (LightGBM) y una arquitectura de redes neuronales profundas del tipo Encoder-Decoder. La elección del modelo más adecuado se realizará en función de su rendimiento en términos de capacidad predictiva y detección efectiva de valores atípicos.

5.1. Imputación de datos nulos: Modelo HoloClean

En esta sección se aplicará el modelo HoloClean sobre un conjunto de datos proporcionado por la entidad financiera ABANCA con el objetivo de evaluar su eficacia en la corrección de registros nulos. La empresa tenía un interés especial en testar la eficacia de este modelo para comprobar si podrían incorporar la herramienta en su proceso de depuración de bases de datos.

El conjunto escogido para testarlo, el cual contiene información de carácter catastral, es importante para la entidad ya que desempeña un papel fundamental en el ámbito de la gestión hipotecaria. Las referencias catastrales que aparecen en dicho dataset identifican los bienes que los clientes aportan como garantía en una solicitud de financiación, uno de los procesos críticos del banco, por lo que garantizar la calidad de este conjunto es crucial para la posterior toma de decisiones en el proceso.

5.1.1. Presentación del conjunto de datos

La base de datos utilizada contiene informacion de caracter catastral, estructurada en dieciseis variables: `REFERECIA_CATASTRAL`, `REFERENCIA_CATASTRAL_ID`, `PROVINCIA_ID`, `AYUNTAMIENTO_ID`, `CODIGO_POSTAL`, `TIPO_VIA`, `VIA`, `NUMERO`, `LETRA`, `BLOQUE`, `ESCALERA`, `PLANTA`, `PUERTA`, `DIRECCION`, `TIPOLOGIA_CATASTRAL` y `REGISTRO`. La primera de ellas, `REFERENCIA_CATASTRAL`, corresponde al identificador único de cada bien inmueble. Esta referencia es un código alfanumérico obligatorio asignado por la Dirección General del Catastro y permite localizar inequívocamente cada inmueble en la cartografía oficial. Las demás variables aportan información sobre la situación geográfica del inmueble identificado por cada referen-

cia.

El objetivo del proyecto, planteado por el departamento de Calidad de Datos de ABANCA, es corregir los registros nulos presentes en las variables `PROVINCIA_ID` y `AYUNTAMIENTO_ID`, que identifican respectivamente el código de la provincia y del ayuntamiento donde está ubicado cada inmueble. Estas dos variables son clave para la localización geográfica y presentaban el mayor número de valores faltantes, lo que dificultaba su uso en procesos posteriores.

Para abordar este problema, se decidió trabajar con un subconjunto reducido de variables del conjunto original, compuesto únicamente por `REFERENCIA_CATASTRAL`, `CODIGO_POSTAL`, `PROVINCIA_ID` y `AYUNTAMIENTO_ID`. En la Tabla 5.1 se tiene una muestra de dicho dataset. Esta selección se justifica porque estas son las variables más relevantes para determinar la ubicación de un inmueble de manera jerárquica y consistente. Las demás variables del dataset; como tipo de vía, número, bloque, escalera o planta, aportan detalles más específicos de la dirección pero no son útiles para inferir la provincia o el ayuntamiento de forma confiable, ya que carecen de relación determinista con estas entidades administrativas y pueden presentar alta variabilidad o ambigüedad.

De este modo, la reducción de variables no solo simplifica el modelo y mejora su eficiencia computacional, sino que se basa en un criterio conceptual sólido: aprovechar únicamente la información que contribuye de forma significativa a predecir la localización geográfica a nivel de provincia y ayuntamiento.

REFERECIA_CATASTRAL	PROVINCIA_ID	AYUNTAMIENTO_ID	CODIGO_POSTAL
9588302ds5098n0181ra	35	016	35002
2257036ng9825e0074pt	32	008	32002
7189965nh4978n0022oa	15	030	15008
36043a021060010001ks	36	043	36828
8221201nj4082s0026hw	15	030	15003

Tabla 5.1: Muestra del conjunto de datos de estudio.

5.1.2. Preprocesado de los datos

Antes de aplicar el modelo HoloClean, se realizó un proceso de preprocesamiento exhaustivo para garantizar la validez y la calidad del conjunto de datos de entrada. La motivación principal de esta limpieza es asegurar que las referencias catastrales utilizadas sean válidas y cumplan con la normativa oficial, evitando así imputaciones incorrectas o inconsistentes que podrían derivarse de datos erróneos.

Según la normativa vigente (Dirección General del Catastro. s.f.), una referencia catastral válida debe cumplir tres requisitos fundamentales:

- Tener una longitud exacta de 20 caracteres alfanuméricos.
- No contener caracteres especiales o símbolos
- Contar con dos dígitos de control finales correctos, calculados mediante un algoritmo de validación.

Aunque la normativa pública especifica la existencia de estos dígitos de control, el algoritmo exacto de su cálculo no se encuentra publicado oficialmente de forma detallada. Sin embargo, existen implementaciones técnicas y documentación sectorial que describen el proceso de validación utilizado en entornos bancarios y administrativos para verificar la corrección formal de las referencias catastrales. En este trabajo se ha adoptado una versión reconstruida del algoritmo de validación, basada en la conversión de caracteres a valores numéricos, un producto escalar con un vector de pesos predefinidos y la obtención de un residuo módulo 23 para la determinación de los dígitos de control.

Este proceso de validación permitió filtrar y eliminar todas aquellas referencias catastrales que no cumplían con estos criterios estrictos. En total, se descartaron 15815 registros inválidos, reduciendo así el tamaño del conjunto de datos original a 252647 registros válidos que cumplen con la estructura requerida.

En la Tabla 5.2 se presenta una muestra de algunos registros eliminados a partir de estas reglas de validación. En efecto, se observa como se tienen diferentes tipos de inconsistencias; algunos registros no cumplen con la longitud pedida mientras que otros contienen caracteres prohibidos.

REFERENCIA_CATASTRAL	PROVINCIA_ID	AYUNTAMIENTO_ID	CODIGO_POSTAL
8584620nh1188s	36	060	36610
u0190138m	48	020	48013
3393910df2739cxj-002			08940

Tabla 5.2: Ejemplos de registros inválidos eliminados durante el preprocesado.

Este filtrado inicial constituye un paso crítico en el proceso de aseguramiento de la calidad de los datos, ya que reduce significativamente el ruido y las inconsistencias estructurales del conjunto de datos, facilitando la aplicación efectiva del modelo HoloClean en las etapas posteriores.

Una vez se ha construido el dataset final que se utilizará para testar el modelo HoloClean, se presentan en la Tabla 5.3 las estadísticas de datos faltantes en este conjunto.

Variable	Nulos absolutos	Porcentaje
`PROVINCIA_ID`	51868	$20,8\,\%$
`AYUNTAMIENTO_ID`	58112	$23,4\,\%$

Tabla 5.3: Porcentaje de registros nulos por variable.

Hacer notar que las variables `CODIGO_POSTAL` y `REFERENCIA_CATASTRAL` no cuentan con registros nulos. Por lo tanto, en términos generales, de las 1010588 celdas que tiene el conjunto de datos, 109980 de ellas están vacías, lo que supone un $10,9\,\%$.

5.1.3. Aplicación del modelo HoloClean

Para la aplicación de HoloClean se siguió la arquitectura y los principios teóricos descritos en el capítulo 3. Para facilitar la implementación, se utilizó el código disponible en el repositorio GitHub, el cual fue adaptado a las necesidades específicas del conjunto de datos proporcionado por ABANCA.

El flujo de trabajo automatizado de HoloClean se basa en los principios descritos por Rekatsinas et al. (2017) que incluye las siguientes fases principales.

Detección de errores

En esta etapa se realiza la identificación de celdas con valores incorrectos. Como ya se expuso en la teoría del método, para este paso puede utilizarse cualquier algoritmo que el usuario decida. Es importante destacar que los resultados del modelo Holoclean dependerán de la eficacia de este método pues determinará si la división entre el dataset con los valores ruidosos y el que contiene los valores limpios está hecha de manera correcta.

En este caso se marcaron como candidatas a reparación aquellas celdas correspondientes a valores ausentes en `PROVINCIA_ID` o `AYUNTAMIENTO_ID`. Así mismo, también se consideraron como celdas erróneas para el modelo aquellas que incumplían dos dependencias funcionales:

`CODIGO_POSTAL` → `PROVINCIA_ID`
`AYUNTAMIENTO_ID` → `PROVINCIA_ID`

Las celdas marcadas conformaron el subconjunto D_n, mientras que el resto de registros completos constituyó D_c y se utilizó como evidencia etiquetada para aprendizaje.

Las dependencias funcionales elegidas establecen que para cada código postal o identificador de ayuntamiento debe existir una única provincia asociada en el conjunto de datos. Durante la compilación del grafo de factores, estas restricciones se modelaron como factores suaves que penalizan las asignaciones inconsistentes pero permiten cierta flexibilidad, incorporando de este modo conocimiento estructural y semántico del dominio en el proceso de inferencia probabilística.

Finalizado este proceso, se detectaron 109980 celdas erróneas o vacías. Lo cual implica que se encontraron todas las celdas nulas que contenía el dataset y además se comprobó que en ningún caso se violaron las dependencias funcionales.

En la Tabla 5.4 se ha escogido un registro marcado como faltante el cual ayudará al lector a seguir el proceso de aplicación del modelo y visualizar los resultados obtenidos en cada paso.

REFERENCIA_CATASTRAL	PROVINCIA_ID	AYUNTAMIENTO_ID	CODIGO_POSTAL
3353707ug4935s0021sj	14		14001

Tabla 5.4: Registro marcado como nulo.

Generación de dominios candidatos

Para cada celda identificada como errónea, se construyó un conjunto reducido de valores plausibles mediante análisis de coocurrencia condicional. Por ejemplo, para un `CODIGO_POSTAL` dado, se recogieron todos los ayuntamientos asociados en los registros limpios junto con sus frecuencias relativas. Esta información se usó para definir los dominios candidatos con un umbral mínimo de frecuencia relativa del 5 %, limitando el tamaño máximo del dominio a 55 valores por celda para garantizar la viabilidad computacional y evitar explotar combinaciones irrelevantes. Los valores de los parámetros del modelo se optimizaron mediante distintas pruebas manuales, en las que se exploraron diferentes configuraciones para seleccionar aquellas que ofrecieran el mejor equilibrio entre precisión de imputación y coste computacional.

A continuación se presentan los resultados de esta etapa para el caso de estudio elegido.

ATRIBUTO	DOMINIO
AYUNTAMIENTO_ID	021, 119, 132, 202, 274, 410

Tabla 5.5: Dominio podado.

Grafo de factores e inferencia probabilística

En esta etapa se tradujeron las restricciones lógicas y las dependencias funcionales en un grafo de factores probabilístico. Cada restricción se modeló como un factor suave con un peso ajustable, de forma que penaliza violaciones pero no las excluye de manera determinista, permitiendo al modelo considerar soluciones plausibles con cierta flexibilidad. Esta relajación de las restricciones garantiza una distribución posterior continua y evita que inconsistencias menores bloqueen la inferencia.

Los pesos asociados a los factores del grafo se aprendieron a partir de la porción limpia del conjunto de datos, utilizando aprendizaje supervisado.

Finalmente, se aplicó muestreo de Gibbs para aproximar la distribución posterior de valores plausibles para cada celda errónea. En cada iteración del muestreo se actualizaba el valor de una celda condicionado a los valores actuales del resto, generando una cadena de Markov que converge a la distribución estacionaria del modelo. Para cada celda se seleccionó como imputación final el valor con mayor probabilidad marginal, incorporando la información derivada de las restricciones, las estadísticas de coocurrencia y el conocimiento externo modelado en los factores del grafo.

ATRIBUTO	VALOR INFERIDO	PROBABILIDAD
AYUNTAMIENTO_ID	021	0,43

Tabla 5.6: Predicción y confianza asociada.

En la Tabla 5.6 está representado el valor que el modelo infiere en este caso para la columna `AYUNTAMIENTO_ID`, 021, junto con su probabilidad asociada 0,43.

5.1.4. Resultados de la imputación

La aplicación del modelo permitió reducir drásticamente la cantidad de registros nulos en el conjunto de datos final como se puede observar en la Tabla 5.7

Variable	Nulos absolutos iniciales	Porcentaje inicial	Nulos absolutos finales	Porcentaje final
`PROVINCIA_ID`	51868	20,8 %	621	0,2 %
`AYUNTAMIENTO_ID`	58112	23,4 %	799	0,3 %

Tabla 5.7: Porcentaje de registros nulos por variable.

Estos resultados representan una corrección automática de más del 98 % de los registros nulos del conjunto de datos.

La empresa dispone de una versión corregida manualmente que se utilizará para validar los resultados. Esta versión fue desarrollada mientras se realizaba la parte de investigación sobre el método HoloClean, ya que, como se ha comentado, la entidad tenía una urgente necesidad de reparar el

dataset. Esta versión corregida contiene únicamente 183 nulos en `PROVINCIA_ID` (0,1 %) y 3017 en `AYUNTAMIENTO_ID` (1,2 %).

Con el fin de evaluar el desempeño del modelo, se calcularon las métricas estándar de clasificación utilizadas habitualmente en tareas de imputación supervisada. Estas métricas permiten cuantificar la calidad de las imputaciones comparándolas con la versión corregida manualmente, considerada como referencia de alta calidad.

Para el cálculo de las métricas se utilizaron las siguientes definiciones:

- **True Positives (TP)**: número de celdas imputadas correctamente.
- **False Positives (FP)**: número de celdas imputadas incorrectamente, es decir, cuyo valor no coincide con el valor verdadero.
- **False Negatives (FN)**: número de celdas nulas que no fueron imputadas por el modelo.

A partir de estas cantidades, se definieron las siguientes métricas:

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  Mide la proporción de imputaciones correctas entre todas las celdas imputadas por el modelo. Un valor alto indica que las imputaciones realizadas son, en su mayoría, fiables.

- **Recall**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  Indica la proporción de valores nulos reales que fueron correctamente imputados. Esta métrica evalúa la cobertura del modelo.

- **F1-score**:

$$F1 = 2\,\frac{\text{Precision x Recall}}{\text{Precision} + \text{Recall}}$$

  Representa la media armónica entre precision y recall. Es especialmente útil cuando se desea un equilibrio entre ambos criterios.

Estas métricas se calcularon de forma independiente para cada una de las dos variables clave. Los resultados obtenidos fueron los siguientes:

Variable	Precision	Recall	F1-score
`PROVINCIA_ID`	1,00	0,997	0,9985
`AYUNTAMIENTO_ID`	0,983	0,948	0,965

Tabla 5.8: Métricas de evaluación del modelo HoloClean.

5.1.5. Conclusiones

La experiencia de aplicar HoloClean sobre un conjunto de datos reales, proporcionados por ABANCA, ha sido muy positiva y demuestra el potencial de este modelo para abordar de forma automatizada uno de los problemas más habituales en el trabajo con bases de datos: la presencia de valores nulos.

El modelo ha logrado imputar de forma correcta la gran mayoría de los registros ausentes, alcanzando una precisión perfecta en la variable `PROVINCIA_ID`, y un rendimiento también muy alto en `AYUNTAMIENTO_ID`. La diferencia entre ambas se puede explicar por la mayor complejidad y variabilidad que presentan los códigos de ayuntamiento frente a los de provincia, lo que dificulta ligeramente el proceso de predicción.

Uno de los aspectos más destacables de HoloClean es su capacidad para combinar información estructural, como dependencias funcionales, con técnicas de aprendizaje probabilístico. Esta combinación le permite hacer imputaciones bien fundamentadas incluso en contextos donde los datos son incompletos o ambiguos. Además, la posibilidad de definir dominios candidatos y modelar restricciones como factores suaves introduce flexibilidad y realismo en el proceso de inferencia.

También resulta relevante que el modelo pueda integrarse de forma razonablemente sencilla en un entorno productivo. Aunque la configuración inicial del sistema y la preparación del conjunto de datos requieren cierto trabajo, el tiempo invertido se compensa claramente con la calidad de los resultados y la reducción del esfuerzo manual en tareas de limpieza.

En definitiva, este estudio ha permitido comprobar de forma empírica que HoloClean no es solo una propuesta teórica potente, sino una herramienta práctica con resultados sólidos. Su incorporación en procesos reales de depuración de datos no solo es posible, sino recomendable, especialmente cuando se manejan volúmenes elevados de información o cuando los recursos para la limpieza manual son limitados.

5.2. Detección de anomalías en series temporales

Una vez expuesto el caso práctico de corrección de valores nulos, en esta sección se abordará el problema de la detección de anomalías en series temporales. En la entidad financiera ABANCA, cada día laborable del año se carga información en la tabla `CONTRATO_POSICION_DIA`, que contiene los registros diarios de cada contrato financiero, incluyendo atributos como saldos, divisas, clasificación contable, entre otros. De forma complementaria, la tabla `CONTRATO` recoge datos asociados, tales como la aplicación origen del contrato. Actualmente, la entidad gestiona contratos provenientes de más de 100 aplicaciones distintas. Entre ellas, la aplicación *Avales* destaca por su relevancia operativa. Para esta aplicación, se ha construido una serie temporal que refleja el número de registros cargados diariamente, y será utilizada como base para el presente estudio de detección de anomalías.

Desde el departamento de Calidad de Datos se planteó la necesidad de incorporar un mecanismo de control sobre esta serie temporal, con el fin de detectar automáticamente desviaciones significativas respecto al comportamiento esperado. Estas anomalías pueden deberse a múltiples causas, como fallos técnicos que deriven en pérdidas de registros, o bien duplicaciones no intencionadas que incrementen el volumen cargado. Ante cualquiera de estos escenarios, es crucial que se active una alerta que permita al equipo responsable revisar y corregir la situación con lo más rápido posible.

Hasta el momento la empresa no tiene un control automatizado para esta serie, por lo que se revisan manualmente sus registros de forma periódica para comprobar si en algún momento difieren de lo esperado. El protocolo de actuación frente a una sospecha de anomalía comienza por un análisis de la posible causa. Una vez localizado el origen del problema, se procede, si es viable, a cargar los

datos correctos. En caso de no ser posible recuperar la información válida, se opta por imputar el valor replicando el del día anterior, minimizando así los efectos del error sobre los sistemas analíticos posteriores.

Debido a esta metodología de corrección inmediata, la serie histórica almacenada no conserva los valores anómalos originales, los cuales no se guardan en ningún otro lugar. Esto implica que el conjunto de entrenamiento disponible no incluye registros etiquetados como anomalías reales, ya que estos fueron corregidos en su momento. No obstante, a partir del 15 de noviembre de 2024, se empezó a registrar de forma manual todo registro sospechoso de ser anómalo para poder utilizarlo a la hora de verificar la capacidad de los modelos para detectar anomalías.

5.2.1. Presentación y tratamiento del conjunto de datos

En este estudio se ha utilizado la serie temporal univariante correspondiente a la aplicación *Avales*, la cual refleja el número de registros cargados diariamente en el sistema, desde el 01/01/2020 hasta el 06/03/2025. La frecuencia de muestreo es diaria, considerando únicamente los días laborables, de lunes a viernes, y excluyendo los días festivos. En total, la serie contiene 1414 observaciones.

Con el fin de preservar una estructura temporal regular y facilitar el posterior ajuste de modelos, se ha aplicado un procedimiento de interpolación lineal exclusivamente en los días festivos, sin modificar el resto de observaciones. Este método consiste en estimar los valores faltantes calculando una recta entre las observaciones inmediatamente anterior y posterior a cada hueco, y asignar al día ausente un valor intermedio proporcional al tiempo transcurrido. Se trata de una técnica simple y eficiente que resulta adecuada cuando se espera una evolución suave y continua entre los puntos adyacentes.

En el contexto bancario, donde las operaciones presentan cierta regularidad en el corto plazo y donde la imputación no debe introducir oscilaciones artificiales, la interpolación lineal constituye una solución conservadora y apropiada. Su uso permite mantener la coherencia de las series sin inducir distorsiones estadísticas que puedan afectar negativamente a la detección de anomalías.

No se han identificado valores nulos adicionales ni registros duplicados, por lo que no ha sido necesaria una limpieza adicional de los datos.

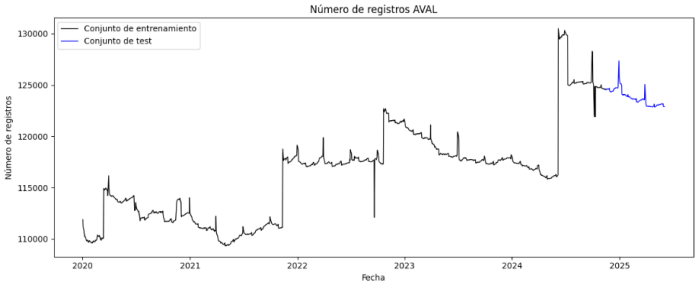


Figura 5.1: Serie temporal asociada a la aplicación Avales.

La Figura 5.1 muestra la evolución temporal de la serie. A partir de una inspección visual preliminar, se observa una tendencia general creciente a lo largo del periodo analizado, interrumpida por varios saltos bruscos. Estos cambios suelen corresponderse con integraciones o modificaciones estructurales en los sistemas de carga de datos llevadas a cabo por la entidad. así mismo no se aprecia un patrón estacional definido.

Para complementar el análisis visual, en la Tabla 5.9 se resumen las estadísticas descriptivas la serie.

Serie	Media	Mediana	Desv. típ.	Mínimo	Máximo
Avales	117093, 88	117404, 50	4870, 77	109335, 00	130505, 00

Tabla 5.9: Estadísticas descriptivas.

5.2.2. Metodología general

Para abordar el problema de detección de anomalías en la serie temporal presentada, se ha diseñado un esquema metodológico común que será aplicado a los tres modelos seleccionados: ARIMA, LightGBM y Encoder-Decoder. Este procedimiento permite garantizar una comparación homogénea de los resultados y evaluar la capacidad predictiva y de detección de anomalías de cada enfoque bajo condiciones equivalentes.

El primer paso consiste en dividir la serie temporal en dos subconjuntos: un conjunto de entrenamiento, que abarca desde el 01/01/2020 hasta el 14/11/2024, conteniendo alrededor del 90 % de los datos y un conjunto de test, que comienza el 15/11/2024 y se extiende hasta el 06/03/2025, con un 10 % de los registros. Esta división no es aleatoria ya que el conjunto de test se ha tomado a partir del día que se comenzaron a guardar los registros de las anomalías.

Dentro del conjunto de entrenamiento, se ha llevado a cabo una segunda división para disponer de un conjunto de validación que permita la optimización de los hiperparámetros de los modelos. Concretamente, se ha reservado el 20 % final del conjunto de entrenamiento para validación, comenzando dicho segmento el 24/11/2023.

Durante la fase de entrenamiento, cada modelo es ajustado utilizando únicamente el 80 % inicial del conjunto de entrenamiento, y posteriormente evaluado sobre el conjunto de validación. Esta evaluación se repite para diferentes combinaciones de hiperparámetros, siguiendo un procedimiento de búsqueda sistemática (*grid search*). La configuración óptima de hiperparámetros se selecciona como aquella que minimiza el valor del RMSE en el conjunto de validación.

Una vez entrenados y validados los modelos, se evalúa su capacidad de predicción sobre el conjunto de test a partir de las métricas descritas en el Capítulo 4: MAE, RMSE y MAPE.

Estas métricas permiten cuantificar la precisión del modelo en la predicción del comportamiento normal de la serie. A partir de los residuos obtenidos durante el entrenamiento (diferencias entre los valores reales y los valores estimados), se calcularán también los límites de control estadístico que servirán para detectar desviaciones significativas en el conjunto de test.

En concreto, se adopta un enfoque basado en gráficos de control aplicados sobre los residuos, utilizando como umbrales los siguientes límites:

$$\text{Limite superior} = \mu_e + k \cdot \sigma_e \qquad \text{y} \qquad \text{Limite inferior} = \mu_e - k \cdot \sigma_e,$$

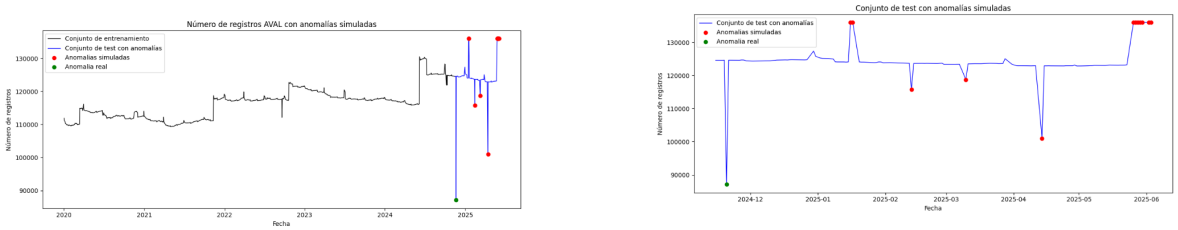
en los que μ_e y σ_e representan, respectivamente, la media y la desviación típica de los errores de predicción en el conjunto de entrenamiento, y k es un parámetro de sensibilidad. En este estudio se ha fijado $k = 3$, valor que corresponde al umbral clásico utilizado en los gráficos de control de Shewhart, ya que bajo una distribución aproximadamente normal de los errores permite capturar un $99,7\,\%$ de

los casos esperados bajo comportamiento normal, reduciendo así la tasa de falsos positivos sin perder capacidad de detección ante desviaciones relevantes.

La estrategia adoptada consiste en calcular, para cada observación del conjunto de test, el error absoluto de predicción y compararlo con estos límites. Si dicho error supera alguno de los umbrales, se considera que existe una anomalía potencial en esa observación.

Dado que durante el periodo monitorizado solo se ha registrado una anomalía real, se ha optado por complementar el análisis mediante la simulación de anomalías adicionales en el conjunto de test. La única anomalía identificada por el equipo de calidad tuvo lugar el día 20/11/2024, donde el número de registros descendió de forma abrupta desde 124614 a 87097, sin que existiera una justificación operativa o técnica aparente para dicha caída.

Para diseñar anomalías simuladas con un mayor realismo, se mantuvo una comunicación directa con el equipo responsable de la monitorización diaria en ABANCA. Gracias a ello, se logró caracterizar el tipo de desviaciones más frecuentes y relevantes en la serie temporal. Se identificaron principalmente anomalías puntuales; es decir, observaciones aisladas claramente fuera de rango, así como cambios bruscos de tendencia. En la Figura 5.2 se tiene una representación de las anomalías simuladas sobre la serie original. Dentro del conjunto de test, que comprende un total de 144 observaciones, se han incorporado 11 anomalías artificiales repartidas en distintos puntos de la serie, además de la anomalía real ya mencionada anteriormente, sumando un total de 12 valores marcados como anómalos. La Tabla 5.10 resume las fechas seleccionadas, los valores reales observados y los valores anómalos introducidos, así como la diferencia entre ambos. Esta información permitirá una evaluación precisa del rendimiento de los modelos propuestos en la detección de este tipo de desviaciones.



(a) Gráfico de la serie con las anomalías simuladas.

(b) Gráfico del conjunto de test con las anomalías simuladas.

Figura 5.2: Anomalías simuladas en la serie de Avales.

La evaluación del rendimiento en la detección de anomalías se realizará mediante métricas de clasificación estándar:

- **Precision**: proporción de anomalías detectadas correctamente respecto al total de anomalías detectadas por el modelo.
- **Recall**: proporción de anomalías reales detectadas respecto al total de anomalías introducidas.
- **F1-score**: media armónica entre precisión y recall.

Estas métricas ofrecen una visión global sobre la eficacia de cada modelo, permitiendo comparar su capacidad tanto para evitar falsos positivos como para no pasar por alto anomalías relevantes. El modelo que obtenga el mejor desempeño en estas métricas será considerado como el más adecuado para su implantación práctica en la entidad.

Fecha	Valor real	Valor simulado (anómalo)	Diferencia
20/11/2024	124614	87097	−37517
16/01/2025	124065	135980	+11915
17/01/2025	124091	135980	+11889
13/02/2025	123677	115765	−7912
10/03/2025	123462	118762	−4700
14/04/2025	122889	100951	−21938
26/05/2025	123176	135980	+12804
27/05/2025	123143	135980	+12837
28/05/2025	123152	135980	+12828
29/05/2025	123151	135980	+12829
02/06/2025	122898	135980	+13082
03/06/2025	122882	135980	+13098

Tabla 5.10: Comparación entre los valores reales y los valores simulados como anomalías en la serie de Avales.

Una vez que se ha expuesto la metodología a seguir, se presentan en las próximas secciones los resultados obtenidos por cada uno de los modelos.

5.2.3. Modelo ARIMA

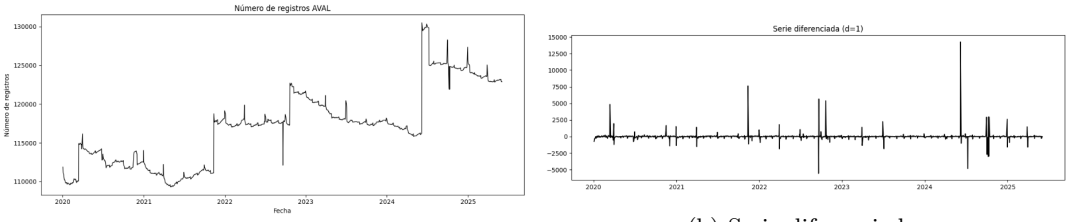
El primer enfoque que se aplicará para la detección de anomalías en las series temporales es el modelo ARIMA, uno de los métodos clásicos más utilizados para modelar dependencias lineales en datos secuenciales.

Antes de proceder a la estimación de los parámetros del modelo, resulta imprescindible realizar un análisis exploratorio de la serie, con el objetivo de identificar sus principales características estructurales. Entre los aspectos más relevantes a estudiar se encuentran la presencia de tendencias, estacionalidad y estacionariedad. Este análisis permitirá determinar si es necesario aplicar transformaciones como diferenciación o incluir componentes estacionales en el modelo.

Análisis exploratorio de la serie

Antes de ajustar un modelo ARIMA sobre la serie de *Avales*, se llevó a cabo un análisis de estacionariedad con el fin de determinar si era necesario aplicar transformaciones para estabilizar su media y varianza a lo largo del tiempo.

La Figura 5.3 muestra la evolución de la serie original junto con su versión diferenciada de orden $d = 1$. Se observa que la serie original presenta una clara tendencia creciente, mientras que en la serie diferenciada se aprecia una media constante.



(a) Serie original.

(b) Serie diferenciada.

Figura 5.3: Comparación de la serie original Avales con la serie diferenciada de orden d=1.

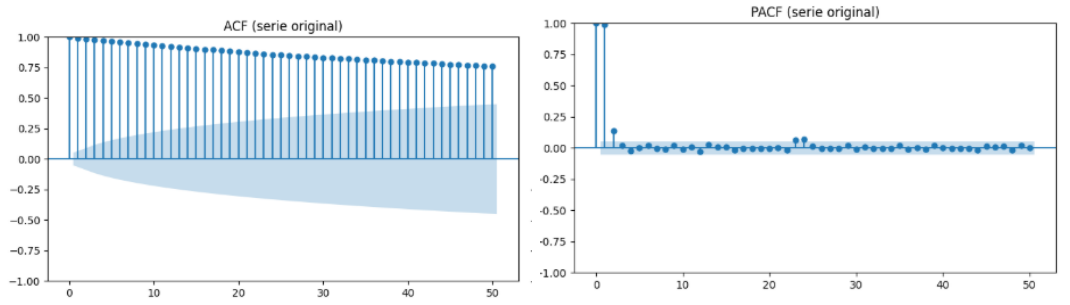
Para evaluar formalmente la estacionariedad de la serie temporal, se ha empleado el test aumentado de Dickey-Fuller (ADF), una prueba estadística ampliamente utilizada en el análisis de series temporales (Dickey y Fuller 1979). El objetivo del test es contrastar la hipótesis nula de que la serie posee una raíz unitaria, es decir, que es no estacionaria. La forma general del modelo sobre el que se aplica el test ADF es la siguiente:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta y_{t-i} + \varepsilon_t,$$

donde Δy_t representa la primera diferencia de la serie y_t, t es una tendencia determinista (opcional), ε_t es un término de error aleatorio, y p es el número de retardos incluidos para corregir la autocorrelación de los residuos. La hipótesis nula del test es $H_0 : \gamma = 0$ (la serie es no estacionaria), frente a la alternativa $H_1 : \gamma < 0$ (la serie es estacionaria).

Los resultados indican que la serie original no es estacionaria (p-valor $= 0,2859$), mientras que la serie diferenciada sí lo es (p-valor $= 0,0000$). En consecuencia, se concluye que es necesario aplicar una diferenciación de orden $d = 1$ para garantizar la estacionariedad del proceso.

Este diagnóstico se ve reforzado por el análisis de autocorrelaciones. La Figura 5.4 muestra las funciones ACF y PACF de la serie original, donde puede apreciarse una lenta disminución de la ACF, típica de procesos no estacionarios, y un corte rápido en la PACF tras el primer retardo. Esto respalda la elección de un modelo ARIMA con componente de diferenciación.



(a) Función de autocorrealciones (ACF).

(b) Función de autocorrelaciones parciales (PACF).

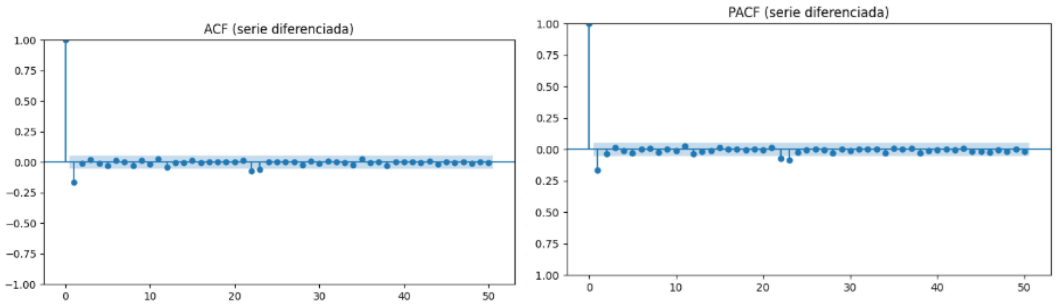
Figura 5.4: Funciones de autocorrelaciones y autocorrelaciones parciales de la serie Avales.

En la Figura 5.3, se observa la estabilización del nivel de la serie tras la transformación, así como por las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) mostradas en la Figura

5.5. En dicha figura se aprecia una rápida disminución de las autocorrelaciones en la serie diferenciada, lo cual es característico de una serie estacionaria.

Además, a partir del análisis visual de la serie y de sus funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), no se observa un patrón estacional claro ni picos recurrentes en retardos específicos que indiquen estacionalidad, lo cual refuerza la hipótesis de que no es necesario incorporar componentes estacionales al modelo ARIMA.

En consecuencia, se procederá a ajustar un modelo ARIMA sobre la serie utilizando una diferenciación de orden $d = 1$, tomando los órdenes asociados a la parte estacional todos iguales a cero.



(a) Función de autocorrealciones (ACF).

(b) Función de autocorrelaciones parciales (PACF).

Figura 5.5: Funciones de autocorrelaciones y autocorrelaciones parciales de la serie Avales con una diferencia.

Estimación del modelo ARIMA

Una vez establecida la necesidad de diferenciar la serie una vez ($d = 1$) para lograr la estacionariedad, y descartada la presencia de componentes estacionales, se procedió a determinar los valores óptimos de los parámetros p y q, correspondientes a los órdenes del componente autorregresivo (AR) y de media móvil (MA), respectivamente.

Para ello, se evaluaron múltiples configuraciones del modelo ARIMA$(p, 1, q)$, restringiendo los valores de p y q al rango $[0, 5]$, y estimando cada combinación sobre el conjunto de entrenamiento. La validación se llevó a cabo empleando un subconjunto reservado desde el 24/11/2023 y utilizando como métrica de evaluación el RMSE. La combinación de parámetros que minimizó dicho error fue la correspondiente al modelo ARIMA$(1, 1, 0)$. Este modelo fue ajustado sobre toda la serie de entrenamiento.

Diagnóstico de residuos del modelo ARIMA

Una vez estimado el modelo ARIMA(1,1,0) sobre la serie temporal, se procede al análisis de los residuos con el objetivo de verificar que cumplan las hipótesis fundamentales del modelo: media nula, varianza constante, incorrelación temporal y normalidad. Estas condiciones aseguran que el modelo ha captado correctamente la estructura subyacente de la serie. La Figura 5.6 muestra la evolución temporal de los residuos, donde se observa una oscilación centrada en torno a cero, con algunas observaciones extremas puntuales.

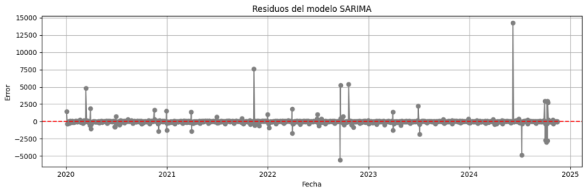


Figura 5.6: Serie temporal de los residuos del modelo ARIMA.

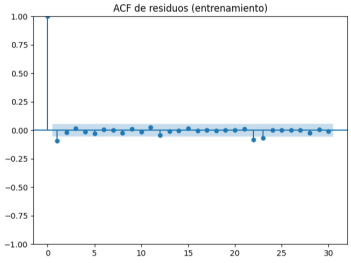


Figura 5.7: Función de autocorrelación (ACF) de los residuos.

En cuanto a la correlación de los residuos, la Figura 5.7 muestra su función de autocorrelación, donde la mayoría de los coeficientes se encuentran dentro del intervalo de confianza del 95 %. Este patrón es característico de un proceso de ruido blanco.

Esta evidencia se respalda por el test de Ljung-Box (Ljung y Box 1978), una prueba estadística ampliamente utilizada para evaluar la independencia de los residuos en un modelo de series temporales. Su objetivo es contrastar la hipótesis nula de ausencia de autocorrelación conjunta hasta cierto número de retardos. Su estadístico de contraste está definido como:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k},$$

donde $\hat{\rho}_k$ representa el coeficiente de autocorrelación muestral en el retardo k, n es el tamaño de la muestra y h el número de retardos considerados. Bajo la hipótesis nula, el estadístico Q sigue una distribución chi-cuadrado con h grados de libertad. En el presente análisis, se ha considerado un valor de $h = 10$, y el p-valor obtenido fue de 0,139, lo que permite mantener la hipótesis nula al nivel del 5 % de significación. Esto indica que no existe evidencia significativa de autocorrelación en los residuos, lo cual es coherente con un comportamiento de ruido blanco.

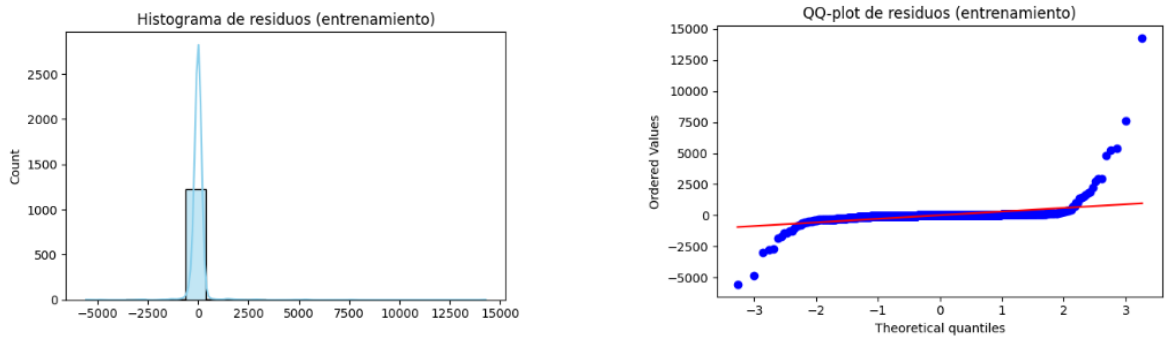
La Figura 5.8 presenta un histograma y un gráfico QQ-plot de los residuos. El histograma sugiere una forma aproximadamente simétrica, pero el QQ-plot revela una cierta desviación respecto a la distribución normal, especialmente en las colas. Esto indica la presencia de valores extremos que podrían afectar a la hipótesis de normalidad.

Para apoyar al análisis visual, se ha aplicado el test de Shapiro-Wilk (Shapiro y Wilk 1965), una prueba estadística ampliamente utilizada por su alta sensibilidad para detectar desviaciones de la normalidad, especialmente en muestras pequeñas o moderadas. Este test tiene como hipótesis nula que la muestra proviene de una distribución normal. EL estadístico de contraste W se define como:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

expresión en la que $x_{(i)}$ son los datos ordenados de menor a mayor, $\bar{x}$ es la media muestral, y los coeficientes a_i se calculan a partir de los valores esperados y la matriz de covarianzas de una muestra ordenada de tamaño n extraída de una distribución normal estándar. El estadístico W toma valores en el intervalo $(0, 1]$, donde valores cercanos a 1 indican normalidad.

En el análisis realizado a la serie, se obtuvo un p-valor de prácticamente igual a cero, lo que permite rechazar de forma concluyente la hipótesis nula al nivel del 5 %.



(a) Histograma de residuos.

(b) QQ-plot de residuos.

Figura 5.8: Evaluación de la normalidad de los residuos del modelo ARIMA.

En conjunto, estos resultados permiten concluir que los residuos del modelo ARIMA(1,1,0) pueden considerarse como ruido blanco: son incorrelados, tienen media cercana a cero, varianza constante y no presentan autocorrelación significativa. Aunque el test de normalidad de Shapiro-Wilk ha rechazado la hipótesis de normalidad, esta desviación no invalida la validez del modelo, ya que las principales aplicaciones del modelo ARIMA, como la predicción y la detección de anomalías, no requieren estrictamente que los residuos sean normales, sino que se comporten como un proceso no estructurado. Por tanto, el modelo se considera adecuadamente ajustado desde el punto de vista estadístico y puede ser empleado con garantías para las tareas posteriores sobre la serie.

Evaluación del modelo ARIMA y detección de anomalías

Una vez validado el modelo ARIMA(1,1,0) sobre el conjunto de entrenamiento, se procedió a evaluar su capacidad de predicción en el conjunto de test, comprendido entre el 15/11/2024 y el 03/06/2025, con un total de 144 observaciones. Las predicciones se generaron de forma autorregresiva, utilizando los valores reales pasados para iterar en cada paso de predicción.

Para cuantificar el rendimiento del modelo en este periodo, se emplearon las tres métricas propuestas en el documento: el error absoluto medio (MAE), la raíz del error cuadrático medio (RMSE) y el error porcentual absoluto medio (MAPE).

Los valores obtenidos para el modelo ARIMA(1,1,0) sobre la serie Avales se resumen en la Tabla 5.11, en la que se comparan los errores en el conjunto de entrenamiento y test.

Estos resultados evidencian un ajuste adecuado del modelo a los datos: el error en el conjunto de test se mantiene próximo al obtenido durante el entrenamiento, sin indicar signos de sobreajuste ni de pérdida de capacidad predictiva. La baja magnitud del MAPE (inferior al 0,13 % en ambos conjuntos) refleja que el modelo logra anticipar con precisión el comportamiento normal de la serie, proporcionando una base sólida para la detección de posibles anomalías mediante análisis de residuos.

Conjunto	MAE	RMSE	MAPE (%)
Entrenamiento	123, 222	468, 43	0, 11
Test	155, 79	661, 56	0, 12

Tabla 5.11: Métricas de error del modelo ARIMA(1,1,0) en los conjuntos de entrenamiento y test.

Evaluación de la detección de anomalías

Una vez ajustado el modelo ARIMA(1,1,0), se procede a evaluar su capacidad para detectar valores anómalos en la serie. Para ello, se analiza el comportamiento de los errores de predicción en el conjunto de test mediante un gráfico de control. En este enfoque, se considera que una observación es anómala cuando el error de predicción supera significativamente el comportamiento habitual observado durante el entrenamiento.

Los límites de control se establecieron calculando la media y desviación estándar de los errores en el conjunto de entrenamiento, fijando los umbrales como:

$$\text{Límite superior} = \mu_e + 3\sigma_e \quad . \quad \text{Límite inferior} = \mu_e - 3\sigma_e$$

La Figura 5.9 muestra el gráfico de control con los errores de test, donde se identifican las observaciones que sobrepasan los límites y, por tanto, se consideran potencialmente anómalas.

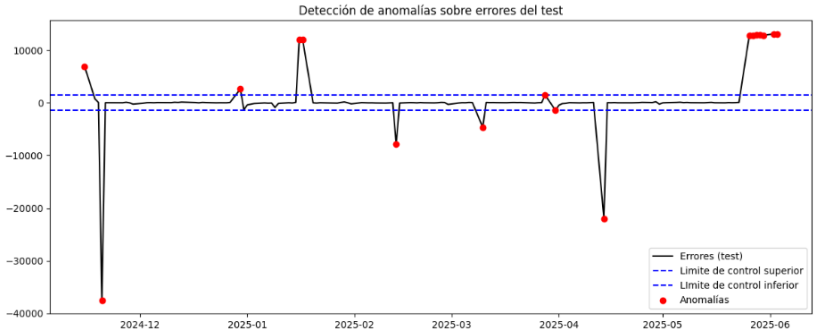


Figura 5.9: Detección de anomalías mediante gráfico de control sobre errores del test.

El modelo detectó un total de 17 anomalías. La Tabla 5.12 recoge estas observaciones junto con la magnitud del error absoluto y una etiqueta que indica si coinciden con alguna de las anomalías simuladas o reales (positivos verdaderos), o si se trata de falsos positivos.

De las 17 anomalías detectadas, 12 fueron verdaderas (11 simuladas y 1 real) y 5 correspondieron a falsos positivos. A partir de esta clasificación, se obtienen las siguientes métricas de evaluación:

- **Precision** $= 0, 71$
- **Recall** $= 1, 0$
- **F1-score**: $= 0, 83$

Estos resultados evidencian que el modelo ARIMA, junto con la estrategia de detección basada en gráficos de control, es altamente efectivo para identificar valores anómalos, logrando detectar la totalidad de las anomalías simuladas con un bajo número de falsos positivos.

Fecha	Error absoluto	Clasificación
15/11/2024	6872, 13	Falso positivo
20/11/2024	37511, 60	Anomalía real
30/12/2024	2622, 76	Falso positivo
16/01/2025	11912, 98	Simulada
17/01/2025	11914, 05	Simulada
13/02/2025	7894, 32	Simulada
10/03/2025	4636, 67	Simulada
28/03/2025	1477, 48	Falso Positivo
31/03/2025	1443, 86	Falso positivo
14/04/2025	22004, 56	Simulada
26/05/2025	12811, 82	Simulada
27/05/2025	12804, 21	Simulada
28/05/2025	12833, 51	Simulada
29/05/2025	12832, 93	Simulada
30/05/2025	12802, 07	Falso Positivo
02/06/2025	13051, 17	Simulada
03/06/2025	13072, 95	Simulada

Tabla 5.12: Anomalías detectadas por el modelo ARIMA y su correspondencia con las anomalías simuladas.

5.2.4. Modelo LightGBM

El segundo enfoque aplicado para la detección de anomalías en la serie de Avales ha sido un modelo de aprendizaje automático basado en árboles de decisión: LightGBM. Este modelo pertenece a la familia de los métodos *gradient boosting* desarrollados en el Capítulo 4. Se trata de una técnica ampliamente utilizada en tareas de predicción tanto supervisadas como no supervisadas debido a su alto rendimiento y flexibilidad.

En términos generales, el *boosting* consiste en construir un conjunto de modelos débiles (habitualmente árboles de decisión de baja profundidad) de forma secuencial, donde cada nuevo árbol se entrena para corregir los errores cometidos por el conjunto anterior. A diferencia del método *bagging* utilizado, por ejemplo, en Random Forests, donde se entrenan modelos de forma paralela y se combinan mediante votación o promedio, el *boosting* se caracteriza por un entrenamiento secuencial y por ponderar más las observaciones difíciles de predecir.

LightGBM (*Light Gradient Boosting Machine*) es una implementación optimizada de esta técnica, desarrollada por Microsoft, que utiliza histogramas discretizados y estrategias de crecimiento de árbol basadas en *leaf-wise* en lugar de *level-wise*, lo que le permite mejorar tanto la velocidad de entrenamiento como la precisión en comparación con otras bibliotecas de boosting tradicionales como XGBoost

o AdaBoost (Microsoft, 2024). Asimismo, es capaz de manejar eficientemente grandes volúmenes de datos, variables categóricas y relaciones no lineales entre las variables.

Una de las ventajas principales del modelo LightGBM frente al modelo ARIMA radica en su capacidad para incorporar múltiples variables explicativas, lo que permite enriquecer la información disponible para la predicción mediante técnicas de ingeniería de características. Además, este modelo no requiere que la serie sea estacionaria ni que se especifique explícitamente la estructura temporal, lo que lo convierte en una alternativa robusta para series con dinámicas complejas.

Construcción del modelo y selección de variables

La construcción del modelo LightGBM necesitó de un proceso previo de ingeniería de características, ya que este tipo de algoritmos no modela directamente la estructura secuencial de la serie, sino que requiere que las dependencias temporales se expresen mediante variables adicionales derivadas.

Con este objetivo, se exploraron distintas combinaciones de transformaciones basadas en retardos y estadísticas móviles de la serie original, evaluando su impacto en el rendimiento predictivo sobre un conjunto de validación. Finalmente, se seleccionó un conjunto de siete variables explicativas que permitieron capturar tanto la dinámica local de corto plazo como cierta información sobre la estabilidad del proceso. Las variables seleccionadas fueron las siguientes:

- **Retardos temporales (lags):**
  Se incorporaron los tres primeros retardos temporales de la serie original:

  $$y_{t-1}, \quad y_{t-2}, \quad y_{t-3}$$

  Estas variables permiten capturar la dependencia inmediata entre los valores pasados y la observación actual, aportando información directa sobre la inercia del sistema.

- **Media móvil de orden 3 y 5:**
  Se incluyeron dos medias móviles centradas en el instante anterior:

  $$\overline{y}_t^{(3)} = \frac{1}{3}(y_{t-1} + y_{t-2} + y_{t-3}), \qquad \overline{y}_t^{(5)} = \frac{1}{5}(y_{t-1} + \cdots + y_{t-5})$$

  Estas variables suavizan la serie y permiten detectar desviaciones respecto al comportamiento promedio reciente, facilitando la detección de anomalías locales.

- **Desviación típica móvil de orden 3 y 5:**
  Se calcularon dos medidas de dispersión móvil:

  $$\sigma_t^{(k)} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(y_{t-i} - \overline{y}_t^{(k)}\right)^2}, \quad \text{para } k = 3, 5$$

  donde $\overline{y}_t^{(k)}$ representa la media móvil de orden k definida previamente. Estas variables proporcionan información sobre la variabilidad reciente de la serie, lo cual es útil para identificar posibles alteraciones en el comportamiento del proceso, como saltos, caídas repentinas o cambios en la dispersión.

La selección final de estas variables se basó en un procedimiento de búsqueda sistemática, evaluando distintas combinaciones sobre un conjunto de validación temporal que preserva la estructura cronológica de los datos. Esta estrategia permitió identificar la configuración que ofrecía el mejor equilibrio entre precisión predictiva y simplicidad del modelo.

Entrenamiento y optimización

El modelo fue entrenado utilizando la biblioteca `LightGBM`, aplicando una búsqueda en rejilla (*grid search*) sobre el conjunto de validación, con el objetivo de identificar la combinación de hiperparámetros que proporcionara el mejor rendimiento en términos de error cuadrático medio (RMSE).

Tras la evaluación de múltiples configuraciones, los hiperparámetros seleccionados fueron los siguientes:

- `boosting_type`: `gbdt` — tipo de técnica de ensamblado utilizada. Se emplea *Gradient Boosting Decision Tree*, el método estándar basado en gradientes.
- `learning_rate`: $0,1$ — tasa de aprendizaje que regula el peso de cada árbol en la secuencia de boosting. Valores más bajos favorecen un entrenamiento más estable pero requieren mayor número de iteraciones.
- `n_estimators`: 200 — número total de árboles (iteraciones) que se construirán en el modelo.
- `num_leaves`: 31 — número máximo de hojas por árbol. Controla la complejidad del modelo: valores más altos permiten capturar interacciones más complejas, aunque con mayor riesgo de sobreajuste.
- `max_depth`: -1 — profundidad máxima permitida para cada árbol. El valor -1 indica que no se impone un límite explícito, y el crecimiento del árbol está controlado por `num_leaves`.
- `min_child_samples`: 20 — número mínimo de muestras necesarias en una hoja para que se permita su división. Actúa como regularizador para evitar sobreajuste en muestras pequeñas.
- `min_child_weight`: $0,001$ — umbral mínimo para la suma de pesos de las observaciones en una hoja. También ayuda a prevenir divisiones no significativas.
- `subsample`: $1,0$ — fracción de muestras seleccionadas aleatoriamente sin reemplazo para entrenar cada árbol. Un valor de $1,0$ indica que se utiliza el conjunto completo en cada iteración.
- `colsample_bytree`: $1,0$ — proporción de características seleccionadas aleatoriamente para construir cada árbol. Con $1,0$, se utilizan todas las variables disponibles.
- `reg_alpha`: $0,0$ — coeficiente de regularización L1. Penaliza la complejidad del modelo sumando el valor absoluto de los coeficientes. En este caso, no se aplicó regularización L1.
- `reg_lambda`: $0,0$ — coeficiente de regularización L2, que penaliza la magnitud de los coeficientes al cuadrado. Tampoco se aplicó regularización L2.
- `random_state`: 42 — semilla utilizada para garantizar la reproducibilidad de los resultados.

Resultados de predicción

El modelo se evaluó sobre el conjunto de test utilizando las métricas MAE, RMSE y MAPE, con el fin de cuantificar su capacidad para anticipar el comportamiento normal de la serie. Los resultados están reflejados en la Tabla 5.13.

Los resultados muestran que el modelo LightGBM ofrece un buen rendimiento predictivo en el conjunto de test, con un error medio absoluto (MAE) de $363,15$ y un MAPE inferior al $0,3\,\%$, lo que indica una alta precisión relativa en la estimación de los valores diarios. Aunque se observa un incremento en los errores respecto al conjunto de entrenamiento, este aumento es moderado y esperable, sugiriendo que el modelo generaliza razonablemente bien sin incurrir en sobreajuste. La baja magnitud del MAPE en ambos conjuntos confirma que el modelo ha capturado adecuadamente la dinámica subyacente de la serie, lo que proporciona una base sólida para la posterior identificación de anomalías mediante análisis de residuos.

Conjunto	MAE	RMSE	MAPE (%)
Entrenamiento	156,04	481,61	0,13 %
Test	363,15	558,05	0,29 %

Tabla 5.13: Métricas de error del modelo LightGBM en los conjuntos de entrenamiento y test.

Detección de anomalías

Como en el caso del modelo ARIMA, se empleó una estrategia de control estadístico para detectar observaciones atípicas en el conjunto de test. Para ello, se calcularon los residuos del modelo en el conjunto de entrenamiento y se construyeron los límites de control a partir de la desviación estándar de estos residuos.

La Figura 5.10 muestra el gráfico de control con los residuos en el conjunto de test, destacando las observaciones identificadas como anómalas.

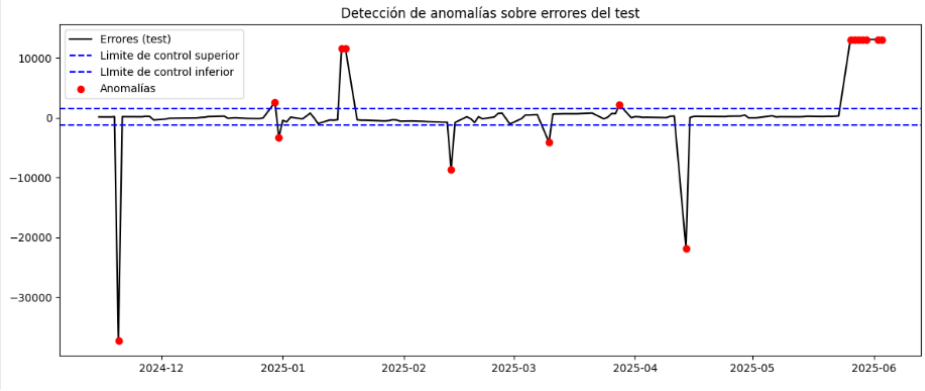


Figura 5.10: Gráfico de control de errores del modelo LightGBM.

El modelo LightGBM detectó un total de 16 observaciones anómalas. La tabla 5.14 presenta la fecha, el valor absoluto del error y la clasificación de cada una según su coincidencia con las anomalías reales o simuladas introducidas en el conjunto de test:

De las 16 anomalías detectadas, 12 corresponden a valores simulados o reales, mientras que 4 son falsos positivos. A partir de esta clasificación, se obtienen las siguientes métricas de evaluación:

- **Precision** = 0,75
- **Recall** = 1,0
- **F1-score** = 0,86

Estos resultados reflejan que el modelo LightGBM, combinado con el enfoque de detección basado en control estadístico, es eficaz para identificar desviaciones significativas en la serie temporal, logrando detectar la totalidad de las anomalías relevantes con un número moderado de falsos positivos.

Fecha	Error absoluto	Clasificación
20/11/2024	37328, 64	Anomalía real
30/12/2024	2611, 21	Falso positivo
31/12/2024	3291, 52	Falso positivo
16/01/2025	11579, 37	Simulada
17/01/2025	11579, 37	Simulada
13/02/2025	8635, 63	Simulada
10/03/2025	4104, 36	Simulada
28/03/2025	2194, 64	Falso positivo
14/04/2025	21915, 36	Simulada
26/05/2025	13113, 64	Simulada
27/05/2025	13113, 64	Simulada
28/05/2025	13113, 64	Simulada
29/05/2025	13113, 64	Simulada
30/05/2025	13113, 64	Falso positivo
02/06/2025	13113, 64	Simulada
03/06/2025	13113, 64	Simulada

Tabla 5.14: Anomalías detectadas por el modelo LightGBM y su correspondencia con las anomalías reales o simuladas.

5.2.5. Modelo Encoder-Decoder

EL último modelo que se aplicará para la detección de anomalías en la serie de Avales es el Encoder-Decoder, ampliamente utilizado en tareas de predicción secuencial. Tal como se detalló en el Capítulo 4, este tipo de arquitectura se compone de dos módulos principales: un *encoder*, que codifica la secuencia de entrada en una representación latente compacta, y un *decoder*, que a partir de dicha representación reconstruye la secuencia original.

Esta estrategia resulta especialmente adecuada para la detección de anomalías en series temporales, ya que permite modelar dependencias temporales complejas sin necesidad de suposiciones explícitas sobre la estructura de la serie. Las observaciones que presentan un error de reconstrucción significativamente superior al esperado pueden ser consideradas como potencialmente anómalas.

La implementación utilizada en este estudio parte del repositorio `encdec-ad-tensorflow` desarrollado por Giammarino (2021), el cual ha sido adaptado para ajustarse específicamente al caso de series temporales univariantes.

Configuración del modelo

El modelo *Encoder-Decoder* fue implementado en `PyTorch`, utilizando una arquitectura basada en redes neuronales LSTM (Long Short-Term Memory). Esta arquitectura fue seleccionada por su capaci-

dad para capturar dependencias de largo plazo en secuencias temporales, lo cual resulta especialmente útil para la predicción y detección de patrones anómalos en series temporales.

Los hiperparámetros del modelo fueron el resultado de una fase previa de optimización mediante validación. A continuación, se detallan los valores utilizados y su interpretación:

- `input_window`: 5 — longitud de la ventana de entrada, es decir, el número de observaciones pasadas utilizadas por el encoder para generar la representación latente. Controla el contexto temporal que el modelo tiene en cuenta.
- `hidden_size`: 64 — dimensión de la capa oculta en las redes LSTM del encoder y del decoder. Cuanto mayor es este valor, mayor es la capacidad del modelo para capturar patrones complejos, pero también aumenta el riesgo de sobreajuste.
- `dropout`: $0,1$ — proporción de unidades ocultas desconectadas aleatoriamente durante el entrenamiento. Actúa como técnica de regularización para prevenir el sobreajuste en redes neuronales.
- `learning_rate`: $0,001$ — tasa de aprendizaje que regula la magnitud de las actualizaciones de los pesos durante el entrenamiento. Valores más bajos conducen a una convergencia más estable pero requieren más épocas.
- `batch_size`: 128 — número de secuencias utilizadas en cada actualización del gradiente. Afecta al tiempo de entrenamiento y a la estabilidad de las actualizaciones.
- `max_epochs`: 1000 — número máximo de iteraciones completas sobre el conjunto de entrenamiento. El modelo puede detenerse antes si se activa la parada temprana.
- `early_stopping_start_epoch`: 1 — época a partir de la cual se comienza a monitorizar la pérdida en validación para activar el mecanismo de *early stopping*.
- `early_stopping_patience`: 5 — número de épocas consecutivas sin mejora en la pérdida de validación que se permiten antes de detener automáticamente el entrenamiento.
- `seed`: 1234 — semilla aleatoria fijada para asegurar la reproducibilidad del entrenamiento y de los resultados del modelo.

El entrenamiento se llevó a cabo sobre la serie de *Avales*, generando ejemplos mediante un esquema de *ventanas deslizantes* que produce pares entrada-salida para el modelo. Es decir, para cada secuencia de longitud L del pasado, el modelo debe predecir o reconstruir la misma secuencia. La función de pérdida utilizada fue el RMSE.

Resultados de predicción

El modelo fue evaluado sobre el conjunto de test utilizando las métricas habituales: MAE, RMSE y MAPE. La Tabla 5.15 resume los resultados obtenidos tanto en entrenamiento como en test:

Conjunto	MAE	RMSE	MAPE (%)
Entrenamiento	$3550,15$	$3963,06$	$2,88\,\%$
Test	$4171,99$	$5198,53$	$10,02\,\%$

Tabla 5.15: Métricas de error del modelo Encoder-Decoder en los conjuntos de entrenamiento y test.

Los resultados obtenidos por el modelo Encoder-Decoder muestran un rendimiento razonable. En el conjunto de entrenamiento, las métricas reflejan un ajuste correcto, con un MAE de $3550,15$, un RMSE de $3963,06$ y un MAPE del $2,88\,\%$, lo que indica una capacidad adecuada del modelo para reconstruir la serie observada durante la fase de aprendizaje.

En el conjunto de test, si bien se observa un incremento esperado en los errores, como es habitual en modelos con cierta capacidad de sobreajuste, las métricas se mantienen dentro de un rango aceptable. En particular, el MAPE asciende al $10,02\,\%$, lo que representa un incremento respecto al entrenamiento, pero sin llegar a valores excesivos que invaliden su utilidad práctica. El aumento en el RMSE (hasta $5198,53$) sugiere la presencia de algunas observaciones con errores de reconstrucción más altos, lo cual es coherente con el objetivo del modelo: identificar comportamientos atípicos en la serie temporal.

En conjunto, estos resultados confirman que el modelo Encoder-Decoder ha logrado aprender las dinámicas principales de la serie, manteniendo un compromiso razonable entre ajuste y capacidad de generalización, lo que lo convierte en un candidato adecuado para la detección de anomalías mediante el análisis del error de reconstrucción.

Detección de anomalías

Al igual que en el caso de los otros dos modelos, se aplica un gráfico de control para la detección de anomalías. En la Figura 5.11 se muestra la evolución de los errores y las anomalías identificadas:

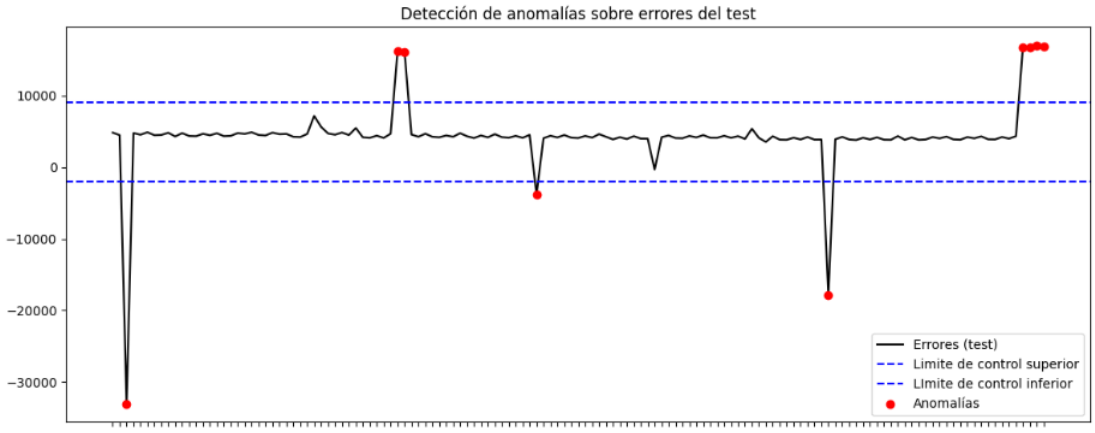


Figura 5.11: Detección de anomalías en el conjunto de test mediante gráfico de control del modelo Encoder-Decoder.

Se identificaron un total de 11 anomalías, detalladas en la Tabla 5.16, junto con el valor del error asociado a cada una de ellas.

De las 11 observaciones señaladas como anómalas por el modelo, todas corresponden efectivamente a valores previamente marcados como anómalos: 1 de ellas fue la anomalía real registrada, y las otras 10 forman parte del conjunto de anomalías simuladas. Esto implica que el modelo no ha generado falsos positivos en la detección.

A partir de esta clasificación se obtienen las siguientes métricas de evaluación:

- **Precision** $= 1,00$
- **Recall** $= 0,917$
- **F1-score** $= 0,957$

Estos resultados reflejan un comportamiento muy sólido por parte del modelo Encoder-Decoder, que logra detectar la práctica totalidad de las anomalías con una precisión perfecta. Aunque no identifica una de las anomalías simuladas, el bajo número de falsos negativos y la ausencia total de falsos positivos refuerzan su idoneidad como herramienta de detección en entornos donde es crucial evitar alarmas injustificadas.

Fecha	Error absoluto	Clasificación
20/11/2024	33075,84	Anomalía real
16/01/2025	16188,19	Simulada
17/01/2025	16145,88	Simulada
13/02/2025	3763,90	Simulada
14/04/2025	17815,01	Simulada
26/05/2025	16759,61	Simulada
27/05/2025	16722,04	Simulada
28/05/2025	17053,60	Simulada
29/05/2025	16819,38	Simulada
02/06/2025	16853,63	Simulada
03/06/2025	16910,02	Simulada

Tabla 5.16: Anomalías detectadas por el modelo Encoder-Decoder sobre el conjunto de test.

El modelo Encoder-Decoder ha demostrado ser eficaz para la detección de anomalías en series temporales univariantes. A pesar de un error más elevado en el conjunto de test comparado con los modelos anteriores, su capacidad para identificar correctamente la mayoría de las anomalías simuladas justifica su utilidad práctica en el entorno analizado.

5.2.6. Comparación de resultados y elección del modelo

Una vez implementados y evaluados los tres modelos propuestos: ARIMA, LightGBM y Encoder-Decoder sobre la serie temporal de *Avales*, se procede a realizar una comparación de sus desempeños desde dos perspectivas complementarias: la capacidad de predicción y la eficacia en la detección de anomalías.

La Tabla 5.17 resume las métricas obtenidas en los conjuntos de entrenamiento y test para cada modelo. Se puede observar que el modelo ARIMA(1,1,0) es el que presenta los mejores resultados en términos de precisión predictiva, con errores significativamente más bajos en todas las métricas evaluadas. En particular, su MAPE en el conjunto de test es de apenas $0,12\,\%$, lo que indica una

capacidad de predicción extremadamente precisa para la serie de *Avales*. El modelo LightGBM, si bien obtiene un rendimiento ligeramente inferior al ARIMA, sigue mostrando un comportamiento aceptable con un MAPE del $0,29\,\%$. Por el contrario, el modelo Encoder-Decoder exhibe errores notablemente superiores, con un MAPE del $10,02\,\%$ en test, lo que refleja mayores dificultades para ajustarse a la dinámica de la serie con la misma precisión que los modelos anteriores. Esta comparación sugiere que, desde el punto de vista puramente predictivo, los modelos tradicionales y basados en árboles superan al enfoque neuronal en este caso concreto.

Modelo	**Entrenamiento**			**Test**		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
ARIMA(1,1,0)	123,22	468,43	0,11	155,79	661,56	0,12
LightGBM	156,04	481,61	0,13	363,15	558,05	0,29
Encoder-Decoder	3550,15	3963,06	2,88	4171,99	5198,53	10,02

Tabla 5.17: Comparación de métricas de error entre los tres modelos aplicados.

El desempeño de los modelos también se evaluó en términos de su capacidad para detectar las anomalías simuladas en el conjunto de test. La tabla 5.18 permite comparar de forma directa el desempeño de los tres modelos en la detección de anomalías simuladas sobre el conjunto de test. Los resultados muestran diferencias significativas en cuanto a precisión, recall y F1-score.

Modelo	**Anomalías detectadas**	**Correcciones correctas**	**Falsos positivos**	**Precision**	**Recall**	**F1-score**
ARIMA	17	12	5	0,71	1	0,83
LightGBM	16	12	4	0,75	1	0,86
Encoder-Decoder	11	11	0	1	0,917	0,957

Tabla 5.18: Comparación de desempeño en la detección de anomalías en el conjunto de test.

El modelo ARIMA presenta una alta capacidad de detección, al identificar las 12 anomalías simuladas, aunque detecta 5 falsos positivos, lo que reduce su precisión. En consecuencia, obtiene una F1-score de 0,83, reflejando un buen rendimiento global, aunque penalizado por su tendencia a sobredetectar eventos anómalos.

Por su parte el modelo LightGBM muestra un comportamiento similar en términos de cobertura de anomalías, pero con un número menor de falsos positivos, lo que mejora su precisión y eleva su F1-score a 0,86. Este modelo se posiciona como una opción equilibrada entre sensibilidad y precisión.

Finalmente el modelo Encoder-Decoder es el que ofrece los mejores resultados en cuanto a precisión, ya que detecta 11 de las 12 anomalías simuladas sin producir ningún falso positivo. Su precisión perfecta y su elevado *recall* se traducen en la mayor F1-score del conjunto, 0,957. Estos resultados ponen de manifiesto su eficacia en la detección de patrones anómalos complejos, minimizando las falsas alarmas.

Discusión y selección del modelo

La elección del modelo más adecuado depende en gran medida del objetivo específico del análisis, ya que cada enfoque presenta fortalezas distintas en términos de predicción y detección de anomalías:

- **Predicción precisa de valores futuros:** El modelo ARIMA destaca como la opción más recomendable cuando el objetivo es anticipar con exactitud la evolución normal de la serie. Sus métricas de error son sustancialmente inferiores al resto de modelos, con un MAPE inferior al $0,13\,\%$ en ambos conjuntos, lo que refleja una excelente capacidad para capturar la dinámica subyacente en contextos lineales y estacionarios (tras diferenciación).
- **Detección robusta de valores atípicos:** El modelo Encoder-Decoder ha demostrado ser el más eficaz en la identificación de anomalías simuladas sin incurrir en falsos positivos, alcanzando una precisión del $100\,\%$ y la mayor F1-score $(0,957)$. Esta arquitectura, basada en redes neuronales recurrentes, permite detectar patrones inusuales complejos que podrían pasar desapercibidos para modelos más rígidos. Es especialmente recomendable en escenarios donde el coste de ignorar una anomalía es elevado y se prioriza la fiabilidad en la señal de alerta.
- **Equilibrio entre predicción y detección:** El modelo LightGBM ofrece un compromiso intermedio. Aunque no alcanza la precisión de ARIMA en la predicción ni la pureza del Encoder-Decoder en la detección, su rendimiento general es competitivo. Su capacidad para integrar variables derivadas (como retardos y estadísticas móviles) y su flexibilidad para capturar relaciones no lineales lo convierten en una herramienta versátil y escalable en entornos con múltiples fuentes de variabilidad.

En el contexto específico de este estudio, centrado en el aseguramiento de la calidad de los datos registrados diariamente en la serie de Avales de la entidad financiera ABANCA, la detección eficaz de registros anómalos adquiere una importancia estratégica. En este entorno, no solo resulta prioritario identificar el mayor número posible de incidencias relevantes, sino también reducir la carga operativa derivada de revisar alertas infundadas, ya que cada verificación implica un consumo de recursos humanos.

En este sentido, si bien el modelo ARIMA proporciona una buena aproximación a la dinámica subyacente de la serie y LightGBM destaca por su capacidad para incorporar múltiples variables, el modelo Encoder-Decoder representa una solución especialmente adecuada. Su alto recall, combinado con una tasa moderada de falsos positivos, ofrece un equilibrio óptimo entre cobertura y eficiencia operativa, alineándose con las prioridades establecidas por la entidad. Por ello, se considera el enfoque más apropiado para desplegar en entornos productivos donde se requiere una supervisión automatizada fiable pero manejable.

Capítulo 6

Conclusiones finales y lineas futuras de investigación

A lo largo de este Trabajo de Fin de Máster se han abordado dos desafíos fundamentales en el ámbito del aseguramiento de la calidad de los datos: el tratamiento de valores nulos y la detección de anomalías en series temporales univariantes. Ambos aspectos se han estudiado en un entorno real, en colaboración con una entidad financiera, lo que ha permitido contrastar las soluciones propuestas en un contexto práctico y de alta exigencia operativa.

En cuanto al tratamiento de valores faltantes, el análisis ha puesto de relieve la utilidad de los métodos clásicos, especialmente en escenarios donde se requiere una elevada trazabilidad y comprensión de las imputaciones realizadas. Sin embargo, también ha quedado patente que, en situaciones más complejas, el uso de modelos probabilísticos como HoloClean aporta ventajas sustanciales. Este enfoque permite realizar imputaciones más precisas, al integrar diversas fuentes de información y restricciones en un único marco coherente. Esta combinación entre métodos tradicionales y técnicas avanzadas ofrece a las organizaciones flexibilidad para adaptarse a distintos niveles de exigencia y tipologías de datos.

En relación con la detección de anomalías en series temporales, se ha diseñado una metodología basada en la predicción de valores esperados y el análisis de los residuos resultantes. Se han aplicado y comparado tres enfoques diferentes: ARIMA, LightGBM y Encoder-Decoder, cada uno con sus propias fortalezas. El modelo ARIMA ha mostrado un excelente comportamiento en términos de ajuste, mientras que LightGBM ha aportado capacidad para incorporar información contextual mediante variables adicionales. El modelo Encoder-Decoder, por su parte, ha resultado especialmente adecuado cuando el objetivo es priorizar la detección de casos anómalos de alto impacto, aspecto especialmente valorado en el entorno empresarial.

En conjunto, este trabajo ha permitido establecer las bases de un sistema automatizado de aseguramiento de la calidad de los datos, combinando robustez metodológica con viabilidad operativa. La metodología seguida, junto con las decisiones adoptadas a lo largo del proyecto, configura un marco reproducible y adaptable a otros conjuntos de datos y dominios de aplicación.

Como continuación natural de esta investigación, se plantean las siguientes líneas de trabajo:

- **Extensión a datos multivariantes:** Generalizar los métodos empleados a series temporales multivariantes permitiría capturar interacciones entre variables y detectar anomalías contextuales que podrían pasar desapercibidas en un análisis univariante.
- **Reentrenamiento adaptativo en producción:** Incorporar estrategias de aprendizaje continuo

o actualización periódica de los modelos facilitaría su adaptación a entornos dinámicos, donde los patrones de comportamiento evolucionan con el tiempo.

- **Integración con alertas automatizadas y visualización operativa:** Conectar los modelos desarrollados con paneles de visualización o sistemas de alerta facilitaría su interpretación y uso práctico por parte de los equipos de negocio o control.
- **Evaluación del impacto económico de las anomalías detectadas:** Estimar los costes derivados de los falsos positivos y falsos negativos permitiría optimizar la intervención sobre los casos más relevantes y maximizar el valor generado por el sistema.

En resumen, este trabajo constituye un punto de partida firme para el diseño e implementación de soluciones orientadas a mejorar la calidad y fiabilidad de los datos. Estas soluciones, correctamente integradas en los procesos organizativos, pueden contribuir de forma significativa a una toma de decisiones más informada y a una gestión más eficiente de los recursos..

Bibliografía

[1] ABANCA. (s.f.). ABANCA Multipaís Particulares y Empresas. https://www.abanca.com/es/cuentas-y-depositos/abanca-multipais/. Accedido 26 de junio de 2025.

[2] Batini C, Cappiello C, Francalanci C, Maurino A (2009) Methodologies for data quality assessment and improvement. ACM Computing Surveys 41(3):Article 16. https://doi.org/10.1145/1541880.1541883.

[3] Batini C, Scannapieca M (2006) Data Quality: Concepts, Methodologies and Techniques. Springer, Berlin. https://doi.org/10.1007/3-540-33173-5

[4] Batista GEAPA, Monard MC (2003) An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5-6):519-533. https://doi.org/10.1080/713827181

[5] Bishop CM (2006).Pattern recognition and machine learning. Springer.

[6] Blázquez-García A, Conde A, Mori U, Lozano JA (2021). A review on outlier/anomaly detection in time series data. ACM Computing Surveys 54(3):1-33. https://doi.org/10.1145/3444690

[7] Breiman L (2001). Random forests. Machine Learning 45(1):5-32. https://doi.org/10.1023/A:1010933404324.

[8] Bontempi G, Ben Taieb S, Le Borgne YA (2012). Machine learning strategies for time series forecasting. En Aufaure MAE, Zimányi(ed) European Business Intelligence. Springer, Berlin, pp 62-77. https://doi.org/10.1007/978-3-642-36318-4_3

[9] Bohannon P, Fan W, Flaster M, Rastogi R (2005) A cost-based model and effective heuristic for repairing constraints by value modification. En Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp 143-154. https://doi.org/10.1145/1066157.1066175

[10] Box GEP, Jenkins GM, Reinsel GC, Ljung GM (2015). Time Series Analysis: Forecasting and Control 5th Edition. Wiley.

[11] Casella G, George EI (1992) Explaining the Gibbs Sampler. The American Statistician 46(3):167-174. https://doi.org/10.1080/00031305.1992.10475878

[12] Chandola V., Banerjee A, Kumar V (2009). Anomaly detection: A survey. ACM Computing Surveys 41(3):1-58. https://doi.org/10.1145/1541880.1541882

[13] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-Decoder approaches. arXiv preprint https://arxiv.org/abs/1409.1259.

[14] Date CJ (2004). An Introduction to Database Systems 8th Edition. Addison-Wesley.

[15] Dickey DA, Fuller WA (1979). Distribution of the estimators for autoregressive time series with a unit root. Journal of the American Statistical Association 74(366):427-431. https://doi.org/10.2307/2286348

[16] Dirección General del Catastro. (s.f.). Referencia catastral. Gobierno de España. https://www.sedecatastro.gob.es/. Accedido 15 de junio de 2025.

[17] Elmasri R, Navathe, SB (2015). Fundamentals of Database Systems 7th Edition. Pearson, Boston.

[18] Enders CK (2010) Applied Missing Data Analysis. The Guilford Press, New York.

[19] Friedman JH (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5):1189-1232. https://doi.org/10.1214/aos/1013203451.

[20] Giammarino F (2021). encdec-ad-tensorflow: LSTM-based encoder-decoder model for anomaly detection in time series [Código fuente]. GitHub. https://github.com/flaviagiammarino/encdec-ad-tensorflow

[21] Goodfellow I, Bengio Y, Courville A (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/.

[22] Han JW, Kamber M, Pei J (2012) Data Mining: Concepts and Techniques 3rd Edition. Morgan Kaufmann Publishers, Waltham.

[23] Hastie T, Tibshirani R, Friedman J (2009). The Elements of Statistical Learning 2nd Edition. Springer.

[24] Heckman JJ (1979) Sample Selection Bias as a Specification Error. Econometrica 47(1):153-161. https://doi.org/10.2307/1912352

[25] Hewamalage H, Bergmeir C, Bandara K (2021) Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting 37(1):388-427. https://doi.org/10.1016/j.ijforecast.2020.06.008.

[26] HoloClean Developers (2019). HoloClean: Holistic Data Repair [Repositorio en GitHub]. https://github.com/HoloClean/holoclean

[27] Hochreiter S, Schmidhuber J (1997). Long short-term memory. Neural Computation 9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[28] Hyndman RJ, Athanasopoulos G (2018). Forecasting: Principles and Practice 2nd Edition. OTexts. https://otexts.com/fpp2/

[29] Kolahi S, Lakshmanan LVS (2009). On approximating optimum repairs for functional dependency violations. ACM International Conference Proceeding Series 361:53-62. https://doi.org/10.1145/1514894.1514901

[30] Kschischang FR, Frey BJ, Loeliger HA (2001) Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory, 47:498-519. http://dx.doi.org/10.1109/18.910572

[31] Levenshtein, VI (1966) Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10:707-710.

[32] Lim B, Zohren S, Roberts S (2021) Time-series forecasting with deep learning: A survey. Philosophical Transactions of the Royal Society 379:20200209. https://doi.org/10.1098/rsta.2020.0209.

[33] Little RJA (1988) A Test of Missing Completely at Random for Multivariate Data with Missing Values. Journal of the American Statistical Association, 83(404):1198-1202. https://doi.org/10.1080/01621459.1988.10478722

[34] Little RJA, Rubin DB (2002) Statistical Analysis with Missing Data 2nd Ed. Wiley Interscience, New York. https://doi.org/10.1002/9781119013563

[35] Ljung GM, Box GEP (1978). On a Measure of Lack of Fit in Time Series Models. Biometrika 65(2):297-303. https://doi.org/10.1093/biomet/65.2.297

[36] Microsoft (2024). LightGBM Documentation. https://lightgbm.readthedocs.io/en/latest/. Accedido 07 de julio de 2025.

[37] Molenberghs G, Kenward M (2007) Missing Data in Clinical Studies. Wiley, England. https://doi.org/10.1002/9780470510445

[38] Montgomery DC (2013). Introduction to statistical quality control 7th Edition. Wiley.

[39] Niako N., Melgarejo JD, Maestre GE, Vatcheva KP (2024). Effects of missing data imputation methods on univariate blood pressure time series data analysis and forecasting with ARIMA and LSTM. BMC Medical Research Methodology 24:320. https://doi.org/10.1186/s12874-024-02448-3

[40] Nielsen MA (2015). Neural Networks and Deep Learning. Determination Press.

[41] Pipino LL, Lee YW, Wang RY (2002) Data quality assessment. Communications of the ACM 45(4):211-218. https://doi.org/10.1145/505248.506010

[42] Rahm E, Do HH (2000) Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin 23(4):3-13

[43] Rekatsinas T, Chu X, Ilyas IF, Ré C (2017) Holoclean: Holistic data repairs with probabilistic inference. Proceedings of the VLDB Endowment 10(11):1190-1201 https://doi.org/10.14778/3137628.3137631

[44] Rubin DB (1976) Inference and missing data. Biometrika 63(3):581-592. https://doi.org/10.1093/biomet/63.3.581

[45] Rubin DB (1987) Multiple Imputation for Nonresponse in Surveys. WJohn Wiley & Sons Inc., New York. http://dx.doi.org/10.1002/9780470316696

[46] Schafer JL (1997) Analysis of Incomplete Multivariate Data. Chapman & Hall/CRC, New York.

[47] Shapiro SS, Wilk MB (1965). An analysis of variance test for normality (complete samples). Biometrika 52:591-611. https://doi.org/10.1093/biomet/52.3-4.591.

[48] Shumway RH, Stoffer DS (2017). Time Series Analysis and Its Applications: With R Examples 4th Edition. Springer.

[49] Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems 27:3104-3112. https://doi.org/10.48550/arXiv.1409.3215

[50] Tatbul N, Lee TJ, Zdonik S, Alam M, Gottschlich J (2018). Precision and recall for time series anomaly detection. Advances in Neural Information Processing Systems 31. https://doi.org/10.48550/arXiv.1803.03639

[51] Zhang C, Yessenov K, Ré C (2017) DeepDive: declarative knowledge base construction. Communications of the ACM 60(5):93-102 https://doi.org/10.1145/3060586