# PROYECTO FIN DE GRADO

**TÍTULO**:

Desarrollo y explicación Post-Hoc de un modelo de mantenimiento predictivo

**AUTOR/A:** Carlos Rubio Hernán

**TITULACIÓN**: Grado en Ingeniería y Sistemas de Datos

**DIRECTOR/A**: Mario Refoyo López

**TUTOR/A**: David Luengo García

**DEPARTAMENTO**: Ingeniería Audiovisual y Comunicaciones

**VºBº TUTOR/A**



**Miembros del Tribunal Calificador:**

**PRESIDENTE/A**: Santiago Higuera

**TUTOR/A**: David Luengo García

**SECRETARIO/A**: David Osés del Campo

**Fecha de lectura: 22 de Julio 2025**

**Calificación:**

**El Secretario/La Secretaria,**

# Agradecimientos

Quiero comenzar expresando mi más sincero agradecimiento a mis padres y mis hermanos, por su ayuda y apoyo incondicional a lo largo de estos años de carrera. Su confianza, paciencia y aliento han sido fundamentales para llegar hasta aquí.

A mis amigos de toda la vida, agradecerles su compañía y apoyo. Vuestra amistad y cariño han sido una fuente de alegría y fuerza todos estos años.

A mi pareja por su apoyo constante y su capacidad para ayudarme a seguir adelante cuando los ánimos bajaban. Tu compañía ha sido clave en este cierre de etapa.

También agradecer a los amigos encontrados durante la carrera, por compartir esfuerzos, risas, noches de estudio y celebraciones. Vuestra compañía ha hecho esta etapa mucho más llevadera y enriquecedora.

Finalmente, quiero agradecer a mis tutores, Mario Refoyo López y David Luengo García, por su guía y dirección en este proyecto. Su experiencia y consejo han sido vitales para la realización de este trabajo.

# Resumen

Este Proyecto de Fin de Grado (PFG), titulado “Desarrollo y explicación Post-Hoc de un modelo de mantenimiento predictivo”, aborda el desarrollo de un sistema de mantenimiento predictivo basado en redes neuronales densas, orientado a estimar de forma probabilística el tiempo restante hasta el fallo (RUL) de motores aeronáuticos simulados.

Para ello, se ha utilizado el conjunto de datos C-MAPSS de la NASA, ampliamente empleado en estudios de mantenimiento predictivo, aplicando un enfoque basado en la predicción de los parámetros de la distribución de Weibull (α y β). El modelo ha sido validado sobre los cuatro subconjuntos (FD001-FD004), obteniendo resultados robustos incluso en condiciones operativas variables, con errores MAE y RMSE situados dentro de unos márgenes razonables (alrededor de 11-24 ciclos).

Además, se ha implementado un análisis “post-hoc” de interpretabilidad mediante la técnica “Integrated Gradients”, que permite descomponer la importancia temporal y sensorial de las predicciones. Esto ha permitido identificar sensores clave con valor explicativo, así como conocer la importancia de los datos de entrada sobre las decisiones finales, añadiendo así transparencia y confianza al sistema predictivo.

Por último, el proyecto acaba con una discusión sobre los resultados y las conclusiones sacadas de estos, así como las limitaciones del modelo, proponiéndose líneas futuras que contemplan mejoras estructurales, validaciones avanzadas y ampliaciones hacía sistemas de aplicación industrial. En conjunto, el proyecto aporta una solución precisa, interpretable y viable como base para sistemas de mantenimiento predictivo en contextos críticos.

# Abstract

This Final Degree Project (PFG) titled “Development and Post-Hoc Explanation of Predictive Maintenance Model”, focuses on the development of a predictive maintenance system based on dense neural networks, aimed at probabilistically estimating the Remaining Useful Life (RUL) of simulated aircraft engines.

To achieve this, the NASA C-MAPSS dataset was employed, which is widely used in predictive maintenance studies. The approach is based on predicting the parameters of the Weibull distribution (α and β). The model was validated across all four subsets (FD001-FD004), obtaining robust results even under variable operating conditions, with MAE and RMSE errors maintained within reasonable bounds (approximately 11-24 cycles).

Furthermore, a post-hoc interpretability analysis was implemented using the Integrated Gradients technique, which decomposes the temporal and sensor-level contributions to the model’s predictions. This allowed the identification of key explanatory sensors and the estimation of input relevance in the final decisions, enhancing the transparency and trustworthiness of the predictive system.

Finally, the project concludes with a discussion of the results and derived conclusions, including model limitations and proposed future work. These future directions include structural improvements, advanced validation, and extensions toward deployment in real-world industrial systems. Overall, the project delivers a precise, interpretable, and viable foundation for predictive maintenance systems in critical environments.

# Índice de Figuras

# Índice de Tablas

# Lista de Acrónimos

**AI**: “Artificial Intelligence” (Inteligencia Artificial).

**ANN**: “Artificial Neural Network” (Red Neuronal Artificial).

**α:** Parámetro de escala de la distribución de Weibull.

**β:** Parámetro de forma de la distribución de Weibull.

**BPTT:** “Backpropagation Through Time” (Retropropagación a través del tiempo).

**CDF:** “Cumulative Distribution Function” (Función de distribución acumulada).

**CMAPSS:** “Commercial Modular Aero-Propulsion System Simulation” (Simulación Modular Comercial del Sistema de Propulsión Aeroespacial).

**CNN:** “Convolutional Neural Network” (Red Neuronal Convolucional).

**CPU:** “Central Processing Unit” (Unidad Central de Procesamiento).

**DL:** “Deep Learning” (Aprendizaje profundo).

**DQN:** “Deep Q-Network” (Red Profunda Q).

**FD001–FD004:** “Fault Dataset 001–004” (Subconjuntos del conjunto de datos C-MAPSS).

**FDP:** Función de Densidad de Probabilidad.

**GMM:** “Gaussian Mixture Model” (Modelo de Mezcla Gaussiana).

**GPU:** “Graphics Processing Unit” (Unidad de Procesamiento Gráfico).

**GRU:** “Gated Recurrent Unit” (Unidad Recurrente con Puertas).

**HF:** “Hazard Function” (Función de riesgo).

**IG:** “Integrated Gradients” (Gradientes Integrados).

**KNN:** “K-Nearest Neighbors” (K Vecinos más Cercanos).

**LIME:** “Local Interpretable Model-agnostic Explanations” (Explicaciones Locales Interpretables Independientes del Modelo).

**LSTM:** “Long Short-Term Memory” (Memoria a Corto y Largo Plazo).

**MAE:** “Mean Absolute Error” (Error Absoluto Medio).

**ML:** “Machine Learning” (Aprendizaje Automático).

**MSE:** “Mean Squared Error” (Error Cuadrático Medio).

**NASA:** “National Aeronautics and Space Administration” (Administración Nacional de Aeronáutica y del Espacio).

**NN:** “Neural Network” (Red Neuronal).

**ODS:** Objetivos de Desarrollo Sostenible.

**PCA:** “Principal Component Analysis” (Análisis de Componentes Principales).

**PDP:** “Partial Dependence Plot” (Gráfico de Dependencia Parcial).

**PDF:** “Probability Density Function” (Función de Densidad de Probabilidad).

**PFG:** Proyecto Fin de Grado.

**Q-LEARNING**: Q-Learning (Aprendizaje basado en valores Q).

**RGPD:** Reglamento General de Protección de Datos.

**RNN:** “Recurrent Neural Network” (Red Neuronal Recurrente).

**RMSE:** “Root Mean Squared Error” (Raíz del Error Cuadrático Medio).

**RSF:** “Random Survival Forest” (Bosque Aleatorio de Supervivencia).

**RUL:** “Remaining Useful Life” (Vida Útil Remanente).

**SARSA**: “State-Action-Reward-State-Action” (Estado-Acción-Recompensa-Estado-Acción).

**SHAP:** “SHapley Additive exPlanations” (Explicaciones Aditivas de Shapley).

**TD-LEARNING:** “Temporal Difference Learning” (Aprendizaje por Diferencias Temporales).

**TTE:** “Time To Event” (Tiempo hasta el Evento).

**WTTE-RNN:** “Weibull Time To Event Recurrent Neural Network” (Red Neuronal Recurrente Weibull para Tiempo hasta el Evento).

**XAI:** “Explainable Artificial Intelligence” (Inteligencia Artificial Explicable).

# Índice de contenidos

# Capítulo 1: Introducción

## 1.1 Introducción

En la actualidad, vivimos en una era tecnológica en continua evolución, donde el aprendizaje automático ("machine learning") ha adquirido un papel fundamental, y mediante el uso de múltiples herramientas y técnicas se ha ido desarrollando a largo de los años. El continuo crecimiento del aprendizaje automático y sus herramientas presenta grandes ventajas para la sociedad, pero también plantea desafíos en cuanto a la comprensión de los métodos empleados y la interpretación de los resultados obtenidos [1].

Este proyecto se centra en el uso de series temporales, cuyo continuo crecimiento en una gran variedad de campos [2], como la salud, la industria y las finanzas, ha llevado al desarrollo de modelos específicos, "time-to-event", que permiten predecir el tiempo hasta la ocurrencia de un evento de interés. Dentro de este ámbito, los **modelos de supervivencia** destacan por su capacidad de proporcionar una estimación de incertidumbre de la predicción. Sin embargo, la interpretación de estos modelos, especialmente en lo que respecta a la explicación de la predicción de incertidumbre o la dispersión asociada a las predicciones, sigue siendo un desafío importante [3]. En el contexto de este proyecto, la Inteligencia Artificial Explicable (XAI) desempeña un papel crucial al proporcionar herramientas para entender adecuadamente y confiar en estos modelos.

En este proyecto se lleva a cabo la implementación de un modelo de supervivencia inspirado en la arquitectura WTTE-RNN [4] aplicado a problemas "time-to-event" en series temporales en el contexto de mantenimiento predictivo. Concretamente, se utiliza una red densa que estima dinámicamente los parámetros de una distribución Weibull (α y β) en cada instancia, permitiendo modelar tanto el tiempo esperado hasta un evento como su incertidumbre. Sobre este modelo, se aplican técnicas de explicabilidad "post-hoc" [2] (métodos usados una vez los modelos ya han sido entrenados) para analizar qué características del historial temporal contribuyen más a la predicción del evento y su dispersión asociada.

Para abordar la explicabilidad del modelo, se ha utilizado "Integrated Gradients" [5], una técnica "post-hoc" ampliamente referenciada en la literatura [2], que permite estimar la contribución de cada entrada a la predicción del modelo. Este enfoque se basa en el cálculo acumulado de gradientes desde una entrada base hasta la observación real, proporcionando una interpretación local de las predicciones en modelos de redes neuronales.

Para interpretar un correcto funcionamiento interno del modelo, se generan visualizaciones basadas en las atribuciones obtenidas mediante "Integrated Gradients" [5]. Estas permiten identificar qué instantes temporales y sensores tienen una mayor influencia en la predicción del tiempo hasta el fallo, permitiendo así ofrecer una visión más comprensible y fiable del comportamiento del modelo. Este análisis aporta una aproximación práctica a la explicabilidad en contextos de mantenimiento predictivo, donde la transparencia del modelo es clave para su adopción.

## 1.2 Objetivos

En el presente proyecto se persigue el siguiente objetivo general:

Evaluar la viabilidad del uso de técnicas de explicabilidad "post-hoc" en modelos de supervivencia aplicados al mantenimiento predictivo, donde se predicen distribuciones de probabilidad en lugar de estimaciones puntuales.

Para lograr dicho objetivo es necesario conseguir los siguientes objetivos:

1) Implementación y posterior evaluación de un modelo de supervivencia inspirado en la arquitectura WTTE-RNN, adaptado a una red densa, aplicado a problemas "time-to-event" en series temporales.
2) Obtención de explicaciones para el modelo implementado mediante técnicas "post-hoc", tanto de las estimaciones de tiempo hasta el evento como de su incertidumbre.
3) Análisis e interpretación de las atribuciones obtenidas para identificar los instantes temporales y variables de entrada más relevantes en las predicciones del modelo, con el fin de mejorar la comprensión del proceso de decisión.

## 1.3 Especificaciones y Restricciones del Diseño

Se parte de las siguientes especificaciones en el proyecto:

1) Tiempo de entrenamiento y evaluación del modelo de supervivencia aceptable.
2) El rendimiento del modelo debe ser adecuado, evitando predicciones erróneas o sesgadas que puedan comprometer la calidad de las explicaciones generadas
3) La implementación del modelo debe ser flexible en cuanto a la longitud y dimensiones de las señales de series temporales con las que se trabaja.
4) El modelo debe obtener buenos resultados de acuerdo con las características más importantes para los algoritmos de explicabilidad (Simplicidad, fidelidad, estabilidad, consistencia) [1].
5) El modelo de supervivencia se debe validar con señales de series temporales reales.

La arquitectura utilizada toma como inspiración el modelo WTTE-RNN ya implementado en GitHub [4], usándolo como punto de partida para desarrollar la red neuronal densa.

## 1.4 Metodología de Trabajo Propuesta

La metodología de trabajo propuesta en el PFG es de carácter iterativo:

1) Estudio y familiarización con el modelo de supervivencia.
2) Implementación del modelo con Python.
3) Simulaciones y test de funcionalidad del modelo para detectar posibles errores y corregirlos (cíclico hasta verificar su funcionamiento).
4) Tras las correcciones aplicar datos reales y obtener las explicaciones proporcionadas por los métodos "post-hoc", específicamente la técnica de "Integrated Gradients".
5) Análisis de los resultados obtenidos y evaluación de su coherencia, tanto a nivel cuantitativo como desde el punto de vista de la interpretabilidad.

Adicionalmente, se hace uso de GitHub para el control de versiones del modelo y de su implementación a lo largo de las diferentes etapas, facilitando de este modo la resolución de errores.

## 1.5 Desglose de Tareas y Cronograma

El desarrollo del proyecto se ha llevado a cabo a lo largo de 1 año y 5 meses con una dedicación parcial, acumulando un total estimado de 330 horas de trabajo efectivo. A continuación, en la Figura 1.1 se detalla el desglose de tareas realizado y su progresión en el tiempo.


Figura 1.1: Desglose de tareas

1) **Tarea 1:** Revisión de los métodos de supervivencia, "machine learning" y explicabilidad.
2) **Tarea 2:** Implementación de los modelos de supervivencia en problemas de series temporales y aplicados a "time-to-event".
3) **Tarea 3:** Simulación y verificación del funcionamiento del modelo sobre los datos.
4) **Tarea 4:** Aplicación de técnicas de explicabilidad "post-hoc", análisis de resultados, interpretación de las atribuciones y extracción de conclusiones.
5) **Tarea 5:** Redacción de la memoria del proyecto y preparación de la presentación.

| Nombre de la Tarea | Duración en horas | Día inicio | Día fin | Duración en dias |
|---|---|---|---|---|
| Tarea 1 | 55 | 01-mar | 29-mar | 28 |
| Tarea 2 | 75 | 30-mar | 22-may | 53 |
| Tarea 3 | 70 | 14-may | 15-abr | 336 |
| Tarea 4 | 50 | 01-feb | 20-jun | 139 |
| Tarea 5 | 80 | 01-mar | 05-jul | 491 |
| Total | 330 | | | 1047 |

Tabla 1.1: Desglose de tareas por horas

## 1.6 Organización de la Memoria

La memoria del proyecto se organiza en los siguientes capítulos.

En este primer capítulo se introduce el contexto general del proyecto, se explican los objetivos, las especificaciones, la metodología, las tareas y la organización de la memoria.

El Capítulo 2 está dedicado al aprendizaje automático, abordando los principales paradigmas, algoritmos y sus aplicaciones más destacadas.

El Capítulo 3 aborda el concepto de Inteligencia Artificial Explicable (XAI), incluyendo las técnicas principales para lograr la interpretabilidad de los modelos complejos y explicando los enfoques de interpretabilidad “ante-hoc” y “post-hoc”

El Capítulo 4 describe los conceptos fundamentales del análisis de supervivencia y sus principales técnicas estadísticas.

El Capítulo 5 presenta los fundamentos del modelo “Weibull Time To Event Recurrent Neural Network” (WTTE-RNN), explicando su estructura, componentes y base teórica.

El Capítulo 6 detalla el desarrollo del modelo implementado, basado en una red neuronal densa para predicción “time-to-event”, así como su entrenamiento, evaluación e integración con técnicas de explicabilidad post-hoc.

El Capítulo 7 recoge los resultados obtenidos durante la experimentación, incluyendo visualizaciones derivadas del análisis de interpretabilidad.

Finalmente, el Capítulo 8 presenta las conclusiones extraídas del trabajo y las posibles líneas de mejora para futuros desarrollos.

Tras las conclusiones, el Capítulo 9 se centra en los presupuestos, seguido del Capítulo 10, donde se trata el impacto del proyecto y su contribución con los ODS (Objetivos de Desarrollo Sostenible) y para finalizar, en el Capítulo 11 se desarrolla un manual de usuario donde se explica el flujo del código y el cuarto muestra gráficos adicionales correspondientes al Capítulo 7.

# Capítulo 2: Aprendizaje Automático

## 2.1 Introducción

En este capítulo se exploran los fundamentos del aprendizaje automático, una rama clave de la inteligencia artificial que permite a las máquinas adquirir conocimientos a partir de datos y realizar predicciones o tomar decisiones sin ser programadas explícitamente para cada tarea. Se trata de una herramienta esencial en múltiples disciplinas debido al aumento del volumen de datos.

A lo largo del capítulo se revisarán los tres paradigmas principales en los que se divide: supervisado, no supervisado y por refuerzo, cada uno diseñado para abordar distintos tipos de problemas y aplicaciones. Además, se examinarán algoritmos comunes dentro de cada categoría, como las redes neuronales, la regresión lineal, los árboles de decisión y el Q-learning.

Por otra parte, se presentarán ejemplos de aplicaciones prácticas para ilustrar el impacto del aprendizaje automático en la vida cotidiana y en la industria, finalizando con una clasificación de los algoritmos según su tipo de aprendizaje, con el objetivo de ofrecer una visión integral sobre cómo elegir el enfoque adecuado para distintos desafíos de análisis de datos y predicción

## 2.2 ¿Qué es el Aprendizaje Automático?

El aprendizaje automático, aprendizaje máquina o "machine learning" se refiere a un conjunto de métodos que permiten a las computadoras mejorar su rendimiento en la realización de tareas específicas a través de la experiencia obtenida a partir de datos. En lugar de ser programadas explícitamente para realizar una tarea, las máquinas utilizan algoritmos para identificar patrones en los datos y hacer predicciones basadas en estos patrones [1].

Su objetivo principal es desarrollar modelos predictivos que puedan generalizarse a partir de ejemplos conocidos a nuevos datos. Esto se logra mediante el entrenamiento de algoritmos usando conjuntos de datos que contienen ejemplos de entradas y salidas esperadas.

Dentro del aprendizaje automático se pueden diferenciar tres paradigmas principales [1]:

- **Aprendizaje Supervisado:** Este tipo de aprendizaje es una técnica clave dentro del campo del "machine learning" y la inteligencia artificial, utilizada para desarrollar modelos predictivos y de clasificación precisos. Se caracteriza por la utilización de conjuntos de datos etiquetados, donde cada ejemplo de entrada está asociado con una salida conocida.

  El objetivo principal es construir un modelo que aprenda a asociar entradas con salidas de forma correcta, permitiendo así generalizar y hacer predicciones precisas sobre datos no vistos anteriormente.

  La metodología usada en este tipo de aprendizaje se fundamenta en la presentación de un conjunto de entrenamiento compuesto por pares de datos de entrada y salida al algoritmo. A medida que el modelo procesa estos datos, ajusta sus parámetros internos para minimizar el error de predicción, haciendo uso de una función de pérdida

para evaluar su entrenamiento. Este proceso iterativo permite al modelo mejorar sus predicciones y su precisión con el tiempo.

Dentro de los problemas de aprendizaje supervisado existen dos categorías principales: la clasificación y la regresión. La clasificación se centra en asignar datos a categorías discretas, mientras que la regresión se utiliza para predecir valores continuos. Se trata de dos métodos esenciales en múltiples aplicaciones prácticas, desde la detección de spam en correos electrónicos hasta la previsión de ventas empresariales.

Se trata de un aprendizaje que ha proporcionado numerosas herramientas eficaces para la toma de decisiones basadas en datos y la automatización de procesos. A pesar de múltiples desafíos, como la necesidad de grandes cantidades de datos etiquetados y la posible propensión a errores humanos, sus beneficios en términos de precisión y aplicabilidad son indispensables en la actualidad.

- **Aprendizaje No Supervisado:** En el aprendizaje no supervisado, se utilizan algoritmos para analizar y agrupar conjuntos de datos sin etiquetas predefinidas. Estos algoritmos deben encontrar estructuras y patrones ocultos en los datos sin necesidad de intervención humana, convirtiéndose en una herramienta ideal para el análisis exploratorio de datos, la segmentación de clientes, la detección de anomalías o la comprensión de datos.

  Dentro del aprendizaje no supervisado pueden destacarse tres tareas principales:

  - **Agrupación de "clusters":** Técnica de minería de datos que organiza los datos no etiquetados en grupos basados en sus similitudes. Existen varios tipos de algoritmos de "clustering" y se clasifican como, "clusters" exclusivos (cada dato pertenece solo a un grupo, sin solapamientos), superpuestos (un mismo dato puede pertenecer a varios grupos con distintos grados de pertenencia), jerárquicos (los datos se agrupan de forma sucesiva en niveles, creando una estructura e árbol) y probabilísticos (cada dato pertenece a los grupos con una determinada probabilidad en lugar de asignaciones fijas).

    La agrupación de "clusters" puede identificar patrones en los datos para agruparlos, ayudando a los científicos e ingenieros de datos identificando diferencias entre los datos que los humanos hayan podido pasar por alto.

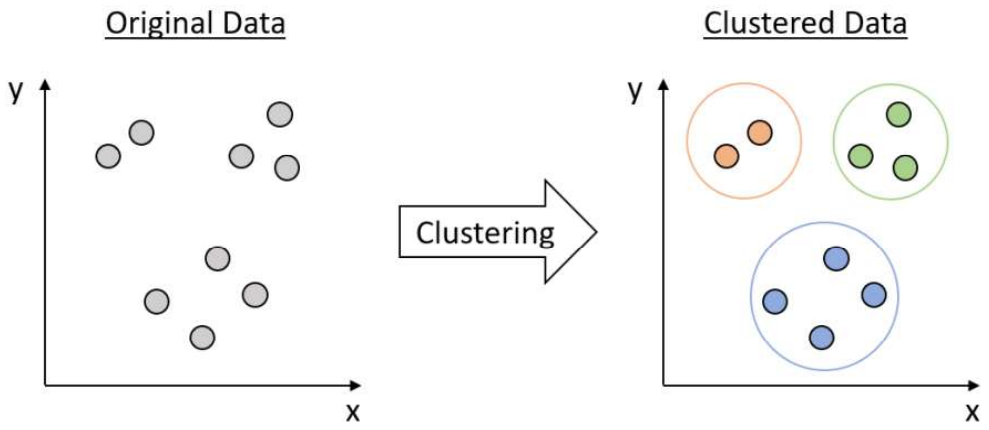


Figura 2.1: Ejemplo de "clustering" **[6]**.

En la Figura 2.1. podemos observar un ejemplo sencillo de “clustering” donde se cuenta con un conjunto de 9 datos originales que se clasifican y agrupan en tres grupos por proximidad, reflejando de este modo un funcionamiento básico de la técnica previamente explicada.

- **Reglas de Asociación:** Técnicas basadas en reglas para detectar relaciones entre variables dentro de un conjunto de datos. Son muy utilizadas en el análisis de cestas de la compra, como en los motores de recomendación de Amazon y Spotify [7], ayudando a entender los hábitos de compra y consumo.

- **Reducción de dimensionalidad:** Conjunto de técnica que reducen el número de características en un conjunto de datos, facilitando el manejo y visualización, además de ayudar a evitar el sobreajuste.

  Algunos métodos comunes incluyen el Análisis de componentes principales (PCA), para conseguir explicar la mayor variabilidad posible de los datos, la Descomposición de Valores Singulares (SVD), para reducir el ruido y la redundancia, y los Codificadores Automáticos, que comprimen y reconstruyen los datos conservando las características más importantes.

  Algunas aplicaciones destacadas del aprendizaje no supervisado son [1]: segmentación de clientes, detección de anomalías, motores de recomendación, análisis de imágenes o exploración de datos, entre otras.

- **Aprendizaje por Refuerzo:** Se trata de un subcampo del aprendizaje automático que se utiliza para entrenar a un agente de “software” para que aprenda a tomar decisiones mediante la interacción con un entorno. El agente recibe recompensas basadas en las acciones que realiza dentro del entorno.

  Como ya se ha comentado, el aprendizaje automático se divide en dos ramas principales, supervisado y no supervisado, donde la principal diferencia es el etiquetado de los datos de entrada que usan.

  El aprendizaje por refuerzo es diferente a estos enfoques, pero se inclina más hacia el extremo supervisado del espectro. Este, se basa en procesos de decisión de Márkov

en los que un usuario cuenta con una serie de estados y puede realizar un conjunto de acciones dentro de cada estado, de modo que, la asociación de estados y acciones se aprende a través de una recompensa o castigo acumulativo por sus acciones.

La asociación tiene lugar en línea, a través de un equilibrio entre la exploración (probar nuevas acciones para un estado dado) y la explotación (usar el conocimiento existente de la asociación de estados/acciones). El resultado deseado es una política óptima de estado a acción que maximiza las recompensas acumuladas a lo largo del tiempo.

El aprendizaje por refuerzo tiene múltiples aplicaciones en la vida real como pueden ser la robótica, los juegos o videojuegos, la optimización de recursos, la gestión de inventarios, la publicidad en línea, entre otras.

Dentro del aprendizaje por refuerzo algunas técnicas destacables son Q-learning, State-Action-Reward-State-Action (SARSA), Temporal Difference Learning (TD-Learning) o Deep Q-Networks (DQN).

En la Figura 2.2 puede observarse un ejemplo de aprendizaje por refuerzo donde el entorno es el juego en sí. El estado es la situación actual del juego, y existe un agente que toma las decisiones. Cuando el agente tome una decisión, es decir, la acción de elegir una casilla, dependiendo de la decisión tomada, recibirá una recompensa de victoria o una penalización como derrota.

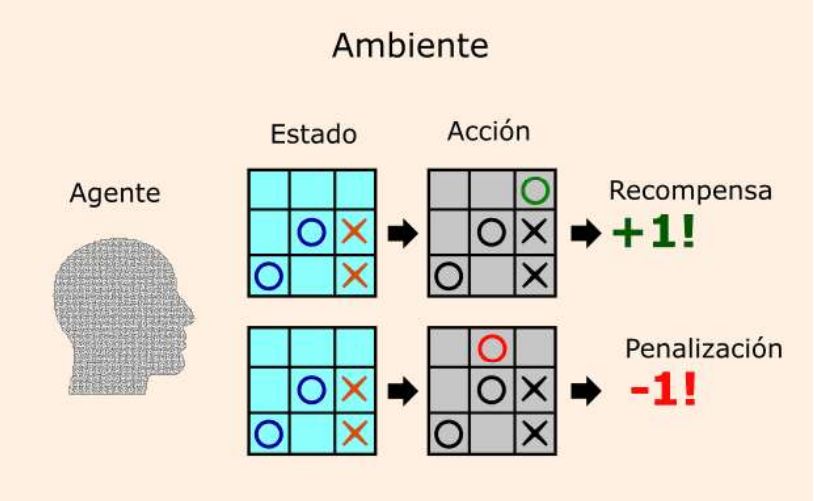


Figura 2.2: Ejemplo de Aprendizaje Por Refuerzo [8]

## 2.3 Algoritmos habituales de Aprendizaje Automático

Existe una amplia variedad de algoritmos de aprendizaje automático (“machine learning”). A continuación, se revisan brevemente algunos de los más populares, divididos en función del tipo de aprendizaje usado [9].

**Aprendizaje Supervisado:**

Este tipo de aprendizaje se basa en entrenar modelos con datos etiquetados para realizar predicciones. Entre sus ejemplos de algoritmos podemos encontrar los siguientes [9]:

- **Regresión Lineal:** Se utiliza para predecir valores continuos, identificando la relación entre una variable dependiente y una o más variables independientes. La regresión lineal simple involucra una variable independiente, mientras que la regresión lineal múltiple incluye varias.

- **Regresión Logística:** Se aplica cuando la variable dependiente es categórica, como en problemas de clasificación binaria (respuestas "sí/no" a las preguntas, por ejemplo). Esta regresión calcula la probabilidad de una clase particular y es útil en tareas de detección de "spam" o la predicción de eventos binarios.

- **K Vecino Más Cercano (KNN):** Es un método de clasificación basado en la proximidad de los puntos de datos. Clasifica un punto nuevo en función de los k puntos de datos más cercanos en el conjunto de entrenamiento, generalmente utilizando la distancia Euclídea.

  KNN es fácil de entender y aplicar, aunque puede volverse computacionalmente costoso a medida que el tamaño del conjunto de datos crece.

- **Árboles de decisión:** Son algoritmos de aprendizaje supervisado no paramétricos que pueden utilizarse tanto para predecir valores numéricos (regresión) como para clasificar datos en categorías. Los árboles de decisión utilizan una secuencia ramificada de decisiones vinculadas que pueden representarse con un diagrama de árbol. En próximos capítulos se describirán más detalles relacionados con estos.

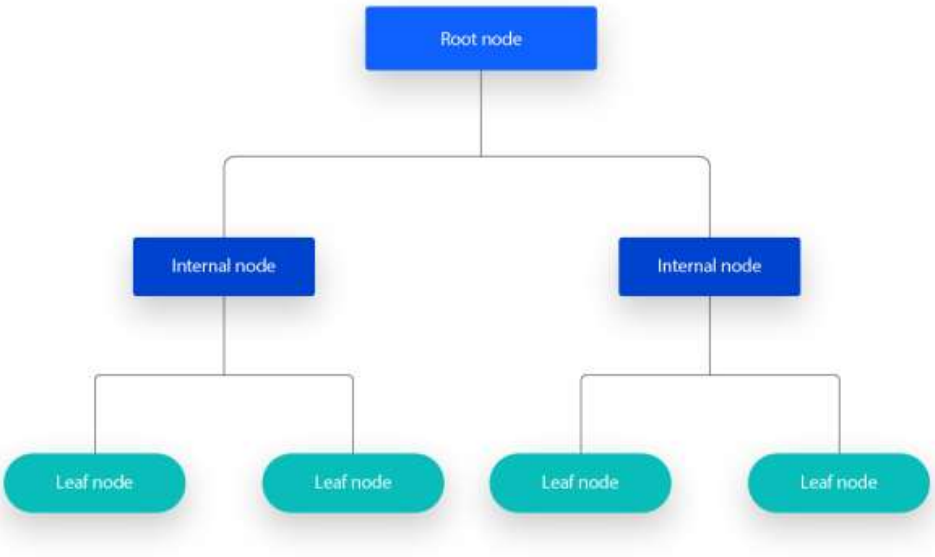


Figura 2.3: Ejemplo de árbol de decisión **[10]**.

En la Figura 2.3 puede observarse como el árbol comienza con un nodo raíz, sin ramas entrantes. Las ramas salientes del nodo raíz alimentan los nodos internos o nodos de decisión. Tanto los nodos internos como los nodos hoja o terminales, se forman mediante evaluaciones basadas en las características disponibles, generando subconjuntos homogéneos que representan resultados específicos dentro del conjunto de datos. Cada camino desde la raíz hasta una hoja representa una regla de decisión basada en los valores de las características.

Los árboles emplean la estrategia "divide y vencerás", identificando los puntos de división óptimos dentro del árbol y repitiendo el proceso hasta lograr clasificar los registros como clases específicas. Además, debe evitarse el sobreajuste, ya que puede llevar a la fragmentación de datos. En este sentido, se usa la poda para reducir la complejidad del árbol y se evalúa el modelo mediante validación cruzada.

Algunas ventajas de los árboles de decisión son su fácil interpretación, la poca preparación de datos necesaria o su flexibilidad. Entre sus desventajas se encuentran el sobreajuste, los estimadores de alta varianza o el posible coste de entrenamiento [10].

- **Bosques aleatorios ("Random Forest"):** en un bosque aleatorio, el algoritmo de machine learning predice un valor o categoría combinando los resultados de una serie de árboles de decisión.

  Se trata de un conjunto de árboles de decisión que operan en paralelo para mejorar la precisión de la clasificación o regresión. Cada árbol de decisión en el bosque se entrena con un subconjunto diferente de datos y características, lo que reduce la varianza y el riesgo de sobreajuste.

  Se trata de un algoritmo altamente versátil y se utiliza en una amplia gama de aplicaciones debido a su robustez y capacidad para manejar grandes conjuntos de datos [9].

- **Redes Neuronales:**

Son una clase de modelos muy utilizados en el aprendizaje supervisado, especialmente en el "Deep learning" o aprendizaje profundo. Estas redes simulan el funcionamiento del cerebro humano, con un enorme número de nodos de procesamiento (neuronas) interconectados, tal como se muestra en la Figura 2.4.

Cada nodo procesa una entrada, aplica una ponderación y, se supera un umbral determinado, activa el nodo siguiente. Es un proceso que permite que las redes neuronales aprendan representaciones complejas de los datos a través del ajuste iterativo de sus pesos mediante algoritmos como el descenso de gradiente.

Gracias al proceso explicado, las redes neuronales se caracterizan por ser buenas reconociendo patrones, y por ello destacan en aplicaciones como la traducción de lenguaje natural, el reconocimiento de imágenes y voz o la creación de imágenes.

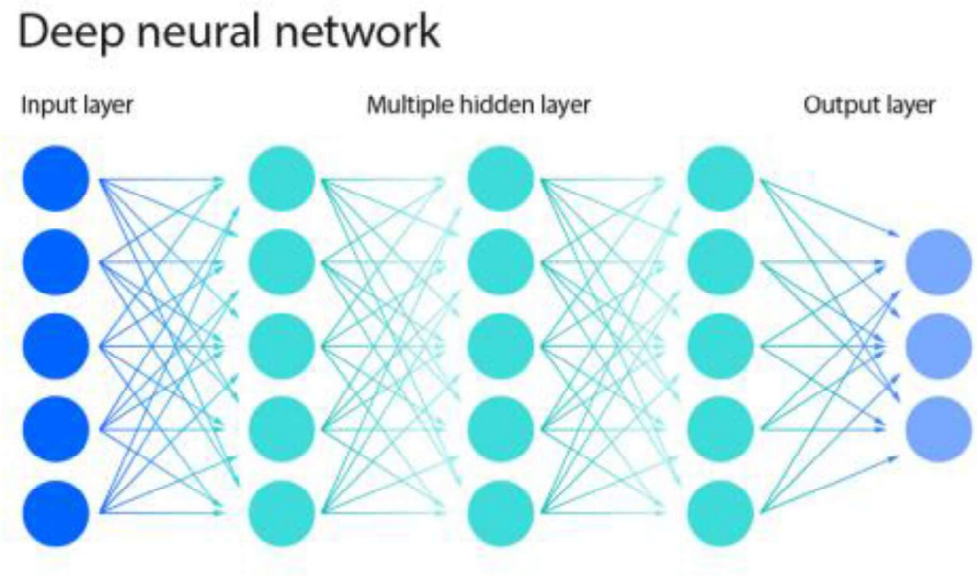


Figura 2.4: Ejemplo de red neuronal (entrada - capas ocultas - salidas) **[9]**.

Dentro de la categoría de Redes Neuronales existen varias arquitecturas diseñadas para diferentes tipos de datos, tales como Redes Neuronales Convolucionales (CNN) y las Redes Neuronales Recurrentes (RNN).

Las CNN están optimizadas para el análisis de datos visuales y son eficaces en la detección de patrones espaciales, como bordes y texturas en imágenes [11]. Por otro lado, las RNN son particularmente útiles en el procesamiento de datos secuenciales o temporales, como texto o series temporales, y se detallarán más adelante en el proyecto debido a su papel fundamental en el modelo WTTE-RNN.

A continuación, se detallan las capas especializadas que componen las CNN, así como algunas de sus aplicaciones principales y sus tipos [12].

1. **Capa Convolucional:** componente principal de una CNN. Es la encargada de aplicar filtros a la entrada para detectar características locales como bordes, texturas y patrones. Como resultado de esta operación se obtiene un mapa de características que resalta las características detectadas en la imagen.

   Esta primera capa realiza convoluciones sobre la entrada, aplicando un filtro que se desplaza sobre la imagen para detectar características locales y almacenar los resultados en un mapa de características.

   Tres hiperparámetros clave en este proceso son: número de filtros (afecta a la profundidad de salida), “stride” (distancia o número de píxeles que el filtro se mueve sobre la matriz de entrada) y “padding” (para ajustarse a la imagen pone a cero todos los elementos que quedan fuera de la matriz de entrada).

2. **Capa de “Pooling”:** es también conocida como reducción de dimensionalidad. Esta capa reduce el tamaño de los mapas de características (reduciendo la dimensionalidad) mediante operaciones de selección de valores máximos (“max-pooling”) o promedios (“average-pooling”), lo que ayuda a reducir el coste computacional y a controlar el sobreajuste.

3. **Capa Totalmente conectada:** Es similar a las capas en las redes neuronales tradicionales, cada nodo en esta capa está conectado a todos los nodos de la capa anterior. Estas capas se encargan de la clasificación final basada en las características aprendidas por las capas convolucionales y de "pooling".

Algunas aplicaciones destacables de las Redes Neuronales Convolucionales son el reconocimiento de imágenes y objetos, visión por computadora, la asistencia médica y los automóviles autónomos, entre otras.

Además, existen varios tipos de Redes Neuronales Convolucionales, entre las que se pueden destacar las siguientes [12]:

- **LeNet-5:** Una de las primeras CNN que funcionó con éxito en el reconocimiento de dígitos escritos a mano.
- **AlexNet:** Ganadora del concurso ImageNet en 2012, ayudando a popularizar el uso de CNN en la clasificación de imágenes
- **VGGNet:** Conocida por su simplicidad y profundidad, utilizando muchos filtros pequeños (3x3)
- **ResNet:** Introdujo las conexiones residuales para facilitar el entrenamiento de redes más profundas

La Figura 2.5 muestra la arquitectura de una red neuronal convolucional típica, en la que una imagen de entrada atraviesa múltiples capas de convolución, "max-pooling", capas totalmente conectadas y una capa final "softmax". A medida que avanza por la red, se reduce la resolución espacial y se incrementa la abstracción de las características, permitiendo la clasificación del contenido visual en diferentes categorías.

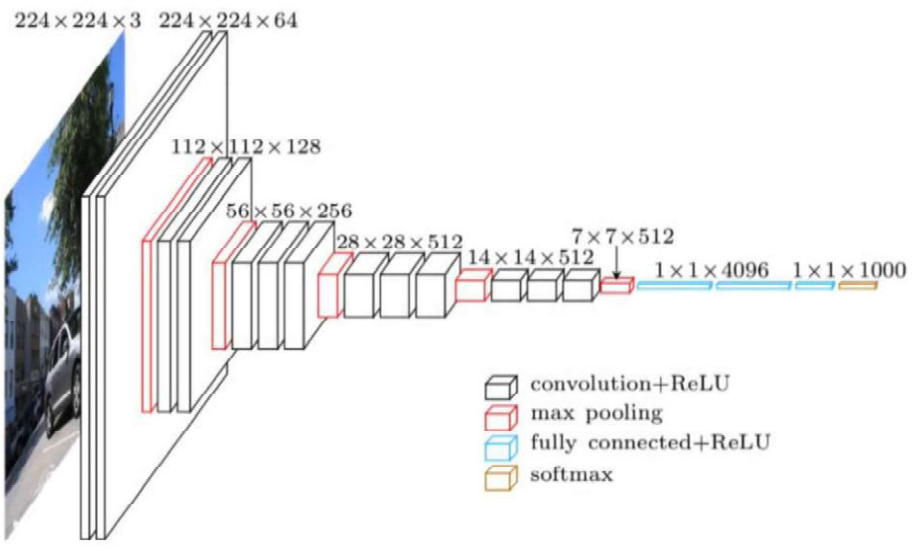


Figura 2.5: Ejemplo de la estructura de una red convolucional **[12]**.

**Aprendizaje No Supervisado:**

El aprendizaje no supervisado busca patrones en datos no etiquetados. Algunos ejemplos destacados son [13]:

- **K-Means "clustering":** La agrupación en "clusters" mediante K-medias es un método de aprendizaje no supervisado común, basado en la agrupación en "clusters" exclusiva en el que los puntos de datos se distribuyen en K grupos, donde K representa el número de "clusters" en función de la distancia desde el centroide de cada grupo. Su funcionamiento se basa en que los puntos de datos más próximos a un determinado centroide se agrupan en la misma categoría.

  Se debe tener en cuenta que un valor K más grande será indicativo de agrupaciones más pequeñas con más granularidad, mientras que un valor K más pequeño tendrá agrupaciones más grandes y menor granularidad.

  Las agrupaciones en "clusters" de k-medias son comunes en la segmentación de mercados, la agrupación de documentos en "clusters", la segmentación de imágenes y la compresión de imágenes [13].

  La Figura 2.6 muestra el funcionamiento básico del algoritmo de agrupamiento "K-Means". A la izquierda se observa el conjunto de datos original, sin ninguna estructura aparente. Tras aplicar el algoritmo "K-Means", el espacio se divide en tres "clusters" claramente diferenciados. Es decir, se observa como identifica patrones en los datos mediante la minimización de distancias "intra-cluster", asignando cada observación al centroide más cercano.

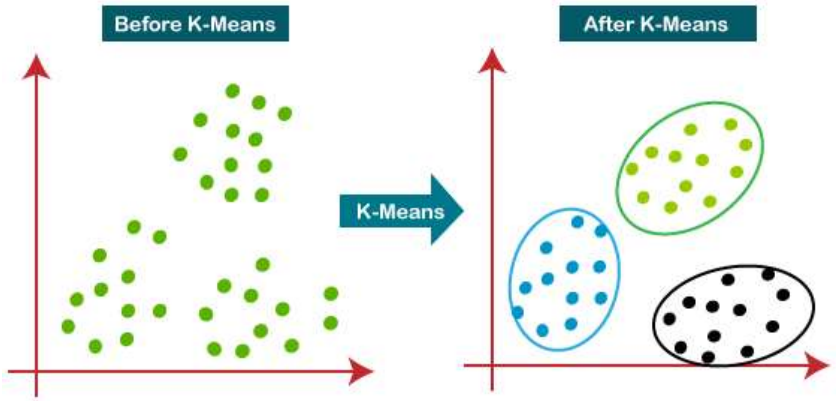


Figura 2.6: Ejemplo de agrupación "K-Means" **[14]**.

- **Análisis de componentes principales (PCA):** Se trata de un tipo de algoritmo de reducción de dimensionalidad que se utiliza para eliminar redundancias y comprimir conjuntos de datos a través de la extracción de características.

  Este método utiliza una transformación lineal para crear una nueva representación de datos, dando como resultado un conjunto de "componentes principales". El primer componente es la dirección que maximiza la varianza del conjunto de datos. Mientras que el segundo componente principal también encuentra la varianza máxima en los datos, pero no tiene ninguna correlación con el primer componente principal, lo que genera una dirección que es perpendicular u ortogonal al primer componente.
  Se trata de un proceso que se repite en función del número de dimensiones, donde el siguiente componente principal es a la dirección ortogonal a los componentes anteriores con mayor varianza.

- **Modelo de Mezcla Gaussiana (GMM):** Es uno de los métodos probabilísticos de agrupación en "clusters" más utilizados. Estos modelos se clasifican como modelos mixtos, lo que significa que están formados por un número sin especificar de funciones de distribución de probabilidad.

  Los GMM se aprovechan principalmente para determinar a qué distribución de probabilidad Gaussiana pertenece un punto de datos determinado. Conociendo la media o la varianza, podemos determinar a qué distribución pertenece un punto de datos determinado. Sin embargo, en los GMM, estas variables no son conocidas, asumiendo así que existe una variable latente (oculta) para agrupar los puntos de datos en clústeres.

  No se requiere utilizar el algoritmo de expectativa-maximización (EM) [13], pero está extendido su uso para estimar las probabilidades de asignación para un punto de datos determinado a un "cluster" de datos.

**Aprendizaje Por Refuerzo:**

El aprendizaje por refuerzo enseña a un agente a tomar decisiones para maximizar recompensas. Algunos de los algoritmos principales dentro de este campo son [9] [15]:

- **Q-Learning:** Es un algoritmo de aprendizaje por refuerzo que se utiliza para aprender la función de valor Q, que estima el valor de tomar una determinada acción en un estado específico. En Q-learning, se asigna un valor Q a cada par estado-acción, que representa la utilidad esperada de tomar esa acción en ese estado y sigue la ecuación de Bellman para actualizar los valores Q iterativamente basados en las recompensas recibidas.

- **State-Action-Reward-State-Action (SARSA):** Es similar a Q-learning ya que es un algoritmo de aprendizaje por refuerzo que también estima la función de valor Q, pero en este caso actualiza los valores de Q para la acción elegida por el agente en lugar de la acción óptima. Resulta útil en entornos donde las políticas cambian rápidamente y se requiere una adaptación más rápida del agente.

- **Temporal Difference Learning (TD-Learning):** Es un método de predicción dentro del aprendizaje por refuerzo basado en la idea de que las predicciones pueden aprenderse a partir de observaciones en un entorno. En TD-Learning se pueden actualizar los valores de Q para los pares estado -acción previos en lugar de únicamente poder actualizar el actual, pudiendo ser útil para el aprendizaje a largo plazo.

- **Deep Q-Networks (DQN):** Se combina el aprendizaje profundo con Q-learning para manejar entornos de alta dimensionalidad, como los videojuegos, por ejemplo. Se utiliza una red neuronal profunda para aproximar la función de valor Q y poder aprender directamente de la estrada cruda, como píxeles de imágenes.

## 2.4 Aplicaciones del Aprendizaje Automático

El aprendizaje automático tiene una amplia gama de aplicaciones en diversos campos. Algunos ejemplos incluyen [9]:

- **Predicción del valor de propiedades inmobiliarias:** Utilizando datos históricos de ventas de casas, las máquinas pueden aprender a predecir el valor de nuevas propiedades basándose en características como tamaño, ubicación y tipo de construcción.

- **Detección de fraude en transacciones financieras:** Algoritmos de aprendizaje automático pueden analizar patrones en datos de transacciones para identificar comportamientos sospechosos y potencialmente fraudulentos [9].

- **Reconocimiento de imágenes:** Sistemas de aprendizaje automático pueden ser entrenados para identificar objetos, personas y escenas en imágenes, aplicándose en áreas como la conducción autónoma y la seguridad.

- **Motores de recomendación**: Haciendo uso de datos históricos de consumo, los algoritmos de IA pueden ayudar a descubrir tendencias que pueden utilizarse para desarrollar estrategias de venta cruzada más eficaces. Los motores de recomendación son usados por minoristas en línea para hacer recomendaciones de productos relevantes a los clientes durante el proceso de pago [9].

## 2.5 Ventajas e inconvenientes

Ventajas:

- **Velocidad y eficiencia:** Una vez entrenado, un modelo de aprendizaje automático puede realizar predicciones rápidamente y a gran escala.

- **Consistencia y reproducibilidad**: Los modelos pueden ofrecer resultados consistentes y ser replicados fácilmente en diferentes entornos.

- **Capacidad de manejar grandes volúmenes de datos:** Los algoritmos de aprendizaje automático pueden procesar y extraer información útil de grandes conjuntos de datos.

Inconvenientes:

- **Complejidad y falta de interpretabilidad:** Muchos modelos de aprendizaje automático, especialmente los más avanzados como las redes neuronales profundas, son considerados "cajas negras" debido a la dificultad de entender cómo toman decisiones.

- **Dependencia de datos de alta calidad**: El rendimiento de los modelos de aprendizaje automático depende en gran medida de la calidad y cantidad de los datos disponibles para el entrenamiento.

Por tanto, se concluye que el aprendizaje automático es una herramienta poderosa para la predicción y el análisis de datos en diversas aplicaciones. Sin embargo, su implementación y uso efectivo requieren una comprensión sólida de sus principios y una consideración cuidadosa de sus limitaciones.

# Capítulo 3: Inteligencia Artificial Explicable (XAI)

## 3.1 Introducción

En este capítulo se introduce el concepto de Inteligencia Artificial Explicable (XAI, de acuerdo con sus siglas en inglés), una disciplina emergente en el ámbito de la inteligencia artificial que busca hacer que los modelos de IA sean comprensibles para los humanos. Durante los últimos años, los modelos de aprendizaje automático se han vuelto cada vez más complejos, lo cual dificulta la interpretación de sus decisiones y predicciones. Esta falta de interpretabilidad representa un desafío importante, especialmente en áreas sensibles como la medicina o las finanzas, donde entender los procesos internos del modelo es fundamental para la confianza y la toma de decisiones informadas.

Es importante señalar que la inteligencia artificial (IA) no se limita a una única metodología, sino que abarca un amplio espectro de enfoques, desde métodos clásicos como el "K-Nearest Neighbors" (KNN) y los árboles de decisión, hasta enfoques más avanzados como las redes neuronales profundas ("Deep Learning").

Mientras que los métodos tradicionales suelen ser más sencillos de interpretar, su capacidad para abordar tareas complejas o datos de alta dimensionalidad es limitada. Por el contrario, los modelos avanzados como las redes neuronales profundas logran una mayor precisión al capturar relaciones complejas en los datos, pero su complejidad dificulta la interpretación de sus decisiones.

Este contraste entre precisión y explicabilidad subraya la importancia de técnicas de interpretabilidad que permitan comprender tanto los métodos tradicionales como los modernos. Además, la capacidad de explicar el algoritmo puede ser esencial o incluso un requerimiento regulatorio, como en el caso del Reglamento General de Protección de Datos (RGPD) [16], que establece el "derecho a la explicación". Estas exigencias normativas han impulsado el desarrollo de la Inteligencia Artificial Explicable (XAI), cuyo objetivo es garantizar que los sistemas de IA sean comprensibles para el ser humano, especialmente en contextos donde las decisiones de los modelos tienen un impacto significativo.

Por último, el marco regulador europeo también se está adaptando al desarrollo de la IA. El "Artificial Intelligence Act" (AI Act), propuesto por el Parlamento Europeo en 2021 [17], establece requisitos específicos relacionados con la transparencia, la seguridad, la precisión y el desarrollo ético de los sistemas de IA, además de introducir un sistema de gobernanza y supervisión para garantizar su cumplimiento.

A lo largo del capítulo, se explora el contexto y la importancia de la explicabilidad en IA, el papel de conceptos clave como la transparencia y la interpretabilidad, y se describen técnicas específicas de interpretabilidad, tanto locales como globales, que permiten analizar modelos complejos.

## 3.2 ¿Qué es XAI?

Cuando hablamos de Inteligencia Artificial Explicable (XAI, del término en inglés "eXplainable Artificial Intelligence"), nos referimos a la explicabilidad de los modelos utilizados y los resultados obtenidos en inteligencia artificial. Por tanto, cuanto más explicable sea un modelo,

más profundo será el entendimiento que los humanos logran y con mayor facilidad podrán explicarse sus resultados y como se llega hasta estos.

La inteligencia artificial explicable (XAI) es un conjunto de procesos y métodos que permiten a los humanos comprender y confiar en los resultados obtenidos a partir de algoritmos de aprendizaje automático o "machine learning" (ML). Su uso general es describir un modelo de IA, su impacto y sus posibles sesgos, ayudando a la hora de caracterizar la precisión, la equidad, la transparencia y los resultados del modelo en la toma de decisiones [18].

En la Figura 3.1 podemos ver la evolución del índice de popularidad (Google Trends) del término "explainable ai" desde el año 2004 hasta la actualidad. Como podemos apreciar, se ha producido una fuerte y continuada subida desde el año 2017 hasta la actualidad.

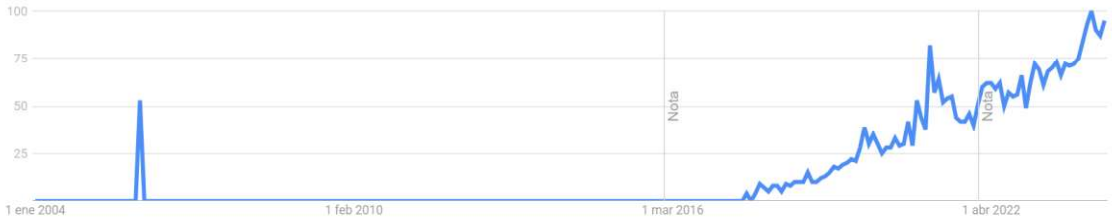


Figura 3.1: Índice de popularidad del término "explainable ai" desde 2004 hasta la actualidad. Gráfico obtenido de Google Trends **[19]**.

**Transparencia, Interpretabilidad y Explicabilidad en XAI**

Dentro de la disciplina XAI es importante tener en cuenta tres conceptos fundamentales [20]:

- **Transparencia:** Indica si se pueden describir y justificar los procesos que calculan los parámetros del modelo y producen los resultados.

- **Interpretabilidad**: Describe la posibilidad de entender el modelo y presentar cómo toma las decisiones de una manera comprensible para los humanos.

  Se trata de un concepto ligado a la capacidad para explicar a un humano los resultados de un modelo (su relación causa-efecto).

  Una de las definiciones más populares de interpretabilidad es la de Doshi-Velez y Kim, quienes la definen como "la capacidad de explicar o presentar en términos comprensibles para un ser humano". Por tanto, se trata de un concepto mayormente conectado con la intuición detrás de las salidas de un modelo [21].

- **Explicabilidad:** Alude a la capacidad de descifrar por qué una determinada observación ha recibido un valor concreto.

  En este caso, se asocia con la lógica interna y los mecanismos que están dentro de un sistema de aprendizaje automático. Cuanto más explicable es un modelo, más profundo es el entendimiento que los humanos logran en cuanto a los procedimientos internos que tienen lugar mientras el modelo se está entrenando o tomando decisiones [20].

Se trata de tres términos muy ligados y con frecuencia empleados de manera intercambiable, ante la falta de consenso sobre sus definiciones precisas.

A la hora de seguir estos tres principios se usan principalmente dos estrategias, o bien desarrollar algoritmos ya interpretables y explicables por su propia naturaleza (regresiones lineales, algunas redes neuronales profundas, etc.), o bien hacer uso de técnicas de interpretabilidad para cumplir con los principios.

Por tanto, la XAI intenta por un lado explicar el comportamiento de determinados modelos opacos ("black box"), y por otro lado diseñar algoritmos inherentemente interpretables ("White box"). De este modo, se contribuye a generar confianza en la toma de decisiones basadas en modelos de IA y se favorece a la mejora del rendimiento y la robustez de los modelos de IA (p. ej., mediante identificación y eliminación de sesgos, comprensión de la información relevante para obtener ciertos resultados o anticiparse a errores).

La XAI se basa en tres aspectos principales de interpretabilidad: la explicación del diseño del modelo, la explicación de los resultados del modelo, así como otros aspectos como la detección de sesgos y el seguimiento periódico del modelo. Además, se debe contar con un factor humano integrado para supervisar el correcto funcionamiento [20].

## 3.3 Técnicas de interpretabilidad

El uso extendido de técnicas de IA en múltiples ámbitos ofrece como resultado un mayor poder predictivo, con la consecuencia de una mayor complejidad en los modelos.

Los diferentes puntos de vista al analizar el panorama emergente de los métodos de interpretabilidad permiten diferenciar varios pilares principales dentro de la XAI, permitiendo así la distinción y división de los métodos.

A la hora de abordar los problemas de interpretabilidad, la diferenciación ya comentada da lugar a uno de los pilares más relevante dentro de la XAI: las técnicas "ante-hoc" y "post-hoc", también conocidas como técnicas de caja negra o "black box" y técnicas de caja blanca o "White box", junto a estrategias complementarias como pueden ser el análisis de datos para identificar sesgos.

Por otra parte, otro de los pilares principales hace referencia al alcance de las explicaciones ofrecidas, existiendo una clara diferenciación entre las explicaciones locales, si el método proporciona una explicación solo para una instancia específica, o explicaciones globales, si el método explica todo el modelo [20].

Esta diferenciación entre interpretabilidad local y global ha favorecido la aparición de técnicas cada vez más sofisticadas (p. ej., PDP, LIME o SHAP) para cubrir la necesidad de explicabilidad e interpretabilidad de modelos.

Además, basado en la arquitectura interna del modelo, si su aplicación está restringida únicamente a una familia específica de algoritmos, entonces el método será clasificado como específico del modelo. En contraste, los métodos que pueden aplicarse en cualquier algoritmo posible se llaman agnósticos del modelo [20].

Por último, otro factor crucial a tener en cuenta es el tipo de datos en los que se puede aplicar el método. Los tipos de datos comúnmente son tabulares e imágenes, pero existen métodos para otros tipos de datos como los datos de texto.

La Figura 3.2 muestra un mapa conceptual que organiza los criterios para clasificar los métodos de interpretabilidad en inteligencia artificial. Se identifican cuatro ejes principales.

1. Finalidad de la interpretabilidad: abarca desde la creación de modelos intrinsecamente interpretables (“White box”) hasta las técnicas “post-hoc” aplicadas a modelos complejos (“black-box”), pasando por objetivos como mejorar la equidad o analizar la sensibilidad de las predicciones.
2. Nivel de explicabilidad: se distinguen los métodos locales (explican predicciones individuales) y globales (describen el comportamiento general del modelo)
3. Tipo de datos: se distinguen los datos tabulares, de texto, las imágenes o los grafos.
4. Compatibilidad con el modelo: en este campo se diferencian los métodos específicos del modelo (adaptados a un tipo concreto de arquitectura) y los agnósticos (aplicables a cualquier modelo).

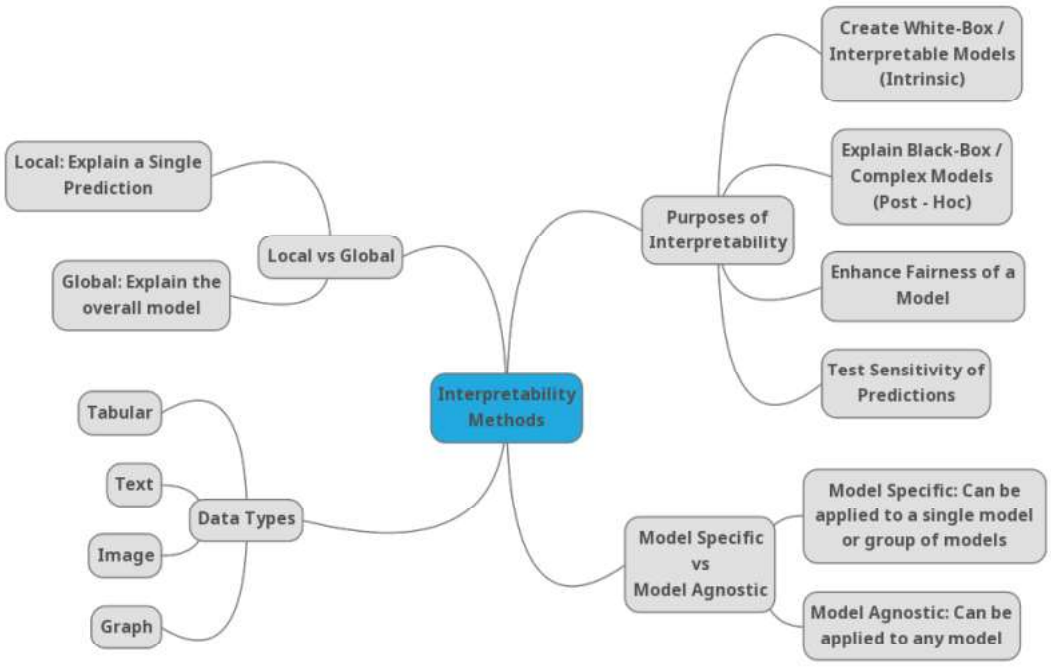


Figura 3.2: Esquema resumido de los diferentes aspectos por los que un método de interpretabilidad podría ser clasificado **[22]**.

Gracias a la diferenciación realizada, podemos saber que las técnicas que se usan en cada situación específica serán diferentes, dependiendo del propósito del modelo, el tipo de datos, o que tipo de modelo se está utilizando, además de su explicabilidad (global o local). La Figura 3.3 muestra presenta un esquema integral para la selección de técnicas de interpretabilidad, considerando diferentes ejes de decisión: el propósito del análisis, el tipo de datos a manejar y el tipo de modelo implicado. En función de estas variables, se muestra una clasificación de métodos organizados en dos grandes grupos.

- Métodos agnósticos al modelo, que pueden aplicarse a cualquier arquitectura y se subdividen según su objetivo en interpretabilidad global (explicación total del modelo) e interpretabilidad local (explicación de predicciones individuales).
- Métodos específicos al modelo, diseñados para arquitecturas concretas y centrados en proporcionar explicaciones locales.

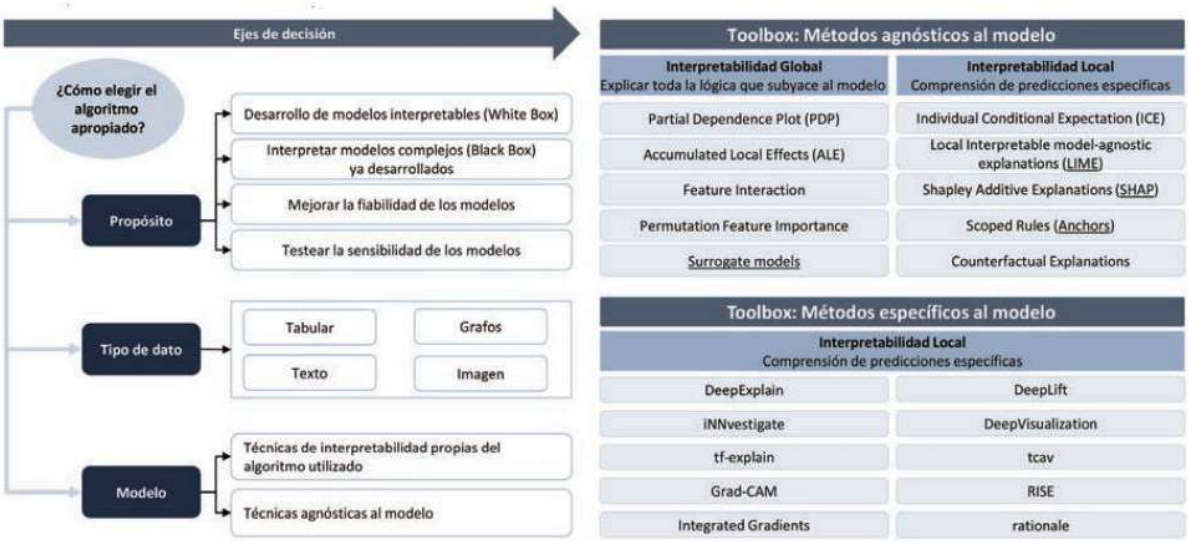


Figura 3.3: Visión general de las técnicas de interpretabilidad **[20]**.

## 3.4 Interpretabilidad “Ante-hoc” vs “Post-hoc”

En las secciones anteriores se han planteado dos enfoques de interpretabilidad de inteligencia artificial: la interpretabilidad “ante-hoc” y “post-hoc”.

La interpretabilidad “ante-hoc” se refiere a los modelos diseñados para ser interpretables desde su construcción, mientras que la interpretabilidad “post-hoc” implica la aplicación de técnicas de explicación e interpretación después de que el modelo ya haya sido entrenado. Se trata de una distinción crucial para seleccionar el enfoque adecuado en función de las necesidades de cada proyecto, ya que ambos presentan ventajas y desafíos en términos de precisión y transparencia. A lo largo de esta sección, se exploran las características de ambos enfoques, sus aplicaciones comunes, así como ejemplos para cada enfoque, además de su impacto en la confiabilidad y el entendimiento de los modelos de IA.

Cabe destacar que en este proyecto nos centramos en las técnicas de interpretabilidad “post-hoc” o de caja negra principalmente. La Figura 3.4 ilustra una relación inversa entre interpretabilidad y capacidad predictiva a través de diferentes familias de modelos de machine learning. En el extremo izquierdo se encuentran los modelos fácilmente interpretables, como la regresión lineal y los árboles de decisión, mientras que en el extremo inferior derecho aparecen los modelos de caja negra (“black box”), como las redes neuronales profundas,

convolucionales y recurrentes, que ofrecen una mayor capacidad predictiva a costa de una menor transparencia.

Es una figura importante a la hora de visualizar el compromiso entre explicabilidad y rendimiento que es esencial para seleccionar el modelo más adecuado según las necesidades y los requisitos de interpretabilidad.

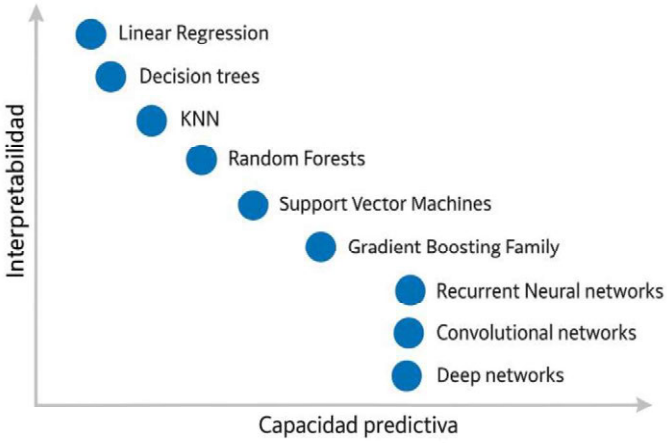


Figura 3.4: Balance entre interpretabilidad y capacidad predictiva por familias de modelos (incluyendo "white" y "black boxes") **[20]**.

### 3.4.1 Interpretabilidad "Ante-hoc"

Los modelos de caja blanca o "White box" previamente mencionados, están basados en el desarrollo de algoritmos que, por diseño, son inherentemente interpretables. Son modelos que se agrupan típicamente según el tipo de algoritmo empleado (modelos lineales, basados en árboles, redes neuronales sencillas con funciones de activación como ReLu, etc.), limitando los parámetros a optimizar para poder conseguir una mayor interpretabilidad.

Además, la interpretabilidad "Ante-hoc" es comúnmente vista como global debido a la naturaleza explicable de sus modelos más comunes como son los modelos lineales, árboles de decisión y los modelos basados en reglas, pero esto no excluye de la posibilidad de ofrecer interpretabilidad local al proporcionar explicaciones específicas para decisiones individuales.

Estos modelos proporcionan una explicación de la lógica interna del algoritmo y de la secuencia de pasos que se dan para llegar a un resultado concreto, permitiendo así una mejor comprensión de los resultados.

Estos son algunos de los modelos "White box" más conocidos:

- **Árboles de Decisión:** Son considerados interpretables debido a su estructura (nodos y ramas), que resulta fácil de visualizar y entender. Cada camino desde la raíz hasta una hoja representa una regla de decisión basada en los valores de las características.

  Algunas ventajas de los árboles de decisión son su fácil interpretación, la poca preparación de datos necesaria o su flexibilidad.

  Entre sus desventajas se encuentran el sobreajuste, la obtención de estimadores de alta varianza o el posible coste del entrenamiento [10].

- **Modelos lineales (Regresión lineal):** son modelos interpretables debido a su forma matemática sencilla, donde cada característica tiene un peso asociado que indica su impacto en la predicción.

  Un modelo de regresión lineal predice el objetivo como una suma ponderada de las entradas de características. La linealidad de la relación aprendida hace que la interpretación sea fácil, siendo estos modelos los más usados por estadísticos, científicos computacionales y otras personas que abordan problemas cuantitativos.

  Los modelos lineales pueden ser utilizados para modelar la dependencia de un objetivo de regresión $y$ respecto a algunas características $x$. Las relaciones aprendidas son lineales y pueden escribirse para una sola instancia de la siguiente manera [1]:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$$

  El resultado predicho para una instancia es una suma ponderada de sus $p$ características donde:

a) $\beta_j$ representan los pesos o coeficientes de las características aprendidos.

b) $\beta_0$ es llamado "offset" o "intercept" y no se multiplica por ninguna característica.

c) $\varepsilon$ es el error que comete el modelo, es decir, la diferencia entre la predicción y el resultado real.

### 3.4.2 Interpretabilidad "Post-hoc"

Las técnicas de interpretabilidad "post-hoc", también conocidas como interpretabilidad de modelos de caja negra o "black box", se centran en explicar la salida de modelos complejos ya entrenados, a partir de la información que proporcionan los pesos asignados a cada variable de entrada y los resultados de los modelos. Son técnicas útiles para comprender los resultados de los modelos, pero sin proporcionar información sobre el proceso de entrenamiento, ni explicar la lógica interna del algoritmo.

Una de las ventajas fundamentales de estas técnicas de interpretabilidad es "que se pueden interpretar modelos opacos a posteriori, sin sacrificar el rendimiento predictivo del modelo".

En cuanto a las técnicas de interpretabilidad "post-hoc" más comunes podemos encontrar las siguientes:

- **PDP** ("Partial Dependence Plots") [23]:

Los gráficos PDP muestran la variación de un modelo de IA en función de una o dos variables independientes en la predicción, permitiendo así evaluar el tipo de relación entre variables independientes y dependientes.

Su funcionamiento consiste en mostrar gráficamente en una curva la variación promedio de la predicción, cuya obtención se logra variando un predictor en todas las observaciones del "dataset", obteniendo luego el impacto medio en la predicción.

Se trata de un método con explicabilidad global y agnóstica del modelo, que permite visualizar la influencia que tiene cada variable individual en la salida del modelo, excluyendo al resto de variables.

- **SHAP** (“Shapley Additive exPlanations”) [24]:

Método de explicación de modelos basado en el Teorema de Valor de Shapley [19], propuesto en 1952 para distribuir el valor de un juego entre los jugadores. SHAP se utiliza para explicar la importancia de cada variable en una predicción concreta.

Es un método donde se calculan los valores de Shapley [25], que se corresponden con la contribución de cada variable a la predicción del modelo, y las variables independientes se interpretan como jugadores que colaboran para recibir el “payout”, que se trata de la predicción concreta realizada por el modelo menos el valor promedio de todas las predicciones.

Los jugadores reparten el “payout” en función de su contribución, en base a su valor de Shapley, lo que refleja la importancia de cada variable.

Se trata de un método que permite la explicación tanto local como global de los resultados del modelo, es decir, la explicación de la influencia de cada variable en observaciones del modelo, así como la importancia de cada variable en los resultados globales del modelo.

A modo ilustrativo, en la Figura 3.5 se muestra cómo los valores SHAP permiten explicar las predicciones individuales realizadas por un modelo de clasificación. En este caso, se presentan dos pacientes reales de un conjunto de datos de cáncer cervical, y cómo las distintas variables influyen positiva o negativamente en su riesgo pronosticado

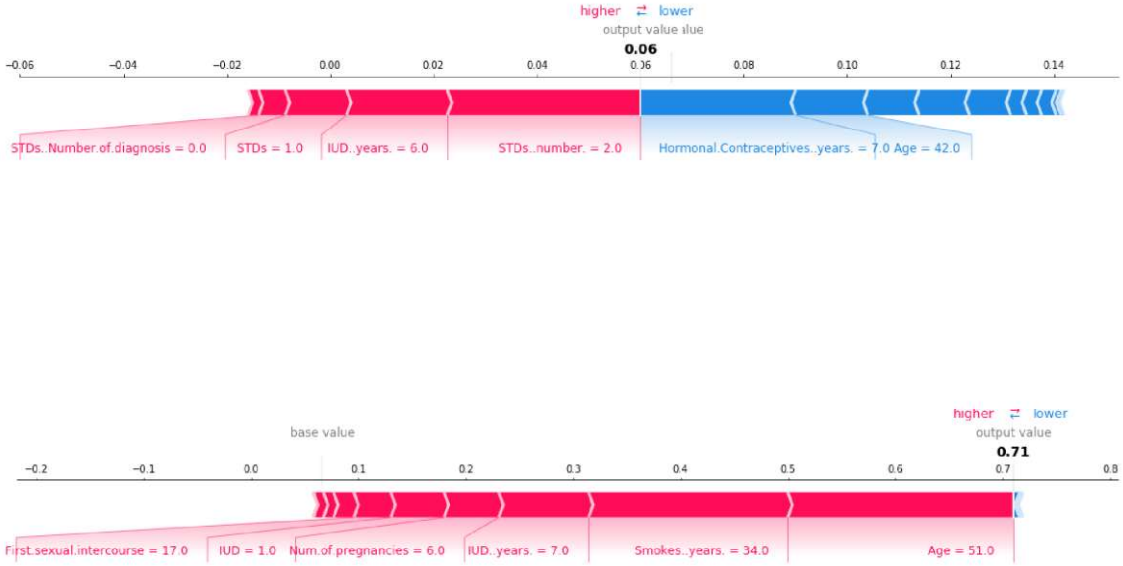


Figura 3.5: Ejemplo de visualización de valores SHAP que explican las probabilidades pronosticadas de cáncer cervical en dos mujeres del conjunto de datos **[26]**.

- **LIME** (“Local Interpretable Model-agnostic Explanations”) [27]:

Método local para comprobar cómo varían las predicciones de un modelo cuando se perturban los datos introducidos. Se aplican los siguientes pasos:

a) Generación de datos sintéticos alrededor de la instancia de datos de entrada: Toma una única predicción y los datos de entrada que la generaron como punto de partida y genera nuevos datos de entrada perturbando esta observación.

b) Entrena un modelo simple sobre los datos sintéticos: se usa un modelo interpretable (usualmente una regresión lineal) entrenado con muestras perturbadas y sus predicciones, de modo que aproxime el comportamiento del modelo de caja negra en la vecindad local de la instancia a analizar.

c) Explicar las predicciones del modelo simple en función de los datos originales: se obtiene la importancia de cada variable en la predicción, por ejemplo, en función de sus coeficientes en la regresión y su signo correspondiente.

d) Cálculo de la explicabilidad: el porcentaje de explicabilidad de LIME es equivalente al coeficiente de ajuste del modelo lineal. Por tanto, el modelo interpretable arroja una buena aproximación de las predicciones de manera local.

- **Anchors** [28]:

Es un método que explica predicciones individuales de modelos de clasificación de caja negra, mediante la búsqueda de reglas de decisión llamadas "anchors" que expliquen el resultado.

Su funcionamiento es similar a LIME: tomando como punto de partida una única predicción y los datos de entrada que la generaron, se generan nuevos datos perturbando esta observación para obtener nuevas predicciones.

Sin embargo, la explicación local de la predicción, que se obtiene buscando reglas de tipo "if-else" capaces de explicar el resultado del modelo. Se considera que una regla explica la predicción cuando los cambios en otras variables independientes no consideradas en la regla no la modifican.

Su objetivo es buscar la regla de decisión que explique el resultado.

- **Integrated Gradients (IG)** [5]:

Es un método de interpretabilidad "post-hoc" diseñado especialmente para redes neuronales profundas. Fue propuesto con el objetivo de superar algunas limitaciones observadas en técnicas basadas únicamente en gradientes instantáneos, como los "saliency maps" tradicionales.

A diferencia de los ya mencionados, que en ocasiones pueden resultar inestables, "Integrated Gradients" se basa en el cálculo acumulado de los gradientes a lo largo de una interpolación lineal entre una entrada base (por ejemplo, un vector nulo o una entrada sin información) y la entrada real que se desea explicar. Esta aproximación genera atribuciones más suaves, coherentes y robustas frente a pequeños cambios en la entrada.

El funcionamiento de "Integrated Gradients" se basa en integrar el gradiente parcial de la predicción respecto a cada característica, desde el punto de referencia hasta la entrada de interés. Este enfoque garantiza propiedades como:

a) Sensibilidad: si una característica cambia el resultado del modelo, su atribución será diferente a 0.

b) Linealidad: si el modelo es una combinación lineal de funciones, las atribuciones se combinan de forma coherente.

Se trata de un modelo que ha ganado relevancia por su solidez teórica, su fácil implementación en modelos existentes y su capacidad para generar explicaciones interpretables incluso en arquitecturas complejas. Se utiliza en tareas de clasificación, regresión y predicción secuencial, resultando muy útil cuando se desea analizar el impacto acumulado de secuencias de entrada, como ocurre en problemas temporales o series sensoriales.

A modo ilustrativo, en la Figura 3.6 se muestra como "Integrated Gradients" es capaz de identificar de forma progresiva las regiones relevantes de la entrada mediante la acumulación de gradientes. Aunque el ejemplo corresponde a una tarea visual, la misma lógica puede extenderse a entradas secuenciales o multivariantes, como las tratadas en este proyecto.

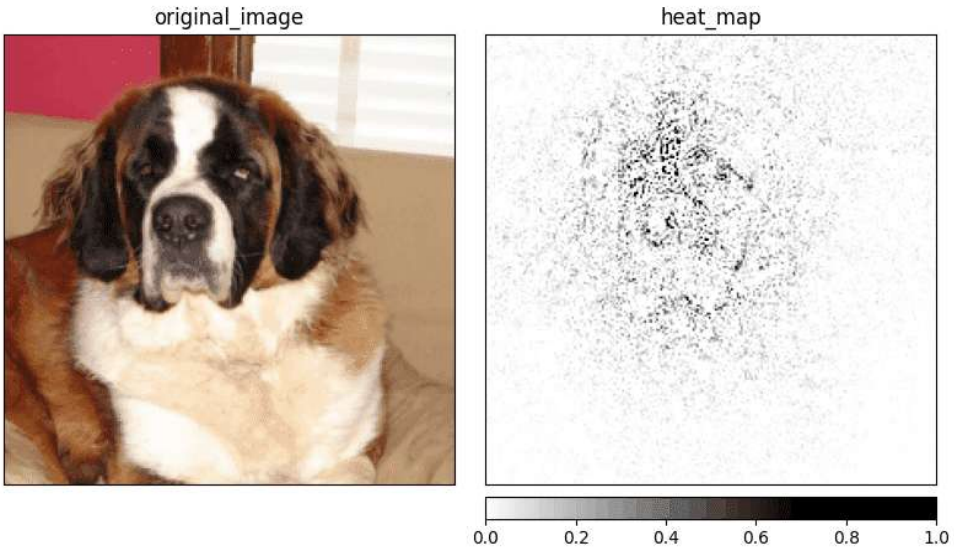


Figura 3.6: Visualización de un mapa de saliencia generado mediante Integrated Gradients (IG) para la clase saint_bernard **[29]**.

A continuación, en la Figura 3.7 se muestran las principales ventajas y desventajas de cada técnica de interpretabilidad "post-hoc" explicadas, además de las ventajas y desventajas de la construcción de modelos "White Box".

| Técnica | Ventajas | Desventajas |
|---|---|---|
| 1 PDP (Partial Dependence Plot) | ✓ Fácil de aplicar e intuitiva implementación.<br>✓ El cálculo de los gráficos de dependencia parcial tiene una interpretación causal. | X Por diseño, no permite ver el impacto de más de 2 variables intuitivamente en el gráfico.<br>X No explica cómo varia la explicación según una única variable independiente si varían el resto de variables independientes. |
| 2 LIME (Local interpretable model-agnostic explanations) | ✓ Dada una predicción, este método evalúa el impacto de ligeras modificaciones en los inputs.<br>✓ Se utiliza un modelo subrogado local para evaluar las diferencias entre las predicciones originales y las modificadas, así como las variables más importantes que contribuyen a la predicción.<br>✓ El método es agnóstico respecto al modelo de predicción utilizado. | X Se asume linealidad local.<br>X Puede arrojar explicaciones contrarias en distintos subconjuntos de datos, por lo que es necesario verificar las explicaciones en rangos representativos del *dataset*.<br>X No da una explicación global del modelo. |
| 3 SHAP (SHapley Additive exPlanations) | ✓ Calcula la contribución de cada variable a una predicción específica.<br>✓ No asume linealidad local.<br>✓ Puede cubrir la importancia global de las características para todo el conjunto de datos.<br>✓ Agnóstico respecto al modelo de predicción utilizado.<br>✓ Muy costoso computacionalmente y asume que las variables del modelo son independientes. | X Puede arrojar explicaciones contrarias en distintos subconjuntos de datos, por lo que es necesario verificar las explicaciones en rangos representativos del *dataset*.<br>X No da una explicación global del modelo. |
| 4 Anchors | ✓ Agnóstico al tipo de modelo y fácil de interpretar.<br>✓ Recoge comportamientos no lineales de modelos complejos. | X Gran número de hiperparámetros (forma de perturbación, precision...).<br>X Requiere discretizar variables continuas en muchos casos, pudiendo llevar a errores en la interpretación. |
| 5 *Construcción de Modelos "White Box"* | ✓ Reduce el esfuerzo en interpretación de modelos tras el entrenamiento, y durante su ciclo de vida.<br>✓ No lleva a contradicciones en la interpretación del modelo y facilita su uso.<br>✓ No requiere del empleo de modelos o técnicas adicionales *post-hoc*. | X Incrementa el esfuerzo durante la construcción del modelo.<br>X No existen técnicas aplicables para todo tipo de modelos, por el momento. |

Figura 3.7: Ventajas y desventajas de los métodos explicados **[20]**.

Cabe destacar como la existencia de dos tipos de métodos de interpretabilidad permite a los interesados tener la posibilidad de elegir la técnica más adecuada, según la necesidad de entender el modelo antes o después del entrenamiento. Los métodos “Ante-hoc” aseguran simplicidad y transparencia desde el principio, mientras que los “Post-hoc” cuentan con la ventaja de ser versátiles y poder aplicarse a cualquier modelo.

# Capítulo 4: Análisis de Supervivencia

## 4.1 Introducción

Durante el Capítulo 4 se trata el análisis de supervivencia [30], un conjunto de técnicas estadísticas diseñadas para estudiar el tiempo hasta que ocurre un evento de interés, como el fallo de una máquina, la duración de un tratamiento médico o la retención de clientes. Se trata de un análisis particularmente útil en estudios longitudinales al largo del tiempo, como las series temporales, y en situaciones en las que no todos los sujetos experimentan el evento de interés, introduciendo de este modo la censura en los datos.

A lo largo del capítulo, se describen los conceptos básicos del análisis de supervivencia, las funciones de supervivencia y riesgo, así como uno de los métodos estadísticos más utilizados, el estimador de Kaplan-Meier [31]. Además, se introduce el modelo de riesgos proporcionales de Cox [32], un enfoque que permite estudiar la relación entre variables explicativas y el tiempo hasta el evento, ajustando por múltiples factores.

También, se discuten aplicaciones en diversos campos, así como las consideraciones importantes para interpretar adecuadamente los resultados.

## 4.2 ¿Qué es el análisis de supervivencia?

El análisis de supervivencia es un conjunto de técnicas que permiten estudiar el tiempo hasta que ocurre un evento, y su dependencia de otras posibles variables explicativas. Por tanto, se puede definir como un conjunto de métodos estadísticos que valoran el tiempo entre un evento inicial (inclusión de un sujeto en el estudio a realizar) y uno final, que sucede cuando este presenta una característica definida con anterioridad (evento).

Una dificultad común radica en que, al final del periodo de seguimiento, frecuentemente hay individuos en los que, por diversos motivos, no se ha podido observar el evento y en los que, por tanto, el tiempo hasta su ocurrencia es desconocido. Este fenómeno se denomina censura y es una característica central del análisis de supervivencia.

Aunque los datos censurados no contienen información directa sobre cuándo ocurrió el evento, aportan información valiosa sobre el período durante el cual no ocurrió. Por ejemplo, si un paciente abandona el estudio a los 3 años sin haber experimentado el evento, sabemos que al menos durante esos 3 años el evento no ocurrió. Esta información permite realizar estimaciones menos sesgadas, al aprovechar tanto los tiempos completos como los incompletos, lo que es esencial para la precisión y utilidad del análisis.

A la hora de calcular la probabilidad de supervivencia, existen varios métodos, siendo el más utilizado el de Kaplan-Meier, debido a su simplicidad y utilidad en el análisis descriptivo de los datos de supervivencia [31].

Por otra parte, el modelo de riesgos proporcionales de Cox es ampliamente utilizado para analizar la relación entre el tiempo hasta el evento y otras variables explicativas, proporcionando un enfoque más completo en estudios que buscan identificar factores asociados al evento [32].

El análisis de supervivencia se suele aplicar comúnmente en la medicina con el fin de valorar la supervivencia de los pacientes a una determinada enfermedad. Se trata de estudios

llamados longitudinales, en los que se sigue a los individuos a lo largo del tiempo (desde la entrada al estudio del individuo hasta la ocurrencia del evento). El evento esperado no necesariamente es la muerte, puede ser tiempo que dura la eficacia de una intervención, tiempo de persistencia de un tratamiento, etc. Siendo, además, una técnica muy útil para otros campos fuera de la medicina, como las campañas publicitarias para la retención de clientes o sus aplicaciones en ingeniería, por ejemplo, la estimación de la vida útil de las máquinas [33].

El nombre "Análisis de supervivencia" proviene de la aplicación prolongada de estos métodos, ya que durante siglos estuvieron únicamente vinculados a la investigación de las tasas de mortalidad, extendiéndose a otros campos más allá de la investigación médica únicamente durante las últimas décadas.

Estos estudios están especialmente diseñados para determinar la probabilidad de un suceso en diferentes intervalos de tiempo, siendo siempre el tiempo la variable dependiente. Además, como ya se ha comentado, pueden participar pacientes con tiempos incompletos que no han llegado al estadio final (evento).

## 4.3 Conceptos básicos

A continuación, se enumeran los conceptos básicos relacionados con el análisis de supervivencia [33] [34]:

1) **Fecha de inicio**: Es igual para todos los pacientes y se corresponde con la inclusión de pacientes tras haber terminado los preparativos.
2) **Fecha de entrada:** Se corresponde a la incorporación del paciente en el estudio. Es importante, porque a partir de ella se calcula el tiempo de supervivencia. Es diferente para cada sujeto.
3) **Fecha de última observación:** Al igual que la fecha de entrada, es diferente para cada paciente y se corresponde con la última visita o revisión realizada.
4) **Tiempo de participación:** Tiempo transcurrido entre la fecha de entrada y la fecha de última observación.
5) **Fecha de cierre:** Se corresponde con la finalización del estudio, y será igual para todos los pacientes.
6) **Duración del estudio:** Intervalo de tiempo transcurrido entre la fecha de entrada y la fecha de cierre.
7) **Tiempo de supervivencia:** Intervalo de tiempo entre la entrada del paciente en el estudio y el evento de interés.
8) **Paciente censurado:** Se trata de aquellos pacientes cuyo seguimiento es incompleto o está perdido debido a la finalización de su participación antes de la fecha de cierre. El tiempo de participación puede terminar por diversos motivos:
    a. El paciente decide no participar más en el estudio y lo abandona.
    b. El paciente se pierde (por ejemplo, cambio de domicilio) y no se dispone de más información sobre él.
    c. El estudio termina antes de aparecer el evento que estudiamos.

Como se muestra en el ejemplo de la Figura 4.1, el estudio comenzó en 1990 y acabó en 2002. El eje A representa años de calendario y el eje B los años desde el diagnóstico. Además, con el círculo blanco se representan los tiempos censurados y con el cuadrado negro la ocurrencia del evento (la muerte en este caso específico).

El paciente A fue diagnosticado en 1990 y desapareció en 1993 (censura de 3 años por pérdida de seguimiento). El B, fue diagnosticado en 1990 y falleció en 1992 (muerte a los 2,5 años). El C seguía vivo al acabar el estudio (censurado a los 12 años por fin del estudio). Al

D se le diagnosticó en 1991 y falleció en 1999, teniendo un tiempo de supervivencia de 8 años. El E fue diagnosticado en noviembre de 1993 y fallecido en julio de 1997 por un accidente de tráfico (dato censurado a los 3,7 años según la definición del evento). Por último, el F, fue diagnosticado en 1996 y seguía vivo al acabar el estudio (dato censurado a los 6 años por fin del estudio).

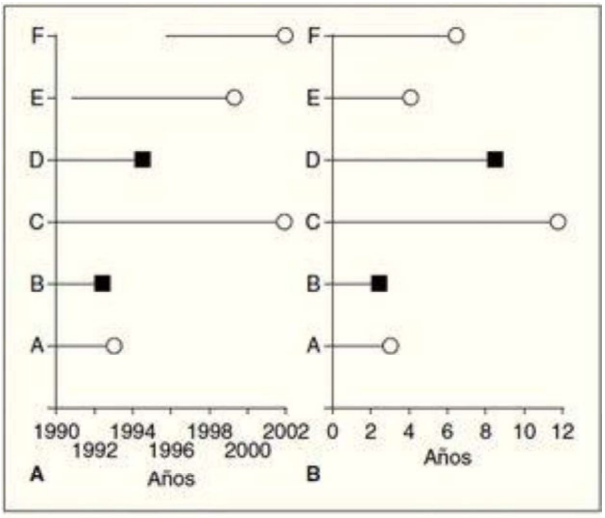


Figura 4.1:Esquema del tiempo de supervivencia tras la entrada en un estudio **[35]**

## 4.4 Consideraciones importantes

Tras haber definido algunos conceptos básicos relacionados con el análisis de supervivencia o “tiempo hasta un evento” (“time-to-event”), es importante aclarar algunos puntos importantes a tener en cuenta para poder llevar a cabo un estudio.

Es necesario usar una definición operativa del punto de partida (ingreso en un hospital, inicio de un tratamiento, etc.) para poder fijar la fecha de entrada en el estudio. La fecha debe ser exacta y no depender de ambigüedades como podría ser la memoria del paciente para indicar el comienzo de sintomatología.

Por otra parte, el evento de interés debe ser claro y estar muy bien definido para poder precisar la fecha de este. En caso de ser la muerte del paciente, es imprescindible que esta sea como consecuencia de la enfermedad objeto del estudio. Si no fuera así, el fallecido debería valorarse como censurado y su tiempo de seguimiento como incompleto. Si este no fuera censurado, se estaría introduciendo en el estudio un sesgo de información.

En la última observación es preciso registrar el estado del sujeto y la fecha de información de dicho estado. Si el evento de interés es la muerte y el paciente ha fallecido, se calculará el tiempo de supervivencia en base a la fecha de defunción. Sin embargo, si este siguiera vivo, el tiempo sería incompleto o censurado, ya que para considerar un tiempo completo es necesaria la presencia del evento.

Por tanto, se pueden diferenciar tres requisitos para hacer un buen uso de los datos:

1. Definición adecuada del origen o inicio del seguimiento.
2. Definición adecuada de la escala del tiempo.
3. Definición precisa del evento de interés.

A continuación, en la Figura 4.2 se muestran algunos ejemplos de nacimiento y muerte, así como posibles casos de censura aplicada a varios casos de uso en la industria.

| Application field | Birth event | Death event | Censoring example |
|---|---|---|---|
| Predictive maintenance in mechanical operations | Time the machine was started for a continuous use | Time when the machine broke down | Machine broke down due to a fire in the factory building |
| Customer analytics | Customer started subscription | Time when customer unsubscribed | Customer died during the observation time |
| Medical research on breast cancer patients | Time the patient was first diagnosed | Time the patient died due to breast cancer | Patient died due to a cardiovascular disease |
| Lifetimes of political leaders around the world | Start of the tenure | Retirement | Leader died during the tenure |

Figura 4.2: Eventos de nacimiento, muerte y posibles censuras aplicado a su uso en la industria **[36]**.

### 4.4.1 Sesgos

Como se ha comentado en apartados anteriores, los datos del estudio pueden estar sesgados, tanto por las censuras, cuando el seguimiento termina antes de producirse el evento o cuando este no llegó a producirse, como por los truncamientos.

En los **truncamientos**, se entró al estudio después del hecho que define el origen en todos los individuos analizados, cuando se tendría que haber entrado con anterioridad para poder realizar un estudio completo.

De este modo, pueden distinguirse cuatro tipos de observaciones de datos:

1. **No truncada, No censurada:** Se inicia al comienzo del estudio y el evento sucede durante el seguimiento realizado.
2. **No truncada, Censurada:** Se inicia al comienzo del estudio, pero el evento no tiene lugar durante el seguimiento.
3. **Truncada, No censurada:** El paciente ya tenía el proceso cuando entró al estudio (entró tarde) y el evento sucede durante el seguimiento realizado.
4. **Truncada, Censurada:** El paciente ya tenía el proceso cuando entró al estudio y el evento no aparece durante el seguimiento realizado.

## 4.5 Función de supervivencia

El conjunto de métodos estadísticos relacionados con el análisis de supervivencia tiene el objetivo de estimar la función de supervivencia a partir de datos de supervivencia.

La **función de supervivencia,** $S(t)$, define la probabilidad de que un sujeto de interés sobreviva más allá del tiempo $t$, o de manera equivalente, la probabilidad de que la duración sea al menos $t$**.** La función de supervivencia de una población se define de la siguiente manera:

$$S(t) = P_r(T > t)$$

En esta ecuación, $T$ es la vida útil aleatoria tomada de la población en estudio y no puede ser negativa. Por ejemplo, en el caso de negocios, es el tiempo en que un cliente puede pagar sus cuotas de préstamo sin incumplir. Las propiedades fundamentales de la función de supervivencia S(t) se resumen a continuación.

1. $0 \leq S(t) \leq 1$
2. $F_T(t) = 1 - S(t)$, donde $F_T(t)$ es la CDF ("Cumulative Distribution Function" probabilidad de que una variable aleatoria continua tome un valor menor o igual que un cierto umbral) de $T$
3. La propiedad anterior implica que $S(t)$ no es una función creciente de $t$.

Al inicio del estudio $(t = 0)$, ningún sujeto ha experimentado el evento aún, de modo que la probabilidad $S(0)$ de sobrevivir más allá del tiempo cero es 1. Por otro lado, $S(\infty) = 0$ ya que, si el periodo de estudio fuera eliminado, todos eventualmente experimentarían el evento de interés y la probabilidad de sobrevivir finalmente caería a 0.

En teoría, la función de supervivencia es continua. Sin embargo, en la práctica, los eventos se observan en una escala de tiempo concreta (por ejemplo, días, semanas, meses, etc.), de modo que el gráfico de la función de supervivencia se convierte en una función escalonada en la práctica. Esta diferencia entre el comportamiento teórico continuo de la función de supervivencia y su representación práctica como función escalonada puede observarse en la Figura 4.3.

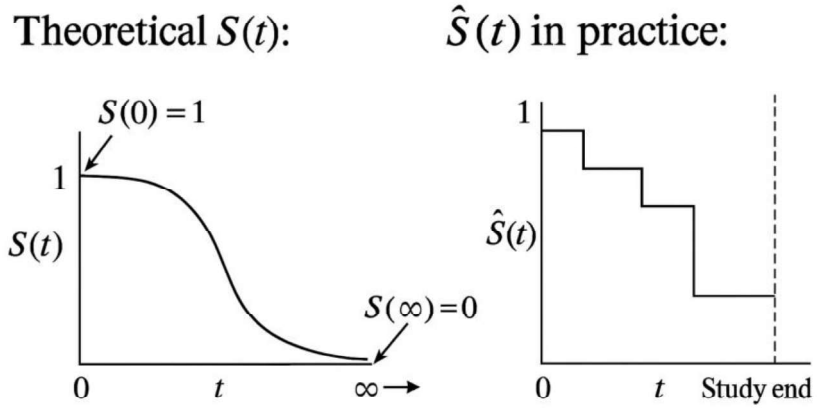


Figura 4.3: Función de supervivencia teórica y práctica **[36]**.

## 4.6 Función de riesgo

Derivada de la función de supervivencia, la **función de riesgo** $h(t)$ proporciona la probabilidad de que el evento ocurra en un tiempo $t$, dado que el sujeto no experimentó el evento hasta el tiempo $t$**.** Esta función describe el potencial instantáneo por unidad de tiempo para que ocurra el evento. Matemáticamente, la función de riesgo en el tiempo $t$ se define:

$$h(t) = \lim_{\delta t \to 0} \frac{P_r(t < T \leq t + \delta t | T > t)}{\delta t}$$

Por lo tanto, la función de riesgo modela qué períodos tienen las mayores o menores probabilidades de un evento. A diferencia de la función de supervivencia, la función de riesgo no tiene que empezar en 1 (cuando t = 0) y bajar a 0 (cuando $t \rightarrow \infty$).

La tasa de riesgo normalmente cambia con el tiempo, pudiendo empezar en cualquier valor y subiendo y bajando con el tiempo. Por ejemplo, si hablamos de la probabilidad de incumplir el pago de una hipoteca, esta puede ser baja al principio e ir aumentando con el tiempo.

En la Figura 4.4 se muestra un ejemplo teórico de una función de riesgo, donde se especifica también la llamada curva de bañera debido, a su forma, mostrando la probabilidad de que ocurra un evento de interés a lo largo del tiempo.

Este gráfico podría estar describiendo la probabilidad de que un cliente se dé de baja de una revista a lo largo del tiempo. Durante los primeros 30 días el riesgo es moderado, ya que es un periodo de prueba. Si el cliente "sobrevive" ese primer periodo, el riesgo disminuye y se estanca. Un tiempo después, el riesgo puede aumentar debido a la necesidad, por ejemplo, de novedades para el cliente, por lo que el gráfico proporciona la información sobre cuando iniciar incentivos para aquellos clientes cuyo riesgo de darse baja esté a punto de aumentar.

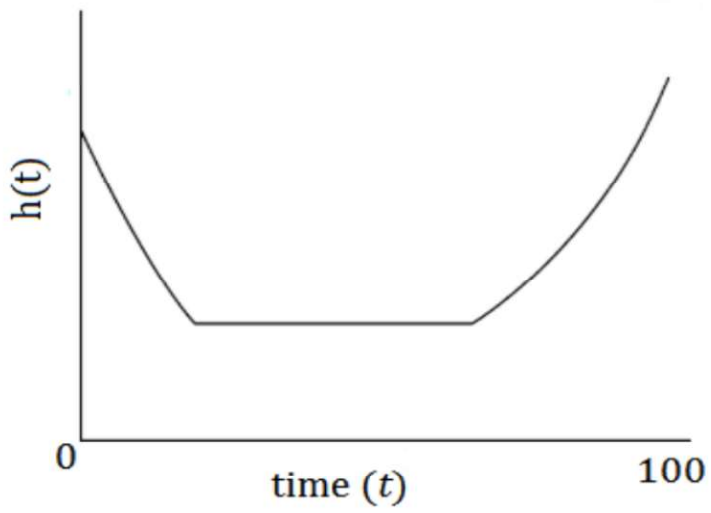


Figura 4.4: Función de riesgo (ejemplo teórico) **[36]**.

La función de riesgo $h(t)$ se puede expresar en términos de la función de densidad $f(t)$ y la función de supervivencia $S(t)$ como:

$$h(t) = \frac{f(t)}{S(t)}$$

Esta expresión refleja la interpretación de $h(t)$ como la tasa instantánea de fallo en tiempo $t$, condicionada a que el evento no haya ocurrido antes. Además, partiendo de esta relación se puede deducir una expresión directa de la función de supervivencia a partir de la función de riesgo:

$$S(t) = \exp\left(-\int_o^t h(u)\, du\right.$$

Esta ecuación muestra cómo la probabilidad de supervivencia hasta un tiempo $t$ decrece en función de la integral acumulada del riesgo. En muchos modelos paramétricos, como el de Weibull, esta relación es clave para construir expresiones cerradas de $S(t)$ y $h(t)$ que dependen de parámetros estructurales del modelo.

Por tanto, el objetivo principal del análisis de supervivencia es estimar e interpretar las funciones de supervivencia y/o de riesgo a partir de los datos de supervivencia.

## 4.7 Métodos estándar

Los métodos estándar para la estimación en el análisis de supervivencia pueden clasificarse en tres grupos principales: paramétricos, semiparamétricos y no paramétricos [36]. La elección de que métodos usar guiada por los datos y la pregunta de investigación de interés, pudiendo usarse más de un enfoque.

- **Métodos Paramétricos:** se basan en la suposición de que la distribución de los tiempos de supervivencia corresponde a una distribución de probabilidad específica. En este grupo se incluyen métodos como las distribuciones exponenciales, Weibull y log-normal, y los parámetros dentro de estos métodos generalmente se estiman utilizando estimaciones de máxima verosimilitud.

- **Métodos No Paramétricos:** no asumen una distribución específica para los tiempos de supervivencia. Principalmente para dar una visión promedio de la población individual. El método univariante más popular es el estimador de Kaplan-Meier [31], utilizado como primer paso en el análisis descriptivo de supervivencia.

- **Métodos Semiparamétricos:** a este grupo corresponde el modelo de regresión de Cox, que se basa en componentes tanto paramétricos como no paramétricos.

El rango de métodos estadísticos disponibles en el análisis de supervivencia es muy extenso. En la Figura 4.5 se presentan algunos de los principales [36], y a continuación se describen brevemente tres métodos relevantes.

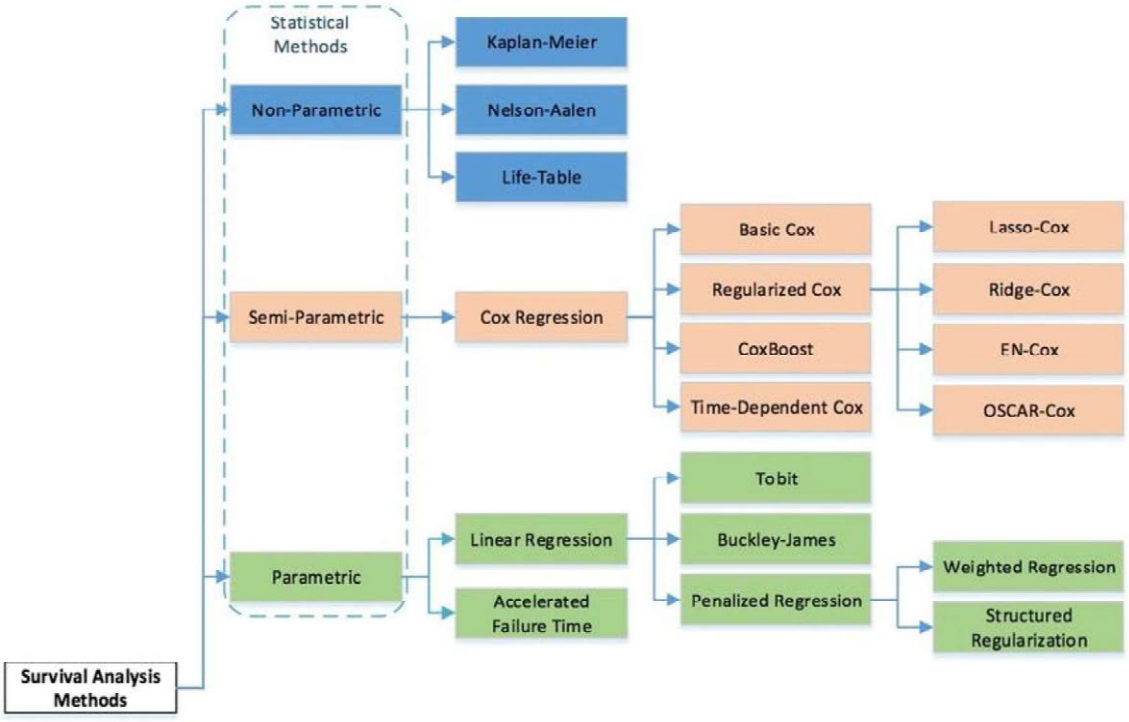


Figura 4.5: Diagrama de métodos estadísticos **[36]**.

### 4.7.1 Estimador de Kaplan-Meier

Este método, también conocido como del "límite del producto", supone que los sujetos que han sido censurados se habrían comportado del mismo modo que los seguidos hasta que tuvo lugar el evento, asumiendo el evento terminal como independiente para cada paciente. Esto permite que las probabilidades de sobrevivir en un tiempo determinado se calculen gracias a una ley multiplicativa de probabilidades.

La idea clave del estimador de Kaplan-Meier es descomponer la estimación de la función de supervivencia $S(t)$ en pasos mas pequeños dependiendo de los tiempos observados de los eventos, de modo que, para cada intervalo, se calcula la probabilidad de sobrevivir hasta el final de este, mediante la siguiente formula [31]:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \frac{n_i - d_i}{n_i},$$

Donde $n_i$ es el número de individuos que están en riesgo en el punto de tiempo $t_i$ y $d_i$ es el número de sujetos que experimentaron el evento en el tiempo $t_i$.

Existen ciertas suposiciones a tener en cuanta al usar el estimador de Kaplan-Meier [31].

- Todas las observaciones, tanto censuradas como incumplidas, se utilizan en la estimación.
- No hay "efecto de cohorte" en la supervivencia, lo que quiere decir que los sujetos tienen la misma probabilidad de supervivencia independientemente de su naturaleza y tiempo de aparición en el estudio.
- Los individuos que son censurados tienen las mismas probabilidades de supervivencia que aquellos que continúan siendo examinados.
- La probabilidad de supervivencia es igual para todos los sujetos

Al tratarse de un método con enfoque univariante, cuenta con la desventaja de no poder estimar la probabilidad de supervivencia considerando todas las características disponibles en los datos. En consecuencia, no muestra estimaciones individuales, sino la distribución de supervivencia de la población en general. En comparación, los modelos semiparamétricos y paramétricos permiten analizar todas las covariables y estimar $S(t)$ con respecto a ellas.

El $\hat{S}(t)$ estimado se puede trazar como una función escalonada de la población total de individuos. Por ejemplo, en la Figura 4.6 puede observarse como, para el tiempo $t = 10$, la probabilidad de sobrevivir más allá de ese tiempo es aproximadamente del 75%. La Figura 4.6 ilustra cómo la probabilidad de supervivencia cambia con el tiempo, destacando momentos clave donde ocurren eventos de incumplimiento y cómo afectan a la estimación general de supervivencia de la población.

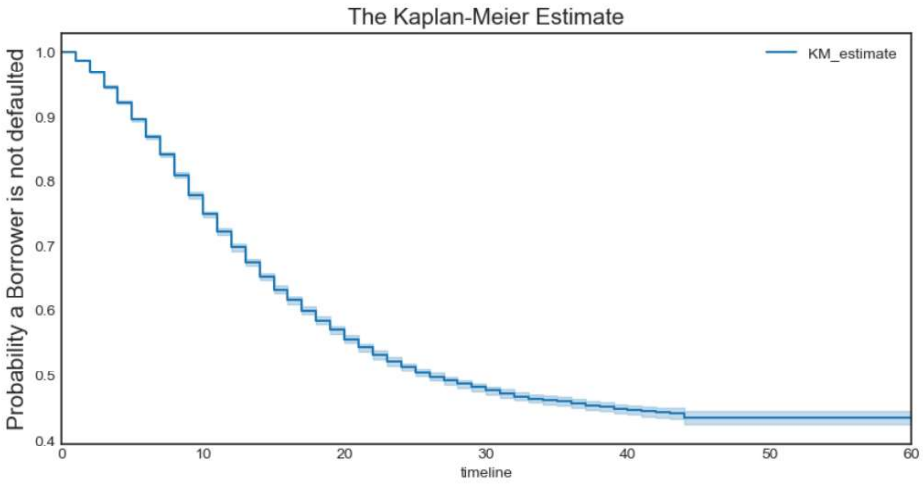


Figura 4.6: Probabilidad de supervivencia en el tiempo usando Kaplan-Meier **[36]**

### 4.7.2 Modelo de Riesgos Proporcionales de Cox

El modelo de riesgos proporcionales de Cox (CoxPH) no solo incluye características de tiempo y censura, sino también datos adicionales como covariables de cada individuo. Este modelo fue introducido en 1972 [32], y sigue siendo ampliamente utilizado en estadísticas de supervivencia multivariadas debido a su implementación relativamente fácil y su interpretación informativa. Describe las relaciones entre la distribución de supervivencia y las covariables.

La variable dependiente, el tiempo hasta el siguiente evento, se expresa mediante la función de riesgo (o intensidad de incumplimiento) de la siguiente manera:

$$\lambda(t|x) = \lambda_0(t)\exp(\beta_1 x_1 + \cdots + \beta_n x_n)$$

donde:

- $\lambda(t|x)$ es la función de riesgo en el tiempo $t$ dadas las covariables $x$.
- $\lambda_0(t)$ es el riesgo basal, el valor del riesgo cuando todas las covariables son iguales a 0.
- $\beta_1\beta_2, \ldots, \beta_p$ son los coeficientes de las covariables $x_1x_2, \ldots, x_p$.

Este método es considerado semiparamétrico y contiene:

- Un conjunto paramétrico de covariables y un componente no paramétrico, $\lambda_0(t)$, que se llama riesgo basal.
- El segundo componente se corresponde con los riesgos parciales o razones de riesgo y definen el efecto de las covariables observadas en el riesgo basal $\lambda_0(t)$.

Estos componentes se estiman mediante verosimilitud parcial y son invariantes en el tiempo.

En general, el modelo de Cox hace una estimación de la función de log-riesgo $\ln\lambda(t|x)$ como una combinación lineal de sus covariables estáticas y el riesgo basal.

### 4.7.3 Bosques aleatorios de supervivencia

Otro enfoque de aprendizaje automático factible que se puede utilizar para evitar la restricción proporcional del modelo de riesgos proporcionales de Cox es un bosque aleatorio de supervivencia [37] (RSF, “Random Survival Forest” por sus siglas en inglés).

El bosque aleatorio de supervivencia se define como un método de árbol que construye una estimación en conjunto para la función de riesgo acumulada. Además de construir los conjuntos a partir de aprendices base, como árboles, puede mejorar sustancialmente el rendimiento de la predicción.

Básicamente, RSF calcula un bosque aleatorio utilizando la prueba de log-rank [37] como criterio de división. Esta prueba es un estadístico no paramétrico ampliamente utilizado en el análisis de supervivencia para comparar funciones de supervivencia entre dos o más grupos, y se adapta aquí como métrica para maximizar la separación entre ramas del árbol según el tiempo hasta el evento. Se calcula los riesgos acumulados de los nodos hoja en cada árbol y se promedian entre todos los árboles para obtener la predicción final.

El árbol se desarrolla hasta su tamaño completo bajo la condición de que cada nodo terminal tenga al menos un número preespecificado de eventos (muertes) y se emplean las muestras fuera de la bolsa (OOB “Out Of Bound”) para estimar el error de predicción del conjunto, a través de la función de riesgo acumulada.

Los bosques aleatorios de supervivencia se engloban dentro de los métodos no paramétricos, ya que no asume una forma funcional específica para relacionar variables ni para la función de supervivencia. Además, podemos encontrar a continuación varias ventajas en cuanto al uso de este método.

- Permite modelas relaciones no lineales y con múltiples interacciones.
- No requiere la asunción de riesgos proporcionales.
- Maneja de forma natural datos censurados.
- Proporciona estimaciones estables y robustas incluso en presencia de ruido o colinealidad.

# Capítulo 5: Modelo "Weibull Time To Event Recurrent Neural Network" (WTTE-RNN)

## 5.1 Introducción

En apartados anteriores se ha revisado el análisis de la predicción del tiempo hasta un evento, resaltando algunos de sus principales desafíos o dificultades, como puede ser el manejo de datos censurados, recurrentes y con covariables que varían en el tiempo. Además, se trata de un campo de estudio relevante en diversas áreas, como la previsión de mantenimiento de maquinaria, la supervivencia médica, la predicción del comportamiento del cliente o las predicciones en series temporales.

A lo largo del Capítulo 5 se explorará el modelo "Weibull Time To Event Recurrent Neural Network" (WTTE-RNN) [4], una arquitectura diseñada para predecir el tiempo hasta un evento en el contexto de series temporales, tratando problemas tanto de tiempo continuo como discreto. Es un modelo que aborda los problemas mencionados de manera eficaz, proponiendo una solución innovadora al combinar modelos de supervivencia con redes neuronales recurrentes (RNN).

Este modelo combina la flexibilidad de las redes neuronales recurrentes con la capacidad de las distribuciones de Weibull para modelar tiempos de ocurrencia, siendo $\alpha$ y $\beta$ los parámetros estimados dinámicamente en cada paso temporal mediante la RNN. Esto permite calcular la probabilidad de que ocurra un evento en el futuro basándose en datos secuenciales, además de ser especialmente útil en el análisis de supervivencia con datos secuenciales.

El capítulo se desarrolla presentando los fundamentos del modelo, los principios de las redes neuronales recurrentes (RNN) y las características de la distribución de Weibull. Además, se analiza cómo este modelo aborda la predicción de eventos en el tiempo y cómo se compara con otros enfoques en términos de precisión y aplicabilidad en problemas de supervivencia.

## 5.2 ¿Qué es WTTE-RNN? Fundamentos

WTTE-RNN es de un modelo basado en una arquitectura de red neuronal recurrente (RNN) para procesar datos secuenciales, estimando en cada paso los parámetros $\alpha$ y $\beta$ de la distribución de Weibull con la que se combina con el fin de modelar el tiempo hasta el evento (TTE, "Time-To-Event").

Estos parámetros permiten calcular la probabilidad de ocurrencia del evento en distintos intervalos temporales y capturar la dinámica cambiante de los datos. En cada paso temporal, la red actualiza su estado oculto con las características de entrada para poder predecir una distribución sobre el tiempo hasta el próximo evento.

La distribución de Weibull es adecuada en este tipo de problemas especialmente debido a la flexibilidad que ofrece para representar distintos comportamientos de riesgo, pudiendo ser estos crecientes, decrecientes o constantes a lo largo del tiempo. Además, esta flexibilidad tiene una gran importancia a la hora de capturar la dinámica temporal de los datos observados [38].

### 5.2.1 Distribución de Weibull

Existe una amplia diversidad de distribuciones apropiadas para modelar tiempos de espera. Las más comúnmente usadas en el análisis de supervivencia y en la ingeniería de

confiabilidad son la exponencial, log-logística, gamma, de Rayleigh y de Weibull para los casos continuos, para analizar datos de la vida útil, mientras que destacan la distribución de Poisson, Geométrica y la de Weibull discreta para los casos discretos [39].

Se opta por la distribución de Weibull [38] para el caso de WTTE-RNN debido a su flexibilidad para representar diferentes patrones de riesgo. Esto permite modelar situaciones en las que el riesgo del evento aumenta, disminuye o permanece constante a lo largo del tiempo, una característica clave en el análisis de supervivencia con datos secuenciales. Además, su parametrización sencilla permite que la RNN ajuste dinámicamente $\alpha$ y $\beta$, adaptándose a los cambios en las características de entrada.

La distribución de Weibull es una distribución flexible gracias a su capacidad para adaptarse a diferentes formas de datos de fallos, desde aquellos que ocurren en los primeros momentos de uso ("infart mortality") hasta los fallos que ocurren después de un largo periodo de uso (desgaste). Se trata de una distribución fácil de discretizar, unimodal pero expresiva, empíricamente flexible, que cuenta con mecanismos de regularización y transformación de ubicación-escala, además de que la función de distribución acumulativa (CDF) y la función cuantil tienen forma cerrada y son numéricamente estables [38].

La distribución de Weibull continua se parametriza generalmente como [38]:

PDF: $F(x) = 1 - \exp(-\left(\left(\frac{t}{\alpha}\right)^{\beta}\right), t \geq 0$

CDF: $f(x) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \exp(-\left(\frac{t}{\alpha}\right)^{\beta}),\ t \geq 0$

HF ("Hazard Function"): $\lambda(\mathrm{x}) = \left(\frac{t}{\alpha}\right)^{\beta-1} \times \frac{\beta}{\alpha}$

Donde $t \in [0, \infty)$, el parámetro de escala $\alpha \in (0, \infty)$, y el parámetro de forma $\beta \in (0, \infty)$. Para indicar que una variable aleatoria sigue una distribución de Weibull se denota como $T \sim Weibull(\alpha, \beta)$.

En la Figura 5.1 pueden observarse dos graficas, en la gráfica (a) se ve como la forma de la distribución cambia cuando se ajustan los parámetros $\beta$ y $\alpha$. A mayor valor de $\beta$, se produce un pico mayor, mientras que valores menores de $\beta$ crean una distribución más dispersa y una cola más larga. Esto permite observar como la distribución Weibull puede adaptarse a diferentes comportamientos de fallos o tiempos de vida. Por otro lado, la gráfica (b) muestra la CDF de la distribución Weibull, lo que facilita la visualización de la probabilidad acumulada hasta un cierto punto. Es una representación útil para entender el tiempo esperado hasta un fallo determinado en función de la forma de la distribución.

La distribución de Weibull es unimodal, lo que significa que tiene como máximo un pico. Además de ser muy expresiva. Cuando $\beta \leq 1$, la función de densidad de probabilidad (PDF) es estrictamente decreciente, cuando $\beta > 1$, la función de riesgo aumenta con el tiempo, lo que modela procesos de envejecimiento o deterioro progresivo, y cuando $\beta = 1$, la función de riesgo es constante. Además, cuando $\beta = 1$, se tiene la distribución exponencial para el caso continuo y la distribución geométrica en el caso discreto.

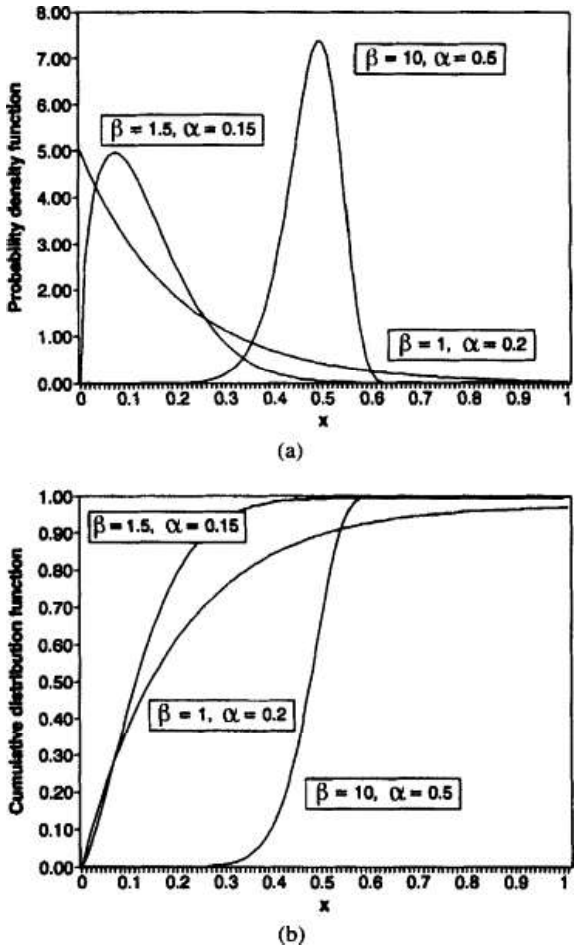


Figura 5.1: Ejemplo de funciones de densidad de probabilidad (PDF) y distribución acumulativa (CDF) de Weibull **[38]**

### 5.2.2 Redes Neuronales Recurrentes (RNNs)

Como ya se ha comentado a lo largo del capítulo 5, las RNNs en WTTE-RNN procesan las características secuenciales de los datos para generar los parámetros $\alpha$ y $\beta$ de la distribución de Weibull en cada paso temporal. Esto permite capturar las dependencias temporales y actualizar la probabilidad de que ocurra el evento a medida que cambian las características de entrada.

Las Redes Neuronales Recurrentes (RNNs) son un método para aplicar Redes Neuronales Artificiales (ANNs) a datos temporales o secuenciales, como secuencias de texto, series temporales, lenguaje o vídeo. A diferencia de las redes neuronales tradicionales, que asumen que las entradas y salidas son independientes, las RNNs consideran la dependencia temporal entre los datos [40]. Esto se logra gracias a "la memoria" de la red, donde las salidas de las capas anteriores se retroalimentan como entradas en los siguientes pasos de tiempo, permitiendo así que la red tenga en cuenta el contexto previo en sus predicciones.

Las RNNs son comparables a los Modelos de Markov Ocultos en términos de mapeo de la historia de una secuencia al espacio oculto, pudiéndose visualizar como una red neuronal profunda que reutiliza los mismos parámetros en cada capa correspondiente a un paso de tiempo. Gracias a este diseño, cuentan con la capacidad de realizar cálculos complejos y reconocer patrones temporales.

En la Figura 5.2 se puede observar cómo en cada paso, se ingresan las características $x_t$ y el estado oculto $h_{t-1}$, y se generan el estado oculto $h_t$ y el valor predicho $y_t$.

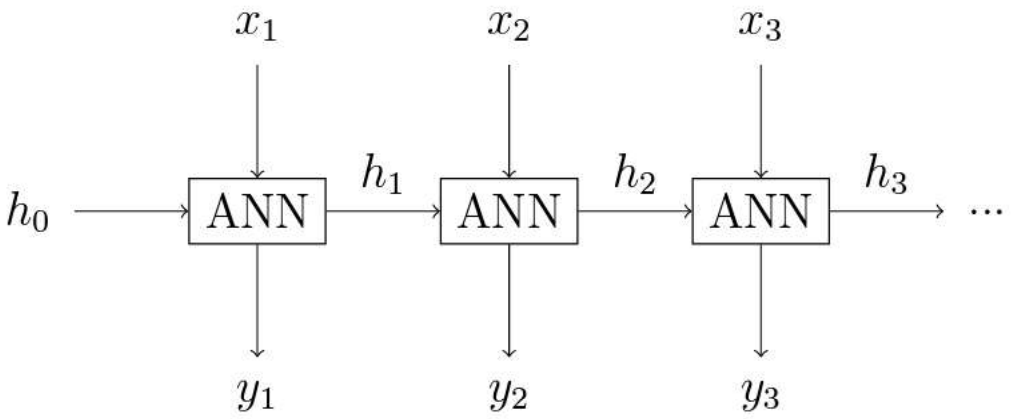


Figura 5.2: Ejemplo gráfico del diseño de RNNs **[41]**

Un aspecto muy importante, sobre el entrenamiento de las RNNs, es que se realiza mediante un proceso conocido como retropropagación a través del tiempo (BPTT), siendo una variante de la retropropagación tradicional adaptada para manejar secuencias. Sin embargo, este enfoque puede presentar algunos problemas técnicos, como el desvanecimiento y la explosión de gradientes, especialmente cuando las redes son muy profundas [40].

Estos problemas se enfocan en el tamaño del gradiente, que es la inclinación de la función de pérdida a lo largo de la curva de error. De modo que, si el gradiente es muy pequeño, este irá disminuyendo hasta llegar a 0, volviéndose insignificante y como consecuencia el algoritmo no estará aprendiendo.

En cuanto a la explosión del gradiente, ocurre cuando este es demasiado grande y como consecuencia el modelo será inestable, haciendo crecer a los pesos del modelo demasiado. Ambos son problemas originados por la naturaleza acumulativa de los errores a lo largo de las secuencias temporales, donde los gradientes pueden volverse tanto excesivamente pequeños como grandes, afectando esto a la estabilidad y capacidad de aprendizaje de la red.

Con el fin de abordar estos desafíos, se han desarrollado variantes de las RNNs, destacando las unidades de memoria a largo plazo (LSTM) y las unidades recurrentes con compuertas (GRU) [40].

- Las LSTM, fueron presentadas como solución al problema del desvanecimiento del gradiente. Se caracterizan por incluir celdas de memoria con puertas que regulan el flujo de información (puerta de entrada, salida y olvido), permitiendo así el mantenimiento de la red y la utilización de información relevante durante de largos intervalos de tiempo.
- Las GRU, simplifican esta estructura con dos puertas: una de actualización y otra de reinicio, facilitando la gestión de la memoria sin comprometer la capacidad de la red para captar dependencias a largo plazo.

Además de las RNNs tradicionales ya mencionadas, existen variantes como las RNNs bidireccionales (BRNNs), que se caracterizan por procesar la información en ambas direcciones, teniendo en cuenta tanto el contexto pasado como el futuro [40]. Las BRNNs son especialmente útiles en aplicaciones donde la comprensión completa de la secuencia es un punto crucial como, por ejemplo, el procesamiento del lenguaje natural, el reconocimiento de voz, la generación de subtítulos para imágenes y la predicción de series temporales [40].

Existen diferentes configuraciones arquitectónicas en las redes neuronales recurrentes (RNN) según el número de entradas y salidas implicadas en la secuencia. Estas variantes se resumen en la Figura 5.3.

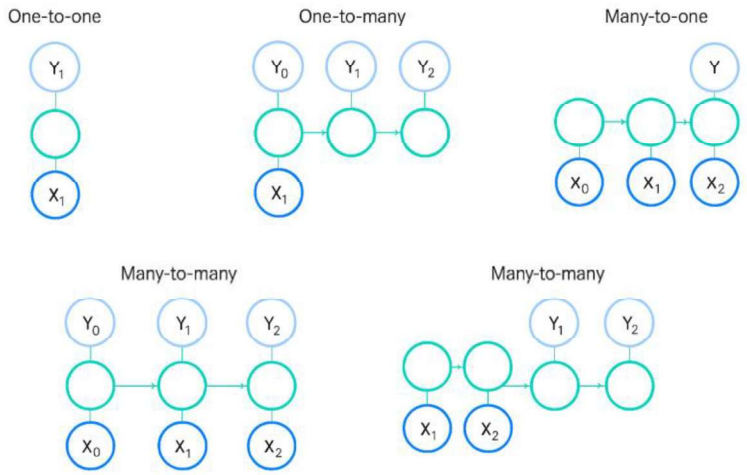


Figura 5.3: Tipos de arquitecturas en redes neuronales recurrentes (RNN) según la correspondencia entre secuencias de entrada y salida **[40]**.

### 5.2.3 Funcionamiento de WTTE-RNN

Con el objetivo de entender el WTTE-RNN y sentar unas bases generales, se describen a continuación sus componentes [41].

**Definición 1 (Tiempos de espera):** Referido a datos o variables aleatorias que modelan el tiempo transcurrido hasta que ocurre un determinado evento (por ejemplo, fallo, muerte, etc.). Se asume que los tiempos de espera $T$ son siempre positivos y sin límite superior, $T \in [0, \infty]$. Los tiempos de espera pueden ser continuos o discretos.

**Definición 2 (Eventos recurrentes):** Dada una línea de tiempo $[0, \infty]$ y puntos marcados en esta, llamamos a estos puntos eventos recurrentes. El tiempo hasta el próximo evento desde alguna posición de la línea de tiempo se llama tiempo hasta el evento o tiempo de espera.

El tiempo entre tales eventos se denomina típicamente tiempos entre llegadas. El tiempo hasta los eventos es fácilmente visualizable con una similitud a la onda de diente de sierra invertida.

La Figura 5.4 ilustra el proceso, mostrando cómo las características de entrada en cada paso, representadas como $x_t$, son transformadas por la RNN para producir valores dinámicos de $\alpha_t$ y $\beta_t$. Estos parámetros ajustan la función de densidad de probabilidad (FDP) de Weibull, proporcionando predicciones adaptativas sobre el tiempo hasta el próximo evento.

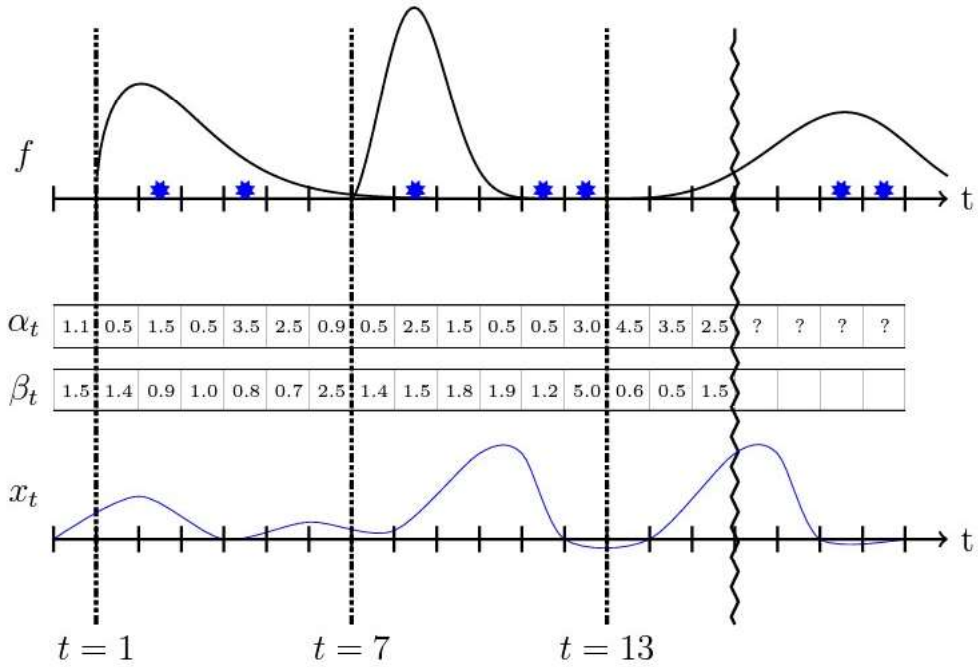


Figura 5.4: Ilustración de la predicción de probabilidad de evento de WTTE-RNN para tres instantes de tiempos diferentes **[41]**.

En conjunto, el modelo WTTE-RNN funciona aprendiendo, a partir de la secuencia de entradas $x_t$, los parámetros dinámicos $\alpha_t$ y $\beta_t$ que definen en cada instante una distribución de Weibull. La Red Neuronal Recurrente (RNN) actúa como mecanismo de memoria temporal, captando patrones y dependencias a lo largo del tiempo, y generando como salida los parámetros que caracterizan la función de densidad de probabilidad del tiempo restante hasta el evento. Estos parámetros se actualizan en cada paso temporal, permitiendo una modelización flexible de los datos de supervivencia, incluso cuando están censurados [41].

En la Figura 5.4 ya introducida anteriormente, se resume este funcionamiento, en la parte inferior se observa la serie temporal de entrada $x_t$ de la que la red extrae en cada instante $t$ una estimación de los parámetros $\alpha_t$ y $\beta_t$ (mostrados en la tabla central). A partir de ellos, se obtiene una función de densidad $f(t)$ (gráfico superior), que representa la probabilidad de ocurrencia del evento en el futuro. Esto permite no solo predecir el momento más probable del evento, sino también estimar su incertidumbre, lo que convierte al modelo en una herramienta poderosa para análisis de supervivencia en series temporales.

# Capítulo 6: Desarrollo del Modelo Propuesto

## 6.1 Introducción

En este capítulo se detalla el proceso de desarrollo e implementación del modelo propuesto para llevar a cabo la predicción del tiempo restante hasta un evento a partir de series temporales. El enfoque se basa en modelar la distribución de tiempos hasta el fallo de maquinaria mediante parámetros de una distribución Weibull [38], permitiendo así obtener, además de una predicción puntual, una estimación de la incertidumbre asociada.

El desarrollo se divide en dos fases: en primer lugar, se ha trabajado con un conjunto de datos sintéticos generados específicamente para validar el correcto funcionamiento de la arquitectura y de las funciones de pérdida personalizadas. Una vez superada esta fase y comprobado su correcto funcionamiento, se ha procedido a adaptar el modelo para trabajar con datos reales del conjunto **CMAPSS** [42] ("Commercial Modular Aero-Propulsion System Simulation"), utilizado de forma habitual en entornos industriales para tareas de mantenimiento predictivo.

Por otra parte, para el desarrollo práctico se ha usado un entorno virtual configurado con Miniconda, permitiendo instalar y controlar de forma precisa las versiones de las bibliotecas necesarias para la ejecución del proyecto, entre las que destacan "PyTorch", "NumPy", "Matplotlib" o "Scikit-learn". Esta estructura de trabajo ha facilitado la reproducibilidad y el mantenimiento del código a lo largo de las distintas fases del proyecto.

## 6.2 Arquitectura propuesta

El modelo propuesto tiene como objetivo estimar los parámetros de una distribución de Weibull [38] (α y β) a partir de ventanas de entrada formadas por secuencias de sensores. Es una elección que permite modelar el tiempo restante hasta un evento futuro (por ejemplo, el fallo de una turbina) no como una predicción puntual, sino como una distribución completa que además proporciona información acerca de la incertidumbre.

Inicialmente, se valoró y se comenzó a utilizar e implementar el modelo WTTE-RNN [41], una red recurrente específica para este tipo de tareas sobre la que se ha profundizado en el Capítulo 5. Sin embargo, con el objetivo de simplificar el diseño y más específicamente, de facilitar la interpretación de los resultados, se ha optó finalmente por utilizar una arquitectura más sencilla y controlable basada en una red densa ("feedforward"). Gracias a este cambio, se ha permitido una mejor integración con las técnicas de explicabilidad empleadas posteriormente y una mayor claridad a la hora de visualizar la influencia de cada entrada en los parámetros de salida.

La arquitectura empleada se compone de las siguientes capas:

- Una **capa de entrada** con un número de nodos igual al tamaño de la ventana de entrada (definido como "window_size") multiplicado por el número de sensores seleccionados.

- Una o más **capas ocultas densas** con funciones de activación "ReLU", que permiten capturar relaciones no lineales entre las variables temporales y la evolución del sistema.

- Una **capa de salida de dos neuronas**, que devuelve los parámetros α y β de la distribución de Weibull estimada para esa ventana. A estos parámetros de salida se les aplica una transformación diferenciable, definida en una clase diferenciada ("WeibullActivation"), que garantiza que sus valores sean positivos, estables y estén dentro de un rango adecuado para representar distribuciones de Weibull.

A lo largo de la experimentación se ha trabajado con distintas profundidades de red y tamaños de capa oculta, buscando un equilibrio entre capacidad predictiva, estabilidad en el entrenamiento y facilidad de interpretación. La selección final se ha basado en los resultados obtenidos tanto con los datos sintéticos usados, como con el conjunto CMAPSS y en la estabilidad de las explicaciones generadas por "Integrated Gradients".

A continuación, en la Figura 6.1 se muestra un fragmento del código en el que se define la clase del modelo propuesto utilizando "PyTorch". Esta clase implementa la arquitectura principal del modelo de supervivencia, permite trabajar tanto con entradas secuenciales como con ventanas aplanadas, y se integra con una activación personalizada que asegura salidas válidas para la distribución.

```
### CREACIÓN DE UNA RED DENSA

class TimeDistributedDenseWeibullNet(nn.Module):
    def __init__(self, input_dim: int, hidden_dims: List[int] = [64, 32], init_alpha: float = 1.0):
        super().__init__()
        dims = [input_dim] + hidden_dims
        layers = []
        for in_dim, out_dim in zip(dims[:-1], dims[1:]):
            layers.append(nn.Linear(in_dim, out_dim))
            layers.append(nn.ReLU())
        self.mlp = nn.Sequential(*layers)
        self.output_layer = nn.Linear(hidden_dims[-1], 2)
        self.weibull_activation = losses.WeibullActivation(init_alpha=init_alpha)

    ##
    def forward(self, x):
        if x.dim() == 2:  # Caso IG: input de ventana aplanado: [B, window_size * n_features]
            x = self.mlp(x)
            x = self.output_layer(x)
            x = self.weibull_activation(x)

            ## MOSTRAR RESULTADOS DE ALPHA Y BETA MEDIA EN EJECUCIONES
            print("α media:", x[:, 0].mean().item(), "β media:", x[:, 1].mean().item())
            return x #forma: [B, 2]

        elif x.dim() == 3:  # Caso normal: batch de secuencias # Caso entrada secuencial: [B, T, F]
            B, T, F = x.shape
            x = x.view(B * T, F)
            x = self.mlp(x)
            x = self.output_layer(x)
            x = self.weibull_activation(x)
            return x.view(B, T, 2) # forma: [B, T, 2]

        else:
            raise ValueError(f"Input tensor has invalid number of dimensions: {x.dim()}")
```

Figura 6.1: Implementación en "PyTorch" de la arquitectura "TimeDistributedDenseWeibullNet"

## 6.3 Preparación del conjunto de datos

El modelo propuesto requiere, como entrada, secuencias temporales que representen el estado progresivo de un sistema físico. En este proyecto, se han empleado dos conjuntos de datos distintos al largo del desarrollo: un "toy dataset" sintético, generado para validar el funcionamiento del modelo, y el conjunto de datos CMAPPS [42] ("Commercial Modular Aero-propulsion System Simulation"), ampliamente utilizado para tareas de mantenimiento predictivo.

### 6.3.1 Formato de entrada: ventanas deslizantes

En ambos conjuntos de datos, las series temporales se transforman mediante un sistema de ventanas deslizantes. Esto consiste en recorrer cada secuencia original y extraer de ella subconjuntos consecutivos de longitud fija (definida por "window_size"), que son usados como ejemplos independientes para el entrenamiento del modelo. De este modo, cada ventana actúa como una fotografía del estado reciente del sistema y permite al modelo predecir la distribución del tiempo restante hasta el evento (tiempo hasta el fallo).

Por ejemplo, para una secuencia temporal $X = [x_1, x_2, \dots, x_T]$, con "window_size" = 100, se generarían $T - 100 + 1$ ventanas:

$$[x_1, \dots, x_{100}], [x_2, \dots, x_{101}], \dots, [x_{T-99}, \dots, x_T]$$

Cada una de estas ventanas se asocia con el tiempo restante hasta el siguiente evento, que permitirá estimar el valor de los parámetros (α, β) que constituye la salida del modelo.

### 6.3.2 Generación y uso del "toy dataset"

Durante las primeras fases del proyecto, se crea un "toy dataset" artificial mediante una función generadora ("get_data") que produce secuencias sintéticas con eventos recurrentes y patrones ruidosos, junto con señales que se repiten de forma regular para facilitar la detección de patrones por el modelo. Haciendo uso de este entorno controlado se puede comprobar si el modelo es capaz de aprender correctamente el patrón temporal subyacente y si las técnicas de explicabilidad son capaces de identificar las zonas relevantes de la secuencia. En resumen, este conjunto de datos fue fundamental para validar:

- El correcto funcionamiento de la arquitectura propuesta.
- La estabilidad numérica de las salidas α y β.
- La integración del método de explicabilidad "Integrated Gradients".

### 6.3.3 Conjunto de Datos CMAPSS

Una vez verificado el modelo con los datos sintéticos del "toy dataset", se ha adaptado la arquitectura para trabajar con el conjunto de datos CMAPSS ("Commercial Modular Aero-Propulsion System Simulation") [42].

Este conjunto de datos es un referente ampliamente utilizado en el ámbito de la predicción del tiempo de fallo (RUL, "Remaining Useful Life") de sistemas aeroespaciales. Este conjunto de datos fue desarrollado por la NASA y se ha convertido en un estándar para el desarrollo y validación de algoritmos de mantenimiento predictivo, al proporcionar registros detallados de

simulaciones realistas del comportamiento de motores bajo distintas condiciones operativas y tipos de fallos.

**Estructura del Conjunto de Datos**

El conjunto de datos CMAPSS está compuesto por varios sub-conjuntos que simulan diferentes condiciones operativas y de fallo de los motores. En cada subconjunto, como se ha mencionado, se incluye información detallada sobre los ciclos de operación del motor, proporcionando una base robusta para la investigación y el desarrollo de algoritmos de pronóstico.

- **Variables de Entrada:**

  a) **ID**: Identificador único para cada motor

  b) **Ciclo**: Registro temporal de cada motor

  c) **Sensores**: El conjunto de datos realiza la lectura de 21 sensores diferentes que miden diversas variables operativas del motor, como la temperatura, presión y velocidad del ventilador, entre otras.

- **Variables Operativas:**

  a) **Condiciones de Vuelo:** Incluye tres condiciones operativas (altitud, velocidad del avión y temperatura ambiente) que afectan el comportamiento del motor.

  b) **Modos de Operación:** Diferentes modos de operación que simulan diversas condiciones de vuelo y carga del motor.

- **Variables de Salida:**

  a) **RUL:** La vida útil remanente en ciclos para cada motor. El objetivo es estimar esta variable, definida como la diferencia entre el número total de ciclos que completa el motor hasta su fallo y el ciclo actual.

**Descripción de los Sensores por Tipo**

A continuación, en la Tabla 6.1 se listan los sensores según su tipo.

| Sensor | Descripción | Tipo |
|---|---|---|
| Sensor 1 | Temperatura del aire de entrada (T2) | Temperatura |
| Sensor 3 | Temperatura de salida del compresor (T24) | Temperatura |
| Sensor 4 | Temperatura de entrada de la turbina (T30) | Temperatura |
| Sensor 5 | Temperatura de salida de la turbina de baja presión (T50) | Temperatura |
| Sensor 17 | Temperatura de la pala de la turbina de alta presión (T48) | Temperatura |
| Sensor 18 | Temperatura de salida de la turbina de alta presión (T49) | Temperatura |
| Sensor 2 | Presión del compresor (P2) | Presión |
| Sensor 8 | Relación de presión del motor (EPR) | Presión |
| Sensor 9 | Presión estática a la salida del compresor de alta presión (Ps30) | Presión |
| Sensor 6 | Velocidad física del ventilador (fan Nf) | Velocidad |
| Sensor 7 | Velocidad física del núcleo (core Nc) | Velocidad |
| Sensor 11 | Velocidad corregida del ventilador (fan Nf_dmd) | Velocidad |
| Sensor 12 | Velocidad corregida del núcleo (core Nc_dmd) | Velocidad |
| Sensor 19 | Velocidad del eje de baja presión | Velocidad |
| Sensor 20 | Velocidad del eje de alta presión | Velocidad |
| Sensor 10 | Relación de flujo de combustible a Ps30 (phi) | Relaciones y Ratios |
| Sensor 13 | Relación de bypass | Relaciones y Ratios |
| Sensor 14 | Relación combustible-aire del quemador (BPR) | Relaciones y Ratios |
| Sensor 21 | Relación combustible-aire integrada del quemador (IRB) | Relaciones y Ratios |
| Sensor 15 | Aire de sangrado (W31) | Flujo |
| Sensor 16 | Demanda de potencia (W32) | Flujo |

Tabla 6.1: Descripción de los sensores por Tipo.

**Sub-Conjuntos dentro CMAPSS**

El conjunto de datos CMAPSS está dividido en cuatro sub-conjuntos principales, diseñados cada uno para evaluar diferentes aspectos y condiciones de fallo [42]:

- **FD001:** Simula un único tipo de fallo en condiciones constantes.
- **FD002:** Incluye el mismo tipo de fallo que FD001, pero bajo condiciones operativas variables (varían las condiciones de vuelo).
- **FD003:** Contiene múltiples tipos de fallos (desgaste en la sección del compresor de baja presión y fallo en la sección de la turbina de alta presión) en condiciones operativas constantes.
- **FD004:** Contiene múltiples tipos de fallos, pero con condiciones operativas variables.

Cada sub-conjunto está diseñado para desafiar diferentes aspectos de los algoritmos de pronóstico, proporcionando una plataforma completa para el desarrollo y evaluación de métodos de estimación RUL.

A continuación, en la Tabla 6.2 se puede observar el número de trayectorias disponibles para cada sub-conjunto, definiendo así su tamaño.

| Sub-conjunto | Entrenamiento | Prueba |
|---|---|---|
| FD001 | 100 | 100 |
| FD002 | 260 | 259 |
| FD003 | 100 | 100 |
| FD004 | 249 | 248 |

Tabla 6.2: Tamaño de los sub-conjuntos de datos de C-MAPSS **[42]**

En este proyecto se ha hecho uso de los cuatro sub-conjuntos para analizar el comportamiento del modelo en distintos escenarios. Se ha implementado una técnica de ventanas deslizantes de tamaño fijo (ajustada en 100 ciclos) para construir la secuencia de entrada. Esta elección permite al modelo aprender dependencias temporales sin exceder los recursos computacionales.

La variable RUL se calcula como la diferencia entre el ciclo final de cada trayectoria y el ciclo actual, con un truncamiento superior aplicado a 150 ciclos para limitar la varianza inicial. Esta decisión técnica busca mejorar la estabilidad del modelo durante el entrenamiento y se muestra en la Figura 6.2, que concatena la evolución del RUL a lo largo de los ciclos de diferentes motores. Además, el preprocesamiento incluye la normalización de los sensores seleccionados, con el fin de estabilizar el entrenamiento y asegurar la comparabilidad entre distintas trayectorias.

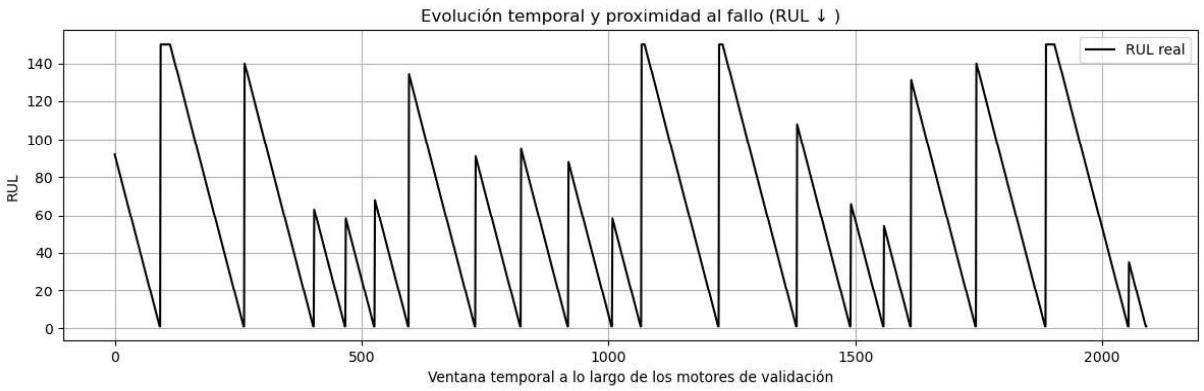


Figura 6.2: Evolución del RUL real a lo largo del tiempo en distintas trayectorias del conjunto CMAPSS (FD001).

Este tratamiento ha permitido adaptar CMAPSS a un entorno de modelado con redes densas y técnicas de explicabilidad “post-hoc”, asegurando una base sólida para los análisis desarrollados en el Capítulo 7 de resultados.

## 6.4 Implementación Técnica

La implementación técnica del modelo propuesto se ha llevado a cabo utilizando el lenguaje de programación Python [43], apoyado en el ecosistema de librerías científicas orientadas a tareas de “machine learning” y análisis de datos. Para garantizar la correcta gestión de dependencias y mantener un entorno de trabajo limpio y reproducible, se ha utilizado

Miniconda, donde se ha creado un entorno virtual específico para este proyecto. Dentro de dicho entorno se han instalado todas las librerías necesarias, incluyendo "PyTorch" [44], "NumPy" [45], "Matplotlib" [46], "Scikit-learn" [47] y otras de utilidad para el preprocesamiento, visualización y entrenamiento de modelos.

El desarrollo del modelo se ha realizado apoyado en "notebooks" de Jupyter que ofrecen una estructura modular y progresiva del código. Específicamente, se han utilizado dos "notebooks" principales:

- Uno dedicado al entrenamiento y validación con los datos sintéticos del "toy dataset", útil para comprobar el correcto funcionamiento del modelo y depurar errores en un entorno controlado.
- El segundo "notebook" está orientado a la aplicación real con el conjunto de datos CMAPSS [42], en el que se integran los pasos de preprocesamiento, normalización, creación de ventanas de entrada, entrenamiento sobre los cuatro subconjuntos (FD001-FD004) y análisis de los resultados.

La arquitectura del modelo, ya descrita en la sección 6.2, se organiza en torno a la clase "TimeDistributedDenseWeibullNet", que permite procesar secuencias y devolver como salida los parámetros de la distribución de Weibull. Esta salida, como ya se ha comentado previamente, se transforma mediante una capa personalizada definida, en la clase "WeibullActivation", cuya finalidad es garantizar la positividad de los parámetros mediante una transformación logarítmica suave, además de aplicar "clamping", que consiste en restringir los valores de salida dentro de un rango mínimo y máximo predeterminado, evitando así que los parámetros tomen valores extremos que podrían provocar errores de cálculo o inestabilidad durante el entrenamiento, para mejorar la estabilidad numérica.

Para entrenar el modelo, se hace uso de una función de pérdida adaptada a la distribución de Weibull, incluida en un archivo auxiliar llamado "losses.py". Esta función calcula la log-verosimilitud negativa como métrica principal para optimizar el ajuste de los parámetros α y β.

Durante el desarrollo del código se han añadido rutinas para representar la función de supervivencia, visualizar las distribuciones generadas y aplicar técnicas de explicabilidad, permitiendo así una comprensión más profunda del comportamiento del modelo.

## 6.5 Técnica de Explicabilidad Utilizada: "Integrated Gradients"

A la hora de analizar la influencia de las entradas en las predicciones del modelo de supervivencia, se ha optado por utilizar la técnica de explicabilidad **"Integrated Gradients" (IG)** [5]. Esta técnica pertenece al conjunto de métodos "post-hoc", es decir, se aplica una vez el modelo ha sido entrenado, sin alterar su estructura interna ni requerir modificaciones en la arquitectura base.

La elección de IG se ha basado en su solidez teórica, su compatibilidad con redes neuronales profundas y su capacidad para generar atribuciones estables y consistentes a lo largo del tiempo, aspecto fundamental en tareas basadas en series temporales.

La técnica se ha implementado sobre las salidas del modelo, específicamente sobre los dos parámetros de la distribución de Weibull (α y β) [38], lo que ha permitido evaluar por separado la influencia de cada característica temporal en la predicción de la esperanza y la dispersión. Para ello, se ha definido una entrada base (entrada de referencia calculada como la media

simple de los datos) y se calculan los gradientes integrados a lo largo de trayectorias lineales entre esta entrada base y la entrada real, acumulando la información de sensibilidad del modelo.

Este enfoque permite identificar, con alta sensibilidad y granularidad, qué instantes y qué sensores contribuyen más a la predicción, generando representaciones gráficas que serán analizadas en el Capítulo 7 de resultados. Como ya se ha comentado, la implementación se ha realizado utilizando herramientas de "PyTorch", desarrollando funciones específicas para aplicar IG de forma diferenciada a α y β, obteniendo así una visión dual de la explicabilidad: tanto del tiempo estimado hasta el evento, como de la incertidumbre asociada a dicha predicción.

En definitiva, el uso de IG ha demostrado ser adecuado para usarse en este proyecto, ya que otras técnicas como "SHAP" [24] o "LIME" [27] requieren un mayor esfuerzo computacional. En cambio, IG se integra de forma natural en el flujo del modelo y permite obtener resultados interpretables incluso en arquitecturas densas como la utilizada en este proyecto.

A continuación, la Figura 6.3 muestra un bloque de código implementado para calcular las atribuciones con "Integrated Gradients" de forma diferenciada para los parámetros de α y β, utilizando como entrada una secuencia temporal y como base una media simple. Se trata de una separación que nos permite analizar la influencia de cada característica sobre ambas dimensiones (esperanza y dispersión) de la predicción de forma independiente.

```python
# Configuración (calcula la media)
n_samples = len(X_val_flat)  # Número de muestras a analizar
importance_alpha = []
importance_beta = []

for i in range(n_samples):
    input_sample = X_val_flat[i:i+1]

    # --- IG para α ---
    def forward_alpha(x):
        return model(x)[:, 0]

    baseline = input_sample.mean(dim=1, keepdim=True).expand_as(input_sample)
    ig = IntegratedGradients(forward_alpha)
    attr_alpha, _ = ig.attribute(input_sample.requires_grad_(), baselines=baseline, return_convergence_delta=True)
    importance_alpha.append(attr_alpha.abs().sum().item())

    # --- IG para β ---
    def forward_beta(x):
        return model(x)[:, 1]

    ig = IntegratedGradients(forward_beta)
    attr_beta, _ = ig.attribute(input_sample.requires_grad_(), baselines=baseline, return_convergence_delta=True)
    importance_beta.append(attr_beta.abs().sum().item())
```

Figura 6.3: Implementación de la técnica "Integrated Gradients" para calcular las atribuciones individualizadas de α y β del modelo de supervivencia.

# Capítulo 7: Resultados

## 7.1 Introducción

A lo largo de este capítulo se presentan los resultados obtenidos tras aplicar el modelo propuesto sobre el conjunto de datos CMAPSS. Una vez verificado su correcto funcionamiento sobre datos sintéticos, se ha entrenado y evaluado el modelo con los cuatro subconjuntos reales (FD001 a FD004), permitiendo analizar su comportamiento en distintos escenarios de fallo y condiciones operativas.

Los resultados se van a estructurar en varias secciones: en primer lugar, se presenta el rendimiento general del modelo para cada subconjunto, evaluando su capacidad para ajustar correctamente los parámetros α y β de la distribución de Weibull [38]. Posteriormente, se analizan ejemplos representativos mediante visualizaciones que permiten interpretar la evolución temporal de la función de supervivencia generada. Finalmente, se introducen los resultados obtenidos mediante la técnica de explicabilidad "Integrated Gradients", con el objetivo de identificar las variables y momentos temporales más relevantes en la predicción de RUL.

El análisis llevado a cabo se centra tanto en la calidad predictiva del modelo como en su interpretabilidad, buscando no solo estimar correctamente el tiempo restante hasta el fallo, sino también comprender qué factores influyen más en dicha estimación. Se trata de un factor especialmente relevante en contextos industriales, donde la trazabilidad y justificación de las predicciones adquieren un valor añadido.

## 7.2 Evaluación por subconjuntos

A continuación, se presentan los resultados obtenidos por el modelo propuesto al ser aplicado sobre los cuatro subconjuntos del conjunto de datos CMAPSS (FD001-FD004). Cada subconjunto como ya se ha introducido en el Capítulo 6, representa un escenario distinto de fallo o condición operativa, lo que permite evaluar la robustez, generalización y utilidad del modelo en contexto variados.

Previamente al entrenamiento, se ha realizado una selección específica de sensores para cada subconjunto, descartando aquellos con baja variabilidad o alta redundancia, siguiendo las recomendaciones propuestas por Saxena [48] en el informe técnico original de la NASA. Esta selección de sensores busca mantener aquellos que resultan más informativos para la predicción de fallo y mejorar la eficiencia del modelo.

### 7.2.1 Especificaciones del modelo

El modelo empleado en todos los experimentos incluidos en este capítulo es una red neuronal densa secuencial. Aplicada sobre ventanas de datos temporales fijas, y diseñada para predecir los parámetros α (escala) y β (forma) de una distribución de Weibull a partir de series temporales de sensores. Se ha utilizado exactamente la misma arquitectura en los cuatro subconjuntos (FD001-FD004), modificando únicamente el grupo de sensores y los datos de entrada correspondiente.

La arquitectura se define como sigue:

- **Capa de entrada:** tensores de forma (100, n), donde n es el número de sensores seleccionados para cada subconjunto. Cda secuencia representa 100 pasos temporales consecutivos.

- **Capa densa oculta ("TimeDistributed"):** 64 neuronas con activación ReLU, aplicada sobre cada timestep de forma independiente mediante "TimeDistributed".

- **Capa de salida:** 2 neuronas (α y β), sin activación directa.

- **Activación final personalizada:** se usa "WeibullActivation", que transforma las salidas crudas en parámetros válidos de la distribución.
    - α: transformado mediante una exponencial suave y "clamping" (restringir los valores de salida dentro de un rango mínimo y máximo predeterminado) para evitar valores extremos.
    - B: transformado mediante una sigmoide escalado a un máximo de 5.0 (valor ajustado como hiperparámetro)

Para el entrenamiento se utilizó:

- **Función de pérdida:** "Weibull NLL Loss" implementada específicamente para estimar parámetros de Weibull en presencia de censura.

- **Optimizador:** Adam.

- **"Batch size":** 32.

- **Número de épocas:** 50.

- **"Split" de entrenamiento/validación:** 80% / 20%, estratificado por unidad ("unit_nr"), aplicado con "GroupShuffleSplit".

Esta configuración se ha mantenido en los cuatro subconjuntos para garantizar una evaluación homogénea del comportamiento del modelo en distintos escenarios de fallo y condiciones operativas.

### 7.2.2 Subconjunto FD001

El subconjunto FD001 representa el caso más simple dentro del conjunto de datos CMAPSS, al simular un único tipo de fallo bajo condiciones de operación constantes. Esto lo convierte en el escenario ideal para evaluar el comportamiento del modelo en datos reales sin la interferencia de variabilidad externa, tras haber sido inicialmente validad su funcionamiento básico en el entorno con el conjunto de datos sintéticos generados y conocido como "toy dataset".

#### Selección de sensores

Para este subconjunto, siguiendo la estrategia de selección de sensores comentada previamente, el modelo ha trabajado con las señales procedentes de los siguientes sensores: s_2, s_3, s_4, s_7, s_8, s_11, s_12, s_15, s_17, s_20 y s_21.

**Entrenamiento del modelo**

Durante el entrenamiento del modelo, se ha observado una evolución progresiva en la reducción de la función de pérdida. En la Figura 7.1 se representa la evolución de la pérdida de entrenamiento y validación a lo largo de 50 épocas.

A pesar de la presencia de alguna oscilación puntual (especialmente en la validación) la tendencia general es decreciente, indicando así una adecuada capacidad de ajuste y la ausencia de sobreajuste significativo.

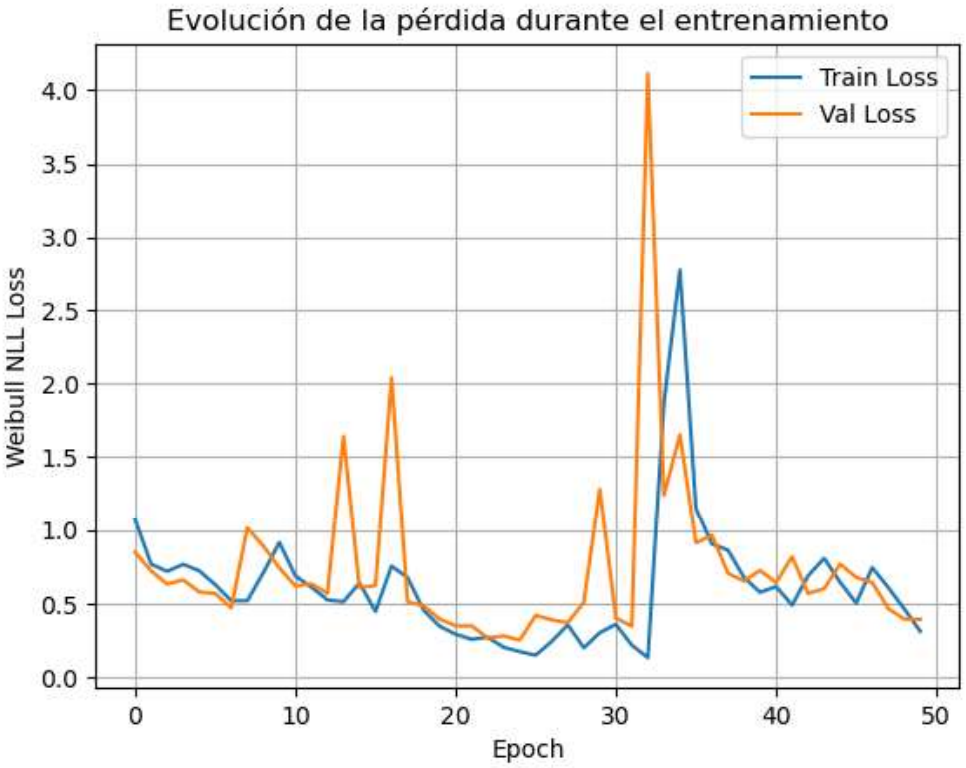


Figura 7.1: Evolución de la pérdida de entrenamiento y validación a lo largo de 50 épocas (FD001).

**Predicción de parámetros Weibull**

La arquitectura usada en el proyecto genera como salida los parámetros **α** (escala) **y β** (forma) para cada ventana temporal de entrada. De este modo, en la Figura 7.2 se muestra una comparación entre el parámetro **α** predicho y el tiempo hasta el fallo real (TTE) en el conjunto de validación. Esta representación permite observar una concordancia clara entre ambas curvas, con una buena capacidad del modelo para aproximar la evolución del fallo inminente.

Además, en la parte inferior de la misma Figura 7.2 se representa la evolución del parámetro **β** (forma)**,** que define la forma de la distribución de Weibull. Se trata de un parámetro que representa un patrón estadístico de fallo. En todo momento se mantiene mayor que 1, lo cual indica una tasa de fallo creciente (típico en mecanismos de deterioro progresivo), se observa una dinámica cíclica en la que **β** tiende a disminuir conforme se aproxima el fallo. Es un comportamiento que sugiere que el modelo es capaz de ajustar el patrón temporal del deterioro de cada unidad.

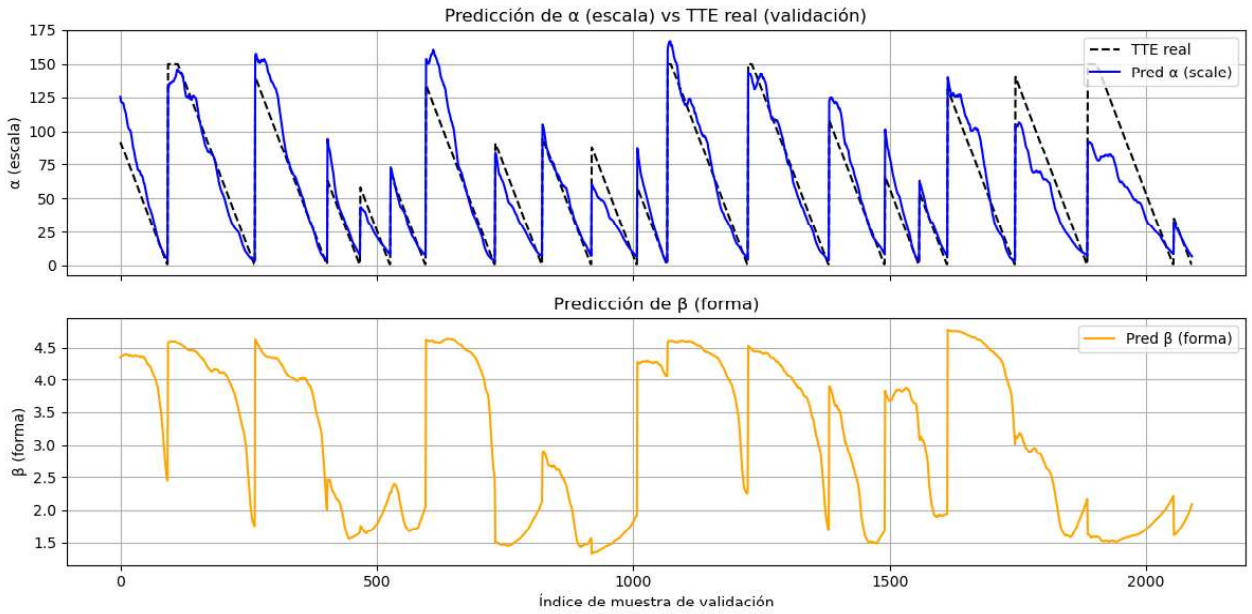


Figura 7.2: Predicción del parámetro α (arriba) y β (abajo) en el conjunto de validación (FD001).

### Evaluación cuantitativa

El rendimiento del modelo debe medirse a nivel numérico para tener la seguridad de que funciona de una manera satisfactoria y cumple con unos estándares razonables. A continuación, en la Tabla 7.1 de muestran las métricas obtenidas para el conjunto de datos FD001.

| Métrica | Valor |
|---|---|
| α media | 58.97 |
| β media | 3.10 |
| MSE (“Mean Squared Error”) entre α predicho y TTE real | 271.71 |
| RMSE (“Root Mean Squared Error”) sobre RUL estimado | 17.22 |
| MAE (“Mean Absolute Error”) sobre RUL estimado | 11.47 |

Tabla 7.1: Evaluación cuantitativa del modelo (FD001).

Desde el punto de vista cuantitativo, se observa una media de valores predichos para α de aproximadamente 59 ciclos, con un β medio de 3.1, lo que implica que hay una distribución moderadamente concentrada en torno al valor esperado. Por otra parte, el error cuadrático medio (MSE) entre Alpha predicho y el TTE real obtenido fue 271.71, que resulta aceptable dada la complejidad del problema y la naturaleza de las secuencias.

Además, el modelo obtuvo un RMSE de 17.22 ciclos y un MAE de 11.47 ciclos, valores que reflejan una capacidad de predicción razonablemente buena, especialmente considerando que no se realiza ningún ajuste específico por motor ni se incorpora información externa. Por tanto, estos resultados sugieren que la red es capaz de capturar patrones de degradación de forma generalizable, sin sobreajuste en los datos de entrenamiento.

### Atribuciones por sensor

En la Figura 7.4 se visualiza el análisis de importancia por sensor a lo largo del tiempo para la unidad 62. Esta unidad ha sido seleccionada como ejemplo por dos motivos principales: por

un lado, presenta una longitud de secuencia superior al tamaño de la ventana definido (100 ciclos), permitiendo analizar una progresión temporal completa; y, por otro lado, sus resultados de inferencia son representativos y reflejan comportamientos relevantes del modelo.

El gráfico muestra para cada uno de los sensores utilizados en FD001, tres columnas: la señal original (izquierda), la atribución IG(α) (centro) y la atribución IG(β) (derecha). Estas curvas permiten evaluar la influencia temporal de cada sensor en la predicción de los parámetros de la distribución de Weibull. En el caso de IG, valores negativos de mayor magnitud indican una mayor influencia hacia la predicción de fallo próximo, ya que reducen el RUL con respecto a la entrada media (el "baseline").

En la propia figura pueden identificarse los sensores s_4, s_7, s_12, s_15 y s_21 como los más relevantes para la predicción del parámetro de escala o tiempo hasta el fallo (α). Son sensores que muestran curvas de IG(α) con una mayor magnitud y estructura temporal, características que indican una alta sensibilidad del modelo ante variaciones.

- S_2 presenta picos positivos al final de la ventana, señalando una respuesta clara ante cambios en su señal cuando el fallo es inminente.
- S_4 muestra una curva decreciente progresiva, indicando contribución acumulativa cuando se aproxima al fallo.
- S_7 destaca por la fuerte atribución negativa del último tramo, alertando al modelo de un deterioro acelerado. Se trata de un sensor que alcanza una importancia negativa de mayor magnitud, indicando su relevancia a la predicción.
- S_12, aunque con una pendiente más sueva, mantiene una atribución consistente que respalda su papel como variable predictiva.
- S_15 y S_21 presentan caídas pronunciadas y sostenidas en las últimas etapas del ciclo, pudiendo actuar como marcadores del deterioro. Su contribución no es tan prominente como la de s_7, pero su comportamiento refuerza su utilidad en fases finales.

Por otra parte, la atribución de IG(β) modela la forma de la distribución Weibull, siendo indicativo para entender el modo de deterioro más que su inminencia ante el fallo, por lo que puede resultar menos estructurada y algo más difusa. Se trata de una diferencia coherente, ya que captura el tipo de progresión del fallo más que su proximidad inmediata. Las gráficas muestran como sensores como el s_17 presentan una tendencia clara y suave en la progresión del modo de deterioro (forma de la distribución Weibull).

En conjunto, esta representación gráfica ofrece una forma visual y cuantitativa de interpretar que sensores y en qué medida guían realmente la predicción del modelo, además de en qué momentos lo hacen. Se trata de una capacidad de explicabilidad especialmente útil para validar el comportamiento del sistema, identificar sensores clave en cada unidad y proporcionar confianza en la toma de decisiones basadas en el modelo.

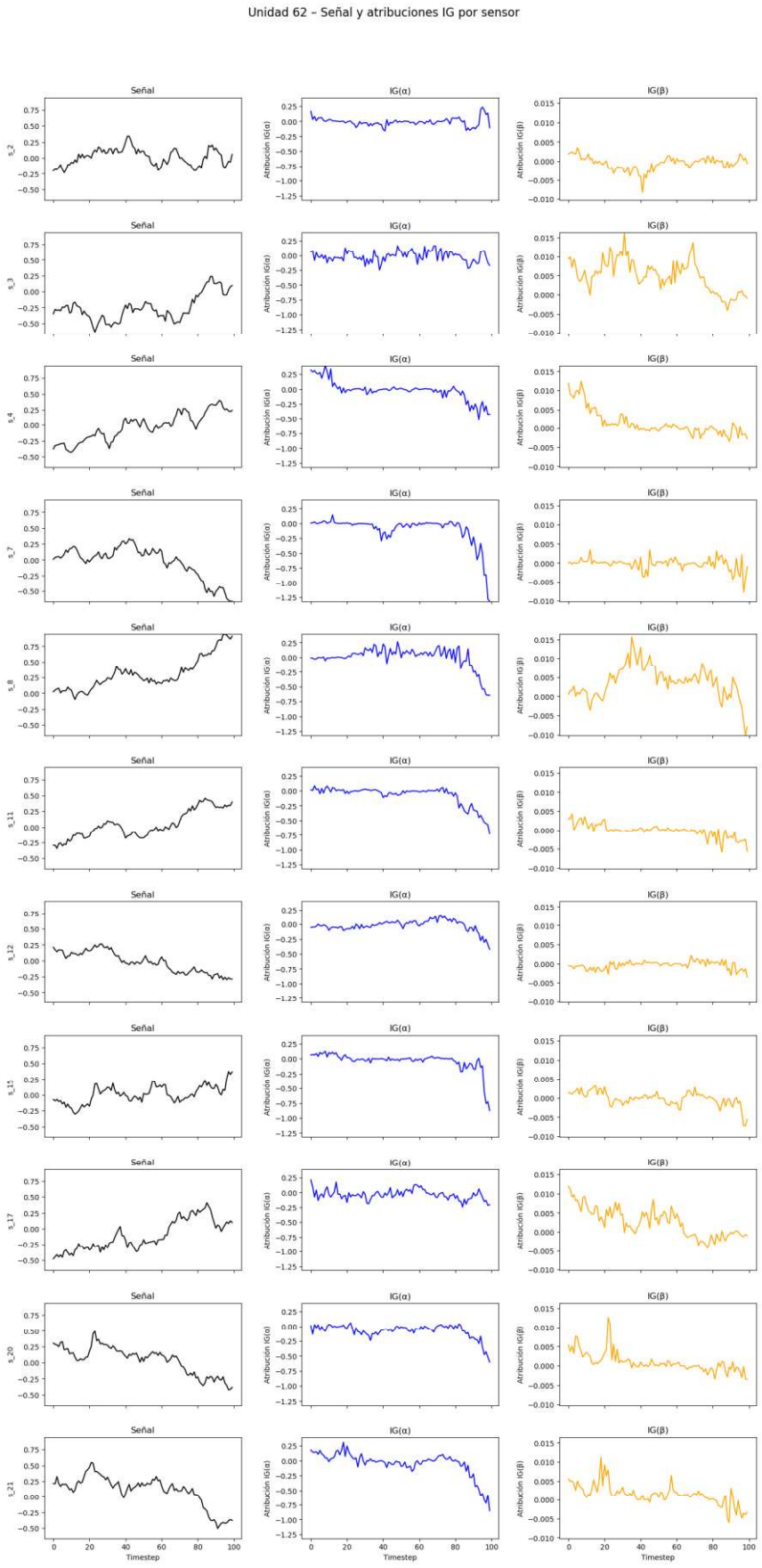


Figura 7.3: Señal e importancia IG por sensor en validación (FD001).

### 7.2.3 Subconjunto FD002

El subconjunto FD002 introduce una nueva dimensión de complejidad frente al caso anterior (FD001), manteniendo un único tipo de fallo, pero bajo múltiples condiciones operacionales. Esto implica que, aunque la tipología del deterioro es homogénea, los motores evolucionan en distintos entornos, exigiendo al modelo una mayor capacidad de generalización.

Siguiendo el mismo criterio ya comentado para FD001, los sensores utilizados para este subconjunto son: s_1, s_2, s_3, s_4, s_7, s_8, s_11, s_12, s_13, s_14, s_15.

**Evolución del entrenamiento**

La Figura 7.5 muestra el comportamiento de la función de pérdida con las mismas condiciones que en el caso anterior (50 épocas, ventana de 100 ciclos). En este conjunto se observa una mayor oscilación en la pérdida de validación, reflejando la heterogeneidad en las condiciones de los motores. Aun así, la tendencia global es claramente descendente, consiguiéndose una convergencia adecuada al final del proceso.

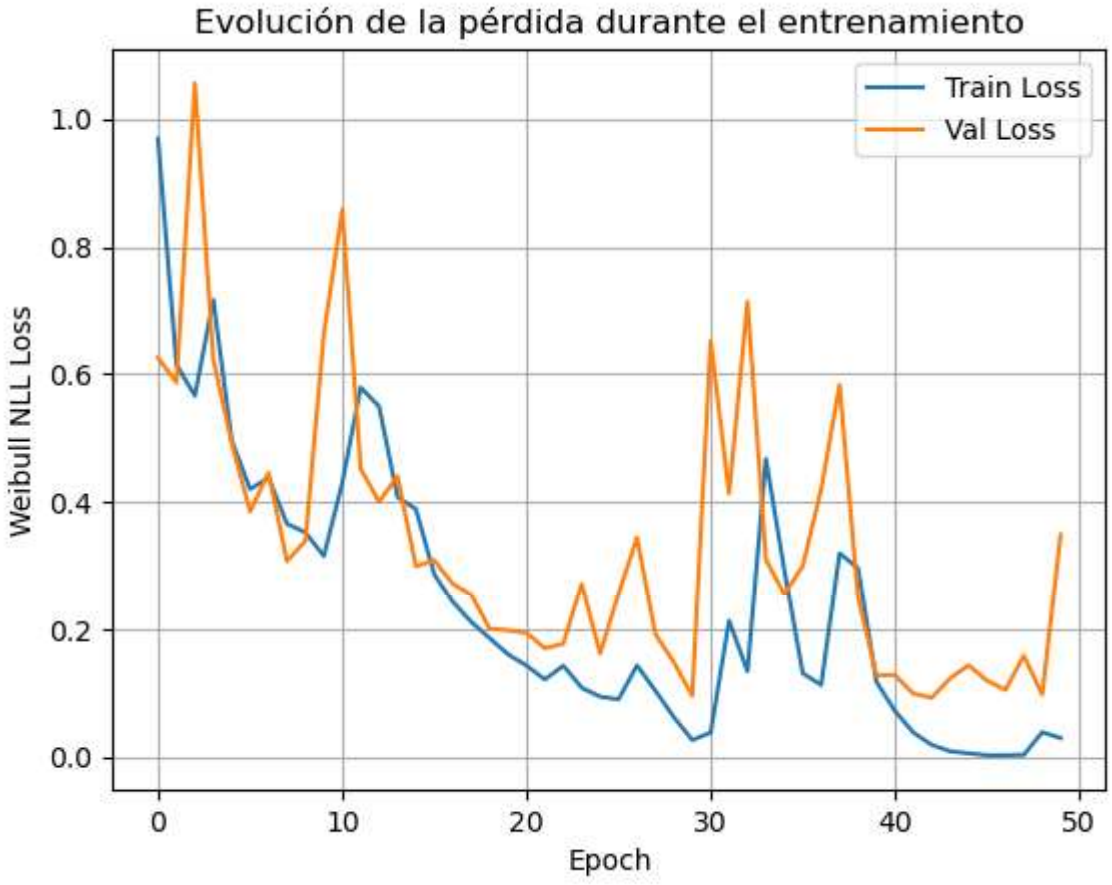


Figura 7.4: Evolución de la pérdida de entrenamiento y validación a lo largo de 50 épocas (FD002).

**Predicción de parámetros Weibull**

En la Figura 7.6 se representan las predicciones del modelo para los parámetros de la distribución de Weibull. Se observa una buena aproximación de α (escala) al TTE real en la mayoría de las unidades, aunque con una mayor dispersión que en FD001. La predicción de β (forma) muestra un patrón de mantenimiento cercano a 5, con caídas bruscas que pueden asociarse a ciertos regímenes de degradación concretos.

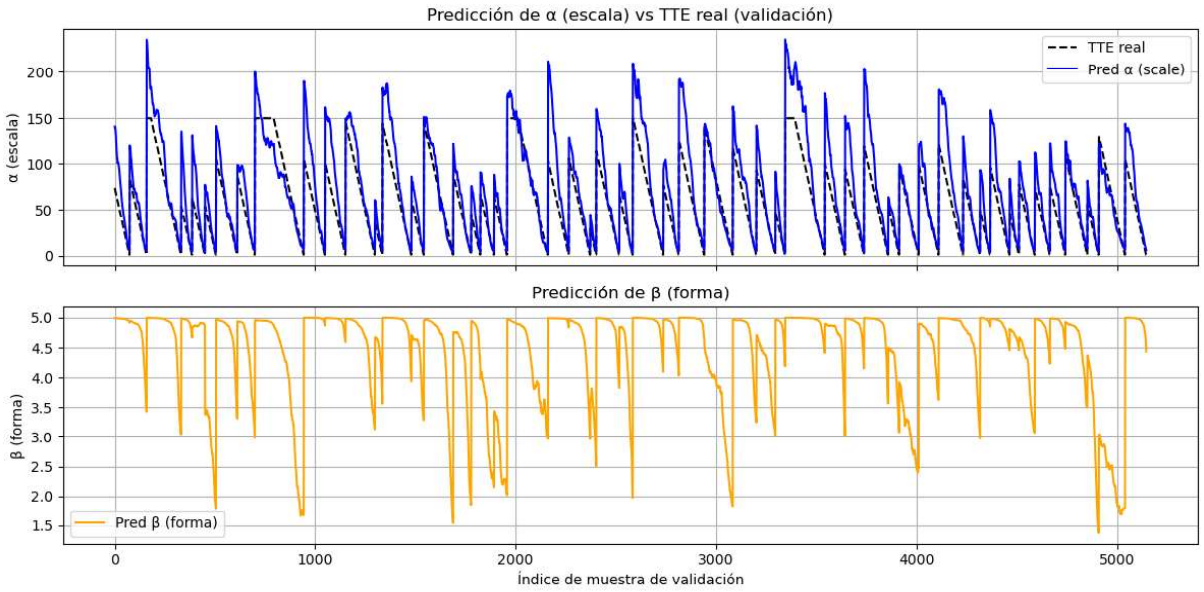


Figura 7.5: Predicción del parámetro α (arriba) y β (abajo) en el conjunto de validación (FD002).

### Evaluación cuantitativa

En la Tabla 7.2 pueden observarse los resultados cuantitativos obtenidos para FD002. Reflejan un incremento esperable del error respecto a FD001, atribuible a la mayor complejidad operativa del conjunto. No obstante, se encuentran en un rango razonable, considerando que el modelo no incorpora información contextual adicional sobre las condiciones de operación.

| Métrica | Valor |
|---|---|
| α media | 73.49 |
| β media | 4.46 |
| MSE (“Mean Squared Error”) entre α predicho y TTE real | 746.55 |
| RMSE (“Root Mean Squared Error”) sobre RUL estimado | 21.69 |
| MAE (“Mean Absolute Error”) sobre RUL estimado | 15.07 |

Tabla 7.2: Evaluación cuantitativa del modelo (FD002).

### Atribuciones por sensor

La Figura 7.8 muestra las atribuciones por sensor en la unidad 62, elegida con los mismos criterios que para el subconjunto FD001. El patrón observado es coherente con el análisis hecho en el apartado anterior (subconjunto FD001), concluyendo con que los sensores s_2, s_3, s_4 y s_7 parecen tener mayor influencia en la predicción de α, con una caída bien definida concentrada en los tramos finales del ciclo, lo que indica una alta sensibilidad del modelo a su evolución temporal en momentos críticos. Este patrón refuerza el papel de ciertos sensores como variables relevantes para anticipar el fallo.

Además, los sensores s_11 y s_15 muestran una tendencia descendente progresiva IG(α), de nuevo en las últimas etapas, lo que puede indicar que el modelo también se apoya en estos para la predicción de α, dándoles importancia junto a los previamente mencionado

En conjunto, los resultados obtenidos del subconjunto FD002 refuerzan la robustez del modelo ante entornos más variados, y demuestran la capacidad de este para adaptarse a escenarios reales con múltiples configuraciones operativas.

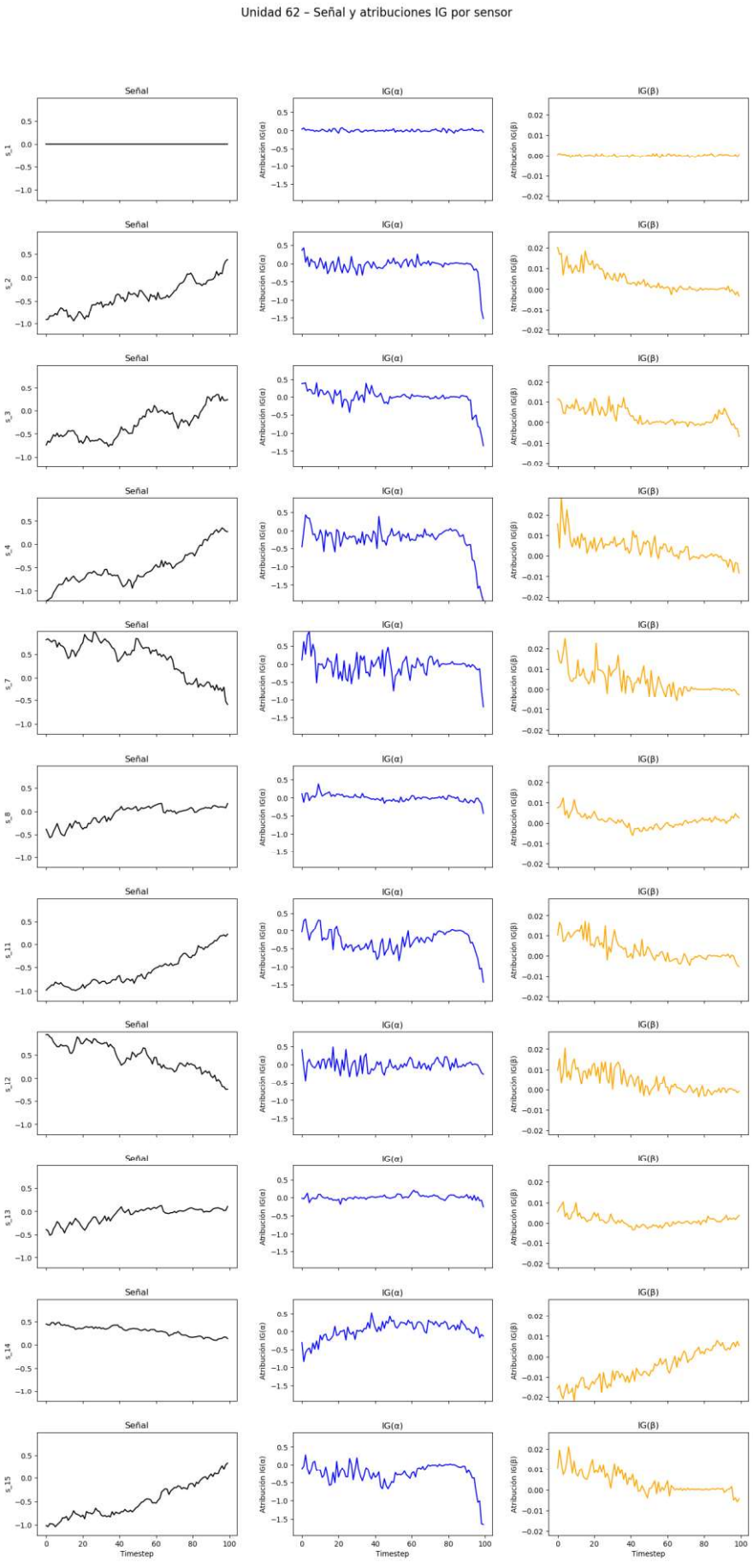


Figura 7.6: Señal e importancia IG por sensor en validación (FD002).

### 7.2.4 Subconjunto FD003

El subconjunto FD003 incorpora variabilidad en las condiciones de funcionamiento del motor, aun manteniendo un único tipo de fallo. Esta combinación representa un entorno más complejo que FD001 y FD002, introduciendo mayor ruido estructural y obligando al modelo a aprender patrones más robustos y capaces de generalizar más allá de un contexto uniforme.

Siguiendo el mismo criterio que en el resto de los casos, los sensores utilizados para este subconjunto son: s_1, s_2, s_3, s_4, s_7, s_8, s_9, s_11, s_12, s_15, s_17.

**Evolución de la pérdida**

En la Figura 7.9 se muestra la evolución de la función de perdida durante el entrenamiento. Al igual que los subconjuntos anteriores, se aprecia una tendencia decreciente en la "Weibull NLL Loss" con cierta oscilación en la validación, que es esperable dada la mayor variabilidad del subconjunto.

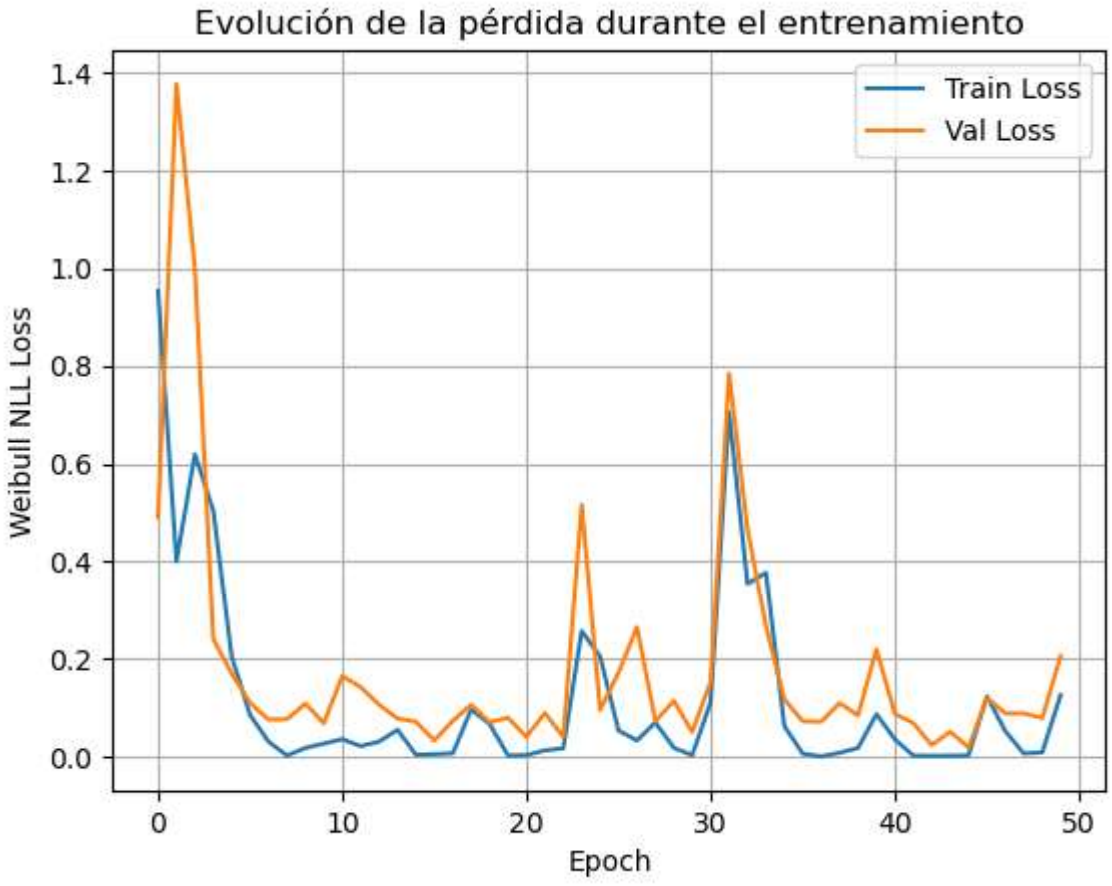


Figura 7.7: Evolución de la pérdida de entrenamiento y validación a lo largo de 50 épocas (FD003).

**Predicción de parámetros de Weibull**

En la Figura 7.10 se presenta las curvas de predicción para los parámetros α (escala) y β (forma). Al igual que en los ejemplos pasados, la predicción de α muestra una buena concordancia con el TTE real, a pesar contar con valores superiores al valor de "threshold" ajustado en el código ("threshold" =150) en este caso. La predicción de β sigue el mismo patrón, valores generalmente altos con oscilaciones que reflejan diferencias en el tipo de deterioro según la unidad.

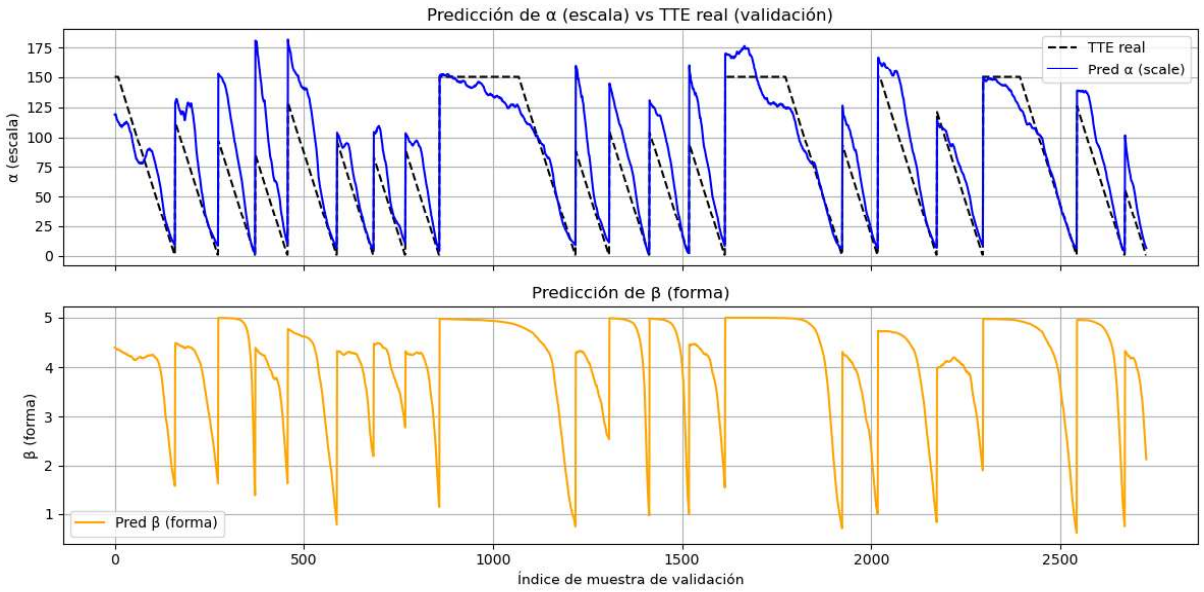


Figura 7.8: Predicción del parámetro α (arriba) y β (abajo) en el conjunto de validación (FD003).

### Evolución cuantitativa

La Tabla 7.3 muestra las métricas obtenidas para el subconjunto FD003, siendo unos resultados muy similares a los obtenidos para FD002, por lo que el modelo mantiene la capacidad de generalización robusta.

| Métrica | Valor |
|---|---|
| α media | 87.55 |
| β media | 4.11 |
| MSE (“Mean Squared Error”) entre α predicho y TTE real | 632.32 |
| RMSE (“Root Mean Squared Error”) sobre RUL estimado | 21.78 |
| MAE (“Mean Absolute Error”) sobre RUL estimado | 16.86 |

Tabla 7.3: Evaluación cuantitativa del modelo (FD003).

### Atribuciones por sensor

En la Figura 7.12 se muestran las atribuciones por sensor, de nuevo para la unidad 62. A pesar de tratarse de un entorno más variable, se observan patrones coherentes. Algunos sensores, como s_7 y s_12, vuelven a destacar como relevantes para la predicción del parámetro α, reforzando así su papel como indicadores robustos del deterioro en distintos escenarios. Estos sensores presentan curvas de IG(α) con valores negativos de mayor magnitud, menor dispersión y una evolución coherente con el avance del fallo, lo que sugiere una contribución estable a la predicción. Además, su presencia reiterada entre los subconjuntos confirma su relevancia transversal en el modelo.

Por otra parte, la atribución para β de nuevo sigue una distribución más difusa, en línea con su papel en la modelización del tipo de deterioro en lugar de su inminencia.

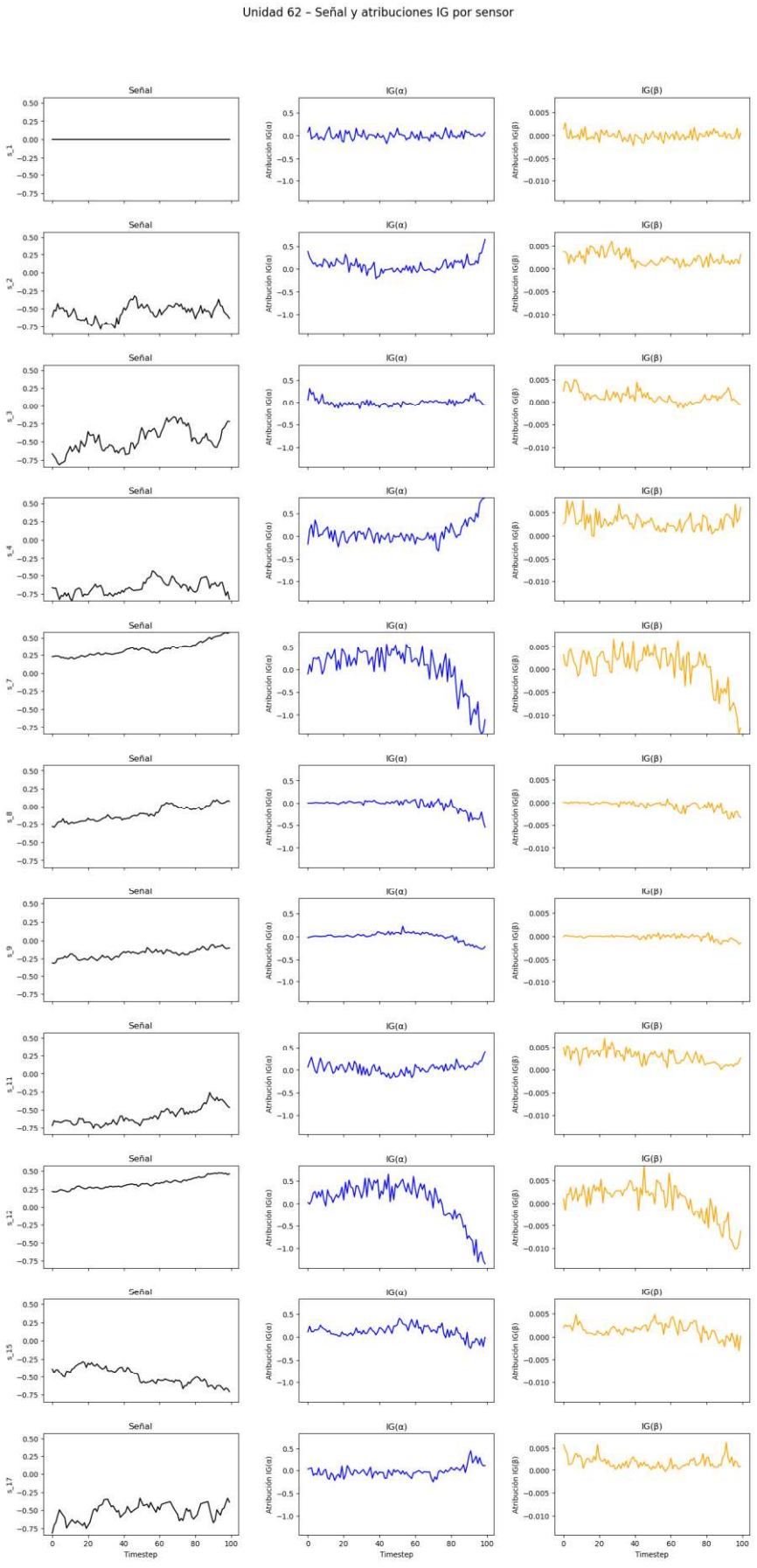


Figura 7.9: Señal e importancia IG por sensor en validación (FD003).

### 7.2.5 Subconjunto FD004

El subconjunto FD004 presenta el escenario más complejo del "dataset" combinando múltiples modos de fallo con condiciones variables de operación. Esta combinación desafía la capacidad de generalización del modelo y permite comprobar su comportamiento bajo la máxima variabilidad estructural. Sensores utilizados: s_1, s_2, s_3, s_4, s_6, s_7, s_8, s_11, s_12, s_15, s_17.

**Evolución de la pérdida**

En la Figura 7.13 se muestra la evolución de la función de pérdida durante el entrenamiento. A diferencia de los subconjuntos anteriores, se observa una mayor oscilación y un valor significativamente superior en la pérdida de validación, probablemente debido al ruido adicional derivado de los distintos modos de fallos. A pesar de ello, el modelo alcanza a converger en entrenamiento, estabilizándose tras las primeras 15-50 épocas. Sin embargo, la marcada diferencia entre la pérdida de entrenamiento y la de validación, sugiere la presencia de sobreajuste, ya que el modelo aprende patrones de entrenamiento que no generaliza adecuadamente al conjunto de validación.

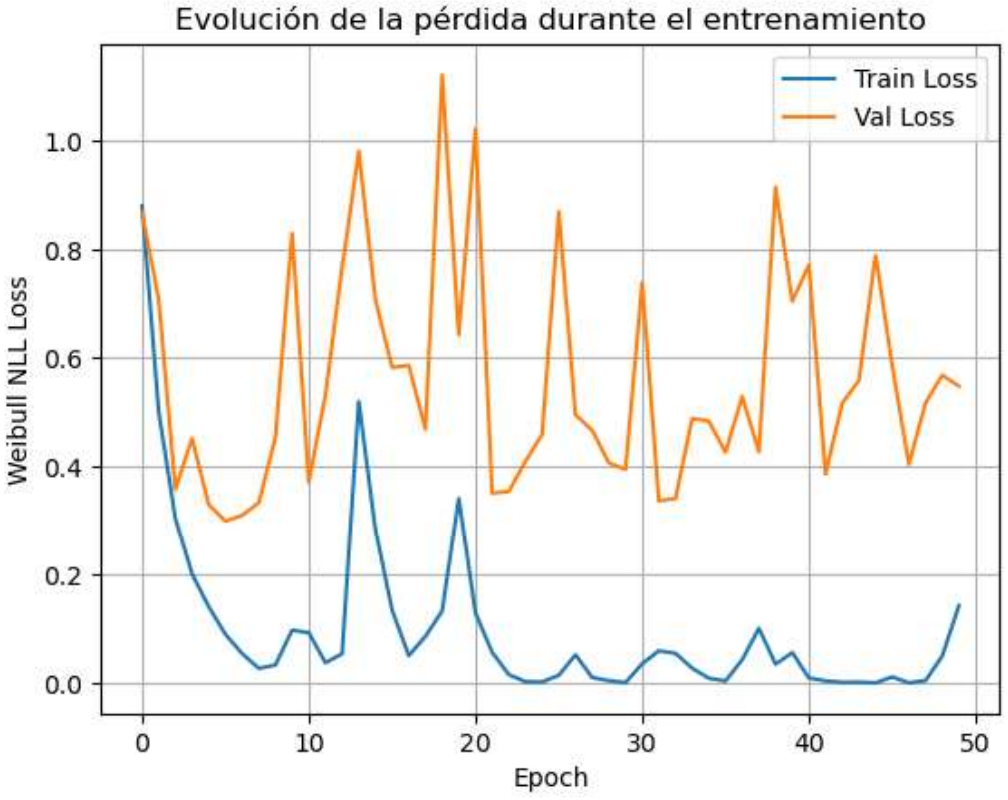


Figura 7.10: Evolución de la pérdida de entrenamiento y validación a lo largo de 50 épocas (FD004).

**Resultados generales**

Las predicciones de los parámetros de Weibull (α y β) y las curvas de importancia acumulada (IG) muestran un comportamiento coherente y similar al observado en los ejemplos previos, con una buena correlación entre α y el TTE real, así como una buena sensibilidad en los tramos de fallo inminente. Por ello, se ha optado por no profundizar en estas figuras para evitar redundancias.

### Evolución cuantitativa

La Tabla 7.4 muestra las métricas obtenidas del subconjunto FD004. Aunque puede observarse un ligero aumento en el error (especialmente en RMSE y MAE), los resultados se mantienen dentro de un rango aceptable, demostrando así que el modelo conserva su capacidad de predicción incluso bajo las condiciones más complejas del conjunto de datos CMAPSS.

| Métrica | Valor |
|---|---|
| α media | 77.64 |
| β media | 3.68 |
| MSE (“Mean Squared Error”) entre α predicho y TTE real | 485.53 |
| RMSE (“Root Mean Squared Error”) sobre RUL estimado | 23.79 |
| MAE (“Mean Absolute Error”) sobre RUL estimado | 17.06 |

Tabla 7.4: Evaluación cuantitativa del modelo (FD004).

### Atribuciones por sensor

En la Figura 7.14 se muestra el análisis de atribuciones IG por sensor para la unidad 62 del subconjunto FD004. Aunque el modelo mantiene un comportamiento razonable (buena correlación entre α y el TTE real), las curvas de IG(α) reflejan atribuciones más dispersas, irregulares y menos estructuradas que en los subconjuntos anteriores.

En este caso, no se identifican sensores claramente dominantes, y algunos que fueron relevantes en FD001-FD003 (como s_4 o s_7) muestran un comportamiento menos definido o incluso inverso. Otros sensores, como s_8 o s_15, presentan ciertas caídas finales en IG(α), pero sin una estructura suficientemente robusta como para interpretarlos como variables clave.

Esta pérdida de nitidez es coherente con la naturaleza del subconjunto FD004, que simula múltiples condiciones operativas y modos de fallo simultáneamente, de modo que las atribuciones tienden a repartirse entre más sensores, reflejando una contribución más compleja y menos localizada. Además, este patrón de atribuciones menos definido refuerza lo observado en el entrenamiento: la diferencia entre la pérdida de entrenamiento y validación y la inestabilidad del modelo en validación sugieren un cierto grado de sobreajuste. Las atribuciones menos estructuradas son un reflejo de que el modelo ha capturado patrones particulares del conjunto de entrenamiento que no se generalizan bien en condiciones diversas como las del subconjunto FD004. Se trata de un subconjunto complejo que requiere de un mayor entrenamiento.

A pesar de ello, la Figura 7.14 confirma que el modelo conserva cierta capacidad explicativa incluso en el entorno más complejo, aunque con menor claridad que en los escenarios controlados.

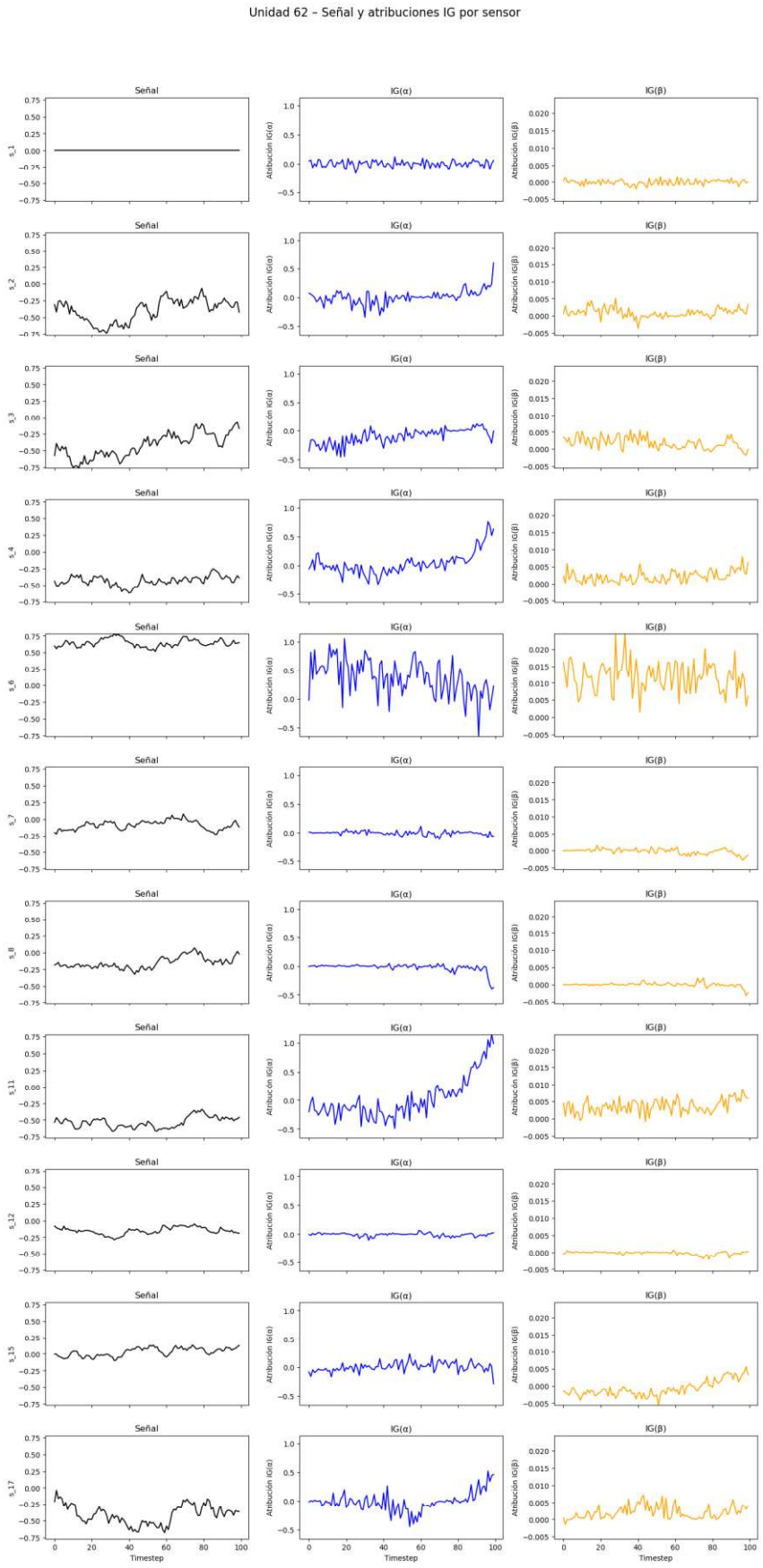


Figura 7.11: Señal e importancia IG por sensor en validación (FD004).

### 7.2.6 Conclusiones de la evaluación por subconjuntos

La evaluación independiente sobre los cuatro subconjuntos del "dataset" C-MAPSS ha permitido comprobar la capacidad del modelo para adaptarse a distintos escenarios de degradación. En todos los casos, se ha mantenido una concordancia aceptable entre el parámetro α y el TTE real, con una estabilidad destacable en β como modulador del tipo de fallo.

Los errores cuantitativos se han mantenido dentro de los márgenes razonables, incluso en los subconjuntos con mayor complejidad, lo que permite afirmar que el modelo es capaz de generalizar sin perder mucha precisión. Además, el uso de "Integrated Gradients" ha permitido obtener información sobre los sensores más relevantes y su consistencia en todos los escenarios, pudiendo destacar algunos como s_4, s_7 o s_12 que han mostrado patrones de atribución repetidos y coherentes.

En el caso puntual del subconjunto FD004, la pérdida de validación ha mostrado mayor oscilación y valores más elevados, lo que, junto con las atribuciones más difusas en IG, sugiere la presencia de cierto grado de sobreajuste. Se considera un comportamiento esperable debido a la elevada variabilidad de condiciones operativas y modos de fallo, que dificultan la generalización del modelo y dispersan la contribución de los sensores relevantes.

En conjunto, estos resultados refuerzan la viabilidad de aplicar este enfoque en escenarios reales, por su capacidad para ofrecer predicciones razonables y robustas, incluso cuando las condiciones operativas son variables e inciertas.

# Capítulo 8: Conclusiones y Líneas Futuras

## 8.1 Introducción

El presente capítulo tiene como objetivo sintetizar las conclusiones más relevantes derivadas del desarrollo del proyecto, así como proponer posibles líneas futuras de trabajo. A lo largo del proyecto se ha implementado y evaluado un modelo basado en redes neuronales densas para la estimación de la vida útil remanente (RUL) en motores aeronáuticos, utilizando el conjunto de datos C-MAPSS de la NASA como referencia.

Partiendo de un entorno controlado con datos sintéticos ("toy dataset"), se ha verificado el correcto funcionamiento del modelo de supervivencia propuesto, para posteriormente adaptarlo a un entorno real con múltiples condiciones operativas y tipos de fallos.

Asimismo, se han integrado técnicas de explicabilidad "post-hoc" ("Integrated Gradients") con el fin de interpretar el comportamiento del modelo y analizar la relevancia temporal y sensorial de las predicciones, aportando una capa de interpretación esencial para su aplicación en contextos críticos.

A continuación, se recogen las conclusiones clave obtenidas, la discusión de los resultados de acuerdo con los objetivos planteados y una reflexión acerca de posibles mejoras y extensiones del sistema que podrían abordarse en trabajos futuros.

## 8.2 Conclusiones

El proyecto ha permitido obtener varias conclusiones significativas sobre el desempeño y la aplicabilidad del modelo denso aplicado:

- **Desempeño del modelo**: El modelo implementado ha demostrado tener una buena capacidad de predicción del parámetro α (escala), manteniendo una concordancia notable con el tiempo hasta el fallo real (TTE), especialmente en los subconjuntos FD001 y FD003. El modelo ha sido capaz de aprender patrones de deterioro de forma generalizable, sin la necesidad de información adicional para realizar ajustes específicos.

  A pesar de que hay conjuntos con condiciones operativas más variables (FD002 y FD004) donde se observan mayores oscilaciones y cierta pérdida de precisión, los valores de RMSE y MAE obtenidos (en torno a 11-17 ciclos), se han mantenido dentro de un rango razonable en todos los casos, siendo esto un indicador de robustez ante entornos más complejos.

- **Evaluación cuantitativa y eficiencia:** El modelo ha alcanzado métricas de error razonablemente bajas en todos los subconjuntos, con un rendimiento destacado en aquellos subconjuntos más controlados. Además, el criterio de selección de sensores ha resultado útil para reducir la dimensionalidad sin perder información relevante, contribuyendo tanto a la eficiencia del entrenamiento como a la claridad en la explicación del modelo.

- **Interpretabilidad y explicabilidad:** Mediante la técnica de "Integrated Gradients" (IG) ha sido posible interpretar de forma visual y cuantitativa la sensibilidad del modelo ante distintos sensores y momentos temporales. Esto ha permitido observar una respuesta consistente ante la inminencia del fallo (α) y una contribución algo más difusa pero coherente en la modelización de la forma del deterioro (β).

  Además, gracias a los resultados obtenidos se han podido destacar algunos sensores como s_4, s_7 y s_12, debido a su contribución destacada de forma transversal en varios subconjuntos, pudiéndose considerar a estos como sensores clave en futuras implementaciones o simplificaciones del modelo.

- **Aplicabilidad práctica**: El sistema propuesto, aunque aún no validado en entornos reales, ha demostrado contar con una estructura funcional y adaptable para escenarios de mantenimiento predictivo. Además, la posibilidad de extraer interpretaciones claras a partir de sus salidas les otorga un valor añadido frente a modelos "black box" puros, facilitando así su integración en sistemas críticos, donde la trazabilidad y la confianza son esenciales.

- **Limitaciones**: Se han identificado ciertas limitaciones. En primer lugar, la arquitectura utilizada es completamente densa, lo que puede limitar su capacidad para capturar relaciones a largo plazo o patrones más complejos que podrían abordarse con arquitecturas recurrentes o convolucionales. En segundo lugar, si bien C-MAPSS es un entorno de simulación realista, no refleja la totalidad de la complejidad que presentaría un despliegue en condiciones operacionales reales: presencia de ruido, fallos múltiples inesperados, sensores ausentes o mal calibrados, y datos heterogéneos entre plataformas.

Por tanto, una de las principales conclusiones es que el sistema desarrollado constituye una base sólida para la predicción de RUL en motores simulados, pero tiene rango de mejora, pudiendo añadir una validación adicional y adaptaciones si se desea extender su uso a contextos industriales reales, con datos de operación de campo no simulados. Esto plantea oportunidades claras de evolución, que se presentan en el siguiente apartado sobre líneas futuras de trabajo.

## 8.3 Líneas Futuras

En base a los resultados y observaciones obtenidas durante el desarrollo de este proyecto, se identifican diversas líneas futuras de trabajo que podrían permitir mejorar, extender y adaptar el sistema propuesto a nuevos contextos. Algunas de las más relevantes son:

- **Validación con datos reales de campo**: Una de las principales proyecciones de futuro sería aplicar el modelo a datos reales procedentes de sistemas industriales (por ejemplo, motores de aviación en operación o turbinas en entornos de producción). Esto permitiría evaluar la robustez del modelo frente a condiciones no simuladas, presencia de ruido, sensores defectuosos o secuencias incompletas, siendo estos algunos aspectos comunes en la práctica real.

- **Incorporación de arquitecturas más complejas:** El enfoque basado en redes densas ha resultado efectivo. Sin embargo, podría explorarse el uso de arquitecturas alternativas como redes recurrentes (LSTM/GRU), convolucionales

(CNN) o combinadas (CNN-RNN) para mejorar la capacidad de capturar relaciones temporales a largo plazo y patrones complejos en las secuencias.

- **Predicción conjunta de RUL y tipo de fallo**: En futuras versiones, el modelo podría incluir una segunda tarea de clasificación además de estimar el tiempo hasta el fallo, prediciendo así el tipo de fallo más probable (fallo mecánico, por ejemplo), integrando así un enfoque multitarea más completo.

- **Refinamiento de técnicas de explicabilidad**: El uso de "Integrated Gradients" ha resultado útil, pero podrían evaluarse otras técnicas de interpretación (como SHAP adaptado a secuencias, LIME o mecanismos de atención) con el fin de obtener explicaciones más precisas y adaptadas a diferentes perfiles de usuario (ingenieros, operadores de planta, etc.).

- **Automatización del proceso de selección de sensores**: En el proyecto actual, la selección de sensores se ha realizado de forma manual y basada en criterios previamente publicados. En futuras implementaciones podría automatizarse este proceso utilizando técnicas de selección de características basadas en importancia relativa, correlación con el RUL o métodos embebidos.

- **Despliegue en tiempo real y optimización de rendimiento**: Una vez se haya validado el sistema en condiciones reales, podría avanzarse hacia su integración en entornos de monitorización en tiempo real, evaluando su latencia, consumo de recursos y posibilidad de ejecución en "hardware" embebido o entornos distribuidos.

Este conjunto de líneas futuras constituye una hoja de ruta realista para convertir el sistema desarrollado en este proyecto en una herramienta más compleja, flexible y cercana a una aplicación industrial real.

# Capítulo 9: Presupuesto

En este apéndice se ofrece una estimación de los costes que conlleva la realización de este proyecto. Se diferencian dos grandes grupos de costes: medios materiales y recursos humanos

## 9.1 Costes de Materiales

A continuación, en la Tabla A.1, se muestran los costes del material utilizado durante el desarrollo del proyecto, ascendiendo a un total de **1.624€.**

| Concepto | Total (€) |
|---|---|
| Portátil HP Pavilion x360 -14-dw0000ns | 950 |
| Pantalla HP | 250 |
| Ratón Targus | 30 |
| Herramientas de desarrollo: Visual Studio y Miniconda | 0 |
| Herramientas de desarrollo: Python | 0 |
| Sistema operativo Windows 10 | 145 |
| Microsoft Office 365 | 99 |
| Base de datos NASA Turbofan Jet Engine | 0 |
| Impresiones | 150 |
| **Total** | **1.624** |

Tabla 9.1: Costes de materiales.

## 9.2 Costes de Personal

De acuerdo con informes salariales recientes [49], el salario medio bruto anual de un ingeniero recién graduado en España ronda los 25.000€. Considerando que hay 1820 horas laborales en un año, un recién graduado en Ingeniería y Sistemas de Datos puede cobrar en torno a 13,7€ la hora. En la Tabla A.2 mostrada a continuación se puede ver a cuanto ascendería el coste total del recién graduado teniendo en cuenta que ha trabajado 330 horas.

Por otra parte, para el desarrollo del proyecto ha sido necesaria la colaboración de un tutor y un director de PFG, estimándose que ambos profesionales cobrarán un 7,5% cada uno de la suma del coste de materiales y la mano de obra. En la Tabla A.2 se representan ambos sueldos, ascendiendo el total del coste de personal a **5.199,15€.**

| Concepto | Cantidad | Coste unitario (€) | Total (€) |
|---|---|---|---|
| Mano de obra | 330 horas | 13,7 | 4.521 |
| Coste de dirección | 2 personas | 339,08 | 678,15 |
| **Total** | | | **5.199,15** |

Tabla 9.2: Costes de personal.

## 9.3 Coste Total

Teniendo en cuenta el coste de materiales y el coste de personal, en la Tabla A.3 puede observarse el coste total del proyecto, que asciende a **6.823,15€.**

| Categoría | Total (€) |
|---|---|
| Costes de Materiales | 1.624 |
| Recursos Humanos | 5.199,15 |
| Total General | **6.823,15€.** |

Tabla 9.3: Costes totales.

# Capítulo 10: Impacto del Proyecto

## 10.1 Introducción

Este apéndice cuenta con el objetivo de analizar el impacto que el desarrollo de este proyecto puede tener desde múltiples perspectivas: social, económica, de salud y de seguridad, tecnológica y con relación a los Objetivos de Desarrollo Sostenible (ODS) [50]. El sistema se ha desarrollado en un entorno académico y simulado, pero su planteamiento tiene una clara orientación práctica hacia el mantenimiento predictivo en sistemas críticos, lo que confiere relevancia a la reflexión sobre su impacto potencial.

## 10.2 Impacto Social

El uso de sistemas de mantenimiento predictivo basados en inteligencia artificial contribuye a una mayor seguridad y fiabilidad en sectores sensibles como puede ser la aviación. A pesar de que el proyecto no se ha desplegado directamente en un entorno social, el desarrollo de tecnologías más transparentes, interpretables y seguras puede influir en gran medida en la confianza pública ante los sistemas automáticos. Además, la incorporación de mecanismos explicativos favorece el despliegue de estos modelos desde un punto de vista ético y social ya que permiten auditar su funcionamiento y detectar posibles sesgos.

## 10.3 Impacto en la Salud y la Seguridad

Uno de los objetivos indirectos del mantenimiento predictivo en sistemas como los motores aeronáuticos es minimizar el riesgo de fallo catastrófico, teniendo esto una implicación directa en la seguridad de los pasajeros y el correspondiente personal técnico. Por tanto, al ofrecer estimaciones confiables del tiempo hasta el fallo (RUL) y permitir una interpretación clara de las decisiones del modelo, se contribuye a mejorar la toma de decisiones, así como la prevención de riesgos. Todo ello se encuentra alineado con los principios básicos de seguridad operacional y fiabilidad en ingeniería de sistemas críticos.

## 10.4 Impacto Económico

El mantenimiento predictivo es una herramienta clave para reducir costes operativos y de reparación en sectores industriales. La posibilidad de anticipar fallos antes de que ocurran permite evitar paradas imprevistas, reducir el coste de reparación y optimizan la vida útil de los activos. Además, el modelo implementado en este proyecto, al mantener una buena eficiencia computacional y explicable, podría contribuir a sistemas con un coste de adopción menor y una menor dependencia ante infraestructuras complejas, lo que le abre las puertas de pequeñas y medianas empresas sin la necesidad de descartar grandes industrias.

## 10.5 Impacto Tecnológico e Industrial

Este proyecto contribuye a la adopción de técnicas de aprendizaje automático en entornos reales, al abordar uno de sus grandes desafíos en la actualidad: la explicabilidad de los modelos. El uso de la técnica de “Integrated Gradients” para analizar la importancia temporal y sensorial en la predicción de fallos proporciona una herramienta útil para desarrolladores y operadores, aumentando así la confianza y dando pie a la supervisión. Gracias a este enfoque

se impulsa el desarrollo de sistemas híbridos, que combinan rendimiento predictivo con interpretabilidad.

## 10.6 Contribución a los Objetivos de Desarrollo Sostenible (ODS)

El proyecto se alinea especialmente con los siguientes ODS de las Naciones Unidas:


Figura 10.1: Objetivos de Desarrollo sostenible vinculados al proyecto

**ODS 8: Trabajo decente y crecimiento económico**

Facilita el desarrollo de soluciones tecnológicas para ser implementadas en la industria, generando así un valor añadido y empleo cualificado.

**ODS 9: Industria, innovación e infraestructura**

Promueve el uso de tecnologías avanzadas y sostenibles para la gestión de infraestructuras críticas

**ODS 12: Producción y consumo responsable**

El proyecto fomenta un mantenimiento eficiente y predictivo, que reduce el desperdicio de recursos y prolonga la vida útil de los equipos.

**ODS 13: Acción por el clima**

Indirectamente, al evitar reparaciones innecesarias y prolongar el uso eficiente de recursos, puede contribuir a la reducción de emisiones en el ciclo de vida de componentes industriales.

# Capítulo 11: Manual de Usuario

## 11.1 Introducción

Este manual de usuario proporciona las instrucciones necesarias para ejecutar el modelo de predicción de vida útil remanente (RUL) desarrollado en este proyecto. El sistema se desarrolla en un entorno Python ejecutado en Jupyter Notebook, y ha sido adaptado a partir de un repositorio público de GitHub [4], de donde se han modificado componentes clave como la arquitectura del modelo (implementando una red densa) y parte del preprocesamiento.

## 11.2 Requisitos previos

Para ejecutar correctamente el sistema, es necesario contar con:

- Sistema operativo: Windows 10 o superior.
- Entorno Python: Se recomienda instalar Miniconda.
- Editor de código recomendado: Visual Studio Code.
- Entorno Jupyter: puede instalarse a través de Conda o como extensión de VS Code.

**Dependencias necesarias.**

Se recomienda crear un entorno virtual e instalar las librerías indicadas en el archivo "requirements.txt", y mostradas en la Figura 8.2, mediante el comando "pip install -r requirements.txt".

```
torch==2.4.1
torchvision==0.19.1
torchaudio==2.4.1
captum==0.7.0
numpy==1.24.4
pandas==2.0.3
scikit-learn==1.3.0
scipy==1.10.1
matplotlib==3.7.3
seaborn==0.13.2
tqdm==4.67.1
ipython==8.12.2
ipykernel==6.29.5
jupyter==1.0.0
joblib==1.4.2
statsmodels==0.14.1
```

Figura 11.1: Contenido del archivo "requirements.txt".

## 11.3 Estructura del proyecto

El proyecto se encuentra organizado dentro de una carpeta llamada "torch-wtte-main", con la siguiente estructura:

- "Data/": contiene los ficheros de entrenamiento, "test" y RUL para FD001-FD004.

- "torch_wtte/losses.py": implementación de las funciones de pérdida.

- "Examples/":
    - "simple_example_CMAPSS.ipynb": "Notebook" principal utilizado para entrenar y visualizar resultados con los datos reales (CMAPSS).

    - "simple_example_ToyDataset.ipynb": Ejemplo con datos sintéticos utilizado previamente para verificar el funcionamiento del modelo.

    - "preprocess_real.py": Módulo para el preprocesamiento de datos reales.

- "Tests/test_losses.py": Validación de las funciones de pérdida.

- "requirements.txt": contiene las librerías necesarias del proyecto para ser instaladas en un entorno virtual.

- "README.md": Información técnica original del repositorio base.

## 11.4 Ejecución paso a paso

Para ejecutar el programa desarrollado, deben realizarse los pasos enumerados a continuación:

1. Clonar el repositorio o descomprimir la carpeta "torch-wtte-main".

2. Crear un entorno virtual en Miniconda:
   *Conda créate – rul_enc Python=3.8*
   *Conda actívate rul_env*

3. Instalar las dependencias necesarias:
   *pip install -r requirements.txt*

4. Abrir Visual Studio Code o Jupyter Notebook y ya dentro, abrir el archivo "simple_example_CMAPSS.ipynb" dentro de la carpeta "/examples".

5. Dentro del "notebook" pueden configurarse los parámetros iniciales como el dataset a usar (FD001-FD004), el tamaño de ventana ("window_size"), el número de épocas ("epoch"), el "threshold" para limitar las muestras, etc.

    - Para modificar el "threshold" es necesario hacerlo directamente desde el archivo "preprocess_real.py".

6. Ejecutar todas las celdas en orden (en primer lugar, las importaciones para confirmas el funcionamiento del resto de código).

## 11.5 Parámetros modificables

Los siguientes parámetros pueden ser modificados por el usuario previo a la ejecución del código.

- "Dataset": Define el subconjunto CMAPSS a emplear y se debe indicar cual se va a usar (FD001-FD004). Si no se indica se ejecutará con el FD001.

- "Window_size": Tamaño de secuencia temporal usado (está establecido en 100).

- "epoch": número de épocas de entrenamiento (está establecido en 50).

- "threshold": Valor que afecta el preprocesamiento y censura de eventos. Debe modificarse en el archivo "preprocess_real.py".

## 11.6 Salidas del sistema

Durante la ejecución del "notebook", se generan automáticamente los gráficos que se han mostrado a lo largo de la memoria, así como los resultados numéricos obtenidos. Estos pueden copiarse y guardarse en el propio equipo donde se ejecuta el programa, pero éste no lo hace de forma automática.

## 11.7 Recomendaciones

Para finalizar, un par de recomendaciones:

- El sistema no requiere GPU, pero puede beneficiarse de ella si está disponible, de modo que no eliminar el fragmento de código empleado para ello.

- Crear un entorno virtual único para el proyecto con el objetivo de evitar cruces de versiones que puedan generar fallos en la ejecución del código

# Referencias

[1] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, Munich, Germany: Christoph Molnar, 2022.

[2] A. Theissler, F. Schillinger, S. Uhlig, G. Reitmayr, A. Schuller y J. Großmann, «Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions,» IEEE Access, vol. 10, pp. 100700-100724, 2022.

[3] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin y N. Díaz-Rodríguez, «Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey,» 2021.

[4] alexkyllo, «A PyTorch implementation of Weibull Time to Event Recurrent Neural Networks for churn prediction tasks.,» 2021. [En línea]. Available: https://github.com/alexkyllo/torch-wtte.

[5] M. Sundararajan, A. Taly y Q. Yan, «Axiomatic Attribution for Deep Networks.,» Proceedings of the 34th International Conference on Machine Learning (ICML), vol. 70, p. 3319–3328, 2017.

[6] T. Amestoy, «Clustering Basics and a Demonstration in Clustering Infrastructure Pathways,» Water Programming Blog, 16 03 2022. [En línea]. Available: https://waterprogramming.wordpress.com/2022/03/16/clustering-basics-and-a-demonstration-in-clustering-infrastructure-pathways/.

[7] C. C. Aggarwal, Recommender Systems: The Textbook, Cham, Switzerland: Springer, 2016.

[8] CEUPE, «Aprendizaje por refuerzo: qué es y cómo funciona,» CEUPE Blog, [En línea]. Available: https://www.ceupe.com/blog/aprendizaje-por-refuerzo.html..

[9] IBM, «Machine Learning,» [En línea]. Available: https://www.ibm.com/es-es/topics/machine-learning.

[10] IBM, «Decision Trees,» IBM Topics, [En línea]. Available: https://www.ibm.com/es-es/topics/decision-trees.

[11] IBM, «Convolutional Neural Networks,» IBM Topics, [En línea]. Available: https://www.ibm.com/es-es/topics/convolutional-neural-networks.

[12] Codificando Bits, «Redes Convolucionales: Introducción,» s.f.. [En línea]. Available: https://codificandobits.com/blog/redes-convolucionales-introduccion/.

[13] IBM, «Aprendizaje no supervisado,» 2024. [En línea]. Available: https://www.ibm.com/es-es/topics/unsupervised-learning..

[14] Pranshu, «K-Means Clustering Simplified in Python,» Analytics Vidhya, 2021. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/04/k-means-clustering-simplified-in-python/.

[15] R. S. Sutton y A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 2018.

[16] Unión Europea, «Recital 71 del Reglamento General de Protección de Datos (GDPR),» [En línea]. Available: https://gdpr-text.com/es/read/recital-71/.

[17] Comisión Europea, «The EU Artificial Intelligence Act,» 2021. [En línea]. Available: https://artificialintelligenceact.eu/.

[18] IBM, «¿Qué es la IA explicable?,» [En línea]. Available: https://www.ibm.com/es-es/topics/explainable-ai.

[19] Google, «"Google Trends: Explore",» [En línea]. Available: https://trends.google.es/trends/explore?q=explainable%20ai&date=all.

[20] Management Solutions, «Explainable Artificial Intelligence,» Management Solutions, 2023. [En línea]. Available: https://www.managementsolutions.com/sites/default/files/minisite/static/22959b0f-b3da-47c8-9d5c-80ec3216552b/iax/pdf/explainable-artificial-intelligence-sp.pdf.

[21] A. Adadi y M. Berrada, «Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),» IEEE Access, vol. 6, pp. 52138-52160, 2018.

[22] P. Linardatos, V. Papastefanopoulos y S. Kotsiantis, «Explainable AI: A Review of Machine Learning Interpretability Methods,» Entropy, vol. 23, nº 1, p. 18, 2021.

[23] J. H. Friedman, «Greedy function approximation: a gradient boosting machine,» Annals of Statistics, vol. 29, nº 5, p. 1189–1232, 2001.

[24] S. M. Lundberg y S.-I. Lee, «A Unified Approach to Interpreting Model Predictions,» Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[25] L. S. Shapley, «A value for n-person games,» de Contributions to the Theory of Games II, Princeton, NJ, Princeton University Press, 1953, p. 307–317.

[26] F. Fliguer, «AAI – Aprendizaje Automático Interpretado,» [En línea]. Available: https://fedefliguer.github.io/AAI/shap.html#ejemplos-4.

[27] M. T. Ribeiro, S. Singh y C. Guestrin, «"Why Should I Trust You?": Explaining the Predictions of Any Classifier,» Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), p. 1135–1144, 2016.

[28] M. T. Ribeiro, S. Singh y C. Guestrin, «Anchors: High-Precision Model-Agnostic Explanations,» de AAAI Conference on Artificial Intelligence, 2018.

[29] K. Erdem, «Medium,» 2023. [En línea]. Available: https://medium.com/@kemalpiro/xai-methods-integrated-gradients-6ee1fe4120d8.

[30] J. A. Martínez Pérez y P. S. Pérez Martínez, «Análisis de supervivencia,» Medicina de Familia. SEMERGEN, vol. 49, nº 5, 2023.

[31] D. W. Wormuth, «Actuarial and Kaplan-Meier survival analysis: There is a difference,» The Journal of Thoracic and Cardiovascular Surgery, vol. 118, nº 4, p. 716–717, 1999.

[32] D. R. Cox, «Regression models and life-tables,» Journal of the Royal Statistical Society: Series B (Methodological), vol. 34, nº 2, pp. 187-202, 1972.

[33] P. Rebasa, «Conceptos básicos del análisis de supervivencia,» Cirugía Española (Cir Esp.), vol. 78, nº 4, pp. 222-223, 2005.

[34] T. G. Clark, M. J. Bradburn, S. B. Love y D. G. Altman, «Survival analysis part I: basic concepts and first analyses,» British Journal of Cancer, vol. 89, nº 3, pp. 232-238, 2003.

[35] V. Abraira, «Análisis del tiempo hasta un evento (supervivencia),» Medicina de Familia. SEMERGEN, vol. 30, nº 5, pp. 223-225, 2004.

[36] L. Löschmann y D. Smorodina, «Survival Analysis: Leveraging Deep Learning for Time-to-Event Forecasting,» 05 02 2019. [En línea]. Available: https://medium.com/data-science/survival-analysis-leveraging-deep-learning-for-time-to-event-forecasting-5c55bd4bb066.

[37] H. Ishwaran, U. B. Kogalur, E. H. Blackstone y M. S. Lauer, «Random survival forests,» The Annals of Applied Statistics, vol. 2, pp. 841-860, 2008.

[38] ScienceDirect, «Weibull Distribution,» 2023. [En línea]. Available: https://www.sciencedirect.com/topics/physics-and-astronomy/weibull-distribution.

[39] A. Sánchez Moreno, «Modelos Matemáticos Utilizados en Análisis de Supervivencia / Mathematical Models in Survival Analysis,» Universidad de Salamanca, Salamanca, 2021.

[40] IBM, «Redes neuronales recurrentes (RNN),» 25 10 2023. [En línea]. Available: https://www.ibm.com/es-es/topics/recurrent-neural-networks.

[41] M. Martinsson y E. E. Martinsson, «WTTE-RNN: A Framework for Churn and Time to Event Prediction,» 2016. [En línea]. Available: https://ragulpr.github.io/assets/draft_master_thesis_martinsson_egil_wtte_rnn_2016.pdf.

[42] Behrad3d, «NASA C-MAPSS Turbofan Engine Degradation Simulation Dataset.,» Kaggle, 2019. [En línea]. Available: https://www.kaggle.com/datasets/behrad3d/nasa-cmaps-turbofan-engine-degradation-simulation.

[43] Python Software Foundation, «Python Programming Language – Official Website,» 2024. [En línea]. Available: https://www.python.org/.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan y e. al., «PyTorch: An imperative style, high-performance deep learning library,» Advances in Neural Information Processing Systems (NeurIPS), vol. 32, p. 024–8035, 2019.

[45] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau y e. al, «Array programming with NumPy,» Nature, vol. 585, nº 7825, p. 357–362, 2020.

[46] J. D. Hunter, «Matplotlib: A 2D graphics environment,» Computing in Science & Engineering, vol. 9, nº 3, pp. 90-95, 2007.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel y e. al., «Scikit-learn: Machine learning in Python,» Journal of Machine Learning Research, vol. 12, p. 2825–2830, 2011.

[48] A. Saxena, K. Goebel, D. Simon y N. Eklund, «Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation,» IEEE, 2008.

[49] Jobted, «Jobted,» 2025. [En línea]. Available: https://www.jobted.es/salario/ingeniero.

[50] N. Unidas, «Objetivos de Desarrollo Sostenible,» s.f.. [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.