COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Comparativa de Modelos de Predicción de Mantenimiento Predictivo y Exploración de Datos de Motores Turbofán de la Nasa.

Autor: **Daniel Losada Rueda**

Director: **Iago Abuín Álvarez**

Madrid - Julio 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Comparativa de Modelos de Predicción de Mantenimiento Predictivo y Exploración de Datos de Motores Turbofán de la Nasa.

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2023/ 2024** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: **Daniel Losada Rueda** Fecha: 04/ 07/ 2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: **Iago Abuín Álvarez** Fecha: 04/ 07/ 2024

COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Comparativa de Modelos de Predicción de Mantenimiento Predictivo y Exploración de Datos de Motores Turbofán de la Nasa.

Autor: **Daniel Losada Rueda**

Director: **Iago Abuín Álvarez**

Madrid - Julio 2024

Agradecimientos

A mis padres.

A mis amigos.

Comparativa de Modelos de Predicción de Mantenimiento Predictivo y Exploración de Datos de Motores Turbofán de la Nasa.

Autor: Losada Rueda, Daniel.
Director: Abuín Álvarez, Iago.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto se centra en la comparación de modelos de predicción de mantenimiento predictivo y análisis de datos de motores turbofán usando el conjunto de datos C-MAPSS de la NASA. El objetivo es evaluar diferentes modelos dentro del Machine Learning, como Support Vector Regression (SVR), Random Forest, Redes Neuronales Convolucionales (1D-CNN) y Redes Neuronales de Memoria a Largo Plazo (LSTM), todo para predecir la Remaining Useful Life (RUL) de los motores.

El análisis incluye un estudio exploratorio de los datos (EDA) para poder entender mejor las características y estructuras de los datos. Cada uno de los modelos se evaluará usando métricas elegidas como el Error Cuadrático Medio (RMSE) y el Error Absoluto Medio (MAE). Se busca no solo identificar el modelo con mejor precisión, sino también explorar la efectividad de diferentes técnicas de preprocesamiento.

En la fase inicial, se realizó una revisión exhaustiva de la literatura para contextualizar los fundamentos teóricos y prácticos de los modelos seleccionados. También se presentaron las diferentes herramientas con las que se iba a trabajar a lo largo del proyecto, con las que se desarrollaría el código necesario para implementar los modelos.

Figura 1: Diagrama Normal del Flujo del trabajo con Datos

La implementación de los modelos se hizo en varias etapas.

1. Se construyeron modelos de regresión como SVR o Random Forest, conocidos por manejar datos no lineales y ser robustos frente a datos ruidosos. Se optimizaron mediante validación cruzada y se buscaron los mejores hiperparámetros.

2. Posteriormente, se implementaron modelos más avanzados como son las Redes Neuronales. Específicamente 1D-CNN y LSTM, se seleccionaron por su capacidad amplia de capturar patrones complejos y dependencias temporales en datos de series temporales. Durante el entrenamiento, se monitorea constantemente la validación para evitar el sobreajuste y asegurar predicciones precisas y generalizables usando ajustes dinámicos de la tasa de aprendizaje para mejorar la convergencia del modelo.

3. Los resultados fueron exhaustivamente analizados y comparados. Se pudo observar que las Redes Neuronales, especialmente las LSTM, ofrecieron las predicciones más precisas debido a su capacidad para capturar dependencias temporales de largo plazo en los datos.

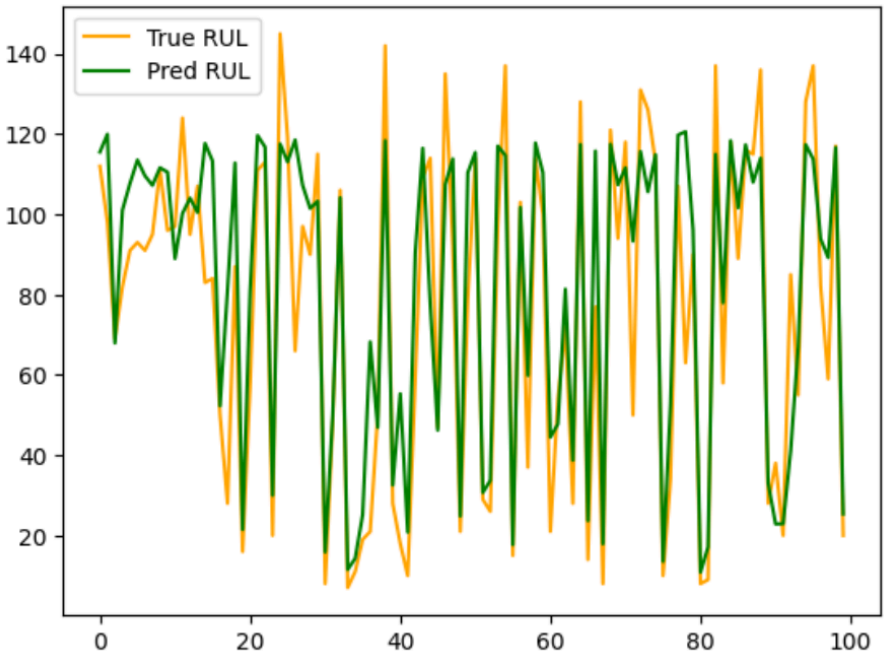


Figura 2: Ejemplo de Predicción del Proyecto

Este proyecto demuestra la importancia real de elegir un modelo adecuado y ser hábil ajustando sus hiperparámetros para poder maximizar el rendimiento predictivo. También resalta la necesidad de un buen preprocesamiento y análisis de los datos para tener un buen material con el que trabajar y poder tener resultados fiables.

Finalmente, cabe destacar que este trabajo contribuye significativamente a la reducción de costos operativos y de mantenimiento, pudiendo así mejorar la eficiencia y sostenibilidad en la industria aeroespacial. Al predecir con precisión la vida de los motores, se pueden planificar mejor las tareas de mantenimiento, reducir tiempos de inactividad no programados y optimizar la gestión de inventarios.

- **Extrapolación a Otros Problemas e Industrias:**

La capacidad de generalización de estos modelos no solo se limita a la predicción de la vida útil de motores turbofan. Estos enfoques pueden extrapolarse a una variedad de problemas en diferentes industrias. Por ejemplo, se pueden aplicar técnicas similares para predecir la vida útil de combustibles, baterías de coches eléctricos, dispositivos móviles y otros equipos industriales.

En la industria automotriz, predecir la RUL de baterías de coches eléctricos puede ayudar a mejorar la eficiencia y la sostenibilidad, permitiendo un mejor manejo del ciclo de vida de las baterías y reduciendo los residuos. En el sector de la energía, la predicción de la vida útil de los combustibles puede optimizar el uso de recursos y reducir los costos operativos. De manera similar, en la industria de la electrónica, predecir la vida útil de componentes críticos de dispositivos móviles puede mejorar la confiabilidad y la satisfacción del cliente, además de contribuir a la reducción de residuos electrónicos.

- **Resultados Finales:**

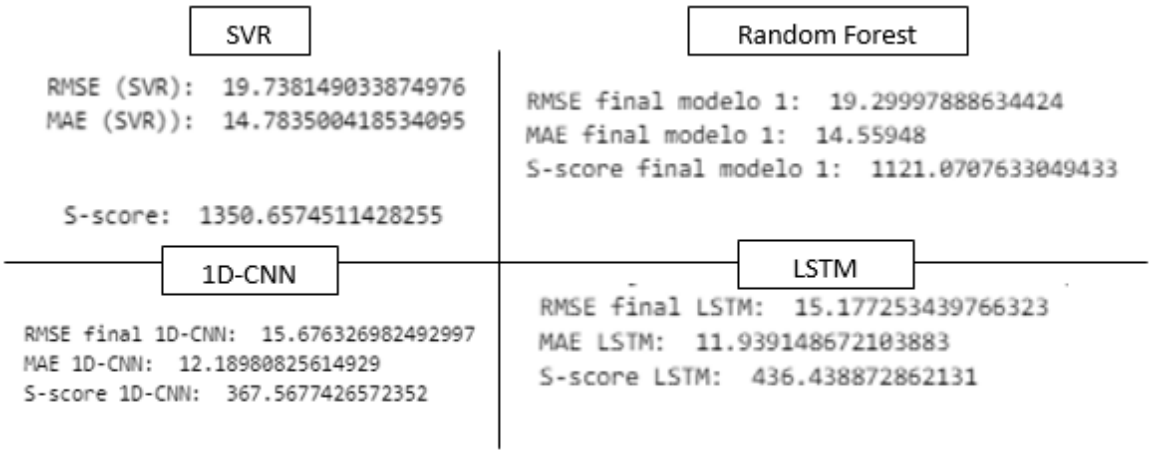


Figura 3: Tabla con Todos los Resultados Finales

Palabras clave: Mantenimiento predictivo, Machine Learning, CMAPSS, NASA, Remaining Useful Life (RUL), Motores turbofán, SVR, Random Forest,1D-CNN, LSTM

Predictive Maintenance Model Comparison and Data Exploration of NASA Turbofan Engines

Author: Losada Rueda, Daniel.
Supervisor: Abuín Álvarez, Iago.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project focuses on the comparison of predictive maintenance models and the analysis of turbofan engine data using the NASA C-MAPSS dataset. The objective is to evaluate different models within Machine Learning, such as Support Vector Regression (SVR), Random Forest, XGBoost, Convolutional Neural Networks (1D-CNN), and Long Short-Term Memory Networks (LSTM), to predict the Remaining Useful Life (RUL) of engines.

The analysis includes an exploratory data analysis (EDA) to better understand the characteristics and structure of the data. Each model will be evaluated using chosen performance metrics like Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). The goal is not only to identify the model with the best precision but also to explore the effectiveness of different preprocessing techniques.

In the initial phase, an exhaustive literature review was conducted to contextualize the theoretical and practical foundations of the selected models. The different tools that will be used throughout the project were also presented, which will be essential to develop the necessary code to implement the models.

Figure 1: Flux Diagram of Data Analysis Work

The implementation of the models was carried out in various stages:

1. SVR and Random Forest models were constructed, known for handling non-linear data and being robust against noisy data. Cross-validation was performed, and the best hyperparameters were sought.

2. More advanced models such as Neural Networks were implemented. Specifically, 1D-CNN and LSTM were selected for their ability to capture complex patterns and temporal dependencies in time series data. During training, validation was constantly monitored to avoid overfitting and ensure precise and generalizable predictions using dynamic adjustments to the learning rate for better model convergence.

3. The results were exhaustively analyzed and compared. It was observed that Neural Networks, especially LSTM, provided the most accurate predictions due to their capacity to capture long-term temporal dependencies in the data.

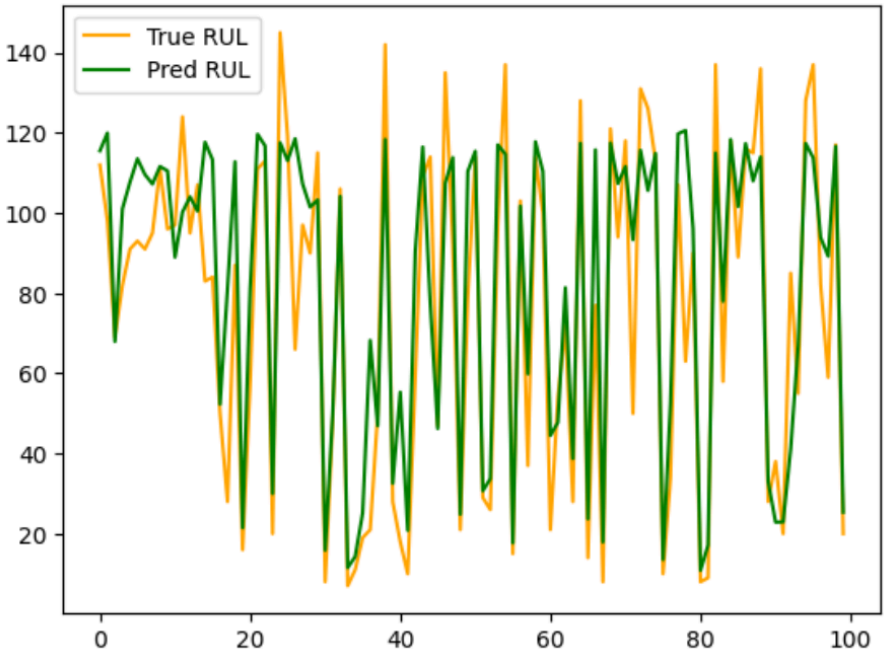


Figure 2: Prediction Example from the Project

This project demonstrates the real importance of choosing an adequate model and being skillful in tuning hyperparameters to maximize predictive performance. It also highlights the necessity of good preprocessing and data analysis to have quality material to work with and achieve reliable results.

Finally, it should be noted that this work significantly contributes to the reduction of operational and maintenance costs, thereby improving efficiency and sustainability in the aerospace industry. By accurately predicting engine life, maintenance tasks can be better planned, reducing unplanned downtime and optimizing inventory management.

- **Extrapolation to Other Problems and Industries:**

The generalization capability of these models is not limited to predicting the useful life of turbofan engines. These approaches can be extrapolated to a variety of problems in different industries. For example, similar techniques can be applied to predict the useful life of fuels, electric car batteries, electronic devices, and other industrial equipment.

In the automotive industry, predicting the RUL of electric car batteries can help improve efficiency and sustainability, allowing better management of battery life cycles and reducing waste. In the energy sector, predicting the life of fuels can optimize resource use and reduce operating costs. Similarly, in the electronics industry, predicting the life of electronic components can improve reliability and customer satisfaction, while contributing to the reduction of electronic waste.

- **Final Results:**

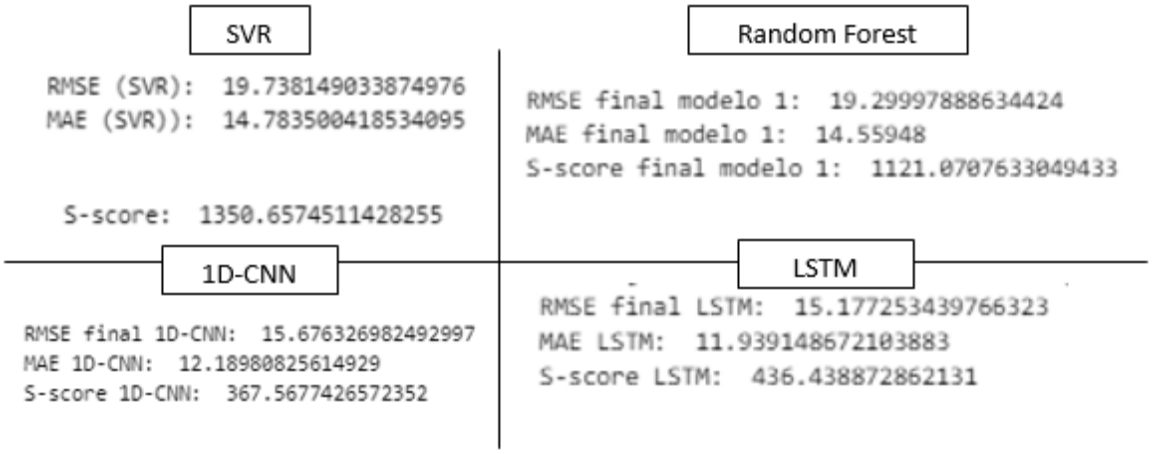


Figure 3: Final Results

Keywords: Predictive maintenance, Machine Learning, C-MAPSS, NASA, Remaining Useful Life (RUL), Turbofan engines, SVR, Random Forest, 1D-CNN, LSTM.Bluetooth, Mobile, Indoor

Índice de la memoria

CAPÍTULO 1. INTRODUCCIÓN

Los sistemas modernos de análisis de datos y aprendizaje automático buscan hoy en día ser más eficientes, escalables y seguros. Tras la proliferación de tecnologías de la información, la industria ha sufrido un cambio muy notable, dejando de estar aislados procesos que antes eran locales, mejorando así su disponibilidad y eficiencia.

El avance en la digitalización ha forzado a las organizaciones a modernizar sus recursos tecnológicos para mantenerse competitivas. Estos avances han mejorado significativamente los servicios en numerosos sectores, incluida la aviación, donde el mantenimiento predictivo de los motores es crucial.

La capacidad de anticipar y prevenir fallas en los motores mediante el análisis de datos detallados ha avanzado, respaldando la adopción de estrategias de mantenimiento predictivo. La adaptación de planes de mantenimiento para responder a posibles fallos permite una reducción significativa en inspecciones periódicas, aliviando el peso de mantenimientos rutinarios y costosos. Evitar paradas no programadas, que a menudo surgen como consecuencia de fallos imprevistos, es otro beneficio clave.

Éste Trabajo Fin de Grado se centra en la necesidad de implementar sistemas de mantenimiento predictivo más eficientes en la industria de la aviación mediante técnicas avanzadas de aprendizaje automático y análisis de datos. El análisis exhaustivo de datos provenientes de motores turbofán, recopilados y ofrecidos por la NASA, será el núcleo de este proyecto. A través de la evaluación comparativa de modelos de predicción, se busca determinar la eficacia de distintos enfoques en la anticipación y mitigación de problemas futuros.

1.1 OBJETIVOS DEL PROYECTO

Los objetivos que se abordarán en este proyecto son:

1. **Recopilación y Descripción de los Datos de la NASA:**

   Obtener el conjunto de datos CMAPSS (*Commercial Modular Aero-Propulsion System Simulation*), el cual cuenta con información sobre los ciclos de operación y datos de sensores de motores turbofán. Este conjunto de datos es proporcionado por la NASA y es ampliamente utilizado para el estudio de la RUL (*Remaining Useful Life)* de componentes industriales.

2. **Exploración y Preprocesamiento de los Datos:**

   Realizar un análisis exploratorio de los datos para comprender mejor su distribución, detectar valores atípicos, posibles anomalías, patrones, etc. Limpiar los datos y realizar transformaciones necesarias para prepararlos para su modelización.

3. **Implementación de Modelos de Predicción**

   Desarrollar y entrenar diferentes modelos de aprendizaje y técnicas de modelado para poder predecir la RUL de los motores basándose en los datos habilitados. Modelos de regresión, Máquinas de soporte Vectorial, Redes Neuronales y otros algoritmos pertinentes. Ajustar hiper parámetros y usar técnicas de cross-validation para optimizar el rendimiento de los modelos.

4. **Evaluación Comparativa**

   Evaluar el rendimiento de los diferentes modelos elegidos e implementados usando métricas estándar como el RMSE (Error Cuadrático Medio), MAE (Error Absoluto Medio). Comparar los resultados de cada modelo y analizar ventajas y desventajas.

5. **Selección del Modelo Óptimo**

   Seleccionar el modelo de predicción más eficiente para la aplicación de mantenimiento predictivo en motores. Basar la selección en los resultados de la evaluación comparativa, considerando puntos como la precisión de las predicciones, eficiencia computacional, facilidad de implementación, etc.

1.2 MOTIVACIÓN DEL PROYECTO

La motivación detrás de este proyecto es destacar la importancia del mantenimiento predictivo en la industria, en este caso en el sector aeronáutico, y su potencial de generalización a otros campos. En la aviación, anticipar y prevenir fallos en motores puede evitar reparaciones costosas y mejorar la eficiencia de operación. Hacer uso de técnicas de Machine Learning permite no solo mejorar la seguridad, sino también optimizar algunos tiempos de mantenimiento, lo que se traduce automáticamente en una reducción significativa de costos operativos.

Como se ha comentado, los resultados obtenidos no solo serán aplicables a la industria aeronáutica, sino que también se considerarán adaptables a otros sectores, como la energía, manufactura, salud, transporte, y en definitiva sectores donde la fiabilidad del equipo es esencial.

Además, este enfoque contribuye a la sostenibilidad al extender la vida útil de los equipos y también reducir el desperdicio de recursos haciendo ver prácticas más responsables y eficientes tanto económicas como ambientales.

CAPÍTULO 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS A UTILIZAR

El avance en la tecnología ha llevado a la creación de una amplia gama de herramientas especializadas para tareas concretas, diseñadas para optimizar diferentes puntos dentro del análisis de datos y el desarrollo de software y visualizaciones. Estas herramientas abarcan desde entornos de desarrollo integrados potentes (IDE), hasta plataformas de gestión y organización de código, las cuales ofrecerán soluciones robustas y escalables.

En este segundo capítulo se revisarán las herramientas que se usarán a lo largo del proyecto. Su correcta selección es esencial ya que determinan un importante acercamiento a alcanzar los objetivos planteados, no solo permitiendo la implementación de modelos de Machine Learning, sino también la organización eficiente de los datos y sus visualizaciones.

Se analizarán las distintas ventajas, funcionalidades, y aplicaciones concretas de cada una.

2.1 HERRAMIENTAS PARA EL DESARROLLO

En proyectos de Machine Learning las herramientas adecuadas son protagonistas para asegurar un eficaz y productivo flujo de trabajo. A continuación se presentan algunas de las principales herramientas que se estarán utilizando en este proyecto:

2.1.1 VISUAL STUDIO CODE

Visual Studio Code (VS Code) es un editor de código fuente ligero pero muy poderoso, el cual fue desarrollado por Microsoft. Ofrece soporte para múltiples enjuagues de programación y ofrece una amplia variedad de extensiones en sus bibliotecas.

- **Extensiones / Marketplace:** Ofrece un Marketplace donde se pueden descargar extensiones como formateadores, depuradores, etc, que simplifican el desarrollo final del código
- **Interfaz de Usuario:** Cuenta con una interfaz limpia, cómoda y altamente personalizable incluyendo paneles y terminales.
- **Terminal Integrado:** Contiene una terminal incorporada que permite ejecutar comandos de Shell directamente desde el editor, no teniendo que usar múltiples aplicaciones.

Se usará a lo largo de este Trabajo Fin de Grado dada su compatibilidad con Python, e integración con Git y GitHub.

Figura 2.1: Logo de Visual Studio Code

2.1.2 PYTHON

Python es un lenguaje de programación de alto nivel, muy conocido por su alta simplicidad y claridad. Se usa ampliamente en proyectos de análisis de datos y Machine Learning gracias a su inmensa colección de biblioteca y herramientas.

- **Gran Comunidad:** Cuenta con una gran comunidad de desarrolladores que aportan recursos, soporte y documentación para diferentes tipos de proyectos.

- **Extensa Biblioteca de Paquetes:** Cuenta con famosas y útiles librerías como Pandas, NumPy, Scikit-Learn, TensorFlow y Keras para analizar los datos y crear modelos.
- **Simplicidad:** Ofrece una sintaxis clara que facilita la escritura y la comprensión de código.

Se usará en este proyecto debido a la fácil integración con otras herramientas y plataformas utilizadas en el proyecto.

Figura 2.2: Logo de Python

2.1.3 GITHUB

Git es un sistema de control de versiones distribuido que permite a los programadores rastrear cambios y versiones actualizadas en el código y trabajar en paralelo. GitHub es una plataforma Web la cual utiliza Git para almacenar y compartir proyectos de programación.

- **Control de Versiones:** Permite rastrear cambios y volver a versiones anteriores del código.
- **Alojamiento de Proyectos:** Ofrece un espacio centralizado para compartir repositorios.

Se usará en este proyecto debido a la posibilidad que ofrece de almacenar el código, la integración y el despliegue continuo.

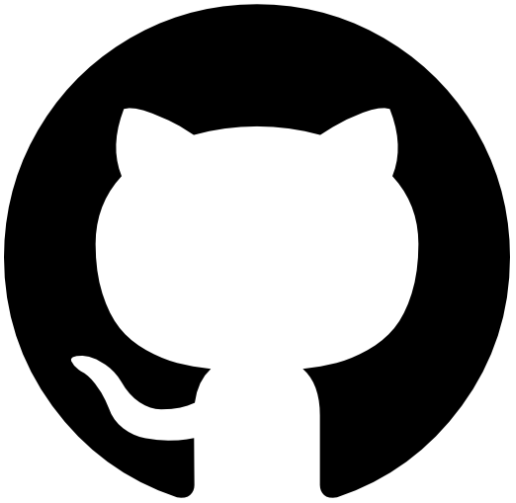


Figura 2.3: Logo de GitHub

2.1.4 JUPYTER NOTEBOOKS

Aplicación Web y extensión dentro de VS Code que permite crear y compartir documentos que contienen código en vivo

- **Interactividad:** Permite ejecutar bloques de código individualmente.
- **Visualización:** Permite la creación de visualizaciones directamente desde el Notebook.
- **Documentación Integrada:** Combina el código con resultados y explicaciones en un solo documento.

Se usará en este proyecto por su facilitación del análisis e iteración rápida durante todo el desarrollo de los diferentes modelos.

Figura 2.4: Logo de Jupyter

2.1.5 MATPLOTLIB

Matplotlib es una biblioteca de visualización de datos en Python la cual ofrece una gran flexibilidad para crear gráficos estáticos, animados e interactivos. Es muy conocido en la comunidad de Python por su capacidad de personalización y su amplia funcionalidad.

Ofrece control total sobre las características de las Visualizaciones, desde los colores hasta la disposición y el estilo de cada elemento. Esto incluye la capacidad de ajustar ejes, etiquetas, leyendas, etc.

- **Documentación y Comunidad:** Cuenta con una extensa documentación que incluye conceptos básicos y algo más avanzados. Esta comunidad es activa y contribuye con tutoriales y ejemplos facilitando la resolución de problemas.
- **Gráficos Avanzados:** Matplotlib permite crear gráficos muy detallados y avanzados, aunque pueda requerir más código que algunas bibliotecas más modernas. Puede combinarse con Plotly para añadir interactividad.
- **Integración con otras Bibliotecas:** Se integra con otras bibliotecas de Python como NumPy, Pandas y SciPy, lo que permite manipular y visualizar datos fluidamente.

Figura 2.4: Logo de Matplotlib

2.1.6 SKLEARN

También comúnmente conocida como scikit-learn, es una biblioteca de Machine Learning en Python que ofrece herramientas sencillas y eficientes para el análisis de datos. Es muy usada en la comunidad de Python gracias a su facilidad de uso.

- **Variedad de Algoritmos:** Incluye desde regresiones lineales hasta técnicas de ensamblado más avanzadas y reducción de dimensionalidad.
- **Documentación y Comunidad:** sklearn tiene una comunidad activa que proporciona todo tipo de ejemplos para facilitar el entendimiento e implementación.
- **Integración con otras Bibliotecas:** Se integra fácilmente con otras bibliotecas de Python como pueden ser NumPy, Pandas y MAtplotlib, lo que hace que se pueda ofrecer una visualización muy completa y eficiente.

Figura 2.5: Logo de SkLearn

2.1.7 *TENSOR FLOW*

Tensor Flow es una biblioteca de código abierto que ha sido desarrollada por Google para el cálculo numérico y Machine Learning. Es muy conocida por su gran capacidad de construcción de redes neuronales profundas, lo que la hace muy útil para proyectos de Deep Learning.

- **Gráficos Computacionales:** Ofrece la creación de gráficos computacionales que pueden describir cómo los datos fluyen a través de una serie de operaciones matemáticas.
- **Escalabilidad:** Diseñado para ser escalable permitiendo el entrenamiento de modelos en muchas GPUs y distribuidos en varios dispositivos, mejorando significativamente el rendimiento y velocidad de entrenamiento.
- **Integración:** Se integra fácilmente con Keras, otra popular biblioteca de Deep Learning, y con otras herramientas de visualización como por ejemplo TensorBoard.

Figura 2.6: Logo de TensorFlow

CAPÍTULO 3. ESTADO DE LA CUESTIÓN

Este Trabajo de Fin de Grado explora tecnologías avanzadas cruciales para el mantenimiento predictivo y el análisis de datos. Se centra en algoritmos de aprendizaje automático y técnicas sofisticadas para manejar grandes cantidades de datos. Estas metodologías facilitan la identificación temprana de posibles fallos en sistemas críticos, lo que mejora la eficiencia, reduce los costos operativos y aumenta la seguridad. El estudio ofrece una visión detallada del estado actual de estas tecnologías, examinando los fundamentos y la evolución histórica del aprendizaje automático y su aplicación en el mantenimiento predictivo para optimizar la fiabilidad y disponibilidad de equipos industriales.

3.1 HISTORIA DEL MACHINE LEARNING

Aunque el Machine Learning parece un campo reciente, sus orígenes se remontan a mediados del siglo XX. Su desarrollo ha estado estrechamente ligado a la evolución de la Inteligencia Artificial, reflejando décadas de investigación en estadística, probabilidad y algoritmos.

La idea de que las máquinas pudieran aprender y adaptarse a través de la experiencia fue planteada por primera vez por Alan Turing en 1950 en su artículo "Computing Machinery and Intelligence". En este trabajo, Turing planteó la famosa pregunta "¿Pueden pensar las máquinas?" Se introdujo el concepto de la prueba de Turing, una evaluación de la capacidad de una máquina para exhibir un comportamiento inteligente similar al de un ser humano mediante la comparación de respuestas.

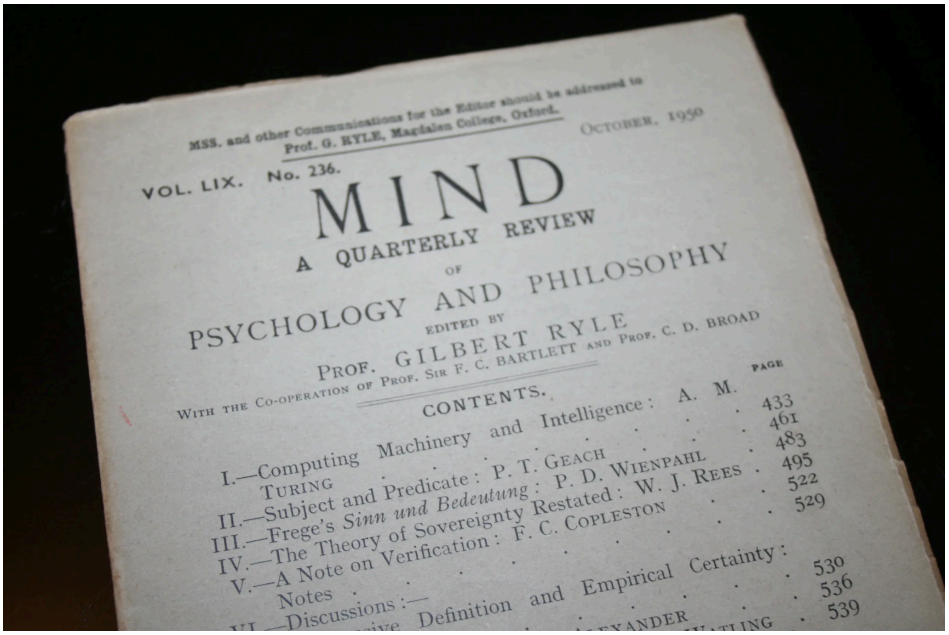


Figura 3.1: Copia de la Primera Edición del Artículo de Alan Turing sobre Inteligencia Artificial

El término "Machine Learning" fue acuñado por Arthur Samuel en 1959. Samuel, trabajando en IBM, desarrolló uno de los primeros programas con capacidad de aprendizaje: un programa de juego de damas que mejoraba con cada partida jugada contra humanos, demostrando que las máquinas podían aprender de los datos y mejorar sus habilidades basándose en experiencias previas.

En las décadas de 1960 y 1970, el Machine Learning se centró en los sistemas expertos, que intentaban codificar el conocimiento humano para que los ordenadores pudieran resolver problemas específicos. Sin embargo, estos sistemas se limitaban por la cantidad de conocimiento que los expertos podían formalmente codificar.

En la década de 1980, el desarrollo de algoritmos como el Backpropagation para redes neuronales marcó un cambio hacia métodos que permitían a los ordenadores aprender de grandes volúmenes de datos. Este cambio fue facilitado por el aumento del poder

computacional y la disponibilidad de más datos, permitiendo el desarrollo de patrones más complejos.

Con el renacimiento de las redes neuronales a principios de los 2000, especialmente con el desarrollo del Deep Learning, se alcanzó un punto de inflexión en el Machine Learning. Investigadores como Geoffrey Hinton, Yann Lecun y Yoshua Bengio lideraron avances en algoritmos capaces de entrenar redes profundas, mejorando significativamente tareas como la visión por ordenador y el procesamiento del lenguaje natural. El éxito de AlexNet en 2012, una arquitectura de red neuronal profunda que utilizaba GPUs para acelerar el entrenamiento de modelos complejos, demostró el potencial del Deep Learning al lograr una mejora significativa en la precisión del reconocimiento de imágenes, estableciendo un nuevo estándar en el campo.

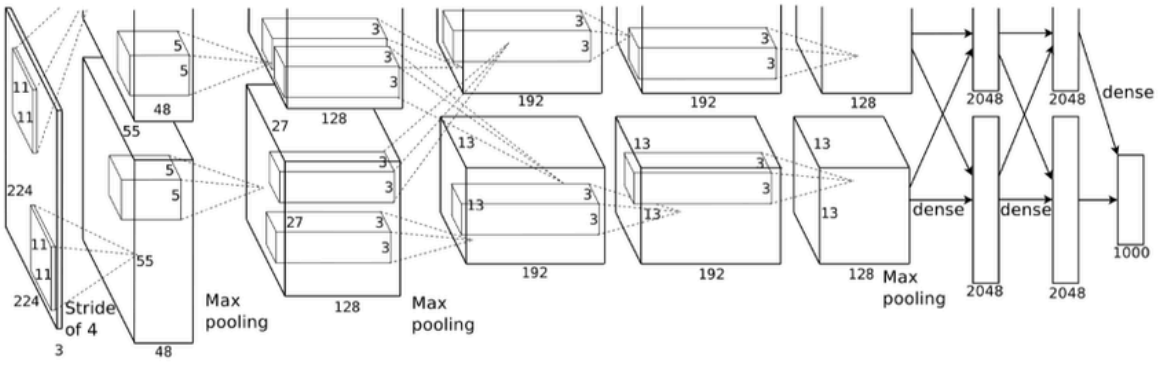


Figura 3.2: Arquitectura AlexNet

3.2 ESTRATEGIAS DE MANTENIMIENTO

3.2.1 MANTENIMIENTO PREVENTIVO (PM)

El **mantenimiento preventivo** comenzó a principios de los años 50 en industrias que reconocieron las ineficiencias del mantenimiento reactivo, que se lleva a cabo ante un fallo inesperado de una máquina que debe ser reparada. La idea era poder aumentar la

disponibilidad y fiabilidad del equipo al identificar y resolver problemas antes de que se pudiesen convertir en fallos.

"En la mitad del siglo XX, el mantenimiento pasó de ser una disciplina reactiva a una preventiva, impulsada por la necesidad de las industrias de aumentar la eficiencia y la productividad." — Kelly, Anthony (2006). Strategic Maintenance Planning.

Algunos de los componentes principales de este tipo de mantenimiento incluye tareas como limpieza, ajustes menores para asegurar el correcto funcionamiento de la maquinaria, realización de inspecciones regulares para detectar desgaste y supervisión de parámetros para identificar problemas potenciales.

3.2.2 MANTENIMIENTO DE CONDICIÓN (CBM)

El monitoreo de condición surgió a mediados del siglo XX con la adopción de prácticas de mantenimiento preventivo en la industria, aprovechando datos y haciendo análisis estadísticos para prevenir posibles paradas de funcionamiento que se convirtiesen en costos y peligros.

El objetivo del mantenimiento basado en la condición es el monitoreo y detección de fallas en el equipo para que el mantenimiento pueda ser programado proactivamente cuando sea necesario y no antes,

La implementación de este tipo monitoreo incluye la identificación de parámetros críticos, definición de condiciones operativas normales y límites, monitoreo continuo y análisis de datos y toma de decisiones.

3.2.3 MANTENIMIENTO PREDICTIVO (PDM)

La industria 4.0, con la integración del big data, el análisis avanzado y el IoT, ha dado lugar a este tipo de mantenimiento, el cual se centra en prever fallos del equipo antes de

que puedan ocurrir. Este enfoque evita costos innecesarios pudiendo programar el mantenimiento en el momento preciso. Su distinción está en la continua recolección de datos a través de distintos sensores que miden variables que se han definido críticas, usando posteriormente métodos estadísticos y algoritmos de Machine Learning para identificar tendencias y patrones.

Los datos de series temporales son cruciales para poder identificar tendencias y patrones que se traduzcan en el estado de salud real de la maquinaria, debiéndose cumplir las 5 V´s de éstos:

- **Volumen**: Gran cantidad de datos para su íntegra comprensión.
- **Velocidad**: Rápida generación y análisis de los datos.
- **Variedad**: Distintos formatos y fuentes
- **Veracidad**: Precisión y confiabilidad de los datos
- **Valor**: Datos que ofrecen ideas para guiar decisiones de mantenimiento

3.3 TIPOS DE APRENDIZAJE Y ALGORITMOS DE INTERÉS

El campo del Machine Learning es muy diverso, abordando distintos enfoques para prácticamente todo tipo de problemas que se pueda relacionar con datos. Para entenderlo de mejor manera, es fundamental conocer los diferentes tipos de aprendizaje que forman los algoritmos de Machine Learning. Estos métodos se clasifican normalmente según el tipo de retroalimentación que se devuelve al sistema de aprendizaje durante el entrenamiento. Se pueden categorizar en tres tipos principales: supervisado, no supervisado y por refuerzo. Cada uno emplea diferentes técnicas y enfoques para extraer patrones útiles, y la elección del tipo depende de la naturaleza de los datos y el problema que se debe resolver.

3.3.1 APRENDIZAJE SUPERVISADO

El aprendizaje supervisado es fundamental en Machine Learning, y se caracteriza por su capacidad para aprender y hacer predicciones a partir de datos etiquetados. El objetivo es construir un modelo que pueda generalizar a partir de la información que tiene disponible y pueda predecir resultados cuando reciba nuevos datos. Cada ejemplo con el que trabaja en el entrenamiento incluye un dato de entrada y una salida esperada o deseada. Se trata de que el modelo aprenda una función para producir la salida adecuada para una entrada dada, ajustando los parámetros para lograr minimizar la diferencia entre valores reales y predicciones lo máximo posible, práctica que se conoce como "minimización de la función de pérdida". Este primer tipo de aprendizaje es esencial en campos como la evaluación de riesgos crediticios, el mantenimiento predictivo en diversas industrias y la detección temprana de enfermedades en pacientes. Sin embargo, hay ciertos desafíos que tiene que enfrentar como lo son el sobreajuste y el subajuste, además de mantener una buena calidad y cantidad de los datos etiquetados con los que se trabaja.

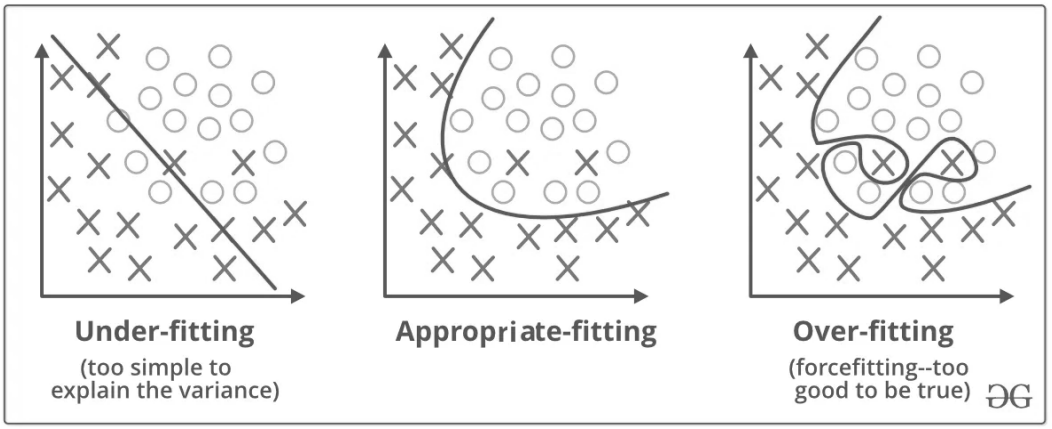


Figura 3.3: Diagrama Representando Subajuste y Sobreajuste

Las tareas de aprendizaje supervisado puede dividirse en dos distintas categorías:

- **Clasificación:** Problemas que tienen etiquetas categóricas, en los que el objetivo es asignar una entrada a una de las categorías predefinidas. Muy común en diagnósticos de pacientes, clasificando como "enfermo" o "no enfermo".

- **Regresión:** Etiquetas con valores continuos, donde el modelo trata de predecir un valor numérico en cambio de categórico, basándose en las entradas. Este tipo de aprendizaje supervisado es el que se va a estar usando a lo largo de todo el proyecto.

3.3.1.1 Regresión Lineal

Su propósito es establecer un modelo para la relación entre un cierto número de características y una variable objetivo continua. Dado un conjunto de puntos, el algoritmo de regresión tratará de establecer un modelo para ajustar la relación de dependencia entre una característica específica independiente de la (X) y el valor "resultado" correspondiente (Y), minimizando el error residual, es decir, la distancia vertical entre el punto y la propia línea elegida.

$$Y = mX + b$$

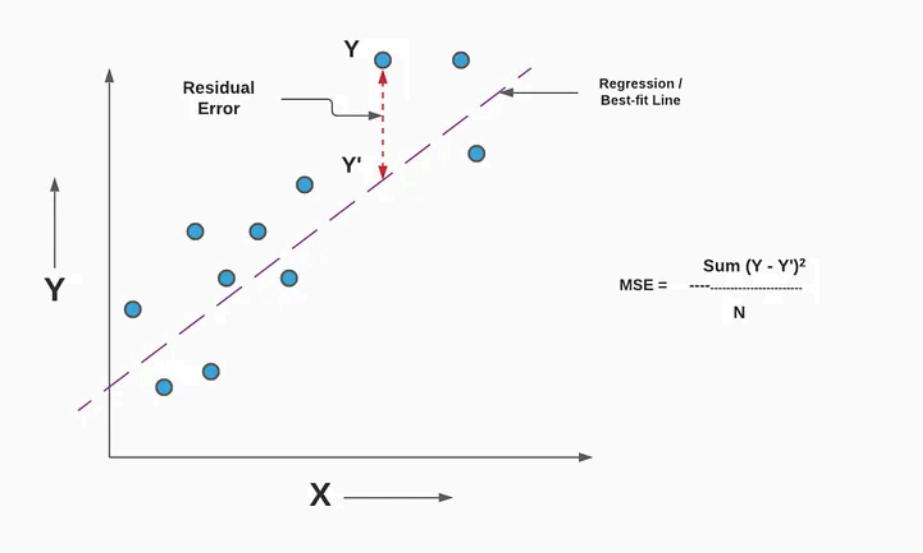


Figura 3.4: Diagrama Explicación de la Regresión Lineal

3.3.1.2 Regresión Logística

Tipo de análisis de regresión usado para predecir el resultado de una variable categórica en función de las variables independientes. Se podría decir que es una regresión logística adaptada a cuando la variable Y es binaria y categórica en cambio de numérica.

$$f(x) = \frac{1}{1 + e^{-x}}$$

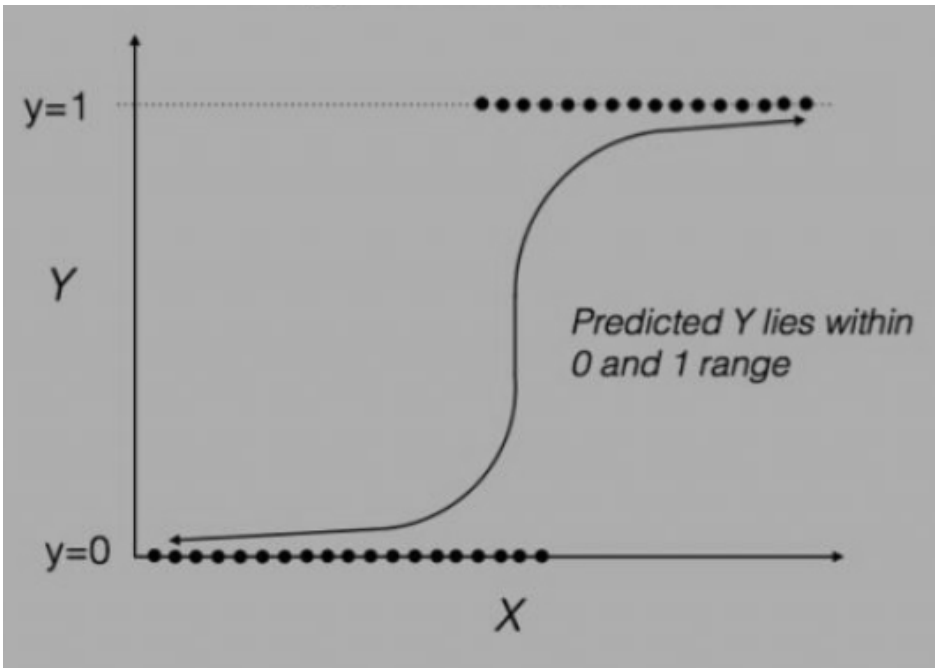


Figura 3.5: Diagrama Explicación de la Regresión Logística

3.3.1.3 Árboles de Decisión

Un árbol de decisión es un método no paramétrico usado para la clasificación. Su objetivo es crear un modelo que pueda predecir el valor de una variable objetivo aprendiendo reglas de decisión simples a partir de las características de los datos. Se crean a partir de un proceso de división binaria para maximizar la pureza de los nodos finales mediante la ganancia de información.

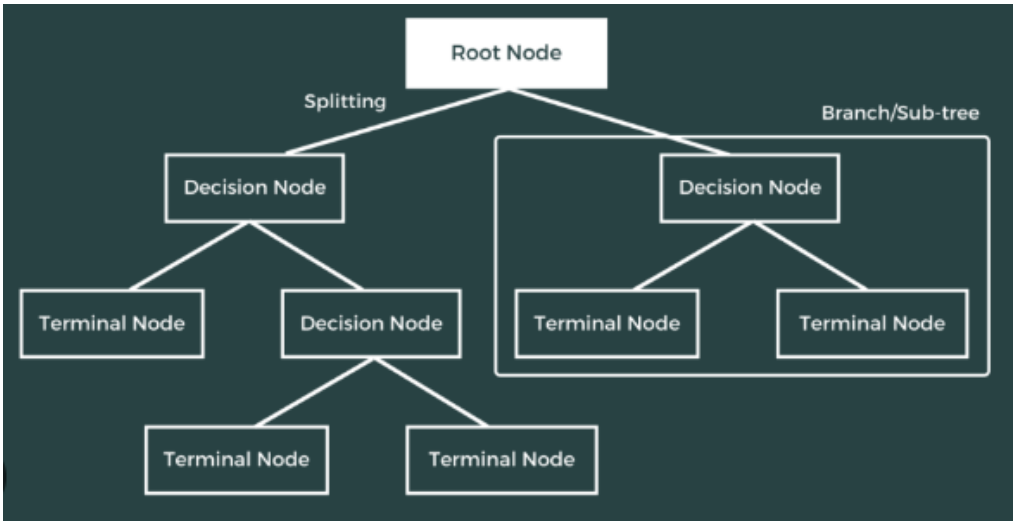


Figura 3.6: Diagrama Explicación de un Árbol de Decisión

3.3.1.4 SVM (Support Vector Machines)

Su objetivo es crear una línea o una especie de frontera de decisión que pueda separar de manera efectiva un conjunto de datos en diferentes clases. Cuando se logra establecer esta frontera, se pueden clasificar nuevos ejemplos en las clases apropiadas con relativa facilidad. Esta frontera de decisión se conoce como hiperplano, y el desafío radica en trazarlo precisamente.

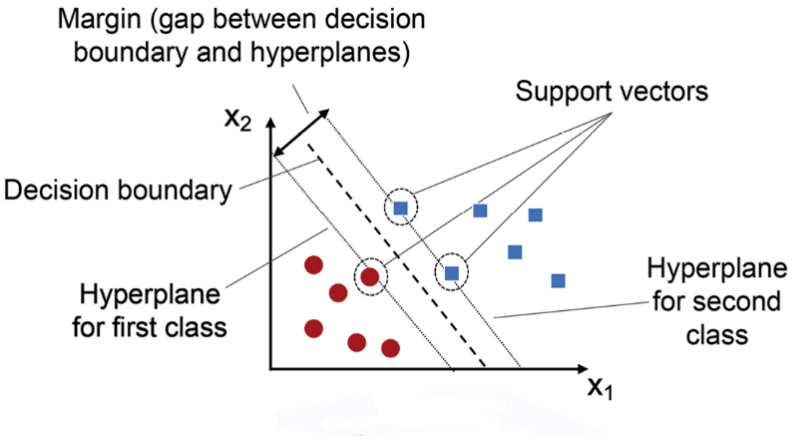


Figura 3.7: Diagrama Explicación de SVM

3.3.2 *APRENDIZAJE NO SUPERVISADO*

En el aprendizaje no supervisado, a diferencia del supervisado, los modelos son entrenados usando datos sin etiquetas, es decir, sin respuestas correctas o deseadas aparentes. El objetivo es descubrir patrones, agrupaciones o alguna estructura común dentro de los datos sin tener que usar intervención externa. Por esta razón, la validación y evaluación de estos modelos se vuelve mucho más complicada debido a la subjetividad y a la ausencia de etiquetas.

Las métricas de similitud son muy importantes en este tipo de aprendizaje, ya que también lo son para técnicas como la reducción de dimensionalidad y el clustering. Su elección dependerá de la propia naturaleza de los datos y del objetivo.

Las tareas de aprendizaje supervisado puede dividirse en dos distintas categorías:

- **Clustering:** Se trata de la división de puntos en grupos en los que los puntos de datos dentro de cada grupo son más similares entre sí que los de otros grupos. Es muy útil para descubrir tipos de comportamientos parecidos ocultos y segmentar poblaciones.
- **Reducción de Dimensionalidad:** Aborda el conocido concepto de "la maldición de la dimensionalidad", en el que los datos se ven afectados al incrementar la cantidad de dimensiones. Recoge las técnicas que tratan de reducir el impacto de esto.

3.3.2.1 K-Means

Tiene como tarea dividir el conjunto de datos en "K" clusters, asignando cada punto al centroide más cercano, basándose en la distancia euclidiana, la cual se calcula como la raíz cuadrada de la suma de las diferencias al cuadrado entre las coordenadas de los puntos.

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

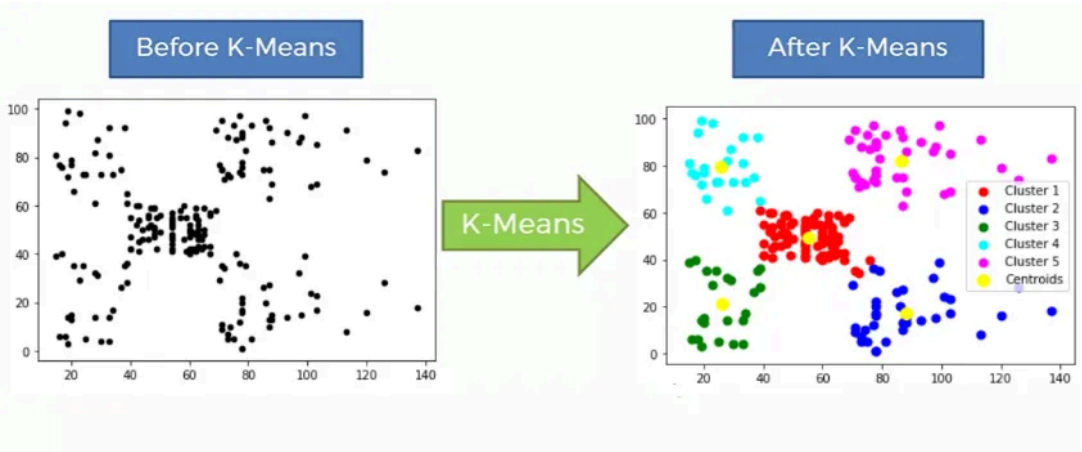


Figura 3.8: Antes y Después del Proceso K-Means

3.3.2.2 Clustering Jerárquico

Consiste en crear una jerarquía de clústeres, pudiendo ser de forma divisiva, la cual divide un cluster grande en otros más pequeños, o aglomerativa, que combina clústeres pequeños en otros más grandes.

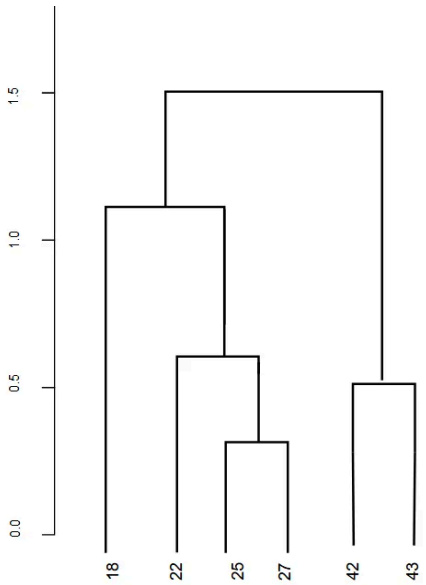


Figura 3.9: Dendograma del Clustering Jerárquico

3.3.2.2 PCA (Principal Component Analysis)

PCA trata de resumir los datos encontrando combinaciones lineales de las características del dataset, lo que puede imaginarse cómo tomar varias fotografías de un objeto en 3D, y luego ordenarlas de la más representativa a la menos representativa.

En la Figura 3.10, se puede observar como se muestran dos componentes principales de un espacio bidimensional, indicando que la componente 1 (PC1) proporciona mayor representatividad de los datos originales.

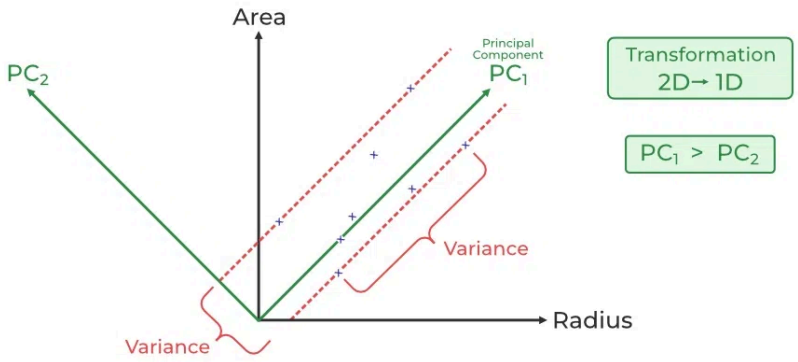


Figura 3.10: Dendograma Ejemplo de PCA

3.3.2 APRENDIZAJE POR REFUERZO

El tercer tipo de aprendizaje, por refuerzo, es un método en el que un agente aprende a tomar decisiones a través de cierta interacción con su entorno. El agente va recibiendo retroalimentación en forma de penalizaciones o recompensas según sus acciones van siendo correctas o erróneas, lo que le permite aprender comportamientos óptimos mediante ensayo y error. Este método es especialmente adecuado para entornos cambiantes donde los sistemas tradicionales en reglas dadas no son tan efectivos.

Básicamente, el aprendizaje por refuerzo se inspira en cómo los seres vivos aprenden en el mundo real de sus experiencias. Similar a un ratón que aprende el camino correcto en un

laberinto en busca de un queso, un agente de aprendizaje por refuerzo explora distintas estrategias y ajusta su comportamiento.

Algunos de los componentes claves son:

- **Entorno**: El sistema o proceso con el que el Agente se va relacionando.
- **Agente**: El encargado de tomar decisiones según interactúa con el entorno.
- **Acciones**: Las posibles decisiones que el agente puede realizar.
- **Estados**: Las diferentes situaciones en las que el entorno puede encontrarse.
- **Recompensas**: La retroalimentación que el agente recibe en función de sus acciones, lo que le ayudará a aprender.

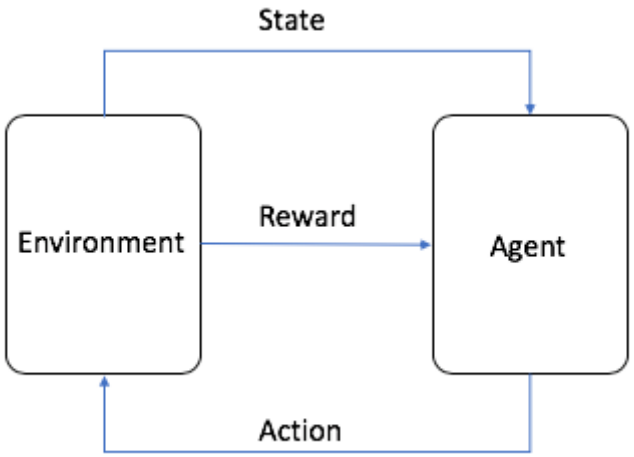


Figura 3.11: Diagrama de los Componentes del Aprendizaje por Refuerzo

Finalmente, a modo de conclusión de este apartado, en la Figura 3.12 se pueden observar las diferencias a grandes rasgos entre los 3 tipos de aprendizaje que se han revisado.

Criteria	Supervised ML	Unsupervised ML	Reinforcement ML
Definition	Learns by using labelled data	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of data	Labelled data	Unlabelled data	No – predefined data
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Supervision	Extra supervision	No supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

Figura 3.12: Tabla con las Diferencias entre los Tipos de Aprendizaje

3.4 DEEP LEARNING

El Deep Learning es una subcategoría del Machine Learning. Mientras que el Machine Learning usa algoritmos para realizar una tarea sin ser explícitamente programado para hacer predicciones o desarrollar un modelo, el Deep Learning enseña un modelo complejo usando algoritmos que están inspirados en la manera de pensar del cerebro humano. Esto permite predecir o clasificar datos no estructurados como texto, documento o imágenes. La red neuronal artificial es la unidad más simple, donde se alimenta un conjunto de datos en los nodos de entrada y se desarrolla un modelo complejo que proporciona un output en el nodo de salida. Este proceso implica el ajuste de los pesos y los sesgos en los nodos ocultos, término previamente presentado y conocido como “backpropagation”.

En la Figura 3.13, se puede observar una red neuronal artificial con varias capas: de entrada, varias ocultas y una capa de salida. Las líneas que conectan los nodos representan los pesos que son ajustados durante el proceso de backpropagation, y cada nodo en una

capa recibe entradas de los nodos de la capa anterior, pasando la salida a los nodos de la siguiente capa.

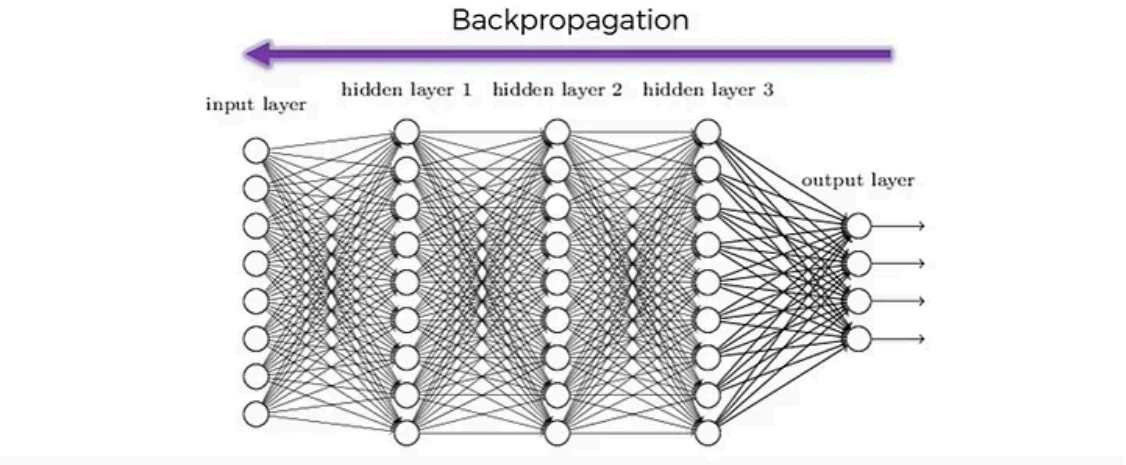


Figura 3.13: Tabla con las Diferencias entre los Tipos de Aprendizaje

Cada neurona tiene sus funciones de activación, que definen su salida dada una entrada concreta. Existen diferentes funciones según las características particulares que afectan la convergencia en el aprendizaje de no linealidades, como la Sigmoide, Tanh o ReLu. Durante el entrenamiento el objetivo es minimizar la función de pérdida, que mide la diferencia entre la salida predicha y los valores de la entrada reales. Algunos algoritmos comunes para la optimización de los pesos de la red son el descenso del gradiente estocástico y Adam.

3.4.1 Redes Neuronales Convolucionales

Son una clase de redes neuronales profundas especialmente eficientes en el procesamiento y análisis de datos que tienen estructura en cuadrícula, como por ejemplo las imágenes. Las CNN han transformado sobre todo el campo del reconocimiento de imágenes y se usan ampliamente en clasificación, detección de objetos y reconocimiento facial.

Algunos de los conceptos que las rodean más importantes son:

- **Capas convolucionales**: Estas capas aplican filtros de convolución a la entrada para poder extraer características locales como bordes, texturas y patrones. Cada filtro produce un mapa de características al recorrer toda la imagen.
- **Capas de Pooling**: Reducen la dimensionalidad de los mapas de características, intentando preservar las características más importantes y disminuyendo la carga de computación. También conocidas como capas de Downsampling.
- **Capas densas**: Después de varias capas de pooling, las características extraídas se introducen en una o más capas densas para realizar la clasificación o regresión final. Cada neurona se conecta a todas las neuronas de la siguiente capa.
- **Funciones de Activación**: Se aplican después de las capas convolucionales y densas para introducir no linealidades en el modelo pudiendo así aprender patrones con más complejidad.

3.4.2 Redes Neuronales de Memoria a Largo Plazo (LSTM)

Las LSTM son una variante especial de las redes neuronales recurrentes, las cuales han sido diseñadas para modelar secuencias de datos a lo largo del tiempo. Cuentan con una arquitectura que les permite almacenar información y actualizarla durante largos periodos.

Algunos componentes clave:

- Células de memoria: Disponen de celdas de memoria capaces de retener información a lo largo de muchas etapas temporales.
- Puerta de entrada: Controla la información que se actualizará en la célula de memoria.
- Puerta de Olvido: Determina qué información en la celda de memoria debe ser eliminada.
- Puerta de Salida: Decide qué parte de la información almacenada en la célula de memoria se usará para generar la salida.
- Estados de Celda y Estados Ocultos: Las LSTM mantienen dos estados diferentes los cuales se actualizan en cada etapa temporal: el de celda, que transporta

información a lo largo de toda la secuencia y el oculto, que produce la salida en cada etapa temporal.

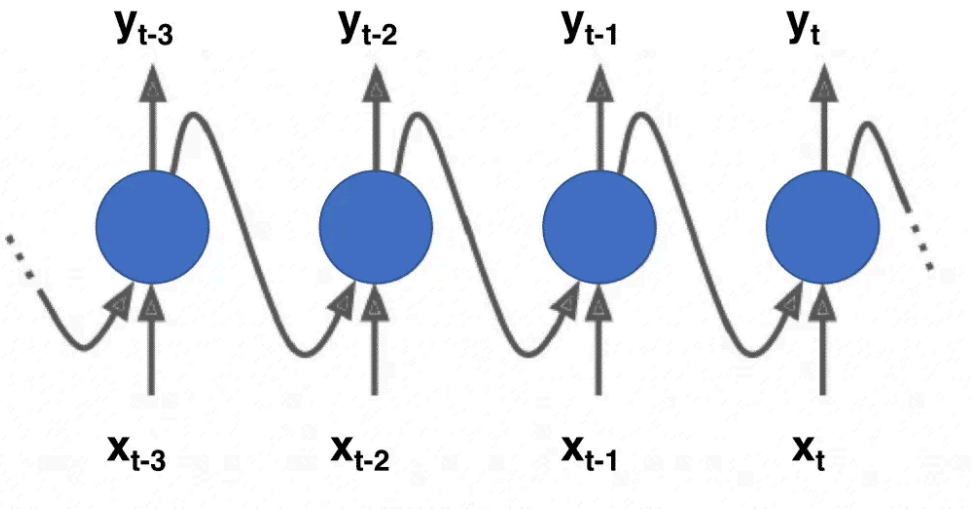


Figura 3.14: Diagrama de Redes Neuronales Recurrentes

CAPÍTULO 4. DEFINICIÓN DEL TRABAJO

4.1 PRESENTACIÓN DEL CONJUNTO DE DATOS

El conjunto de datos ***CMAPSS*** se basa en un sistema informático de alta fidelidad diseñado para poder simular el funcionamiento de un motor turbofán comercial de gran volumen, ofreciendo un modelo termodinámico completo del motor que puede replicar su funcionamiento con alta precisión. Además, este modelo incluye un componente atmosférico que tiene en cuenta varias condiciones operativas, como altitudes, velocidades relativas y temperaturas a nivel del mar.

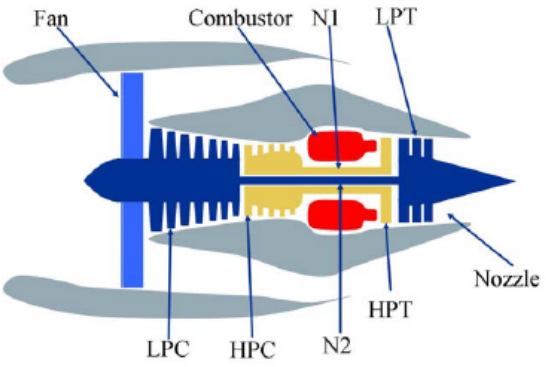


Figura 4.1: Diagrama Simplificado del Motor Simulado en C-MAPSS ["Damage Propagation Modelling.pdf"]

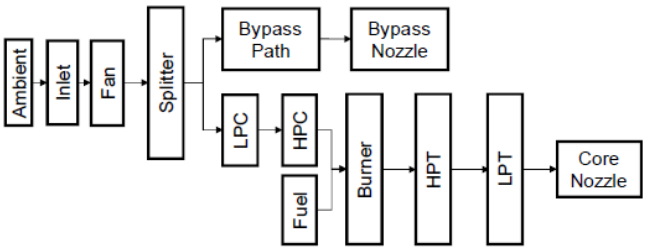


Figura 4.2: Conexión de Módulos para la Simulación ["Damage Propagation Modelling.pdf"]

El modelo es presentado inicialmente como un sistema acoplado de ecuaciones no-lineales, cuyas entradas se clasifican en condiciones operativas ($w(t)$), y parámetros del estado de salud ($\theta(t)$). Todos estos enfoques permiten simular el comportamiento de un motor turbofan exponiéndose a diferentes condiciones operativas, acercándose lo máximo posible a un escenario realista para recoger datos detallados para el desarrollo de modelos de mantenimiento predictivo.

Existen diferentes parámetros usados como referencias para representar un escenario concreto y el comportamiento esperado del motor.

Name	*Symbol*
Fuel flow	W_f
Fan efficiency modifier	fan_eff_mod
Fan flow modifier	fan_flow_mod
Fan pressure-ratio modifier	fan_PR_mod
LPC efficiency modifier	LPC_eff_mod
LPC flow modifier	LPC_flow_mod
LPC pressure-ratio modifier	LPC_PR_mod
HPC efficiency modifier	HPC_eff_mod
HPC flow modifier	HPC_flow_mod
HPC pressure-ratio modifier	HPC_PR_mod
HPT efficiency modifier	HPT_eff_mod
HPT flow modifier	HPT_flow_mod
LPT efficiency modifier	LPT_eff_mod
HPT flow modifier	LPT_flow_mod

Figura 4.3:Condiciones Operativas w(t), C-MAPSS ["Damage Propagation Modelling.pdf"]

T48 (EGT)	Total temperature at HPT outlet	°R
SmFan	Fan stall margin	--
SmLPC	LPC stall margin	--
SmHPC	HPC stall margin	--

Figura 4.4:Parámetros del Estado de Salud θ(t), C-MAPSS ["Damage Propagation Modelling.pdf"]

Los outputs del modelo, se adquieren a través de 21 sensores, los cuales miden el estado del motor y la información de 3 configuraciones operativas. Estas 24 señales de series temporales tienen ruido al estar representando datos reales, y muestran una degradación hasta que el motor falla. Parte de la información de los sensores será útil para indicar una tendencia de degradación directa al motor, mientras que otras serán inútiles. Los significados físicos de los sensores se pueden observar en la Figura 4.5.

Sensor Number	Symbol	Description	Units	Trend
1	T2	Total temperature at fan inlet	°R	~
2	T24	Total temperature at LPC outlet	°R	↑
3	T30	Total temperature at HPC outlet	°R	↑
4	T50	Total temperature at LPT outlet	°R	↑
5	P2	Pressure at fan inlet	psia	~
6	P15	Total pressure in bypass-duct	psia	~
7	P30	Total pressure at HPC outlet	psia	↓
8	Nf	Physical fan speed	rpm	↑
9	Nc	Physical core speed	rpm	↑
10	epr	Engine pressure ratio	–	~
11	Ps30	Static pressure at HPC outlet	psia	↑
12	Phi	Ratio of fuel flow to Ps30	pps/psi	↓
13	NRf	Corrected fan speed	rpm	↑
14	NRc	Corrected core speed	rpm	↓
15	BPR	Bypass ratio	–	↑
16	farB	Burner fuel-air ratio	–	~
17	htBleed	Bleed enthalpy	–	↑
18	Nf_dmd	Demanded fan speed	rpm	~
19	PCNfR_dmd	Demanded corrected fan speed	rpm	~
20	W31	HPT coolant bleed	lbm/s	↓
21	W32	LPT coolant bleed	lbm/s	↓

Figura 4.5:Lista de Significados Físicos de los Sensores C-MAPSS

El conjunto de datos CMAPSS cuenta con 4 subconjuntos, como se puede ver en la Figura 4.6. Cada uno está configurado de manera diferente en cuanto a número de motores, condiciones operativas y tipos de falla. *FD001* contiene datos de una sola condición operativa (nivel del mar) y un modo de falla (degradación del HPC), *FD002* contiene múltiples condiciones operativas y un modo de falla, *FD003* es parecido a FD001 pero con dos modos de falla y *FD004* combina múltiples condiciones operativas con dos modos de falla. Cada subconjunto de datos proporciona también los datos de entrenamiento, datos de

prueba y RUL, en el que "*train_FD00X*" contiene datos de entrenamiento desde el inicio hasta la falla completa del motor, "*test_FD00X*" contiene datos de prueba de motores hasta cierto tiempo no conocido antes de la falla completa, y "*RUL_FD00X*" con los valores reales de la RUL, que se usarán para evaluar la precisión de predicción de los modelos utilizados.

Dataset	FD001	FD002	FD003	FD004
Number of engines	100	260	100	249
Number of training samples	20,631	53,579	24,270	61,249
Number of test samples	100	259	100	248
Number of the data column	26	26	26	26
Average life span (cycles)	206	206	247	245
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

Figura 4.6:Información sobre Subconjuntos FD00X

Este conjunto comprende simulaciones completas de vuelos de sistemas turbofán que replican datos registrados de vuelos de aviones comerciales, cubriendo condiciones como ascenso, crucero y descenso.

El procedimiento descrito por los autores para generar estos datos es el siguiente:

1. **Definición de Datos de Vuelo:** Los datos de vuelo se establecen basándose en registros obtenidos de un avión comercial.
2. **Imposición de Degradación:** Se aplica degradación a los componentes del motor para simular un proceso de deterioro.
3. **Simulación de Vuelo Degradado:** Se realiza una simulación del vuelo con la degradación impuesta, considerando la degradación comentada.
4. **Evaluación de la Condición de Salud:** Se hace una evaluación de la condición de salud del motor, permitiendo así que la unidad continúe volando con una degradación que crece hasta que el índice de lud del motor llegue a cero (HI = 0), lo que marca el fin de su vida útil.

5. **Adición de Ruido de Sensor:** Se añade ruido del sensor a la respuesta de motor simulado, añadiendo variaciones de medición realistas para un simulación más realista.

Una última aclaración en esta sección respecto a la variable a predecir, RUL. La columna de RUL se genera usando un enfoque inverso del tiempo, restando la unidad de tiempo actual a la unidad de tiempo máxima. Haciendo este cálculo para cada motor, se obtienen los valores de RUL de cada registro.

Como se ha comentado previamente, los valores reales de RUL están disponibles en el conjunto de datos *RUL_FD00X,* y se utilizarán para evaluar el rendimiento de los modelos predictivos.

4.2 EDA (EXPLORATORY DATA ANALYSIS) OF FD001

En este proyecto, se trabajará y realizarán los distintos modelos de predicción basándose en el primer conjunto de datos FD001, dando espacio a mejoras para futuros trabajos. Por esta razón, se analizarán los archivos: train_FD001, test_FD001 y RUL_FD001.

Primero, se hace un preprocesamiento de los datos:

Tras hacer una inicial carga de librerías y datos del conjunto de entrenamiento, en la Figura 4.7 se puede observar la estructura inicial del conjunto.

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

Figura 4.7: Estructura Resultado de la instancia Head() sobre el Conjunto Train

Como se comentó previamente, se puede ver que el conjunto de datos tiene 26 columnas.

A continuación, se revisarán cuántos motores existen, y después de cuántos ciclos falla cada uno (midiendo los valores únicos de las columnas 1 y 2).

```
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100]
```

Figura 4.8:Output del estudio de Valores Únicos de la columna 1

El número de valores únicos en la primera columna indica el total de motores estudiados en este conjunto de datos, por lo tanto, viendo la Figura 4.8, hay 100 motores en total. Por otro lado, en la Figura 4.9 se pueden observar los ciclos que se recorrieron previo al fallo total de cada motor. Por ejemplo, el motor 1 falló después de 192 ciclos, y así sucesivamente.

```
[192 287 179 189 269 188 259 150 201 222 240 170 163 180 207 209 276 195
 158 234 195 202 168 147 230 199 156 165 163 194 234 191 200 195 181 158
 170 194 128 188 216 196 207 192 158 256 214 231 215 198 213 213 195 257
 193 275 137 147 231 172 185 180 174 283 153 202 313 199 362 137 208 213
 213 166 229 210 154 231 199 185 240 214 293 267 188 278 178 213 217 154
 135 341 155 258 283 336 202 156 185 200]
```

Figura 4.9:Output del estudio de Valores de la Columna de Ciclos por cada Motor de Train

Existen dos maneras de pensar sobre el degradamiento de la RUL: el modelo de degradación lineal y el modelo de degradación lineal por tramos. El modelo de degradación lineal por tramos hace referencia a la no linealidad desde el inicio del funcionamiento del motor. Funciona asignando un número RUL inicial fijo, y cuando el

número de ciclos llegue a ese número, empezará a seguir un modelo de degradación lineal normal. En la Figura 4.10 se puede observar una representación a medida que avanza el tiempo de los dos tipos de degradación, la lineal y la lineal a tramos respectivamente.

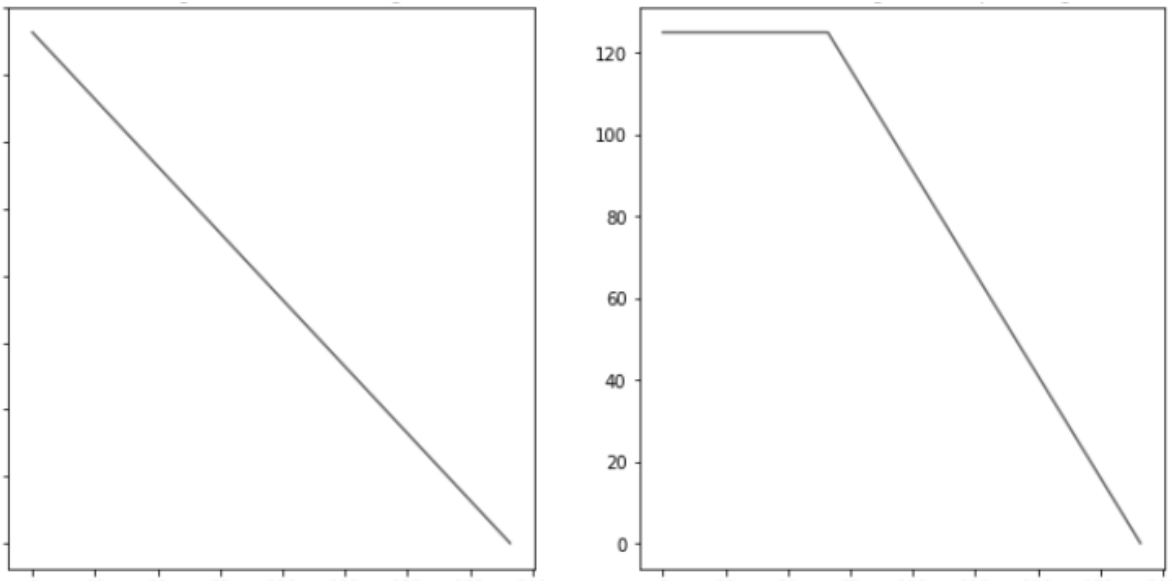


Figura 4.10:Output del estudio de Valores de la Columna de Ciclos por ada Motor

A continuación, se hace un análisis del conjunto de prueba.

Los datos de prueba, como se comentó en el apartado de presentación de los datos, están disponibles durante un número arbitrario de ciclos aleatorios por cada motor.

```
Ciclos totales por motor en conjunto TEST:
[ 31  49 126 106  98 105 160 166  55 192  83 217 195  46  76 113 165 133
 135 184 148  39 130 186  48  76 140 158 171 143 196 145  50 203 198 126
 121 125  37 133 123 156 172  54 152 146  73  78 303  74 144 189 164 121
 113 136 160 176  94 147 159 232 155 168  71 147  71 187  54 152  68 131
 112 137  88 205 162  72 101 133 213 162  73 172  34 110  56  68 177 146
 234 150 244 133  89  97 134 121  97 198]
```

Figura 4.11:Output del estudio de Valores de la Columna de Ciclos por cada Motor de Test

Como se puede observar en la Figura 4.11, se confirma que en el conjunto de Test, los ciclos de cada motor acaban aleatoriamente antes de la falla completa.

```
[112  98  69  82  91  93  91  95 111  96  97 124  95 107  83  84  50  28
  87  16  57 111 113  20 145 119  66  97  90 115   8  48 106   7  11  19
  21  50 142  28  18  10  59 109 114  47 135  92  21  79 114  29  26  97
 137  15 103  37 114 100  21  54  72  28 128  14  77   8 121  94 118  50
 131 126 113  10  34 107  63  90   8   9 137  58 118  89 116 115 136  28
  38  20  85  55 128 137  82  59 117  20]
```

Figura 4.12:Output del estudio de Valores de la Columna de Ciclos por cada Motor de Test

Por otro lado, en la Figura 4.12, se obtienen las RUL´s reales de cada motor del conjunto Test.

Para estudiar la validez o utilidad de todos los sensores, se hace una visualización de las distribuciones de los sensores del conjunto haciendo uso de violin plots.

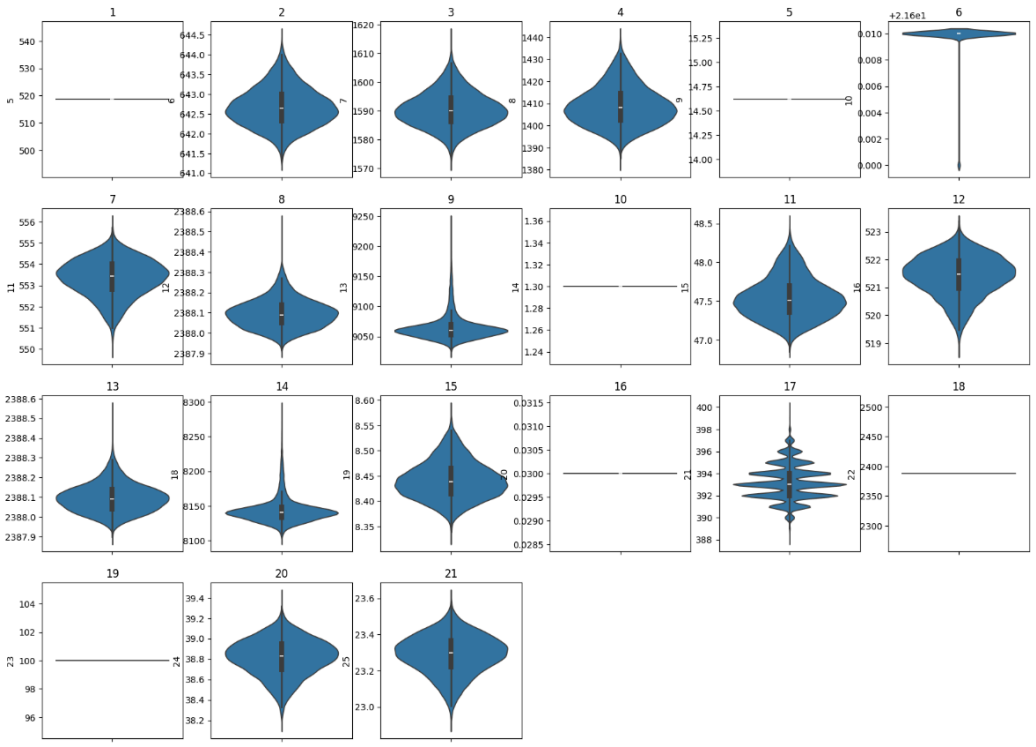


Figura 4.13:Violin Plots de la Distribución de los Sensores

Como se observa en la Figura 4.13, hay sensores que no aportan información al conjunto de datos, ya que su distribución es constante por tener siempre el mismo dato. Estos sensores son: 1, 5, 6, 10, 16, 18 y 19. Para todos los modelos que se lleven a cabo, como se trabajará con el mismo dataset FD001, se eliminarán estas columnas para nuestro análisis, ya que no ofrecen ningún tipo de información válida.

Atendiendo a la hora de generalizar bien de un algoritmo, es decir, tener buen rendimiento en datos nuevos que nunca ha visto, lo hará mejor si estos nuevos datos tienen una distribución parecida a los datos con los que se entrenó el modelo. Se puede decir que el modelo tendrá potencial buena generalización.

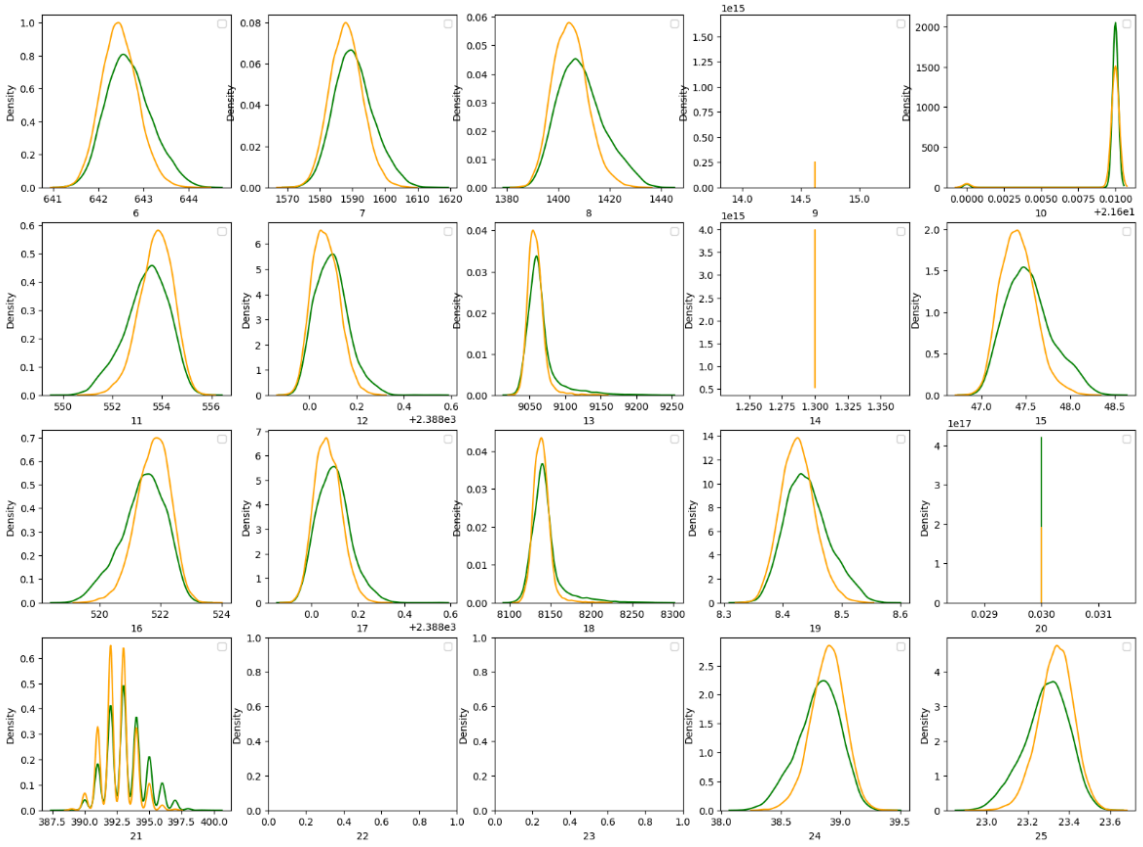


Figura 4.14:Gráficos de densidad de los Sensores

Cabe destacar, que en el conjunto de datos, a la hora de realizar la simulación lo más afín a la realidad posible, se ha añadido ruido matemático a las mediciones de los sensores. Este ruido trata de simular las condiciones reales en las que los sensores sufren variaciones por diferentes factores tanto externos como internos. Esto se conoce, y los modelos se entrenarán considerando este ruido, ya que se asume que no afectará el desempeño de un modelo bien entrenado, pero es importante considerar que en futuros trabajos sería interesante explorar cierto uso de filtros para poder mitigar el impacto de este ruido en los datos. Filtros de media móvil o métodos de suavizado para intentar mejorar la precisión del modelo eliminando componentes ruidosos. Por ahora se trabajará con estos datos ruidosos para evaluar la robustez de los modelos aun con condiciones ruidosas.

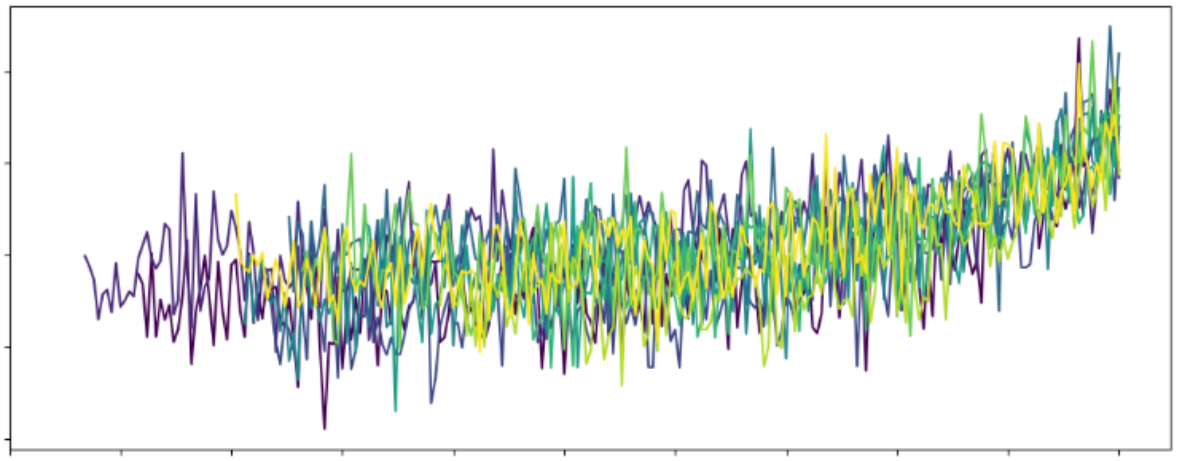


Figura 4.15:Muestra del Ruido de un Sensor

4.3 SELECCIÓN DE MODELOS

Una vez comentado esto, después del preprocesamiento de los datos, se ha decidido dividir la creación de los modelos en 2: modelo de regresión y modelo de redes neuronales. Se ha decidido hacer esta partición ya que permite una progresión lógica en la complejidad de los modelos usados. Empezando con regresiones siempre permite establecer una especie de base sólida usando técnicas más fácilmente interpretables, avanzando hacia modelos de redes neuronales que pueden ofrecer mayor precisión a la hora de capturar patrones dada su complejidad avanzada.

4.3.1 MODELOS DE REGRESIÓN

Los modelos que se basan en la regresión son fundamentales en el ámbito del Machine Learning supervisado para la predicción de valores continuos. En este proyecto, los modelos de regresión ayudarán a predecir la RUL de los motores basándonos en los datos preprocesados de los sensores de interés. Modelo a utilizar:

- **SVR:** útil para problemas donde la relación entre variables no es lineal, esta capacidad lo hace adecuado para analizar datos complejos de sensores.
- **Random Forest:** su capacidad para tratar datos con muchas características lo hace interesante, además de la resistencia frente al sobreajuste.
- **XGBoost:** eficiente y capaz de manejar grandes volúmenes de datos con varias características. Además, al tener capacidad de mejorar continuamente las predicciones corrigiendo errores lo hace muy interesante.

4.3.2 MODELOS DE REDES NEURONALES

Tras analizar el modelo de regresión que se llevará a cabo, el siguiente paso es hablar del modelo de redes neuronales. Estas redes, como se ha comentado previamente, son especialmente útiles a la hora de identificar patrones complejos y no lineales en los datos, lo que las convierte en las más adecuadas para problemas de predicción complejos.

Se usarán dos tipos principales de redes: Convolucionales de una dimensión (1D-CNN) y de Memoria a largo plazo (LSTM).

- **1D-CNN:** Las 1D-CNN aplican convoluciones a lo largo de una dimensión, ideales para analizar datos secuenciales como este caso de sensores. Detectan muy eficazmente patrones en series temporales, útil para predecir la RUL del motor.
- **LSTM:** Tipo de redes neuronales recurrentes (RNN), por lo que son robustas para la predicción de series temporales. En datos de sensores de motor, estas redes pueden aprender dependencias a largo plazo importantes para la RUL.

4.4 IMPLEMENTACIÓN DE LOS MODELOS ELEGIDOS

En este proyecto, se ha optado por llevar a cabo el análisis y estudio de los modelos de predicción usando el modelo de degradación por tramos debido a su capacidad para capturar comportamientos no lineales observados en el deterioro de los motores turbofan. Este modelo permite que se asigne un valor de RUL constante durante un periodo antes de comenzar a disminuir de manera lineal, representando realmente como muchas máquinas pasan de fases de funcionamiento estables a una rápida degradación debido a una falla desafortunada que va creciendo. Se quiere obtener predicciones precisas y robustas en escenarios y patrones de desgaste no uniformes, y además, la degradación por tramos facilita la modelización de sistemas complejos y la sensibilidad del modelo a adaptarse a cambios bruscos en el estado del motor, que es lo más parecido a la realidad.

Capturar este tipo de transiciones bruscas más precisamente es crucial para poder llegar a optimizar procesos industriales y reducir costos consecuencia de fallos inesperados y/o mantenimientos que no se han programado.

- MODELOS DE REGRESIÓN

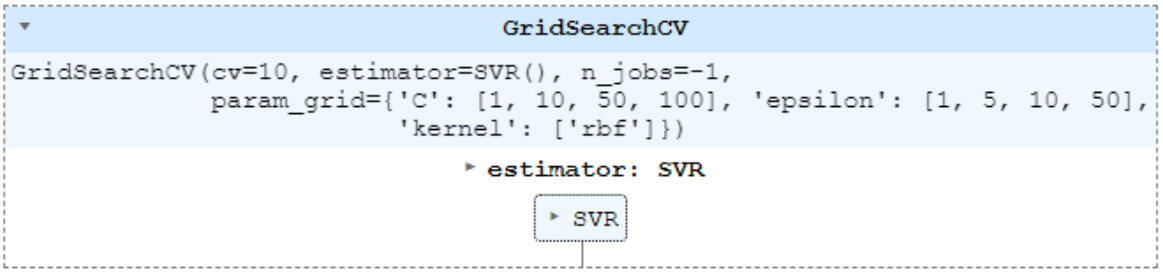
4.4.1 SVR

En el proceso de entrenamiento del modelo SVR es importante escalar los datos. Normalizarlos para que todos los atributos puedan estar dentro de un mismo rango proporcional. Se conoce que los modelos de Machine Learning funcionan mucho mejor cuando las características se normalizan y pasan a tener una escala similar entre ellos. Si no se hiciese el escalado, valores atípicos podrían ser muy determinantes para el cálculo.

En este caso, se usa StandardScaler() de SkLearn, ajustando para que los datos pasen a tener una media de 0 y una desviación estándar de 1, quitando la columna del identificador del motor antes del escalado, y volviendo a incorporarla después.

Para empezar teniendo alguna idea más sobre qué parámetros utilizar previos a llevar a cabo las predicciones, se hará uso de Grid Search. Se seleccionará el conjunto de datos que tenga la mejor performance con la validación cruzada como hiperparámetro.

La validación cruzada de estos hiperparámetros es computacionalmente muy densa, pero después de cierto tiempo esperando la validación cruzada, se obtiene esta combinación final.




```
{'C': 10, 'epsilon': 10, 'kernel': 'rbf'}
```

Figura 4.16:Output Grid Search

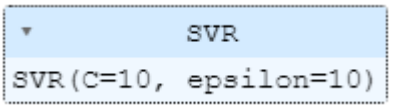


Figura 4.17:Output de Mejores Parámetros

- **C (parámetro de penalización):** Este hiperparámetro controla el equilibrio entre un margen que sea amplio y un error en la clasificación. Un valor alto de este hiperparámetro intenta clasificar de manera correcta absolutamente todos los puntos del conjunto de entrenamiento, lo que puede estrechar el margen y posiblemente acabar en un sobreajuste. C = 10 indica que el modelo es bastante estricto con los errores de entrenamiento, haciendo que las desviaciones de los puntos de datos de entrenamiento se penalizan fuertemente.

- **€ (Margen de Insensibilidad)**: El parámetro épsilon define la zona en la que las predicciones del SVR no incurren en ninguna penalización. Establece una especie de tolerancia, en otras palabras, dentro de la cual los errores no son considerados. 10 es un valor relativamente alto, por lo que el modelo permitirá más desviaciones en las predicciones, lo cual puede ser útil en escenarios como este en los que se espera una variabilidad natural en los datos.

En resumen, el modelo SVR con C=10 y ϵ=10 se ha configurado para ser muy estricto a la hora de minimizar los errores grandes. Esto puede ser adecuado para situaciones en las que se quiere evitar un sobreajuste y aceptar que pequeñas desviaciones en las predicciones sean tolerables. El modelo se entrenará para minimizar los errores grandes pero permitirá que los errores más pequeños no sean penalizados. Esto hace que la generalización sea mejor con presencia de ruido en los datos.

Es preciso señalar que las predicciones se han pensado para ser realizadas usando las últimas 5 mediciones por cada motor, Se considera que las mediciones más recientes

ofrecen la información que es más relevante y precisa sobre el estado actual de desgaste del motor, poniendo el foco en el comportamiento más crítico del motor justo antes de la falla.

Este enfoque se hace para que el modelo pueda reducir la influencia que pueden tener mediciones anteriores, las cuales podrían no reflejar condiciones más importantes y recientes a la falla del motor. Se usan 5 para no tener un número demasiado alto que pueda llevar a disminuir la relevancia de las mediciones que son más recientes, ni muy bajo.

```
RMSE (SVR):  19.738149033874976
MAE (SVR)):  14.783500418534095

S-score:  1350.6574511428255
```

Figura 4.18:Resultados SVR

Como se puede apreciar en la Figura 4.18, donde se reflejan los resultados de las métricas del modelo SVR, se ha añadido "S-score". Esto se debe a la necesidad de tener una tercera métrica que combine el RMSE y MAE para dar otro punto de vista.

El RMSE indica que en promedio las predicciones del RUL están a 19.74 unidades de los valores reales. Por otro lado, se obtiene un MAE de 14.78, y un S-score de 1350.66

En la Figura 4.19, se visualiza la capacidad del modelo SVR para predecir el RUL de los motores del conjunto de datos, destacando tanto las fortalezas a la hora de seguir la tendencia general como el margen de mejora de éstas predicciones.

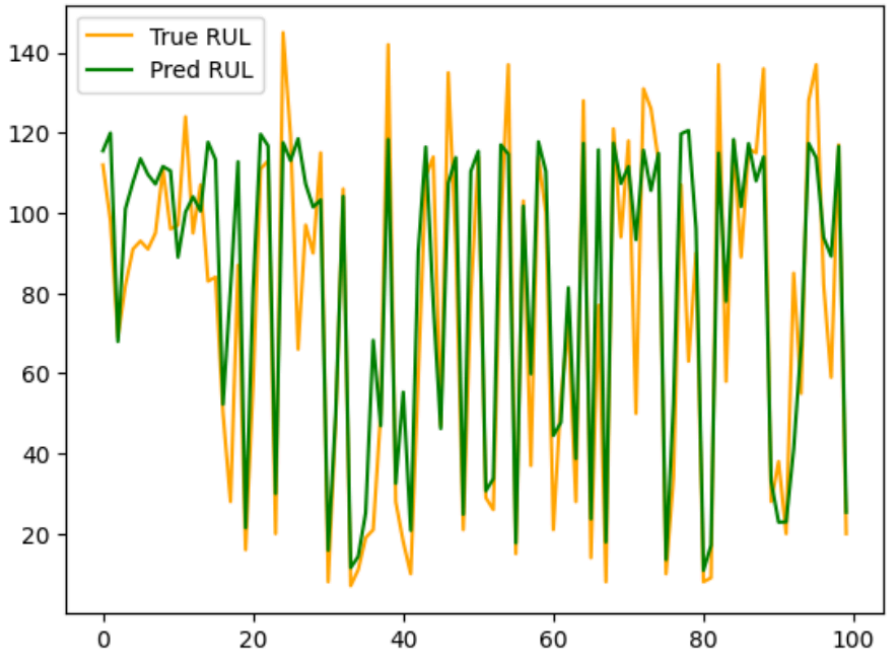


Figura 4.19:Gráfica de las Predicciones de Rul, con sus Valores Reales

4.4.2 RANDOM FOREST

A diferencia del primer modelo SVR, en este segundo modelo de Random Forest, se decide no escalar los datos con los que se van a trabajar. Esto se debe a que los árboles de decisión no requieren un previo escalado de los datos con los que se entrenará, por la forma en la que estos algoritmos funcionan.

Dado que estos algoritmos trabajan básicamente siguiendo una serie de comparaciones simples como si una característica es mayor que un umbral impuesto para poder dividir, o seleccionar aleatoriamente diferentes características, los hace inherentes a una normalización de los datos.

A continuación, se sigue el mismo procedimiento que en el modelo anterior, que es ejecutar un Grid Search con ciertos parámetros para poder encontrar el de mejor performance.

- "**n_estimators**": Se eligen valores a probar desde el 100 hasta el 300 dando saltos de 50, es decir, 100, 150, 200, 250, 300. Este hiperparámetro indica el número de árboles que tendrá el Random Forest.
- "**max_features**": Indica el número de características a considerar en la búsqueda de la mejor division. Se prueban "auto", "sqrt" y "log2". Sqrt toma la raíz cuadrada del número total de características para cada división, evitando el sobreajuste por usar pocas características. Log2 calcula el logaritmo en base 2 del número total de las características.

Tras hacer uso del GridSearch, debido a la carga computacional que esta búsqueda suele llevar, y por ende a un error, se decide rapidamente probar con dos modelos:

```
modelo_rf = RandomForestRegressor(n_estimators=150, max_features="sqrt",
                                  n_jobs=-1)

modelo_rf2 = RandomForestRegressor(n_estimators= 350, max_features = "sqrt",
                                   n_jobs = -1)
```

Figura 4.20: Comparativa 2 Modelos Random Forest

El primer modelo, con estimadores más bajos del rango con el que se intentó llevar a cabo el GridSearch, y el segundo modelo con 350, para ver cómo se comporta con pocos estimadores y con muchos.

```
RMSE modelo 1:  19.11920759073916

RMSE modelo 2:  19.075916353906162
```

Figura 4.21: Performance de los 2 Modelos de Random Forest

Como se puede apreciar en la Figura 4.21, ambos modelos tienen un rendimiento muy parecido. El modelo 2, con 350 estimadores, tiene un rendimiento un poco mejor pero la diferencia es muy pequeña (aproximadamente 0.043).

La similitud entre los modelos deja ver que el modelo Random Forest es consistente, pero una diferencia tan pequeña en el rendimiento usando más del doble del número de estimadores hace que el beneficio por el coste adicional en términos de recursos computacionales no sea rentable. Por esta razón, se decide seguir adelante con el modelo 1

```
RMSE final modelo 1:  19.29997888634424
MAE final modelo 1:  14.55948
S-score final modelo 1:  1121.0707633049433
```

Figura 4.22: Resultados Finales Modelo 2 Random Forest

En la Figura 4.22, comparando rápidamente con el modelo SVR anterior, se puede apreciar que el modelo de Random Forest ha mejorado ligeramente en el RMSE y MAE. Estas pequeñas diferencias de errores indican que el algoritmo Random Forest tiene un rendimiento algo mejor en la predicción de la RUL de los motores.

A parte, el S-Score es menor, de 1300 a 1100, por lo que Random Forest no solamente predice con mayor precisión, sino que tiene menos penalizaciones por desviaciones en las predicciones hechas por el modelo.

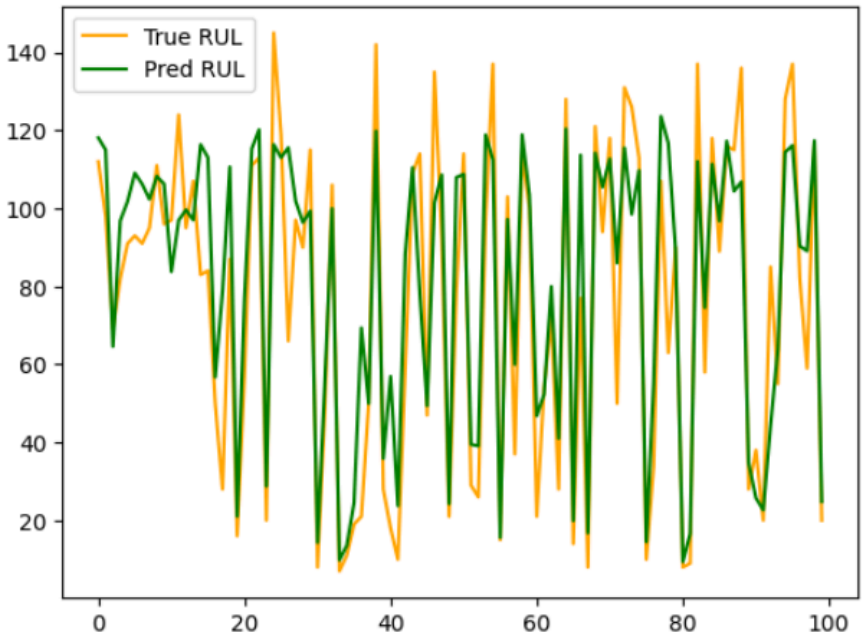


Figura 4.23: Gráfica de las Predicciones de Rul, con sus Valores Reales del Modelo final Random Forest

- MODELOS DE REDES NEURONALES

Para seguir con una progresión lógica de complejidad de los modelos que se van realizando, se llevará a cabo primero el modelo 1D-CNN y posteriormente el modelo LSTM.

Esto se debe a que las redes neuronales convolucionales son más simples de implementar en comparación con las de memoria a largo plazo.

Al entrenar modelos de Machine Learning, especialmente redes neuronales, es común evaluar el rendimiento del modelo con los datos no vistos y con los del entrenamiento. Cuanta más veces se lleve a cabo este tipo de concepto, más posibilidades de evitar el sobreajuste y por lo tanto mejor generalización con nuevos datos. Por ello, se dividirá el conjunto de datos también en parte de entrenamiento y parte de validación haciendo uso de la función "train_test_split" de SkLearn, con un 15% del conjunto de entrenamiento para usarlo de validación, y el 85% para entrenar.

4.4.3 1D - CNN

Inicialmente, se realiza la división con la que los datos de entrenamiento se validará con la función train_test_split, obteniendo las siguientes proporciones que se ven en la Figura 4.24 atendiendo a los porcentajes previamente comentados.

```
datos de entrenamiento procesados:  (10888, 30, 14)
targets de entrenamiento procesados:  (10888,)
datos de validación procesados:  (1922, 30, 14)
targets de validación procesados:  (1922,)
```

Figura 4.24: Forma del Conjunto de datos para la Validación de 1D-CNN tras el uso de la función train_test_split de Sklearn

Para el entrenamiento de este modelo de redes convolucionales, el optimizador Adam ha sido el seleccionado debido a varias ventajas que resultan en un buen entrenamiento y robusto. Adam (Adaptive Moment Estimation) fusiona las ventajas de dos métodos de optimización muy conocidos como son el AdaGrad y RMSProp.

Adam ajusta automáticamente el learning rate para cada parámetro del modelo, lo cual significa que cada parámetro cuenta con su propio learning rate que se va modificando a medida que pasa el entrenamiento, pudiendo mejorar así la convergencia y la estabilidad total del modelo. Por otro lado, Adam es muy eficiente en términos de computación, haciendo que sea atractivo a la hora de usarlo para modelos muy grandes y con grandes volúmenes de datos. Además, requiere poca memoria relativamente, lo que lo hace muy accesible en distintos casos de hardware no muy avanzado, como es el caso actual. Por último, Adam evita fluctuaciones bruscas haciendo uso del promedio de gradientes y el promedio de los cuadrados de los gradientes, ayudando a suavizar finalmente la trayectoria de los parámetros.

```
Epoch 1/20
171/171 - 4s - loss: 3350.3274 - val_loss: 583.4427 - 4s/epoch - 22ms/step
Epoch 2/20
171/171 - 2s - loss: 511.7538 - val_loss: 470.4346 - 2s/epoch - 14ms/step
Epoch 3/20
171/171 - 2s - loss: 459.7480 - val_loss: 449.7238 - 2s/epoch - 14ms/step
Epoch 4/20
171/171 - 2s - loss: 441.5507 - val_loss: 425.3024 - 2s/epoch - 14ms/step
Epoch 5/20
171/171 - 3s - loss: 418.2554 - val_loss: 404.6565 - 3s/epoch - 15ms/step
Epoch 6/20
171/171 - 2s - loss: 391.0753 - val_loss: 364.2635 - 2s/epoch - 14ms/step
Epoch 7/20
171/171 - 3s - loss: 361.2571 - val_loss: 331.2771 - 3s/epoch - 15ms/step
Epoch 8/20
171/171 - 2s - loss: 326.4818 - val_loss: 295.0557 - 2s/epoch - 14ms/step
Epoch 9/20
171/171 - 3s - loss: 302.0748 - val_loss: 274.5176 - 3s/epoch - 15ms/step
Epoch 10/20
171/171 - 3s - loss: 287.3081 - val_loss: 279.2363 - 3s/epoch - 15ms/step
Epoch 11/20
171/171 - 3s - loss: 277.8326 - val_loss: 278.6193 - 3s/epoch - 15ms/step
Epoch 12/20
171/171 - 3s - loss: 268.6728 - val_loss: 248.3424 - 3s/epoch - 15ms/step
Epoch 13/20
...
Epoch 19/20
171/171 - 3s - loss: 245.3909 - val_loss: 236.4340 - 3s/epoch - 16ms/step
Epoch 20/20
171/171 - 3s - loss: 244.7764 - val_loss: 241.4721 - 3s/epoch - 17ms/step
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Figura 4.25: Entrenamiento de la Red Neuronal 1D-CNN

Para tener un buen rendimiento en el conjunto de prueba, es importante detener el entrenamiento del modelo cuando la pérdida de validación sea moderada, en cambio de cuando sea mínima. Esto es una práctica que ayuda a prevenir el sobreajuste y por lo tanto, como se sabe, aumentar la generalización del modelo.

Se elige entrenar el modelo durante 20 épocas. Entrenar el modelo por más épocas reducía aún más la pérdida de validación, pero no mejoraba posteriormente el rendimiento del modelo con los datos de prueba, porque se había metido en el área del sobreajuste, en la que actúa muy bien para los datos vistos pero disminuye la generalización.

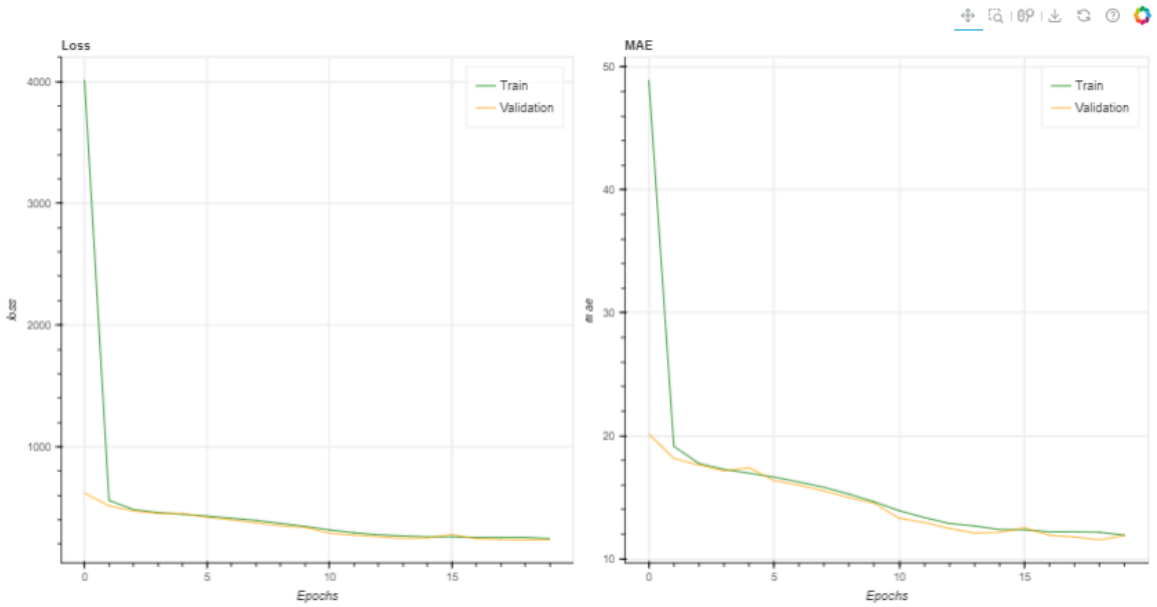


Figura 4.26: gráficas de Loss y MAE del proceso de entrenamiento de 1D-CNN

Las gráficas de Loss y MAE muestran una rápida disminución al principio seguida de una estabilización, indicando que el modelo aprende rápidamente y luego converge hacia un rendimiento óptimo. La proximidad entre las curvas de entrenamiento y validación sugiere una buena generalización sin sobreajuste, demostrando que el modelo es consistente y preciso tanto en los datos de entrenamiento como en los de validación. Este comportamiento es ideal para la tarea de predicción del RUL.

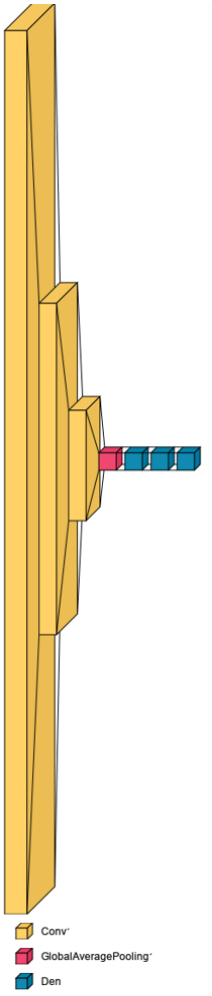


Figura 4.27: Estructura de la Red Neuronal 1D-CNN ploteada con Visualkeras

Haciendo referencia a la Figura 4.26, la red 1D-CNN se compone de varias capas. Primero , cuenta con una capa de convolución 1D con 256 filtros, un tamaño de kernel de 7 y función de activación ReLU, que aplica estos filtros a la entrada. A continuación, le sigue una segunda capa de convolución 1D similar, con mismos números solo que cuenta con 96 filtros. La tercera capa de convolución 1D tiene 32 filtros y los mismos tamaños de kernel y función ReLU. Después de las capas de convolución, se tiene una capa de Global Average Pooling 1D, la cual hace una operación de pooling global promedio en la entrada, disminuyendo las dimensiones y dando una ayuda para evitar el sobreajuste. Por último, una capa densa con 64 unidades y activación ReLU, seguida de otra capa densa con 128

unidades y activación ReLU también, que proporcionan toda la conectividad. La capa de salida cuenta con una sola unidad, para la tarea de regresión.

Una vez comentada la estructura de nuestro modelo neuronal, finalmente se echa un vistazo a los resultados finales de las métricas utilizadas.

```
RMSE final 1D-CNN:  15.676326982492997
MAE 1D-CNN:  12.18980825614929
S-score 1D-CNN:  367.5677426572352
```

Figura 4.28: Resultados Finales del Modelo 1D-CNN

Se observa que el modelo 1D-CNN supera al modelo Random Forest tanto en RMSE como en el MAE dejando entrever una mejor precisión. Además, el S-score del modelo es menos de la mitad que el anterior, representando una notable mejoría y superioridad de este modelo 1D-CNN.

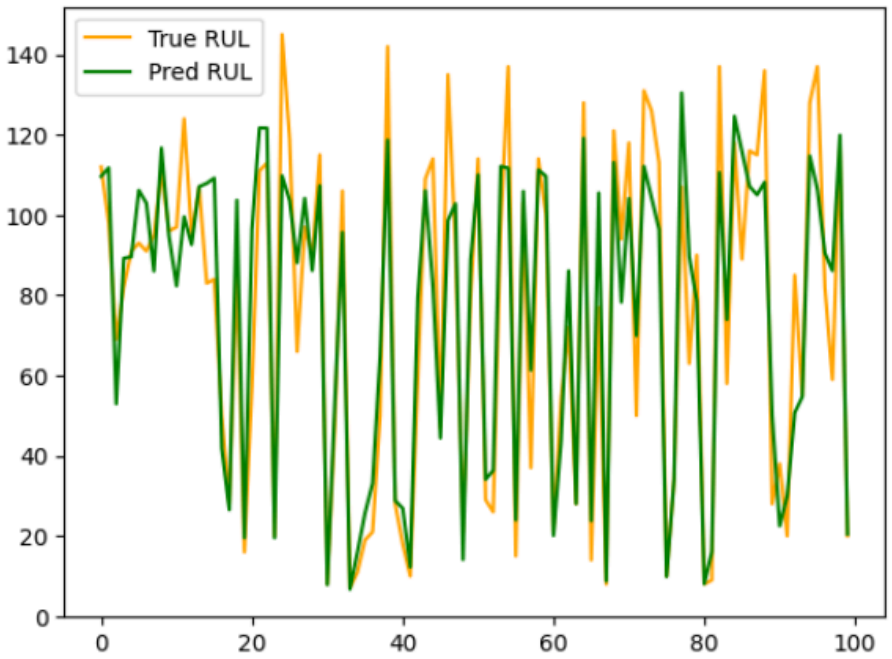


Figura 4.29: Gráfica de las Predicciones de Rul, con sus Valores Reales del Modelo final 1D-CNN

4.4.4 LSTM

En este último modelo, también se llevará a cabo la división en el conjunto de entrenamiento para su validación, con la función train_test_split de SkLearn. Se obtienen las siguientes proporciones.

```
datos de entrenamiento procesados (LSTM):  (15071, 30, 14)
targets de entrenamiento procesados (LSTM):  (15071,)
datos de validación procesados (LSTM):  (2660, 30, 14)
targets de validación procesados (LSTM):  (2660,)
```

Figura 4.30: Forma del Conjunto de datos para la Validación de 1D-CNN tras el uso de la función train_test_split de Sklearn

Para el entrenamiento de este último modelo, se decide hacer como con la primera Red Neuronal 1D-CNN. Se emplea el optimizador Adam, con una tasa de aprendizaje del 0.0001 y un equilibrio de épocas para que sea posible el buen aprendizaje de la Red Neuronal sin llevar a posibles situaciones de sobreajuste.

```
Epoch 1/20
236/236 - 14s - loss: 7090.5386 - mae: 74.0895 - val_loss: 5827.7144 - val_mae: 65.9736 - 14s/epoch - 59ms/step
Epoch 2/20
236/236 - 10s - loss: 5097.2026 - mae: 60.4448 - val_loss: 4297.5161 - val_mae: 54.7164 - 10s/epoch - 42ms/step
Epoch 3/20
236/236 - 11s - loss: 3680.2834 - mae: 49.8701 - val_loss: 3003.4302 - val_mae: 44.6041 - 11s/epoch - 45ms/step
Epoch 4/20
236/236 - 11s - loss: 2509.9175 - mae: 40.1753 - val_loss: 1976.7479 - val_mae: 35.1193 - 11s/epoch - 45ms/step
Epoch 5/20
236/236 - 11s - loss: 1602.0262 - mae: 31.2243 - val_loss: 1227.7334 - val_mae: 27.5269 - 11s/epoch - 46ms/step
Epoch 6/20
236/236 - 11s - loss: 983.6459 - mae: 24.4207 - val_loss: 733.2879 - val_mae: 21.0249 - 11s/epoch - 47ms/step
Epoch 7/20
236/236 - 11s - loss: 597.3524 - mae: 19.2911 - val_loss: 452.8993 - val_mae: 17.1100 - 11s/epoch - 46ms/step
Epoch 8/20
236/236 - 11s - loss: 382.4052 - mae: 15.7651 - val_loss: 306.7156 - val_mae: 14.3301 - 11s/epoch - 46ms/step
Epoch 9/20
236/236 - 11s - loss: 262.8015 - mae: 13.1901 - val_loss: 237.6027 - val_mae: 12.7197 - 11s/epoch - 48ms/step
Epoch 10/20
236/236 - 12s - loss: 202.6266 - mae: 11.5174 - val_loss: 184.6693 - val_mae: 10.9407 - 12s/epoch - 52ms/step
Epoch 11/20
236/236 - 12s - loss: 169.7311 - mae: 10.3421 - val_loss: 189.9268 - val_mae: 10.6259 - 12s/epoch - 50ms/step
Epoch 12/20
236/236 - 12s - loss: 154.0060 - mae: 9.6197 - val_loss: 146.6316 - val_mae: 9.2111 - 12s/epoch - 50ms/step
Epoch 13/20
...
Epoch 19/20
236/236 - 12s - loss: 103.8740 - mae: 7.2551 - val_loss: 103.0896 - val_mae: 7.2019 - 12s/epoch - 50ms/step
Epoch 20/20
236/236 - 11s - loss: 97.0231 - mae: 6.9992 - val_loss: 105.9312 - val_mae: 7.0799 - 11s/epoch - 48ms/step
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Figura 4.31: Entrenamiento de la Red Neuronal LSTM

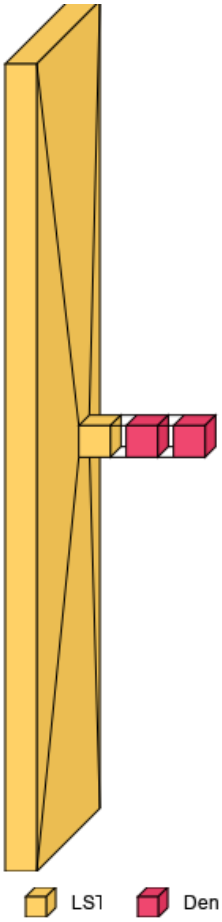


Figura 4.32: Estructura de la Red Neuronal LSTM ploteada con Visualkeras

Haciendo referencia a la Figura 4.32, la red LSTM se compone de varias capas especializadas. Primero, cuenta con una capa LSTM con 256 unidades, que captura las dependencias a largo plazo en los datos de entrada. Luego, por una segunda capa LSTM con 128 unidades, que es más precisa aún más la representación. A continuación, se agrega una capa de Dropout con una tasa del 0.2 para evitar el sobreajuste, la cual decide aleatoriamente apagar el 20% de las neuronas, mejorando así la capacidad de generalización del modelo. Luego, una capa densa con 64 unidades y activación ReLU proporciona una mayor conectividad y aprendizaje no lineal. Por último, una capa de salida con una sola unidad se utiliza para la tarea de regresión, prediciendo el Remaining Useful Life (RUL) del motor.

Una vez comentada la estructura del último modelo neuronal, finalmente se echa un vistazo a los resultados finales de las métricas utilizadas.

```
RMSE final LSTM:  15.1772534397663 23
MAE LSTM:  11.939148672103883
S-score LSTM:  436.438872862131
```

Figura 4.33: Resultados Finales del Modelo LSTM

Se observa que el modelo LSTM supera al modelo 1D-CNN tanto en RMSE como en el MAE, indicando una mejor precisión en las predicciones. Sin embargo, al comparar el S-score, se puede notar que aunque el modelo LSTM tiene un desempeño superior en términos de error cuadrático medio y error absoluto medio, su S-score es ligeramente superior al del modelo 1D-CNN, lo que sugiere que, aunque el LSTM es altamente preciso, el 1D-CNN sigue siendo una opción competitiva y robusta en términos de estabilidad de las predicciones.

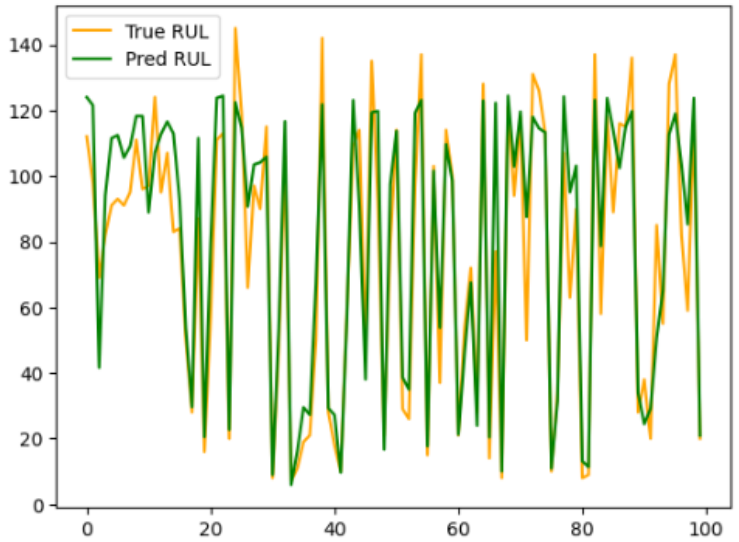


Figura 4.34: Gráfica de las Predicciones de Rul, con sus Valores Reales del Modelo final LSTM

CAPÍTULO 5. ANÁLISIS DE RESULTADOS

A continuación, se detalla el análisis de los resultados:

1. Evaluación de Métricas de Rendimiento:

A lo largo del desarrollo del proyecto, se han utilizado diferentes métricas de rendimiento para poder medir y evaluar la precisión y la eficacia de los diferentes modelos desarrollados. Las métricas han sido 3:

- **RMSE** (Root Mean Squared Error): La cual mide la diferencia promedio entre los valores predichos y los reales. Cuanto más bajo es el valor del RMSE, mayor precisión tiene. Los valores que se han obtenido muestran variaciones importantes.
- **MAE:** (Mean Absolute Error): Mide la magnitud promedio de los errores en un conjunto de predicciones sin tener en cuenta la dirección. También, un MAE más bajo implica mayor precisión. Se eligió para dar un punto de vista más complementario además de la información del RMSE.
- **S-Score:** El S-Score evalúa específicamente la precisión de las predicciones del RUL, llevando a cabo penalizaciones tanto en las predicciones que se pasan de conservadoras como las que son excesivamente optimistas. Ha sido fundamental para tener un mejor punto de vista sobre la eficacia de los modelos, y cabe comentar que se decidió añadir dado su común uso en diferentes papers sobre CMAPSS.

Se han desarrollado varios modelos de Machine Learning en los que se incluyen Regresiones y Redes Neuronales. Dentro de las Regresiones, Support Vector Regression (SVR) y Random Forest. En las Redes Neuronales, se optó por las convolucionales (1D-CNN) y las recurrentes de largo plazo (LSTM). La elección de estos algoritmos fue

con intención de llevar una especie de progresión en un tipo de algoritmos más simples, y a continuación otros más complejos.

A continuación, se lleva a cabo una comparación de los modelos:

- **Support Vector Regression:**

  Este primer modelo, ha mostrado una buena primera toma de contacto. Sin embargo, como era de esperar dada su simpleza y su primer puesto, aunque la precisión es aceptable, es el menos efectivo para predecir la RUL de los motores. Se produjo una búsqueda de los mejores hiperparámetros y estimadores por medio de un GridSearch, que pese a su gran carga computacional, fue útil.

  (**RMSE**: *19,74* ; **MAE**: *14,78* ; **S-SCORE**: *1350,66*)

- **Random Forest:**

  Para este segundo modelo, ya se vieron mejoras. La precisión, y capacidad para capturar patrones de los datos fue mejor pero limitada en comparación con los modelos de aprendizaje profundo. No se llevó a cabo una normalización de los datos ya que como se comentó previamente, las técnicas de aleatoriedad de estos algoritmos no tiene en cuenta el escalamiento. Cabe destacar que debido a problemas por carga computacional, se decidió hacer dos modelos extremistas en cuanto a número de árboles se refiere, y elegir el mejor. Dado a que la mejora del modelo no era proporcional al aumento de carga de cómputo que más del doble de árboles precisaba, se decidió seguir adelante con el primer modelo con 150 árboles en cambio de 350.

  (**RMSE**: *19,23* ; **MAE**: *14,50* ; **S-SCORE**: *1121,10*)

- **1D-CNN:**

  Este primer modelo dentro de las redes neuronales ha mostrado resultados prometedores, con un RMSE y MAE más bajos comparados con los modelos

tradicionales. Además, el S-Score es significativamente mejor, menos de la mitad en comparación con el modelo previo, dejando clara la capacidad de las redes neuronales convolucionales para hacer predicciones más equilibradas y precisas de la RUL. Interesante comentar, que debido al gran poder de aprendizaje de las redes neuronales, durante el entrenamiento también se llevó a cabo un proceso de división de los datos de train, para evitar un posible sobreajuste.

(**RMSE**: *15.68* ; **MAE**: *12.19* ; **S-SCORE**: *367,57*)

- **LSTM:**

Este modelo de red neuronal ha demostrado ser el más preciso de todos los evaluados, consiguiendo los valores más bajos en RMSE y MAE en comparación con los demás modelos. Esto resalta la capacidad de las redes neuronales LSTM para capturar y aprender patrones de largo plazo en datos de series temporales, lo que es muy importante para la predicción de la RUL. Aunque el S-Score es ligeramente superior al del modelo 1D-CNN, sigue siendo muy competitivo, reflejando la robustez y precisión del modelo LSTM. Durante el entrenamiento, se utilizaron técnicas avanzadas de ajuste y regularización para optimizar el rendimiento del modelo y prevenir el sobreajuste, asegurando así su efectividad en datos no vistos.

(**RMSE**: *15,18* ; **MAE**: *11,94* ; **S-SCORE**: *436,44*)

CAPÍTULO 6. CONCLUSIONES

En este capítulo se destacan los resultados más importantes del proyecto y se realiza un análisis crítico de los mismos.. A continuación, se presentan las principales conclusiones del estudio.

Análisis Crítico

Un aspecto importante a considerar en la evaluación de los modelos es su capacidad de generalización y seguridad en las predicciones. En el contexto de mantenimiento predictivo, es crucial no solo minimizar los errores de predicción, sino también entender las implicaciones de estos errores.

Si se traza una línea de pendiente uno, las predicciones pueden estar por encima o por debajo de ésta línea.

Predicciones por Encima de la Línea Real: Estas predicciones son problemáticas porque indican que quedan más ciclos de vida de los que realmente quedan, lo que podría llevar a un fallo catastrófico si se confía en estas predicciones. En otras palabras, pueden llevar a que se posponga el mantenimiento necesario, aumentando el riesgo de accidentes.

Predicciones por Debajo de la Línea Real: Aunque estas predicciones no son ideales y reflejan un error en la estimación del RUL, son más seguras. Decir que quedan menos ciclos de vida de los que realmente quedan ofrece un margen de seguridad, ya que se realizaría el mantenimiento antes de lo necesario, evitando potenciales fallos y accidentes. Esto proporciona mayor seguridad operativa al reducir la probabilidad de un fallo inesperado.

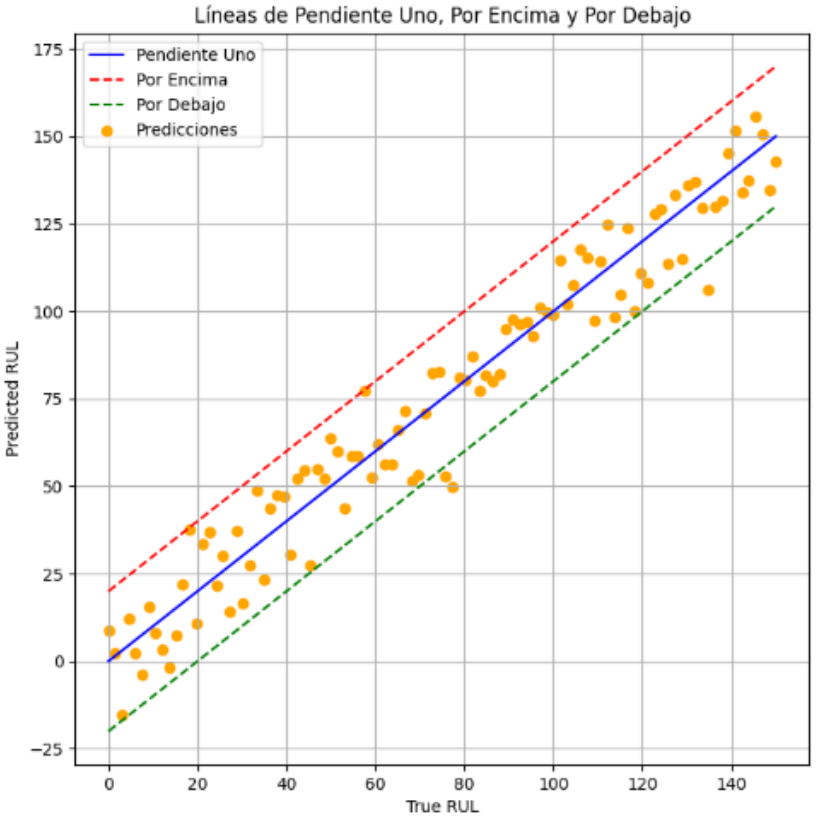


Figura 6.1: Predicciones Seguras e Inseguras

Impacto en la Industria

La implementación de estos modelos de predicción tiene implicaciones significativas para la industria:

Reducción de Costos Operativos y de Mantenimiento: Al predecir con precisión el RUL de los motores, las empresas pueden optimizar sus planes de mantenimiento, reducir tiempos de inactividad no programados y gestionar mejor sus inventarios. Esto conduce a una mayor eficiencia operativa y una reducción de los costos asociados.

Mejora de la Seguridad: Predicciones precisas del RUL permiten identificar problemas potenciales antes de que ocurran, mejorando la seguridad de las operaciones. Esto es especialmente importante en la industria aeroespacial, donde la seguridad es primordial.

Extrapolación a Otras Industrias

La capacidad de generalización de estos modelos no solo se limita a la predicción de la vida útil de motores turbofan. Estos enfoques pueden extrapolarse a una variedad de problemas en diferentes industrias.

CAPÍTULO 7. TRABAJOS FUTUROS

El ámbito del mantenimiento predictivo tiene amplia generalidad, es amplio, y está en constante evolución. En el desarrollo de estos modelos, con las técnicas avanzadas queso actualizan a la orden del día, se presentan muchas oportunidades para mejorar este trabajo:

1. **Nuevos Modelos de Degradación:**

   Como se sabe, en este proyecto se ha usado un modelo de degradación por tramos para los motores. Sin embargo, en entornos reales la degradación de las máquinas puede no llegar a seguir un patrón lineal nunca. Sería beneficioso para el trabajo explorar y desarrollar modelos de degradación más realistas que tengan en cuenta factores como la variabilidad estacional, efectos acumulativos de factores de estrés… En resumen, modelos de degradación no lineales más basados en procesos físicos.

2. **Experimentación con Datasets Más Complejos:**

   El dataset utilizado en este proyecto es uno de los más simples de CMAPSS. Futuras investigaciones podrían centrarse en los otros tres datasets, que cuentan con mayores niveles de complejidad y variaciones significativas en los patrones. Esto robustecería los modelos y ofrecería una mejor evaluación de su aplicabilidad .

3. **Expansión a Otros Dominios Industriales**

   Aunque este proyecto se ha centrado en las turbinas de avión, las metodologías y modelos desarrollados pueden adaptarse muy bien a otras industrias. La fabricación, automoción y la generación de energía cuentan con necesidades parecidas de mantenimiento predictivo.

4. **Despliegue en Sistemas en Tiempo Real**

El desarrollo de sistemas de mantenimiento predictivo en tiempo real es una dirección prometedora. Integrar los modelos predictivos con sistemas de monitoreo en tiempo real y desarrollar plataformas que permitan la toma de decisiones automatizada y proactiva puede revolucionar la gestión del mantenimiento en la industria.

CAPÍTULO 8. REFERENCIAS

[1] Bendezu, K.. "DIAGRAMA UML Y ARQUITECTURA DEL SISTEMA". Sistemas Distribuidos 2013. Febrero, 2013. http://comparape.blogspot.com.es/2013/02/diagrama-uml-y-arquitectura-del-sistema.html.

[2] Herrero Alcántara, T. "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/.

[3] Loeffler, B. "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx

[4] Goodfellow, I., Bengio, Y., & Courville, A. "Deep Learning." MIT Press, 2016. http://www.deeplearningbook.org/

[5] Chen, T., & Guestrin, C. "XGBoost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 785-794, 2016. https://dl.acm.org/doi/10.1145/2939672.2939785

[6] LeCun, Y., Bengio, Y., & Hinton, G. "Deep learning." Nature, 521(7553), 436-444, 2015. https://www.nature.com/articles/nature14539

[7] United Nations. "Transforming our world: the 2030 Agenda for Sustainable Development." Resolution adopted by the General Assembly on 25 September 2015. https://sdgs.un.org/2030agenda

[8] Babu, G. S., Zhao, P., & Li, X. "Deep convolutional neural network based regression approach for estimation of remaining useful life." International conference on database systems for advanced applications. Springer, Cham, 2016. https://link.springer.com/chapter/10.1007/978-3-319-32025-0_29

[9] Zhang, J., Yang, D., & Niu, Q. "A review on remaining useful life estimation methods for lithium-ion batteries." Journal of Power Sources, 196(15), 6007-6014, 2011. https://www.sciencedirect.com/science/article/pii/S0378775311011693

[10] Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., & Siegel, D. "Prognostics and health management design for rotary machinery systems—Reviews, methodology and

applications." Mechanical systems and signal processing, 42(1-2), 314-334, 2014. https://www.sciencedirect.com/science/article/pii/S0888327013004088

[11] Yin, S., & Kaynak, O. "Big data for modern industry: challenges and trends [point of view]." Proceedings of the IEEE, 103(2), 143-146, 2015. https://ieeexplore.ieee.org/document/7001397

[12] Schmidhuber, J. "Deep Learning in Neural Networks: An Overview." Neural Networks, 61, 85-117, 2015. https://www.sciencedirect.com/science/article/pii/S0893608014002135

[13] ISO 31000:2018. "Risk management — Guidelines." International Organization for Standardization, 2018. https://www.iso.org/standard/65694.html

[14] IBM. "What is Machine Learning?" IBM, 2021. https://www.ibm.com/topics/machine-learning

[15] Patro, R. "Cross-Validation: K Fold vs Monte Carlo." Towards Data Science, 2021. https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-24cd3e03fe82

[16] Iberdrola. "Deep Learning: ¿Qué es y para qué sirve?" Iberdrola, 2021. https://www.iberdrola.com/innovacion/deep-learning#:~:text=El%20deep%20learning%2C%20o%20aprendizaje,ha%20llamado%20la%20atenci%C3%B3n%20de

[17] Datademia. "¿Qué es Deep Learning y qué es una red neuronal?" Datademia, 2021. https://datademia.es/blog/que-es-deep-learning-y-que-es-una-red-neuronal#:~:text=Deep%20learning%20o%20aprendizaje%20profundo,predicciones%20con%20una%20gran%20precisi%C3%B3n.

[18] Nour Al-Rahman Al-Serw. "K-Means: The Maths Behind It, How It Works, and an Example." Medium, 2021. https://nouralserw.medium.com/k-means-the-maths-behind-it-how-it-works-and-an-example-67fdcfcb80f0

[19] Soumallya Bishayee. "Everything You Need to Know About Artificial Neural Network." Medium, 2021. https://medium.com/@soumallya160/everything-you-need-to-know-about-artificial-neural-network-8234138de191

[20] Jordi Torres. "Redes Neuronales Recurrentes." Torres.ai, 2021. https://torres.ai/redes-neuronales-recurrentes/

[21] Koen Peters. "Predictive Maintenance of Turbofan Engines." Towards Data Science, 2021. https://towardsdatascience.com/predictive-maintenance-of-turbofan-engines-ec54a083127

[22] "Supervised vs Unsupervised vs Reinforcement." Aitude, 2021. https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/

[23] Alberto Rubiales. "Qué es Underfitting y Overfitting." Medium, 2021. https://medium.com/@rubialesalberto/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9

[24] "SVM Classifier Example in Python." Vitalflux, 2021. https://vitalflux.com/classification-model-svm-classifier-python-example/

[25] "A Complete Introduction to K-Means." Towards Data Science, 2021. https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

[26] "Análisis de Datos: Técnicas y Ejemplos Prácticos." RPubs, 2021. https://rpubs.com/dsfernandez/661689

[27] Chang Woo Hong, Changmin Lee, Kwangsuk Lee, Minseung Ko. "Remaining Useful Life Prognosis for Turbofan Engine Using Explainable Deep Neural Networks with Dimensionality Reduction." Yonsei University, 2020. https://www.researchgate.net/publication/346727375_Remaining_Useful_Life_Prognosis_for_Turbofan_Engine_Using_Explainable_Deep_Neural_Networks_with_Dimensionality_Reduction

ANEXO I: ALINEACIÓN DEL PROYECTO

CON LOS ODS

El impacto de los modelos predictivos de fallos de motores que se han desarrollado en este proyecto va más allá de lo técnico, generando beneficios industriales muy significativos y contribuyendo directamente a los ODS.

- **Reducción de Fabricación:**

  La implementación de estos modelos permite, como se ha estado viendo durante todo el proyecto, anticiparse a fallos en los motores de aviones antes de que ocurran, reduciendo así la necesidad de fabricar piezas de repuesto en exceso. Esto permite a las empresas producir únicamente el inventario necesario según las predicciones del modelo, disminuyendo costos de operación y minimizando el desperdicio de los recursos. Este primer punto se alinea con el **ODS 9** de Industria, Innovación e Infraestructura, optando por una industrialización inclusiva y sostenible.

- **Reducción de Inventario:**

  La gestión del inventario (reducido) del que se acaba de hablar es un punto crítico en la industria. Tener inventario en exceso inmoviliza capital y por supuesto aumenta costos de almacenamiento, mientras que tener un inventario insuficiente puede dar lugar a una producción interrumpida. Los modelos desarrollados en este proyecto optimizan el inventario a tener por una empresa al predecir con precisión

cuándo se necesitarán repuestos. Esto hace que los costos asociados al almacenamiento se reduzcan, y por lo tanto se alinea con el **ODS 12** de Producción y Consumo Responsables.

- **Reducción de Costos y Mejora de la Competitividad:**

  La aplicación de los modelos resulta en una gran reducción de costos de operación. Se reduce la necesidad de fabricar en exceso, y la optimización del inventario liberan capital que puede ser reinvertido en otros aspectos. Esto crea la consecuencia de que las empresas aumenten su competitividad, pudiendo ofrecer productos de mayor calidad a precios más competitivos. Esto se alinea con el **ODS 8** de Trabajo Decente y Crecimiento Económico, apostando por un crecimiento económico sostenible.

- **Impacto en la Sostenibilidad y Medio Ambiente:**

  La mejor gestión del inventario y la optimización de la fabricación tiene un impacto directo en el medio ambiente. Reducir desperdicios de recursos y producciones excesivas hace que la huella de carbono y el consumo de materiales disminuya. Por otro lado, la reducción del inventario implica menos espacio de almacenamiento y por ende menor uso de energía. Esto se alinea con el **ODS 13** de Acción por el Clima y el **ODS 7** de Energía Asequible y No Contaminante.

- Alineación con las Políticas de Sostenibilidad Corporativa:

  Por último, llevar a cabo estos modelos de predicción no solo tiene beneficios operativos y económicos como se ha comentado, sino que también refuerza el compromiso de las empresas con las políticas de sostenibilidad. Al llevar a cabo prácticas más sostenibles, las empresas mejoran su imagen corporativa externa y cumplen con las expectativas de los accionistas y reguladores. Estas últimas mejoras estan alineadas con el **ODS 16** , de Paz, Justicia e Instituciones Sólidas haciendo que las instituciones sean mas transparentes y responsables con sus actos.

A modo de conclusión, este Trabajo Fin de Grado no solo mejora la eficiencia y reduce costos directamente, sino que tiene un impacto muy positivo significativo en la sostenibilidad y competitividad de las empresas. Por lo tanto, al alinearse con muchos Objetivos de Desarrollo Sostenible, este trabajo demuestra que la innovación tecnológica puede estar directamente conectada con el desarrollo sostenible, impulsando prácticas industriales más responsables, beneficiando tanto a las empresas como al medio ambiente y a la sociedad en general.

ANEXO II

Finalmente, se ofrece el código desarrollado:

https://github.com/losadarr/C-digo-Trabajo-Fin-de-Grado-Ingenieria-de-Telecomunicaciones