Universidad Pontificia Comillas, ICAI

DESARROLLO DE UN DASHBOARD VISUAL Y PREDICCIÓN SOBRE DATOS CMAPSS (Commercial Modular Aero-Propulsion System Simulation)

Autor: Losada Rueda, Daniel

Tutora: Coronado Vaca, María

MADRID | Junio, 2024

Agradecimientos

A mis padres, por enseñarme el valor de la constancia, el esfuerzo, y ofrecerme el regalo de estudiar una carrera tan especial.

RESUMEN DEL PROYECTO

En la actualidad, el Machine Learning y la ciencia de datos se han convertido en herramientas esenciales en nuestro día a día, ofreciendo soluciones innovadoras para diversas industrias. Se ha demostrado que las predicciones basadas en datos pueden mejorar significativamente la seguridad operativa y la eficiencia en entornos industriales.

En este Trabajo Fin de Grado, se ha realizado un estudio sobre el mantenimiento predictivo usando datos del dataset *Commercial Modular Aero-Propulsion System Simulation* (CMAPSS) de la Nasa. El objetivo principal ha sido desarrollar un modelo predictivo para anticipar posibles fallas en los motores conociendo su *Remaining Useful Life* (RUL). Además, se ha desarrollado un dashboard en Power BI para facilitar la visualización y análisis de los datos. Los resultados obtenidos han demostrado la efectividad del modelo SVR, así como la utilidad del dashboard para el monitoreo de la salud de los motores, permitiendo una intervención preventiva eficiente.

Palabras clave: Mantenimiento Predictivo, Machine Learning, CMAPSS, Power BI, Modelos Predictivos, Análisis de Datos

ABSTRACT

In today´s world, Machine Learning and data science have become essential tools in our daily lives, offering innovative solutions for various industries. It has been demonstrated that data-driven predictions can significantly enhance operational safety and efficiency in industrial environments. In this bachelor's Thesis, a study on predictive maintenance using data from NASA's *Commercial Modular Aero-Propulsion System Simulation* (CMAPSS) dataset has been conducted. The main objective was to develop a predictive model to anticipate potential engine failures by determining their *Remaining Useful Life* (RUL). Additionally, a dashboard in Power BI has been developed to facilitate data visualization and analysis. The obtained results demonstrated the effectiveness of the SVR model, as well as the utility of the dashboard for the monitoring of engine health, enabling efficient preventive intervention.

Keywords: Predictive Maintenance, Machine Learning, CMAPSS, Power BI, Predictive Models, Data Analysis

ÍNDICE DE LA MEMORIA

CAPÍTULO 1. **INTRODUCCIÓN**

1.1 Objetivos del Proyecto

Los objetivos que se abarcarán en este proyecto son:

1. **Desarrollo de Modelo Predictivo:** Utilizar técnicas de Machine Learning para la creación de un modelo que prediga el RUL (Remaining Useful Life) de los motores turbofán con alta precisión. Esto incluirá el preprocesamiento del dataset elegido, la selección del algoritmo adecuado, el entrenamiento del modelo con datos históricos y la evaluación de su rendimiento.
2. **Creación de un Dashboard en Power BI:** Implementar un cuadro de mandos o dashboard en la herramienta de Microsoft Power BI que permita a un hipotético ingeniero controlador del estado de los motores ver resultados del modelo, métricas y parámetros. Debe ser intuitivo y proporcionar una representación clara y precisa de los datos.

3. **Contextualización y Revisión de Literatura:** Ofrecer un contexto histórico y técnico sobre el mantenimiento predictivo, historia del machine learning, distintos tipos de aprendizaje y las tecnologías relacionadas.

1.2 Metodología

Para el desarrollo de este trabajo, se seguirá una metodología sistemática que incluirá varias etapas cruciales. Inicialmente, se realizará una revisión bibliográfica para comprender las técnicas actuales de Machine Learning aplicadas al mantenimiento predictivo y a la predicción de la vida útil restante (RUL) de motores turbofán. Esta revisión abarcará la evolución del Machine Learning, aplicaciones en mantenimiento predictivo, algoritmos y técnicas de predicción de RUL, y herramientas de visualización de datos. A continuación, se utilizarán datos del dataset *Commercial Modular Aero-Propulsion System Simulation* (CMAPSS) de la NASA para recopilar y preparar las características adicionales. El modelo predictivo se desarrollará con el algoritmo Support

Vector Regression (SVR), entrenando y evaluando el modelo con métricas como RMSE, R^2 y MAE. Además, se creará un dashboard en Power BI para monitorear la salud de los motores, proporcionando visualizaciones del estado actual de los motores, identificación de motores críticos y mediciones de sensores.

1.3 Motivación

El análisis de datos y la visualización de los mismos son componentes críticos en el campo del Business Analytics, ya que facilitan a las empresas la capacidad de interpretar volúmenes grandes de datos y la toma de decisiones informadas. Con el actual auge del Big Data, la necesidad de herramientas y técnicas avanzadas para poder analizar y gestionar los datos se ha vuelto más elevada. Actualmente, las empresas tienen que enfrentarse al desafio de extraer información útil o valor de los datos que se generan y recopilan diariamente. La motivación principal de este proyecto es poder desarrollar un sistema predictivo basándose en herramientas de Machine Learning y la posterior creación de un dashboard que facilite la visualización de distintos resultados e información.

Este Trabajo Fin de Grado se enfoca concretamente en los datos facilitados por la NASA, con el objetivo de predecir el RUL (Remaining Useful Life) de motores turbofán. La predicción para evitar posibles fallos inesperados y para planificar el mantenimiento continuado es crucial, ya que reducirá costos de operación y mejorará la seguridad y eficiencia. EL dashboard en Power BI permitirá a un técnico que lo visualice obtener insights de valor de manera intuitiva y atractiva visualmente.

CAPÍTULO 2. ESTADO DE LA CUESTIÓN

En este proyecto de Fin de Grado, se examinan tecnologías de vanguardia esenciales para el mantenimiento predictivo y el análisis avanzado de datos, enfocándose en algoritmos de aprendizaje automático y técnicas sofisticadas para analizar grandes volúmenes de datos. Estas metodologías permiten la detección temprana de fallos potenciales en sistemas críticos, mejorando la eficiencia, reduciendo costos operativos y aumentando la seguridad operacional. Este estudio proporciona una comprensión profunda del estado actual de estas tecnologías, abordando los fundamentos del aprendizaje automático, su evolución histórica y su aplicación en el contexto del mantenimiento predictivo, optimizando la fiabilidad y disponibilidad de los equipos industriales.

2.1 EVOLUCIÓN HISTÓRICA DEL MACHINE LEARNING

Aunque parece un campo relativamente moderno, el Machine Learning tiene raíces que se remontan a la mitad del siglo XX. Su historia, y la de la Inteligencia Artificial han ido siempre muy a la par, y ambas reflejan varias décadas de investigación en campos como la estadística, probabilidad, y algoritmos.

Todo comenzó con la idea de que las máquinas pudiesen aprender y adaptarse a través de la propia experiencia. Esta idea fue propuesta por primera vez por el archiconocido Alan Turing en 1950, en su artículo "*Computing Machinery and Intelligence*", donde se plantea la pregunta del millón: "¿Pueden pensar las máquinas?" y se introduce el concepto de la prueba de Turing, una herramienta de evaluación de la capacidad de una máquina para exhibir un comportamiento inteligente similar al de un ser humano comparando respuestas de personas sujetas al experimento, con las de ordenadores.

Sin embargo, el término "Machine Learning" fue forjado por Arthur Samuel en 1959. Mientras trabajaba en IBM, Samuel desarrolló uno de los primeros programas con capacidad de aprendizaje. Consistía en un programa de juego de damas que mejoraba con cada partida que jugaba contra humanos. Este proyecto pudo demostrar de manera

práctica que las máquinas gozaban de la capacidad de aprender de los datos y mejorar sus habilidades basándose en experiencias previas de manera autónoma.

Poco después, durante los años 60 y 70, el Machine Learning estuvo dominado por la idea de los sistemas expertos, en los que se intentaba codificar el conocimiento humano en formas tal que los ordenadores pudieran utilizarlo para resolver problemas concretos. Por desgracia, estos sistemas se topaban con la barrera del propio conocimiento que los expertos podían codificar formalmente, limitando asi sus capacidades.

En 1980, gracias al desarrollo de algoritmos como el *Backpropagation* para redes neuronales (algoritmo que ajusta los diferentes pesos de las conexiones entre las neuronas, con el objetivo de reducir el error en la predicción de un modelo), todo comenzó a enfocarse en métodos que permitían a los ordenadores aprender a partir de grandes volúmenes de datos. Este cambio fue de la mano con el aumento en el poder computacional y la disponibilidad por supuesto de más datos, lo que permitía desarrollar patrones más sutiles y complejos.

Con el renacimiento de las redes neuronales en el comienzo de los años 2000, especialmente con el desarrollo del *Deep Learning* (subconjunto del machine learning enfocado a la creación de redes neuronales artificiales, es decir, sistemas que pretenden imitar al cerebro humano adaptándose y aprendiendo a partir grandes cantidades de datos), se logró un punto de inflexión para el Machine Learning. Poderosos investigadores como Geoffrey Hinton, Yann Lecun y Yoshua Bengio lideraron avances en algoritmos que podían entrenar redes profundas, llevando a mejoras significativas en tareas de visión por ordenador, NLP (procesamiento del lenguaje natural) y otros dominios. El éxito de AlexNet (arquitectura de red neuronal profunda que explotaba el poder de las GPU para acelerar el entrenamiento de modelos complejos) en 2012, fue un momento decisivo que demostró el potencial del Deep Learning, ya que se logró una mejora muy significativa en la precisión del reconocimiento de imágenes con respecto a modelos tradicionales, estableciendo un nuevo estándar.

MACHINE LEARNING	**DEEP LEARNING**
Subconjunto de la Inteligencia Artificial	Subconjunto del Machine Learning

Puede entrenarse con conjuntos más pequeños de datos	Requiere gran volumen de datos
Necesita intervención humana para corregir y aprender	Aprende autonómicamente del entorno y errores del pasado
Menor precisión y entrenamiento corto	Mayor precisión y entrenamiento más largo
Correlaciones simples y lineales	Correlaciones complejas y no lineales
Puede entrenarse en CPU	Debe entrenarse en GPU

Figura 2.1: Tabla con las diferencias entre Machine Learning y Deep Learning.

Actualmente, el Machine Learning es un campo de investigación cambiante y de muy rápido crecimiento que ayuda a impulsar incontables avances en casi todos los sectores de la sociedad. Desde aplicaciones en medicina para diagnosticar enfermedades de manera más efectiva hasta el desarrollo de sistemas de recomendación en plataformas de streaming. Es una herramienta actual que se ha convertido en más que esencial en nuestras vidas hoy en día.

A pesar de sus grandes avances, este campo enfrenta continuamente muchos desafíos, como son los sesgos en los datos, cuestiones de privacidad, etc. Pero la comunidad científica está dispuesta y continúa explorando para asegurar que el Machine Learning no solo avance en su capacidad técnica, sino que también en su aplicación justa y ética.

2.2 CATEGORÍAS DE APRENDIZAJE EN MACHINE LEARNING

El campo de Machine learning abarca mucha diversidad, con distintos enfoques para prácticamente todo tipo de problemas de datos. Para tener un entendimiento eficaz, es esencial entender los diversos tipos de aprendizaje que forjan los algoritmos de Machine Learning. Estos métodos se cuelen clasificar según la propia naturaleza de la retroalimentación que se le ofrece al sistema de aprendizaje durante el entrenamiento.

El aprendizaje se puede categorizar generalmente en tres tipos: supervisado, no supervisado y por refuerzo. Cada uno hace uso de distintos enfoques y técnicas para poder extraer patrones útiles. Elegir uno u otro dependerá de la propia naturaleza de los datos a tratar y del problema a resolver.

2.2.1 *APRENDIZAJE SUPERVISADO*

El aprendizaje supervisado es uno de los pilares más fundamentales dentro del Machine Learning, caracterizado por su capacidad de aprendizaje y hacer predicciones a partir de datos etiquetados. El objetivo principal es construir un modelo que sea capaz de generalizar a partir de la información disponible y predecir resultados para datos nuevos que no se vieron durante el entrenamiento.

Cada ejemplo en el conjunto de entrenamiento incluye un dato de entrada (vector de características) y una salida deseada (también llamada etiqueta). El modelo intentará aprender una función que produzca la salida adecuada para una entrada dada. Este proceso conlleva ajustar los distintos parámetros del modelo tal que se minimice la diferencia entre los valores reales del entrenamiento y las predicciones del modelo, proceso comúnmente conocido como ”minimización de la función de pérdida”.

Este aprendizaje es fundamental en varios campos de la ciencia y tecnología, donde se usan modelos predictivos para todo, desde la evaluación del riesgo de crédito hasta la detección temprana de enfermedades hasta, la especialización de este proyecto, el mantenimiento predictivo en diferentes industrias.

Como todo aprendizaje, enfrenta también varios desafíos, especialmente relacionados con los términos *overfitting* y *underfitting* previamente vistos. Además de esto, la calidad y cantidad de los datos etiquetados son fundamentales para la construcción de modelos efectivos, lo que hace que el preprocesamiento y limpieza de datos sean pasos cruciales del proceso de modelado.

Las tareas se pueden dividir en dos categorías:

i. **Clasificación**: Problemas con etiquetas categóricas, cuyo objetivo es asignar una entrada a una de las categorías definidas previamente. Un ejemplo puede ser un diagnóstico médico donde las entradas podrían clasificarse como “enfermo” o “no enfermo”.

ii. **Regresión**: Etiquetas con valor continuo. En este caso, a diferencia de la clasificación, el modelo intentará predecir un valor numérico según las entradas. Un ejemplo sería la predicción del precio de inmuebles teniendo en cuenta ubicación, tamaño, cercanía a servicios de interés, etc.

- **ALGORITMOS COMUNES EN APRENDIZAJE SUPERVISADO:**

1. Regresión Lineal

La regresión lineal busca establecer una relación que sea lineal entre una variable independiente **Y** ý una o más variables independientes **X**. La expresión matemática es $\mathbf{Y = X\beta + \epsilon;}$ Cogeremos cada predicción del modelo y lo compararemos con el valor real. Esta diferencia se llama “residuo”, y la suma de los cuadrados de los residuos será lo que habrá que reducir ajustando los coeficientes $\boldsymbol{\beta}$**.**

2. Regresión Logística

Al contrario que en la regresión lineal, la regresión logística esta ideada para casos donde la variable Y es binaria y categórica. La relación matemática es la siguiente $\boldsymbol{\sigma(z) = 1 + e - z1;}$ Para medir la eficiencia del modelo, atendemos a la "verosimilitud", que mide cuánto de probable es que nuestras predicciones sean correctas dadas las etiquetas reales. El objetivo es ajustar el modelo para que esta verosimilitud sea lo más alta posible, intentando encontrar los mejores parámetros del modelo que hagan que nuestras predicciones sean lo más probables posibles dadas las etiquetas conocidas.

3. Árboles de Decisión

Los árboles de decisión se crean a partir de un proceso de división binaria. Su objetivo es dividir los datos de tal manera que los nodos finales engloben datos lo más homogéneos posibles. Esto se consigue maximizando la pureza de los hipotéticos

nodos finales. El método más común es el de la ganancia de información, que mide la reducción en la entropía antes y después de la división que se estudia hacer. Entropía es una medida de aleatoriedad que se calcula según la distribución de las clases. La división seguirá hasta que se alcance un numero X de muestras por hoja.

La combinación de múltiples arboles de decisión, de tal manera que cada árbol se entrena en un subconjunto aleatorio del conjunto de datos, promediando finalmente y obteniendo las predicciones de todos los árboles se denomina RANDOM FORESTS.

4. Máquinas de Vectores de Soporte (SV)

Las SVM tienen como objetivo correlacionar datos a un espacio de características de alta dimensión tal que los puntos de datos se puedan dividir en categorías, aunque los datos no se puedan separar de forma lineal de otra manera. Es un tipo de algoritmo más complejo, y su proceso de entrenamiento conlleva ajustar el hiperplano para que la distancia entre los puntos de datos más próximos de cada clase sea máxima.

2.2.2 *APRENDIZAJE NO SUPERVISADO*

En el aprendizaje no supervisado los modelos son entrenados utilizando datos sin etiquetar. A diferencia del aprendizaje supervisado, donde cada ejemplo del entrenamiento tiene una etiqueta o respuesta buena, el aprendizaje no supervisado trata con conjuntos de datos que solamente contienen entradas sin etiquetar. El objetivo principal es descubrir patrones, agrupaciones, o posibles estructuras comunes dentro de los datos que puedan ofrecer características suficientemente significativas sin la necesidad de intervenciones externas. En este tipo de aprendizaje, evaluar y validar modelos es mucho más complejo, debido a que la subjetividad adopta un papel importante, ya que los resultados pueden ser muy ambiguos y dependen del conocimiento del analista el problema, dada la ausencia de etiquetas o respuestas correctas.

Una parte muy importante de este aprendizaje son las métricas de similitud. Estas métricas son cruciales para técnicas como por ejemplo reducción de dimensionalidad, o clustering, ya que la formación de grupos o la proyección de datos dependen de cómo se calcula la similitud entre los diferentes puntos de datos. La elección de la métrica optima

dependerá de la propia naturaleza de los datos y el objetivo del problema en cuestión. Algunas de las más utilizadas:

i. **Distancia Euclidiana:** Es la más común y mide la distancia ordinaria entre dos puntos dentro del espacio euclidiano, es decir, la recta más corta entre los puntos. Se utiliza ampliamente en métodos de clustering en los que es necesario calcular la distancia entre los puntos de un clúster, y sus centros.
Si P = (p1,p2,...,pn) y Q = (q1,q2,...,qn) son dos puntos en un espacio euclidiano, la distancia euclidiana entre P y Q:

$$\mathbf{D\ (P,\ Q)} = (q1-p1)^2 + (q2-p2)^2 + ... + (qn-pn)^2$$

ii. **Similitud Coseno:** Esta métrica mide el coseno del ángulo formado entre dos vectores en el espacio n-dimensional. Es muy útil para casos en los que la magnitud de los vectores no es el foco relevante, como por ejemplo procesamiento de texto y sistemas de recomendación, donde la orientación de los vectores es mucho más importante que la magnitud.
Si tenemos dos vectores de características A y B, la similitud coseno será:

$$\textbf{Similitud Coseno (A, B)} = \frac{A \cdot B}{\|A\|\|B\|}$$

- **ALGORITMOS COMUNES EN APRENDIZAJE SUPERVISADO:**

1. Clustering

El clustering es una técnica esencial en el aprendizaje no supervisado cuyo objetivo es dividir el conjunto de datos en diferentes grupos, o clústeres, tal que los puntos de datos dentro de cada grupo sean más similares entre sí que con los de otros grupos. Esta técnica se utiliza mucho para descubrir estructuras ocultas en los datos, segmentar poblaciones en diferentes grupos según características que se compartan, y en aplicaciones como la detección de anomalías. Los algoritmos de clustering son

muy variados, cada uno con sus propios métodos y métricas para definir cuan parecidos son los datos para su agrupación y optimizar la formación de los diferentes grupos o clústeres. Estos algoritmos pueden ser muy útiles en la exploración de datos y en la toma de decisiones basadas en diferentes patrones que se puedan identificar sin tener etiquetas anteriormente.

- **K-means:** Algoritmo que divide el conjunto de datos en K clústeres asignando cada punto al centroide más cercano de todos los posibles, basándose en la distancia euclidiana. Es muy útil en conjuntos de datos grandes, pero hay que especificar el número de clústeres con anterioridad, y luego ajustarlo según los intereses.

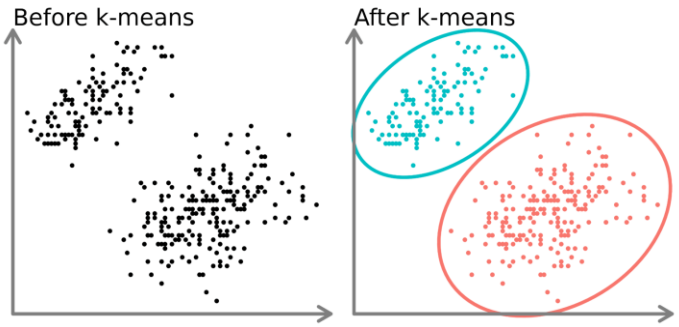


Figura 2.2: Visualización del clustering antes y después de aplicar el algoritmo k-means.

- **Clustering Jerárquico:** Método que crea una jerarquía de clústeres. Puede ser de manera aglomerativa, en la que se combinan clústeres pequeños en clústeres más grandes o divisiva, dividiendo un clúster grande en clústeres más pequeños, y es útil para entender las relaciones entre los datos.

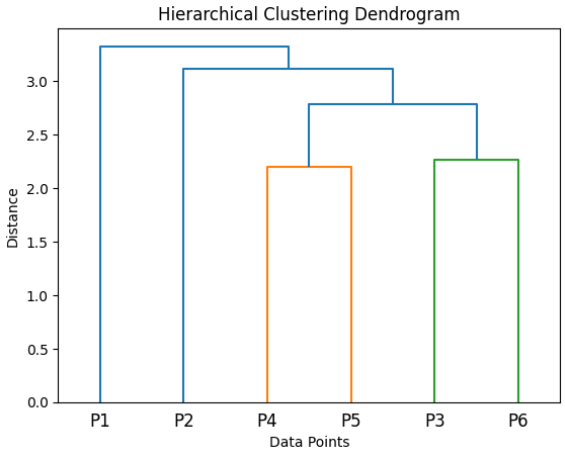


Figura 2.3: Diagrama de agrupamiento jerárquico

2. <u>Reducción de Dimensionalidad (PCA):</u>

La reducción de dimensionalidad es una de las técnicas más esenciales dentro del aprendizaje no supervisado, que consiste en simplificar la mayor cantidad posible de datos sin llegar a perder información crítica. Esta técnica se centra en "la maldición de la dimensionalidad", problema que puede hacer que los modelos de aprendizaje no supervisado sean mucho menos eficaces y más difíciles de interpretar. La técnica más común se denomina PCA (Principal Component Analysis), que transforma datos de altas dimensiones en formatos más compactos de baja dimensión mucho más manejables, manteniendo eso sí la esencia de los datos originales sin llegar a perder mucha información. Mejora la eficiencia dentro de los algoritmos de este aprendizaje y hace más sencilla la visualización y comprensión de relaciones algo más complejas entre los datos. Campos en los que su uso es muy común son el procesamiento de datos y el análisis de texto, generalmente donde los datos contengan un gran número de variables.

2.2.3 *ALGORITMOS AVANZADOS Y DEEP LEARNING*

Este tercer tipo de aprendizaje, el Deep Learning, es una rama avanzada del aprendizaje automático o no supervisado que ha ido ganando prominencia en los últimos años. Esto se debe a su amplia capacidad para procesar grandes volúmenes de datos y aprender patrones subyacentes complejos a un nivel sin precedentes. Hace uso de redes neuronales con muchas capas (de ahí viene su término "profundo") para la posterior construcción de modelos complejos que puedan recoger y representar patrones sofisticados en el conjunto de datos.

El Deep Learning ha sido crucial en el avance en áreas como el reconocimiento de imágenes y voz. También ha sido aplicado en sistemas autónomos (como vehículos), medicina (diagnósticos entrenados automáticos) y sobre todo en el procesamiento del lenguaje natural, transformando servicios que han llegado para quedarse y utilizar diariamente.

Como se ha comentado anteriormente, utiliza **redes neuronales artificiales**. Estas redes son conjuntos de algoritmos diseñados para reconocer patrones y están compuestas por

nodos o “neuronas” que se conectan entre sí en distintas capas. Cada neurona de una capa recibe entradas de múltiples neuronas de la capa anterior, hace los cálculos pertinentes y pasa su salida a las neuronas de la capa siguiente, y así sucesivamente. Lo realmente especial en este aprendizaje son sus **capas ocultas**, que hacen transformaciones progresivas de los datos de entrada concluyendo y extrayendo características de alto nivel a partir de características más sencillas en cada paso.

Cada neurona tiene sus **funciones de activación**, que definen la salida de esa neurona en concreto dadas unas entradas. Existen distintas funciones en función de las características particulares que afectan a la convergencia a la hora de aprender no-linealidades, como son la Sigmoide, Tanh o ReLu. Durante el entrenamiento, el objetivo es minimizar una **función de pérdida**, la cual mide la diferencia entre la salida predicha y los valores reales. Algunos algoritmos de optimización y actualización de pesos de la red comunes son el descenso del gradiente estocástico (SGD) y Adam.

- **ALGORITMOS COMUNES EN DEEP LEARNING:**

1. Redes Neuronales Convolucionales (CNNs)

Este tipo de redes son especialmente poderosas para procesamientos de video, análisis de imágenes médicas, y reconocimiento visual en general. Hacen uso de la operación matemática convolución, que les permite capturar de manera eficaz las distintas relaciones espaciales y temporales en el conjunto de datos de la imagen.

2. Long Short-Term Memory Networks (LSTMs)

Están basadas en las Redes Neuronales Recurrentes (RNNs), que se especializan en el trabajo con secuencias de datos o series temporales con la característica de su memoria de corto plazo que les permite recordar entradas anteriores en una secuencia. LSTMs fueron diseñadas para aprender dependencias a largo plazo en los datos de una secuencia. Útiles en tareas que requieren esta memorización de información por periodos prolongados, como por ejemplo generación de texto y reconocimientos de voz.

3. Redes Generativas Antagónicas (GANs)

Las GANs tienen un enfoque relativamente nuevo y con mucha proyección dentro del aprendizaje no supervisado. Su funcionamiento consta de dos redes neuronales cuyo objetivo es competir entre sí. Una intentará crear datos que parecerán reales, y la otra intentará distinguir entre estos y los verdaderamente reales. Su uso ha avanzado generación de imágenes realistas, síntesis de voz, mejora de fotos, etc. Aplicaciones muy comunes e innovadoras hoy en día.

Concluyendo esta sección de los distritos tipos de aprendizaje y sus algoritmos más comunes, en este proyecto nos enfocaremos principalmente haciendo uso de algoritmos avanzados de estas categorías para poder desarrollar modelos de predicción que sean capaces de realizar predicciones precisas y útiles en el ámbito del mantenimiento predictivo. (**Random Forests, Regresión Lineal, CNNs, LSTM**)

CAPÍTULO 3. DESCRIPCIÓN DE TECNOLOGÍAS Y HERRAMIENTAS DISPONIBLES

El avance de la tecnología ha resultado en un amplio abanico de herramientas especializadas y diseñadas para conseguir la optimización de distintos puntos dentro del análisis de datos y el desarrollo de software. Desde entornos de desarrollo integrado (IDE) muy potentes, hasta plataformas de gestión y organización de código, estas facilitan las soluciones dándoles robustez y escalabilidad.

En este capítulo se explorarán las diferentes tecnologías y herramientas actuales que existen para tratar el problema del proyecto y las empleadas a lo largo del desarrollo. La buena elección de estas herramientas es muy importante para poder acercarse a los objetivos planteados anteriormente, ya que nos darán acceso a no solo la implementación de modelos de Machine Learning, sino también a la organización eficiente y clara de los datos y sus visualizaciones. Se detallarán las funcionalidades, ventajas y aplicaciones específicas de cada una de las herramientas para dar una visión completa y general de cómo se podrían usar para alcanzar los objetivos.

3.1 HERRAMIENTAS PARA LA VISUALIZACIÓN

La visualización es primordial en el análisis de datos, ya que permite la interpretación de enormes volúmenes de datos intuitivamente. Herramientas como Power BI, Tableau, además de las librerías de visualización en Python han transformado la presentación de datos.

3.1.1 *POWER BI*

Figura 3.1: Logo de Power BI

Power BI es una herramienta de visualización de datos desarrollada por Microsoft y utilizada ampliamente por empresas de todos los tamaños para la creación de informes y dashboard interactivos.

- Conectividad con variadas fuentes de datos: Puede conectarse a diversas fuentes como bases de datos SQL, Excel y servicios Web, integrando datos de diferentes orígenes para su análisis.
- Interactividad y personalización: Tiene una amplia variedad de visualizaciones interactivas personalizables para lograr necesidades concretas. Se pueden crear gráficos, tablas, mapas y otros elementos visuales para hacer la exploración de los datos más dinámica.
- Actualización en tiempo real: Cuenta con la posibilidad de actualizar automáticamente los dashboards y gráficos mientras los datos van cambiando, garantizando que las decisiones se tomarán en base a la información más reciente.
- Integración con otros servicios de Microsoft: Se integra con productos de Microsoft como Azure, Office 365 y Sharepoint, dando lugar a un ecosistema cohesivo para la gestión y análisis de los datos.

Será la herramienta elegida para la realización de este proyecto

3.1.2 *TABLEAU*

Figura 3.2: Logo de Tableau

Tableau es otra potente herramienta usada para visualizar datos conocida por su capacidad de manejar grandes conjuntos de datos. Se usa en diversas industrias por su versatilidad.

- Comunidad y soporte: Tiene una gran comunidad de usuarios y un amplio abanico de recursos educativos y soporte, lo cual facilita el aprendizaje continuo.
- Visualizaciones avanzadas: Ofrece una variada colección de tipos de gráficos, incluyendo series temporales, mapas geográficos, diagramas de dispersión, etc.
- Calculaciones complejas: Consta de un potente lenguaje de cálculo propio llamado Tableau Calculations, que facilita la creación de campos calculados, parámetros y expresiones LOD (Level Of Detail).
- Rendimiento: Conocido por su capacidad de manejo con volúmenes de datos grandes. Usa un motor de datos en memoria que se llama Hyper, con el objetivo de optimizar el rendimiento de los análisis.

3.1.3 *MATPLOTLIB (LIBRERÍA DE PYTHON)*

Figura 3.3: Logo de Matplotlib

Librería de visualización de datos en Python que ofrece flexibilidad para crear gráficos estáticos, animados e interactivos. Muy famosa en la comunidad de Python debido a su capacidad de personalización y amplia funcionalidad.

- Personalización detallada: Permite el control total sobre las características de las visualizaciones, desde colores hasta disposición y estilo.
- Documentación y comunidad: Tiene una extensa documentación y una comunidad que ayuda con tutoriales y ejemplos.
- Gráficos interactivos: Aunque inicialmente fue pensada y diseñada para gráficos estáticos, permite crear gráficos interactivos usando bibliotecas adicionales como mpld3 o Plotly.

3.1.4 *PLOTLY (LIBRERÍA DE PYTHON)*

Figura 3.4: Logo de Plotly

Plotly es una librería de gráficos interactivos en Python que permite crear visualizaciones interactivas y personalizables. A diferencia de Matplotlib, la cual se centra en gráficos estáticos, ofrece herramientas para crear gráficos que se pueden manipular directamente, ya sea hacer zoom, seleccionar datos en tiempo real, añadir anotaciones, etc.

- Facilidad de integración: Se puede integrar con otras librerías como Pandas, NumPy y SciPy, creando así un flujo de trabajo y visualización de manera sencilla.
- Soporte para gráficos 3D: Soporta la creación de distintos gráficos en 3D además de los 2D estándar, útiles para conjuntos de datos multidimensionales.
- Compatibilidad con Dash: Cuenta con la integración de Dash, un marco que permite construir aplicaciones web interactivas para la visualización de conjuntos de datos, haciendo más facil la creación de dashboards.

3.1.5 *SEABORN (LIBRERÍA DE PYTHON)*

Figura 3.5: Logo de Seaborn

Seaborn es una librería de visualización de datos que se basa en Matplotlib, y proporciona una interfaz de alto nivel para crear gráficos estadísticos fáciles de interpretar. Es muy común en el análisis exploratorio de datos ya que es muy simple y fácilmente sus visualizaciones tienen mucha calidad.

- Integración con Pandas: Se integra con Pandas permitiendo la creación de gráficos a partir de DataFrames y Series directamente de Pandas.

- Análisis Estadístico: Trae funcionas que facilitan las visualizaciones de relaciones estadísticas entre variables, como gráficos de regresión y distribución conjunta.
- Facilidad de uso: Sintaxis más simple y concisa que la de Matplotlib, siendo más sencilla de usar para los usuarios.

3.2 HERRAMIENTAS PARA EL DESARROLLO Y ANÁLISIS DE CÓDIGO

3.2.1 *PYTHON*

Figura 3.6: Logo de Python

Python es un lenguaje de programación de alto nivel, el cual ha ganado mucha popularidad los últimos años en los campos de la ciencia de datos y el aprendizaje automático. Esto es debido a su sintaxis clara, facilidad de uso, amplia comunidad que trabaja de forma activa y una enorme colección de bibliotecas especializadas en la ciencia de datos.

- Interoperabilidad: Tiene facil integración con otros lenguajes y tecnologías, como por ejemplo C, C++ y Java, lo que lo convierte en un lenguaje muy versátil y útil en varios entornos de desarrollo.
- Biblioteca Estándar: Cuenta con módulos y paquetes muy extensos para tareas muy variadas. Abarcan desde la manipulación de archivos hasta la compleja computación científica.
- Comunidad activa: De las características más importantes. Tiene una comunidad global activa que contribuye con librerías, documentación, herramientas, etc., facilitando resolver los problemas.

- Integración con bibliotecas esenciales para Ciencia de Datos y Machine Learning: Pandas, NumPy, Matplotlib, Seaborn, Plotly, Scikit-Learn, TenorFlow, Keras… La combinación de estas bibliotecas ofrecen un desarrollo y entrenamiento de modelos de machine learning y Deep learning, además de análisis y visualización de datos poderoso.

3.2.2 *DAX (DATA ANALYSIS EXPRESSIONS)*

DAX es un lenguaje de fórmulas y consultas usando Power BI para la realización de cálculos, creación de medidas y columnas calculadas. Se asemeja a las fórmulas de Excel, pero es mucho más poderoso y está optimizado para el análisis de datos.

3.2.3 *POWER QUERY (LENGUAJE M)*

Figura 3.7: Logo de Power Query

Power Query es una herramienta de Power BI la cual fue diseñada para la extracción, transformación y carga de datos, proceso comúnmente conocido como ETL. Usa lenguaje de fórmulas que se llama M (Power Query Formula Language), que es flexible y potente para la manipulación de datos.

- Extracción y transformación de Datos: Permite conectar, limpiar y transformar datos de muchas fuentes, pudiendo eliminar columnas, cambiar tipos de datos, filtrar filas y combinar tablas.
- Lenguaje M: M es un lenguaje usado para definir todas las operaciones de ETL. Su conocimiento permite usar transformaciones más avanzadas y personalizadas, aunque muchas otras se pueden hacer mediante la interfaz gráfica.

- Automatización de Procesos: Con Power Query se pueden hacer la actualización de datos y transformaciones aplicadas de manera automatizada, asegurándose así de que los dashboards siempre estén actualizados.

3.2.4 *VISUAL STUDIO CODE*

Figura 3.8: Logo de Visual Studio Code

Visual Studio Code (VS Code) se trata de un entorno de desarrollo integrado (IDE) que fue desarrollado por Microsoft. Debido a su flexibilidad, ligereza y personalización se ha ganado un sitio entre los IDEs más utilizados por los desarrolladores de todo el mundo. Incluye extensiones para muchos lenguajes de programación: Python, TypeScript, Java, JavaScript, C++, C# y varios más. Tiene un diseño modular que permite el uso de extensiones, pudiendo añadir solamente las características más necesitadas por el usuario.

- Terminal Integrado: Cuenta con una terminal integrada que permite ejecutar los comandos de Shell de manera directa desde el editor, haciendo más facil la gestión del entorno sin tener que usar muchas aplicaciones.
- Git y Control de Versiones: Tiene una integración de manera nativa con Git, permitiendo a los desarrollado.
- Extensiones y Marketplace: Existe un Marketplace donde se pueden descargar extensiones como formateadores de código, herramientas varias de integración, depuradores y más, que facilitan el código.
- Interfaz de Usuario: Tiene una interfaz de usuario bastante limpia, cómoda y personalizable que incluye soporte para temas personalizados, paneles, terminales, etc.

En esta sección, se han explorado una amplia gama de herramientas que se consideran esenciales para el desarrollo y análisis de proyectos como este, de ciencias de datos y

Machine Learning. Desde lenguajes de programación, poderosas bibliotecas, hasta entornos de desarrollo integrados. Su combinación desempeña un papel crucial en la simplificación de tantos procesos complejos

CAPÍTULO 4. **TRABAJO**

4.1 DESCRIPCIÓN DEL CONJUNTO DE DATOS

En este proyecto, se hará uso del dataset CMAPSS (Commercial Modular Aero-Propulsion System Simulation), de la NASA. Este conjunto de datos es muy utilizado en la investigación y el desarrollo de algoritmos de mantenimiento predictivo por su riqueza y complejidad. El dataset CMAPSS cuenta con información detallada sobre el funcionamiento de motores turbofán, recopilada de simulaciones que representan condiciones operativas realistas y muy variadas.

- **Estructura del Dataset**

El dataset CMAPSS está organizado en varios archivos, cada uno correspondiente a diferentes configuraciones y modos de falla de los motores. Estos archivos están etiquetados como FD001, FD002, FD003 y FD004. Cada archivo presenta características únicas en términos de condiciones operativas y modos de falla. Por ejemplo:

- **FD001:** Tiene datos de una sola condición operativa (nivel del mar) y un modo de falla (degradación del HPC).
- **FD002**: Incluye múltiples condiciones operativas y un modo de falla.
- **FD003**: Parecido a FD001 pero con dos modos de falla.
- **FD004**: Combina múltiples condiciones operativas con dos modos de falla.

Para este proyecto, se ha decidido trabajar con el primer dataset, FD001. La elección de FD001 se debe a su simplicidad relativa, lo que deja centrarse en el desarrollo y evaluación del modelo sin añadir complejidad de múltiples modos de falla y más condiciones operativas. Por otro lado, FD001 ofrece un muy buen punto de partida para entender los fundamentos del mantenimiento predictivo y sentar las bases para futuras investigaciones con conjuntos más complejos.

Cada dataset (FD001, FD002, FD003, FD004) tiene tres archivos principales:

- **train_FD00X:** Con los datos de entrenamiento, los cuales incluyen datos de los motores desde el inicio hasta el fallo.
- **test_FD00X:** Con los datos de prueba, incluyendo datos de motores hasta cierto tiempo antes de la falla.
- **RUL_FD00X:** Con los valores reales de la RUL de los motores en el conjunto de prueba, los cuales se usarán para evaluar la precisión del modelo.

Las principales características del conjunto de datos:

Número de Unidad: Identifica cada motor de forma única. Cada motor tiene muchos ciclos operativos registrados.

Ciclos: Indica la edad del motor en términos de operatividad, siendo un ciclo un periodo de operación del motor. Es un contador que aumenta con cada ciclo operativo del motor.

Ajustes Operativos: Tres configuraciones que representan diferentes condiciones operativas bajo las cuales funciona el motor. Influyen en el rendimiento y la salud del motor.

Mediciones de Sensores: 21 sensores que monitorean varias características del motor, ofreciendo una visión detallada de su estado en cada ciclo operativo.

unit_number	time_in_cycles	operational_setting_1	operational_setting_2	operational_setting_3	sensor_measurement_1
1	1	-0.0007	-0.0004	100.0	518.67
1	2	0.0019	-0.0003	100.0	518.67
1	3	-0.0043	0.0003	100.0	518.67
1	4	0.0007	0.0000	100.0	518.67
1	5	-0.0019	-0.0002	100.0	518.67
1	6	-0.0043	-0.0001	100.0	518.67
1	7	0.0010	0.0001	100.0	518.67
1	8	-0.0034	0.0003	100.0	518.67
1	9	0.0008	0.0001	100.0	518.67
1	10	-0.0033	0.0001	100.0	518.67

Figura 4.1: Primeras 10 filas del dataset FD001

Se puede observar la estructura de los datos mostrando las primeras 10 filas del dataset FD001. La tabla enseña las características clave del conjunto de datos que se han comentado anteriormente. Es importante comentar que la tabla muestra solo una parte de las características del dataset, ya que en realidad existen 20 mediciones de sensores adicionales.

- **Importancia del Dataset CMAPSS**

El dataset CMAPSS es fundamental para el desarrollo de técnicas de mantenimiento predictivo debido a las siguientes razones:

Variedad de Condiciones Operativas: Incluye datos de motores operando bajo diversas condiciones, lo que permite desarrollar modelos robustos capaces de manejar diferentes escenarios.

Detallado y Completo: Proporciona un seguimiento exhaustivo de los motores, capturando una amplia gama de mediciones de sensores que reflejan el estado del motor en cada ciclo.

Realismo de Simulaciones: Los datos simulados reflejan condiciones operativas realistas, proporcionando una base sólida para el desarrollo de modelos aplicables en situaciones del mundo real.

4.2 PREPARACIÓN DEL CONJUNTO DE DATOS Y ANÁLISIS EXPLORATORIO

Esta sección se detallan los pasos iniciales para poder convertir el conjunto de datos en un formato más susceptible para el análisis y modelado. Desde la carga y exploración inicial de los datos, hasta la división en subconjuntos de Train y Test. Además, se entrará más en detalle realizando un análisis exploratorio para comprender mejor las características del dataset, identificar posibles relaciones y detectar valores atípicos o anomalías que puedan afectar más adelante el desarrollo del modelo predictivo.

1. Carga de Librerías

Para empezar con la preparación de datos, es necesario importar las librerías esenciales que permitan manipular, analizar y visualizar el conjunto de datos. Pandas para la manipulación, Numpy para los cálculos y Seaborn para visualización de los datos.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```
✓ 8.8s

Figura 4.2: Import de Librerías necesarias

2. Carga del Conjunto de Datos y Exploración Inicial

El conjunto CMAPSS se ofrece en formato CSV, el cual se carga en un DataFrame de pandas para facilitar su análisis. Como se ha expuesto anteriormente, se trabajará con el dataset FD001. Para entender la estructura, es importante una exploración inicial para ver tipo de datos y verificar valores que puedan faltar.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   unit_number           20631 non-null  int64
 1   time_in_cycles        20631 non-null  int64
 2   operational_setting_1 20631 non-null  float64
 3   operational_setting_2 20631 non-null  float64
 4   operational_setting_3 20631 non-null  float64
 5   sensor_measurement_1  20631 non-null  float64
```

Figura 4.3: Import de Librerías necesarias

	count	mean	std	min	25%	50%	75%	max
unit_number	20631.0	51.506568	2.922763e+01	1.0000	26.0000	52.0000	77.0000	100.0000
time_in_cycles	20631.0	108.807862	6.888099e+01	1.0000	52.0000	104.0000	156.0000	362.0000
operational_setting_1	20631.0	-0.000009	2.187313e-03	-0.0087	-0.0015	0.0000	0.0015	0.0087
operational_setting_2	20631.0	0.000002	2.930621e-04	-0.0006	-0.0002	0.0000	0.0003	0.0006
operational_setting_3	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
sensor_measurement_1	20631.0	518.670000	0.000000e+00	518.6700	518.6700	518.6700	518.6700	518.6700
sensor_measurement_2	20631.0	642.680934	5.000533e-01	641.2100	642.3250	642.6400	643.0000	644.5300
sensor_measurement_3	20631.0	1590.523119	6.131150e+00	1571.0400	1586.2600	1590.1000	1594.3800	1616.9100
sensor_measurement_4	20631.0	1408.933782	9.000605e+00	1382.2500	1402.3600	1408.0400	1414.5550	1441.4900
sensor_measurement_5	20631.0	14.620000	1.776400e-15	14.6200	14.6200	14.6200	14.6200	14.6200
sensor_measurement_6	20631.0	21.609803	1.388985e-03	21.6000	21.6100	21.6100	21.6100	21.6100
sensor_measurement_7	20631.0	553.367711	8.850923e-01	549.8500	552.8100	553.4400	554.0100	556.0600
sensor_measurement_8	20631.0	2388.096652	7.098548e-02	2387.9000	2388.0500	2388.0900	2388.1400	2388.5600
sensor_measurement_9	20631.0	9065.242941	2.208288e+01	9021.7300	9053.1000	9060.6600	9069.4200	9244.5900
sensor_measurement_10	20631.0	1.300000	0.000000e+00	1.3000	1.3000	1.3000	1.3000	1.3000
sensor_measurement_11	20631.0	47.541168	2.670874e-01	46.8500	47.3500	47.5100	47.7000	48.5300
sensor_measurement_12	20631.0	521.413470	7.375534e-01	518.6900	520.9600	521.4800	521.9500	523.3800
sensor_measurement_13	20631.0	2388.096152	7.191892e-02	2387.8800	2388.0400	2388.0900	2388.1400	2388.5600
sensor_measurement_14	20631.0	8143.752722	1.907618e+01	8099.9400	8133.2450	8140.5400	8148.3100	8293.7200
sensor_measurement_15	20631.0	8.442146	3.750504e-02	8.3249	8.4149	8.4389	8.4656	8.5848
sensor_measurement_16	20631.0	0.030000	1.387812e-17	0.0300	0.0300	0.0300	0.0300	0.0300
sensor_measurement_17	20631.0	393.210654	1.548763e+00	388.0000	392.0000	393.0000	394.0000	400.0000
sensor_measurement_18	20631.0	2388.000000	0.000000e+00	2388.0000	2388.0000	2388.0000	2388.0000	2388.0000
sensor_measurement_19	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
sensor_measurement_20	20631.0	38.816271	1.807464e-01	38.1400	38.7000	38.8300	38.9500	39.4300
sensor_measurement_21	20631.0	23.289705	1.082509e-01	22.8942	23.2218	23.2979	23.3668	23.6184

Figura 4.4: Estadísticas Descriptivas de las Características del DataFrame

Observando estas imágenes se pueden ver las estadísticas descriptivas básicas de cada columna, revelando la distribución y variabilidad de los datos. La Unidad y los Ciclos serán datos de tipo **int64**, mientras que el resto de los ajustes operacionales y medidas de sensores **float64**. Por otro lado, no existen valores nulos.

Con respecto a las estadísticas, el conjunto de datos tiene unidades de motor del 1 al 100, cada una con un numero diferente de ciclos máximos. Calculando el número máximo de ciclos que cada motor alcanzó previo al fallo, en la Figura 4.5, se observa un promedio de fallos entre 199 y 206 ciclos, con desviación 46 y un rango de [128, 362].

	time_in_cycles
count	100.000000
mean	206.310000
std	46.342749
min	128.000000
25%	177.000000
50%	199.000000
75%	229.250000
max	362.000000

Figura 4.5: Estadísticas Descriptivas de los Ciclos máximos de cada Motor

Por último, los sensores que más fluctúan son el 14 y el 9, mientras que los sensores 19,18,20,1 no lo hacen, debido a la versión del conjunto de datos con la que se está trabajando.

3. Normalización y División de Train y Test

Escalar las características a un rango que sea uniforme es crucial en la fase de preparación de datos, ya que la estabilidad de los algoritmos de Machine Learning se ve muy mejorada. También se dividen los datos en Train y Test para posteriormente evaluar el rendimiento con datos no conocidos durante el entrenamiento.

```
# Normalización de los datos
scaler = MinMaxScaler()
df_normalized_train = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
df_normalized_test = pd.DataFrame(scaler.fit_transform(test_df), columns=test_df.columns)
```

Figura 4.6: Normalización de Datos usando Librería Sklearn

4. Visualización de Distribuciones de Datos

Para realizar de forma correcta la visualización de los datos respectivos a los sensores, es muy importante la Vida Útil Remanente (RUL). RUL ilustra el comportamiento de los motores según van acercándose a su falla final. Es importante porque es interesante usar la RUL como el eje X de las visualizaciones, viendo, así como se desarrollan algunos sensores mientras el fallo está próximo.

Una manera de calcular la RUL es razonar con el Ciclo máximo (el valor máximo de la variable Ciclo por cada motor), restándole el actual. De esta manera, se conocerán los

ciclos restantes para que el motor correspondiente deje de funcionar en un momento concreto.

Antes de la visualización, la RUL es calculada y añadida al DataFrame.

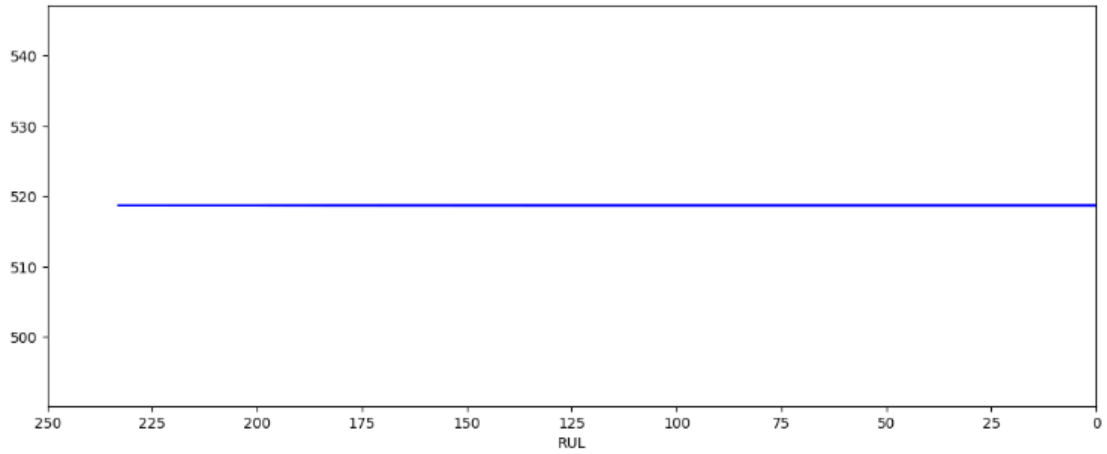


Figura 4.7: Comportamiento Constante común en Sensores 1,5,6,10,16,18 y 19

Tras la visualización del cambio en los sensores a medida que la RUL va disminuyendo linealmente acercándose al valor máximo de Ciclo por motor, se observa que en los sensores 1, 5, 6, 10, 16, 18 y19 las mediciones se mantienen constantes sin ningún tipo de cambio apreciable. Estos sensores serán descartados para seguir con el desarrollo del modelo, ya que aportan información irrelevante acerca del desgaste o deterioro de los motores.

5. Correlaciones

La matriz de correlación es una herramienta muy común a la hora de realizar el análisis, ya que nos permite identificar qué variables están altamente correlacionadas entre sí, dejando entrever una posible útil reducción de la dimensionalidad del conjunto de datos mejorando la eficiencia del modelo sin perder información relevante.

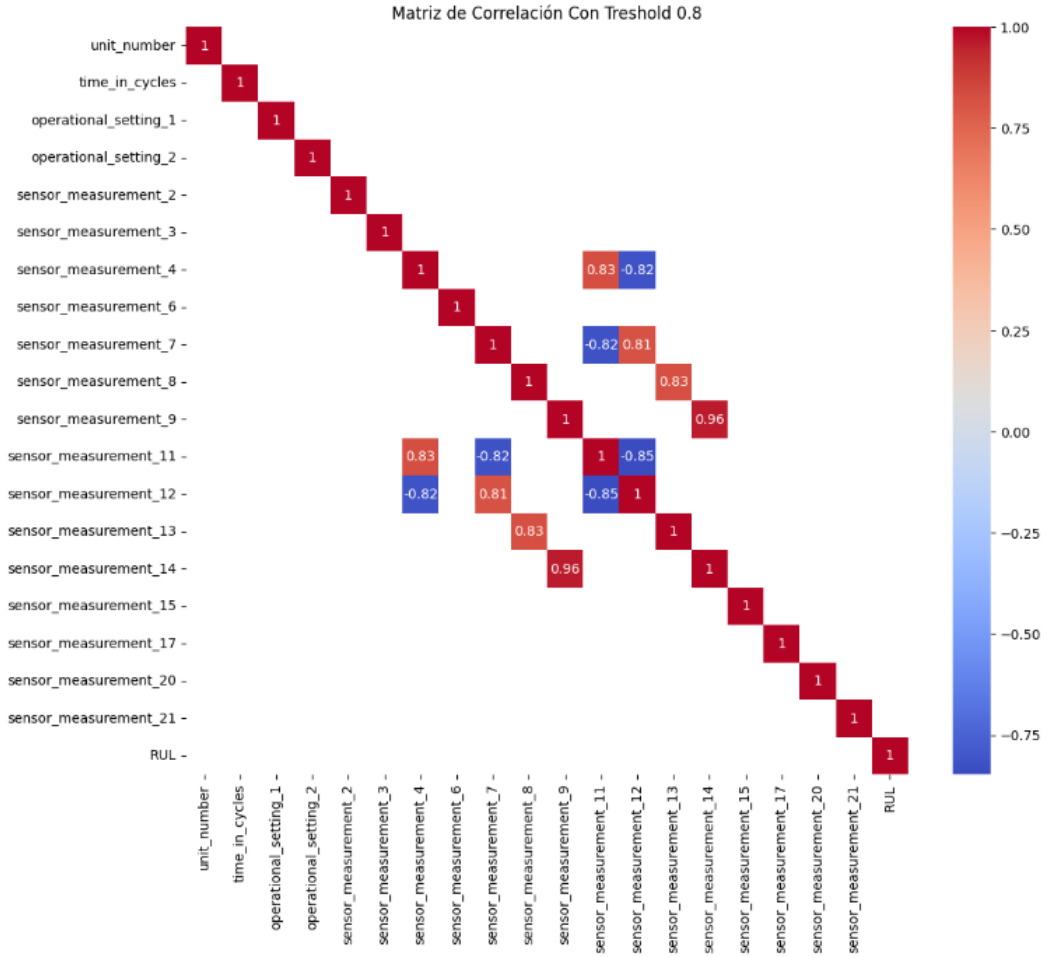


Figura 4.8: Matriz de Correlación tras Imponer un Umbral de 0.8

Destacan varias correlaciones fuertes entre algunos sensores. Los pares de sensores fuertemente correlacionados tras imponer un treshold de 0.8 son: 4-11, 7-12, 8-13, 9-14.

6. Valores atípicos

Los valores atípicos son datos que se desvían significativamente de otros en el conjunto, y pueden llegar a ser indicativos de errores de medición o fenómenos poco comunes. Identificarlos es crucial para evitar un posible bias en el modelo que pueda distorsionar los futuros resultados. Para calcular estos valores, se usa el método del rango intercuartílico (IQR)Se identifican valores por debajo del primer cuartil Q1 (-[1,5 x IQR]), y por encima del cuartil Q3 (1,5 x IQR).

```
Outliers encontrados:
operational_setting_1     105
operational_setting_2       0
operational_setting_3       0
sensor_measurement_1        0
sensor_measurement_2      128
sensor_measurement_3      165
sensor_measurement_4      120
sensor_measurement_5        0
sensor_measurement_6      406
sensor_measurement_7      110
sensor_measurement_8      320
sensor_measurement_9     1686
sensor_measurement_10       0
sensor_measurement_11     167
sensor_measurement_12     146
sensor_measurement_13     161
sensor_measurement_14    1543
sensor_measurement_15     120
sensor_measurement_16       0
sensor_measurement_17      81
sensor_measurement_18       0
sensor_measurement_19       0
sensor_measurement_20     117
sensor_measurement_21     136
dtype: int64
```

Figura 4.9: Tabla Valores Atípicos de las Características

Como se puede observar, varias características cuentan con valores atípicos. El estudio de valores atípicos es una fase importante en el análisis exploratorio, sin embargo, para este proyecto específico nos interesa mantener la mayor cantidad posible. Para preservar la integridad y continuidad de los datos, sin perder ciclos importantes para el análisis, se decide no eliminar los valores.

4.3 SELECCIÓN Y DESARROLLO DEL MODELO

4.3.1 SELECCIÓN DEL MODELO

En cualquier proyecto de análisis avanzado de datos, la selección del algoritmo adecuado es la parte más importante de un proyecto. En este Trabajo Fin de Grado, se trata de predecir la RUL de los motores del dataset CMAPSS, por lo que se pueden considerar varios algoritmos de Machine Learning.

Uno de los algoritmos más simples y comunes para estas tareas de regresión es la Regresión Lineal, la cual es facil de entrenar e interpretar. Por otro lado, una interesante segunda opción son los Árboles de decisión o Random Forests, los cuales son algoritmos muy versátiles que pueden tratar con datos no lineales y combinaciones difíciles de características.

Finalmente, después de considerar ventajas y desventajas, se decide utilizar Support Vector Regression (SVR). La decisión consta de varios factores:

1. Capacidad para Trabajar con Alta Dimensionalidad
2. Generalización eficaz
3. Flexibilidad
4. Robustez pese a Intenso uso Computacional

Como se introdujo anteriormente, en la sección de algoritmos de aprendizaje no supervisado, SVR es la utilización de Support Vector Machines (SVM) para tareas de regresión.

Las SVM son algoritmos muy versátiles usados principalmente para clasificación, mediante la búsqueda del hiperplano que mejor separe las clases en el espacio de las características. Sin embargo, su funcionamiento puede extenderse a tareas más allá de la clasificación, extrapolando su uso en problemas de regresión a través de SVR.

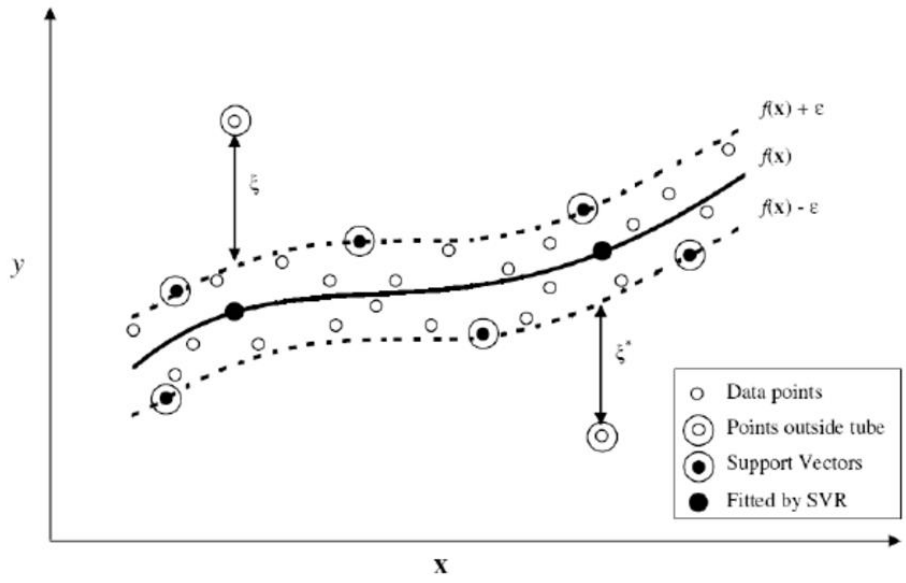


Figura 4.10: Diagrama Explicación Visual del Funcionamiento de SVR

SVR adapta los mismos principios de las SVM a problemas donde el objetivo es predecir valores continuos en cambio de clases discretas. El objetivo es encontrar una función lo más cercana posible de los valores reales de los datos, que prediga los valores continuos.

Como se puede observar en la Figura 4.10, interesa que la función sea lo más plana o "suave" posible, sin ser demasiado irregular pero dentro de un cierto rango de tolerancia (**ε**) siendo la principal diferencia con respecto a una regresión lineal. Los puntos que caen dentro de estos límites se ignoran a la hora de minimizar la función de pérdida durante el ajuste del modelo. Ajustar el modelo usando solamente los puntos fuera del límite **ε** reduce la carga computacional y permite capturar comportamientos más complejos.

4.3.2 Desarrollo del Modelo:

1. Modelo Baseline

Para poder medir la efectividad de cualquier modelo predictivo, es importante empezar con un modelo base. Este modelo base es útil como punto de referencia para comparar el funcionamiento de modelos más avanzados. En los problemas de análisis predictivo, se usa frecuentemente un modelo de regresión lineal como baseline debido a su simplicidad.

```
######################################################################
#Creacion y ajuste del modelo de Regresion Lineal
lm = LinearRegression()
lm.fit(X_train, y_train)

#Predicción y Evaluación del conjunto de entrenamiento
y_predicted_train = lm.predict(X_train)

mse_train = mean_squared_error(y_train, y_predicted_train)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_predicted_train)
mae_train = mean_absolute_error(y_train, y_predicted_train)
print('Conjunto Train RMSE: {}, R2: {}, MAE: {}'.format(rmse_train, r2_train, mae_train))

#Predicción y Evaluación del conjunto de prueba
y_predicted_test = lm.predict(X_test)

mse_test = mean_squared_error(y_test, y_predicted_test)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_predicted_test)
mae_test = mean_absolute_error(y_test, y_predicted_test)
print('Conjunto Test RMSE: {}, R2: {}, MAE: {}'.format(rmse_test, r2_test, mae_test))

######################################################################
```

✓ 0.0s

```
Conjunto Train RMSE: 44.66819159545423, R2: 0.5794486527796773, MAE: 34.11528656498055
Conjunto Test RMSE: 31.952633027734382, R2: 0.40877368076601606, MAE: 25.540326671232616
```

Figura 4.11: Extracto de Código de la Creación del Modelo Baseline

Los resultados para el modelo de regresión lineal muestran que hay un margen considerable de mejora, ya que el modelo no está capturando completamente la variabilidad de los datos, especialmente en el conjunto Test.

Esta diferencia sugiere que el modelo puede estar sufriendo underfitting por no estar aprendiendo suficientes patrones de los datos.

2. Desarrollo de SVR

Para mejorar el aprendizaje, una de las estrategias que se seguirán será el re ajuste del tratamiento del RUL, haciendo uso de la función ".clip" de Numpy para manejar mejor los valores extremos de esta variable. Esto puede ayudar a mejorar la capacidad del modelo para predecir valores bajos del RUL (críticos en el mantenimiento predictivo).

```
#########################################################################
#SVR
from sklearn.svm import SVR

#Creación y Ajuste del modelo SVR
svr = SVR(kernel='linear')
svr.fit(X_train, y_train_conclip)

#Predicción y Evaluación del conjunto de entrenamiento
y_predicted_train = svr.predict(X_train)
mse_train = mean_squared_error(y_train_conclip, y_predicted_train)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train_conclip, y_predicted_train)
mae_train = mean_absolute_error(y_train_conclip, y_predicted_train)
print('Conjunto Train RMSE: {}, R2: {}, MAE: {}'.format(rmse_train, r2_train, mae_train))

#Predicción y Evaluación del conjunto de prueba
y_predicted_test = svr.predict(X_test)
mse_test = mean_squared_error(y_test, y_predicted_test)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_predicted_test)
mae_test = mean_absolute_error(y_test, y_predicted_test)
print('Conjunto Test RMSE: {}, R2: {}, MAE: {}'.format(rmse_test, r2_test, mae_test))
#########################################################################
```

✓ 18.3s

```
Conjunto Train RMSE: 21.57826397506789, R2: 0.7318795396979632, MAE: 17.55405145918994
Conjunto Test RMSE: 21.580480163289746, R2: 0.7303113540952123, MAE: 17.205234410172086
```

Figura 4.12: Extracto de Código de la Creación del Modelo SVR

Se observan mejoras significativas en todas las métricas utilizadas. En el Error Cuadrático Medio (RMSE) se ha obtenido una mejora del 51.7% en el Conjunto Train, y un 32.4% en el Conjunto Test. Esto indica que el modelo SVR cuenta con un error de predicción

mucho menor, dejando claro que SVR es capaz de capturar mucho mejor las relaciones no lineales en los datos, con mejor precisión general de las predicciones.

El Coeficiente de Determinación (R^2), en el Conjunto Train, mejoró un 26% mientras que en el Conjunto Test se obtuvo una notable mejora del 78%. Esta mejora muestra que el modelo SVR es más eficaz a la hora de capturar la estructura subyacente de los datos explicando una mayor proporción de la varianza en los datos de RUL.

Por último, las predicciones de SVR están más cerca de los valores reales de RUL, reflejándose en la mejora del 48,5% en el Conjunto Train y del 32.6% en el Conjunto Test.

3. Mejoras al Modelo

Para mejorar el modelo inicial, se usan dos técnicas esenciales de ingeniería de características: la creación de características polinómicas, y la selección de las características más relevantes.

La creación de características polinómicas consiste en la agrupación o combinación entre características existentes. Este enfoque puede revelar patrones o tendencias que no son tan evidentes solamente con las características originales. Por ejemplo, si se tienen dos características X e Y, las nuevas características resultantes de su combinación polinómica pueden ser $X^2, Y^2, X \cdot Y$. En este caso se hizo esto con todas las características.

```
######################################################################
from sklearn.preprocessing import PolynomialFeatures

#Combinaciones polinómicas
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.fit_transform(X_test)

#Características antes y después
print("Número de características antes de la transformación:", X_train.shape[1])
print("Número de características después de la transformación:", X_train_poly.shape[1])
######################################################################
```

✓ 0.0s

```
Número de características antes de la transformación: 14
Número de características después de la transformación: 120
```

Figura 4.13: Extracto de Código de la Combinación de Características Polinómicas

Como se ve en la salida del extracto de código de la Figura 4.13, debido a las nuevas combinaciones se aumenta el número de las características a estudiar. Como es de esperar, este aumento de número puede dar lugar a nuevos patrones muy útiles, pero también a nuevas combinaciones ruidosas que pueden no aportar nueva información válida, por lo que se llevará a cabo una selección de características para mantener las que más contribuyen a la mejora del rendimiento del modelo. Para ello se utiliza la función "SelectFromModel" poniendo el umbral = "mean", para que únicamente se mantengan las características cuya importancia (o coeficiente absoluto) sea mayor que la media de todas las importancias de las características. Con esto se consigue reducir la dimensionalidad mejorando el rendimiento, disminuyendo la complejidad computacional, mejorando la interpretabilidad, y disminuyendo el riesgo de overfitting.

Una vez ajustado el modelo final, se hacen las predicciones y evaluaciones pertinentes.

```
######################################################################
#Creación y Ajuste de la segunda versión del modelo SVR
svr_2 = SVR(kernel='linear')
svr_2.fit(X_train_transformed[:, selected_c2], y_train_conclip)

#Predicción y Evaluación del conjunto de entrenamiento
y_predicted_train_2 = svr_2.predict(X_train_transformed[:, selected_c2])
mse_train = mean_squared_error(y_train_conclip, y_predicted_train_2)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train_conclip, y_predicted_train_2)
mae_train = mean_absolute_error(y_train_conclip, y_predicted_train_2)
print('Conjunto Train RMSE: {}, R2: {}, MAE: {}'.format(rmse_train, r2_train, mae_train))

#Predicción y Evaluación del conjunto de prueba
y_predicted_test_2 = svr_2.predict(X_test_transformed[:, selected_c2])
mse_test = mean_squared_error(y_test, y_predicted_test_2)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_predicted_test_2)
mae_test = mean_absolute_error(y_test, y_predicted_test_2)
print('Conjunto Test RMSE: {}, R2: {}, MAE: {}'.format(rmse_test, r2_test, mae_test))
######################################################################
```

✓ 24.6s

```
Conjunto Train RMSE: 19.746789101481127, R2: 0.7754619593165268, MAE: 15.670850363968567
Conjunto Test RMSE: 20.55613819605453, R2: 0.755305891345072, MAE: 15.84224221959272
```

Figura 4.14: Extracto de Código del Ajuste del Modelo SVR final

Finalmente, en la Figura 4.14 se refleja la última mejora notable del modelo final.

4.4 DESARROLLO DEL DASHBOARD

En esta sección, se detalla el proceso del desarrollo del dashboard interactivo utilizando la herramienta Power BI. El principal objetivo es ofrecer una herramienta visual y dinámica que ejerza como un simulacro de monitoreo del estado de los motores para que hipotéticos mecánicos y gestores de mantenimiento puedan tomar decisiones sobre el mantenimiento de estos.

El dashboard se construirá haciendo uso del conjunto FD001 con el que se ha trabajado anteriormente, concretamente con el conjunto *train_FD001* y *RUL_FD001*, imaginando que el flujo de datos fuese en tiempo real sobre mediciones hechas en vivo de los sensores.

El dashboard tiene varios objetivos:

- Monitoreo: permitir la visualización del estado actual de los motores, mostrando datos relevantes como el Remaining Useful Life, mediciones de sensores y otros parámetros.
- Identificación de Motores Críticos: ofrecer una manera rápida y eficiente de identificar motores que requieren atención inmediata.
- Interactividad: proporcionar opciones de filtrado y selección para personalizar la visualización de los datos según las necesidades.

A continuación, el proceso de desarrollo:

1. **Preparación de Datos**

Se cargan los datos en Power BI realizando transformaciones necesarias para el formato correcto.

Como se puede observar en la figura 4.15, existen varias fuentes posibles para la importación de los datos en Power BI. En este caso, se hace uso de la importación mediante “Texto o CSV” ya que es el formato de nuestro dataset.

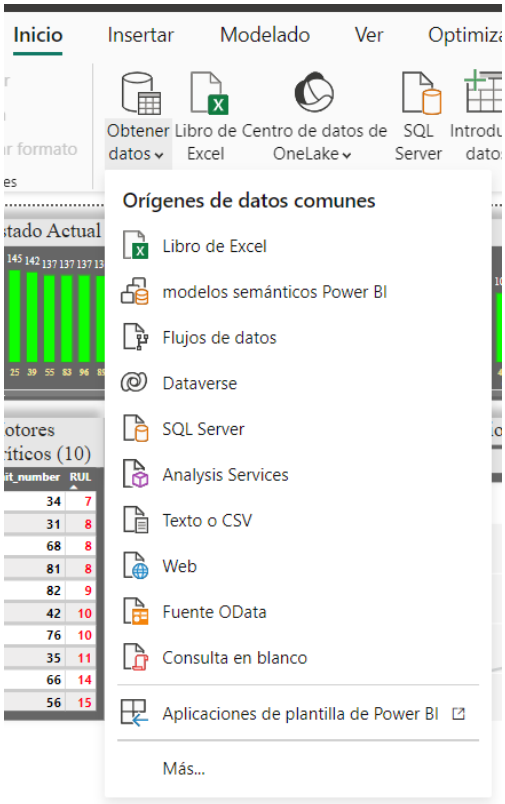


Figura 4.15: Importación Inicial de datos

La transformación inicial se lleva a cabo cambiando los signos "." por "," para que pueda detectar que el número es decimal, y después elegir el correcto tipo de dato en *Power Query*, además de deshacerse de las columnas que se decidieron eliminar.

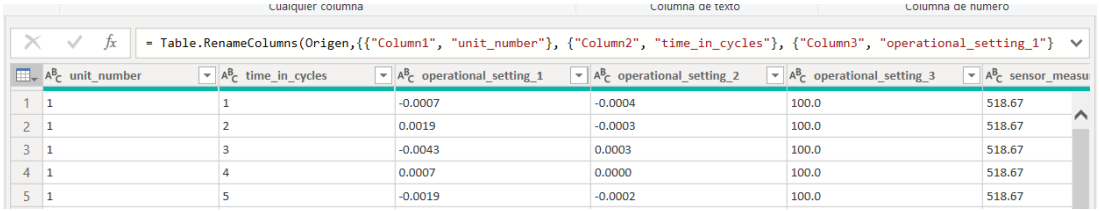


Figura 4.16: Datos previos a la Transformación inicial

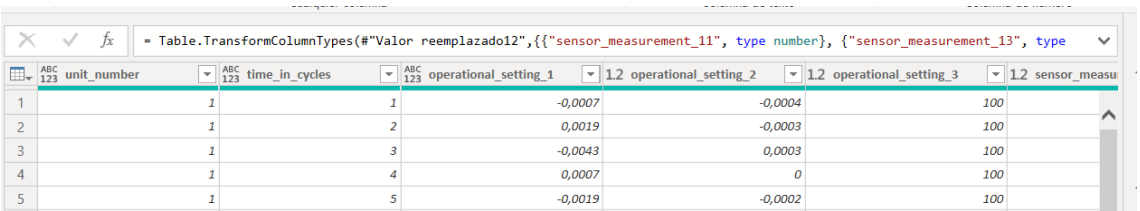


Figura 4.17: Datos posteriores a la Transformación inicial

Para la correcta administración de la relación entre *train_FD001* y *RUL_FD001*, se crea una nueva columna "*unit_number*" en el conjunto *RUL_FD001* para poder relacionar los conjuntos de forma "uno a varios".

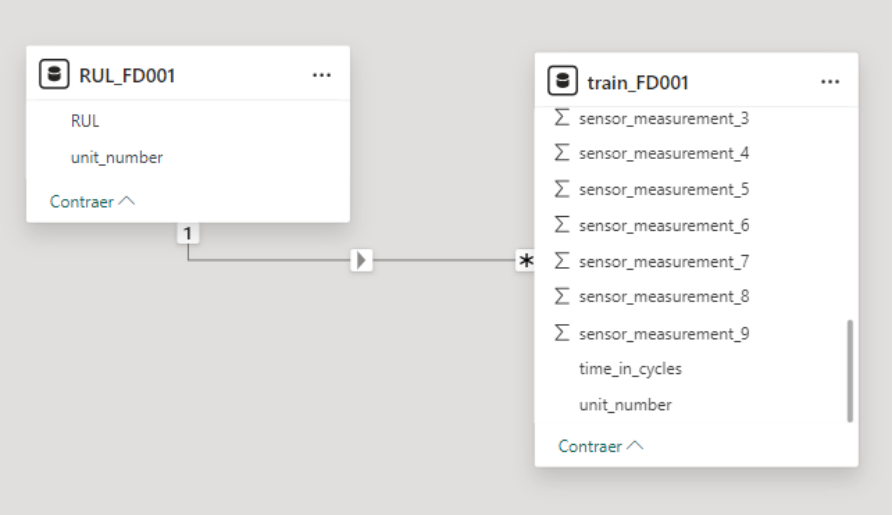


Figura 4.18: Relación "uno a varios" entre train_FD001 y RUL_FD001

2. Diseño de Visualizaciones

- *"Estado Actual de los Motores"*

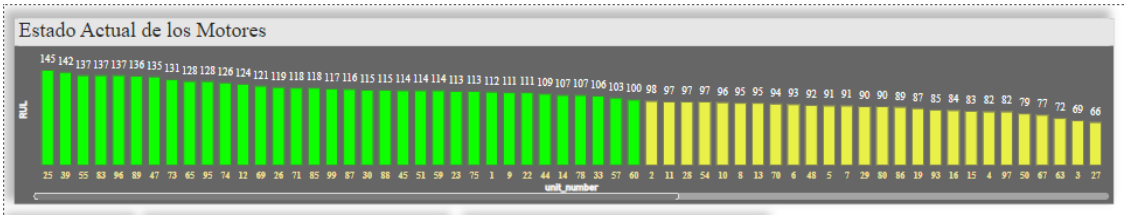


Figura 4.19: Visualización "Estado Actual de los Motores" del Dashboard

Esta primera visualización tiene como objetivo mostrar el estado actual de los motores haciendo uso de un gráfico de barras representando el RUL de cada motor. Las barras, las cuales están ordenadas de mayor a menor, cuentan con una codificación de color para indicar el nivel de criticidad. Si se encuentra en la franja **[150, 100]** será **Verde** indicando que está en buen estado, si está en la siguiente franja **(100, 50]** será **Amarillo**, significando que requieren monitoreo, y por último si se encuentran en la franja **<50**, se indicará en **Rojo** queriendo decir que se necesita atención inmediata.

Cada barra está etiquetada con el numero de motor correspondiente pudiendo así identificar rápidamente cada unidad.

- *"Motores Críticos (10)"*

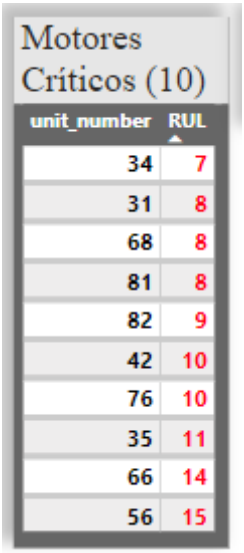


Figura 4.20: Visualización "Motores Críticos (10)" del Dashboard

Esta segunda visualización muestra los diez motores que tienen el RUL más bajo, representando los motores verdaderamente críticos que necesitan atención inmediata. Cada fila de la tabla incluye el número de unidad del motor y su RUL correspondiente, ordenados de más a menos crítico. Permite visualizar rápidamente los motores con acciones de mantenimiento prioritarias a las demás.

- *"Medición del sensor por Ciclos"*

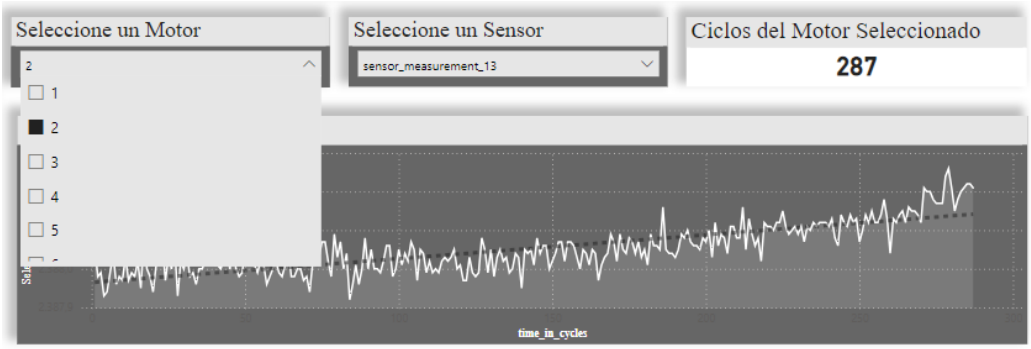


Figura 4.21: Visualización "Medición del sensor por Ciclos" del Dashboard

El objetivo de esta tercera visualización es ofrecer una visión más detallada del comportamiento de u motor específico a lo largo de su ciclo de vida, a medida que estos ciclos aumentan.

Consta de 2 selecciones: “Selección de motor” y “Selección de sensor”, con el que se puede concretar qué sensor de qué motor se quiere ver. La selección final será reflejada en el gráfico de líneas de la parte de abajo mostrando la tendencia y reflejando los ciclos totales del motor seleccionado en la parte superior derecha.

- DASHBOARD FINAL

Figura 4.22: Dashboard Final

Finalmente, como se puede apreciar en la Figura 4.22, el Dashboard final constará de la agrupación de todas las visualizaciones comentadas, habiendo añadido una especie de chuletario en forma de tabla de pandas como objeto visual de Python, para que el hipotético ingeniero o mecánico sepa a qué sensor se esta haciendo referencia en cada momento.

A modo de conclusión, este dashboard está diseñado para ofrecer una herramienta visual y funciona que ayude a los responsables de mantenimiento para monitorizar los motores. Deja ver de un vistazo qué motores requieren atención, analizar los datos de los sensores para detectar patrones y anomalías haciendo uso de posibles límites conocidos, y tomar decisiones en base a esta información. La adición de la tabla informativa de sensores asegura que los usuarios tengan toda la información necesaria al alcance de la mano.

CAPÍTULO 5. **RESULTADOS, CONCLUSIONES Y TRABAJO FUTURO**

El objetivo principal de este proyecto fue desarrollar un modelo de predicción para lograr estimar la vida útil (RUL) de los motores utilizando el conjunto CMAPSS y crear un dashboard interactivo en Power BI para facilitar su monitoreo y mantenimiento.

MODELO SVR

Para el desarrollo del modelo predictivo, se hizo una exploración de diferentes técnicas de regresión, eligiendo finalmente Support Vector Regression (SVR) por su capacidad para manejar alta dimensionalidad de características y generalización eficaz.

En la Figura 5.1 se muestran los desempeños en el conjunto Train y Test, del modelo baseline, el modelo SVR inicial, y el Modelo SVR mejorado, respectivamente.

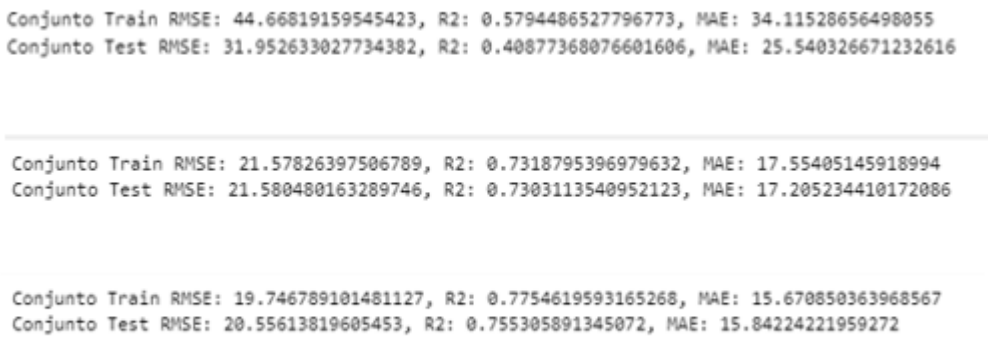


Figura 5.1: Comparación de Resultados de los Modelos Utilizados. Baseline, SVR inicial y SVR mejorado, respectivamente

El modelo mejorado demostró una mejora significativa en comparación con el modelo inicial. Se consiguió reducir el error cuadrático medio (RMSE), el error absoluto medio (MAE) y aumentar el coeficiente de determinación (R^2) gracias al uso de combinaciones polinómicas y la posterior selección de las características más relevantes, evitando el sobreajuste y mejorando su generalización.

DASHBOARD EN POWER BI

Posteriormente, para la elaboración del dashboard en power BI, se llevó a cabo la carga, transformación, y procesamiento de los datos en la herramienta Power Bi, también haciendo uso de Power Query. Después, tras hacer la relación entre las tablas cargadas, se diseñaron diferentes gráficas apilándolas en el cuadro de mandos o dashboard final, ofreciendo ciertos insights importantes: Estado Actual de los Motores, Motores Críticos, Ciclos del Motor Seleccionado, Medición del Sensor por Ciclos.

Cabe destacar, que el planteamiento de este dashboard se ha realizado con la intención de ofrecer una herramienta eficaz para el monitoreo y mantenimiento de los motores. Sin embargo, es importante dejar claro que este es un modelo inicial que simula condiciones ideales con datos históricos, pero está pensado para usar datos en tiempo real. Esto quiere decir que, para alcanzar su máximo potencial, sería interesante integrarlo con sistemas de adquisición de datos en tiempo real, lo cual permitiría una monitorización continua y actualizada del estado de los motores.

Conclusiones

El desarrollo de este proyecto ha permitido lograr varios objetivos clave planteados inicialmente.

Se ha desarrollado un modelo SVR robusto y preciso, usando técnicas avanzadas. El modelo mejorado demostró una capacidad significativa para predecir con precisión la vida útil restante de los motores del conjunto de datos CMAPSS.

Por otro lado, el dashboard en Power BI proporciona una herramienta valiosa para el mantenimiento de los motores. Facilita la visualización del estado de éstos, y el análisis en detalle de las mediciones de los diferentes sensores.

Por último, se incluye una revisión detallada de la literatura, consiguiendo proporcionar el contexto necesario para poder entender el desarrollo del modelo, el contexto científico actual y la implementación del dashboard.

Trabajo Futuro

A pesar de los logros alcanzados, existen varias áreas en las que se podría mejorar y expandir en futuros trabajos:

1. **Datos en Tiempo Real:** Integrar datos en tiempo real para reflejar el estado de los motores continuadamente, ayudando a detectar problemas reales eficientemente.
2. **Visualizaciones Adicionales:** Incluir más gráficos con información de interés como fallas históricas de los motores, sensores dañados, etc. Esto ayudaría a planificar mejor el mantenimiento y predecir futuros fallos
3. **Modelo Predictivo Mejorado:** Explorar y mejorar otros algoritmos de aprendizaje automático, como redes neuronales profundas o investigar técnicas de ensamblado para combinar modelos y mejorar las predicciones.

Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado

ADVERTENCIA: Desde la Universidad consideramos que ChatGPT u otras herramientas similares son herramientas muy útiles en la vida académica, aunque su uso queda siempre bajo la responsabilidad del alumno, puesto que las respuestas que proporciona pueden no ser veraces. En este sentido, NO está permitido su uso en la elaboración del Trabajo fin de Grado para generar código porque estas herramientas no son fiables en esa tarea. Aunque el código funcione, no hay garantías de que metodológicamente sea correcto, y es altamente probable que no lo sea.

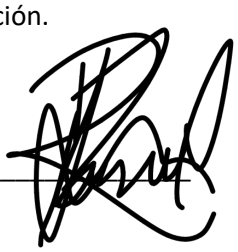
Por la presente, yo, Daniel Losada Rueda, estudiante de Ingeniería de las Telecomunicaciones y Business Analytics de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado "DESARROLLO DE UN DASHBOARD VISUAL Y PREDICCIÓN SOBRE DATOS CMAPSS", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

1. **Brainstorming de ideas de investigación:** Utilizado para idear y esbozar posibles áreas de investigación.
2. **Referencias:** Usado conjuntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
3. **Metodólogo:** Para descubrir métodos aplicables a problemas específicos de investigación.
4. **Interpretador de código:** Para realizar análisis de datos preliminares.
5. **Corrector de estilo literario y de lenguaje:** Para mejorar la calidad lingüística y estilística del texto.
6. **Traductor:** Para traducir textos de un lenguaje a otro.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: 21/06/2024

Firma: _____



CAPÍTULO 6. BIBLIOGRAFÍA.

[1] IBM. (n.d.). **Machine Learning: Conceptos y Aplicaciones**. IBM. Available: IBM Machine Learning.

[2] Amazon Web Services. (n.d.). **¿Qué es el Machine Learning?**. AWS. Available: AWS Machine Learning.

[3] Future Space. (2022). **Machine Learning: Los Orígenes y la Evolución**. Available: Future Space Machine Learning.

[4] Telefónica Tech. (2021). **¿Qué Algoritmo Elegir en Machine Learning?**. Available: Telefónica Tech Algoritmos ML.

[5] Chang Woo Hong. (2021). **List and Physical Meanings of the C-MAPSS Dataset**. Available: ResearchGate C-MAPSS Dataset.

[6] Analytics Vidhya. (2020). **Tutorial de Support Vector Regression para Machine Learning**. Available: Analytics Vidhya SVR Tutorial.

[7] Verma, N. (2020). **Introducción a Support Vector Regression (SVR) en Machine Learning**. Medium. Available: Medium SVR Introduction.

[8] Microsoft. (n.d.). **Power BI: Herramienta de Visualización de Datos**. Available: Microsoft Power BI.

[9] BERGHOUT Tarek. (2021). **Preparación y Ejemplo de Aplicación del Nuevo Conjunto de Datos C-MAPSS 2021**. Available: MathWorks C-MAPSS Dataset.

[10] NASA. (n.d.). **CMAPSS Jet Engine Simulated Data**. Available: NASA CMAPSS Data.

[11] Iberdrola. (n.d.). **Deep Learning**. Available: Iberdrola Deep Learning.

[12] Mecalux. (n.d.). **Mantenimiento Predictivo: Claves y Herramientas**. Available: Mecalux Mantenimiento Predictivo.

[13] Tableau. (n.d.). **¿Qué es Tableau?**. Available: Tableau.

[14] Matplotlib. (n.d.). **Documentación de Matplotlib**. Available: Matplotlib.

[15] DataScientest. (2021). **Cómo Hacer Data Visualization con Plotly**. Available: DataScientest Plotly.

[16] Seaborn. (n.d.). **Documentación de Seaborn**. Available: Seaborn.

[17] Python Software Foundation. (n.d.). **Python**. Available: Python.

[18] Asociación para el Progreso de la Dirección (APD). (2021). **Algoritmos del Machine Learning**. Available: APD Machine Learning.

[19] Microsoft Learn. (n.d.). **Introducción a Power BI**. Available: Microsoft Learn Power BI.

[20] Sandip Kumar Lahiri, Kartik Chandra Ghanta. (2008). **Diagrama Esquemático de Support Vector Regression con Función de Pérdida Sensible-e**. Available: ResearchGate SVR.

[21] Gabriel Duarte Pasa, Ivo Paixao de Medeiros, Takashi Yoneyama. (2021). **Resumen de los Subconjuntos del Conjunto de Datos CMAPSS**. Available: ResearchGate CMAPSS Subsets.

[22] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Available: Springer Link

[23] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Available: MIT Press

[24] Peng, Y., Zhang, L., & Liu, X. (2010). A review on prognostics and health management (PHM) methods for engineering systems. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, *5*(1), 61-73.

ANEXO I

Enlace al Código del Proyecto

El código utilizado para desarrollar el modelo se encuentra disponible en el siguiente enlace de GitHub:

https://github.com/losadarr/TFG-Analytics-Code

Enlace al Dashboard Interactivo

El dashboard interactivo desarrollado en Power BI está disponible en el siguiente enlace:

https://app.powerbi.com/view?r=eyJrIjoiZjczZDU4YWItZGRlMS00MzVkLWE1ZGItMTE1YTAzYzEyZWJjIiwidCI6ImJjZDI3MDFjLWFhOWItNGQxMi1iYTIwLWYzZTNiODMwNzBjMSIsImMiOjh9