ESCUELA DE NEGOCIOS

MAESTRÍA EN INTELIGENCIA DE NEGOCIOS Y CIENCIA DE DATOS

MODELO DE PREDICCIÓN O DIAGNÓSTICO EN LENGUAJE PYTHON, USANDO TÉCNICAS DE MACHINE LEARNING DE UN MOTOR TIPO TURBOFAN

Profesor

Manuel Eugenio Morocho

Autor

Alejandro Andrés Cuzco Peñafiel

2024

RESUMEN

Tanto en la industria aeronáutica como en otras industrias, es prioritario el desarrollo de soluciones estratégicas. En el ámbito de la aviación, siendo un pilar importante en la transportación tanto mundial como nacional, los motores que usan las aeronaves deben operar en perfecto estado, ya que un fallo de los mismos conllevaría la vida de cientos y hasta miles de personas. Bajo ese contexto, se propone realizar un modelo de predicción o diagnóstico en Python para motores del tipo turbofans.

Por esta razón, se propone la utilización de algoritmos de Regresión Lineal, Random Forest para la predicción del RUL de motores turbofan. La simulación de la degradación del motor se llevó a cabo utilizando C-MAPSS, un paquete de datos publicado por la NASA.

Este modelo es lo más cercano a la realidad, debido a que los datos usados son una simulación creada por la NASA. Tras aplicar este modelo se han entrenado los dos algoritmos, mediante el uso de lenguaje Python, y se han comparado las métricas con las obtenidas anteriormente, donde podemos observar que Random Forest proporciona los mejores resultados.

Palabras claves: RUL, sensores, mantenimiento, detección

ABSTRACT

Both in the aeronautical industry and in other industries, the development of strategic solutions is a priority. In the field of aviation, being an important pillar in both global and national transportation, the engines used by aircraft must operate in perfect condition, since a failure of the same would lead to the lives of hundreds and even thousands of people. In this context, it is proposed to create a prediction or diagnosis model in Python for turbofan type engines.

For this reason, the use of Linear Regression algorithms, Random Forest for the prediction of the RUL of turbofan engines is proposed. The simulation of the engine degradation was carried out using C-MAPSS, a data package published by NASA.

This model is the closest to reality, because the data used is a simulation created by NASA. After applying this model, the two algorithms have been trained, using Python language, and the metrics have been compared with those obtained previously, where we can observe that Random Forest provides the best results.

Keywords: RUL, sensors, maintenance, detection

Contenido

Índice de Tablas

ÍNDICE DE FIGURAS:

1. Introducción.

Hoy por hoy, la industria aeronáutica es un pilar fundamental en la economía de todos los países y en la transportación global. Dentro de la industria a través del uso y del tiempo, la ocurrencia de fallos en motores, y demás equipamiento de navegación es mucho más probable. Esto puede llegar a ser un gran problema para las aerolíneas, generando atrasos, suspensiones de vuelos, e incluso accidentes. Es por ello que los programas de mantenimiento tanto predictivo, como preventivo y la adopción de estrategias tecnológicas diversas juegan un rol fundamental y primordial para abordar ciertos desafíos que propone dicha industria.

Uno de los mayores desafíos es la reducción de los tiempos de para y un mejor manejo de los programas de mantenimiento que aporte a la operatividad de la empresa. Bajo esa óptica se propone establecer planes de mantenimiento predictivo con el enfoque a posteriori de levantar estrategias empresariales basadas en la analítica de datos.

El estudio se centrará en la base de datos CMAPSS realizada por la NASA en la que se detallan diferentes características de motores turbofan con sus ciclos de vida restantes antes del fallo. Aplicaremos las metodologías de Aprendizaje Supervisado a esta base de datos y realizaremos una comparativa de las métricas alcanzadas por cada uno de los algoritmos

Otro aspecto a tomar en cuenta es la influencia y aporte que tiene la exactitud de las mediciones de cada uno de los tipos de sensores. Debido a que en la base de datos no se tiene en cuenta esta incertidumbre de cada uno de los sensores, se ha analizado el aporte de la medición de cada uno de ellos en el cálculo del RUL.

Finalmente, se realizará el análisis comparativo entre los resultados del análisis por RMSE y por Random Forest con el fin de verificar el mejor rendimiento para la predicción de vida útil restante, que en nuestro caso será Random Forest.

2. Revisión de literatura relacionada al problema.

2.1 Discusión de los hallazgos de la literatura académica

La industria aeronáutica es una de las industrias más vanguardista y tecnológicas dentro del campo de la ingeniería. Así como otras industrias, busca la reducción de costos y la mejora de sus procesos, con el objetivo de tener un sector cada vez más sostenible, moderno, ecológico y avanzado. En este aspecto es donde el mantenimiento predictivo se vuelve importante y fundamental. Dicho de otra manera, mantenimiento predictivo te brinda la capacidad de poder estimar cuándo un componente puede fallar, minimizando la posibilidad de cualquier fallo en el componente.

Es por eso que no solo es necesario comprender cómo funcionan los sistemas y componentes, sino que también es necesario saber cómo actuarán con el tiempo y su posible degradación. Aquí es donde entra el campo del pronóstico, que tiene como objetivo predecir el estado futuro de un elemento teniendo en cuenta su estado actual y sus estados pasados.

La efectividad de la vida útil de un elemento es tan importante que cada vez más, grandes empresas como Airbus se embarcan en este mundo. Junto a esto, todos los días salen a la luz nuevos métodos de análisis predictivo, mejorando los métodos existentes o incluso revolucionando el sector, haciendo que todo lo anterior quede obsoleto. Este es el caso de las redes neuronales que, desde sus primeras aplicaciones, sus usos no han dejado de crecer, dando resultados y precisión nunca antes alcanzados, haciéndolos especialmente atractivos para esta tarea (Sánchez, 2020).

Tanto en la industria aeroespacial, como en otras industrias (farmacéutica, comercio, medicina, educación) las técnicas de Big Data mejoran significativamente la toma de decisiones. A través de la información basada en evidencia, lo cual reduce la incertidumbre y aumenta la precisión en la planificación estratégica (Bedoya, Góngora, & García, 2024).

Big Data contribuye a la optimización de procesos y la eficiencia operacional. La capacidad de analizar grandes volúmenes de datos en tiempo real permite a las

organizaciones identificar y eliminar ineficiencias, mejorar la productividad y reducir costos.

2.1.1 Historia y evolución de los motores turbo fans.

En la aviación son comunes los motores tipo “turbofan” o también conocidos como turboventilador, el cual es un motor de reacción (Heywood, 1988).

2.1.2 Orígenes y Primeros Desarrollos

La idea de esta clase de turbinas surgió en la época de la Segunda Guerra Mundial, debido a la necesidad de aviones más rápidos y eficientes. El primer motor a reacción operativo, el Heinkel HeS 3, diseñado por Hans von Ohain, voló en 1939. Sin embargo, este era un motor de flujo axial puro, que carecía del ventilador adicional que caracteriza a los motores turbofans modernos (García Chico, 2022).

Para 1940, el ingeniero británico Frank Whittle, uno de los pioneros de los motores a reacción, sugirió la idea de añadir un ventilador grande al frente de un motor a reacción para mejorar la eficiencia del combustible y reducir el ruido. No obstante, fue en la década de 1950 cuando se realizaron los primeros desarrollos significativos en motores turbo fans, impulsados principalmente por la creciente demanda de aviación comercial (UKPO, 1998).

2.1.3 Desarrollos Claves en las Décadas de 1950 y 1960

El Pratt & Whitney JT3D, desarrollado en 1958, es considerado uno de los primeros motores turbo fans comerciales exitosos. Basado en el diseño del motor a reacción JT3C, el JT3D (Lemus & Werner, 2017) incorporaba un ventilador que mejoraba significativamente la eficiencia del combustible. Este motor impulsó aviones icónicos como el Boeing 707 y el Douglas DC-8, marcando el comienzo de una nueva era en la aviación comercial (Manjarrés, 2002).

Durante la década de 1960, hubo grandes avances en la ciencia de materiales y diseño aerodinámico que permitieron la creación de motores turbo fans más eficientes y potentes (Rodrigo Ramírez, 2022). El Rolls-Royce RB211, desarrollado a finales de la década de 1960, introdujo el uso de materiales

compuestos y nuevas tecnologías de enfriamiento, estableciendo un nuevo estándar en la industria (Car, Oller, & Oñate, 2000).

2.1.4 Evolución Tecnológica en las Décadas de 1970 y 1980

La década de 1970 vio la introducción de motores turbo fans de alto bypass, donde una mayor proporción del aire ingresa directamente al ventilador sin pasar por la cámara de combustión (Pérez & Uribe, 2022). Este diseño mejoró la eficiencia del combustible y redujo el ruido, haciéndolos ideales para la aviación comercial. El General Electric CF6, utilizado en aviones como el Boeing 747 y el Airbus A300, es un ejemplo destacado de esta generación de motores (Petrescu, 2017).

En la década de 1980, los avances en la informática y la simulación por computadora permitieron un diseño más preciso y optimización de los motores turbo fans. La introducción del motor CFM56, desarrollado por CFM International, una empresa conjunta entre General Electric y Safran Aircraft Engines, destacó por su confiabilidad y eficiencia, convirtiéndose en uno de los motores más utilizados en aviones comerciales de fuselaje estrecho (Vilana Arto, 2010).

2.1.5 Innovaciones Modernas y el Futuro

Desde la década de 1990 hasta la actualidad, los motores turbo fans han seguido evolucionando con el enfoque en la sostenibilidad y la eficiencia ambiental. Los motores modernos, como el Pratt & Whitney PW1000G y el Rolls-Royce Trent XWB, incorporan tecnologías avanzadas como el engranaje de reducción para ventiladores y materiales cerámicos avanzados que soportan temperaturas más altas, mejorando la eficiencia del combustible y reduciendo las emisiones de carbono (Hernández Bernardo, 2018).

El desarrollo de motores híbridos y eléctricos es una tendencia emergente en la industria, con el objetivo de reducir aún más el impacto ambiental de la aviación. Proyectos como el E-Fan X de Airbus y Rolls-Royce están explorando la viabilidad de sistemas de propulsión híbridos, combinando motores turbo fans tradicionales con motores eléctricos (Castro, 2023).

2.1.6 Componentes y funcionamiento básico de un motor turbo fan.

Los turbofan es el resultado de años de investigaciones con el fin de aprovechar las ventajas de los motores turborreactor y del turbohélice (Cumpsty, 2003). Esta clase de motores fueron diseñados para crear empuje adicional usando un flujo de aire secundario alrededor de la cámara de combustión. Este flujo de aire produce un mayor empuje, refrigera el circuito del motor y ayuda a suprimir el ruido del escape (Mattingly, 2002).

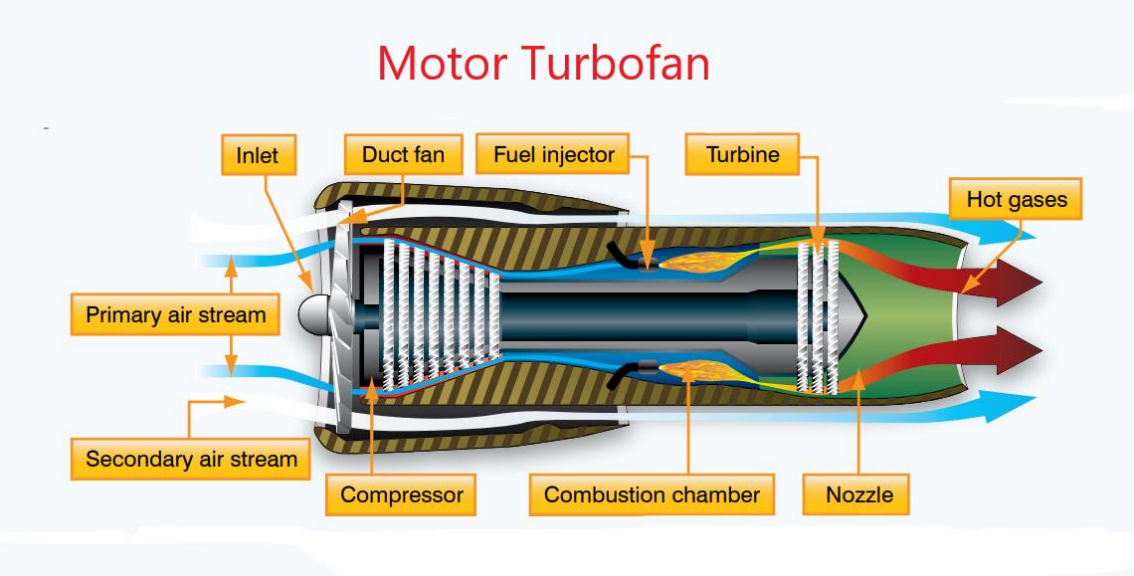

Ilustración 1: Motor turbofans. Fuente: Pilot's Handbook of Aeronautical Knowledge, FAA-H-8083-25B

1. **Entrada de Aire (Inlet)**: Es la parte delantera del motor que capta el aire y lo dirige hacia el compresor. Está diseñada para minimizar la resistencia y maximizar la eficiencia en la captación de aire.
2. **Fan (Ventilador)**: Es una gran hélice que se encuentra justo detrás de la entrada de aire. El fan mueve una gran cantidad de aire hacia el motor y alrededor del motor (bypass), proporcionando la mayor parte del empuje en un turbofan.
3. **Compresor**: Compuesto por una serie de álabes que comprimen el aire que entra, aumentando su presión y temperatura. Está dividido en etapas, generalmente un compresor de baja presión (LPC) y uno de alta presión (HPC).

4. **Cámara de Combustión**: Aquí es donde el aire comprimido se mezcla con el combustible y se quema. La combustión produce gases calientes y de alta presión.
5. **Turbina**: Compuesta por álabes similares a los del compresor, pero aquí se extrae energía de los gases calientes de combustión para hacer girar el compresor y el fan. También está dividida en turbina de alta presión (HPT) y de baja presión (LPT).
6. **Escape (Nozzle)**: La boquilla de escape acelera los gases calientes hacia fuera del motor, produciendo un empuje adicional.
7. **Bypass Duct**: Canal por donde el aire bypass (aire que no entra en el compresor, sino que pasa directamente del fan al escape) fluye alrededor del motor, proporcionando empuje adicional y aumentando la eficiencia.

2.1.7 Funcionamiento Básico

1. **Entrada de Aire**: El aire entra al motor a través del inlet.
2. **Compresión**: El fan mueve el aire hacia el compresor. Parte del aire es desviado alrededor del motor (bypass) y el resto entra en el compresor. El compresor aumenta la presión y temperatura del aire.
3. **Combustión**: El aire comprimido se mezcla con el combustible en la cámara de combustión y se quema. Esto aumenta significativamente la temperatura y presión de los gases resultantes.
4. **Turbina**: Los gases calientes pasan a través de la turbina. La energía de estos gases hace girar la turbina, que a su vez hace girar el compresor y el fan. Después de pasar por la turbina, los gases han perdido algo de su energía, pero aún tienen suficiente para producir empuje al salir por el escape.
5. **Escape**: Los gases de combustión, ahora de menor presión, se aceleran al salir por la boquilla de escape, produciendo empuje. Simultáneamente, el aire bypass también se acelera y se mezcla con los gases de combustión en la salida, contribuyendo al empuje total del motor.

2.2 Importancia de la predicción y diagnóstico en la industria aeronáutica.

La predicción y el diagnóstico en la industria aeronáutica son de vital importancia debido a múltiples factores críticos que afectan la seguridad, la eficiencia

operativa, y la rentabilidad de las operaciones. A continuación, se detallan las razones principales que subrayan la importancia de estas actividades en el sector aeronáutico.

2.2.1 Seguridad.

- Prevención de Fallos Catastróficos: La capacidad de predecir fallos potenciales y diagnosticar problemas en tiempo real es crucial para prevenir accidentes y asegurar la seguridad de los pasajeros y la tripulación. Las herramientas de predicción y diagnóstico pueden identificar problemas antes de que ocurran, permitiendo acciones preventivas (Manrique, 2023).

- Mantenimiento Predictivo: A través del análisis predictivo, las aerolíneas pueden realizar mantenimiento predictivo, reparando o reemplazando componentes antes de que fallen. Esto reduce la probabilidad de fallos en vuelo y mejora la confiabilidad de las aeronaves (González, 2018).

2.2.2 Eficiencia Operativa.

- Reducción de Tiempos de Inactividad: La predicción y el diagnóstico eficientes permiten programar el mantenimiento de manera proactiva, minimizando los tiempos de inactividad no planificados. Esto asegura que las aeronaves estén en operación más tiempo, mejorando la utilización de la flota (Berges, 2020).

- Optimización de Recursos: Con un diagnóstico preciso, las aerolíneas pueden optimizar la asignación de recursos de mantenimiento, asegurando que los técnicos y las partes necesarias estén disponibles cuando y donde se necesiten (Papakostas, Papachatzakis, Xanthakis, Mourtzis, & Chryssolouris, 2010).

2.2.3 Costos.

- Ahorro en Costos de Mantenimiento: El mantenimiento predictivo y el diagnóstico temprano permiten evitar reparaciones costosas y reducir los costos asociados con fallos inesperados. Esto incluye tanto los costos directos de reparación como los indirectos, como la reprogramación de vuelos y la compensación a los pasajeros.

- Extensión de la Vida Útil de los Componentes: Al monitorear y mantener los componentes de manera más eficiente, las aerolíneas pueden extender la vida útil de estos, aplazando la necesidad de reemplazos costosos.

2.2.4. Competitividad y Satisfacción del Cliente.

- Mejora de la Confiabilidad del Servicio: Una mayor confiabilidad y menos retrasos y cancelaciones mejoran la percepción de la aerolínea por parte de los clientes, aumentando la satisfacción y la lealtad.

- Ventaja Competitiva: Las aerolíneas que implementan sistemas avanzados de predicción y diagnóstico pueden diferenciarse de sus competidores ofreciendo un servicio más seguro y confiable.

2.2.5. Cumplimiento Normativo.

- Adherencia a las Regulaciones: La industria aeronáutica está altamente regulada, y el mantenimiento predictivo ayuda a las aerolíneas a cumplir con los estrictos requisitos de seguridad y mantenimiento impuestos por organismos reguladores como la FAA (Federal Aviation Administration) y la EASA (European Union Aviation Safety Agency).

- Transparencia y Trazabilidad: Los sistemas de predicción y diagnóstico generan registros detallados de todas las actividades de mantenimiento y las condiciones de los componentes, lo cual es vital para auditorías y revisiones regulatorias.

2.2.6. Innovación y Desarrollo Tecnológico.

- Fomento de la Innovación: El desarrollo e implementación de tecnologías de predicción y diagnóstico impulsan la innovación en la industria. Esto incluye el uso de Big Data, Machine Learning, y IoT (Internet de las Cosas), que no solo mejoran las operaciones actuales, sino que también abren nuevas oportunidades para futuras mejoras.

- Adopción de Tecnologías Avanzadas: El uso de análisis de datos avanzados y tecnologías emergentes como la inteligencia artificial permite a las aerolíneas mantenerse a la vanguardia de la tecnología, facilitando mejoras continuas en la seguridad y la eficiencia.

2.3 Modelos de Predicción y Diagnóstico.

La selección de características es un paso crítico en el desarrollo de modelos de predicción y diagnóstico, especialmente en contextos complejos como la industria aeronáutica. La selección adecuada de características puede mejorar la precisión del modelo, reducir el tiempo de entrenamiento y facilitar la interpretación del modelo. Aquí se presentan métodos y técnicas comunes para la selección de características relevantes, específicamente en el contexto de la predicción y diagnóstico de motores tipo turbofan en la industria aeronáutica (Fu & Avdelidis, 2023).

2.4 Importancia de la Selección de Características.

1. **Mejora del Rendimiento del Modelo**: Reducir la dimensionalidad del espacio de características puede mejorar significativamente el rendimiento del modelo de Machine Learning, evitando el sobreajuste y reduciendo el ruido en los datos.
2. **Reducción de la Complejidad Computacional**: Menos características significan modelos más simples y rápidos de entrenar, lo cual es crucial en aplicaciones en tiempo real como el diagnóstico de motores.

3. **Interpretabilidad**: Modelos con menos características son más fáciles de interpretar, lo cual es vital en un entorno crítico como la industria aeronáutica.

2. 5 Técnicas de Selección de Características

2.5.1. Métodos Basados en Filtros (Filter Methods)

Estos métodos seleccionan características basándose en estadísticas que miden la relación entre cada característica independiente y la variable objetivo. Algunos de los métodos más comunes incluyen:

- **Chi-Cuadrado (Chi-Square)**: Evalúa la independencia entre las características categóricas y la variable objetivo.
- **Coeficiente de Correlación de Pearson**: Mide la correlación lineal entre características numéricas y la variable objetivo.
- **Análisis de Varianza (ANOVA)**: Evalúa si hay diferencias significativas entre los grupos de datos categóricos.

2.5.2. Métodos Basados en Envoltura (Wrapper Methods)

Estos métodos evalúan las características basándose en la calidad de los modelos que utilizan diferentes subconjuntos de características.

- **Método de Selección hacia Adelante (Forward Selection)**: Comienza con un modelo vacío y agrega características una por una, evaluando el rendimiento del modelo en cada paso.
- **Método de Selección hacia Atrás (Backward Elimination)**: Comienza con todas las características y elimina una a una las que tienen menor impacto.
- **Método de Selección Recursiva (Recursive Feature Elimination, RFE)**: Entrena el modelo y elimina recursivamente las características menos importantes.

2.5.3. Métodos Basados en el Modelo (Embedded Methods)

Estos métodos realizan la selección de características durante el proceso de entrenamiento del modelo.

- **Regresión Lasso (Least Absolute Shrinkage and Selection Operator)**: Introduce una penalización que puede hacer que los coeficientes de algunas características sean exactamente cero, eliminándolas efectivamente.
- **Árboles de Decisión y Random Forests**: Proveen una medida de importancia de las características basada en el criterio de partición (Gini o entropía).

2.6. Selección de Características en Motores Tipo Turbofan

Variables Comunes Utilizadas

En el contexto de la predicción y diagnóstico de motores tipo turbofan, las características pueden incluir variables relacionadas con el rendimiento del motor, condiciones de operación y variables ambientales. Algunas características comunes son:

- **Sensores de Temperatura y Presión**: Datos de sensores que monitorean la temperatura y presión en diferentes partes del motor.
- **Vibraciones**: Medidas de vibración que pueden indicar problemas mecánicos.
- **Flujos de Combustible y Aire**: Cantidades de combustible y aire que ingresan al motor.
- **RPM (Revoluciones por Minuto)**: Velocidad de rotación de diferentes componentes del motor.
- **Condiciones Ambientales**: Temperatura, presión y humedad del entorno operativo.

3. Identificación del objeto de estudio

En la industria aeronáutica, los motores más usados a nivel de vuelos comerciales son los motores tipo turbofan, estos al ser el corazón de un avión,

el dispositivo que le da el impulso al mismo, merece estar en perfectas condiciones para su funcionamiento rutinario. Se calcula que el 90% de aviones militares y comerciales utilizan este tipo de motores.

Sin embargo, estos motores están sujetos a un alto desgaste debido a su naturaleza de uso, largas horas de vuelo, bajas de temperatura, extensas horas de trabajo, etc., adicional a ello factores externos a las mismas turbinas que podrían ocasionar un fallo. Un fallo en la industria aeronáutica significaría pérdidas millonarias y sobre todo la vida de seres humanos.

Es por ello que mediante técnicas de mantenimiento predictivo sujetas a técnicas de Big Data, cada vez más, la industria aeronáutica ha desarrollado modelos que permitan predecir el fallo prematuro en piezas importantes, con el fin de ser reemplazadas anticipadamente.

Por eso, este trabajo de investigación se centrará en el estudio de motores turbofans, donde se investigará las características técnicas, comportamiento de funcionamiento, desgaste de estos motores y en base a la base de datos de las mediciones de diferentes sensores de los mimos establecer su comportamiento y modos de fallo

4. Planteamiento del problema

El reemplazo de un motor tipo turbofan es costoso, por temas tiempos como recursos económicos. El mantenimiento preventivo, predictivo y basado en la confiabilidad son las estrategias predominantes en la industria aeronáutica. Por otro lado, el departamento de mantenimiento de las aerolíneas o prestados externos de servicios está compuesto por equipos multidisciplinarios (ingenieros eléctricos, electrónicos, mecánicos, aeronáuticos, etc) que intervienen desde la creación de los planes de mantenimiento, la programación de los mismos, así también como en la ejecución y puesta en marcha de las tareas de mantenimiento como tal. Sin mencionar, que los overhalls programados se los planifican de acuerdo a los mantenimientos de los sistemas y subsistemas adicionales comprenden y son parte del avión

Mientras los motores están funcionando, se realiza como parte de las tareas de vuelo el monitoreo y evaluación de las variables y mediciones que intervienen y son partes del funcionamiento del mismo, así como los valores de todos los sensores que monitorean el funcionamiento de los turbofans. En este sentido, se tienen alarmas en caso de fallo y tipos de fallo, los cuales pueden ser tolerables y que no afectarían inmediatamente el funcionamiento del motor como tal, sino que pueden dar indicios de un comportamiento diferente o anómalo y que debe ser considerado una vez en tierra.

5. Objetivos.

El presente trabajo de titulación se centrará en predecir la vida útil restante (RUL) de cada motor en el conjunto de datos de prueba. RUL es equivalente a la cantidad de vuelos restantes para el motor después del último punto de datos en el conjunto de datos de prueba. Con el objetivo de

5.1 Objetivo general.

- Determinar la vida útil restante (RUL) de cada motor en el conjunto de datos de prueba, con el fin de mejorar la eficiencia operativa y sus horas de trabajo, mediante técnicas de machine learning para motores tipo turbofan

5.2 Objetivo específico.

- Analizar mediante matriz de correlación la importancia de ciertos sensores para descartar algunos de ellos que no son predictores.
- Realizar un análisis comparativo de los resultados arrojados por el análisis mediante RCME, y los resultados obtenidos mediante random forest-
- Determinar el número de ciclos operativos restantes antes del fallo en el conjunto de prueba, es decir, el número de ciclos operativos después del último ciclo que el motor seguirá funcionando.

6. Justificación y aplicación de la Metodología a utilizar

6.1 Recolección de datos.

La obtención de los datos principalmente viene de fuentes primarias, estos datos fueron generados utilizando el simulador de motores C-MAPSS (Commercial

Modular Aero-Propulsion System Simulation) desarrollado por la NASA. Aquí se detalla la procedencia de los mismos:

1. Simulador C-MAPSS:

Este simulador fue diseñado para modelar la degradación y fallo de motores tipo turbofan en condiciones operacionales variables. Permite investigar y desarrollar métodos para el monitoreo y mantenimiento predictivo de motores aeronáuticos.

2. Contenido de los Datos:

- Condiciones de Vuelo: Los datos incluyen una variedad de condiciones de vuelo y modos de operación.

- Sensores de Motor: La base de datos contiene lecturas de múltiples sensores que monitorean parámetros críticos del motor, como temperatura, presión, flujo de aire y vibraciones.

- Ciclos de Vuelo: Cada registro en la base de datos corresponde a un ciclo de vuelo completo, proporcionando una secuencia temporal de datos que reflejan la degradación progresiva del motor.

6.2 Identificación y descripción de variables.

Los datos a ser utilizados son de tipo simulados y reales, los datos reales son mediciones temporales.

Datos Simulados: Todos los datos de sensores y configuraciones son simulados y representan mediciones en diferentes condiciones de operación y ciclos de vida de los motores.

Mediciones Temporales: Los datos son secuenciales y reflejan el comportamiento del motor a lo largo de varios ciclos de vuelo, lo que permite analizar la degradación progresiva del motor.

En las diferentes columnas que comprenden la base de datos el atributo "*unit nr*", que hace referencia al motor al que corresponden los datos. Como podemos observar en esta base de datos tenemos datos pertenecientes a 100 motores similares. El atributo "*time cycles*" hace referencia al número de ciclos o número de vuelos que ha desarrollado cada motor. Las columnas correspondientes a setting 1, setting 2, setting 3 hacen referencia al modo de operación de los motores. Sin embargo, en el desarrollo de la presente investigación estos

atributos no son útiles por lo que posteriormente serán eliminados. Las columnas s 1, ..., s 21 son las mediciones realizadas por los diferentes sensores.

Los datos de los 21 sensores generalmente no vienen con unidades de medida explícitas en la documentación disponible. Sin embargo, en el contexto de monitoreo de motores y mantenimiento predictivo, estos sensores suelen medir diversas variables físicas, cada una con su propia unidad de medida. En la siguiente table se detalla cuáles son las dimensiones de cada sensor y que representa cada uno de ellos (ver Tabla 1).

Variable	**Dependiente/ Independiente**	**Fuente de Datos**	**Tipo de Datos**	**Descripción**
unit_number	Independiente	Simulación C-MAPSS	Categórico	Identificador único para cada motor (unidad).
time_in_cycles	Dependiente	Simulación C-MAPSS	Numérico (Entero)	Ciclo de vuelo actual del motor, representa la vida útil restante del motor hasta el fallo.
setting_1	Independiente	Simulación C-MAPSS	Numérico (Flotante)	Parámetro de configuración del motor, ajustado durante la operación.
setting_2	Independiente	Simulación C-MAPSS	Numérico (Flotante)	Parámetro de configuración del motor, ajustado durante la operación.
setting_3	Independiente	Simulación C-MAPSS	Numérico (Flotante)	Parámetro de configuración del motor, ajustado durante la operación.
sensor_measurement_1	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Total temperature at fan inlet: Grados Celsius (°C)

sensor_measurement_2	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Total temperature at LPC outlet: Grados Celsius (°C)
sensor_measurement_3	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Total temperature at HPC outlet: Grados Celsius (°C)
sensor_measurement_4	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Pressure at fan inlet: Libras por pulgada cuadrada (psi)
sensor_measurement_5	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Pressure in bypass-duct: Libras por pulgada cuadrada (psi)
sensor_measurement_6	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Pressure at HPC outlet: Libras por pulgada cuadrada (psi)
sensor_measurement_7	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Physical fan speed: RPM (revoluciones por minuto)
sensor_measurement_8	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Physical core speed: RPM (revoluciones por minuto)
sensor_measurement_9	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Engine pressure ratio (P50/P2): Adimensional
sensor_measurement_10	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Static pressure at HPC outlet: Libras por pulgada cuadrada (psi)
sensor_measurement_11	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Ratio of fuel flow to Ps30: Adimensional
sensor_measurement_12	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Corrected fan speed: RPM (revoluciones por minuto)

sensor_measurement_13	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Corrected core speed: RPM (revoluciones por minuto)
sensor_measurement_14	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Bypass ratio: Adimensional
sensor_measurement_15	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Burner fuel-air ratio: Adimensional
sensor_measurement_16	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Bleed enthalpy: Adimensional
sensor_measurement_17	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Demanded fan speed: RPM (revoluciones por minuto)
sensor_measurement_18	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Demanded corrected fan speed: RPM (revoluciones por minuto).
sensor_measurement_19	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	HPT coolant bleed: Adimensional
sensor_measurement_20	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	LPT coolant bleed: Adimensional
sensor_measurement_21	Independiente	Sensores del motor (Simulado)	Numérico (Flotante)	Burner pressure: Libras por pulgada cuadrada (psi)

Tabla 1: agrupa los diferentes tipos de variables, con sus unidades, nombres, tipos de dato

6.3 Visualización de variables

El conjunto de datos del turbofan presenta cuatro conjuntos de datos de complejidad creciente (ver tabla 2) (NASA, 2020). Los motores funcionan

normalmente al principio, pero desarrollan fallas con el tiempo. Para los conjuntos de entrenamiento, los motores funcionan hasta que fallan, mientras que en los conjuntos de prueba la serie temporal finaliza "en algún momento" antes de la falla. El objetivo es predecir la vida útil restante (RUL) de cada motor turbofan.

Dataset	Condiciones de operación	Modo de falla	Número de motores (paquete de entrenamiento)	Número de motores (paquete de de testeo)
FD001	1	1	100	100
FD002	6	1	260	359
FD003	1	2	100	100
FD004	6	2	248	249

Tabla 2: Revisión de la composición del dataset

Los conjuntos de datos incluyen simulaciones de múltiples motores turbofan a lo largo del tiempo; cada fila contiene la siguiente información:

1. Número de unidad del motor

2. El tiempo, en ciclos

3. Tres configuraciones operativas

4. 21 lecturas de sensores

A continuación, en la siguiente tabla, se representa los datos estadísticos descriptivos más importantes de las variables que componen la base de datos.

	Unit_nr	time_cycles
count	20631	100
mean	51.51	206.31
std	29.23	46.34
min	1	128
25%	26	177
50%	52	199

75%	77	229.25
max	100	362

Tabla 3: Datos estadísticos descriptivos de la base de datos

Cuando se realiza un análisis simple de las estadísticas descriptivas de unit_nr, se puede apreciar que el conjunto de datos tiene un total de 20631 filas, los números de unidad comienzan en 1 y terminan en 100. Tanto la media y los cuantiles no se alinean con las estadísticas descriptivas, esto es debido a que cada unidad tiene diferentes ciclos de tiempo máximos y, por lo tanto, un número diferente de filas. Al inspeccionar el tiempo máximo de ciclos, puede ver que el motor que falló primero lo hizo después de 128 ciclos, mientras que el motor que funcionó durante más tiempo se averió después de 362 ciclos. El motor medio frena entre 199 y 206 ciclos, sin embargo, la desviación estándar de 46 ciclos es bastante grande.

La descripción del conjunto de datos también indica que los turbofan funcionan en una única condición de funcionamiento. Comprobemos la configuración para la verificación.

Al revisar las estadísticas descriptivas de los datos del sensor, buscando indicadores de fluctuación de la señal (o ausencia de la misma).

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
s_1	20631	518.67	0.00	518.67	518.67	518.67	518.67	518.67
s_2	20631	642.68	0.50	641.21	642.33	642.64	643.00	644.53
s_3	20631	1590.52	6.13	1571.04	1586.26	1590.10	1594.38	1616.91
s_4	20631	1408.93	9.00	1382.25	1402.36	1408.04	1414.56	1441.49
s_5	20631	14.62	0.00	14.62	14.62	14.62	14.62	14.62
s_6	20631	21.61	0.00	21.60	21.61	21.61	21.61	21.61
s_7	20631	553.37	0.89	549.85	552.81	553.44	554.01	556.06
s_8	20631	2388.10	0.07	2387.90	2388.05	2388.09	2388.14	2388.56
s_9	20631	9065.24	22.08	9021.73	9053.10	9060.66	9069.42	9244.59

s_10	20631	1.30	0.00	1.30	1.30	1.30	1.30	1.30
s_11	20631	47.54	0.27	46.85	47.35	47.51	47.70	48.53
s_12	20631	521.41	0.74	518.69	520.96	521.48	521.95	523.38
s_13	20631	2388.10	0.07	2387.88	2388.04	2388.09	2388.14	2388.56
s_14	20631	8143.75	19.08	8099.94	8133.25	8140.54	8148.31	8293.72
s_15	20631	8.44	0.04	8.32	8.41	8.44	8.47	8.58
s_16	20631	0.03	0.00	0.03	0.03	0.03	0.03	0.03
s_17	20631	393.21	1.55	388.00	392.00	393.00	394.00	400.00
s_18	20631	2388.00	0.00	2388.00	2388.00	2388.00	2388.00	2388.00
s_19	20631	100.00	0.00	100.00	100.00	100.00	100.00	100.00
s_20	20631	38.82	0.18	38.14	38.70	38.83	38.95	39.43
s_21	20631	23.29	0.11	22.89	23.22	23.30	23.37	23.62

Tabla 4: estadísticas descriptivas de los datos del sensor

Al observar la desviación estándar, queda claro que los sensores 1, 10, 18 y 19 no fluctúan en absoluto; estos se pueden descartar con seguridad ya que no contienen información útil. La inspección de los cuantiles indica que los sensores 5, 6 y 16 tienen poca fluctuación y requieren una inspección adicional. Los sensores 9 y 14 tienen la mayor fluctuación, sin embargo, esto no significa que los otros sensores no puedan contener información valiosa.

Se creará un modelo de regresión lineal de referencia para poder comparar los resultados con los arrojados por el cálculo del RUL realizado mediante random forest.

6.4 Limpieza, pre-procesamiento y/o transformación de datos.

A continuación, se describe la metodología utilizando las bibliotecas pandas, numpy, y scikit-learn para llevar a cabo cada uno de los procesos de limpieza, pre procesamiento y transformación

En una primera fase de depuración se descartarán todos los sensores que tenga una desviación estándar cero “0” y posterior a ello en una segunda instancia se hará un análisis de importancia para descartar otros sensores que no aporten al cálculo del RUL.

6.5 Selección de modelo estadístico.

6.5.1 Modelo de regresión lineal.

Un modelo de regresión lineal busca establecer una relación lineal entre una variable dependiente (en este caso, el RUL - Remaining Useful Life) y una o más variables independientes (sensores y configuraciones del motor). El objetivo es predecir el RUL basado en los datos de los sensores.

Primero, definiremos una pequeña función para evaluar nuestros modelos. Elegí incluir el error cuadrático medio (RMSE), ya que dará una indicación de cuántos ciclos de tiempo están equivocadas las predicciones en promedio, y la varianza explicada (o puntuación R^2) para indicar qué proporción de nuestra variable dependiente puede ser explicada por la variable independiente. variables que utilizamos.

Eliminaremos unit_nr, time_cycle, configuraciones y sensores que no contienen información. La columna RUL del conjunto de entrenamiento se almacena en su propia variable. Para nuestro conjunto de prueba eliminamos las mismas columnas. Además, solo nos interesa el último ciclo de tiempo de cada motor en el conjunto de prueba, ya que solo tenemos valores True RUL para esos registros.

Configurar la regresión lineal es bastante sencillo. Creamos una instancia del modelo simplemente llamando al método y asignándolo a la variable 'lm'. A continuación, ajustamos el modelo pasando nuestro 'X_train' y 'y_train'. Finalmente, predecimos tanto en el tren como en el conjunto de prueba para obtener una imagen completa de cómo se comporta nuestro modelo con los datos que se le presentaron.

6.5.2 Bosque aleatorio

Una de las fortalezas clave del Random Forest (RF) o bosque aleatorio, sobre los árboles de decision, es su capacidad para generar árboles variados. Es decir, al crear un árbol de decisión único, el algoritmo intenta crear un nodo de decisión basado en una única característica (de todas las características disponibles) que

divide mejor su conjunto de datos (Breiman, 2001). Para el siguiente nodo, volverá a examinar todas las funciones disponibles para crear la siguiente mejor división. Si ajustara un árbol de decisión por segunda vez en estas condiciones, obtendría exactamente el mismo árbol (Ho, 1995). Sin embargo, un RF solo considera un subconjunto de todas las entidades al crear una división. Esto obliga al algoritmo a crear árboles diferentes, ya que es posible que la función para crear la mejor división no esté disponible. Potencialmente, se pueden crear combinaciones de divisiones que funcionen mejor que la mejor división de un árbol de decisión normal (Hastie & Friedman, 2001).

Al examinar la documentación de sklearns RandomForestRegressor se muestra que considera todas las características de forma predeterminada, esencialmente creando el mismo árbol una y otra vez. Por lo tanto, especificamos "*max_features*" como la raíz cuadrada de las funciones disponibles. Adicional, se configura "*random_state*" para que los árboles siempre se generen de la misma manera. De lo contrario, la generación aleatoria del árbol afectará los resultados del modelo, lo que dificultará juzgar si un modelo funciona mejor porque cambiamos algunas características o debido a la aleatoriedad (Criminisi, 2012).

Los bosques aleatorios sobresalen principalmente en el aprendizaje de patrones de datos complejos y no varian al escalamiento o transformaciones de características. Dado que todas las funciones ya son numéricas

El mayor desafío al adaptar Random Forests es el sobreajuste. Los parámetros max_ Depth, min_samples_leaf, ccp_alpha y min_impurity_decrease ayudan a reducir el sobreajuste y generar modelos generales con mejor rendimiento.

6.6 Análisis de la Matriz de Correlación

La matriz de correlación mostrará cómo se relacionan las variables entre sí:

Valores Cercanos a 1 o -1: Indican una fuerte correlación positiva o negativa, respectivamente.

Valores Cercanos a 0: Indican poca o ninguna correlación.

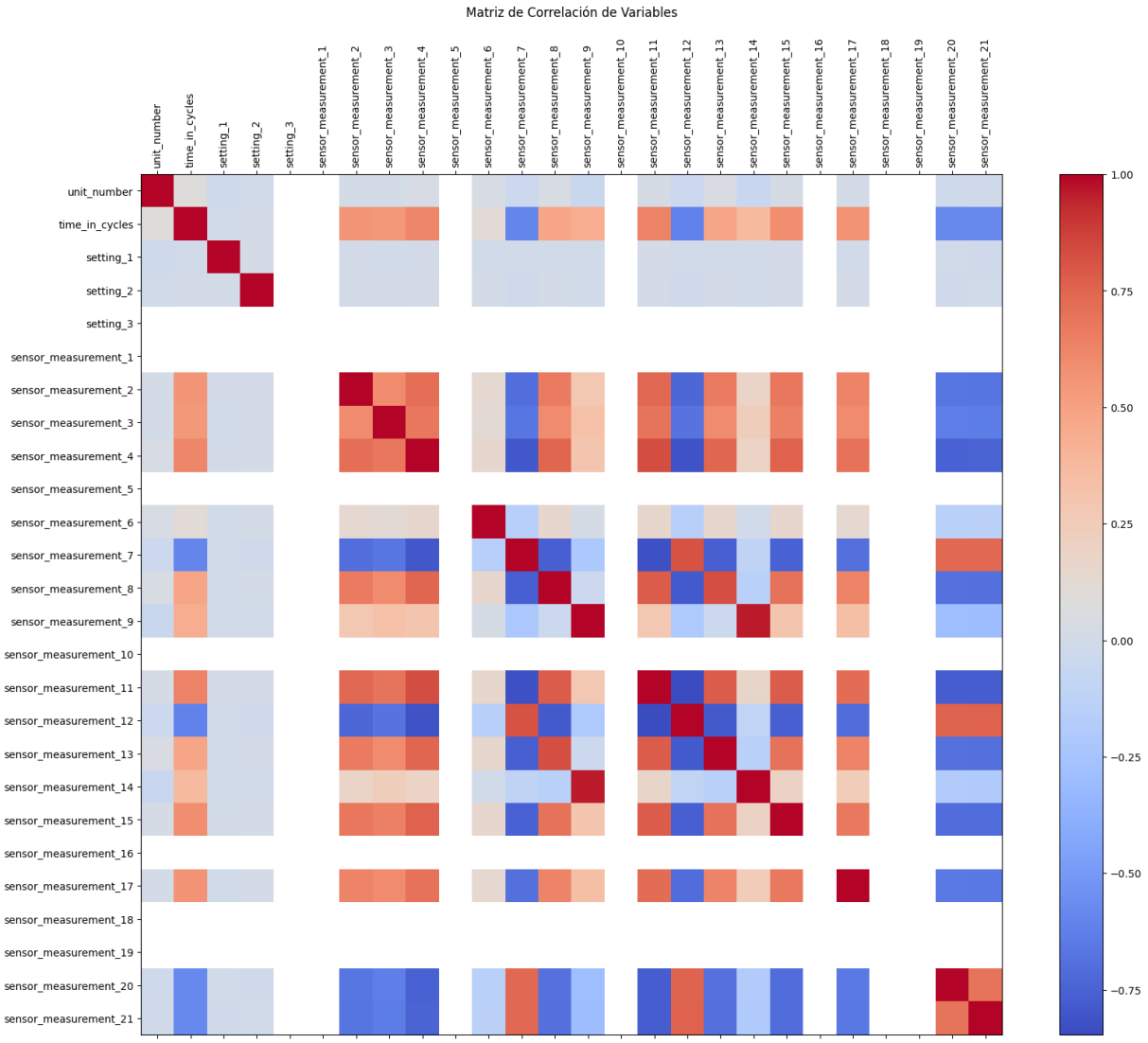

Ilustración 2: Cuadro de correlaciones

La matriz de correlación proporciona una visión cuantitativa de las relaciones entre las variables en el conjunto de datos "NASA C-MAPSS". Este análisis es un paso importante en la exploración de datos y puede ayudar a identificar qué variables pueden ser más útiles para el desarrollo de modelos predictivos.

7. Resultados.

7.1 Análisis de resultados.

Matemáticamente podemos usar max_time_cycle - time_cycle para calcular nuestro RUL deseado. Como queremos tener en cuenta el max_time_cycle de

cada motor, agruparemos el marco de datos por unit_nr antes de calcular max_time_cycle. Luego, max_time_cycle se fusiona nuevamente en el marco de datos para permitir un cálculo fácil de RUL restando las columnas max_time_cycle - time_cycle. Luego eliminamos max_time_cycle porque ya no es necesario e inspeccionamos las primeras filas para verificar nuestro cálculo de RUL.

Comencemos trazando el histograma de RUL máximo para comprender su distribución.

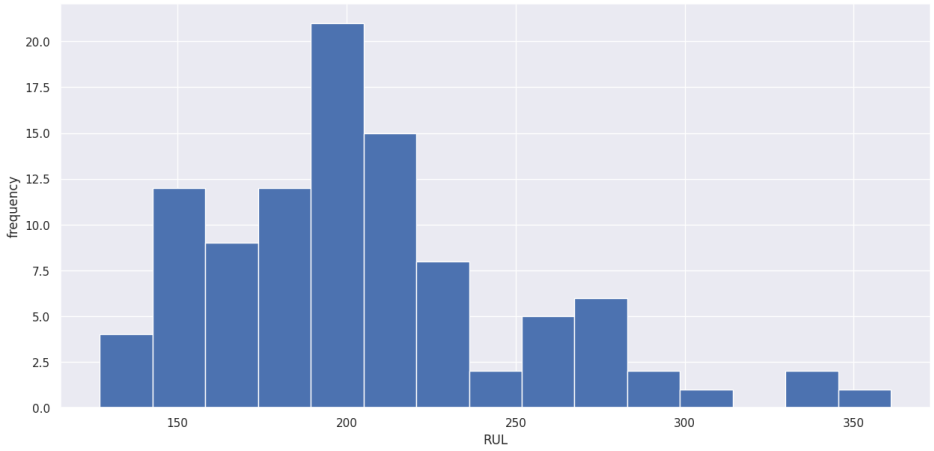

Ilustración 3: histograma de RUL

Se calculará una variable objetivo para la vida útil restante (RUL). La variable objetivo tendrá dos propósitos:

1. Servirá como nuestro eje X mientras trazamos las señales de los sensores, lo que nos permitirá interpretar fácilmente los cambios en las señales de los sensores cuando los motores estén a punto de averiarse.
2. Servirá como variable objetivo para nuestros modelos supervisados de aprendizaje automático.

El siguiente análisis se dará bajo propias estimaciones. Para ello se supondrá que el RUL disminuye con el tiempo y tiene un valor de 0 en el último ciclo del motor, esa disminución puede ser o no lineal, pero para efectos de este estudio se tomará en cuenta que es lineal.

Debido a la gran cantidad de motores, no es factible trazar cada motor para cada sensor. Los gráficos ya no serían interpretables con tantas líneas en un gráfico.

Por lo tanto, se eligió trazar cada motor cuyo unit_nr sea divisible por 10. Invertimos el eje X para que RUL disminuya a lo largo del eje, con un RUL de cero que indica falla del motor. Debido a la gran cantidad de sensores, comentaré algunos gráficos que son representativos de todo el conjunto. Recuerde, según nuestras estadísticas descriptivas, definitivamente debemos inspeccionar las gráficas de los sensores 5, 6 y 16.

El gráfico de los sensores 1, 10, 18 y 19 es similar, la línea plana indica que los sensores no contienen información útil, lo que reconfirma nuestra conclusión de las estadísticas descriptivas. Los sensores 5 y 16 también muestran una línea plana; estos se pueden agregar a la lista de sensores para excluir.

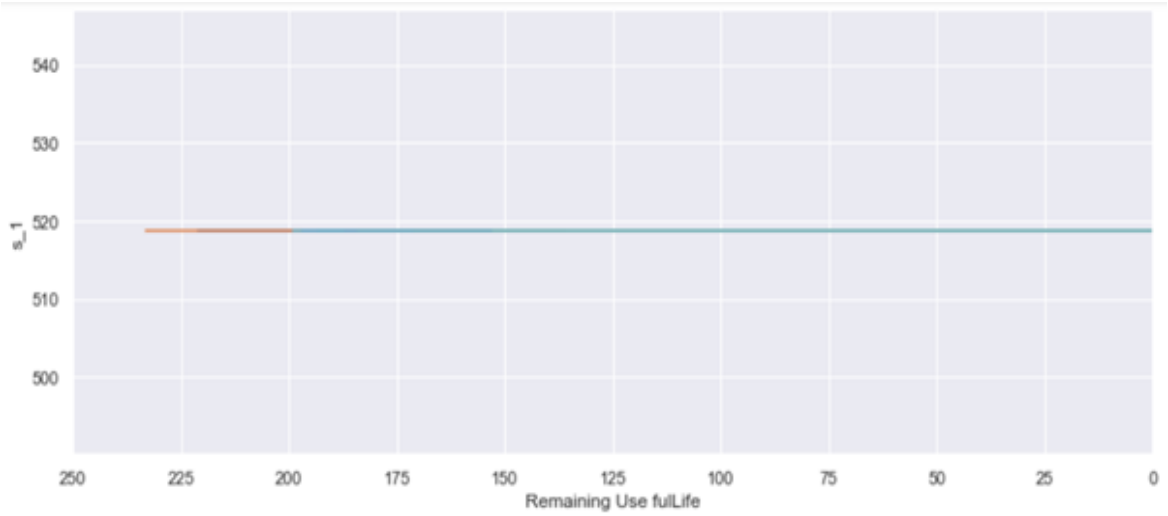

Ilustración 4: Sensor 1. Tota, temperature at fan inlet: Grados Celsius (°C) vs RUL

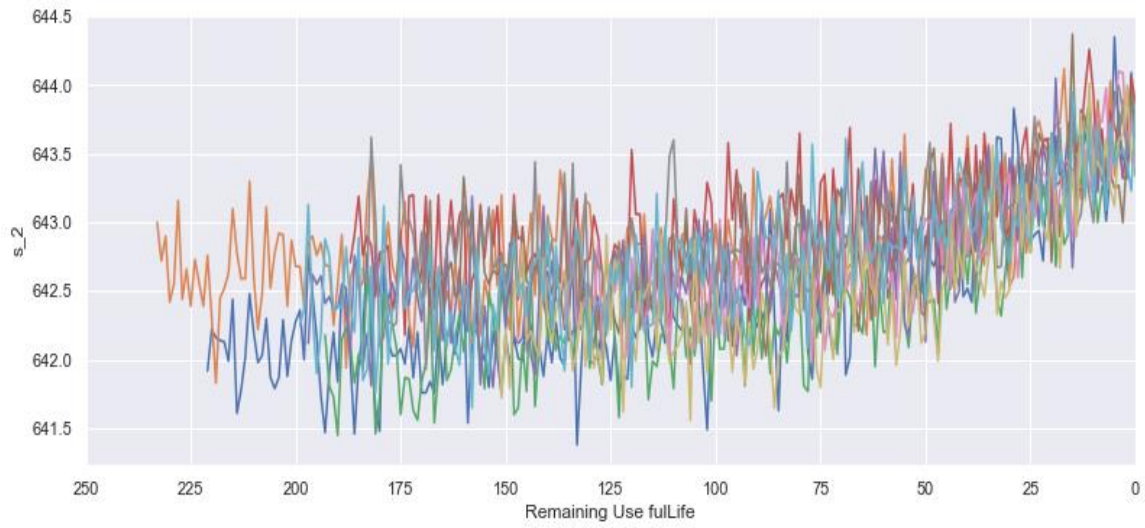

Ilustración 5: Sensor 2. Total temperature at LPC outlet: Grados Celsius (°C) vs RUL

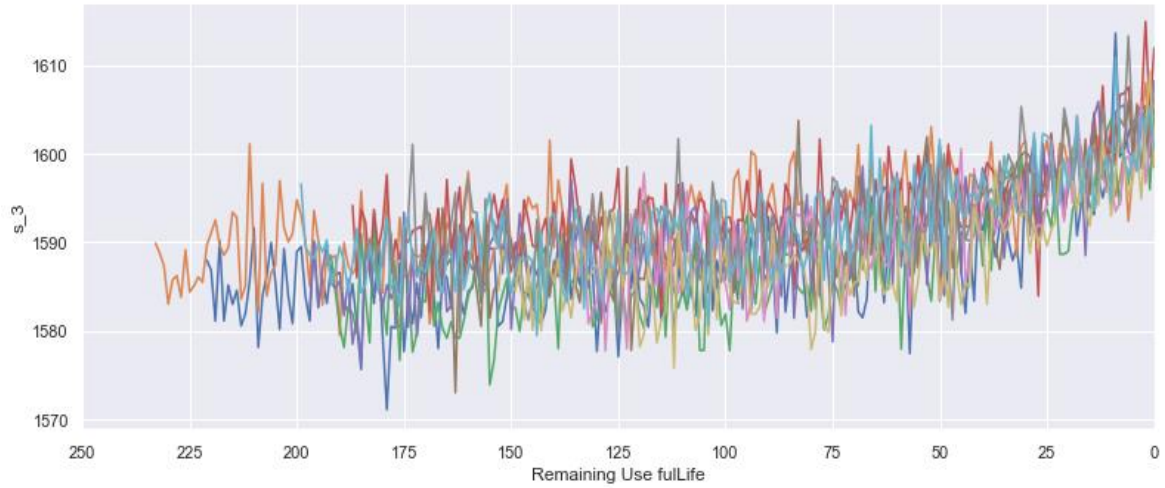

Ilustración 6: Sensor 3. Total temperature at HPC outlet: Grados Celsius (°C)vs RUL

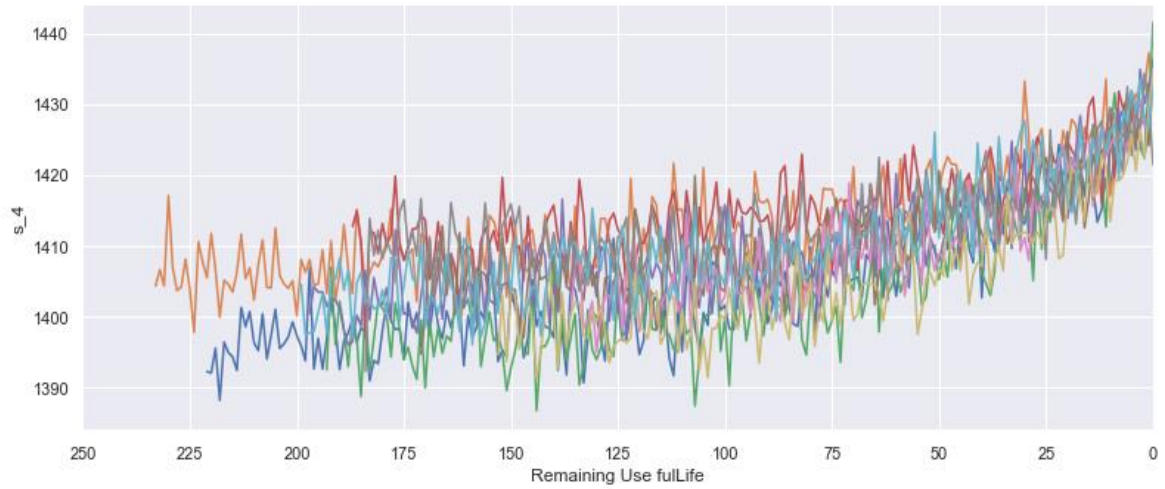

Ilustración 7: Sensor 4. Pressure at fan inlet: Libras por pulgada cuadrada (psi) vs RUL

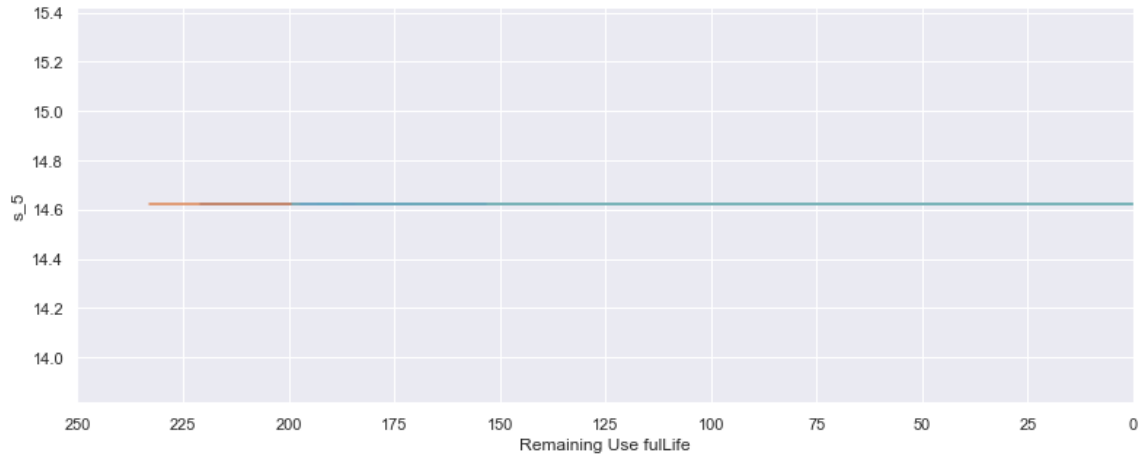

Ilustración 8: Sensor 5. Pressure in bypass-duct: Libras por pulgada cuadrada (psi) vs RULº

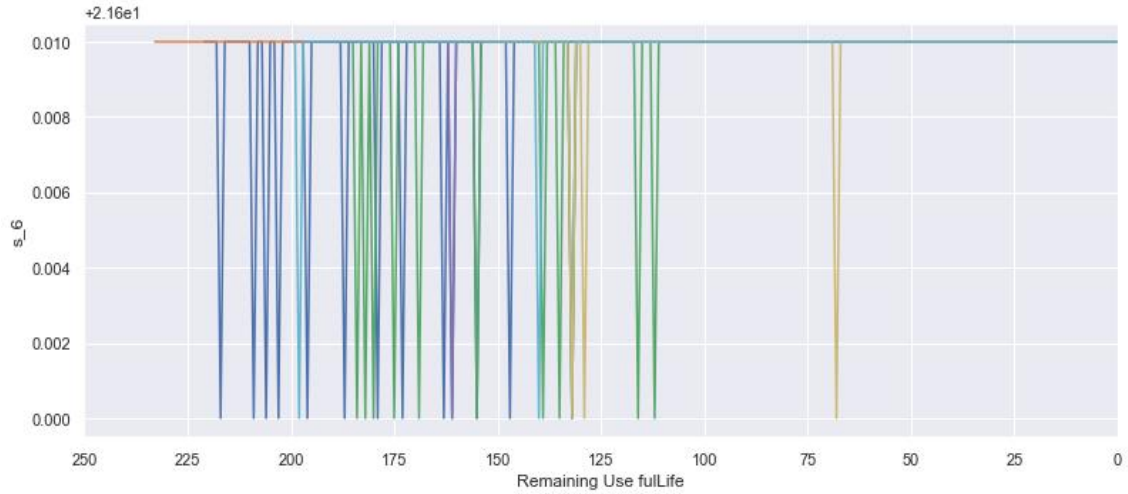

Ilustración 9: Sensor 6. Pressure at HPC outlet: Libras por pulgada cuadrada (psi) vs RUL

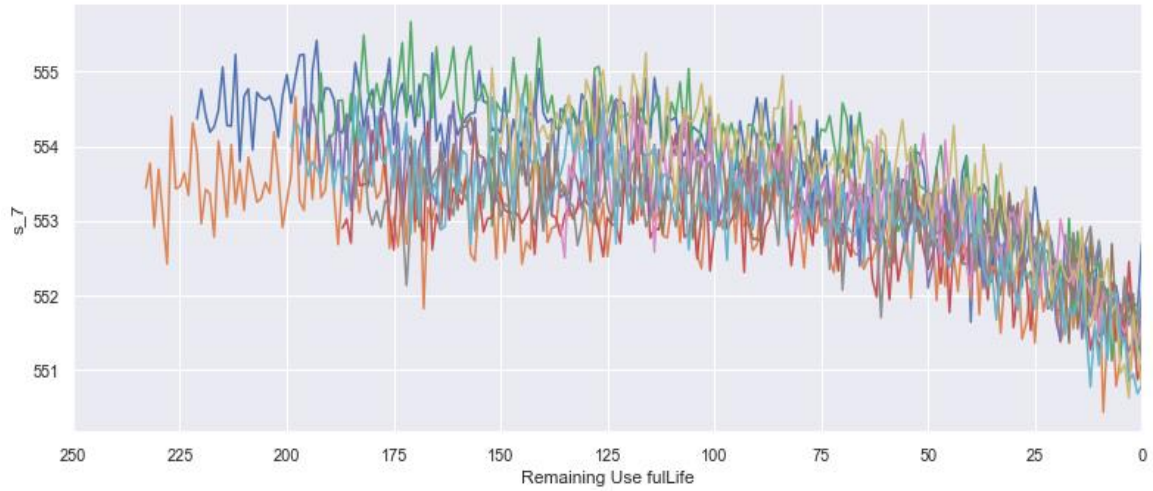

Ilustración 10: Sensor 7. Physical fan speed: RPM (revoluciones por minuto) vs RUL

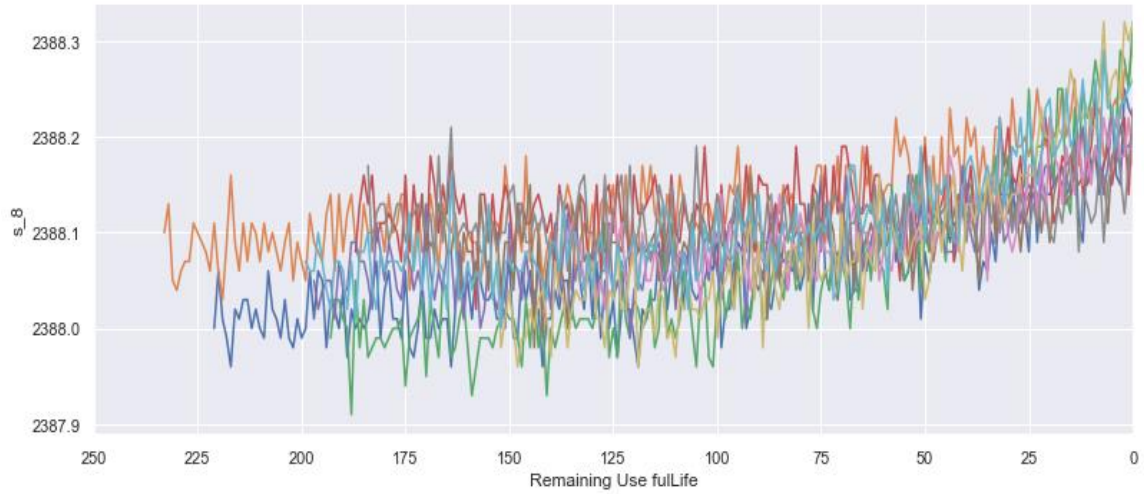

Ilustración 11: Sensor 8. Physical core speed: RPM (revoluciones por minuto) vs RUL

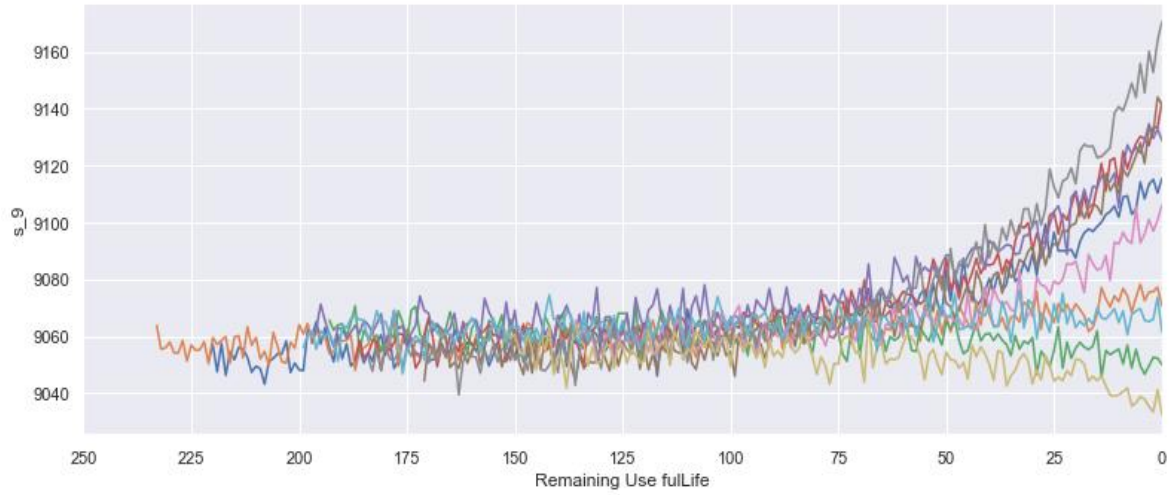

Ilustración 12: Sensor 9. Engine pressure ratio (P50/P2): Adimensional vs RUL

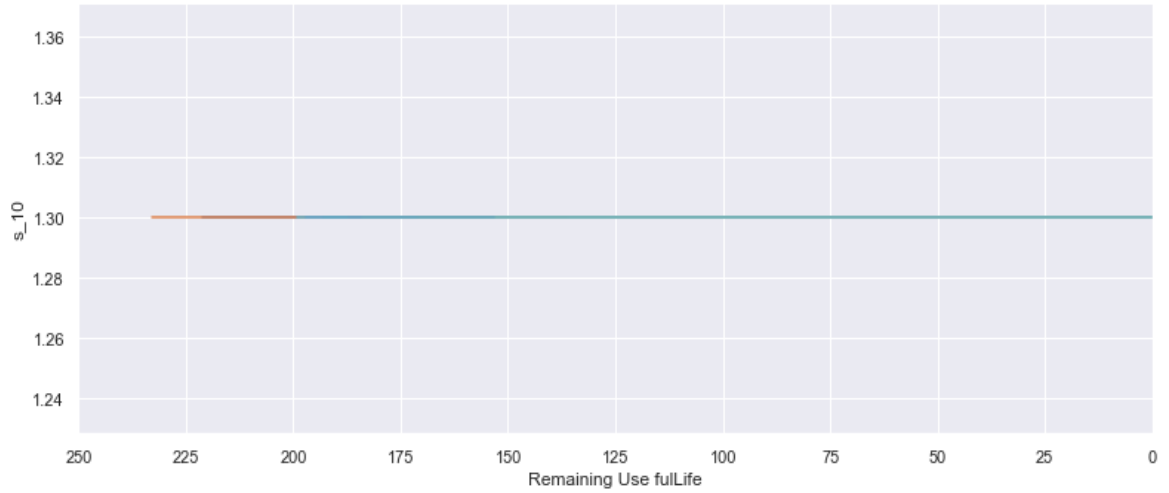

Ilustración 13: Sensor 10. Static pressure at HPC outlet: Libras por pulgada cuadrada (psi) vs RUL

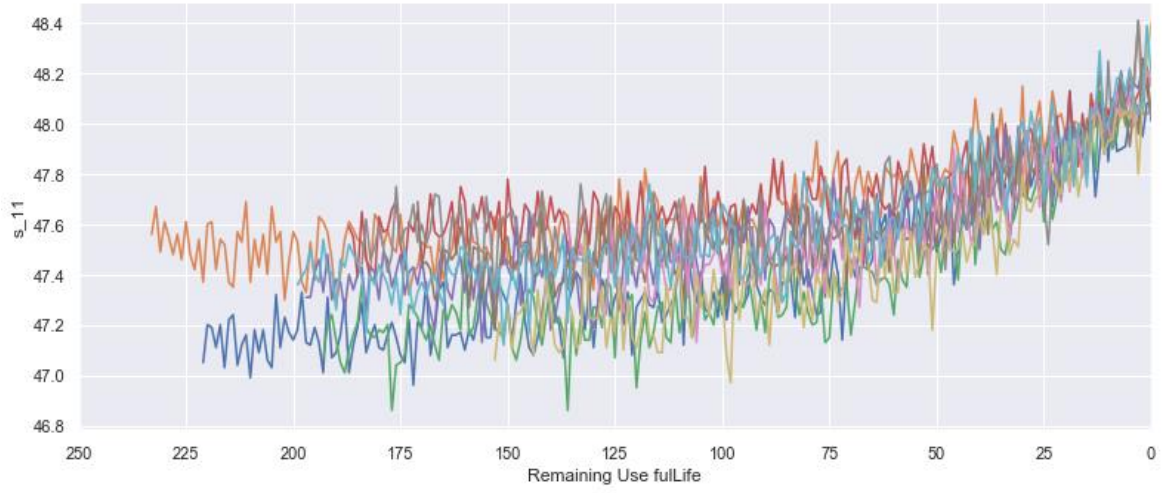

Ilustración 14: Sensor 11. Ratio of fuel flow to Ps30: Adimensional vs RUL

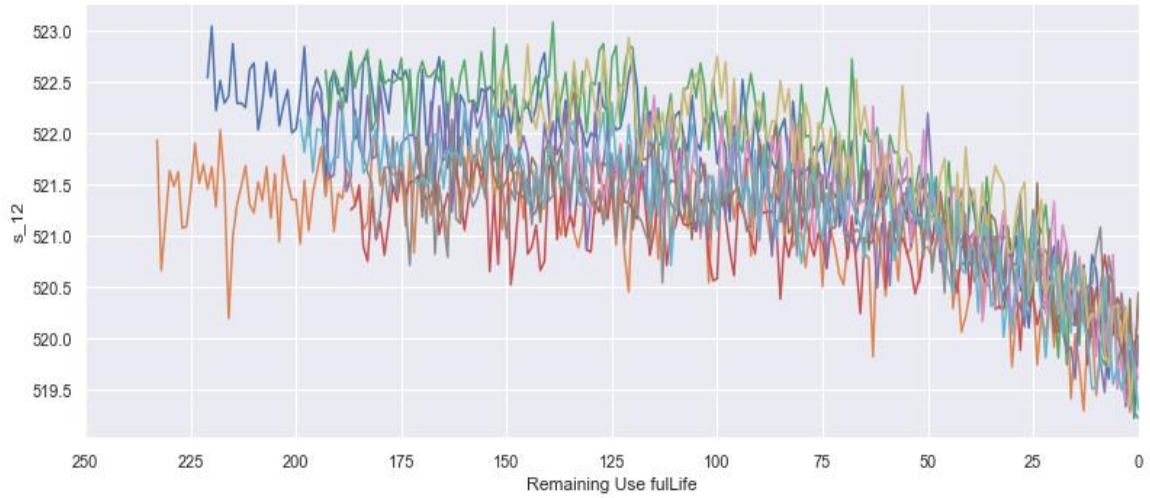

Ilustración 15: Sensor 12. Corrected fan speed: RPM (revoluciones por minuto) vs RUL

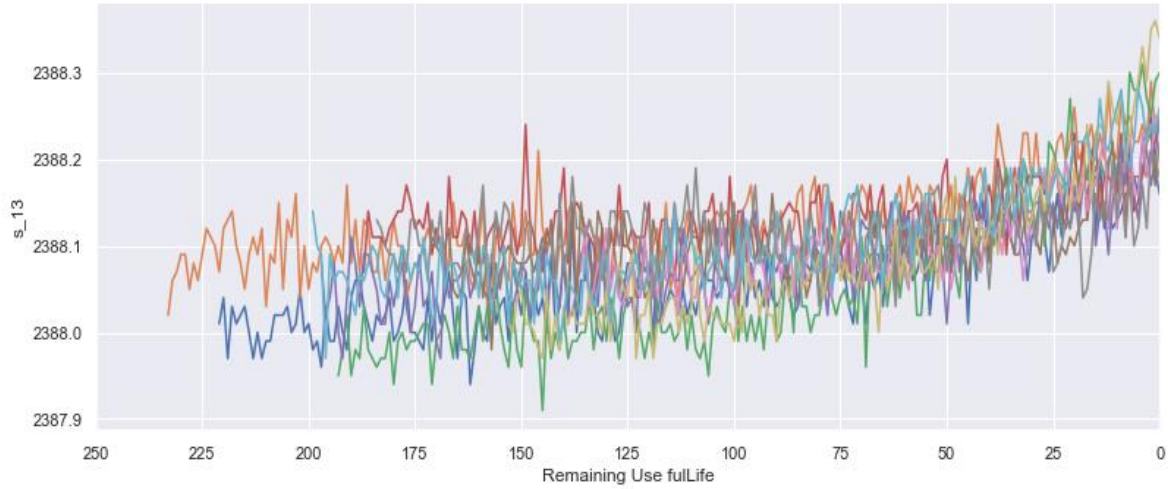

Ilustración 16: Sensor 13. Corrected core speed: RPM (revoluciones por minuto) vs RUL

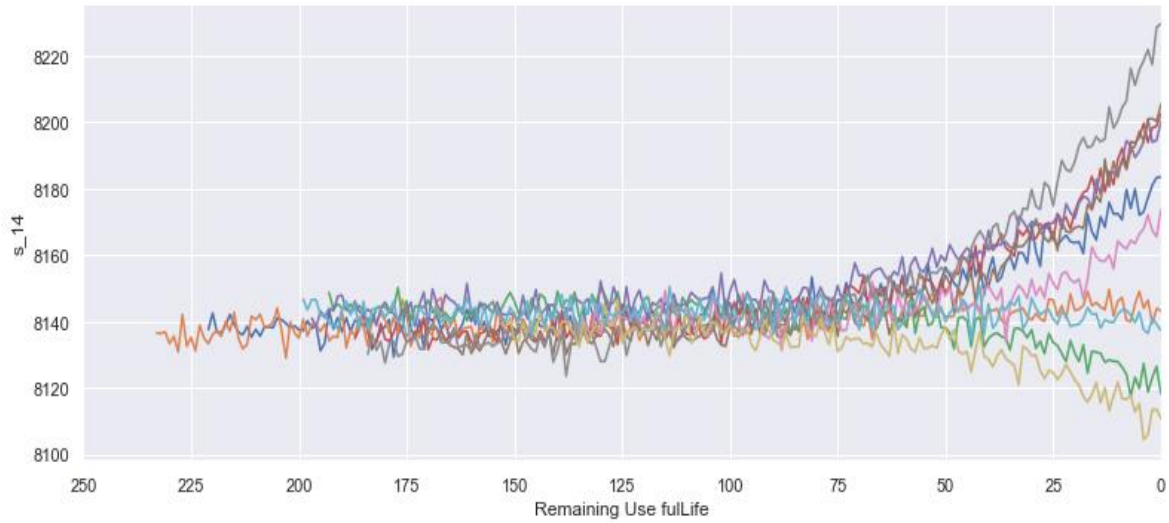

Ilustración 17: Sensor 14. Bypass ratio: Adimensional vs RUL

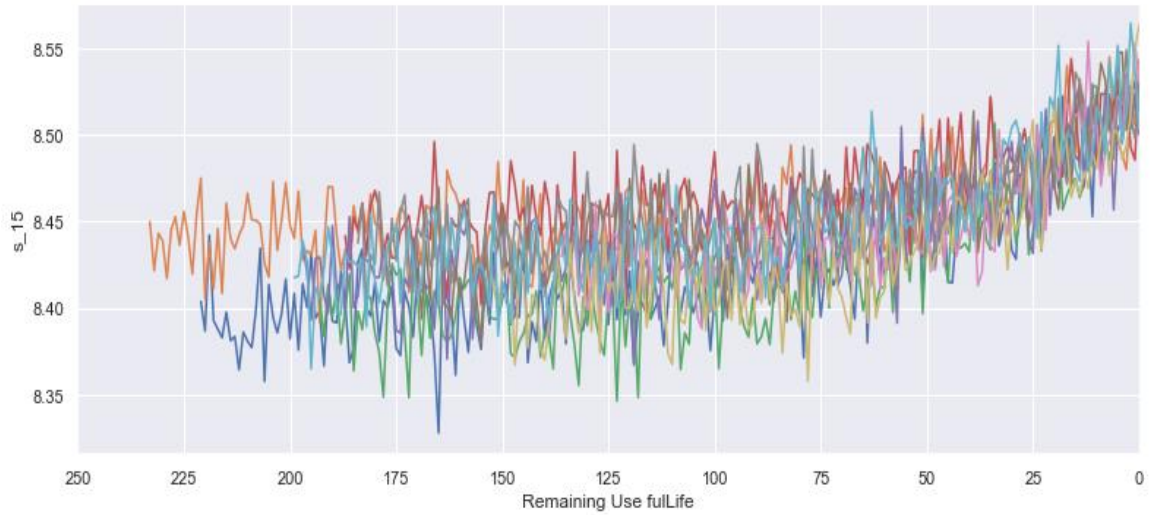

Ilustración 18: Sensor 15. Burner fuel-air ratio: Adimensional vs RUL

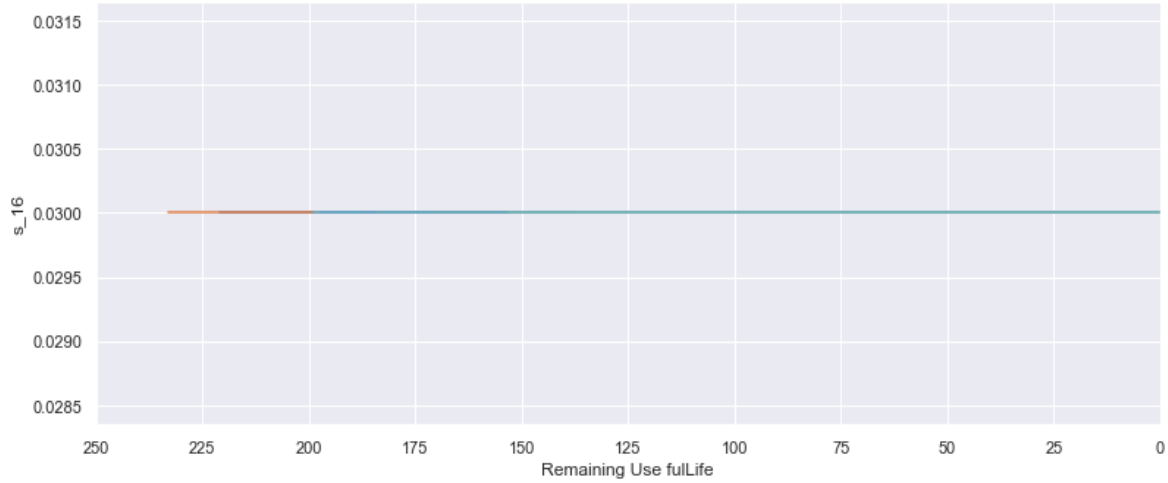

Ilustración 19: Sensor 16. Bleed enthalpy: Adimensional vs RUL

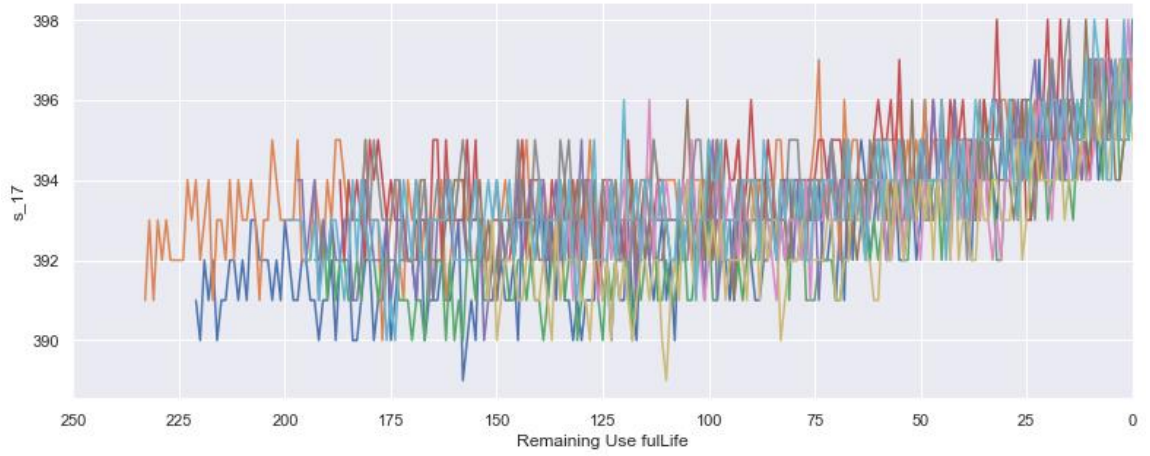

Ilustración 20: Sensor 17. Demanded fan speed: RPM (revoluciones por minuto) vs RUL

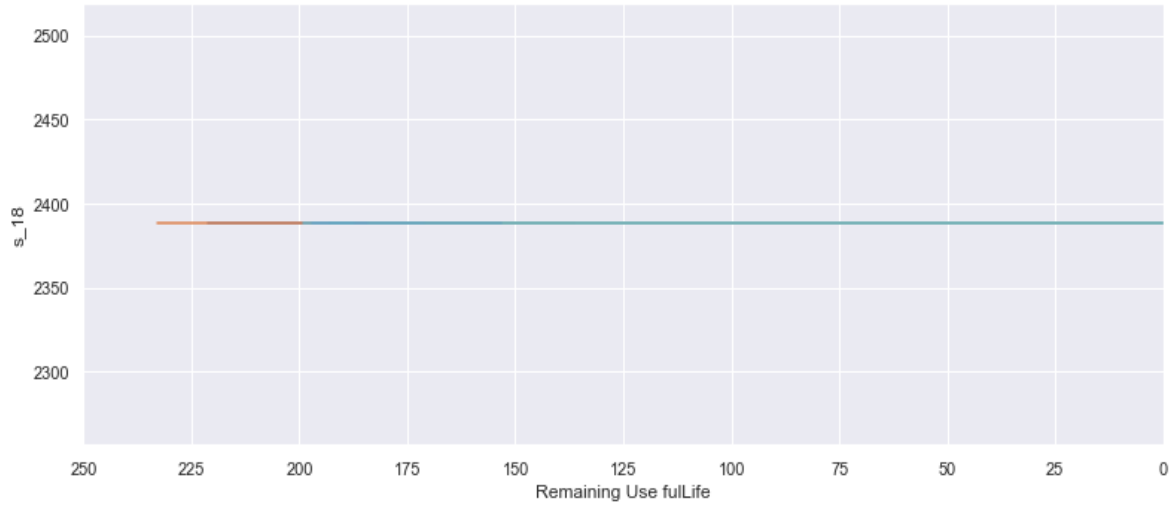

Ilustración 21: Sensor 18. Demanded corrected fan speed: RPM (revoluciones por minuto). Vs RUL

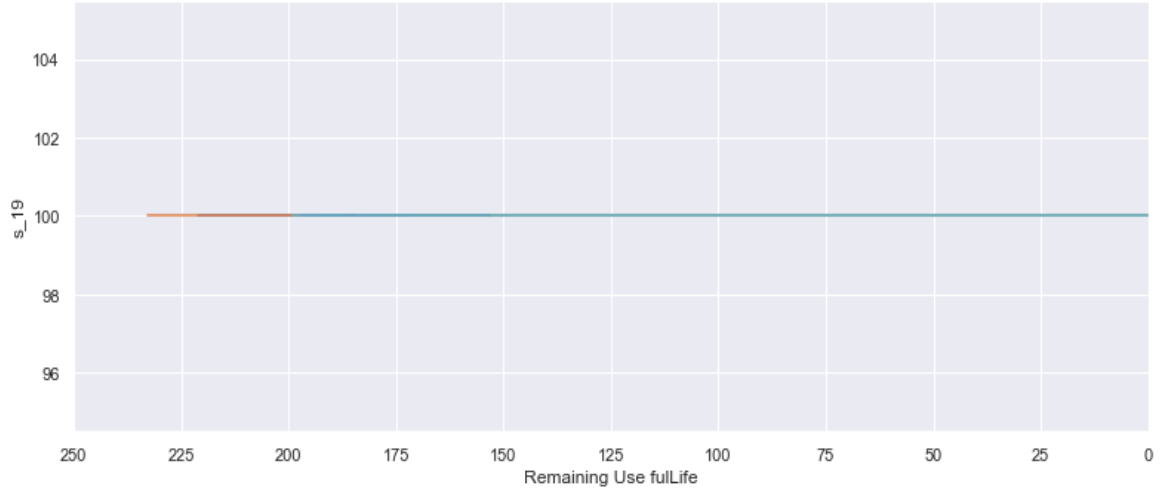

Ilustración 22: Sensor 19. HPT coolant bleed: Adimensional vs RUL

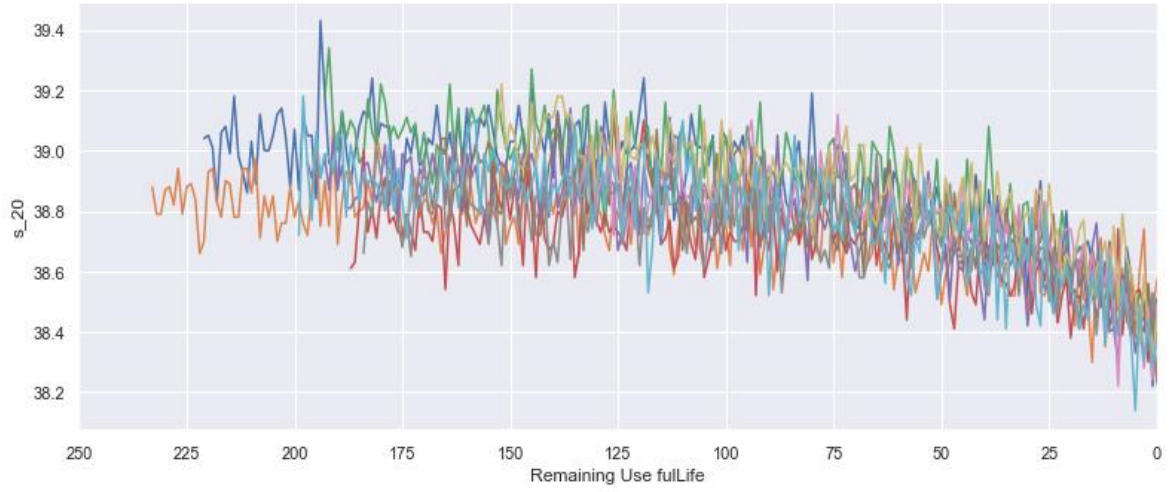

Ilustración 23: Sensor 20. LPT coolant bleed: Adimensional vs RUL

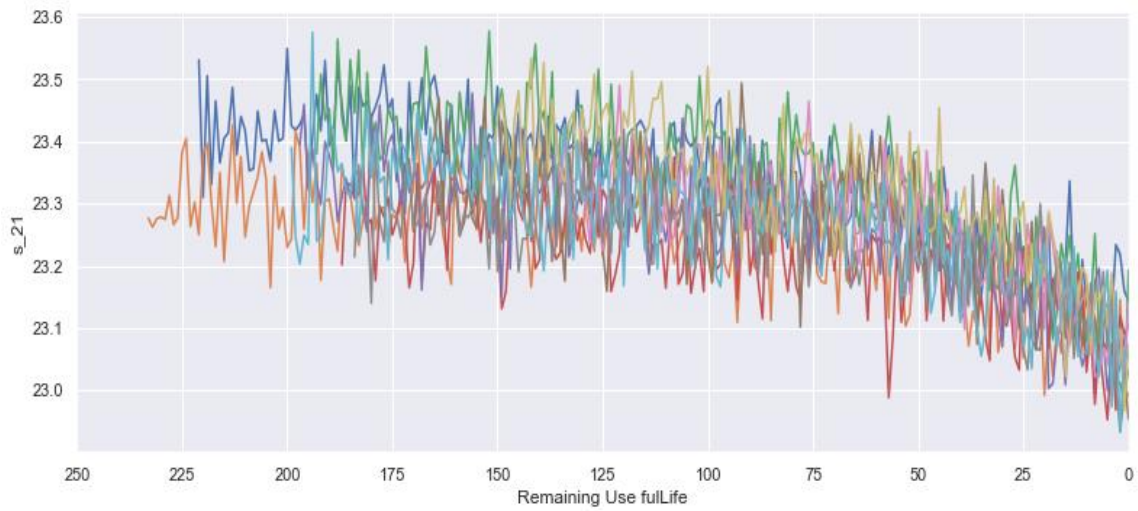

Ilustración 24: Sensor 21. Burner pressure: Libras por pulgada cuadrada (psi) vs RUL

Según nuestro análisis de datos exploratorios, podemos determinar que los sensores 1, 5, 6, 10, 16, 18 y 19 no contienen información relacionada con RUL ya que los valores de los sensores permanecen constantes a lo largo del tiempo. Comencemos el desarrollo de nuestro modelo con un modelo de regresión lineal de referencia. El modelo utilizará los sensores restantes como predictores (NASA, 2020).

El sensor 2 muestra una tendencia ascendente; se puede observar un patrón similar para los sensores 3, 4, 8, 11, 13, 15 y 17.

Las lecturas del sensor 6 alcanzan un máximo descendente a veces, pero no parece haber una relación clara con la disminución del RUL.

El sensor 7 muestra una tendencia a la baja, que también se puede observar en los sensores 12, 20 y 21.

El sensor 9 tiene un patrón similar al del sensor 14.

Es hora de crear nuestro modelo de referencia que tenga en cuenta todos los sensores que contienen información relacionada con RUL.

7.1.1 Regresión lineal inicial

Tenga en cuenta que el RMSE en el equipo de prueba es más bajo. Esto puede explicarse porque el RUL del conjunto de entrenamiento contiene más muestras de RUL alto en comparación con el conjunto de prueba y el RUL alto no

necesariamente se correlaciona con las señales del sensor. Ver imágenes a continuación

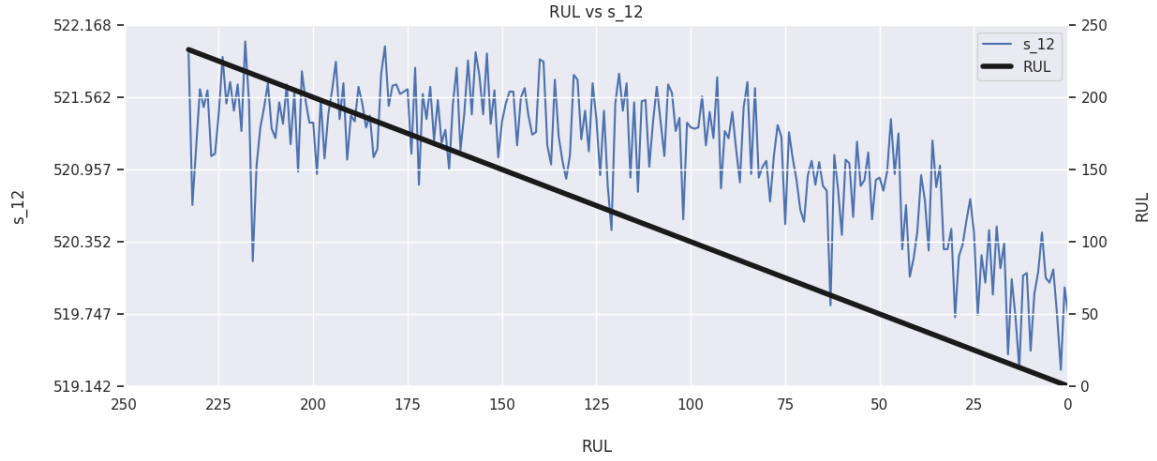

Ilustración 25: RUL vs medida del sensor 12

Ilustración 26: RUL vs medida del sensor 7

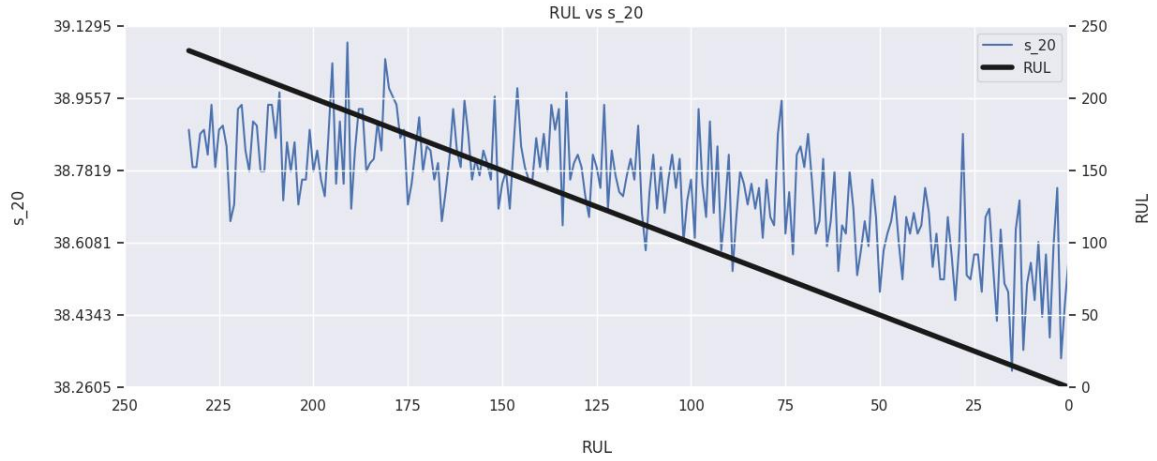

Ilustración 27: RUL vs medida del sensor 20

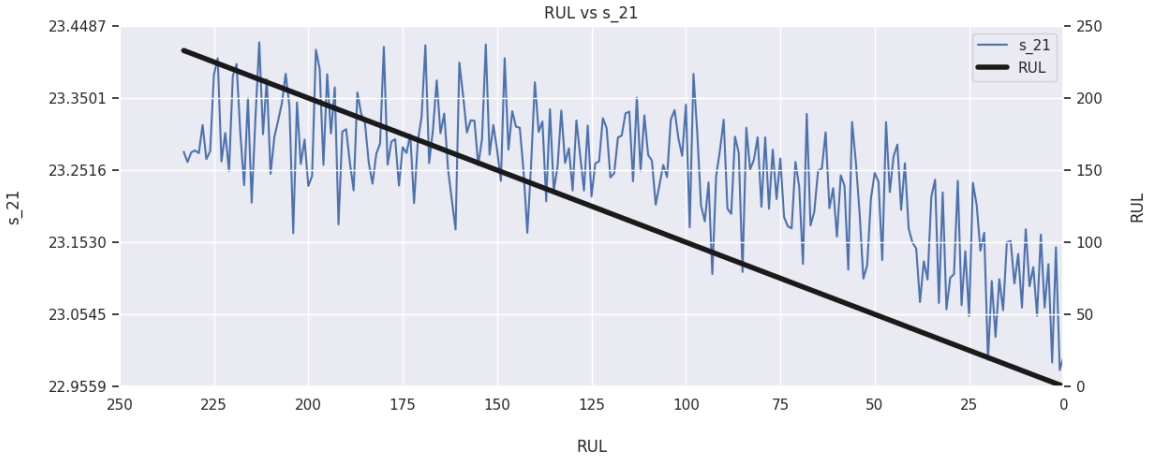

Ilustración 28: RUL vs medida del sensor 21

Para el ciclo 250 - 100 (más o menos), la señal media del sensor es bastante estable, mientras que el RUL calculado está disminuyendo.

Por debajo de 100 ciclos, tanto la señal media del sensor como el RUL calculado están disminuyendo. En esencia, la mayor correlación entre la señal del sensor y el RUL calculado en valores de RUL más bajos facilita que el algoritmo produzca predicciones más precisas

Mediante el siguiente gráfico se logró determinar que sensores aportan y en qué orden de prioridad aportan a nuestro análisis y a determinar el funcionamiento del motor. Encontrando que los datos arrojados por el sensor 6 y el sensor 10, no aportan a lo que sería la construcción de nuestro random forest (fig 29 y 30).

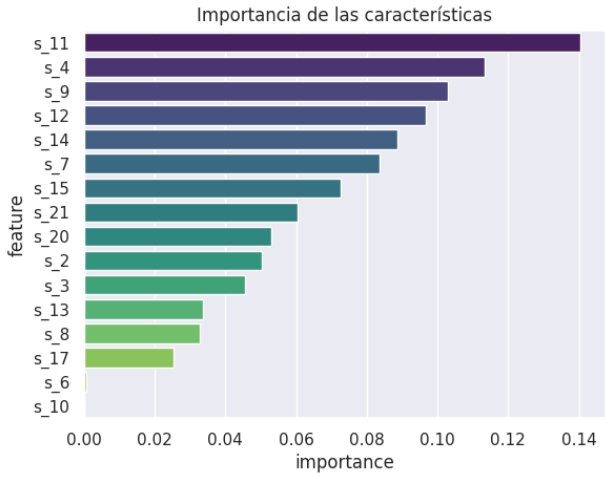

Ilustración 29: Grado de importancia de los sensores en la construcción del random forest

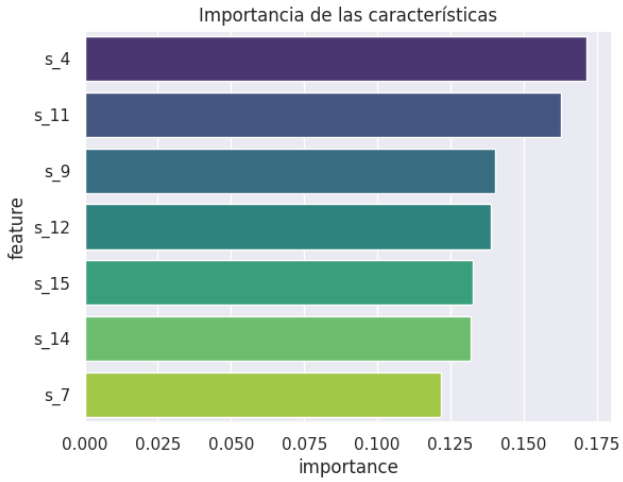

Ilustración 30: Grado de importancia de algunos sensores, descartando algunos de ellos por su grado de importancia

7.2 *Interpretación de resultados.*

Lo interesante es que la media y los cuantiles no se alinean claramente con las estadísticas descriptivas de un vector del 1 al 100, esto se puede explicar debido a que cada unidad tiene diferentes ciclos de tiempo máximos y, por lo tanto, un número diferente de filas. Al inspeccionar el tiempo máximo de ciclos, puede ver que el motor que falló primero lo hizo después de 128 ciclos, mientras que el motor que funcionó durante más tiempo se averió después de 362 ciclos. El motor medio frena entre 199 y 206 ciclos, sin embargo, la desviación estándar de 46 ciclos es bastante grande.

Si observamos las desviaciones estándar de los ajustes 1 y 2, tienden a ser inestables. Sin embargo, las fluctuaciones son tan pequeñas que no se pueden identificar otras condiciones de funcionamiento.

Finalmente, inspeccionaremos las estadísticas descriptivas de los datos del sensor, buscando indicadores de fluctuación de la señal (o ausencia de la misma).

En la segunda parte se restablecerá nuestra suposición de RUL para mejorar nuestra precisión y ajustaremos una regresión de vector de soporte (SVR) en un intento de mejorar aún más nuestros resultados

Esto sería nuestro árbol de decisiones, sin tomar en cuenta el sensor 6 y el sensor 10:

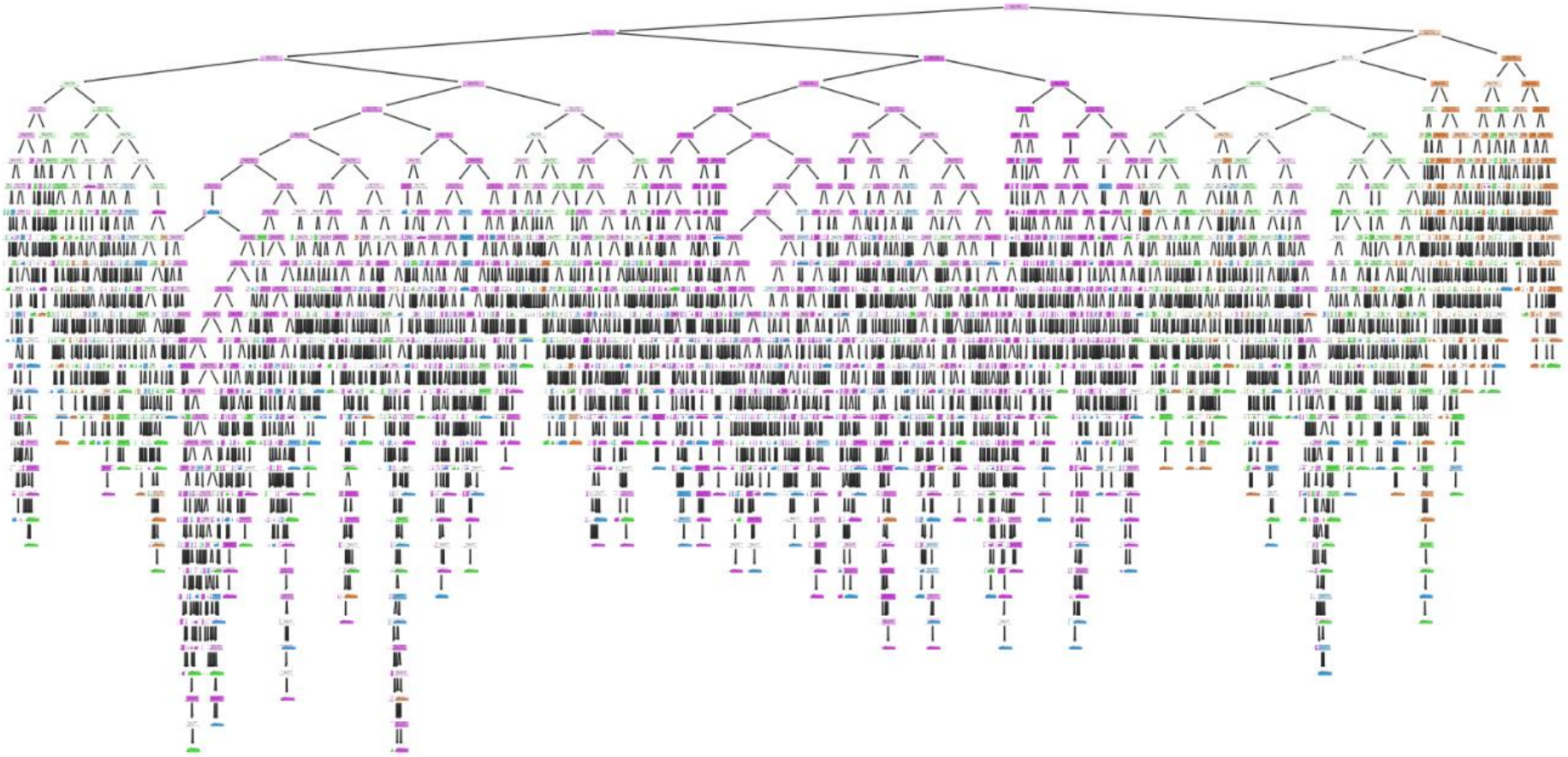

Ilustración 31: Random Forest sin podar ramas

Se lo puede comparar con el árbol que fue generado teniendo en cuenta todos nuestros sensores:

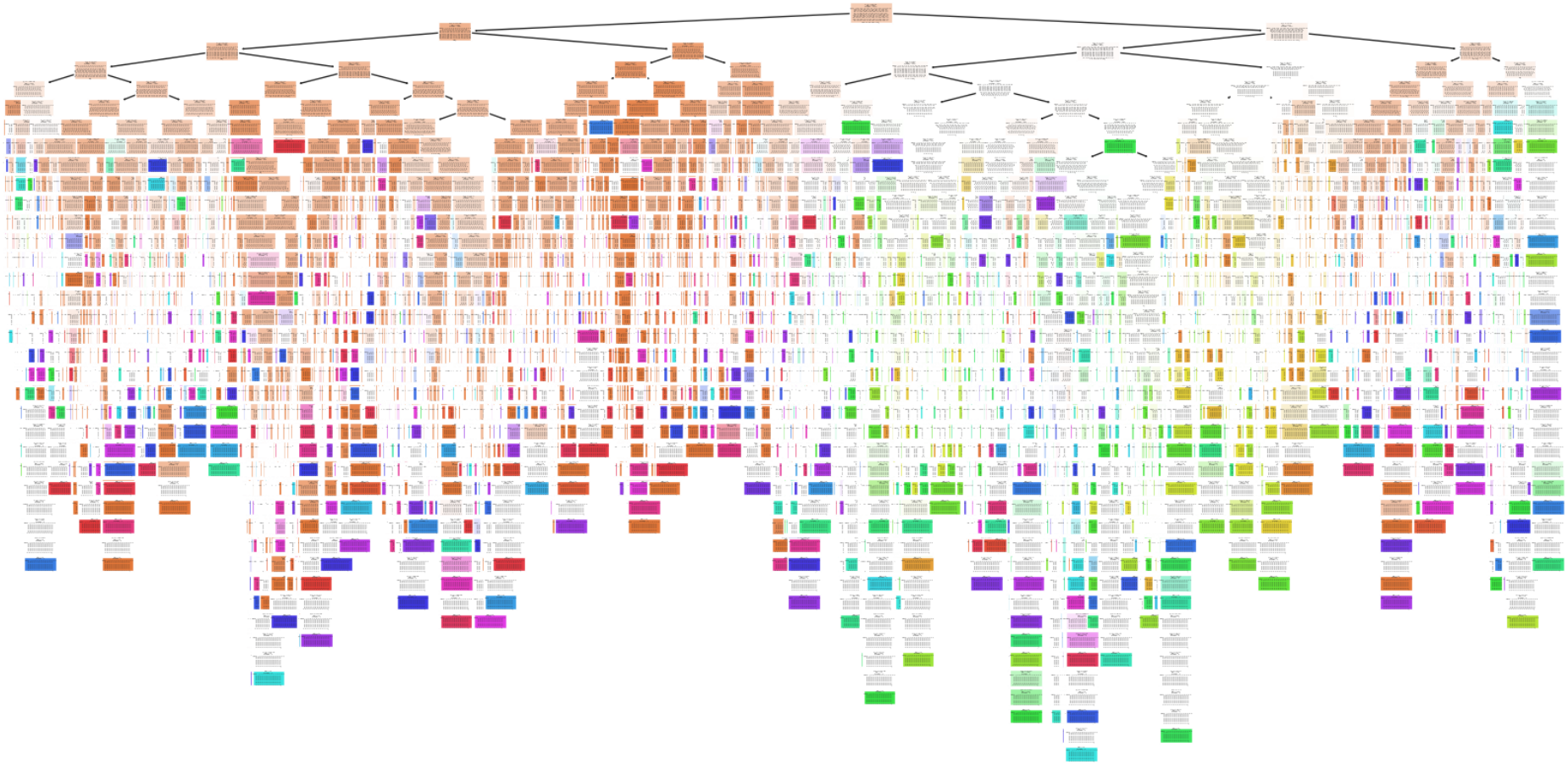

Ilustración 32: random forest estilizado

Por medio del cálculo del RUL, cada vez de manera más automatizada y con un sesgo menor, se da paso a un proceso más tecnológico aplicando el análisis de datos

8. Discusión de los resultados

A partir del análisis realizado en torno a los modelos aplicados a la base de datos, regresión lineal como Random Forest, se ha calculado lo siguiente:

Tabla 5: Tabla comparativa de los resultados tanto en el método de regresión lineal como en random forest

	REGRESIÓN LINEAL		**RANDOM FOREST**		**RANDOM FOREST MODIFICADO**	
	RMSE	**R2**	**RMSE**	**R2**	**RMSE**	**R2**
Set train	44.67	0.579	15.46	0.949	40.48	0.654
Set test	31.95	0.409	33.03	0.368	30.64	0.456

Para evaluar cuál modelo es mejor, consideramos las siguientes métricas:

RMSE: Un valor menor de RMSE indica un mejor rendimiento del modelo porque significa que las predicciones están más cerca de los valores reales.

R^2: Un valor de R^2 más cercano a 1 indica un mejor ajuste del modelo a los datos.

Para los datos de entrenamiento, el Random Forest tiene un RMSE más bajo y un R^2 más alto, indicando un mejor rendimiento en comparación con la regresión lineal.

Para los datos de prueba, el Random Forest modificado muestra un mejor rendimiento con un RMSE más bajo y un R^2 más alto que la regresión lineal. En comparación de los tres tampoco tiene un valor extremadamente alto comparando los dos, por lo que podríamos aceptar que el random forest sin modificar nos entrega resultados fiables para ambos casos.

Tenga en cuenta que el RMSE en el equipo de prueba es más bajo. Esto puede explicarse porque el RUL del conjunto de entrenamiento contiene más muestras de RUL alto en comparación con el conjunto de prueba y el RUL alto no necesariamente se correlaciona con las señales del sensor.

8.1 Desempeño del Modelo Predictivo.

- **Precisión del Modelo**: El modelo de Random Forest mostró una alta precisión en el conjunto de entrenamiento con un RMSE cercano a 0 y un R2 elevado, indicando que el modelo se ajusta bien a los datos de entrenamiento. Sin embargo, el desempeño en el conjunto de prueba fue significativamente peor, con un RMSE alto y un R2 bajo, lo que sugiere sobreajuste.
- **Errores de Predicción**: La gran diferencia entre el RMSE del conjunto de entrenamiento y el conjunto de prueba indica que el modelo puede estar aprendiendo ruidos o patrones específicos del conjunto de entrenamiento que no se generalizan bien a nuevos datos.

8.2 Importancia de los Sensores.

- **Sensores Relevantes**: Los análisis mostraron que ciertos sensores, como el s_12, tienen una relación más fuerte con el RUL en comparación con otros sensores. Esto implica que estos sensores capturan información crítica sobre la degradación del motor.
- **Menos Relevantes**: Sensores como s_6 y s_10 mostraron menor relevancia y podrían no estar aportando información útil para la predicción del RUL.

8.3 Distribución de Fallos.

- **Ciclos de Vida de los Motores**: La distribución de los ciclos de vida de los motores proporciona información sobre los patrones de uso y desgaste de los motores en la flota.

9. Propuesta de Soluciones

9.1 Mejora del Modelo Predictivo:

- **Validación Cruzada**: Implementar validación cruzada más robusta, como GroupKFold, para asegurar que el modelo se evalúa adecuadamente en diferentes subconjuntos de datos.
- **Ajuste de Hiperparámetros**: Realizar una búsqueda más exhaustiva de hiperparámetros mediante RandomizedSearchCV o GridSearchCV para encontrar los parámetros óptimos que mejoran la generalización del modelo.
- **Feature Engineering**: Crear nuevas características derivadas que puedan capturar mejor la información de los sensores, como tasas de cambio o promedios móviles.

9.2 Monitoreo y Mantenimiento Predictivo.

- **Implementación de Mantenimiento Predictivo**: Utilizar las predicciones de RUL para planificar el mantenimiento preventivo de los motores, asegurando que se realicen intervenciones antes de que ocurra una falla crítica.
- **Monitoreo en Tiempo Real**: Integrar sistemas de monitoreo en tiempo real que recopilen datos continuamente y actualicen las predicciones de RUL, proporcionando alertas tempranas sobre posibles fallos.

9.3 Optimización de Recursos.

- **Gestión de Inventario**: Optimizar la gestión de inventario de repuestos basado en las predicciones de RUL, asegurando que las piezas necesarias estén disponibles cuando se necesiten.
- **Asignación de Recursos de Mantenimiento**: Planificar y asignar recursos de mantenimiento de manera más eficiente, minimizando los tiempos de inactividad y maximizando la disponibilidad de la flota.

9.4 Investigación y Desarrollo.

- **Nuevas Bases de Datos**: Desarrollar y recopilar nuevas bases de datos con condiciones de operación más variadas, diferentes tipos de motores y más tomas de datos por vuelo para mejorar la robustez del modelo.
- **Colaboración con Fabricantes**: Trabajar en conjunto con los fabricantes de motores para mejorar el diseño y mantenimiento basado en los datos reales de operación y degradación.

10. **Implicaciones para la organización**.

Recopilando datos de cada uno de los sensores, analizando la influencia de cada uno de ellos en el cálculo de la vida útil restante del motor, con el cálculo final del RUL, obliga a que la organización, empresa, en este caso aerolínea, ajuste sus estrategias, sus planes de mantenimiento en pos de sus ciclos restantes de funcionamiento.

La empresa, en este caso la aerolínea, tendría un mayor compromiso en optimizar y reducir sus consumos de insumos de mantenimiento, repuestos adicionales y paradas de reparaciones no planificadas, alíneadas a una reducción de impacto ambiental, y un ahorro económico alto.

La implementación de la predicción del RUL en el departamento de mantenimiento tiene el potencial de transformar la gestión de activos, mejorando la eficiencia operativa, reduciendo costos, incrementando la seguridad y proporcionando una ventaja competitiva significativa. Esto permite a las organizaciones no solo mantener sus equipos en este caso su flota aérea, en mejores condiciones, sino también optimizar sus procesos y recursos de manera más efectiva.

- **Optimización del Mantenimiento:**
- **Mantenimiento Predictivo**: Con la capacidad de predecir el RUL de los motores, las aerolíneas pueden implementar programas de mantenimiento predictivo, realizando mantenimiento justo antes de que sea necesario y evitando fallos inesperados.

- **Reducción de Costos**: Al evitar mantenimientos no necesarios y minimizar tiempos de inactividad no planificados, las aerolíneas pueden reducir significativamente los costos operativos y de mantenimiento.

 Seguridad Aumentada:

- **Prevención de Fallos**: La predicción precisa del RUL ayuda a identificar problemas potenciales antes de que ocurran, mejorando la seguridad general de las operaciones de vuelo.
- **Regulación y Cumplimiento**: Mantener los motores en condiciones óptimas ayuda a cumplir con las regulaciones de seguridad y mantenimiento de la aviación.

 Gestión de Recursos:

- **Planificación Eficiente**: Conocer el estado de los motores permite a las aerolíneas planificar mejor la asignación de aeronaves y recursos de mantenimiento, optimizando la utilización de la flota.
- **Inventario de Repuestos**: La predicción del RUL permite una gestión más eficiente del inventario de repuestos, asegurando que las piezas necesarias estén disponibles cuando se necesiten.

 Mejora en la Experiencia del Cliente:

- **Minimización de Retrasos**: Al reducir la cantidad de fallos mecánicos inesperados, se minimizan los retrasos y cancelaciones de vuelos, mejorando la experiencia del cliente.
- **Confiabilidad del Servicio**: La capacidad de operar una flota de manera más confiable y eficiente aumenta la reputación de la aerolínea en términos de puntualidad y servicio.

 Análisis y Mejora Continua:

- **Retroalimentación del Sistema**: El análisis continuo de los datos de los motores y las predicciones de RUL proporciona retroalimentación para

mejorar continuamente los modelos predictivos y las estrategias de mantenimiento.

- **Innovación en Procesos**: La implementación de técnicas avanzadas de análisis de datos puede impulsar la innovación en otros procesos operativos y de mantenimiento dentro de la aerolínea.

 Decisiones Estratégicas:

- **Inversión en Tecnología**: Los resultados positivos pueden justificar inversiones en tecnologías avanzadas de monitoreo y análisis de datos.
- **Colaboración con Fabricantes**: Las aerolíneas pueden trabajar más estrechamente con los fabricantes de motores para optimizar los diseños y las estrategias de mantenimiento basadas en datos reales.

11. Conclusiones y recomendaciones.

11.1 Conclusiones.

- El histograma reconfirma que la mayoría de los motores empiezan a mostrar signos de desgaste avanzados alrededor de los 200 ciclos en adelante. Además, la distribución está sesgada hacia la derecha, con pocos motores que duren más de 300 ciclos.
- Mientras que el modelo tiene un rendimiento excelente en el conjunto de entrenamiento, su rendimiento en el conjunto de prueba indica que no generaliza bien, sugiriendo un problema de sobreajuste que debería abordarse en futuros estudios, bajo otras técnicas
- Un RMSE más bajo y un R^2 más alto en el conjunto de entrenamiento indican que el modelo tiene una alta capacidad de predicción para los datos con los que fue entrenado
- Los resultados obtenidos representan un estudio básico acerca del funcionamiento de un motor tipo turbofans; sin embargo, para una aplicación más real, es necesario, el uso de nuevas bases de datos que permitan a los algoritmos entrenarse en una mayor variedad de condiciones de operación y tipos de motor, además, considerar fases como el despegue y el aterrizaje, etapas de turbulencia, etc. Por lo tanto, aún el campo de investigación es extenso previa aplicación en operaciones reales aeronáuticas.

11.2 Recomendaciones.

- Para futuras investigaciones se recomienda realizar un análisis de supervivencia, o análisis de tiempo, el cual consiste en analizar el tiempo que transcurre hasta que ocurre un evento, en este caso la falla del motor, estimando de manera más exacta el tiempo que le quedaría de funcionamiento a los motores.
- Se recomienda explorar otros algoritmos de machine learning y técnicas de ensamble para comparar el rendimiento. La integración de más datos operativos y condiciones externas podría mejorar la precisión del modelo.
- Se recomienda aplicar técnicas de regularización, validación cruzada más robusta, y posiblemente simplificar el modelo para mejorar su capacidad de generalización

12. Referencias.

ADMINISTRATION, F. A. (2023). *Pilot's Handbook of Aeronautical Knowledge .* U.S. Department of Transportation .

Bedoya, J., Góngora, C., & García, M. (2024). Usos del Big Data en las empresas: Un instrumento de prestigio y de supervivencia hoy. *Dominio De Las Ciencias, 10*(2), 1024-1042. doi:https://doi.org/10.23857/dc.v10i2.3843

Berges, E. (2020). *Implementación y mejora de la digitalización del sistema de seguimiento del avance de la producción en el marco de la industria 4.0 dentro del sector aeroespacial.* Sevilla.

Breiman, L. (2001). Random Forest. *Machine Learning*, 5-32.

Car, E. J., Oller, S., & Oñate, E. (2000). *Tratamiento numérico de los materiales compuestos.* Barcelona: International Centre for Numerical Methods in Engineering (CIMNE).

Castro, O. (2023). *LA SOSTENIBILIDAD MEDIOAMBIENTAL EN EL SECTOR AERONÁUTICO .* COIAE.

Criminisi, A. S. (2012). Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. *Foundations and Trends® in Computer Graphics and Vision, 7*, 81-227.

Cumpsty, N. (2003). *Jet Propulsion: A Simple Guide to the Aerodynamics and Thermodynamic Design and Performance of Jet Engines.* Cambridge University Press.

Fu, S., & Avdelidis, N. (2023). Prognostic and Health Management of Critical Aircraft Systems and Components: An Overview. *Sensors*, 1-65.

García Chico, M. (2022). Preliminary design of a propulsive platform for the mcdonnell Douglas F-4 Phantom II combat aircraft. *Buleria*.

González, J. (2018). *González, Z. (2018). Diseño de una metodología para diagnosticar fallas en los componentes de productos aeronáuticos con apoyo del método centrado en confiabilidad.* Caracas.

Hastie, T., & Friedman, J. T. (2001). *The Elements of Statistical Learning* (Vol. 2). New York: springer.

Hernández Bernardo, V. (2018). Análisis y estudio de los motores a reacción civiles. *UVA*.

Heywood, J. (1988). *internal Combustion engine fundamentals.* USA: Mc Graw Hill.

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd international conference on document analysis and recognition, IEEE*, 278-282.

Lemus, P., & Werner, A. (2017). Propuesta de mejora en la eficiencia del motor V2500 de Airbus A 320. *USAC*.

Manjarrés, F. (2002). Análisis del rendimeinto y confiabilidad del motor Pratt & Whitney JT8D. *ESPE*, 16-20.

Manrique, E. (2023). *UTILIZACIÓN DE BLOCKCHAIN E INTELIGENCIA ARTIFICIAL PARA EL MANTENIMIENTO DE AVIONES.* Madrid.

Mattingly, J. D. (2002). *Aircraft Engine Design.* AIAA Education Series.

NASA. (2020). NASA’s CMAPSS data.

Papakostas, N., Papachatzakis, P., Xanthakis, V., Mourtzis, D., & Chryssolouris, G. (2010). An approach to operational aircraft maintenance planning. *Decision support systems*, 604-612.

Pater, I. d., Reijns, A., & Mitic, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect remaining useful life prognostics. *Reliability Engineering & System Safety,, 221*, 108341.

Pérez, M. Á., & Uribe, J. F. (2022). Análisis paramétrico para motores turbofán de alto bypass mediante el software matlab. .

Petrescu, R. V. (2017). Home at Airbus. *Journal of Aircraft and Spacecraft Technology*.

Rodrigo Ramírez, J. (2022). *Application of ROM-based methods to aeroengines health monitoring, performance, and control.* Madrid: Uiversidad Politécnica de Madrid.

Sánchez, Á. (2020). *Estimation of the Remaining Useful Life (RUL) of a turbofan using CNN-GRU.* Madrid: ETSIAE.

Shin, J.-H., & Jun, H.-B. (2015). On condition based maintenance policy. *Journal of Computational Design and Engineering*, 119-127.

UKPO. (1998). Patente 347.206 (1930) por Mejoras relativas a la propulsión de aeronaves y otros vehículos. . *MEHER- HOMJI*, 249- 256.

Vilana Arto, J. R. (2010). Modelización de redes virtuales de fabricación global en la industria aeronáutica. *UPM*.

Xiongzi, C., Jinsong, Y., Diyin, T., & W. Yingxun. (2011). Remaining useful life prognostic estimation for aircraft subsystems or components: A review. *10th International Conference on Electronic Measurement & Instruments,. 2*, págs. 94- 98. IEEE.

Anexos.

Anexo 1, código fuente:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

Profundicemos cargando los datos e iniciando nuestro Análisis
Exploratorio de Datos (EDA) inspeccionando las primeras filas, para tener
una idea de cómo se ven los datos.

# definir ruta de archivo y archivos para leer datos
dir_path = '/content/sample_data/tesis/'
train_file = 'train_FD001.txt'
test_file = 'test_FD001.txt'

# definir nombres de columnas para una fácil indexación
index_names = ['unit_nr', 'time_cycles']
setting_names = ['setting_1', 'setting_2', 'setting_3']
sensor_names = ['s_{}'.format(i+1) for i in range(0,21)]
col_names = index_names + setting_names + sensor_names

# leer data
train = pd.read_csv((dir_path+train_file), sep='\s+', header=None,
                    names=col_names)
test = pd.read_csv((dir_path+test_file), sep='\s+', header=None,
                    names=col_names)
y_test = pd.read_csv((dir_path+'RUL_FD001.txt'), sep='\s+', header=None,
                    names=['RemainingUsefulLife'])

# comprobar la forma e inspeccionar las primeras filas
print(train.shape)
train.head()

# inspeccionar ciclos de tiempo
train[index_names].groupby('unit_nr').max().describe()

# inspeccionar la configuración
train[setting_names].describe()

# inspeccionar los valores del sensor, específicamente si hay alguno con
una desviación estándar cercana a 0
# (las lecturas de los sensores que no cambian con el tiempo no contienen
información)
```

```
train[sensor_names].describe().transpose()

# Antes de comenzar a trazar los datos, calculemos RUL.

def add_remaining_useful_life(df):
    # Obtenga el número total de ciclos para cada unidad
    grouped_by_unit = df.groupby(by="unit_nr")
    max_cycle = grouped_by_unit["time_cycles"].max()

    # Combinar el ciclo máximo nuevamente en el marco original
    result_frame = df.merge(max_cycle.to_frame(name='max_cycle'),
left_on='unit_nr', right_index=True)

    # Calcular la vida útil restante para cada fila.
    remaining_useful_life = result_frame["max_cycle"] -
result_frame["time_cycles"]
    result_frame["RUL"] = remaining_useful_life

    # elimine max_cycle ya que ya no es necesario
    result_frame = result_frame.drop("max_cycle", axis=1)
    return result_frame

train = add_remaining_useful_life(train)
train[index_names+['RUL']].head()

def plot_sensor(sensor_name):
    plt.figure(figsize=(13,5))
    for i in train['unit_nr'].unique():
        if (i % 10 == 0):  # sólo trazar cada 10 unidad_nr
            plt.plot('RUL', sensor_name,
                     data=train[train['unit_nr']==i])
    plt.xlim(250, 0)  # invertir el eje x para que RUL cuente
regresivamente hasta cero
    plt.xticks(np.arange(0, 275, 25))

    plt.ylabel(sensor_name)
    plt.xlabel('Remaining Use fulLife RUL')
    plt.show()

for sensor_name in sensor_names:
    plot_sensor(sensor_name)

Según los gráficos, puede ver que los sensores 1, 5, 6, 10, 16, 18 y 19
no contienen información relacionada con la vida útil restante, ya que
los valores de los sensores permanecen constantes a lo largo del tiempo.

Es hora de crear nuestro modelo de referencia que tenga en cuenta todos
los sensores que contienen información relacionada con RUL.
```

Regresión inicial

```
# Primero crea una función de evaluación.
def evaluate(y_true, y_hat, label='test'):
    mse = mean_squared_error(y_true, y_hat)
    rmse = np.sqrt(mse)
    variance = r2_score(y_true, y_hat)
    print('{} set RMSE:{}, R2:{}'.format(label, rmse, variance))
# elimine columnas no deseadas y divida la variable de destino del
conjunto de entrenamiento
drop_sensors = ['s_1','s_5','s_6','s_10','s_16','s_18','s_19']
drop_labels = index_names+setting_names+drop_sensors

X_train = train.drop(drop_labels, axis=1)
y_train = X_train.pop('RUL')

# Dado que los verdaderos valores RUL para el equipo de prueba solo se
proporcionan para el último ciclo de tiempo de cada motor
# el conjunto de prueba se divide en subconjuntos para representar el
mismo
X_test = test.groupby('unit_nr').last().reset_index().drop(drop_labels,
axis=1)


print(X_train.columns)  # comprobar las columnas restantes

# crear y ajustar modelo
lm = LinearRegression()
lm.fit(X_train, y_train)

# predecir y evaluar
y_hat_train = lm.predict(X_train)
evaluate(y_train, y_hat_train, 'train')

y_hat_test = lm.predict(X_test)
evaluate(y_test, y_hat_test)

Tenga en cuenta que el RMSE en el equipo de prueba es más bajo. Esto
puede explicarse porque el RUL del conjunto de entrenamiento contiene más
muestras de RUL alto en comparación con el conjunto de prueba y el RUL
alto no necesariamente se correlaciona con las señales del sensor.

fig, ax1 = plt.subplots(1,1, figsize=(13,5))

signal = ax1.plot('RUL', 's_21', 'b',
                  data=train.loc[train['unit_nr']==20])
plt.xlim(250, 0)
plt.xticks(np.arange(0, 275, 25))
ax1.set_ylabel('s_12', labelpad=20)
```

```
ax1.set_xlabel('RUL', labelpad=20)

ax2 = ax1.twinx()
rul_line = ax2.plot('RUL', 'RUL', 'k', linewidth=4,
                    data=train.loc[train['unit_nr']==20])
ax2.set_ylabel('RUL', labelpad=20)

ax2.set_ylim(0, 250)  # establezca los límites del eje que desea mostrar
claramente
ax2.set_yticks(
    np.linspace(ax2.get_ybound()[0], ax2.get_ybound()[1], 6))  # elija un
número entero para dividir claramente su eje, en nuestro caso 6
ax1.set_yticks(
    np.linspace(ax1.get_ybound()[0], ax1.get_ybound()[1], 6))  # Aplicar
el mismo espacio al otro eje.

lines = signal+rul_line
labels = [line.get_label() for line in lines]
ax1.legend(lines, labels, loc=0)


plt.show()

import matplotlib.pyplot as plt
import numpy as np

sensors = ['s_7', 's_12', 's_20', 's_21']  # Lista de sensores de interés

for sensor in sensors:
    fig, ax1 = plt.subplots(1, 1, figsize=(13, 5))

    signal = ax1.plot('RUL', sensor, 'b',
                      data=train.loc[train['unit_nr'] == 20],
label=sensor)
    plt.xlim(250, 0)
    plt.xticks(np.arange(0, 275, 25))
    ax1.set_ylabel(sensor, labelpad=20)
    ax1.set_xlabel('RUL', labelpad=20)

    ax2 = ax1.twinx()
    rul_line = ax2.plot('RUL', 'RUL', 'k', linewidth=4,
                        data=train.loc[train['unit_nr'] == 20],
label='RUL')
    ax2.set_ylabel('RUL', labelpad=20)

    ax2.set_ylim(0, 250)  # establecer los límites del eje que desea
mostrar claramente
    ax2.set_yticks(
```

```
        np.linspace(ax2.get_ybound()[0], ax2.get_ybound()[1], 6))  #
elija un número entero para dividir claramente su eje, en nuestro caso 6
    ax1.set_yticks(
        np.linspace(ax1.get_ybound()[0], ax1.get_ybound()[1], 6))  #
Aplicar el mismo espacio al otro eje.

    lines = signal + rul_line
    labels = [line.get_label() for line in lines]
    ax1.legend(lines, labels, loc=0)

    plt.title(f'RUL vs {sensor}')
    plt.show()
```

Para el ciclo 250 - 100 (más o menos), la señal media del sensor es bastante estable, mientras que el RUL calculado está disminuyendo.
Por debajo de 100 ciclos, tanto la señal media del sensor como el RUL calculado están disminuyendo. En esencia, la mayor correlación entre la señal del sensor y el RUL calculado en valores de RUL más bajos facilita que el algoritmo produzca predicciones más precisas.

```
# datos de preparación
# elimine columnas no deseadas y divida la variable de destino del
conjunto de entrenamiento
drop_sensors = ['s_1','s_5','s_16','s_18','s_19']  # por el momento
mantendremos s_6 y s_10
drop_labels = index_names+setting_names+drop_sensors
remaining_sensors = ['s_2', 's_3', 's_4', 's_6', 's_7', 's_8', 's_9',
's_10',
       's_11', 's_12', 's_13', 's_14', 's_15', 's_17', 's_20', 's_21']


X_train = train.drop(drop_labels, axis=1)
y_train = X_train.pop('RUL')
#y_train_clipped = y_train.clip(upper=125)  # apply RUL clipping from
earlier posts

# Dado que los verdaderos valores RUL para el equipo de prueba solo se
proporcionan para el último ciclo de tiempo de cada motor,
# el conjunto de pruebas se representa en subconjuntos
X_test = test.groupby('unit_nr').last().reset_index().drop(drop_labels,
axis=1)

# importando el módulo necesario
from sklearn.ensemble import RandomForestRegressor

# random forest
rf = RandomForestRegressor(n_estimators=90, max_features="sqrt",
random_state=42)
#rf.fit(X_train, y_train_clipped)
```

```
rf.fit(X_train, y_train)


# predecir y evaluar
y_hat_train = rf.predict(X_train)
#evaluate(y_train_clipped, y_hat_train, 'train')
evaluate(y_train, y_hat_train, 'train')

y_hat_test = rf.predict(X_test)
evaluate(y_test, y_hat_test)

# realizar algunas comprobaciones en el diseño de un ÚNICO árbol
print(rf.estimators_[5].tree_.max_depth)  # comprobar cuántos nodos hay
en el camino más largo
rf.estimators_[5].tree_.n_node_samples    # comprobar cuántas muestras
hay en los últimos nodos

# bosque aleatorio modificado
rf = RandomForestRegressor(n_estimators=90, max_features="sqrt",
random_state=42,
                           max_depth=8, min_samples_leaf=50)
#rf.fit(X_train, y_train_clipped)
rf.fit(X_train, y_train)

# predecir y evaluar
y_hat_train = rf.predict(X_train)
#evaluate(y_train_clipped, y_hat_train, 'train')
evaluate(y_train, y_hat_train, 'train')

y_hat_test = rf.predict(X_test)
evaluate(y_test, y_hat_test)
#el valor del rmse es mayor en el conjunto de entrenamiento pero el r2 es
menor

#y_train_discretized = pd.cut(y_train_clipped, bins=4, labels=False)
y_train_discretized = pd.cut(y_train, bins=4, labels=False)

from sklearn.ensemble import RandomForestClassifier
# Entrenar el modelo Random Forest
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train_discretized)

# Obtener la importancia de las características la importancia de los
sensores,
importances = clf.feature_importances_
print(importances)
# Imprimir las características
feature_names = X_train.columns
print(feature_names)
```

```
# Crear un DataFrame de las características y su importancia
feature_importances = pd.DataFrame({'feature': feature_names,
'importance': importances})
feature_importances = feature_importances.sort_values(by='importance',
ascending=False)
print(feature_importances)

# Visualizar la importancia de las características
plt.figure()
sns.barplot(x='importance', y='feature', data=feature_importances,
palette='viridis')
plt.title('Importancia de las características')
plt.show()

print(X_train.columns)
X_train = X_train.drop(columns=['s_6', 's_10',
's_17','s_8','s_13','s_20','s_21', 's_2','s_3'])
#y_train_clipped = test.drop(columns=['s_6', 's_10'])

#y_train = test.drop(columns=['s_6', 's_10'])

from sklearn.ensemble import RandomForestClassifier
# Entrenar el modelo Random Forest
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train_discretized)

# Obtener la importancia de las características la importancia de los
sensores,
importances = clf.feature_importances_
print(importances)
# Imprimir las características
feature_names = X_train.columns
print(feature_names)
# Crear un DataFrame de las características y su importancia
feature_importances = pd.DataFrame({'feature': feature_names,
'importance': importances})
feature_importances = feature_importances.sort_values(by='importance',
ascending=False)
print(feature_importances)

# Visualizar la importancia de las características
plt.figure()
sns.barplot(x='importance', y='feature', data=feature_importances,
palette='viridis')
plt.title('Importancia de las características')
plt.show()

# Determinar el índice de la característica más importante
```

```
most_important_feature_index = 2
# Buscar el árbol que tiene la característica más importante en su nodo
raíz
target_tree_index = None
for i, estimator in enumerate(clf.estimators_):
  if estimator.tree_.feature[0] == most_important_feature_index:
    target_tree_index = i
    break

# Ploteamos el árbol de decisión
from sklearn.tree import plot_tree
plt.figure(figsize=(30, 15))
plot_tree(clf.estimators_[target_tree_index],
feature_names=feature_names, filled=True)
plt.show()

X_test=X_test.drop(columns=['s_6', 's_10',
's_17','s_8','s_13','s_20','s_21', 's_2','s_3'])

# Obtener las probabilidades de las clases positivas
y_prob = clf.predict_proba(X_test)[:,:]

# Obtener las predicciones del modelo
y_pred = clf.predict(X_test)

# Importar la función accuracy_score desde sklearn.metrics
from sklearn.metrics import accuracy_score

# Calcular la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo: {accuracy:.2f}")

# Ploteamos el árbol de decisión
y_train_discretized.unique()

from matplotlib import pyplot as plt
X_train['s_4'].plot(kind='hist', bins=20, title='s_4')
plt.gca().spines[['top', 'right',]].set_visible(False)

y_train.unique()

!apt-get install graphviz
!pip install graphviz



from subprocess import call
from IPython.display import Image
from sklearn.tree import export_graphviz
import graphviz
```

```
import os

# Crear y ajustar un modelo RandomForestClassifier
rf = RandomForestClassifier() # Inicializar el modelo de bosque aleatorio
rf.fit(X_train, y_train_discretized) # Suponiendo que X_train e y_train
ya estén definidos

def export_rf_visual(estimator, file_name, dpi):
    # Exportar el árbol de decisión a un archivo .dot
    export_graphviz(estimator,
                    out_file=f'{file_name}.dot',
                    feature_names=X_train.columns, # nombres de las
columnas
                    filled=True,
                    special_characters=True,
                    rotate=True)

    # Convertir el archivo .dot a .png
    call(['dot', '-Tpng', f'{file_name}.dot', '-o', f'{file_name}.png',
f'-Gdpi={dpi}'])

    print(f'estimator successfully exported to: {file_name}.png')
    return


# Crear y ajustar un modelo RandomForestClassifier
rf = RandomForestClassifier() # Inicializar el modelo de bosque aleatorio
rf.fit(X_train, y_train_discretized) # Suponiendo que X_train e y_train
ya estén definidos


# Exportar y visualizar en Google Colab
file_name = 'tree_FD003'
export_rf_visual(rf.estimators_[2], file_name, dpi=400)



# Mostrar la imagen resultante
Image(filename=f'{file_name}.png')

drop_sensors = ['s_1','s_5', 's_9', 's_14', 's_16','s_18','s_19']
drop_labels = index_names+setting_names+drop_sensors
remaining_sensors = ['s_2', 's_3', 's_4', 's_6', 's_7', 's_8', 's_10',
       's_11', 's_12', 's_13', 's_15', 's_17', 's_20', 's_21']

X_train = train.drop(drop_labels, axis=1)
y_train = X_train.pop('RUL')
#y_train_clipped = y_train.clip(upper=125)
```

```
X_test = test.groupby('unit_nr').last().reset_index().drop(drop_labels,
axis=1)

rf.get_params()

# encontrar ccp_alphas para amenorar las ramas
path = rf.estimators_[5].cost_complexity_pruning_path(X_train,
y_train_discretized)
ccp_alphas, impurities = path.ccp_alphas, path.impurities

fig, ax = plt.subplots(figsize=(13,5))

# el nodo raíz del árbol (indexado con [:-1]) se deja fuera del análisis,
ya que sin él no habría árbol
ax.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-
post")
ax.set_xlabel("alfa efectivo")
ax.set_ylabel("impureza total de las hojas")
ax.set_title("impureza total vs alfa efectivo para el conjunto de
entrenamiento")
plt.show()

Las disminuciones de impurezas normalmente se calculan para un único
árbol de decisión. Entonces, primero extraigamos los datos requeridos de
un solo árbol.

rf_dict = {
    'id_node': list(range(rf.estimators_[5].tree_.node_count)),
    'impurity': rf.estimators_[5].tree_.impurity,
    'samples': rf.estimators_[5].tree_.n_node_samples,
    'id_left_child': rf.estimators_[5].tree_.children_left,
    'id_right_child': rf.estimators_[5].tree_.children_right
}

impurity_df = pd.DataFrame(rf_dict)
print(impurity_df.shape)
impurity_df.head(10)

impurity_df['impurity_decrease'] = np.nan
samples_total = rf.estimators_[5].tree_.node_count

for idx in impurity_df.index[1:]:  # skip the first node, there aren't
any splits prior to it
    if impurity_df.iloc[idx]['id_left_child'] == -1:
        continue  # we can't calculeate impurity decrease for leave
nodes, as they no longer split
    else:
        impurity_P, samples_P = impurity_df.iloc[idx][['impurity',
'samples']]
```

```
        id_L, id_R = impurity_df.iloc[idx][['id_left_child',
'id_right_child']].astype(int)
        impurity_L, samples_L = impurity_df.iloc[id_L][['impurity',
'samples']]
        impurity_R, samples_R = impurity_df.iloc[id_R][['impurity',
'samples']]

        impurity_decrease = samples_P / samples_total * (
            impurity_P - samples_R / samples_P * impurity_R -
            samples_L / samples_P * impurity_L
            )
        impurity_df.at[idx, 'impurity_decrease'] = impurity_decrease
```

Es mejor visualizar el resultado para tener una idea de los valores adecuados para min_impurity_decrease.

```
impurity_df['impurity_decrease'].plot(kind='hist', bins=50)
```

La disminución de impurezas está muy sesgada hacia la derecha, esto puede explicarse porque los primeros nodos del árbol contribuyen en gran medida a la disminución del error. La verificación de los descriptivos muestra que el 25 % de los valores para la disminución de impurezas se encuentran por debajo de 14,59, lo que parece un límite superior adecuado para el parámetro min_impurity_decrease.

```
max_depth = [None] + list(range(3, 33, 3))  # max depth of the first RF
was 33, I don't expect we need bigger depth
min_samples_leaf = list(range(1, 102, 10))  # with crude tweaking we set
this to 50, for grid search I'll allow double
min_impurity_decrease = list(np.arange(0,147)/10)
ccp_alpha = list(np.round(np.linspace(0, 2, 81), decimals=3))

parameters = {
    'max_depth': max_depth,
    'min_samples_leaf': min_samples_leaf,
    'min_impurity_decrease': min_impurity_decrease,
    'ccp_alpha': ccp_alpha
}

tuning_options = len(max_depth) * len(min_samples_leaf) *
len(min_impurity_decrease) * len(ccp_alpha)
print(tuning_options)
```

Tenga en cuenta que si quisiéramos probar todas las combinaciones posibles, tendríamos que montar más de 1,5 millones de modelos. Una tarea tediosa. Afortunadamente, aplicar Randomsearch seleccionando al azar 60 combinaciones únicas debería acercarte al 95% de la solución óptima [8].

Aumentar el número de iteraciones aumenta la probabilidad de encontrar una mejor solución. Además de utilizar Randomsearch, prefiero mantener baja la cantidad de árboles para acelerar los tiempos de entrenamiento. Esta combinación permite una búsqueda relativamente rápida.

Ejecuté la búsqueda un par de veces y elegí la configuración con mejor rendimiento. Hay una cosa más que discutir antes de mostrar el código: necesitamos crear conjuntos de validación para validar los hiperparámetros elegidos.

Validación
Crear un conjunto de validación para estos datos requiere tener en cuenta un factor importante. Los motores que están incluidos en el conjunto de entrenamiento no pueden incluirse en el conjunto de validación y viceversa.

Normalmente, crearía una división aleatoria en sus datos donde el 80 % pertenece al conjunto de entrenamiento y el 20 % al conjunto de validación. Sin embargo, si dividimos aleatoriamente sin tener en cuenta los números de unidades, podríamos terminar con parte de los datos de un solo motor tanto en el tren como en los conjuntos de validación. Luego, el modelo podría aprender a extrapolar entre pasos de tiempo y hacer predicciones muy precisas sobre el conjunto de validación. Sin embargo, con datos verdaderamente nuevos, el rendimiento del modelo se vería afectado.

Para evitar esta forma de "fuga de datos", debemos asegurarnos de que todos los registros de un solo motor estén asignados al conjunto de capacitación o de validación. Para lograr esta forma de división de datos usaremos GroupKFold.

```
from sklearn.model_selection import RandomizedSearchCV, GroupKFold

ITERATIONS = 300

rf = RandomForestRegressor(n_estimators=100, max_features="sqrt",
random_state=42)
gkf = GroupKFold(n_splits=3)


regressor = RandomizedSearchCV(rf,
                               parameters,
                               cv = gkf.split(train,
groups=train['unit_nr']),
                               verbose=2,
                               error_score='raise',
                               n_iter=ITERATIONS,
                               n_jobs=-2,
```

```
                                # scoring https://scikit-
learn.org/stable/modules/model_evaluation.html#scoring-parameter
                                scoring= 'neg_root_mean_squared_error')
regressor.fit(X_train, y_train_discretized)
```

Dado que los tiempos de entrenamiento son relativamente bajos, configuré el número de iteraciones en 300. A continuación, se crea una instancia de RandomizedSearchCV con un RF simple, los parámetros para tomar muestras para el ajuste de hiperparámetros y GroupKFold, donde los grupos se basan en unit_nr. La búsqueda aleatoria tarda un poco menos de 15 minutos; podemos convertir los resultados en un marco de datos para una inspección más detallada.

```
results = pd.DataFrame(regressor.cv_results_)
columns = ['param_min_samples_leaf', 'param_min_impurity_decrease',
'param_max_depth', 'param_ccp_alpha',
           'mean_test_score', 'std_test_score', 'rank_test_score']
results[columns].sort_values('mean_test_score', ascending=False)
```

Inspeccionar los resultados ayuda a comprender qué hiperparámetros funcionan bien y podrían usarse para refinar su espacio de búsqueda (lo dejaremos así por ahora).

Desafortunadamente, los resultados no son reproducibles, lo que significa que reiniciar el kernel y ejecutar el portátil nuevamente no produce los mismos resultados. Algo que intentaré abordar en el próximo análisis. Después de ejecutar la búsqueda un par de veces, el conjunto de parámetros con mejor rendimiento que encontré fue:

```
rf = RandomForestRegressor(n_estimators=100, max_features="sqrt",
random_state=42,
                           min_samples_leaf=11,
min_impurity_decrease=0.0,
                           max_depth=15, ccp_alpha=0.125)
rf.fit(X_train, y_train_discretized)

# predict and evaluate
y_hat_train = rf.predict(X_train)
evaluate(y_train_discretized, y_hat_train, 'train')

y_hat_test = rf.predict(X_test)
evaluate(y_test, y_hat_test)

def export_rf_visual(estimator, file_name, dpi):
    # Exportar el árbol de decisión a un archivo .dot
    export_graphviz(estimator,
                    out_file=f'{file_name}.dot',
                    feature_names=X_train.columns, # nombres de las
columnas
```

```
                    filled=True,
                    special_characters=True,
                    rotate=True)

    # Convertir el archivo .dot a .png
    call(['dot', '-Tpng', f'{file_name}.dot', '-o', f'{file_name}.png',
f'-Gdpi={dpi}'])

    print(f'estimator successfully exported to: {file_name}.png')
    return

# Crear y ajustar un modelo RandomForestClassifier
rf = RandomForestClassifier() # Inicializar el modelo de bosque aleatorio
rf.fit(X_train, y_train_discretized) # Suponiendo que X_train e y_train
ya estén definidos


# Exportar y visualizar en Google Colab
file_name = 'tree_FD003'
export_rf_visual(rf.estimators_[2], file_name, dpi=400)



# Mostrar la imagen resultante
Image(filename=f'{file_name}.png')
from sklearn.tree import export_graphviz
from subprocess import call
from IPython.display import Image
import os

# Función para exportar el árbol de decisión a un archivo .dot y
convertirlo a imagen
def export_tree_image(tree, feature_names, file_name='tree'):
    # Exportar el árbol de decisión a un archivo .dot
    export_graphviz(tree, out_file=f'{file_name}.dot',
                    feature_names=feature_names,
                    filled=True,
                    rounded=True,
                    special_characters=True)

    # Convertir el archivo .dot a imagen .png
    call(['dot', '-Tpng', f'{file_name}.dot', '-o', f'{file_name}.png',
'-Gdpi=300'])

    # Mostrar la imagen
    return Image(filename=f'{file_name}.png')

# Seleccionar el árbol con la mayor profundidad
max_depth_tree = max(rf.estimators_, key=lambda est: est.tree_.max_depth)
```

```
# Exportar y visualizar el árbol
feature_names = X_train.columns  # Asegúrate de que `X_train` esté
definido con las características correctas
tree_image = export_tree_image(max_depth_tree, feature_names)
tree_image
```