UNIVERSIDAD EUROPEA DE MADRID

FACULTAD DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA (MUIA)

Gemelo digital de un proceso aeronáutico y análisis de sensibilidad analítico

CURSO 2022-24

GRUPO M11

Alejandro NÚÑEZ FONTÁN

Título: Gemelo digital de un proceso aeronáutico y análisis de sensibilidad analítico con inteligencia artificial

Autor: Alejandro Núñez Fontán

Tutor: Miguel Ángel Castillo Acero

Titulación: Máster Universitario en Ingeniería Aeronáutica

Curso: 2022-2024

Resumen:

Como un sistema altamente seguro y confiable, la aviación enfrenta desafíos que incluyen la transformación digital, costos elevados de operación y producción, baja eficiencia en el mantenimiento y tecnología intensiva. Una de las tecnologías más beneficiosas es el gemelo digital (DT), que puede resolver los problemas mencionados anteriormente.

En este proyecto, se va a realizar un gemelo digital sobre el funcionamiento o las prestaciones de una determinada configuración de motor para distintos conjuntos de los parámetros de entrada y salida que determinan las actuaciones de dicho motor, en este caso un turbofán de doble flujo. El gemelo digital va a estar basado en un código de Python que simula toda la física y termodinámica embebida en el software Gasturb.

Posteriormente, se va a construir un sistema de calificación que evalúe la calidad de cada iteración en función de unos rangos aceptables y un valor óptimo para cada parámetro en función de los resultados obtenidos. Por último, se va a entrenar a un algoritmo de inteligencia artificial para que aprenda a calificar cada iteración del mismo modo en base a los conjuntos de datos obtenidos y encontrar lógicas de comportamiento entre los distintos subconjuntos de parámetros.

Tabla de contenido

Tabla de figuras

Tablas

1. Objetivo

El objetivo de este proyecto es aportar las bases de conocimiento para crear un gemelo digital de un proceso aeronáutico y realizar un análisis de sensibilidad analítico con alguna técnica de 'machine learning' que permita optimizar su funcionamiento y monitorear en tiempo real los parámetros que lo definen para la toma eficiente de decisiones.

El gemelo digital que se va a desarrollar para estudiar con detalle los elementos clave de esta metodología es la simulación de las prestaciones de un motor turbofán de doble flujo en diferentes condiciones de funcionamiento según los parámetros o conjuntos de parámetros que se quieran optimizar en cada caso y que definen intrínsicamente dichas prestaciones.

2. Introducción

Este trabajo responde al reto de elegir un proceso aeronáutico y caracterizar todos los parámetros, variables, condicionantes, etc. que pueden afectar los resultados. El trabajo se centra en realizar un modelo analítico al nivel de gemelo digital, que tiene en cuenta todos los elementos que influyen en el proceso. La variación de estos condicionantes de partida tiene un efecto potencial en las características finales que serán correlacionadas con un análisis de sensibilidad numérico, si es posible con una aproximación de inteligencia artificial.

Un gemelo digital es una réplica virtual de un objeto, proceso o sistema del mundo físico. Es una representación digital precisa y detallada que combina datos tomados por sensores en tiempo real con modelos y simulaciones para permitir la monitorización, análisis y predicción de su comportamiento y tendencias.

El gemelo digital se crea utilizando tecnologías como la inteligencia artificial, la Internet de las cosas (IoT), la realidad aumentada y la computación en la nube. Recopila datos en tiempo real del objeto o sistema físico mediante sensores, dispositivos y otros medios de recopilación de datos. Estos datos se utilizan para actualizar y ajustar constantemente la réplica digital, lo que permite una visión precisa y actualizada del objeto o sistema en cualquier momento y analizar las tendencias.

La idea detrás de un gemelo digital es poder comprender, analizar y optimizar el funcionamiento de un objeto o sistema en el mundo físico sin tener que intervenir directamente en él. Por ejemplo, en la industria manufacturera, se puede crear un gemelo digital de una planta de producción para simular y optimizar procesos, hacer mantenimiento preventivo, predecir problemas y mejorar la eficiencia.

Además, los gemelos digitales también se utilizan en sectores como la medicina, la energía, la construcción, la logística y la gestión de ciudades inteligentes, entre otros. Ayudan a tomar decisiones informadas, prevenir fallas, optimizar el rendimiento y reducir costos mediante la simulación y análisis en un entorno virtual antes de implementar cambios o mejoras en el mundo real.

En resumen, un gemelo digital es una réplica virtual en tiempo real de un objeto o sistema del mundo real que permite el monitoreo, análisis y optimización de su funcionamiento. Es una herramienta poderosa para mejorar la eficiencia, la productividad y la toma de decisiones en diversos sectores.

La implementación de la tecnología del gemelo digital implica varios pasos y componentes clave. A continuación, se describen los pasos generales para implementar un gemelo digital:

1. Definir el objetivo: Lo primero es definir el propósito y el objetivo del gemelo digital. ¿Qué objeto o sistema se va a replicar digitalmente y con qué finalidad? Esto puede incluir optimizar el rendimiento, predecir fallas, mejorar la eficiencia, simular escenarios, entre otros.
2. Recopilación de datos: Para crear y actualizar el gemelo digital, es necesario recopilar datos del objeto o sistema en tiempo real. Esto puede incluir datos de sensores, dispositivos IoT, sistemas de monitoreo, bases de datos, etc. Es importante determinar qué datos son relevantes y necesarios para el gemelo digital.
3. Modelado y simulación: Con los datos recopilados, se crean modelos y simulaciones que representen el objeto o sistema en el gemelo digital. Estos modelos pueden variar en complejidad según los requisitos del proyecto. Se utilizan técnicas como la inteligencia artificial, la física computacional y otras herramientas de modelado para crear una representación precisa.
4. Integración de datos en el modelo: Los datos recopilados en tiempo real se integran con los modelos y simulaciones del gemelo digital. Esto permite que el gemelo digital se actualice y refleje los cambios y condiciones en el objeto o sistema del mundo físico. La integración puede requerir el uso de plataformas de IoT, sistemas de almacenamiento en la nube, APIs y otras tecnologías de conectividad.
5. Análisis y visualización: Una vez que el gemelo digital está en funcionamiento, se pueden realizar análisis y visualizaciones para comprender y tomar decisiones basadas en los datos recopilados. Esto puede incluir la detección de anomalías, la identificación de patrones, la simulación de escenarios, la optimización de procesos, etc. Se pueden utilizar herramientas de análisis de datos y visualización para explorar y comprender la información del gemelo digital.
6. Retroalimentación y mejora continua: El gemelo digital es un proceso en evolución. Se requiere una retroalimentación constante del objeto o sistema del mundo físico y del rendimiento del gemelo digital para realizar ajustes y mejoras. Esto implica analizar los resultados, compararlos con los datos del mundo real y realizar cambios en los modelos y simulaciones para que el gemelo digital sea cada vez más preciso y útil.

Es importante destacar que la implementación de un gemelo digital puede variar dependiendo de la aplicación y el sector. La colaboración entre expertos en el dominio del objeto o sistema real, científicos de datos, ingenieros y profesionales de TI es fundamental para lograr una implementación exitosa y obtener beneficios significativos del gemelo digital.

3. Estado del arte

El origen del concepto gemelo digital puede fijarse en el año 2002 en la Universidad de Michigan en el contexto de una presentación realizada por Michael Grieves sobre la gestión del ciclo de vida de los productos. A pesar de que inicialmente no se denominó a este concepto gemelo digital, sí que se definieron los tres aspectos básicos que debía reunir: mundo real, mundo virtual y transferencia de información entre ambos. Desde entonces el concepto se ha ido desarrollando, matizando y más recientemente ha comenzado a ser una realidad en grandes empresas por todo el mundo.

La Industria 4.0 nace de ciertos avances sobre las tecnologías digitales y de automatización desarrolladas a lo largo de la tercera revolución industrial. Estos avances consisten principalmente en la mejora de las capacidades de procesamiento de datos gracias a filosofías como los sistemas distribuidos o el Edge Computing, así como el desarrollo de elementos electrónicos cada vez más pequeños, y en la mejora de las velocidades de comunicación con desarrollos como el LTE o el 5G. Estos avances han permitido combinar la parte física de la producción con la parte virtual, creando los llamados sistemas ciberfísicos (CPS) y permitiendo una integración vertical y horizontal de la producción. Es decir, la Industria 4.0 busca optimizar los procesos de producción mediante el uso de CPS que permitan formas de producción sostenibles.

La aplicación del gemelo digital (DT) se remonta a la década de 1970; en el programa Apolo, la Administración Nacional de Aeronáutica y del Espacio (NASA) construyó dos vehículos espaciales idénticos; el primero, dejado en la Tierra, mapearía el estado de funcionamiento del segundo vehículo. En 2010, la NASA señaló que un DT está compuesto por productos físicos, productos virtuales y comunicaciones entre los dos vehículos. El DT integra simulaciones “multiphysics”, “multiscale” y probabilísticas para mapear el estado del vehículo físico en tiempo real a través de actualizaciones de sensores y datos históricos de servicio. En 2012, el Laboratorio de Investigación de la Fuerza Aérea (AFRL) consideró que el DT del fuselaje (ADT) es un sistema integrado compuesto por datos, modelos y herramientas de análisis. Este sistema puede representar el fuselaje de la aeronave en todo su ciclo de vida y tomar decisiones sobre toda la flota y un solo fuselaje según información incierta. Tao y otros añadieron dos dimensiones basadas en la definición de la NASA, y el DT se compone de cinco partes, a saber, física, virtual, conexiones, datos y servicios.

Supongamos que el CAD, el CAE, el CAM, la ingeniería de sistemas tradicional y otras tecnologías sientan una base sólida para el desarrollo del DT; en ese caso, el rápido crecimiento de tecnologías avanzadas como los Sistemas Ciberfísicos (CPS), IoT, big data, IA, la informática en la nube, la informática en el borde, redes de sensores, identificación por radiofrecuencia y redes celulares 5G están proporcionando avances para el desarrollo del DT en todos los campos. En los últimos años, diversos campos han comenzado a darse cuenta gradualmente de la importancia y los beneficios del DT y han comenzado a investigarlo. La aplicación del DT es diversa y muchas empresas actualizan continuamente el desarrollo del concepto del DT según sus productos y necesidades. Por ejemplo, GE cree que el DT representa la forma de software de activos y procesos y se puede utilizar para comprender, predecir y optimizar el rendimiento. Su propósito es mejorar el rendimiento de los activos y procesos.

Siemens propuso el concepto de DT integral y señaló que se pueden entregar productos ideales a través de un mapeo preciso entre productos DT, producción y gestión. El nuevo concepto de Dassault es el "3D EXPERIENCE TWIN", que enfatiza la consistencia de la experiencia, la consistencia de los sujetos, una sola fuente de datos y la unidad macro y micro entre los productos físicos y los gemelos.

PTC, una empresa de software americana especializada en modelos del ciclo de vida de producto, propone el concepto de "DT + VR" al centrarse más en la sensación de realidad y la escena de los productos.

Por su parte, ESI, una empresa de software europea con soluciones de simulación multifísica, considera que la solución de gemelos híbridos permite a las empresas proporcionar virtualmente un mantenimiento predictivo y optimizar las operaciones auxiliares de los productos mediante la conexión de la información actual de IoT y los datos históricos.

Aunque la literatura anterior muestra que las definiciones de los gemelos digitales (DT) son diferentes en la academia y la industria, todavía existen algunas características comunes:

(1) utilizan activos físicos para representar la realidad;

(2) el DT construido puede mejorarse o actualizarse basándose en la retroalimentación mutua entre activos físicos y virtuales

(3) modelos de alta fidelidad son esenciales para la realización del DT. Estas características son similares a los elementos de la simulación, los sistemas ciberfísicos (CPS) y el Internet de las cosas (IoT). Cabe mencionar que existen diferencias y conexiones entre ellos.. La simulación es la base del DT, el DT es el requisito previo para el desarrollo de los CPS y el IoT es la infraestructura del DT y los CPS.

La simulación es la tecnología principal para realizar el DT, que puede crear y ejecutar el DT. Además, es una tecnología central para asegurar que el DT y las entidades físicas correspondientes puedan lograr un bucle cerrado efectivo. La principal diferencia entre el DT y las simulaciones tradicionales radica en que el DT requiere una iteración continua entre entidades físicas y virtuales. Las entidades digitales virtuales construidas por el DT necesitan utilizar simulaciones en constante evolución, con alta frecuencia, y acompañar todo su ciclo de vida.

En el ámbito de la fabricación, tanto los sistemas ciberfísicos (CPS) como los DT incluyen dos partes: el mundo físico real y el mundo de la información virtual. El mundo físico lleva a cabo las actividades de producción reales, mientras que diversas aplicaciones y servicios realizan la gestión inteligente de datos, el análisis y los cálculos en el mundo de la información virtual. El DT enfatiza el mapeo en tiempo real de entidades físicas y modelos virtuales, mientras que los CPS se encargan de la recopilación de información, el procesamiento y el control de todo el sistema.

Por lo tanto, el DT es el requisito previo para la implementación de los CPS. El IoT realiza la identificación de información y el seguimiento entre entidades a través de datos en red; proporciona una recopilación, transmisión y conexión de datos integral; y realiza una interconexión efectiva entre modelos virtuales y entidades físicas, lo cual es uno de los elementos clave del DT y los CPS.

La ingeniería digital es un método digital integrado que utiliza la fuente de modelos autorizados y la fuente de datos del sistema para respaldar todas las actividades, desde el desarrollo del concepto hasta la gestión de desechos. Construye un ecosistema de ingeniería digital que aprovecha los avances digitales e innovaciones tecnológicas para completar un cambio de paradigma centrado en el modelo.

El modelo del sistema digital utiliza datos para representar los elementos e información del sistema. Está basado en el modelo de datos del sistema y constituye la base de la ingeniería digital. Como tal, la cantidad y los tipos de datos en el modelo de sistemas digitales crecerán a medida que el sistema madure a lo largo de su ciclo de vida. Utiliza la clasificación definida por el modelo de sistemas digitales para proporcionar a la ingeniería digital datos sistemáticos de ingeniería, datos de proyectos y datos de soporte del sistema.

El hilo digital es un marco analítico a nivel empresarial con atributos de extensibilidad, configurabilidad y componentes. Basado en la plantilla del modelo de sistemas digitales, puede transferir información precisa al objeto correcto en el correcto momento y lugar durante el ciclo de vida del sistema. El hilo digital debe materializarse en el entorno del gemelo digital (DT).

El núcleo del ecosistema de ingeniería digital está formado por el modelo de sistemas digitales, el hilo digital y el DT, y su interacción puede mejorar la eficiencia, agilidad y flexibilidad en todo el proceso de adquisición, lo que permite tomar decisiones mejores a lo largo del ciclo de vida.

Precedentes del gemelo digital en la industria aeronáutica

Las técnicas de simulación tradicionales, como la dinámica de fluidos computacional, el método de elementos finitos, la simulación de Monte Carlo, etc., pueden reproducir la historia en tiempo continuo del vuelo y generar una gran cantidad de datos de simulación para evaluar el rendimiento de vuelo y predecir los futuros requisitos de mantenimiento e invasiones utilizando técnicas de simulación basadas en diversas aplicaciones.

Sin embargo, en comparación con la modelización y simulación tradicionales, el DT tiene las siguientes ventajas: ciclo de diseño corto, alta confiabilidad, mantenimiento frecuente y costos bajos de mantenimiento. No solo puede acortar los tiempos y la duración de las pruebas de certificación de aeronaves, eliminar grietas y fallos accidentales, sino también reducir la frecuencia de las inspecciones de mantenimiento estructural.

En 2011, la Fuerza Aérea de los Estados Unidos propuso un proceso de predicción de vida rediseñada de la estructura de la aeronave para aprovechar al máximo la informática digital de alto rendimiento. Señalaron que en 2025 entregarían el primer nuevo tipo de avión y un modelo digital del avión al mismo tiempo. Consideraban que el DT era la reconstrucción de la predicción y gestión de la vida estructural y también enumeraron las tecnologías requeridas, que incluyen:

- Modelización multiphysics.
- Modelización de daños en múltiples escalas.
- Integración del método de elementos finitos estructurales y modelos de daños.
- Cuantificación de la incertidumbre.
- Modelización y control.
- Manipulación de bases de datos grandes.
- Análisis estructural de alta resolución.

En 2012, la Fuerza Aérea de Investigación de la Fuerza Aérea (AFRL) propuso por primera vez el concepto de ADT: el ADT es un modelo estructural de aeronaves que puede cumplir con los requisitos de misión de la aeronave en todo su ciclo de vida. Es un submodelo que incluye electrónica, control de vuelo, sistema de propulsión y otros subsistemas.

El ADT puede lograr los siguientes objetivos:

(1) un diseño y certificación más eficiente, que no solo reduce la cantidad de cambios de diseño después de las pruebas de certificación de la aeronave, sino que también acorta la cantidad y el tiempo de las pruebas de certificación, reduciendo así el costo de adquisición
(2) eliminar las grietas y fallas accidentales, y aumentar la disponibilidad y confiabilidad de la aeronave
(3) minimizar la cantidad y frecuencia de inspecciones estructurales y reducir los costos de mantenimiento y soporte.

Además, la AFRL señaló que todavía existen muchos desafíos en la modelización de la física y la ingeniería en la implementación del ADT. Estos desafíos se pueden dividir en las siguientes categorías:

- Establecimiento de condiciones iniciales.
- Aplicación de cargas de vuelo.
- Selección e integración de submodelos.
- Gestión y reducción de la incertidumbre.

En el mismo año, la NASA colaboró con la Oficina de Investigación Científica de la Fuerza Aérea (AFOSR) y sostuvo que, en comparación con la generación actual, el futuro necesitaría una masa más ligera y, al mismo tiempo, soportaría cargas más altas y condiciones de servicio más extremas durante más tiempo. Para superar las limitaciones de los métodos tradicionales, se necesita un cambio fundamental de paradigma. Este cambio de paradigma, es decir, el DT, enfatiza la combinación de simulaciones de ultra alta fidelidad, un sistema de gestión de salud y la recopilación de la mayor cantidad de datos históricos posible para reflejar la vida útil de su "gemelo" en vuelo, finalmente alcanzando un nivel de seguridad y confiabilidad mejorado.

En 2013, la AFRL desarrolló el programa ADT Spiral 1, en el que se está desarrollando y demostrando un proceso de seguimiento probabilístico de fatiga de aeronaves individuales. Una gran cantidad de fuentes de incertidumbre, como las propiedades de crecimiento de grietas por fatiga, el tamaño inicial de las grietas y las cargas, se capturan en forma de distribuciones de probabilidad que se utilizan en las predicciones de crecimiento de grietas por fatiga. El marco de demostración incluye cargas y entornos aplicados, datos de geometría y material, modelos multidisciplinarios multiescala informados por la física, rango de respuesta estructural y confiabilidad, y aeronaves físicas.

En 2014, para integrar mejor el DT y los materiales de sensores, la NASA investigó cómo modelar la geometría de fabricación y la microestructura de componentes tanto como sea posible. Esta investigación tiene como objetivo desarrollar el concepto de materiales de sensores de modo que se puedan utilizar dentro del marco del DT. Al combinar el DT y la tecnología de materiales de sensores, se puede lograr la inspección y el mantenimiento automáticos basados en evidencia de vehículos aeroespaciales. Basándose en la definición del DT de la NASA, el Departamento de Defensa de los Estados Unidos (DoD) propuso que el DT era un sistema habilitado por un hilo digital. El hilo digital puede proporcionar datos al modelo de ordenador original de la pieza.

Es necesario actualizar el modelo físico con la máxima fidelidad utilizando datos de registros de vuelo, datos del sistema de monitorización aérea y datos históricos de la flota para lograr un diagnóstico de fallos predecible y la evolución de la fatiga. El Consejo Nacional de Investigación de Canadá (NRC) define el ADT, basado en el Departamento de Defensa (DoD), que revisa y evalúa el ADT propuesto por el AFRL de los Estados Unidos, y enfatiza el papel del DT en el seguimiento individual de aeronaves.

El marco describe 5 proyectos: base de datos común de la flota, DT individual, evaluación cuantitativa de riesgos, aeronaves físicas individuales, inferencia bayesiana.

En 2019, la Fuerza Aérea de los Estados Unidos propuso el plan "La fuerza aérea digital" para enfrentar el complejo entorno de seguridad, que hizo hincapié en las ventajas de red de los sensores y las herramientas de análisis. Utilizando de manera integral las "herramientas de Trinidad" de arquitectura de sistemas abiertos modulares, desarrollo ágil de software e ingeniería digital, el avión de combate se actualiza cada 4 años con alta frecuencia, logrando así una investigación y desarrollo en espiral. El DT se puede utilizar para la optimización del mantenimiento y el progreso de las inspecciones basado en una evaluación cuantitativa de riesgos, y para gestionar el ciclo de vida estructural de la aeronave con costos significativamente reducidos mientras se maximiza la disponibilidad.

Por lo tanto, la aeronave con DT combina estrechamente la tecnología de DT con los eslabones cruciales, escenarios y objetos cruciales en la ingeniería aeronáutica. La ingeniería de aeronaves en el espacio físico se simula, monitorea y refleja en tiempo real en función de modelos y datos, luego se analiza, evalúa, pronostica, gestiona y optimiza utilizando algoritmos, métodos de gestión, conocimientos de expertos y software. Las funciones incluyen la aplicación de servicio en diversas escenas y objetos en la dimensión espacial y la gestión de ingeniería del sistema en la dimensión temporal.

Aplicación del DT

El DT se aplica a lo largo de todo el ciclo de vida de las aeronaves. Un modelo de alta fidelidad y datos abundantes son condiciones vitales para garantizar la implementación del DT, que puede servir mejor a las actividades seguras de la aeronave. En la literatura se han encontrado referencias que artículo revisan el estado de aplicación del DT en la industria de la aviación desde 2010 hasta agosto de 2021. Su aplicación abarca cinco aspectos: diseño de optimización estructural, ensamblaje de productos, fabricación de productos, operación y mantenimiento, y otros campos relacionados con la aviación.

Como una tecnología clave de la industria de fabricación de aeronaves, la tecnología de colaboración en el ensamblaje es de gran importancia para mejorar la competitividad y el nivel de fabricación. La tecnología de colaboración en el ensamblaje se refiere a la utilización de herramientas tecnológicas para mejorar la eficiencia y la precisión en los procesos de ensamblaje industrial. Esto implica el uso de software, hardware y sistemas de comunicación que facilitan la colaboración entre los trabajadores, equipos y departamentos involucrados en el proceso de ensamblaje. Algunas tecnologías de colaboración comunes en el ensamblaje incluyen sistemas de gestión de la cadena de suministro (SCM), software de diseño asistido por computadora (CAD), sistemas de manufactura asistida por computadora (CAM), robótica colaborativa, sistemas de comunicación y colaboración en línea, y sistemas de seguimiento y control de la producción.

Las aeronaves son productos complejos, y su proceso de ensamblaje implica una alta complejidad, una fuerte dinámica, incertidumbre y mantenimiento frecuente, por lo que los requisitos de control de calidad para el ensamblaje de aeronaves son extremadamente estrictos. Los datos de calidad tradicionales de las aeronaves están dispersos en numerosos sistemas de negocios y de gestión, a menudo se utiliza el procesamiento manual para localizar problemas de calidad y su gestión en bucle cerrado es débil.

Aplicación del DT en los procesos de ensamblaje

El proceso de ensamblaje se divide en tres fases: programación en el taller, ensamblaje en el sitio y retroalimentación de problemas. Combinar el DT con el hilo digital puede crear un ciclo de información cooperativo e interconectado, que se denomina fabricación en bucle cerrado y permite a las empresas sincronizar y optimizar la producción entre el diseño, la programación de producción, la fabricación, la automatización y el uso. La adquisición, procesamiento y gestión de datos físicos de ensamblaje contribuyen a establecer el DT de ensamblaje y a predecir impulsados por grandes datos, y a completar servicios de gestión y control de producción de ensamblaje basados en DT. El modo de trabajo basado en la coordinación digital puede obtener con precisión el tamaño, la forma, la posición espacial, la precisión de la coordinación y otra información de los productos, y se puede establecer un modelo de coordinación de ensamblaje basado en datos de medición para controlar la calidad del ensamblaje de aeronaves, lo que aumenta en gran medida la eficiencia del ensamblaje. El DT puede lograr la integración de los datos de calidad del ensamblaje de aeronaves y utilizar el modelo 3D de la aeronave para lograr la sincronización del mundo virtual-físico del proceso de ensamblaje, y mejorar la eficiencia y precisión de la calidad del ensamblaje en el mundo físico. La combinación de monitoreo de desplazamiento multipunto en línea con la teoría de completitud de matrices puede lograr la percepción en tiempo real del desplazamiento del ensamblaje de aeronaves. El DT proporciona apoyo de conocimiento para el diseño inteligente de la línea de ensamblaje de aeronaves para mejorar la eficiencia, el rendimiento y la visibilidad. El sistema basado en `blockchain` proporciona una plataforma de gestión para registrar con precisión los datos de trazabilidad de las piezas de repuesto de aeronaves, logra la integridad de la información en el proceso de operación de transacción y mejora la calidad de los datos de trazabilidad y el intercambio de información confiable en la cadena de suministro de piezas de repuesto. La plataforma consta de una capa de hardware, una capa de software y una capa de aplicación, que pueden optimizar y controlar inteligentemente el proceso de ensamblaje de productos aeroespaciales.

El DT de ensamblaje de aeronaves puede recopilar datos de calidad completo para realizar la modelización sincrónica del proceso de ensamblaje del producto y la generación de paquetes de datos de ensamblaje, lo que hace que el proceso de productos complejos sea rastreable y mejora eficazmente la capacidad de gestión de calidad del proceso de ensamblaje final de aeronaves, la calidad del ensamblaje y la eficiencia.

Aplicación del DT en los procesos de manufactura

En la actualidad, la tecnología del DT es la más activa en la fabricación inteligente. Airbus y Boeing están construyendo aeronaves gemelas digitales, y GE está desarrollando motores gemelos digitales. La digitalización de la fabricación acelera la aplicación práctica de modelos complejos de productos virtuales a lo largo de todo el ciclo de generación de productos. En particular, los modelos de alta fidelidad de productos manufacturados son esenciales para reducir la brecha entre el diseño y la fabricación y reflejar el mundo real y virtual.

Dado que el fresado de las palas o álabes del motor es propenso a deformaciones y requiere mucho tiempo, la precisión y eficiencia del proceso de fresado de las palas del motor se ven afectadas. El modelo DT de procesos de fresado integra datos relevantes del proceso de mecanizado y modelos de fuerza de corte, lo que permite reconocer desviaciones en la trayectoria directamente en el proceso de mecanizado y optimizar el proceso de mecanizado.

La eficiencia y precisión del mecanizado de las palas pueden mejorarse mediante métodos de optimización de parámetros, que integran algoritmos inteligentes con bases de datos de procesamiento y finalmente se aplican a la fabricación de palas. Construyendo el DT de una pala de turbina de motor aeroespacial y simulando su proceso de fabricación y proceso de inspección de calidad, se logran los objetivos de reducir el tiempo de mecanizado, mejorar la precisión del mecanizado y aumentar la capacidad de fabricación. La optimización del proceso de fabricación de aeronaves también puede aumentar eficazmente su eficiencia y seguridad.

Sin embargo, durante la fabricación aditiva con DT, existen incertidumbres en el rendimiento de los materiales y en la estabilidad del proceso, lo que permite la gestión y control de la incertidumbre del proceso de fabricación mediante el diseño de controladores para el DT. El proceso de mecanizado se simula en el espacio virtual y se realiza un mecanizado de precisión en tiempo real. El control numérico por computadora monitoriza las señales y emite una advertencia temprana para el proceso problemático.

La simulación del sistema y la optimización de parámetros del modelo se llevan a cabo a través de la interacción y el registro de datos. El módulo DT recibe datos de retroalimentación en tiempo real de la medición de la fabricación y las pruebas de rendimiento y realiza correcciones durante todo el proceso de optimización. El algoritmo de simulación de mecanizado optimizado en tres ejes ha sido probado y verificado por la Corporación de Aeronaves Comerciales de China Ltd.; puede seguir eficientemente el proceso de mecanizado real. Se propuso un método de modelado DT basado en el principio de la bio-mezcla y desarrolló varios modelos de DT, como el modelo geométrico, el modelo de comportamiento y el modelo de proceso, que pueden interactuar entre sí para formar una representación real completa del proceso físico de procesamiento, garantizando así la multi-fisicalidad, dinámica e integración del proceso de procesamiento del producto.

Combinado con la tecnología del DT, se pueden abordar problemas de optimización discretos a gran escala en el taller. Garantizar el control de calidad y costos del proceso de fabricación aditiva de componentes metálicos de la industria aeronáutica para asegurar que todas las piezas instaladas en cada aeronave cumplan con las condiciones de seguridad. Basado en los datos de función específica del proceso de visualización 3D y compatible con el uso de paneles de análisis de negocios basados en la web para ayudar a los operadores a tomar decisiones correctas. Integrando los recursos de fabricación virtuales correspondientes en el sistema de producción ciberfísica, se pueden empaquetar y definir una variedad de servicios de fabricación en un formato estandarizado.

Técnicas de DT

El DT proporciona una plataforma estandarizada para el desarrollo de la industria de la aviación inteligente, logra el intercambio consistente y sin interrupciones de información técnica y la interoperabilidad entre dominios, al mismo tiempo que reduce el riesgo de operación de aeronaves y mejora la seguridad y eficiencia de la operación de aeronaves. La construcción del modelo de DT de la industria de la aviación involucra tres tecnologías clave: la tecnología de fusión de datos, la tecnología de modelado de alta fidelidad multidimensional y multiscale, y la

fusión con la Nueva Tecnología de la Información (Nueva TI). Al mismo tiempo, considerando la estrecha relación entre la operación segura de aeronaves y los factores humanos, esta sección analiza la necesidad de la tecnología de interacción humano-máquina (HMI) de DT.

La fusión de los datos recopilados puede proporcionar una base para el modelado de alta fidelidad multidimensional y multiscale, la integración con la Nueva TI y el análisis de HMI. Al mismo tiempo, HMI y la Nueva TI pueden promover el modelado de alta fidelidad multidimensional y multiscale. Los cuatro elementos clave trabajan juntos para permitir que el DT de la industria de la aviación proporcione mejores servicios inteligentes.

Las tecnologías tradicionales de fusión de datos incluyen la fusión de probabilidad (por ejemplo, fusión bayesiana), la fusión de creencias basada en evidencia (por ejemplo, la teoría de Dempster-Shafer) y la fusión basada en conjuntos aproximados. Sin embargo, los datos de DT en la industria de la aviación tienen atributos de multi-fuentes y heterogeneidad. Es necesario garantizar la calidad de los datos, reducir la incertidumbre de los datos, unificar los estándares de datos, mejorar la disponibilidad de los datos, eliminar las islas de información de los datos y mejorar la capacidad de gestión de la información del DT. Desde la perspectiva computacional, la "fusión de datos e información" es la tecnología clave para impulsar un DT, que promueve el flujo de información desde datos sensoriales crudos hasta la comprensión avanzada y percepciones, y mejora la eficiencia y el efecto de aplicación de la fusión de datos mediante el uso de big data, aprendizaje automático y métodos de aprendizaje profundo.

Las ventajas de las operaciones de fusión se enumeran de la siguiente manera:

- La fusión de sensores puede proporcionar una mejor calidad de señal; y
- Tanto la fusión de modelos físicos como la fusión de modelos de datos pueden mejorar el rendimiento del modelo;
- La fusión de sensores y modelos físicos puede formar un modelo físico adaptable;
- La fusión de sensores y modelos de datos mejora la robustez del modelo impulsado por datos;
- La precisión de la predicción se mejora mediante la fusión del modelo de datos y el modelo físico;
- La integración de sensores, modelos físicos y modelos de datos permite a los gestores tomar decisiones más fiables.

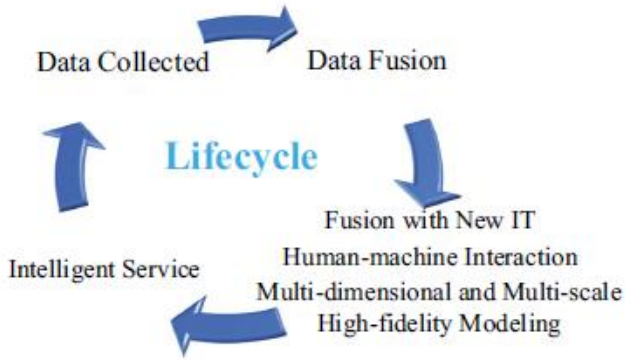

Figura 1: Ciclo conceptual de los algoritmos de inteligencia artificial

Las aeronaves se pueden dividir en cuatro niveles de escala: estructura, parte, subparte y componentes. Y las entidades físicas en diferentes niveles corresponden a los modelos de DT respectivos.

A través de la verificación de simulación y la optimización iterativa, se realizan respectivamente el DT de unidad, el DT de subsistema, el DT de sistema y el DT de sistemas complejos, y, finalmente, se logra el apoyo mutuo y la integración del sistema en DT de diferentes niveles.

Al mismo tiempo, la aeronave de DT completa la modelización virtual de entidades físicas desde cuatro dimensiones: geométrica (G), física (P), de comportamiento (B) y de reglas (R). Todas las dimensiones están relacionadas, combinadas e integradas para formar un modelo de espacio digital interconectado con alta fidelidad, que proporciona soporte técnico para la realización de simulaciones de fusión multidimensional de "geometría-física-comportamiento-reglas" con multidisciplina, multimodalidad y multiscale.

A través del establecimiento de la relación de asociación de los diversos modelos dimensionales, se forma un DT de aeronave virtual integral, y la operación y simulación visual del modelo se realizan en forma de representación tridimensional. En el mundo de la información, se mapean completamente las funciones y el rendimiento en tiempo real de cada nivel del sistema de la aeronave civil física, y se analiza el objeto de mantenimiento de la aeronave civil a nivel por nivel. Se puede modelar y simular con alta eficiencia y precisión, y el mantenimiento de los sistemas de aeronaves civiles se puede llevar a cabo a múltiples escalas.

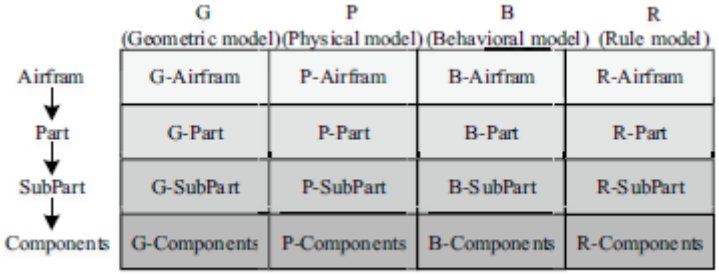

Figura 2: Escalas del mantenimiento de aeronaves

La integración de DT en la industria de la aviación y las nuevas tecnologías de la información (New IT) contribuyen a la integración profunda de todos los elementos, procesos completos y datos comerciales completos de las entidades físicas y realizan la construcción precisa del modelo de ciclo de vida completo de los productos físicos de la industria de la aviación, la percepción completa y real de la escena física y la realización de la interacción dinámica en tiempo real y servicios inteligentes de productos de la industria de la aviación, personal relacionado y entornos relacionados.

Combination	Benefits
DT + IoT	IoT provides technical support for the overall perception of the physical world and helps the real-time, reliable, and efficient transmission of twin data through wired or wireless networks.
DT+AR/VR/MR	To provide multi-dimensional and multi-time scale high-fidelity digital mapping for aviation physical entities. Realize the integration of visualization and virtual reality, make the virtual model truly present a physical entity, and enhance the functions of the physical entity.
DT + Edge computing	Edge computing technology can filter, specify and process some data collected from the physical world in real-time on the edge side, thus realizing the immediate decision-making, quick response, and timely execution of users. Cloud-edge data collaborative processing for different needs is realized, which improves data processing efficiency, reduces cloud data load and data transmission delay, and provides a guarantee for the real-time performance of DT.
DT+ cloud computing	Cloud computing can dynamically meet the different computing, storage, and operation requirements of aviation DT.
DT +5G	The 5G communication technology has the features of high speed, high-capacity, low delay, and high reliability, which can meet the data transmission requirements of the DT in the aviation industry, thus better promoting the application of the DT in the aviation industry.
DT + Big Data	Big data can extract more valuable information from the massive data generated by the aviation industry DT at high speed, to better explain and predict the result and process of the aviation industry DT.
DT + Blockchain	Blockchain can provide a reliable guarantee for the security of the aviation industry DT, and ensure that twin data can be retained and tracked throughout the whole process.
DT + AI	AI automatically performs data preparation, analysis, and fusion through intelligent matching of the best algorithm. Perform in-depth knowledge mining on twin data to generate various types of services. Significantly enhance the value of data and the responsiveness and accuracy of services.

Figura 3: Combinaciones de gemelo digital e inteligencia artificial aplicados en la industria aeronáutica

Los seres humanos desempeñan un papel vital en todo el ciclo de vida de una aeronave, especialmente en la operación y el mantenimiento, donde a menudo se producen comportamientos de emergencia impredecibles. El DT puede aliviar este problema, lograr un modo de interacción más flexible, analizar escenarios complejos de existencia humana, mejorar la eficiencia de la aeronave y la seguridad operativa, y respaldar una ICI (Interacción Persona-Computadora) más efectiva y segura.

Revisión de las herramientas de producción aeronáutica

La industria aeronáutica utiliza una variedad de herramientas y tecnologías para los gemelos digitales y la optimización de recursos. Estas herramientas ayudan a las empresas a mejorar la eficiencia operativa, reducir costos y optimizar el rendimiento de aeronaves y sistemas:

- Software de Gemelos Digitales (Digital Twin Software): Estos son sistemas informáticos que crean una réplica virtual de una aeronave, incluyendo sus sistemas, componentes y subsistemas. Algunos ejemplos incluyen Siemens NX, ANSYS Twin Builder, Dassault Systèmes CATIA, y varios otros programas de modelado y simulación.
- Simulación y Modelado: Se utilizan herramientas de simulación como MATLAB, Simulink, y programas de dinámica de fluidos computacional (CFD) para analizar el comportamiento de las aeronaves en diferentes condiciones, lo que permite optimizar su diseño y funcionamiento.

- IoT y Sensores: Los sensores y dispositivos IoT (Internet de las cosas) se utilizan para recopilar datos en tiempo real de aeronaves y componentes. Esto permite el monitoreo constante de su estado y el rendimiento en vuelo, lo que facilita la detección temprana de problemas y la toma de decisiones informadas.
- Big Data y Analytics: Se utilizan herramientas de análisis de datos para procesar y analizar grandes conjuntos de datos generados por sensores y otros sistemas. Esto ayuda a identificar patrones, tendencias y oportunidades de optimización.
- Machine Learning e Inteligencia Artificial: El aprendizaje automático se utiliza para desarrollar algoritmos que pueden predecir el desgaste de componentes, identificar anomalías y optimizar la operación de aeronaves.
- Realidad Virtual (RV) y Realidad Aumentada (RA): Estas tecnologías se utilizan para visualizar los gemelos digitales en entornos 3D, lo que facilita la colaboración y la toma de decisiones.
- Gestión de Mantenimiento Asistido por Computadora (CMMS): Las soluciones CMMS ayudan a gestionar el mantenimiento de aeronaves al rastrear el historial de mantenimiento, programar tareas y gestionar piezas de repuesto.
- Plataformas de Gestión de Datos: Se utilizan sistemas de gestión de datos para almacenar, organizar y acceder a datos críticos relacionados con las aeronaves y su rendimiento.
- Automatización y Robótica: Los sistemas automatizados y robots se utilizan en la fabricación y el mantenimiento de aeronaves para aumentar la eficiencia y reducir los errores humanos.
- Herramientas de Planificación y Programación: Software de planificación de recursos empresariales (ERP) y herramientas de programación ayudan a gestionar la producción, el mantenimiento y la logística en la industria aeronáutica.

Estas herramientas y tecnologías se utilizan en conjunto para crear y mantener gemelos digitales de aeronaves, lo que permite a las empresas aeroespaciales optimizar recursos, mejorar la seguridad y la eficiencia, y reducir los costos operativos.

Los gemelos digitales son una nueva faceta de la Industria 4.0 que surgen debido a las nuevas necesidades de las industrias y el avance hacia un mundo cada vez más digitalizado.

Un gemelo digital es una copia virtual de un producto o un proceso que permite simularlo, predecirlo y optimizarlo, antes de realizar cualquier actuación sobre el propio activo físico. De este modo, el mundo de la ingeniería es capaz de examinar sus diseños, probar su funcionamiento, detectar errores e implementar posibles modificaciones, antes de materializar dichos modelos en la vida real.

Una de las premisas básicas de un gemelo digital es su grado de similitud con el producto o proceso que representan, es por ello que los gemelos digitales no tienen un carácter estático, sino que se van actualizando con la información obtenida del producto/proceso real, para adaptarse a posibles nuevas condiciones de funcionamiento o cambios que se puedan producir a lo largo del tiempo. De este modo se genera un circuito cerrado de retroalimentación entre el modelo virtual y el modelo real, que permite a las empresas realizar una optimización continua de sus productos y procesos.

Los gemelos digitales son tecnologías muy avanzadas que presentan diferencias en función del uso o la aplicación que se les dé. En ese sentido pueden diferenciarse tres grandes grupos o tipos de gemelos digitales:

- Gemelo digital de producto: Son creados cuando un nuevo producto es definido y diseñado. Este tipo de gemelos permiten la visualización, simulación y validación de los productos, tanto su parte mecánica, eléctrica o electrónica. Gracias a todo ello se evita la necesidad de utilizar numerosos prototipos, se reduce el tiempo de desarrollo total, se mejora la calidad del producto final fabricado y se pueden realizar acciones más rápidas en respuesta a demandas de los clientes.
- Gemelos digitales de rendimiento: Este tipo de gemelos digitales capturan grandes cantidades de información en tiempo real que se genera durante el funcionamiento de los procesos o los productos. Dicha información se analiza extrapolándola al gemelo digital y esto se traduce a su vez en nuevos datos y estadísticas que pueden realimentarse al modelo físico para obtener un estado de funcionamiento óptimo.
- Gemelo digital de proceso: Un gemelo digital de una máquina ayuda a validar la eficacia con la que funcionará esa máquina cuando se integre en el proceso de fabricación en la planta real, incluso antes de iniciar la producción. Este tipo de gemelos digitales no tiene porqué limitarse exclusivamente a una máquina del sistema productivo, sino que por medio de la interconexión de los gemelos digitales de cada una de las máquinas se puede generar un gemelo digital de todo el proceso productivo. De este modo es posible simular el funcionamiento de líneas completas de producción, probar el desempeño de las máquinas a partir de la programación desarrollada, depurar fallos y en definitiva realizar una puesta en marcha virtual de la línea productiva y crear una metodología de producción que mantenga su eficiencia en diversas situaciones.

Las principales aplicaciones que se le suele dar a un gemelo digital son:

- El mantenimiento predictivo. Con datos en tiempo real de la planta se pueden hacer predicciones para optimizar las planificaciones de producción y de mantenimientos.
- Control de procesos. Conociendo muy bien los parámetros de salida que se buscan, se puede hacer un gemelo que sepa qué entradas son necesarias para obtener el producto deseado a un coste bajo.
- Control de calidad. Un control del proceso que optimice, no solo las entradas, sino todos los parámetros del proceso, para obtener la mejor calidad posible a los costes más bajos.

En la literatura se han encontrado artículos que examinan el desarrollo de la definición de DT y comparan sus conceptos con Internet de las cosas, sistemas ciberfísicos e hilo digital, entre otros. También, presentan la línea de tiempo de investigación del DT en la aviación en los últimos 10 años por parte de autoridades como el Laboratorio de Investigación de la Fuerza Aérea y la Administración Nacional de Aeronáutica y del Espacio. Hay artículos que revisan el estado de la investigación más avanzada del DT en todo el ciclo de vida del sistema de aviación y concluyen que el DT en la aviación se utiliza principalmente en la fabricación y el mantenimiento, y debe prestar atención a la aplicación del DT en vehículos aéreos no tripulados (UAV).

Finalmente, se resume que la fusión de datos, la modelización de alta fidelidad, la integración con la Nueva Tecnología de la Información y la interacción entre humanos y máquinas son las tecnologías clave del DT en la aviación.

Las posibles direcciones de investigación futura podrían ser el control de riesgos con el gemelo de proceso digital, el gemelo digital interactivo y el aprendizaje de datos del DT.

Con un algoritmo de inteligencia artificial (IA) y datos de un software de simulación de motores de avión en operación, se puede conseguir identificar diferentes tendencias que afectan directamente a la propia operación del motor y de las que se extraen conclusiones con suficiente peso para ser tenidas en cuenta.

- Predicción de fallos: Utilizando técnicas de aprendizaje supervisado, puedes entrenar algoritmos para predecir posibles fallos en los motores. Esto implica analizar patrones en los datos históricos y alertar sobre condiciones que podrían llevar a un fallo.
- Optimización del rendimiento: La IA puede ayudar a optimizar el rendimiento de los motores al analizar datos en tiempo real y ajustar parámetros para mejorar la eficiencia y reducir el desgaste.
- Mantenimiento predictivo: Implementar un sistema de mantenimiento predictivo basado en la IA puede ayudar a prever cuándo es probable que un componente del motor necesite ser reemplazado o reparado, minimizando el tiempo de inactividad.
- Análisis de datos en tiempo real: La IA puede procesar grandes cantidades de datos en tiempo real para identificar patrones, tendencias y anomalías, lo que permite una toma de decisiones más rápida y precisa.
- Optimización de combustible: Analizar el rendimiento del motor y ajustar automáticamente los parámetros de operación para maximizar la eficiencia del combustible.
- Detección de anomalías: Implementar algoritmos de detección de anomalías para identificar cualquier comportamiento inusual en los motores que pueda indicar problemas potenciales.
- Simulación y diseño de nuevos componentes: Simular y diseñar nuevos componentes de motores de avión, optimizando su rendimiento y eficiencia.
- Análisis de vibraciones: Analizar datos de vibración del motor para detectar posibles problemas mecánicos antes de que se conviertan en fallas críticas.
- Mejora de la eficiencia operativa: La IA puede analizar patrones operativos y sugerir ajustes para mejorar la eficiencia operativa, reducir costos y aumentar la vida útil del motor.
- Cumplimiento de normativas y seguridad: Monitorear y garantizar el cumplimiento de normativas de seguridad y regulaciones, así como para prevenir posibles riesgos.

La aplicación específica dependerá de los objetivos que se quieran conseguir y de los datos disponibles. Es importante asegurarse de tener conjuntos de datos de calidad y trabajar en estrecha colaboración con expertos en la industria para garantizar resultados precisos y útiles.

Identificar posibles fallos en la operación de un motor de avión a partir de los parámetros típicos de comportamiento implica monitorear y analizar una variedad de variables para detectar cualquier desviación significativa de las condiciones normales.

Las variables principales son:

- Temperatura del motor: Supervisar la temperatura del motor es crítico. Desviaciones inusuales podrían indicar problemas con el sistema de refrigeración, fallos en el sistema de combustible o desgaste excesivo de componentes.
- Presión de aceite y combustible: Mantener un ojo en la presión del aceite y del combustible es esencial para detectar posibles fugas o problemas en los sistemas de lubricación y combustible.

- Velocidad del motor: Variaciones inesperadas en la velocidad del motor pueden señalar problemas con el sistema de control, el combustible o la mecánica interna.
- Consumo de Combustible: Cambios inusuales en el consumo de combustible pueden indicar problemas en la eficiencia del motor, como obstrucciones en los filtros de combustible o problemas de inyección.
- Vibraciones: El monitoreo de las vibraciones puede ayudar a identificar desequilibrios en el motor, desgaste de componentes o problemas en la alineación.
- Niveles de emisión: Analizar las emisiones del motor puede revelar problemas en la combustión, como combustión incompleta o exceso de emisiones contaminantes.
- Presión del compresor: Controlar la presión en el compresor es esencial para detectar problemas en esta etapa del proceso, como obstrucciones o desgaste en las palas.
- Sensores de salud del motor: Utilizar sensores específicos de salud del motor que monitorean constantemente diversos parámetros y emiten alertas en caso de desviaciones significativas.
- Análisis espectral de sonido: Realizar análisis espectral del sonido del motor puede ayudar a identificar frecuencias inusuales que podrían indicar problemas en componentes específicos.
- Sistemas de diagnóstico basados en IA: Implementar algoritmos de inteligencia artificial para analizar datos en tiempo real y detectar patrones que podrían indicar fallos potenciales.
- Histórico de mantenimiento: Revisar el historial de mantenimiento y comparar datos actuales con eventos anteriores para identificar posibles patrones de fallos.

Es importante establecer límites y umbrales para cada parámetro, y cualquier desviación significativa debería activar alertas o procedimientos de mantenimiento preventivo. Además, la colaboración con expertos en la industria de la aviación y la utilización de sistemas de monitoreo avanzados son clave para garantizar la seguridad y el rendimiento del motor.

Una vez analizadas todas las posibilidades que ofrecen los distintos algoritmos de inteligencia artificial en función de los objetivos que se quieren conseguir en este proyecto, el siguiente paso es seleccionar el algoritmo que mejor se adapte a las necesidades identificadas que rigen las bases de este proyecto.

El objetivo que se pretende alcanzar en este proyecto al realizar una aproximación a los algoritmos de inteligencia artificial es identificar lógicas de comportamiento entre los parámetros de los distintos subconjuntos de datos y que a su vez definen las prestaciones de un motor en base a todas las iteraciones realizadas en el código de Python, para posteriormente calificar dichas iteraciones en función de cómo se comporten esas relaciones entre los distintos parámetros.

El siguiente paso es definir y desarrollar un sistema de inteligencia artificial que se adapte de la mejor manera a los conjuntos de datos obtenidos de la ejecución en distintas iteraciones del código de Python que define las prestaciones de este motor y que permita la obtención tanto de nuevos valores como de conclusiones sobre las relaciones existentes entre todos estos parámetros a través del aprendizaje automático.

4. Descripción de la planta de potencia

Los motores de aviación tipo turbofán son una generación de motores de reacción que ha reemplazado a los turborreactores. También se suelen llamar turborreactores de doble flujo. Se caracterizan por disponer de un ventilador, fan, en la parte frontal del motor. El aire entrante se divide en dos caminos: flujo de aire primario y flujo secundario o flujo derivado. El flujo primario penetra al núcleo del motor —compresores y turbinas— y el flujo secundario se deriva a un conducto anular exterior y concéntrico con el núcleo. Los turbofanes tienen varias ventajas respecto a los turborreactores: consumen menos combustible, lo que los hace más económicos, producen menor contaminación y reducen el ruido ambiental.

El índice de derivación, también llamado relación de derivación, es el cociente de la masa del flujo secundario entre la del primario. Se obtiene dividiendo las secciones transversales de entrada a sus respectivos conductos. En aviones civiles suele interesar mantener índices de derivación altos ya que disminuyen el ruido, la contaminación, el consumo específico de combustible y aumentan el rendimiento. Sin embargo, aumentar el flujo secundario reduce el empuje específico a velocidades cercanas o superiores a las del sonido, por lo que para aeronaves militares supersónicas se utilizan turbofanes de bajo índice de derivación.

- Turbofán de bajo índice de derivación: su índice de derivación está entre 0,2 y 2.5. Fue el primero en desarrollarse y fue ampliamente utilizado en la aviación civil hasta que se sustituyó por los de alta derivación. Es habitual que exista un carenado a lo largo de todo el conducto del flujo secundario hasta la tobera del motor. Operan de forma óptima entre Mach 1 y Mach 2, por lo que en la actualidad se utilizan principalmente en aviación militar. Sin embargo, algunas aeronaves comerciales siguen haciendo uso de ellos, como el MD-83 con el Pratt & Whitney JT8D y el Fokker 100 con el Rolls-Royce Tay.

- Turbofán de alto índice de derivación: poseen un índice de derivación considerablemente superior (mayor que 5). Estos motores representan una generación más moderna, especialmente usados en aeronaves civiles. La mayor parte del empuje, alrededor de un 80 %, proviene del primer compresor o ventilador, que tiene una función básicamente propulsiva, similar a una hélice. Está situado en la parte delantera del motor y movido por un eje conectado a la última etapa de la turbina. El restante 20 % de la fuerza impulsora proviene de los gases de escape de la tobera. Los más recientes tienen un índice de derivación en torno a 10, como los que usan el Boeing 787 o el Airbus 380. En motores con relaciones de derivación muy altas, sobre todo junto a relaciones de compresión también elevadas, aparecen problemas de diseño debido a que el ventilador debe girar a una velocidad muy inferior a los compresores y turbinas de alta presión. Por este motivo, suelen incorporar dos ejes concéntricos que permiten ajustar las velocidades de rotación de forma independiente.

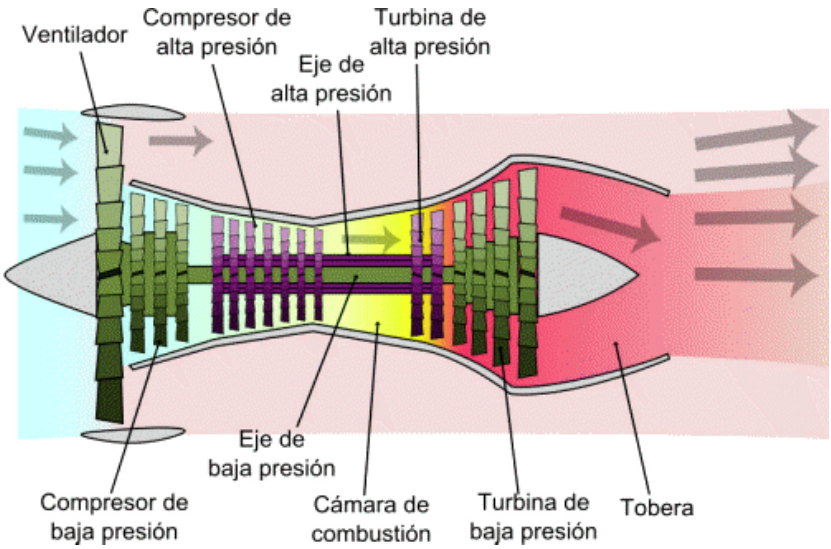

Figura 4: Estructura básica de un motor turbofán de doble flujo

Un motor de aviación, en este caso un turbofán de 2 ejes, está compuesto principalmente por las secciones fría y caliente. Esto es, la estructura anterior a la cámara de combustión (fría) y la estructura posterior a la cámara de combustión (caliente). La zona fría del motor está compuesta por el fan o ventilador, la LPC ('Low Pressure Compressor') y la HPC ('High Pressure Compressor'). Por su parte, la zona caliente del motor está compuesta por la HPT ('High Pressure Turbine'), la LPT ('Low Pressure Turbine') y la tobera de salida, si bien en esta última las temperaturas de los gases son del orden de la mitad que las que se alcanzan en la salida de la cámara de combustión.

A continuación, se detallan las principales funciones de cada subsección del motor:

- FAN (Fan - Ventilador): El ventilador es la primera etapa del motor y se encuentra en la parte frontal. Su función principal es tomar una gran cantidad de aire del entorno y comprimirlo ligeramente antes de enviarlo a la siguiente etapa, llamada el Compresor de Baja Presión (LPC). El ventilador suele estar conectado a una parte del motor llamada la "fan case" (carcasa del ventilador) y es responsable de generar la mayor parte del empuje en motores de doble flujo.
- LPC (Low-Pressure Compressor - Compresor de Baja Presión): El LPC es la siguiente etapa del motor después del ventilador. Su función es comprimir aún más el aire que entra en el motor antes de enviarlo a la siguiente etapa, que es el Compresor de Alta Presión (HPC). El LPC consta de una serie de álabes o etapas de compresión (booster) que aumentan gradualmente la presión del aire.
- HPC (High-Pressure Compressor - Compresor de Alta Presión): El HPC sigue al LPC y es responsable de aumentar aún más la presión del aire. Este compresor consta de varias etapas de compresión y funciona a una presión más alta que el LPC. El aire se comprime aún más a medida que pasa por el HPC antes de pasar a la etapa de la cámara de combustión.
- Cámara de combustión (Combustion Chamber): La cámara de combustión en un motor de turbina de gas es donde se mezcla el aire comprimido con combustible y se enciende para crear gases calientes de alta energía. Estos gases impulsan las etapas de la turbina, generando potencia. La cámara también incluye sistemas de enfriamiento para proteger las partes internas y dirige los gases de escape hacia la turbina.

- HPT (High-Pressure Turbine - Turbina de Alta Presión): Después de la cámara de combustión, los gases calientes y de alta presión pasan a través de la HPT, que es una etapa de turbina. La HPT está conectada al mismo eje que el HPC y aprovecha la energía cinética de los gases calientes para hacer girar el eje. Este giro impulsa tanto el HPC como el LPC, lo que ayuda a mantener el proceso de compresión del aire.
- LPT (Low-Pressure Turbine - Turbina de Baja Presión): Después de pasar por la HPT, los gases de escape de menor presión y temperatura pasan a través de la LPT, que es otra etapa de turbina. La LPT está conectada al eje de la turbina de baja presión y aprovecha la energía restante en los gases de escape para hacer girar el eje y proporcionar potencia adicional, especialmente para el ventilador en motores de doble flujo.

Estas etapas son características de los motores de turbina de gas y son típicas en motores de aviación, donde el proceso de compresión y expansión de los gases es fundamental para la generación de potencia y la propulsión de la aeronave. Cada etapa cumple una función específica en este ciclo termodinámico.

N1 y N2 son términos comunes que se utilizan para describir las velocidades de rotación de los dos ejes principales en un motor turbofán de dos ejes. Estas dos velocidades son fundamentales para el funcionamiento y control del motor. N1 se refiere a la velocidad de rotación del eje de baja presión en el motor turbofan de dos ejes. El eje de baja presión consiste en una serie de rotores y álabes que comprimen el aire de admisión antes de que entre en la cámara de combustión. N1 es esencialmente la velocidad a la que giran estos componentes en el eje de baja presión. Esta velocidad se mide en revoluciones por minuto (RPM). N2, por otro lado, se refiere a la velocidad de rotación del eje de alta presión en el motor. El eje de alta presión está compuesto por otro conjunto de rotores y álabes que comprimen aún más el aire después de que ha pasado por el eje de baja presión. N2 representa la velocidad a la que giran estos componentes en el eje de alta presión, y también se mide en revoluciones por minuto (RPM).

El control de las velocidades de N1 y N2 es fundamental para el rendimiento y la operación segura del motor turbofán. Los pilotos y el sistema de control del motor pueden ajustar estas velocidades para cambiar el rendimiento del motor, como la potencia, la velocidad y la eficiencia. A menudo, N1 se utiliza para controlar la potencia del motor, mientras que N2 se utiliza para mantener una velocidad de giro constante y garantizar una operación segura dentro de los límites del motor.

A continuación, se muestra una imagen de los componentes o subsecciones que definen la estructura de un turbofán de 2 ejes:

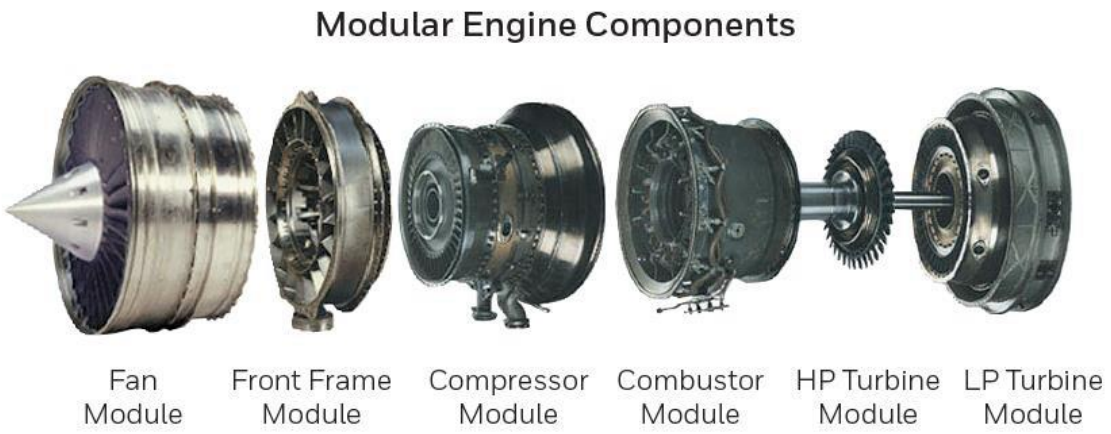

Figura 5: Componentes o submódulos de un motor turbofán de doble flujo

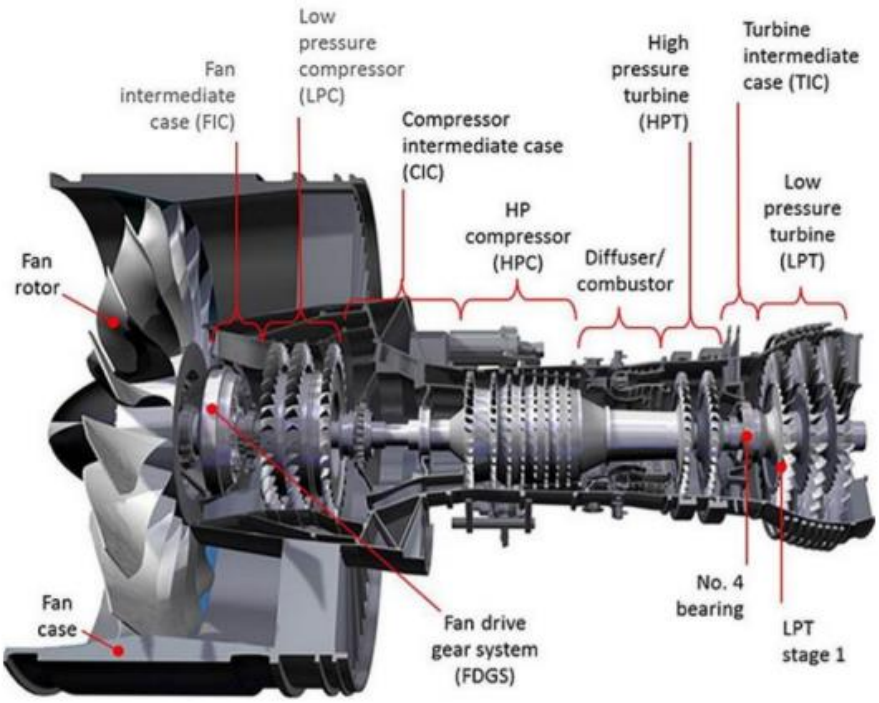

Figura 6: Elementos clave del funcionamiento de un motor turbofán de doble flujo

Debido a la gran complejidad se estas máquinas, los motores de aviación están equipados con una variedad de sensores y medidores para medir y monitorear sus prestaciones y condiciones de funcionamiento. Esto es fundamental en la industria debido a la imperante necesidad de controlar en todo momento cuáles son las prestaciones reales de los motores sin tener que bajarlos del avión y por tanto parar la actividad de las aerolíneas, los principales clientes de los fabricantes de avión.

Los programas de mantenimiento de motores, como el Programa de Mantenimiento por Hora de Vuelo (Power by the Hour - PBH), son acuerdos en los que los operadores de aeronaves pueden alquilar motores o componentes de motor y pagar una tarifa fija por hora de vuelo o una tarifa mensual. Estos programas suelen incluir el mantenimiento, las reparaciones y el reemplazo de piezas, lo que permite a las aerolíneas y operadores de aeronaves planificar mejor sus costos y garantizar un rendimiento confiable de los motores.

Esta nueva tendencia, radicalmente opuesta a la que se ha dado siempre en la industria, donde las aerolíneas compran tanto el avión como los motores en propiedad y se tienen que hacer cargo de su mantenimiento, obliga a los fabricantes a controlar de manera estricta las prestaciones de sus motores en todo momento, así como optimizar el diseño para reducir los costes del mantenimiento no programado. Por tanto, estos últimos años han dejado de manifiesto la necesidad de tener herramientas ofimáticas potentes que permitan conocer y controlar las prestaciones de cada motor para anticipar fallas y corregir cuanto antes en el proceso de diseño posibles errores de concepto, o bien implementar mejoras que consigan alargar la vida útil de estos motores sin que esto suponga un detrimento en las prestaciones.

En el funcionamiento de un turbofán de dos ejes, hay varios rendimientos y parámetros clave que se deben medir y monitorear para garantizar un funcionamiento eficiente y seguro de la aeronave. Algunos de los rendimientos importantes incluyen:

- Empuje: El empuje es uno de los parámetros más críticos. Se mide en libras de fuerza o newtons y representa la fuerza de empuje que el motor genera para propulsar la aeronave hacia adelante. El empuje debe estar en línea con las necesidades de la aeronave para el despegue, el crucero y el aterrizaje.
- Consumo de combustible: La eficiencia del motor se mide a menudo mediante la relación entre el empuje y el consumo de combustible. Los motores más eficientes consumen menos combustible para generar la misma cantidad de empuje.
- Eficiencia de compresión: La eficiencia del proceso de compresión en las etapas del compresor es importante. Una alta eficiencia de compresión garantiza que se alcance la presión de combustión necesaria con el menor consumo de energía.
- Eficiencia de la cámara de combustión: La eficiencia de la combustión es crítica para maximizar la conversión de la energía del combustible en energía cinética en los gases de escape. Una combustión incompleta puede reducir la eficiencia y aumentar las emisiones.
- Temperaturas: Las temperaturas son fundamentales para medir el rendimiento. Esto incluye la temperatura del aire de admisión, la temperatura de la cámara de combustión y las temperaturas de los gases de escape. Mantener estas temperaturas dentro de los límites especificados es crucial para la integridad del motor y el rendimiento.
- Presiones: Las presiones en diferentes puntos del motor, como la entrada, el compresor y la salida, se deben medir para garantizar un flujo de aire adecuado y una compresión eficiente.
- RPM (Revoluciones por minuto): La velocidad de rotación del eje de la turbina y el compresor es esencial para controlar el rendimiento y garantizar que estén dentro de los límites seguros.

- Vibraciones: La detección de vibraciones anormales en el motor es importante para identificar posibles desequilibrios o problemas en los componentes.
- Emisiones de gases contaminantes: Monitorear las emisiones de gases contaminantes, como NOx, CO2 y CO, es esencial para cumplir con las regulaciones medioambientales.
- Rendimiento de la turbina de baja presión (LPT): Esto incluye la eficiencia de la LPT y su capacidad para aprovechar la energía restante en los gases de escape para impulsar el ventilador.
- Rendimiento del ventilador: En un turbofán de dos ejes, el rendimiento del ventilador, que está conectado a la turbina de baja presión, es crucial para generar un empuje adicional y mantener una relación aire-combustible adecuada.

El monitoreo constante de estos parámetros y el cumplimiento de los límites y especificaciones del fabricante son esenciales para garantizar un funcionamiento seguro y eficiente del motor de un turbofán de dos ejes. Los datos recopilados permiten a los operadores y pilotos tomar decisiones informadas y realizar el mantenimiento necesario para mantener el motor en óptimas condiciones.

5. Parametrización del ciclo termodinámico

Introducidos el modelo de cálculo, las hipótesis y las compatibilidades del motor, es fundamental presentar las ecuaciones que se han implementado en los algoritmos numéricos. Estas ecuaciones no solo definen las variables en cada punto del ciclo, sino que junto con las condiciones de vuelo y los parámetros tecnológicos seleccionados se parametriza el ciclo, que en última instancia es lo que añade valor a la aplicación, pues su finalidad última radica en poder variar todos los parámetros en función del diseño al que se quiera llegar. La consideración más importante de las ecuaciones de cara a su posterior implementación en el código es que se trata de ecuaciones específicas, pues no aparece el gasto de aire en ninguna de ellas. Esto se hace así puesto que el motor todavía no ha sido dimensionado, por lo que se desconoce el tamaño de sus turbomáquinas y el gasto que las atravesará. Una vez que se elija un empuje en el punto de diseño, el empuje específico en dicho punto determinará unívocamente el gasto de aire que tendrá el motor.

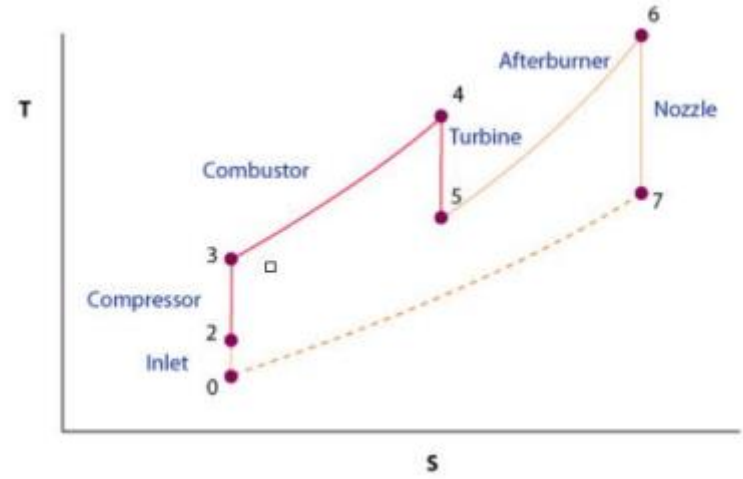

Figura 7: Ciclo termodinámico del funcionamiento de un motor turbofán de doble flujo

El principal cambio que el modelo introduce en las ecuaciones que se conocen del ciclo es efectivamente el hecho de que las propiedades del fluido son ahora dependientes con la temperatura. De esta forma, muchas de las ecuaciones no podrán expresar la entalpía constante con T, por lo que deberán ser reescritas siguiendo la ecuación de Gibbs-Duhem para relacionar entalpía, temperatura, presión y densidad en cada punto del ciclo:

$$\mathrm{Tds} = \mathrm{dh} - \mathrm{vdp}$$

La ecuación puede ser reescrita como sigue:

$$ds = \frac{dh}{T} - \frac{vdp}{T}$$

Siendo En los procesos isoentrópicos se puede transformar la ecuación de la siguiente manera:

$$ds = 0, \qquad \frac{dh}{T} = \frac{Rdp}{T} \rightarrow \frac{C_p(T)}{dT} = \frac{Rdp}{T}$$

Con esta relación se podrá relacionar presión y temperatura en dos puntos del ciclo de un mismo proceso isoentrópico.

A continuación, se procederá a desarrollar cada una de las ecuaciones que determinan la evolución termodinámica del fluido en cada etapa del motor:

Difusor

Las condiciones a la entrada son: T_0, P_0, M_0, π_d

El difusor eleva la presión del fluido desde las condiciones 0 a 1, produciéndose una pérdida de carga dada por π_d. Las entalpías de remanso entre ambos puntos se relacionan a través de la velocidad de vuelo U:

$$h_{00} - h_0 = \int_{T_0}^{T_{00}} C_P(T)dT = \frac{U^2}{2}$$

siendo $T_{00}=T_{01}$

A continuación, hay que relacionar la velocidad del sonido con la presión y la densidad en un medio isoentrópico:

$$a^2 = \frac{dp}{d\rho}$$

Primero, se deriva la ecuación de los gases ideales respecto a la densidad:

$$dp = dpRT + \rho R dt$$

$$dT = \frac{1}{R\rho}(dp - d\rho RT) = \frac{d\rho}{p} - \frac{d\rho}{\rho}$$

Utilizando ahora la ecuación de Gibbs-Duhem se llega a:

$$C_p(T)\left(\frac{dp}{p} - \frac{d\rho}{\rho}\right) = \frac{Rdp}{p} \rightarrow \frac{C_p(T)dT}{T} = \frac{Rdp}{p}$$

$$\frac{dp}{p}\big(C_p(T) - R\big) = \frac{d\rho}{\rho}C_p(T)$$

$$a^2 = \frac{dp}{d\rho} = \frac{p}{\rho}\frac{C_p(T)}{C_p(T) - R} = RT\frac{C_p(T)}{C_p(T) - R}$$

De esta manera se puede relacionar la entalpía en 1 y 0 a través del número de Mach:

$$\int_{T_0}^{T_{01}} C_p(T)dT = \frac{U^2}{2} = \frac{1}{2}{M_0}^2{a_0}^2 = \frac{1}{2}{M_0}^2 RT_0\frac{C_p(T)}{C_p(T) - R}$$

De esta ecuación se obtiene T_{01}. De la caída de presión de remanso del difusor se calcula la presión de remanso a la salida de este:

$$p_{01} = \pi_d p_0$$

Compresor

Las condiciones a la entrada son: T_{02}, p_{02}, π_c, e_{pc}

El compresor eleva la presión del fluido desde las condiciones de 2 a 3 mediante un proceso de aceleración y desaceleración del fluido en las etapas de rótor y estátor.

Partiendo de la ecuación de Gibbs-Duhem diferencial con el rendimiento politrópico de compresor:

$$e_{pc}\frac{C_p(T)}{T} = \frac{Rdp}{T}$$

Se integra la ecuación entre las condiciones 1 y 2:

$$\int_{T_{02}}^{T_{03}} \frac{C_p(T)}{T} dT = R\int_{P_{02}}^{P_{03}} \frac{dp}{p}$$

Se resuelve la ecuación obteniendo T_{02}:

$$\int_{T_{02}}^{T_{03}} \frac{C_p(T)}{T} dT = Rlog(\pi_c)$$

De esta forma ya se puede calcular el trabajo específico del compresor:

$$W_{ec} = \int_{T_{02}}^{T_{03}} C_p(T)dT$$

Por otra parte, se despeja la p_{02} sin más que hacer: $p_{02} = \pi c p_{01}$. Se procede ahora a calcular el rendimiento isoentrópico del compresor, para lo cual se necesita la temperatura T_{02s}. De la ecuación de Gibbs-Duhem que relaciona el punto 01 con 02s del proceso isoentrópico:

$$\int_{T_{02}}^{T_{03s}} \frac{C_p(T)}{T} dT = Rlog(\left(\frac{P_{03}}{P_{02}}\right))$$

Una vez calculada se procede a expresar el rendimiento isoentrópico en términos de la entalpía de remanso:

$$\eta_{isoc} = \frac{\int_{T_{02}}^{T_{03}} C_p(T)dT}{\int_{T_{02}}^{T_{03}} C_{p_g}(T)dT}$$

$$\pi_c = \frac{p_{03}}{p_{02}}$$

Finalmente, el gasto de sangrado se produce en la última etapa del compresor axial. Este gasto es proporcional a la temperatura fin de combustión.

Cámara de combustión

Condiciones a la entrada: T_{03}, p_{03}, T_{04}

La combustión es el proceso donde a partir de unos reactivos se obtienen unos productos y un salto entálpico que se traduce en energía aprovechable por la turbina para mover al compresor, así como para acelerar los gases de salida en la tobera.

Algunas de las características de la combustión que se han contemplado para su implementación numérica son:

- La combustión no es completa; existe un rendimiento de combustión dado por ηcc
- No se ha estudiado la cinética de la combustión, en cuyo caso los resultados obtenidos serían mucho más precisos.
- La proporción aire-fuel es siempre tal que se obtienen las mismas especies químicas de productos, no apareciendo nunca: CO, OH, NO, O, H. Estas especies aparecerían si la combustión fuera pobre, fenómeno que en principio no se ha contemplado, por lo que siempre se tendrán los compuestos mencionados en el análisis de fluido empleado.
- La combustión produce una pérdida de carga dada por un factor que en principio el usuario que utilice la herramienta podrá elegir. En la teoría este factor es proporcional a la relación entre el gasto de aire que suministra el compresor y el gasto de aire máximo que se podría tener.
- Los compuestos de $N2$ y Ar son inertes y por tanto la cantidad de estos es la misma tras la combustión.
- El dosado estequiométrico es fijo y depende siempre de la composición química de la mezcla.

Posteriormente tiene lugar el balance energético de la combustión, del que se despeja el dosado:

$$f_1 = \frac{m}{m_f - m_s}$$

$$\eta_{cc} H_p f_1 = (1 + f_1) \int_{T_{ref}}^{T_{04}} C_{p_g}(T) dT - \int_{T_{ref}}^{T_{04}} C_{p_a}(T) dT$$

La temperatura fin de combustión $T03$ es conocida, pues el usuario la elegirá libremente o vendrá asociada a un motor según su nivel tecnológico. Por último, la presión al final de la combustión vendrá dada por la caída de presión de remanso en la misma:

$$\Delta_{p_{cc}} = \frac{p_{04}}{p_{03}}$$

Turbina

Condiciones a la entrada: T_{05}, p_{05}, W_{ec}, e_{pt}

Como se ha comentado anteriormente la turbina aprovecha el salto entálpico producido en la combustión para que, a través de un proceso de expansión, se extraiga la energía del fluido para alimentar el compresor y otros equipos auxiliares. En primer lugar, se vuelve a utilizar la ecuación de Gibbs-Duhem para obtener la temperatura a la salida de la turbina, T_{05}:

$$\int_{T_{04}}^{T_{05}} \frac{C_p(T)}{T} dT = R log\left(\frac{p_{05}}{p_{04}}\right)$$

Ahora se obtiene la temperatura $T05s$ para calcular el rendimiento isoentrópico de la turbina. Para ello se vuelve a escribir la ecuación de Gibbs-Duhem en términos del proceso 03-04s:

$$\int_{T_{04}}^{T_{05s}} \frac{C_{p_g}(T)}{T} dT = R log\left(\frac{p_{05}}{p_{04}}\right)$$

Una vez que se tiene $T05s$ se está en condiciones de calcular el rendimiento isoentrópico y la relación de expansión de turbina:

$$\eta_{isot} = \frac{\int_{T_{04}}^{T_{05}} C_p(T)dT}{\int_{T_{04}}^{T_{05}} C_{p_g}(T)dT}$$

$$\pi_t = \frac{p_{04}}{p_{05}}$$

Por último, se establece el acoplamiento de potencia entre el compresor y la turbina a través de la siguiente ecuación de compatibilidad:

$$W_{ec} = \eta_{mec}(1 - sangrado)(1 + f_1)\, C_p(T)dT$$

donde el trabajo de la turbina es:

$$W_{et} = \int_{T_{04}}^{T_{05}} C_p(T)dT$$

Posquemador

Condiciones a la entrada: T_{05}, p_{05}

Como se comentó en el esquema del motor, el postquemador es una segunda cámara de combustión en la que vuelve a inyectarse combustible para obtener aún más energía y acelerar más los gases a la salida de la tobera, consiguiendo así un extra de empuje. Las consideraciones del modelo de la postcombustión son las mismas que las que se tienen para la combustión, solo que ahora el rendimiento de postcombustión ηpc, será diferente. Por otra parte, las especies químicas que se obtienen son las mismas que antes, y los gastos de Ar y N_2 tambíen se mantienen constantes por ser gases inertes. El postquemador es en esencia un conducto, por lo que también se produce pérdida de carga, que al igual que en la cámara de combustión vendrá dado por la proporción de gasto de gas y gasto máximo de gas, aunque sigue siendo un factor que el usuario podrá fijar.

El balance de masa en el posquemador se refiere a la conservación de masa entre los gases de escape de la cámara de combustión principal y el combustible inyectado:

$$m_{fuel} = m_{esc}$$

El balance de energía tiene en cuenta la energía suministrada por el combustible y la energía contenida en los gases de escape:

$$Q_{fuel} = m_{esc} C_p (T_6 - T_5)$$

La eficiencia del posquemador puede definirse como la relación entre la energía agregada al flujo de gases de escape y la energía suministrada por el combustible:

$$\eta_{ab} = \frac{m_{esc} C_p (T_6 - T_5)}{m_{fuel} LHV}$$

Tobera

Condiciones a la entrada: T_{06}, p_{06}, π_{tob}

La tobera se aprovecha el salto entálpico que queda tras la expansión en la turbina para acelerar los gases de salida a través de ella y así obtener empuje para propulsar al motor. La tobera empleada para este análisis es convergente. No se ha hecho el estudio con una tobera convergente-divergente por simplificar las ecuaciones, puesto que la mayoría de los turborreactores instalados históricamente en aviones de combate llevan tobera convergente. Las ecuaciones que se muestran a continuación resuelven la tobera partiendo de la caída de presión de remanso πtob que se produce en la misma. Este dato tecnológico podrá venir asociado a un nivel de motor o ser elegido libremente por el usuario. El procedimiento para resolver la tobera comienza con la hipótesis de que está bloqueada. De esta forma, se fija que el número de Mach es el crítico e igual a 1, y se despeja el cociente crítico de presiones π_{crit} para comprobar si la presión a la salida es mayor o menor que la ambiente, comprobando así la hipótesis de partida.

$$M_6 = \frac{c_6}{a_6} = 1 = \frac{2C_p(T_6) - R)\int_{T_7}^{T_{06}} C_{p_g}(T)}{RT_7 C_p T_7}$$

Una vez que se tiene despejada la $T7$, a la que se asigna el nombre de T_{7crit} puesto que aún no sabemos si la tobera está bloqueada o adaptada, se despeja la $\pi crit$ de la siguiente ecuación de Gibbs-Duhem evaluada entre los puntos del proceso 06-7:

$$\int_{T_7}^{T_{06}} \frac{C_{p_g}(T)}{T} dT = Rlog\left(\frac{1}{\pi_{tob}\pi_{crit}}\right)$$

Como ya se ha comentado, aparece una doble casuística para distinguir si la tobera está bloqueada o adaptada:

Si $p_7 > p_0$, la tobera está bloqueada:

$$p_7 = \frac{p_{06}}{\pi_{crit}}$$

Por su parte, la temperatura a la salida de la tobera es la calculada previamente, $T_6 = T_{6crit}$.

Si $p_7 < p_0$, la tobera está adaptada:

$$p_7 = p_0$$

Entonces la hipótesis de tobera bloqueada, al ser incorrecta, obliga a recalcular la T_6, pues el número de Mach a la salida no es 1:

$$\int_{T_7}^{T_{06}} \frac{C_{p_g}(T)}{T} dT = Rlog\left(\frac{p_{06}}{p_0\pi_{tob}}\right)$$

Finalmente, sea cual sea la condición, el siguiente paso es calcular la velocidad y el área de la tobera.

Para ello basta con evaluar la diferencia de entalpías a la entrada y salida de la tobera, y aplicar la ecuación de continuidad a la salida de misma:

$$\int_{T_7}^{T_{06}} \frac{C_{p_g}(T)}{T} dT = \frac{{c_7}^2}{2}$$

$$A_7 = \frac{m_7 R T_7}{p_7 c_7}$$

Las estaciones de estudio del motor se definen gráficamente de la siguiente manera:

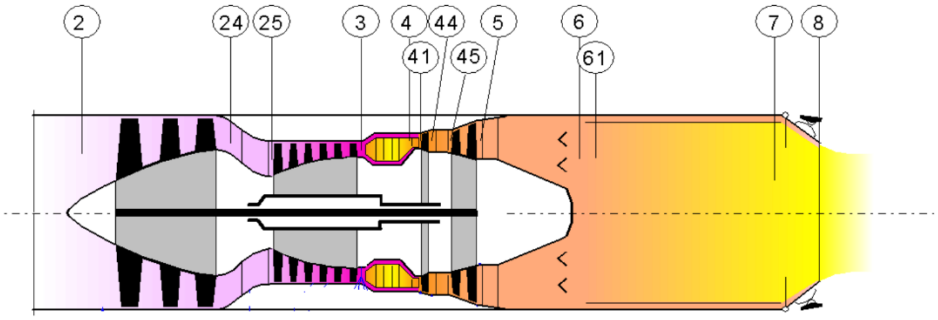

Figura 8: Estaciones para el estudio de las prestaciones de un motor turbofán de doble flujo

6. Gemelo digital de la planta de potencia

El primer paso para construir un gemelo digital es identificar el objetivo de la simulación. Es decir, qué se quiere conseguir y cómo. En este caso, lo que se pretende es obtener unos conjuntos de datos que definen el funcionamiento del motor en un régimen de vuelo determinado y con una configuración determinada que posteriormente se analizarán con herramientas de inteligencia artificial para optimizar la ejecución de estas simulaciones en base a los datos de entrada que se quieran seleccionar.

Para ello, será necesario construir una réplica virtual mediante simulaciones de un software, en este caso Gasturb, de la planta de potencia en cuestión.

Gasturb es un software desarrollado por el Dr. Ing. Horst Friedrich y su equipo en el Instituto de Propulsión Aeroespacial de la Universidad de Stuttgart, Alemania. Su desarrollo comenzó en la década de 1970 y desde entonces ha sido ampliamente utilizado en la industria aeroespacial, así como en otras áreas de la ingeniería mecánica donde se utilizan turbinas de gas.

El motor, como ya se ha indicado, es un turbofán de 2 ejes. El ejercicio constará de simular una serie de combinaciones tanto de los parámetros que definen el régimen de vuelo al que está sometido el motor (temperaturas, presiones, humedad, velocidades…) como de los parámetros que definen la configuración seleccionada del motor (ángulos de paso, 'tip clearance', número de álabes por etapa…) para poder extrapolar los resultados y posteriormente optimizar su simulación directamente mediante alguna de las técnicas más habituales de inteligencia artificial o 'machine learning'.

Para poder calcular los distintos parámetros de rendimiento que definen las prestaciones de un turbofán, es imprescindible poder medir de algún modo las magnitudes físicas que los definen. Por ello se instalan multitud de sensores y medidores que captan las magnitudes físicas que presenta el motor en un momento dado y bajo un régimen de funcionamiento dado.

Los sensores y medidores en un motor de aviación se colocan estratégicamente en varias partes del motor para monitorear sus prestaciones, temperatura, presión y otros parámetros críticos. La ubicación de cada sensor o medidor está diseñada para garantizar un funcionamiento seguro y eficiente del motor:

- Medidor de flujo de aire: Se coloca en la entrada de aire del motor para medir la cantidad de aire que ingresa. Esto es fundamental para calcular la mezcla aire-combustible adecuada y ajustar la potencia del motor en función de la densidad del aire
- Sensor de temperatura del aire de admisión: Se ubica en la entrada de aire para medir la temperatura del aire entrante. La temperatura afecta la densidad del aire y, por lo tanto, influye en la eficiencia del motor.
- Medidores de presión: Se colocan en diferentes puntos del motor, incluyendo la entrada de aire, el compresor y la salida de gases de escape. Sirven para medir la presión del aire y los gases en diversas etapas del proceso de compresión y expansión.
- Medidores de temperatura del gas de escape: Están ubicados en la salida de gases de escape del motor y miden la temperatura de los gases calientes. Esto es importante para monitorear la eficiencia del motor y asegurar que no se excedan las temperaturas límite.
- Medidores de presión de aceite: Se instalan en diferentes partes del sistema de lubricación del motor para asegurarse de que la presión del aceite sea adecuada para la lubricación de los componentes.

- Medidores de RPM (revoluciones por minuto): Se encuentran en el eje de la turbina y miden la velocidad de rotación del motor. Esto es crítico para el control y la monitorización del rendimiento del motor.
- Sensores de vibración: Se colocan en varias partes del motor para detectar cualquier vibración anormal que pueda indicar problemas en los componentes o el equilibrio del motor.
- Sensores de emisiones: En algunos motores, especialmente aquellos utilizados en aplicaciones comerciales, se colocan sensores de emisiones para medir y monitorear las emisiones de gases contaminantes, como dióxido de nitrógeno (NOx), dióxido de carbono (CO2) y monóxido de carbono (CO), para cumplir con las regulaciones medioambientales.
- Medidores de humedad del aire: En motores que operan en diferentes condiciones climáticas, se pueden instalar sensores de humedad del aire para tener en cuenta la humedad al calcular la mezcla aire-combustible y ajustar el rendimiento.

Estos sensores y medidores permiten a los operadores y pilotos monitorear y controlar las prestaciones del motor y detectar problemas potenciales para garantizar un funcionamiento seguro y eficiente. La ubicación específica de estos dispositivos varía según el diseño del motor y las necesidades de medición.

El primer paso es seleccionar combinaciones verosímiles de las magnitudes que definen las prestaciones del motor para obtener las curvas características y los parámetros clave que definen la eficiencia de su funcionamiento.

A continuación, se va a realizar un ejemplo completo de todos los parámetros y curvas que se pueden obtener en cada simulación, en este caso para las siguientes magnitudes de entrada o 'inputs' del gemelo digital implementado en el software:

Property	*Unit*	*Value*
Altitude	m	11000
Delta T from ISA	K	0
Relative Humidity [%]		0
Mach Number		0,8

Tabla 1: Condiciones ambiente para el estudio de un motor turbofán de doble flujo

Property	*Unit*	*Value*
Intake Pressure Ratio		0,99
No (0) or Average (1) Core dP/P		1
Inner Fan Pressure Ratio		2
Booster Map Type (0/1/2)		0
Outer Fan Pressure Ratio		1
Compr. Interduct Press. Ratio		0,99
HP Compressor Pressure Ratio		20
Bypass Duct Pressure Ratio		0,98
Turb. Interd. Ref. Press. Ratio		0,98
Design Bypass Ratio		5
Burner Exit Temperature	K	1600
Burner Design Efficiency		0,9995
Burner Partload Constant		1,6
Fuel Heating Value	MJ/kg	43,124
Overboard Bleed	kg/s	0
Power Offtake	kW	0
HP Spool Mechanical Efficiency		1
LP Spool Mechanical Efficiency		1
Burner Pressure Ratio		0,97
Turbine Exit Duct Press Ratio		0,98
Hot Stream Mixer Press Ratio		0,99
Cold Stream Mixer Press Ratio		0,99
Mixed Stream Pressure Ratio		1
Mixer Efficiency		0,5
Design Mixer Mach Number		0,247
Design Mixer Area	m²	0

Tabla 2: Propiedades básicas estándar I del funcionamiento de un motor turbofán de doble flujo

Property	*Unit*	*Value*
Booster(0)or HPC(1)Handl.Bleed		0
Rel. Handling Bleed to Bypass		0
Rel. Enthalpy of HP Handl.Bleed		0
Rel. HP Leakage to Bypass		0
Rel. Overboard Bleed W_Bld/W25		0,005
Rel. Enthalpy of Overb. Bleed		1
Recirculating Bleed W_reci/W25		0
Rel. Enthalpy of Recirc Bleed		1
Number of HP Turbine Stages		1
HPT NGV 1 Cooling Air / W25		0,05
HPT Rotor 1 Cooling Air / W25		0,05
HPT Cooling Air Pumping Dia	m	0
Number of LP Turbine Stages		1
LPT NGV 1 Cooling Air / W25		0
LPT Rotor 1 Cooling Air / W25		0,03
Rel. Enth. LPT NGV Cooling Air		0,6
Rel. Enth. of LPT Cooling Air		0,6
Rel.HP Leakage to LPT exit		0
Rel. Fan Overb.Bleed W_Bld/W13		0
Core-Byp Heat Transf Effectiven		0
Coolg Air Cooling Effectiveness		0
Bleed Air Cooling Effectiveness		0

Tabla 3: Propiedades básicas estándar II del funcionamiento de un motor turbofán de doble flujo

Station	W kg/s	T K	P kPa	WRstd kg/s
amb		216,65	22,632	
1	36,608	244,44	34,509	
2	36,608	244,44	34,164	100,000
13	30,506	301,31	65,511	48,249
21	6,101	313,11	68,328	9,432
25	6,101	313,11	67,645	9,527
3	5,918	784,20	1352,891	0,731
31	5,278	784,20	1352,891	
4	5,406	1600,00	1312,304	0,984
41	5,711	1559,91	1312,304	1,026
43	5,711	1142,07	285,169	
44	6,016	1125,10	285,169	
45	6,016	1125,10	279,465	4,310
49	6,016	818,81	65,512	
5	6,199	812,77	65,512	16,102
6	6,199	812,77	64,201	
16	30,506	301,31	64,201	
64	36,705	392,87	63,404	
8	36,705	392,87	63,404	68,493
Bleed	0,031	784,20	1352,889	

Efficiencies:	isentr	polytr	RNI	P/P
Outer LPC	0,8800	0,8905	0,409	1,918
Inner LPC	0,7800	0,8003	0,409	2,000
HP Compressor	0,8600	0,9039	0,605	20,000
Burner	0,9995			0,970
HP Turbine	0,9000	0,8823	1,801	4,602
LP Turbine	0,9100	0,8938	0,558	4,266
Mixer	0,5000			

HP Spool mech Eff 1,0000 Nom Spd 22800 rpm
LP Spool mech Eff 1,0000 Nom Spd 14600 rpm

P2/P1= 0,9900 P25/P21=0,9900 P45/P44= 0,9800

hum [%]	war0	FHV	Fuel
0,0	0,00000	43,124	Generic

FN	=	7,70	kN
TSFC	=	16,6510	g/(kN*s)
WF Burner	=	0,12822	kg/s
s NOX	=	0,6624	
BPR	=	5,0000	
Core Eff	=	0,5476	
Prop Eff	=	0,6931	
P3/P2	=	39,600	
P16/P6	=	1,00000	
A63	=	0,17313	m²
A163	=	0,51813	m²
A64	=	0,69126	m²
XM63	=	0,24330	
XM163	=	0,23818	
XM64	=	0,24700	
P63/P6	=	0,99000	
P163/P16	=	0,99000	
A8	=	0,29907	m²
CD8	=	0,95000	
Ang8	=	25,00	°
P8/Pamb	=	2,80150	
WLkBy/W25	=	0,00000	
WCHN/W25	=	0,05000	
WCHR/W25	=	0,05000	
Loading	=	100,00	%
WCLN/W25	=	0,00000	
WCLR/W25	=	0,03000	
WBHD/W21	=	0,00000	
far7	=	0,00351	
WBLD/W25	=	0,00500	
PWX	=	0,0	kW
P16/P13	=	0,9800	
P6/P5	=	0,9800	

Tabla 4: Output I por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas

	Units	St 2	St 21	St 25	St 3	St 4	St 44	St 45	St 5	St 6	St 13	St 16	St 64	St 8
Mass Flow	kg/s	36,6077	6,10128	6,10128	5,91824	5,40582	6,01595	6,01595	6,19899	6,19899	30,5064	30,5064	36,7054	36,7054
Total Temperature	K	244,442	313,114	313,114	784,201	1600	1125,1	1125,1	812,765	812,765	301,31	301,31	392,872	392,871
Static Temperature	K	232,78	298,25	298,25	778,625	1579,26	1090,45	1097,63	779,499	806,385	291,96	296,659	388,189	327,894
Total Pressure	kPa	34,1639	68,3278	67,6445	1352,89	1312,3	285,169	279,465	65,5117	64,2014	65,5114	64,2012	63,4036	63,4036
Static Pressure	kPa	28,7978	57,6052	57,0292	1316,83	1238,64	250,105	251,867	55,5846	62,2146	58,6731	60,7991	60,7807	33,5334
Velocity	m/s	152,973	173,098	173,098	110,1	229,611	288,832	257,534	274,068	120,853	137,015	96,6767	97,3468	362,613
Area	m²	0,555263	0,052385	0,052914	9,1235E-3	8,6165E-3	0,026067	0,029222	0,091049	0,190839	0,318029	0,441964	0,691261	0,284117
Mach Number		0,5	0,5	0,5	0,2	0,3	0,45	0,4	0,5	0,217	0,4	0,28	0,247	1
Density	kg/m³	0,43098	0,672859	0,666131	5,89175	2,73237	0,799033	0,799397	0,24842	0,26878	0,700095	0,713973	0,545464	0,356277
Spec Heat @ T	J/(kg*K)	1003,47	1005,94	1005,94	1094,94	1276,13	1204,52	1204,52	1135,26	1135,26	1004,91	1004,91	1016,51	1016,51
Spec Heat @ Ts	J/(kg*K)	1003,19	1004,76	1004,76	1093,61	1273,83	1198,17	1199,61	1126,79	1133,7	1004,61	1004,72	1016,06	1010,31
Enthalpy @ T	J/kg	-53881,5	15081,3	15081,3	506601	1,51299E6	918415	918415	551778	551778	3176,98	3177	95827,4	95827,3
Enthalpy @ Ts	J/kg	-65581,9	99,8733	99,8733	500540	1,48663E6	876703	885253	514221	544475	-6209,53	-1496,19	91089,2	30083
Entropy Function @ T		-0,694205	0,17187	0,17187	3,48626	6,58223	5,04107	5,04107	3,71219	3,71219	0,03692	0,03692	0,97189	0,971889
Entropy Function @ Ts		-0,865075	1,1668E-3	1,1668E-3	3,45925	6,52446	4,90987	4,9371	3,54787	3,68076	-0,073323	-0,017527	0,929642	0,334912
Exergy	J/kg	27257,1	85465,6	84840,5	556543	1,36855E6	774880	773624	399413	398157	79336	78079,6	111813	111813
Gas Constant	J/(kg*K)	287,05	287,05	287,05	287,05	287,046	287,047	287,047	287,047	287,047	287,05	287,05	287,05	287,05
Fuel-Air-Ratio		0	0	0	0	0,024295	0,021777	0,021777	0,02112	0,02112	0	0	3,5054E-3	3,5054E-3
Water-Air-Ratio		0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 5: Output I por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas

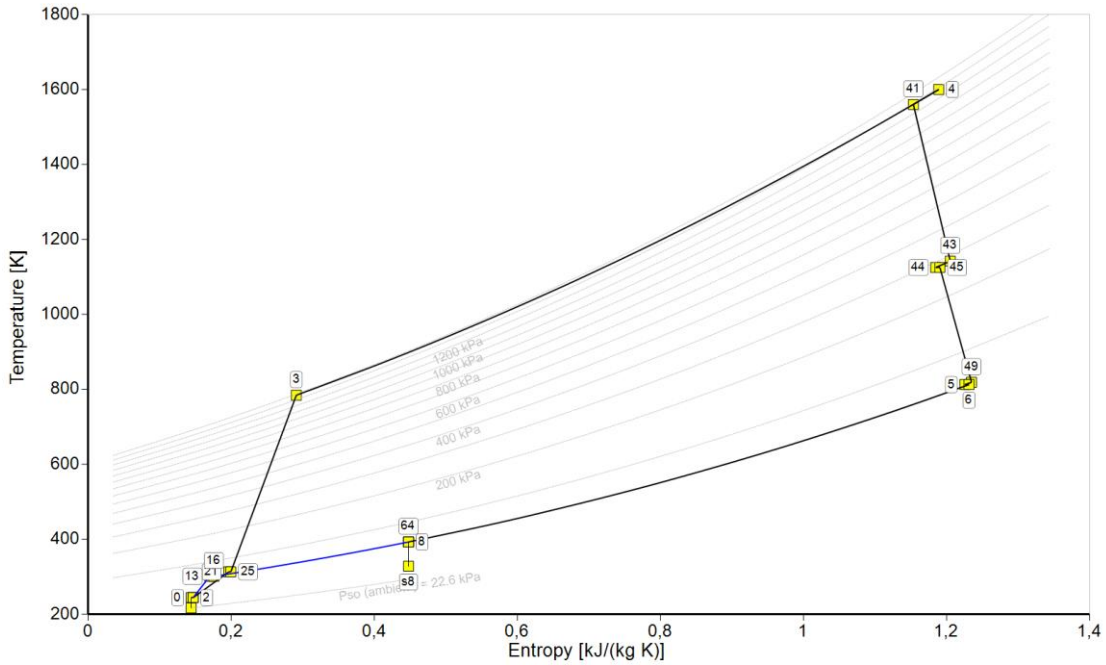

Figura 9: Diagrama temperatura vs entropía

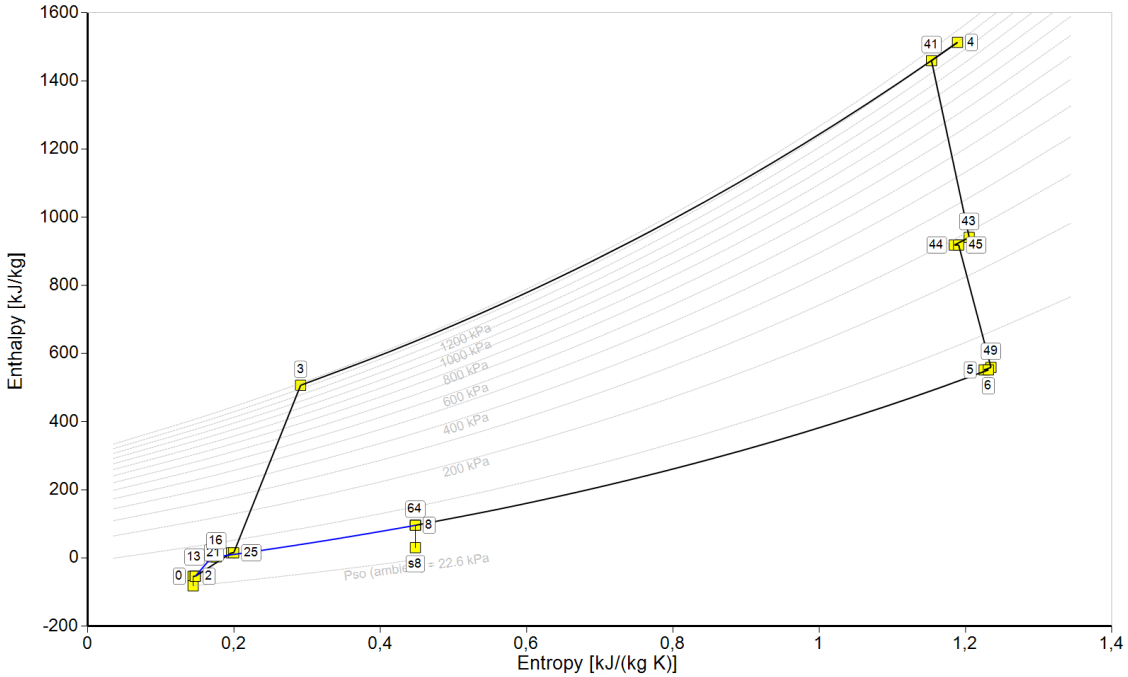

Figura 10: Diagrama entalpía vs entropía

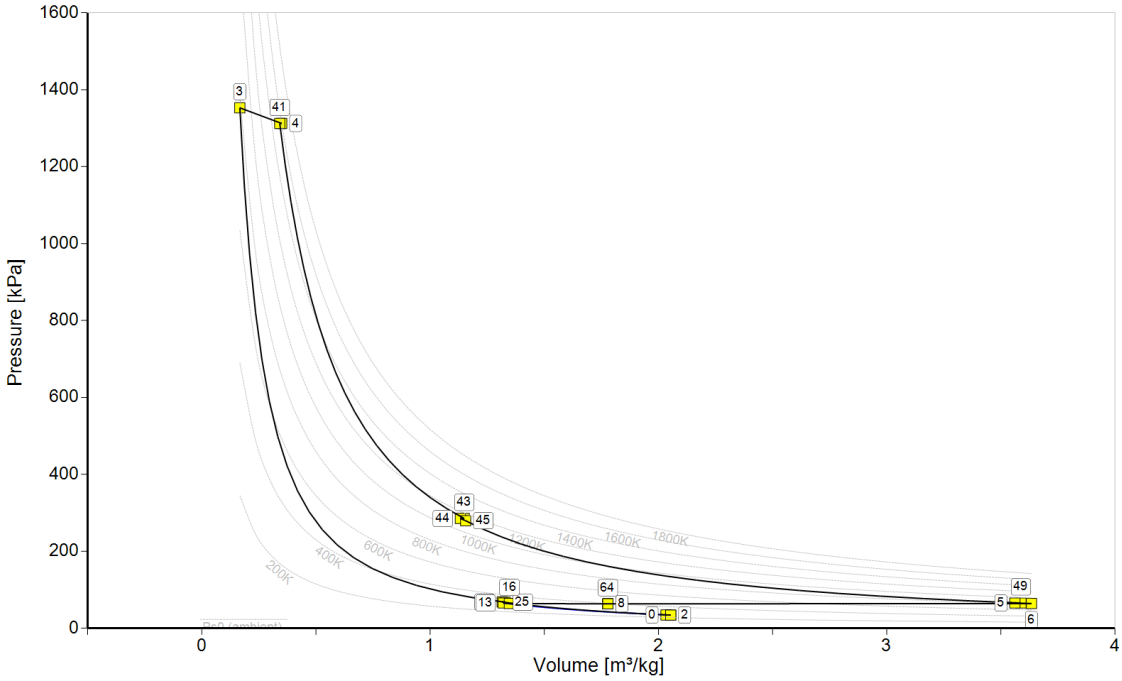

Figura 11: Diagrama presión vs volumen

Las variables de diseño sobre las que se realiza el análisis paramétrico son: la temperatura de salida de la turbina (Burner Exit Temperature), la presión de entrada a la turbina (HP Compressor Pressure Ratio), y la relación de bypass. Esto ayuda a observar cómo afectan al rendimiento del motor la variación de estas variables.

- Burner Exit Temperature: afecta directamente la eficiencia del ciclo termodinámico del motor y, por lo tanto, al empuje y la eficiencia. Sin embargo, los valores más altos también pueden llevar a una vida útil más corta del motor debido a mayores tensiones térmicas. En este caso, se ha seleccionado un rango entre 1.400-1.950 k con un step de 50.

- HP Compressor Pressure Ratio es un parámetro clave para determinar el rendimiento del motor. Aumentar la presión de entrada a la turbina puede aumentar la eficiencia del ciclo, pero también puede aumentar la carga térmica y mecánica en el motor. En este caso, se ha seleccionado un rango entre 6-24 con un step de 2.

- La relación de 'bypass' (la proporción del aire que pasa alrededor del núcleo del motor en comparación con el que pasa a través del núcleo) es un factor crucial para determinar el equilibrio entre el empuje y la eficiencia del combustible. Una mayor relación de' bypass' puede mejorar la eficiencia del combustible, pero puede disminuir el empuje máximo. Considerando que el tipo de motor es un 'Low-Bypass Ratio', se ha fijado a 5.

Se ejecuta el análisis, y representa la relación Burner Exit Temperature vs Specific Thrust:

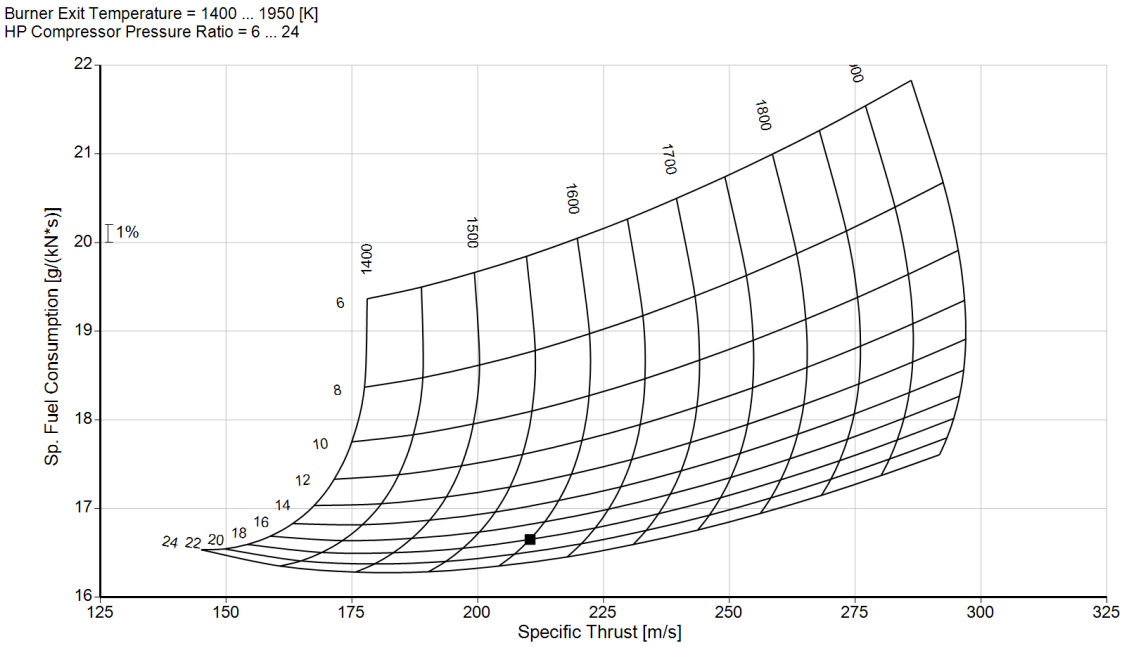

Figura 12: Diagrama consumo específico de combustible vs empuje específico

La gráfica que representa la temperatura a la salida de la cámara de combustión frente al consumo específico y al empuje específico de un motor de avión se conoce como el "mapa de rendimiento del motor" o "mapa de funcionamiento del motor". Este mapa es una herramienta fundamental para analizar y comprender el rendimiento del motor en diferentes condiciones de operación.

El consumo específico se refiere a la cantidad de combustible necesaria para producir una unidad de empuje durante un período de tiempo determinado. Por lo general, se expresa en términos de masa de combustible (generalmente libras o kilogramos) por unidad de empuje (generalmente libras o kilogramos de fuerza) por hora.

El empuje específico, por otro lado, es una medida del empuje producido por unidad de flujo de masa de combustible. Se expresa en términos de fuerza (libras o kilogramos de fuerza) por masa de combustible (libras o kilogramos) por segundo.

El mapa de rendimiento del motor muestra la relación entre la temperatura a la salida de la cámara de combustión y el consumo y empuje específico en un rango de condiciones de operación. En el eje horizontal se representa el empuje específico, mientras que en el eje vertical se representa el consumo específico. La temperatura a la salida de la cámara de combustión se puede representar mediante líneas de contorno o colores en el mapa.

En el mapa, cada punto representa una combinación específica de consumo y empuje específico, y se calcula la temperatura a la salida de la cámara de combustión correspondiente. Estas mediciones se obtienen mediante pruebas en bancos de pruebas o mediante modelos y simulaciones computacionales. Es importante tener en cuenta que cada motor tendrá su propio mapa de rendimiento específico, ya que está influenciado por factores como el diseño del motor, la geometría de la cámara de combustión, la relación de compresión, el flujo de aire y las características del combustible utilizado.

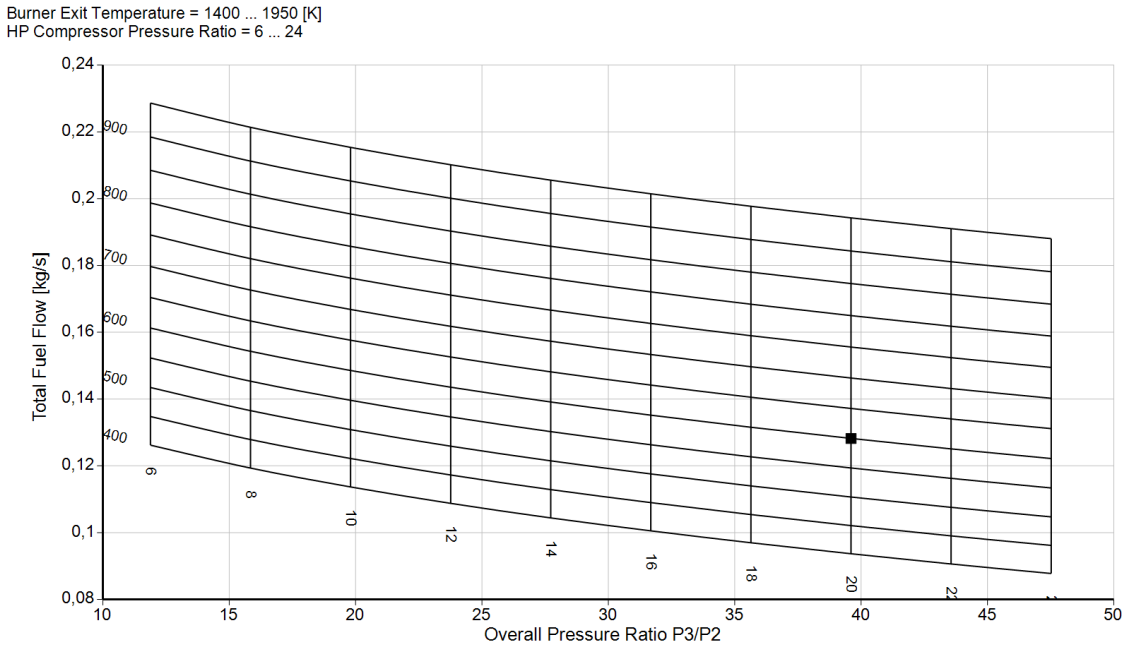

Figura 13: Diagrama flujo de combustible vs relación de presiones P3/P2

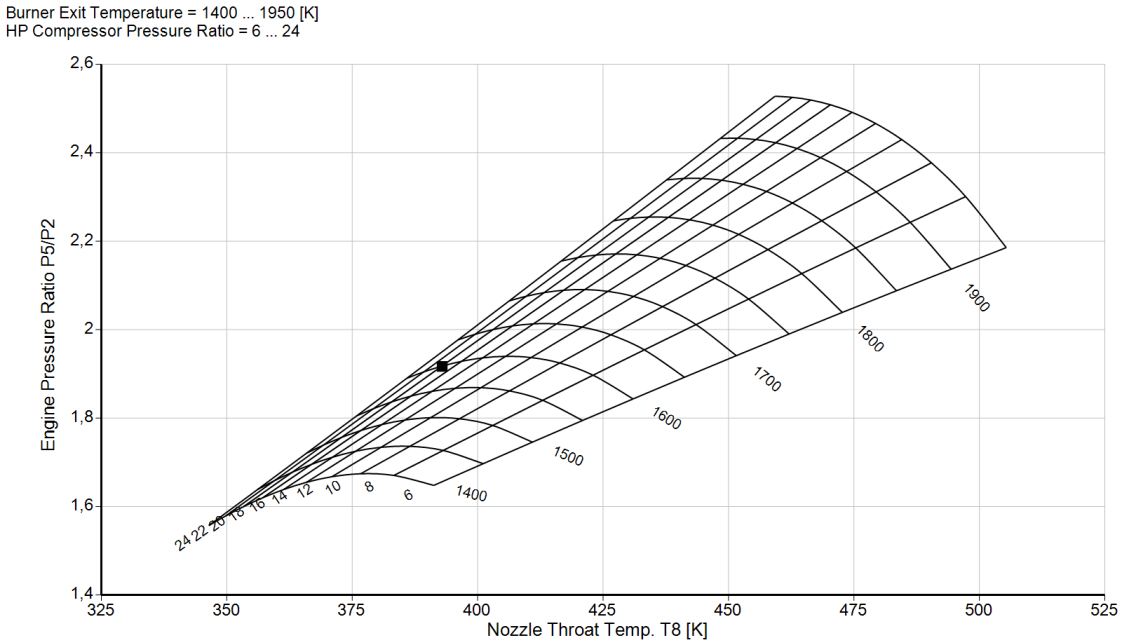

Figura 14: Diagrama EPR (P5/P2) vs temperatura de salida del motor (T8)

A continuación, se calculan los parámetros que definen las prestaciones del motor fuera del punto de diseño:

Tabla 6: Output I por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas fuera del punto de diseño

Station	W kg/s	T K	P kPa	WRstd kg/s
amb		216,65	22,632	
1	36,608	244,44	34,509	
2	36,608	244,44	34,164	100,000
13	30,506	301,31	65,511	48,249
21	6,101	313,11	68,328	9,432
25	6,101	313,11	67,645	9,527
3	5,918	784,20	1352,891	0,731
31	5,278	784,20	1352,891	
4	5,406	1600,00	1312,304	0,984
41	5,711	1559,91	1312,304	1,026
43	5,711	1142,07	285,166	
44	6,016	1125,10	285,166	
45	6,016	1125,10	279,463	4,310
49	6,016	818,81	65,511	
5	6,199	812,77	65,511	16,103
6	6,199	812,77	64,201	
16	30,506	301,31	64,201	
64	36,705	392,87	63,403	
8	36,705	392,87	63,403	68,494
Bleed	0,031	784,20	1352,889	

Efficiencies:	isentr	polytr	RNI	P/P
Outer LPC	0,8800	0,8905	0,409	1,918
Inner LPC	0,7800	0,8003	0,409	2,000
HP Compressor	0,8600	0,9039	0,605	20,000
Burner	0,9995			0,970
HP Turbine	0,9000	0,8823	1,801	4,602
LP Turbine	0,9100	0,8938	0,558	4,266
Mixer	0,5000			

HP Spool mech Eff 1,0000 Speed 22800 rpm
LP Spool mech Eff 1,0000 Speed 14600 rpm

P2/P1= 0,9900 P25/P21=0,9900 P45/P44= 0,9800

hum [%]	war0	FHV	Fuel
0,0	0,00000	43,124	Generic

FN	=	7,70 kN
TSFC	=	16,6511 g/(kN*s)
WF Burner	=	0,12822 kg/s
s NOX	=	0,6624
BPR	=	5,0000
Core Eff	=	0,5476
Prop Eff	=	0,6931
P3/P2	=	39,600
P5/P2	=	1,9176 EPR
P16/P6	=	1,00001
A63	=	0,17313 m²
A163	=	0,51813 m²
A64	=	0,69126 m²
XM63	=	0,24331
XM163	=	0,23818
XM64	=	0,24700
P63/P6	=	0,99000
P163/P16	=	0,99000
A8	=	0,29907 m²
CD8	=	0,95000
Ang8	=	25,00 °
P8/Pamb	=	2,80147
WLkBy/W25	=	0,00000
WCHN/W25	=	0,05000
WCHR/W25	=	0,05000
Loading	=	100,00 %
WCLN/W25	=	0,00000
WCLR/W25	=	0,03000
WBHD/W21	=	0,00000
far7	=	0,00351
WBLD/W25	=	0,00500
PWX	=	0,0 kW
P16/P13	=	0,9800
P6/P5	=	0,9800

Tabla 7: Output II por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas fuera del punto de diseño

Spool Speeds:	LP Spool	HP Spool
Absolute [RPM]	14600,0	22800,0
Relative	1,0000	1,0000

Booster Map Type = 0
Fan Inner (Booster) Map is Scaled from Fan Outer Map

	LPC	Booster	HPC
Surge Margin [%]	45,477	45,477	24,104
Handling Bleed WB,hd/W22		0,0000	

Map Coordinates:	LPC		HPC	HPT	LPT
Map Speed	1,0000		1,0000	1,0000	1,0000
Map Coordinate Beta	0,5000		0,5000	0,5000	0,5000
Reynolds Corrections:	LPC	Booster	HPC	HPT	LPT
Efficiency	1,0000	1,0000	0,9891	1,0000	0,9873
Flow	1,0000	1,0000	0,9945	1,0000	0,9937

Modifiers:	Bypass	LPC-Core	Booster	HPC	HPT	LPT
Delta Efficiency [%]	0,000	0,000	0,000	0,000	0,000	0,000
Delta Flow Capacity [%]	0,000		0,000	0,000	0,000	0,000
Delta Core Mixer Area [%]	0,000					
Delta Byp Mixer Area [%]	0,000					
Delta Nozzle Area [%]	0,000					
Delta Comp Interd P/P [%]	0,000					
Delta Burner P/P [%]	0,000					
Delta Turb Interd P/P [%]	0,000					
Delta Turbine Exit P/P [%]	0,000					
Delta Bypass P/P [%]	0,000					

Tabla 8: Tabla 7: Output III por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas fuera del punto de diseño

	Units	St 2	St 21	St 25	St 3	St 4	St 44	St 45	St 5	St 6	St 13	St 16	St 64	St 8
Mass Flow	kg/s	36,6077	6,10128	6,10128	5,91824	5,40582	6,01595	6,01595	6,19899	6,19899	30,5064	30,5064	36,7054	36,7054
Total Temperature	K	244,442	313,114	313,114	784,202	1600	1125,1	1125,1	812,765	812,765	301,31	301,31	392,872	392,871
Static Temperature	K	232,78	298,25	298,25	778,625	1579,26	1090,45	1097,63	779,499	806,384	291,96	296,659	388,189	327,894
Total Pressure	kPa	34,1639	68,3278	67,6445	1352,89	1312,3	285,166	279,463	65,511	64,2008	65,5114	64,2012	63,403	63,403
Static Pressure	kPa	28,7978	57,6052	57,0292	1316,83	1238,64	250,102	251,864	55,584	62,2139	58,6731	60,799	60,7801	33,5331
Velocity	m/s	152,973	173,098	173,098	110,1	229,611	288,835	257,537	274,069	120,854	137,015	96,6773	97,3476	362,613
Area	m²	0,555263	0,052385	0,052914	9,1235E-3	8,6165E-3	0,026067	0,029222	0,091049	0,190839	0,318029	0,441964	0,691262	0,28412
Mach Number		0,5	0,500001	0,500001	0,200002	0,3	0,450006	0,400005	0,500001	0,217003	0,4	0,280002	0,247003	1
Density	kg/m³	0,43098	0,672859	0,666131	5,89175	2,73237	0,799023	0,799387	0,248417	0,268777	0,700095	0,713973	0,545458	0,356274
Spec Heat @ T	J/(kg*K)	1003,47	1005,94	1005,94	1094,94	1276,13	1204,52	1204,52	1135,26	1135,26	1004,91	1004,91	1016,51	1016,51
Spec Heat @ Ts	J/(kg*K)	1003,19	1004,76	1004,76	1093,61	1273,83	1198,17	1199,61	1126,79	1133,7	1004,61	1004,72	1016,06	1010,31
Enthalpy @ T	J/kg	-53881,5	15081,3	15081,3	506601	1,51299E6	918414	918414	551778	551778	3176,94	3176,96	95827,4	95827,3
Enthalpy @ Ts	J/kg	-65581,9	99,8353	99,8288	500540	1,48663E6	876702	885252	514221	544475	-6209,55	-1496,29	91089,1	30083
Entropy Function @ T		-0,694205	0,17187	0,17187	3,48626	6,58223	5,04107	5,04107	3,71219	3,71219	0,03692	0,03692	0,97189	0,971889
Entropy Function @ Ts		-0,865075	1,1664E-3	1,1663E-3	3,45925	6,52446	4,90987	4,93709	3,54787	3,68076	-0,073323	-0,017528	0,929641	0,334911
Exergy	J/kg	27257,1	85465,5	84840,5	556543	1,36855E6	774879	773623	399412	398156	79336	78079,6	111813	111813
Gas Constant	J/(kg*K)	287,05	287,05	287,05	287,05	287,046	287,047	287,047	287,047	287,047	287,05	287,05	287,05	287,05
Fuel-Air-Ratio		0	0	0	0	0,024295	0,021777	0,021777	0,02112	0,02112	0	0	3,5054E-3	3,5054E-3
Water-Air-Ratio		0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 9: Tabla 7: Output IV por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas fuera del punto de diseño

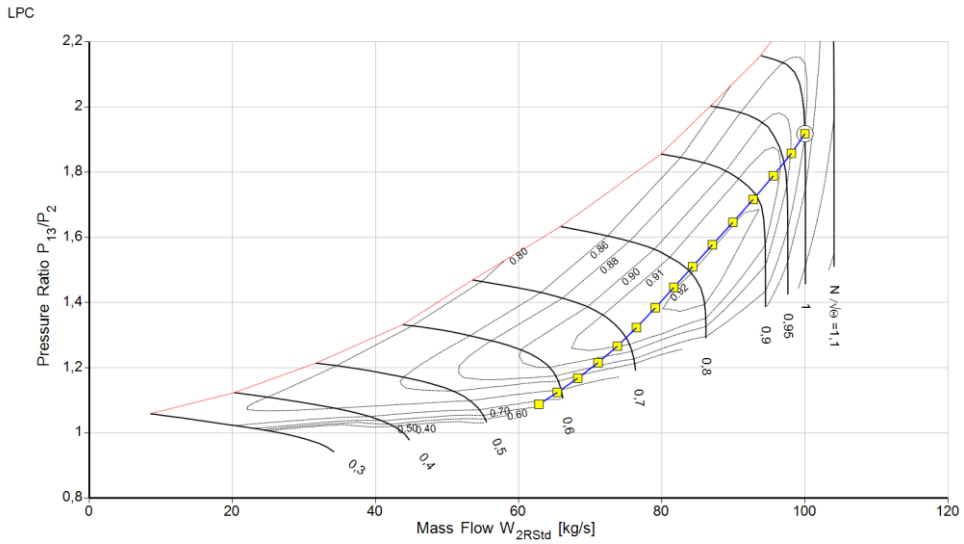

Figura 15: Diagrama relación de presión LPC (P13/P2) vs flujo másico corregido

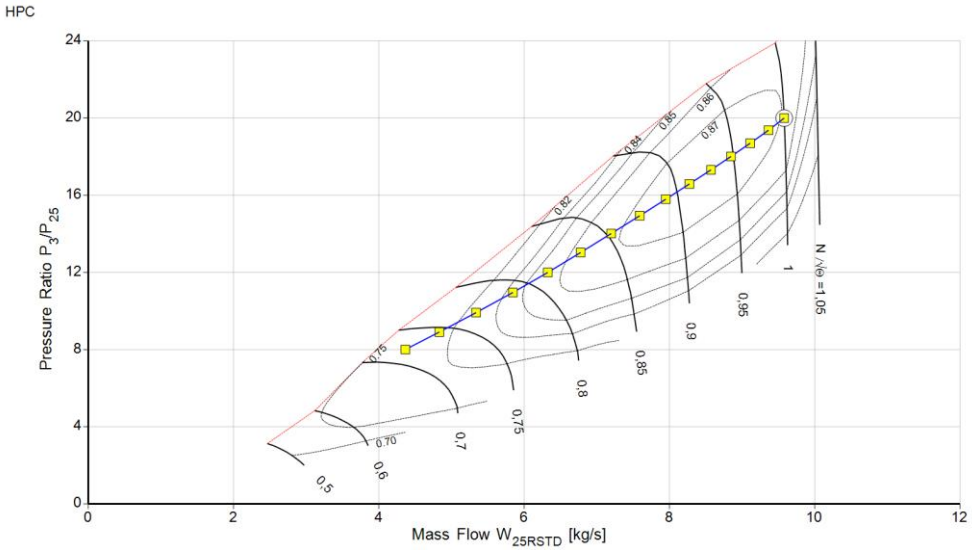

Figura 16: Diagrama relación presiones HPC (P3/P25) vs flujo másico corregido

Además, para el despegue, es decir, seleccionando las vueltas (RPM) máximas a altitud 0:

Station	W kg/s	T K	P kPa	WRstd kg/s
amb		288,15	101,325	
1	122,006	324,96	154,443	
2	122,006	324,96	152,899	85,861
13	104,928	372,05	237,277	50,915
21	17,078	382,15	244,858	8,138
25	17,078	382,15	243,035	8,199
3	16,565	878,75	3941,160	0,744
31	14,772	878,75	3941,160	
4	15,156	1727,88	3818,847	0,985
41	16,010	1686,34	3818,847	1,028
43	16,010	1246,05	836,279	
44	16,864	1228,66	836,279	
45	16,864	1228,66	819,589	4,305
49	16,864	933,27	230,464	
5	17,376	926,43	230,464	13,698
6	17,376	926,43	227,128	
16	104,928	372,05	231,993	
64	122,305	456,67	228,058	
8	122,305	456,67	228,058	68,408
Bleed	0,085	878,75	3941,158	

Efficiencies:	isentr	polytr	RNI	P/P
Outer LPC	0,9198	0,9246	1,308	1,552
Inner LPC	0,8153	0,8271	1,308	1,601
HP Compressor	0,8763	0,9124	1,714	16,216
Burner	0,9999			0,969
HP Turbine	0,8921	0,8737	4,789	4,566
LP Turbine	0,9164	0,9036	1,479	3,556
Mixer	0,5000			

HP Spool mech Eff 1,0000	Speed	22800 rpm	
LP Spool mech Eff 1,0000	Speed	13869 rpm	
P2/P1= 0,9900	P25/P21=0,9926	P45/P44= 0,9800	

hum [%]	war0	FHV	Fuel
0,0	0,00000	43,124	Generic

FN	=	19,82	kN
TSFC	=	19,3795	g/(kN*s)
WF Burner	=	0,38410	kg/s
s NOX	=	1,6523	
BPR	=	6,1442	
Core Eff	=	0,4892	
Prop Eff	=	0,7714	
P3/P2	=	25,776	
P5/P2	=	1,5073	EPR
P16/P6	=	1,02142	
A63	=	0,17313	m²
A163	=	0,51813	m²
A64	=	0,69126	m²
XM63	=	0,20399	
XM163	=	0,25364	
XM64	=	0,24713	
P63/P6	=	0,99284	
P163/P16	=	0,98881	
A8	=	0,29907	m²
CD8	=	0,95000	
Ang8	=	25,00	°
P8/Pamb	=	2,25076	
WLkBy/W25	=	0,00000	
WCHN/W25	=	0,05000	
WCHR/W25	=	0,05000	
Loading	=	29,80	%
WCLN/W25	=	0,00000	
WCLR/W25	=	0,03000	
WBHD/W21	=	0,00000	
far7	=	0,00315	
WBLD/W25	=	0,00500	
PWX	=	0,0	kW
P16/P13	=	0,9777	
P6/P5	=	0,9855	

Tabla 10: Output I por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas en despegue

Spool Speeds:	LP Spool	HP Spool
Absolute [RPM]	13869,3	22800,0
Relative	0,9500	1,0000

Booster Map Type = 0
Fan Inner (Booster) Map is Scaled from Fan Outer Map

	LPC	Booster	HPC
Surge Margin [%]	78,118	78,118	30,787
Handling Bleed WB,hd/W22		0,0000	

Map Coordinates:	LPC		HPC	HPT	LPT
Map Speed	0,8239		0,9052	0,9618	0,9090
Map Coordinate Beta	0,3426		0,5305	0,5019	0,3553
Reynolds Corrections:	LPC	Booster	HPC	HPT	LPT
Efficiency	1,0000	1,0000	1,0000	1,0000	1,0000
Flow	1,0000	1,0000	1,0000	1,0000	1,0000

Modifiers:	Bypass	LPC-Core	Booster	HPC	HPT	LPT
Delta Efficiency [%]	0,000	0,000	0,000	0,000	0,000	0,000
Delta Flow Capacity [%]	0,000		0,000	0,000	0,000	0,000
Delta Core Mixer Area [%]	0,000					
Delta Byp Mixer Area [%]	0,000					
Delta Nozzle Area [%]	0,000					
Delta Comp Interd P/P [%]	0,000					
Delta Burner P/P [%]	0,000					
Delta Turb Interd P/P [%]	0,000					
Delta Turbine Exit P/P [%]	0,000					
Delta Bypass P/P [%]	0,000					

Tabla 11: Output II por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas en despegue

	Units	St 2	St 21	St 25	St 3	St 4	St 44	St 45	St 5	St 6	St 13	St 16	St 64	St 8
Mass Flow	kg/s	122,006	17,0776	17,0776	16,5652	15,1562	16,864	16,864	17,3763	17,3763	104,928	104,928	122,305	122,305
Total Temperature	K	324,957	382,152	382,152	878,75	1727,88	1228,66	1228,66	926,432	926,432	372,051	372,051	456,673	456,673
Static Temperature	K	314,449	369,654	369,731	872,409	1705,83	1191,53	1199,23	902,014	921,411	358,963	365,583	451,31	381,851
Total Pressure	kPa	152,899	244,858	243,035	3941,16	3818,85	836,279	819,589	230,464	227,128	237,277	231,993	228,058	228,058
Static Pressure	kPa	136,229	217,872	216,409	3832,27	3604	733,683	738,87	206,696	222,154	209,236	218,137	218,643	120,684
Velocity	m/s	145,586	158,772	158,28	118,648	238,98	301,588	268,869	239,064	108,403	162,477	114,215	104,832	390,956
Area	m²	0,555263	0,052385	0,052914	9,1235E-3	8,6165E-3	0,026067	0,029222	0,091049	0,190839	0,318029	0,441964	0,691262	0,28413
Mach Number		0,409658	0,412437	0,411115	0,204301	0,300998	0,450718	0,400598	0,407382	0,182882	0,428223	0,298318	0,247128	1
Density	kg/m³	1,50925	2,05328	2,03906	15,303	7,36035	2,14511	2,14641	0,7983	0,83994	2,03063	2,07867	1,68774	1,10103
Spec Heat @ T	J/(kg*K)	1006,97	1011,95	1011,95	1116,18	1292,68	1225,3	1225,3	1164,98	1164,98	1011,07	1011,07	1026,19	1026,19
Spec Heat @ Ts	J/(kg*K)	1006,06	1010,86	1010,87	1114,77	1290,69	1219,28	1220,65	1159,52	1163,86	1009,93	1010,51	1025,3	1015,1
Enthalpy @ T	J/kg	27025,1	84706,1	84706,1	611185	1,68138E6	1,04626E6	1,04626E6	683735	683735	74519,1	74519,1	161073	161073
Enthalpy @ Ts	J/kg	16427,5	72101,8	72179,9	604147	1,65283E6	1,00078E6	1,01011E6	655159	677859	61319,7	67996,6	155578	84649,4
Entropy Function @ T		0,302249	0,871598	0,871598	3,925	6,94208	5,42376	5,42376	4,24391	4,24391	0,777525	0,777525	1,50779	1,50779
Entropy Function @ Ts		0,186809	0,754827	0,755563	3,89698	6,88417	5,29287	5,32007	4,13506	4,22177	0,651761	0,715941	1,46563	0,871365
Exergy	J/kg	36226,6	85764,6	85146,4	589510	1,40755E6	772395	770727	400853	399647	80757,4	78894,5	103631	103631
Gas Constant	J/(kg*K)	287,05	287,05	287,05	287,05	287,046	287,046	287,046	287,046	287,046	287,05	287,05	287,05	287,05
Fuel-Air-Ratio		0	0	0	0	0,026002	0,023307	0,023307	0,022605	0,022605	0	0	3,1504E-3	3,1504E-3
Water-Air-Ratio		0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 12: Output III por estación de las prestaciones de un motor turbofán de doble flujo para las condiciones dadas en despegue

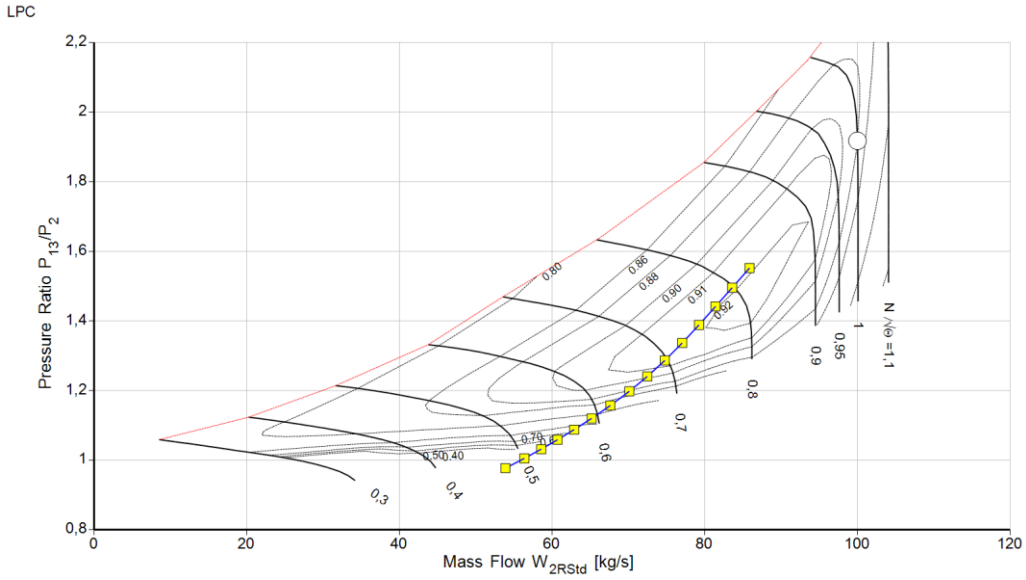

Figura 17: Diagrama relación presiones LPC (P13/P2) vs flujo másico corregido en despegue

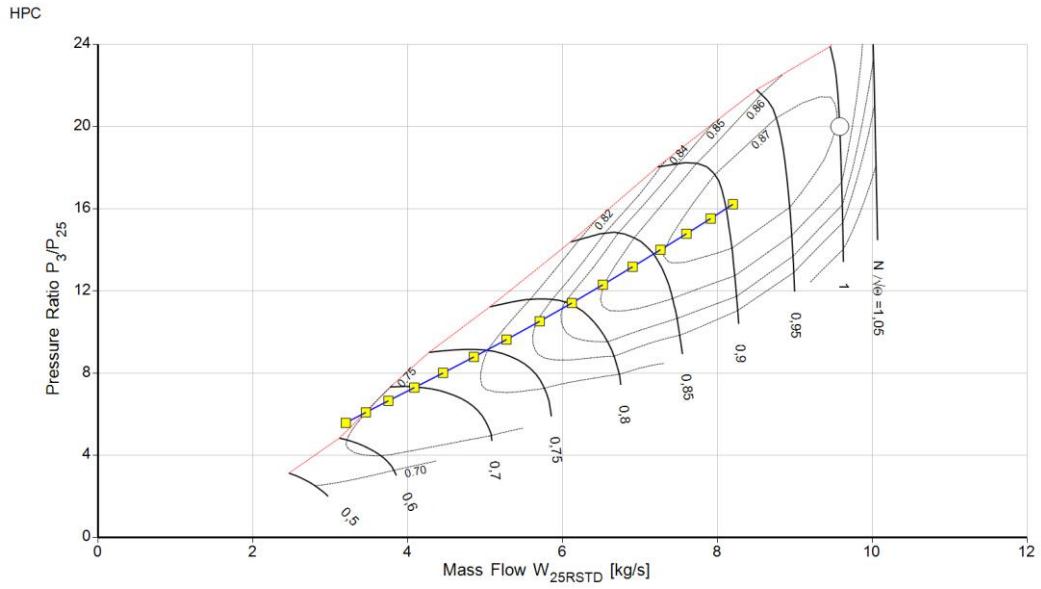

Figura 18: Diagrama relación presiones HPC (P3/P25) vs flujo másico corregido en despegue

7. Resultados de la simulación del gemelo digital

Por comodidad en el procesado de datos, y la necesidad de tener una muestra suficientemente grande con distintas iteraciones y distintos outputs de la misma configuración de motor, la idea detrás de este proyecto no pasa por la simulación caso a caso de cada conjunto de datos de los parámetros de entrada en el software sino en el desarrollo de un código en algún lenguaje de programación que posteriormente permita la extracción masiva de estos outputs. Por supuesto, este código debe estar basado tanto en la física como en la termodinámica definida en las bases del software que se quiere replicar, en este caso Gasturb.

Para ello, aprovechando la documentación ya existente en distintas webs sobre cómo se debe simular este tipo de software, y por supuesto una primera aproximación particular del 'software', se ha modelado un código en Python que simula de manera básica pero confiable el funcionamiento de este software.

El primer paso es programar el modelo físico-termodinámico en un lenguaje de programación que permita la rápida extracción y procesamiento de datos para obtener una muestra suficientemente grande que permita aplicar la inteligencia artificial con ciertas garantías. Para ello, se utiliza como base un paquete llamado Huracán y que lleva ya embebido este modelo, por supuesto realizando ligeras variaciones ad hoc para el software y los datos de entrada en cuestión para este proyecto específico.

Huracán es un paquete de modelado de motores de código abierto, de 0 dimensiones y orientado a objetos, diseñado para el análisis preliminar y el diseño de motores de admisión de aire, divulgación y propósitos educativos.

Modelado del motor

En el siguiente diagrama se muestra cómo Huracán modela un motor.

Fundamentalmente:

- El trabajo se distribuye a través de los ejes.
- Cada flujo posee una instancia de gas que atraviesa una serie de procesos.

Hipótesis importantes

- Modelo de gas ideal.
- Compresión y expansión adiabáticas.
- Intercambios de calor isobáricos (aunque se puede proporcionar una relación de presión en estos casos).

Ideas clave

- Compartimentalización del modelo de gas, métodos de procesos termodinámicos y clases de componentes.
- Las operaciones de división y fusión de gases se realizan en tiempo de ejecución.
- Las funciones de flujo son asumidas por funciones del sistema en tiempo de ejecución cuando se crea un sistema.

El diagrama de modelado del motor se vería de la siguiente manera:

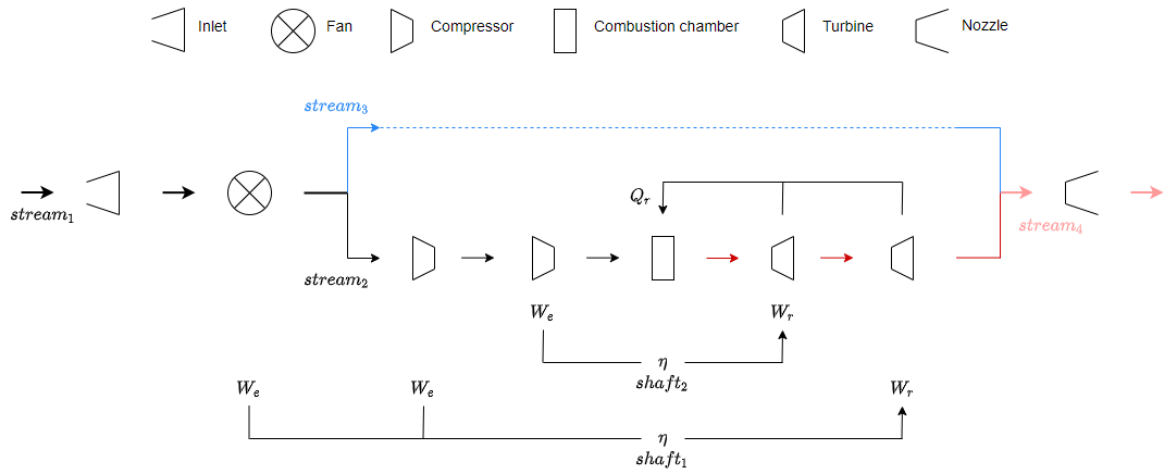

Figura 19: Modelado de la estructura de un motor a través del código de Python que simula el software Gasturb

A continuación, una breve descripción del código y cómo se organiza:

- Importación de Módulos: Se importan varios módulos de la biblioteca 'huracan', como 'shaft', 'inlet', 'fan', 'compressor', 'combustion_chamber', 'turbine', 'nozzle', 'gas', y 'fuel'.
- Definición de Parámetros: Se definen varios parámetros del motor, como el flujo de masa (mf), la velocidad Mach (M), la temperatura (t), la presión (p), y la relación de presión del 'bypass' (bpr). Se define también el combustible (f) utilizando la clase 'fuel' con un valor específico del Poder Calorífico Inferior (LHV).
- Definición de Gas y Fluidos: Se define el gas utilizando la clase 'gas' con parámetros como la masa, el calor específico y la relación de calor específico dependientes de la temperatura. El fluido de trabajo se define como la composición de los gases y combustibles anteriores.
- Creación de Componentes: Se crean instancias de componentes como el inlet, fan, compresores, cámara de combustión, turbina y boquilla, cada uno con sus respectivos parámetros y eficiencias.
- Configuración de Ejes: Se crean dos ejes ('shaft1' y 'shaft2') que agrupan diferentes componentes en secuencias. Cada eje tiene su propia eficiencia y se especifica la eficiencia de la caja de cambios ('eta_gearbox') para el primer eje.
- Definición de Corrientes (Streams): Se crea una corriente ('stream') que representa el flujo principal a través del motor. Esta corriente se divide en corrientes principales y de derivación utilizando la relación de presión del 'bypass'.
- Configuración de la Red de Componentes: Se conectan los componentes en secuencia y se especifica la relación de conexión entre ellos.
- Ejecución de la Simulación: Se ejecuta la simulación del flujo de trabajo de la corriente ('stream.run()').
- Visualización de Resultados: Se generan gráficos para visualizar las propiedades termodinámicas del flujo a lo largo del proceso, como temperatura entropía (T-S), presión-volumen (p-V), entalpía-presión (H-p), y temperatura-presión (T-p).

El siguiente paso es seleccionar la configuración de interés, que como ya se ha indicado previamente consta de un turbofán de 2 ejes o doble flujo, el motor predominante en la aviación comercial, con relaciones de 'bypass' aproximadamente entre 5 y 10.

A continuación, se muestra el código resumen, donde ya se encuentran implícitas todas las leyes físicas y térmicas que definen el comportamiento de este tipo de motor en la realidad, para el siguiente conjunto de los parámetros de entrada.

Estas leyes se definen previamente paso a paso en los distintos módulos y submódulos a los que se hace referencia en el código resumen.

Para un caso concreto, es decir, un caso de prueba con valores predefinidos:

```
#Caso único
from huracan.engine import shaft
from huracan.thermo.fluids import gas, fuel
from huracan.components import inlet, fan, compressor, combustion_chamber, turbine, nozzle
mf = 1440
M  = 0.4
t  = 281.65
p  = 89874
bpr = 9.6
f = fuel(LHV=43e6)
g = gas(mf=mf,cp=lambda T: 1150 if T > 600 else 1000,k=lambda T: 1.33 if T > 600 else 1.4,m=M, t_0=t, p_0=p)

i  = inlet             (PI=0.98)
fn = fan               (eta=0.92, PI=1.54)
c1 = compressor        (eta=0.89, PI=1.54)
c2 = compressor        (eta=0.89, PI=9.86)
cc = combustion_chamber(fuel=f, eta=0.965, PI=0.98)
t1 = turbine           (eta=0.89)
t2 = turbine           (eta=0.89)
n  = nozzle            (eta=1)
shaft1 = shaft(fn, c1, t2, eta=0.98, eta_gearbox=0.975)
shaft2 = shaft(c2, t1, eta=0.98)
stream = g-i-fn
s1core, s1bypass = stream*(bpr/(bpr+1))
s1core-c1-c2-cc-t1-t2
s2 = s1core-s1bypass
s2-n
stream.run()
stream.plot_T_S(show=True, legend=True)
stream.plot_p_V(show=True, legend=True)
stream.plot_H_p(show=True, legend=True)
stream.plot_T_p(show=True, legend=True)
```

Figura 20: Caso de prueba para los parámetros de entrada dados que simula el funcionamiento de un motor turbofán de doble flujo a través del software Gasturb

Ejecutando este código resumen, se obtienen las siguientes gráficas que definen los valores de los parámetros fundamentales que definen las prestaciones de un motor en cada estación:

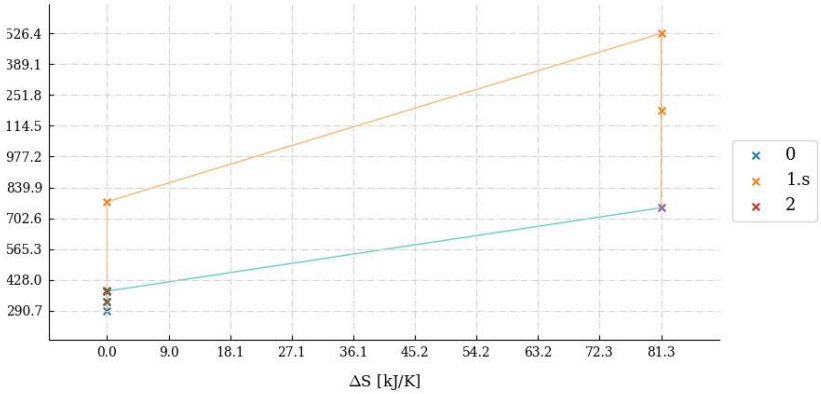

Figura 21: Ciclo termodinámico de un motor turbofán de doble flujo en función de la variación de entropía por estación

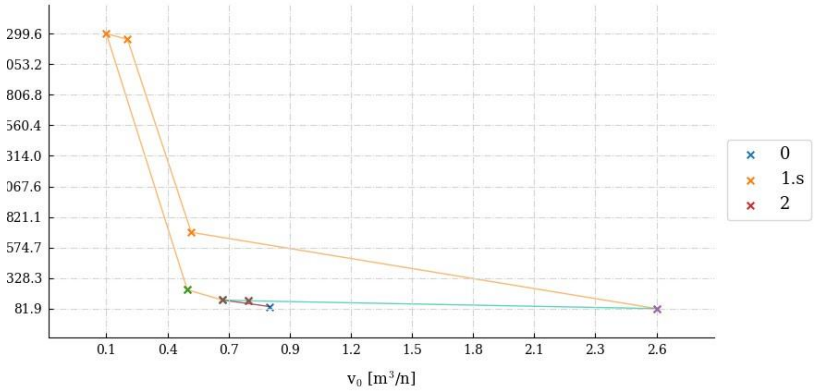

Figura 22: Ciclo termodinámico de un motor turbofán de doble flujo en función de la variación de volumen por estación

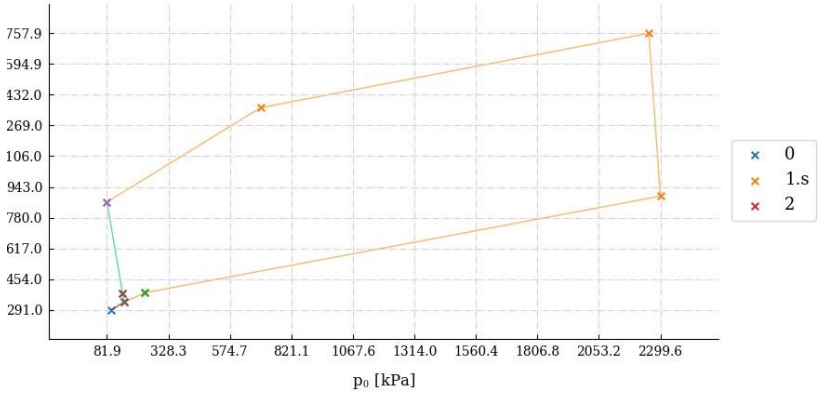

Figura 23: Ciclo termodinámico de un motor turbofán de doble flujo en función de la variación de presión por estación

No obstante, el principal output que se quiere obtener y es el que realmente aporta valor para la extracción de conclusiones sobre el funcionamiento real del motor en función de los distintos parámetros de entrada es la obtención tanto de la temperatura de remanso como de la presión de remanso en cada estación del motor.

A continuación, se muestran estos valores para la iteración correspondiente al caso de prueba del código resumen:

```
0.il     Inlet
          T0 = 290.663         [K]
          p0 = 98,342.012      [Pa]
0.fn     Fan
          T0 = 332.145         [K]
          p0 = 151,446.698     [Pa]
1.s.cp1  Compressor
          T0 = 381.145         [K]
          p0 = 233,227.915     [Pa]
1.s.cp2  Compressor
          T0 = 776.396         [K]
          p0 = 2,299,627.238   [Pa]
1.s.cc   Combustion chamber
          T0 = 1,526.436       [K]
          p0 = 2,253,634.693   [Pa]
1.s.tb1  Turbine
          T0 = 1,183.280       [K]
          p0 = 697,073.895     [Pa]
1.s.tb2  Turbine
          T0 = 749.192         [K]
          p0 = 81,889.154      [Pa]
2.nz     Nozzle
          T0 = 377.634         [K]
          p0 = 144,754.103     [Pa]
```

Figura 24: Output del código de Python que simula el software Gasturb para los parámetros de entrada definidos anteriormente

Para obtener estos datos de salida, primero ha sido necesario definir los datos de entrada sobre los que el software va a aplicar toda la física y termodinámica definida en los distintos módulos y submódulos de los que se compone este código resumen.

Para esta primera iteración con los datos de entrada estáticos, son los siguientes:

```
# Define los valores de la primera iteración
mf = 1440
M  = 0.4
t  = 281.65
p  = 89874
bpr = 9.6
eta_gearbox=0.975
PI_i=0.98
eta_fn=0.92
PI_fn=1.54
#eta_c1=eta_t2
#rta_c2=eta_t1
eta_c1=0.89
PI_c1=1.54
eta_c2=0.89
PI_c2=9.86
eta_cc=0.965
PI_cc=0.98
eta_t1=0.89
eta_t2=0.89
eta_n=1
LHV=43e6
```

Figura 25: Parámetros de entrada necesarios para ejecutar las simulaciones

Así, se a continuación se identifica cada parámetro con la magnitud que representa:

- 'mf' es el gasto másico de aire que entra al motor
- 'M' es el número de Mach
- 't' es la temperatura estática ambiente
- 'p' es la presión estática ambiente
- 'bpr' es la relación de bypass entre el flujo secundario y el flujo primario
- 'eta_gearbox' es el rendimiento del engranaje de los ejes
- 'PI_i es la relación de presiones en el inlet
- 'eta_fn' es el rendimiento del fan o ventilador
- 'PI_fn' es la relación de presiones en el fan o ventilador
- 'eta_c1' es el rendimiento del compresor de baja, LPC
- 'PI_c1' es la relación de presiones en el compresor de baja, LPC
- 'eta_c2' es el rendimiento del compresor de alta, HPC
- 'PI_c2' es la relación de presiones en el compresor de alta, HPC
- 'eta_cc' es el rendimiento de la cámara de combustión
- 'PI_cc' es la relación de presiones en la cámara de combustión
- 'eta_t1' es el rendimiento de la turbina de alta, HPT
- 'eta_t2' es el rendimiento de la turbina de baja, LPT
- 'eta_n' es el rendimiento de la tobera
- 'LHV' es el poder calorífico del combustible

Sabiendo ya generar este conjunto de datos de salida y que posteriormente se podrán utilizar para definir las prestaciones del motor según las leyes físico-térmicas que rigen el funcionamiento de un turbofán, el siguiente paso es programar una serie de bucles que permitan extraer de una manera cómoda y relativamente rápida tanto la variación en los datos de entrada que se introducen en el software como la generación de los datos de salida.

Para ello, se construyen primero conjuntos de datos para cada parámetro que tiene interferencia en el desarrollo del software, tanto los de entrada como de salida, y se asigna una variación porcentual, en este caso aleatoria entre el -10% y el 10%, sobre los datos fijos que se han definido previamente en la primera iteración, mostrados en la página anterior, según sea el tamaño de la propia variable y el rango permitido sobre el que puede desplazarse.

A continuación, se modifica el código del caso de prueba para que recoja estas variaciones y poder iterar sobre el modelo que simula la termo-física de las prestaciones de este tipo de motor:

```
# Crear instancias con los valores de las variables actualizadas
f = fuel(LHV=round(variables_random['LHV'],0))
g = gas(mf=variables_random['mf'],
        cp=lambda T: 1150 if T > 600 else 1000,
        k=lambda T: 1.33 if T > 600 else 1.4,
        m=variables_random['M'],
        t_0=variables_random['t'],
        p_0=variables_random['p'])
i = inlet(PI=variables_random['PI_i'])
fn = fan(eta=variables_random['eta_fn'], PI=variables_random['PI_fn'])
c1 = compressor(eta=variables_random['eta_c1'], PI=variables_random['PI_c1'])
c2 = compressor(eta=variables_random['eta_c2'], PI=variables_random['PI_c2'])
cc = combustion_chamber(fuel=f, eta=variables_random['eta_cc'], PI=variables_random['PI_cc'])
t1 = turbine(eta=variables_random['eta_t1'])
t2 = turbine(eta=variables_random['eta_t2'])
n = nozzle(eta=variables_random['eta_n'])
shaft1 = shaft(fn, c1, t2, eta=variables_random['eta_c1'], eta_gearbox=eta_gearbox)
shaft2 = shaft(c2, t1, eta=variables_random['eta_c2'])
stream = g - i - fn
s1core, s1bypass = stream * (variables_random['bpr'] / (variables_random['bpr'] + 1))
s1core - c1 - c2 - cc - t1 - t2
s2 = s1core - s1bypass
s2 - n
stream.run()
```

Figura 26: Código de Python sobre el que se va a iterar para generar la base de datos que alimente a los algoritmos de aprendizaje automático

Es fundamental definir a continuación unos rangos máximos y mínimos de variación aceptable para cada parámetro, tanto los de entrada como los de salida, y un valor óptimo, para posteriormente poder analizar con cierta agilidad el comportamiento de estos conjuntos de datos en función primero de si están dentro o no de este rango aceptable de variación y segundo cuán cerca o lejos están de ese valor óptimo definido previamente.

Se consigue así que de una forma rápida y ordenada, por supuesto basada en estimaciones lógicas y ejecutadas íntegramente por el propio lenguaje de programación, se asigne una especie de calificación para el valor de cada parámetro por separado y finalmente para los conjuntos de datos de entrada y salida, para poder por fin obtener lo que sería la calificación final de esa iteración o, lo que es lo mismo, de la media ponderada de esos conjuntos de datos y por ende de cada uno de los parámetros que definen el funcionamiento del software.

El criterio de calificación, aunque ya se ha introducido en este mismo apartado, sería:

1. Si el valor de algún parámetro ya sea de entrada o de salida está fuera de los rangos definidos como aceptables, la iteración se califica con un 0 y se desecha.

2. Si el valor de todos los parámetros tanto de entrada como de salida están dentro de los rangos definidos como aceptables, será la distancia entre cada valor y el valor óptimo definido para cada parámetro el que califique del 1 al 10 cada parámetro de esa iteración, siendo un 10 si coincide con dicho valor óptimo y bajando progresivamente la puntuación se vaya alejando de ese valor óptimo, pero siempre dentro de los rangos aceptables.

Los límites de operación de un motor de avión pueden variar según el diseño específico del motor y el fabricante. Los parámetros operativos típicos incluyen temperaturas, presiones, ciclos de vida y gasto másico de aire.

A continuación, se proporcionan rangos generales que podrían aplicarse a motores de avión, si bien estos valores pueden variar según el modelo y la tecnología del motor:

Temperaturas:

- Temperatura de Entrada de Aire (TAT): Puede variar entre -40°C y +60°C.
- Temperatura del Combustor: Depende de la fase de operación del motor o del diseño de la propia cámara, pero un rango aceptable puede ir desde los 1400ºC hasta los 2000ºC.
- Temperatura de la Turbina: Puede variar entre los 900ºC y los 1500ºC.

Presiones:

- Presión de Entrada de Aire (PIT): Varía con la altitud, pero puede oscilar entre 500 hPa y 1000 hPa, es decir, aproximadamente 1 atmósfera
- Presión en el Compresor: Varía entre 2 y 5 atmósferas
- Presión en la Cámara de Combustión: Varía entre 3 y 30 atmósferas, es la zona con mayor presión del motor
- Presión en la Turbina: Varía entre 1 y 2 atmósferas.

Ciclos de Vida: El número de ciclos de vida de un motor puede variar, pero los motores modernos suelen estar diseñados para operar durante miles de horas antes de requerir una revisión o reemplazo importante. Esto puede ser del orden de 20,000 a 50,000 ciclos de vida, pero es importante verificar las especificaciones del fabricante para un motor específico.

Gasto Másico de Aire: El flujo de aire a través del motor, o gasto másico de aire, puede variar según las condiciones de vuelo. Puede oscilar entre 100 kg/s y 300 kg/s, dependiendo del tamaño y tipo de motor.

Estos son valores generales y pueden variar significativamente según el tipo de motor, su aplicación (por ejemplo, motores de turbina, motores de pistón), y el fabricante. Además, los motores modernos están equipados con sistemas de control y monitoreo avanzados que permiten ajustes dinámicos dentro de ciertos límites para adaptarse a diferentes condiciones de vuelo y mantener un rendimiento eficiente y seguro.

En cuanto a otros parámetros fundamentales en el entendimiento de la buena operación de cualquier motor, el empuje específico (TSFC, por sus siglas en inglés) y el consumo específico de combustible (SFC, por sus siglas en inglés) son dos parámetros importantes que describen la eficiencia de un motor de avión.

Empuje Específico (TSFC):

El empuje específico se refiere a la cantidad de empuje que produce un motor en relación con la cantidad de combustible que consume. Se expresa típicamente en términos de libras de empuje por libra de combustible por hora (lb/lbf·hr) o en unidades métricas como Newtons por Newton hora (N/N·s). Valores típicos de TSFC pueden variar según el tipo de motor y la aplicación, pero para motores a reacción modernos, los valores pueden oscilar entre 0.4 y 0.8 lb/lbf·hr (o equivalente en unidades métricas). Un TSFC más bajo indica una mayor eficiencia en términos de consumo de combustible para generar una determinada cantidad de empuje.

Consumo Específico de Combustible (SFC):

El SFC es una medida inversa de la eficiencia del motor y se refiere a la cantidad de combustible consumido por la unidad de tiempo y por la unidad de empuje. Se expresa típicamente en términos de libras de combustible por hora por libra de empuje (lb/hr·lbf) o en unidades métricas como kilogramos de combustible por Newton hora (kg/N·s). Los valores típicos de SFC para motores a reacción suelen estar en el rango de 0.5 a 1.0 lb/hr·lbf. Un SFC más bajo indica una mayor eficiencia, ya que el motor está utilizando menos combustible para generar la misma cantidad de empuje.

Estos valores pueden variar según el tipo de motor, la altitud, la velocidad y otras condiciones de operación. En general, los diseñadores de motores y aeronaves trabajan para mejorar la eficiencia del TSFC y SFC para lograr un mejor rendimiento y menor consumo de combustible.

Además de los parámetros mencionados anteriormente, hay varios otros parámetros que están limitados y controlados durante la operación de un motor de avión para garantizar un rendimiento seguro y eficiente. Algunos de estos parámetros adicionales incluyen:

- Presión de entrada de aire (PIT): Varía según la altitud y puede estar dentro del rango de 500 hPa a 1000 hPa.
- Presión de Escape (Jet Pipe Pressure): La presión en el conducto de escape del motor es monitoreada para asegurar que esté dentro de los límites establecidos. Variaciones inusuales pueden indicar problemas en la salida de gases del motor. Suele tener límites específicos para evitar daños a la salida de gases.
- Temperatura de entrada del aire (TAT): Puede variar entre -40°C y +60°C
- RPM (Revoluciones por Minuto): La velocidad de rotación del eje del motor se mantiene dentro de ciertos límites para evitar daños a los componentes y garantizar un funcionamiento seguro. Los límites de RPM varían según la fase de vuelo y el tipo de motor, pero podrían estar entre 80% y 105% de la RPM nominal.
- Temperatura del Aceite: La temperatura del aceite del motor se supervisa para evitar el sobrecalentamiento y asegurar una adecuada lubricación. Puede tener límites entre 60°C y 150°C, dependiendo del tipo de aceite y las especificaciones del motor.
- Temperatura del Gas de Escape (EGT, por sus siglas en inglés): La temperatura del gas de escape se controla para evitar que alcance niveles que puedan dañar los componentes y para garantizar un rendimiento eficiente. Los límites de EGT pueden variar significativamente, pero podrían estar en el rango de 600°C a 1000°C o más.
- Presión del Compresor: La presión en la etapa de compresión se mantiene dentro de límites seguros para asegurar un flujo de aire adecuado a través del motor.
- Flujo de Combustible: El flujo de combustible hacia el motor es controlado para mantener una mezcla adecuada y evitar problemas de combustión. Los límites del flujo de combustible se establecen para garantizar una combustión eficiente y evitar problemas de sobrecalentamiento.
- Limitaciones de Potencia: Hay límites específicos de potencia establecidos para diversas fases de vuelo, como despegue, crucero y aterrizaje. Estos límites aseguran un rendimiento seguro en diferentes condiciones. Establecidas para diversas fases de vuelo, como despegue, crucero y aterrizaje.
- Velocidad del Aire (Mach): La velocidad del avión en relación con la velocidad del sonido se controla para evitar problemas aerodinámicos y estructurales. Los límites de velocidad Mach pueden variar y se establecen para evitar problemas aerodinámicos y estructurales.

- Vibraciones: Niveles de vibración fuera de lo común pueden indicar desequilibrios en el motor o problemas en los componentes. Límites específicos de vibraciones para garantizar el funcionamiento suave del motor y prevenir desgastes prematuros.
- Monitorización del Sistema de Control (FADEC, por sus siglas en inglés): En motores modernos, los sistemas de control de motores totalmente automáticos (FADEC) supervisan y ajustan continuamente múltiples parámetros para optimizar el rendimiento y la eficiencia.

Es crucial entender que estos límites y parámetros pueden variar entre diferentes modelos de motores y fabricantes. Además, las condiciones operativas específicas, como la altitud, la velocidad y la carga, también pueden influir en los límites establecidos. La información sobre estos límites se encuentra en los manuales de operación y mantenimiento proporcionados por el fabricante del motor y de la aeronave.

Para ello, se definen los rangos permitidos para las variables, tanto las de entrada como las de salida, así como su valor óptimo, para poder calificar de manera proporcional las puntuaciones obtenidas para cada parámetro y posteriormente para cada itetación:

```
# Valores óptimos de los parámetros
valores_optimos = {'t': (300, 250, 350),
                   'p': (101325, 85000, 130000),
                   'bpr': (9, 5, 12),
                   'eta_fn': (0.9, 0.80, 1.0),
                   'eta_c1': (0.95, 0.80, 1.0),
                   'eta_c2': (0.95, 0.80, 1.0),
                   'eta_cc': (0.98, 0.80, 1.1),
                   'eta_t1': (0.96, 0.80, 1.0),
                   'eta_t2': (0.96, 0.80, 1.0),
                   'eta_n':(0.95, 0.80, 1.1),
                   'mf': (1500, 1200, 1800),
                   'PI_i':(1.25, 0.5, 2),
                   'PI_fn': (1.25, 0.5, 2),
                   'PI_c1': (5, 1, 10),
                   'PI_c2': (8.5, 5, 12),
                   'PI_cc': (1.03, 0.9, 1.1),
                   'M': (0.8, 0, 0.85),
                   'LHV': (44e6, 30e6, 60e6),
                   'eta_gearbox': (0.975, 0.95, 1)}

valores_optimos_out = {'i.t0': (300, 250, 350),
                       'i.p0': (95000, 80000, 120000),
                       'fn.t0': (300, 250, 375),
                       'fn.p0': (150000, 100000, 200000),
                       'c1.t0': (450, 300, 600),
                       'c1.p0': (300000, 190000, 400000),
                       'c2.t0': (850, 600, 900),
                       'c2.p0': (2250000, 1500000, 3000000),
                       'cc.t0': (1500, 1200, 1900),
                       'cc.p0': (2250000, 1500000, 3000000),
                       't1.t0': (1250, 900, 1500),
                       't1.p0': (600000, 400000, 850000),
                       't2.t0': (850, 600, 1000),
                       't2.p0': (70000, 30000, 100000),
                       'n.t0': (450, 300, 600),
                       'n.p0': (150000, 100000, 200000)}
```

Figura 27: Definición de los rangos permitidos para cada parámetro, ya sea de entrada o de salida, así como de su valor óptimo

Con todo lo citado con anterioridad, y una vez desarrollado todo el código necesario primero para definir las variaciones aleatorias que se quieren probar en los distintos bucles que iteran sobre los datos de entrada y segundo para asignar una calificación final a los conjuntos de datos que definen cada iteración, se obtiene un listado de calificaciones correspondientes a las distintas iteraciones ejecutadas en el código.

Este número de iteraciones, así como los criterios de variación de cada variable de entrada y la función que califica cada iteración se definen en el propio código y es el que va a definir el tamaño de la muestra de los datos con los que se quiere entrenar posteriormente a los sistemas de inteligencia artificial.

```python
# Número de iteraciones
num_iteraciones = 50000

# Rango de variación
variacion_base = 0.05 # Variación base para parámetros 1 < x < 100000 -> 5%
variacion_minima = 0.01  # Variación mínima para parámetros x < 1 -> 1%
variacion_maxima = 0.1  # Variación máxima para parámetros x > 100000 -> 10%

# Calificación iteraciones
def calcular_puntuacion(parametro, valor_iteracion, valor_optimo, rango_min, rango_max):
    # Calcula la puntuación de un parámetro en una iteración en función de su cercanía al valor óptimo.
    if valor_iteracion < rango_min or valor_iteracion > rango_max:
        return 0  # Valor fuera de rango, puntuación mínima (0)
    else:
        diferencia = abs(valor_optimo - valor_iteracion)
        rango_valores = rango_max - rango_min
        proporcion = diferencia / rango_valores
        puntuacion = 10 - (proporcion * 10)
        puntuacion = max(1, round(puntuacion, 0))  # La puntuación mínima es 1
        return puntuacion
```

Figura 28: Definición del número de iteraciones, la variación aleatoria a aplicar a cada parámetro según su tamaño y la función que calcula la puntuación en función de los rangos mostrados en la figura anterior

```python
# Iterar sobre cada variable de entrada y actualizar su valor
for iteracion in range(num_iteraciones):
    variables_random = dict()
    for var in variables:
        # Determinar la variación aleatoria según el valor del parámetro dentro del rango especificado
        if variables[var] < 1:
            variacion = variacion_minima
        if variables[var] > 100000:
            variacion = variacion_maxima
        else:
            variacion = variacion_base
        cambio_porcentual_aleatorio = random.uniform(-variacion, variacion)
        # Aplicar la variación aleatoria al valor actual
        variable_random = variables[var] * (1 + cambio_porcentual_aleatorio)
        # Constuir un diccionario
        variables_random[var] = variable_random
        #Obtener los valores óptimos para el parámetro actual
        valor_optimo, rango_min, rango_max = valores_optimos[var]
        # Calcular la puntuación para el parámetro actual
        puntuacion_parametro = calcular_puntuacion(var, variable_random, valor_optimo, rango_min, rango_max)
        puntuaciones_parametros[var] = puntuacion_parametro
```

Figura 29: Definición del bucle que asigna variaciones aleatorias a cada parámetro de entrada en función de su tamaño

Con todo esto, ya solo queda organizar la extracción de los datos de salida que se obtienen de cada iteración del código para poder enviar directamente estos paquetes de datos a un sistema de inteligencia artificial que se encargue tanto de aprender a estimar nuevas calificaciones para nuevas iteraciones, ya fuera del código que define el funcionamiento del software, como de encontrar esas relaciones existentes entre los distintos parámetros que definen las actuaciones de un motor, en este caso un turbofán de doble flujo.

```
# Calcular las puntuaciones de cada parámetro y la puntuación total promedio de la iteración
if 0 in puntuaciones_parametros.values():
    puntuaciones_iteraciones.append(0)
else:
    puntuacion_iteracion = round(sum(puntuaciones_parametros.values()) / len(puntuaciones_parametros),3)
    puntuaciones_iteraciones.append(puntuacion_iteracion)

if 0 in puntuaciones_parametros_out.values():
    puntuaciones_iteraciones_out.append(0)
else:
    puntuacion_iteracion_out = round(sum(puntuaciones_parametros_out.values()) / len(puntuaciones_parametros_out),3)
    puntuaciones_iteraciones_out.append(puntuacion_iteracion_out)

if 0 in puntuaciones_parametros.values() or 0 in puntuaciones_parametros_out.values():
    puntuaciones_iteraciones_TOTAL.append(0)
else:
    suma_puntuaciones_parametros_in = sum(puntuaciones_parametros.values())
    suma_puntuaciones_parametros_out = sum(puntuaciones_parametros_out.values())
    num_parametros = len(puntuaciones_parametros) + len(puntuaciones_parametros_out)
    puntuacion_iteracion_TOTAL = round((suma_puntuaciones_parametros_in + suma_puntuaciones_parametros_out) / num_parametros
    puntuaciones_iteraciones_TOTAL.append(puntuacion_iteracion_TOTAL)
```

Figura 30: Definición del criterio para almacenar las puntuaciones de cada iteración en un archivo único tanto para los parámetros de entrada, como los de salida, como los de cada iteración

Para la correcta generación de los 'DataFrame' en los que se va a estructurar y almacenar la información que se obtiene de la ejecución del propio código, es necesario nombrar las columnas de forma que todo sea trazable y se puedan realizar comprobaciones in situ de una manera rápida y sencilla:

```
# Crear un DataFrame con los datos de salida
nombres_columnas_out = ['T0_inlet [K]',
                        'p0_inlet [Pa]',
                        'T0_fan [K]',
                        'p0_fan [Pa]',
                        'T0_compressor1 [K]',
                        'p0_compressor1 [Pa]',
                        'T0_compressor2 [K]',
                        'p0_compressor2 [Pa]',
                        'T0_combustionchamber [K]',
                        'p0_combustionchamber [Pa]',
                        'T0_turbine1 [K]',
                        'p0_turbine1 [Pa]',
                        'T0_turbine2 [K]',
                        'p0_turbine2 [Pa]',
                        'T0_nozzle [K]',
                        'p0_nozzle [Pa]']

df1 = pd.DataFrame(datos_out, columns=nombres_columnas_out)

# Crear un DataFrame con los datos de entrada
nombres_columnas_in = ['mf',
                       'M',
                       't',
                       'p',
                       'LHV',
                       'bpr',
                       'eta_fn',
                       'eta_c1',
                       'eta_c2',
                       'eta_t1',
                       'eta_t2',
                       'eta_n',
                       'eta_cc',
                       'PI_i',
                       'PI_fn',
                       'PI_c1',
                       'PI_c2',
                       'PI_cc']

df2 = pd.DataFrame(datos_in, columns=nombres_columnas_in)
```

Posteriormente, con las columnas ya reconocidas, se procede a concatenar el resultado de todas las iteraciones para su posterior extracción en un solo archivo dependiendo de su naturaleza:

```
# Crear los DataFrames de puntuaciones de parámetros de salida
puntuaciones_in_df = pd.DataFrame([puntuaciones_parametros])
puntuaciones_out_df = pd.DataFrame([puntuaciones_parametros_out])
puntuacion_total_df = pd.DataFrame([puntuaciones_iteraciones_TOTAL])
puntuacion_total_df_transp = puntuacion_total_df.stack().reset_index(drop=True)
puntuaciones_parametros_in_norm_df = pd.DataFrame([puntuaciones_parametros_in_norm_iter], columns=nombres_columnas_in)
puntuaciones_parametros_out_norm_df = pd.DataFrame([puntuaciones_parametros_out_norm_iter], columns=nombres_columnas_out)

# Crear los DataFrames donde se van agrupando las iteraciones
df1_aux = pd.concat([df1_aux, df1])
df2_aux = pd.concat([df2_aux, df2])
puntuaciones_in_df_aux = pd.concat([puntuaciones_in_df_aux, puntuaciones_in_df])
puntuaciones_out_df_aux = pd.concat([puntuaciones_out_df_aux, puntuaciones_out_df])
```

Figura 31: Definición de los DataFrame donde se visualizan con una estructura particular los valores correspondientes a cada iteración

El último paso es calificar con una etiqueta la puntuación de cada iteración para poder aplicar algoritmos de clasificación, que se realiza a través de la siguiente función:

```
# Definir la función para clasificar según la puntuación
def clasificar_puntuacion(puntuacion):
    if puntuacion >= 9:
        return 'Sobresaliente'
    elif puntuacion >= 7 and puntuacion <9:
        return 'Notable'
    elif puntuacion >= 5 and puntuacion <7:
        return 'Bien'
    elif puntuacion >= 3 and puntuacion < 5:
        return 'Deficiente'
    else:
        return 'Muy deficiente'

# Iterar sobre las filas de df_ia y clasificar la columna adicional de df_ia_clasif
df_ia_clasif = pd.DataFrame()
for index, row in df_ia.iterrows():
    puntuacion = row['Puntuación iteración']  # Obtener la puntuación de la fila actual
    etiqueta = clasificar_puntuacion(puntuacion)  # Clasificar la puntuación
    df_ia_clasif.at[index, 'Clasificación'] = etiqueta  # Asignar la etiqueta a la columna 20 de df_ia_clasif
```

Figura 32: Definición de la función que clasifica con una etiqueta la puntuación de cada iteración y las almacena en un archivo único

En el siguiente apartado se explican algunos algoritmos de inteligencia artificial y su aplicación a problemas de este tipo para optimizar la obtención de los resultados y su posterior análisis.

8. Análisis de sensibilidad analítico con IA

En este caso, se opta por probar tanto un algoritmo de regresión numérica para obtener las calificaciones de las iteraciones en función de las calificaciones de los parámetros como un algoritmo de clasificación en función de las 'tags' o etiquetas que resumen la lógica de los criterios técnicos detrás de la calificación de cada iteración para poder establecer conjuntos de calificaciones y los motivos que las sustentan.

Regresión:

Los algoritmos que parten de un conjunto de datos etiquetados se denominan supervisados pues se supone que un "instructor" o supervisor está mostrando al aprendiz los datos de entrenamiento al mismo tiempo que le indica cuál es la respuesta correcta en cada caso. Si el nombre de "supervisado" está o no bien escogido es algo que invita a discusión...

En cualquier caso este tipo de algoritmos son los que más éxito tienen, pues son bien conocidos y es muy sencillo evaluar su rendimiento (partimos de datos etiquetados, podemos dividirlos en los bloques de entrenamiento y prueba, entrenar nuestro modelo y ver hasta qué punto es capaz de predecir las etiquetas del conjunto de pruebas).Tal y como se ha explicado, estos algoritmos tienen como objetivo encontrar la forma de relacionar las características que forman el conjunto de entrenamiento con sus etiquetas, y esta relación, una vez identificada, es la que se utilizará con nuevos datos para predecir sus etiquetas.

El algoritmo k-NN (k-Nearest Neighbors o k-vecinos más próximos) es, probablemente, el algoritmo de Machine Learning más sencillo: "memoriza" los datos de entrenamiento y, a la hora de realizar una predicción para un nuevo dato, sencillamente devuelve el dato de entrenamiento más próximo (o el resultado de considerar los k datos de entrenamiento más próximos).

Este algoritmo es de tipo "aprendizaje basado en instancias" o "aprendizaje no generalizable" pues su objetivo no es desarrollar un modelo que represente los datos de entrenamiento y permita generalizarlo aplicándolo a nuevas muestras.

El algoritmo k-NN puede utilizarse también en escenarios de regresión: una vez identificados los k vecinos de cada punto, en lugar de considerar su clase y establecer un sistema de votación, se considerará el valor que toma la etiqueta para cada uno de ellos y se devolverá como predicción el valor medio de dichos valores.

Al igual que ocurre con su equivalente para escenarios de clasificación, es posible escoger los pesos a aplicar a los valores extraídos de los vecinos, el algoritmo a usar para identificar los k vecinos y la métrica de distancia.

Los algoritmos basados en árboles de decisión son ampliamente usados tanto en análisis de regresión como en clasificación y, en entornos tabulares, son los que frecuentemente ofrecen mejor rendimiento. Básicamente generan una estructura tipo "árbol de decisión" conteniendo preguntas de tipo "if-else" que permiten alcanzar el valor buscado. Una vez construido el árbol a partir del conjunto de datos de entrenamiento, es posible realizar predicciones haciendo pasar a los nuevos puntos por el árbol, respondiendo a cada una de las preguntas.

Una de las grandes ventajas de este tipo de algoritmos es que no son sensibles a la escala de los datos, lo que simplifica la fase de preprocesamiento.

Los árboles de decisión también pueden ser usados en análisis de regresión. El algoritmo aprende de los mismos bloques de características y etiquetas que, en este caso, son números. A la hora de realizar una predicción, se devuelve el valor medio de las etiquetas de las muestras en el nodo.

Gradient Boosting, Gradient Tree Boosting o Gradient Boosted Regression Trees (GBRT), es una familia de algoritmos usados tanto en clasificación como en regresión basados en la combinación de modelos predictivos débiles (weak learners) -normalmente árboles de decisión- para crear un modelo predictivo fuerte. La generación de los árboles de decisión débiles se realiza de forma secuencial, creándose cada árbol de forma que corrija los errores del árbol anterior. Los aprendices suelen ser árboles "poco profundos" (shallow trees), de apenas uno, dos o tres niveles de profundidad, típicamente.

Este tipo de algoritmos suelen ofrecer los mejores resultados en escenarios tabulares, y son especialmente destacables las implementaciones de LightBGM y XGBoost.

Uno de los parámetros de este tipo de argumentos es el learning rate o tasa de aprendizaje, que controla el grado de mejora de un árbol respecto del anterior. Una tasa de aprendizaje pequeña supone una mejora más lenta, pero adaptándose mejor a los datos, lo que se traduce generalmente en mejoras en el resultado a costa de un mayor consumo de recursos.

En análisis de regresión suelen utilizarse como función de impureza el error cuadrático medio o el error absoluto medio. En ambos casos se utiliza el valor medio de la variable objetivo a la que vamos a representar por $\bar{y}$:

$$\bar{y} = \frac{1}{n}\sum y_i$$

Este valor hace referencia a un nodo concreto, de forma que n es el número de muestras del nodo e yi son los valores de la variable objetivo para dichas muestras.

En el caso de trabajar con el error cuadrático medio (mean squared error) la función de impureza viene dada por la siguiente fórmula:

$$mse = \frac{1}{n}\sum (y_n - \bar{y})^2$$

La función de impureza correspondiente al error absoluto medio (mean absolute error) viene dada por:

$$mae = \frac{1}{n}\sum |y_i - \bar{y}|$$

Los modelos lineales asumen la existencia de una relación lineal entre la variable objetivo y las características predictivas:

$\hat{y} = w0 + w1x1 + w2x2 +... + wpxp$

donde $\hat{y}$ es la variable objetivo a predecir, wi son los coeficientes y xi son las características predictivas.

El modelo más sencillo es el de regresión lineal, que aproxima la variable objetivo minimizando la suma de los cuadrados de las desviaciones (método de los mínimos cuadrados).

En todo caso este algoritmo parte de la suposición de que las características predictivas son independientes. Cuando existe una dependencia lineal (colinealidad), los coeficientes de la función de regresión calculados tienden a ser muy elevados, volviéndose la función excesivamente sensible a las variaciones de las características predictivas -frecuentemente aleatorias-, generando valores con una alta varianza (lo que se conoce por sobre-entrenamiento).

La regresión lineal es un método estadístico que se utiliza para modelar la relación entre una variable dependiente y una o más variables independientes. El objetivo es encontrar una relación lineal que describa la variabilidad de la variable dependiente en función de las variables independientes.

- Modelo Matemático: En su forma más básica, el modelo de regresión lineal se representa como: $y=mx+b$ Donde y es la variable dependiente, x es la variable independiente, m es la pendiente de la línea (coeficiente) y b es la intersección con el eje y (también conocida como término de sesgo o intercepto).
- Entrenamiento: Durante el entrenamiento del modelo de regresión lineal, se ajustan los parámetros m y b para minimizar la suma de los cuadrados de las diferencias entre las observaciones reales y las predicciones del modelo.
- Evaluación: La eficacia del modelo de regresión lineal se evalúa mediante métricas como el coeficiente de determinación (R^2), el error cuadrático medio (MSE) o el error absoluto medio (MAE).

Los modelos lineales ordinarios asumen la ya mencionada relación lineal entre las características predictivas y la variable objetivo. Concretamente, entre otras cosas presuponen que la distribución de la variable objetivo es gaussiana. Esto supone, por ejemplo, que un aumento en el valor de una de las características va a suponer una variación equivalente en la respuesta de la función, lo que no siempre es cierto:

La variable dependiente (la variable objetivo) puede no tener una distribución continua: si estamos prediciendo la edad de una persona en función de características físicas, el valor predicho no puede ser negativo, y el valor máximo también tiene un límite, aunque sea menos preciso. O si estamos prediciendo el número de hijos de una pareja, el resultado no puede ser "2.34 hijos". De hecho, la distribución en este último ejemplo va a estar fuertemente desequilibrada hacia la izquierda (la mayoría de las parejas hoy día tienen pocos hijos -o ninguno- frente a las que tienen muchos).

La influencia de las características predictivas en la variable objetivo no tiene por qué ser estrictamente lineal: la relación entre la edad y diferentes indicadores de la salud de un individuo no va a ser siempre lineal. Por ejemplo, la pulsación máxima en reposo a los 80 años no es la mitad de la máxima recomendada a los 40, ni 10 veces menos que la máxima recomendada a los 8 años. Probablemente existen relaciones más complejas que, en una fórmula, se expresaría con potencias o funciones no lineales (logaritmos, etc.). Esta relación es lo que se denomina función de enlace.

Los modelos generales generalizados eliminan estos condicionantes permitiendo distribuciones no gaussianas en la variable objetivo, así como funciones de enlace más complejas.

La regresión lineal es el método más sencillo entre los modelos lineales. Como se ha comentado en la introducción, estima la variable objetivo mediante una combinación lineal de las características predictivas, minimizando la suma de los cuadrados de los residuales, residual sum of squares o RSS (lo que se conoce como técnica de mínimos cuadrados):

$$RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

donde n es el número de muestras, yi es el valor real observado y f(xi) el valor predicho. Esta expresión es la función de coste de la regresión lineal.

El entrenamiento de una regresión lineal es muy sencillo pues existen ecuaciones matemáticas que nos devuelven los parámetros del algoritmo directamente. Así, si la relación entre las variables predictivas y la variable objetivo es la ya comentada:

$\hat{y} = w0 + w1x1 + w2x2 + ... + wpxp$

Se puede expresar esta misma ecuación de la siguiente forma:

$\hat{y} = w^T x$

Donde W es el vector con los coeficientes, WT es la transpuesta de W (resultado de mostrar el vector no verticalmente sino horizontalmente) y X es el vector de características:

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \qquad X = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Usando esta nomenclatura, la función de coste "error cuadrático medio" puede expresarse de la siguiente forma:

$$mse = \frac{1}{n} \sum (W^T X_i - y_i)^2$$

Donde Xi es el vector de características correspondiente a la muestra i-ésima.

Pues bien, la llamada "ecuación normal" nos devuelve los valores del vector W que minimizan esta ecuación:

$$\widehat{W} = (X^T . X)^{-1} X^T y$$

Tal y como se ha comentado, el regresor lineal ofrece la ventaja de que el mínimo de la función de coste puede extraerse directamente de la llamada "ecuación normal". Sin embargo, esta ecuación implica calcular la inversa del resultado de un producto de matrices, lo que resulta inviable si el tamaño de la matriz de características es excesivamente grande.

Como alternativa, el Regresor Descenso de Gradiente Estocástico es entrenado usando como optimizador -tal y como podemos imaginar por su nombre- el algoritmo de Descenso de Gradiente Estocástico, evaluando el gradiente a partir de una muestra escogida aleatoriamente.

Algoritmos de clasificación:

Estos algoritmos se utilizan para identificar patrones y estructuras en los datos y asignar una etiqueta o categoría a cada uno de ellos, lo que resulta útil en una amplia variedad de aplicaciones.

El algoritmo Random Forest (o Bosque Aleatorio en español) fue desarrollado a finales de la década de 1990 por los estadísticos Leo Breiman y Adele Cutler, y publicado en un artículo titulado "Random Forests" publicado en la revista Machine Learning en el año 2001. Desde entonces, Random Forest se ha convertido en uno de los algoritmos de aprendizaje automático más populares y ampliamente utilizados debido a su capacidad para manejar conjuntos de datos grandes y complejos y su buen rendimiento en una amplia variedad de aplicaciones.

A partir de un conjunto de entrenamiento, este algoritmo genera un cierto número de árboles de decisión, siendo entrenado cada uno de ellos con un subconjunto aleatorio de muestras extraídas con reemplazo. Además, durante la división de cada nodo, en lugar de considerar todas las características predictivas para encontrar el mejor criterio de división, solo considera un subconjunto de ellas. En la implementación de Scikit-Learn se calcula el valor medio de las predicciones probabilísticas.

Como consecuencia de la aleatoriedad introducida, el sesgo o error del modelo aumenta ligeramente, pero, gracias a la combinación de los modelos, decrece la varianza compensando el efecto negativo mencionado, dando como resultado un mejor resultado. Otra desventaja de este algoritmo es que el resultado es más difícil de interpretar que el devuelto por un único árbol de decisión.

Random Forest es un algoritmo de aprendizaje supervisado que se puede utilizar para tareas de clasificación y regresión. Es una combinación de múltiples árboles de decisión, donde cada árbol se entrena de forma independiente y luego se combina para realizar predicciones más precisas y robustas.

- Modelo Ensemble: Los árboles de decisión individuales en Random Forest se entrenan utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y diferentes subconjuntos de características. Esto se conoce como ‘bootstrap’ y selección de características aleatorias.
- Votación: En el caso de clasificación, las predicciones de cada árbol se combinan mediante votación (mayoría), mientras que, en la regresión, se promedian las predicciones de cada árbol para obtener la predicción final.
- Beneficios: Random Forest es robusto frente al sobreajuste, puede manejar conjuntos de datos grandes con características diversas, y es menos sensible a valores atípicos y datos ruidosos en comparación con un solo árbol de decisión.
- Parámetros Importantes: Algunos parámetros importantes en Random Forest incluyen el número de árboles en el bosque, la profundidad máxima de los árboles individuales, y el número mínimo de muestras requeridas para dividir un nodo.
- Evaluación: Para la clasificación, la precisión, la sensibilidad, la especificidad y el área bajo la curva ROC (AUC-ROC) son métricas comunes de evaluación. Para la regresión, el error cuadrático medio (MSE) y el coeficiente de determinación (R^2) son métricas comunes.

Los árboles extremadamente aleatorios (Extra-Trees o Extremely Randomized Trees) fueron propuesto por los investigadores Pierre Geurts, Damien Ernst y Louis Wehenkel en 2006. Este algoritmo lleva la aleatoriedad de Random Forest un paso más allá: además de considerar un subconjunto de las características predictivas para cada uno de los árboles a crear, a la hora de escoger una característica y un valor de corte (threshold) para dividir cada nodo, en lugar de escoger el threshold que mejor divida cada característica (y escoger la característica y valor de corte que minimice el criterio de impureza), se genera un valor de corte aleatorio para cada característica planteada, escogiéndose como regla de división el mejor de ellos. Otra diferencia con Random Forest es que las muestras con las que se entrena cada aprendiz se escogen sin reemplazo.

El boosting es una técnica de aprendizaje automático que involucra una familia de "meta-algoritmos" de ensamblado de modelos débiles que tienen como objetivo convertirlos en modelos fuertes. Para ello se entrenan de forma recursiva, asignándoles un peso en función de su eficiencia. Además, los datos analizados reciben también un peso: cada vez que se entrena un modelo, los pesos de los datos se modifican de forma que aquellos que han recibido una predicción incorrecta ven aumentada su importancia relativa. De esta forma, el siguiente modelo puede concentrarse en ellos con el objeto de minimizar el error global.

Existen muchos algoritmos de boosting, siendo AdaBoost el más conocido históricamente, pero en la actualidad la familia de algoritmos Gradient Boosting es la más utilizada y potente, familia que incluye modelos tan conocidos y utilizados como XGBoost y LightGBM.

AdaBoost entrena de forma secuencial un conjunto de aprendices débiles a partir de un algoritmo base común. Todos los aprendices son entrenados con el mismo conjunto de datos, pero éstos van recibiendo pesos que dependen de los errores cometidos por cada aprendiz.

Así, todas las muestras reciben inicialmente un peso inicial wi de 1/n (suponiendo que haya n muestras). El primer aprendiz es entrenado y se estima su tasa de error. Suponiendo que estemos trabajando en un problema de clasificación, esta tasa de error (en general, para el aprendiz j-ésimo) sería:

Tasa de error del aprendiz j-ésimo:

$$r_j = \frac{Suma\ de\ los\ pesos\ de\ las\ muestras\ mal\ clasificadas}{Suma\ de\ todos\ los\ pesos}$$

Si no ha habido muestras mal clasificadas, esta tasa de error será 0. Por el contrario, si todas las muestras han sido mal clasificadas, esta tasa será 1.

A continuación, el aprendiz recibe un peso que será tenido en cuenta al final del proceso para estimar una predicción final. Este peso viene dado por la siguiente función:

Peso recibido por el aprendiz j-ésimo:

$$\alpha_j = \eta . log \frac{1 - r_j}{r_j}$$

A semejanza de lo que ocurre en AdaBoost, Gradient Boosting entrena un número de aprendices débiles en secuencia, intentando que cada uno de ellos se centre en los errores que el anterior haya cometido. La mayor diferencia con AdaBoost es que Gradient Boosting entrena cada aprendiz no en los datos de entrenamiento, sino en los errores residuales cometidos por el aprendiz que lo precede, es decir, en la diferencia entre los valores de la variable objetivo y las predicciones del aprendiz anterior.

El parámetro learning_rate determina el grado de contribución de cada aprendiz al resultado final.

El algoritmo k-Nearest Neighbors (k-NN o k-vecinos más próximos) es un algoritmo de Machine Learning extremadamente simple: en clasificación, asigna una etiqueta de clase a un punto de datos basándose en las etiquetas de clase de los k vecinos más cercanos en el conjunto de entrenamiento. Los vecinos se seleccionan según la distancia euclidiana u otra medida de distancia, y el número k se especifica previamente.

Este algoritmo se basa en la suposición de que los datos con etiquetas semejantes se encuentran cerca unos de otros.

El enfoque usado en k-NN es de tipo "aprendizaje basado en instancias" o "aprendizaje no generalizable" pues su objetivo no es desarrollar un modelo que represente los datos de entrenamiento y permita generalizarlo aplicándolo a nuevas muestras.

Las redes neuronales artificiales (artificial neural networks, ANN) o, simplemente, redes neuronales, son un modelo computacional basado en el comportamiento observado en sus equivalentes biológicos. Están formadas por un conjunto de unidades de procesamiento denominadas neuronas artificiales, unidas unas con otras, que transforman los datos de entrada que llegan a ellas en otros datos de salida. La información que llega a la red neuronal la recorre mientras se transforma en cada neurona artificial atravesada, hasta alcanzar el final de la red, en el que se devuelve el valor resultante.

Las neuronas artificiales que forman una red neuronal pueden ser consideradas versiones modernas de la Adaline. Concretamente, vimos que en la Adaline el valor z (net input o entrada neta) pasaba por una función identidad (es decir, no se modificaba) y era éste el valor que se consideraba para calcular el error de la neurona (dicho valor se pasaba a continuación por una función escalón para devolver una de las dos clases en las que este algoritmo podía clasificar una muestra).

En una neurona artificial moderna, la entrada neta pasa por una función de activación que puede ser una función sigmoidea u otras, y es este valor resultante el que se devuelve como valor de salida, sirviendo también para evaluar el error cometido por la neurona.

Las neuronas artificiales están agrupadas en capas y éstas se dividen en tres tipos:

- Una capa de entrada que, estrictamente hablando, no está formada por neuronas artificiales, simplemente recibe los datos de entrada.
- Capas ocultas, que reciben este nombre porque no son visibles ni desde la entrada ni desde la salida. El número de capas ocultas es variable y depende del objetivo perseguido.
- Una capa de salida que, en función del tipo de escenario en el que nos encontremos (clasificación o regresión) tendrá una o más neuronas.

La información fluye desde la capa de entrada hasta la capa de salida, siendo procesada en las capas ocultas, y la salida de todas las neuronas sirve de entrada para todas las neuronas de la siguiente capa. En arquitecturas más complejas, la información no solo se desplaza desde la entrada hasta la salida (pueden existir realimentaciones, por ejemplo) y, además de neuronas artificiales, pueden existir otros elementos.

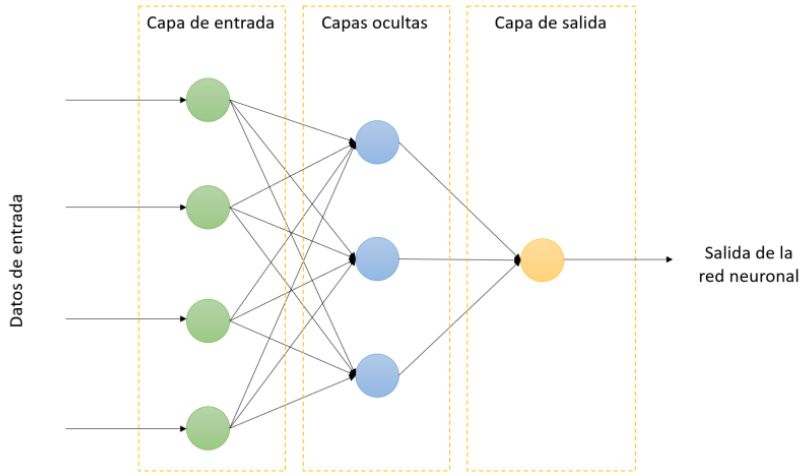

Figura 33: Red neuronal básica que explica los fundamentos detrás de los algoritmos de inteligencia artificial

La arquitectura mencionada al comienzo de esta sección (redes neuronales formadas por capas de neuronas, navegando los datos desde la capa de entrada hasta la capa de salida sin realimentaciones, sirviendo la salida de cada neurona de entrada para todas las neuronas de la siguiente capa) se corresponde con lo que se denomina perceptrón multicapa.

A pesar del nombre, esta arquitectura no está basada en perceptrones, simplemente se ha conservado el término "perceptrón" en el nombre por motivos históricos.

Desde cierto punto de vista, ya se ha visto que es posible distinguir dos escenarios posibles en un análisis de clasificación: un primer escenario en el que queremos clasificar las muestras en dos clases (clasificación binaria) y un segundo escenario en el que queremos clasificar las muestras en más de dos clases (clasificación multiclase).

Algunos algoritmos de clasificación (Random Forest o Naive Bayes) son capaces de realizar su trabajo tanto en entornos de clasificación binaria como de clasificación multiclase. Otros algoritmos, sin embargo, son puros clasificadores binarios (Support Vector Machines o clasificadores lineales). Aun así, hay métodos que nos permiten utilizar estos últimos también en clasificación multiclase. Para esto hay dos estrategias principales: One-versus-All (uno-contra-todos) y One-versus-One (uno-contra-uno).

En resumen, tanto la regresión lineal como Random Forest son algoritmos populares en el campo del aprendizaje automático, cada uno con sus propias características, fortalezas y aplicaciones adecuadas. La elección entre estos algoritmos depende de varios factores, como la naturaleza de los datos, la complejidad del problema y los requisitos de precisión.

El siguiente código carga los datos obtenidos de la ejecución del 'script' de Python previamente desarrollado y que simula la lógica detrás del software Gasturb, los divide en conjuntos de entrenamiento y prueba para regresión y clasificación, entrena modelos de regresión lineal y clasificación de bosque aleatorio, y por último evalúa el rendimiento de ambos modelos.

Regresión Lineal

El rendimiento del modelo se evalúa utilizando el método score del modelo de regresión lineal, que calcula el coeficiente de determinación del modelo en los datos de prueba:

- Cargar datos desde el archivo CSV: Los datos se cargan desde un archivo CSV especificado por la variable archivo utilizando la función pd.read_csv() de pandas.
- Dividir los datos en características (X) y etiquetas (y): Las características se extraen del DataFrame datos y se almacenan en X_regresion, mientras que las etiquetas se almacenan en y_regresion.
- Dividir los datos en conjuntos de entrenamiento y prueba: Se dividen las características y las etiquetas en conjuntos de entrenamiento y prueba utilizando train_test_split() de scikit-learn.
- Inicializar y entrenar el modelo de regresión lineal: Se inicializa un objeto LinearRegression() y se entrena utilizando los datos de entrenamiento (X_train_regresion y y_train_regresion) mediante el método fit().
- Evaluar el rendimiento del modelo de regresión: Se utiliza el método score() para calcular el coeficiente de determinación del modelo en los datos de prueba (X_test_regresion y y_test_regresion), lo que proporciona una medida de la bondad de ajuste del modelo.
- Finalmente, la variable puntuacion_regresion almacena el rendimiento del modelo de regresión lineal en los datos de prueba.

En el contexto de la función train_test_split de scikit-learn, los parámetros test_size y random_state tienen los siguientes significados:

- test_size: Este parámetro determina el tamaño del conjunto de prueba en relación con el tamaño total del conjunto de datos. Un valor de 0.2 indica que el 20% de los datos se reservarán para el conjunto de prueba, mientras que el 80% restante se utilizará para el conjunto de entrenamiento.
- random_state: Este parámetro se utiliza para inicializar el generador de números aleatorios que divide los datos en conjuntos de entrenamiento y prueba. Proporcionar un valor fijo garantiza que cada vez que ejecutes la función train_test_split con el mismo valor de random_state, obtendrás la misma división de datos. Esto es útil para la reproducibilidad de los resultados. Si no se proporciona ningún valor, el conjunto de datos se dividirá de forma aleatoria cada vez que se ejecute la función.

En resumen, test_size=0.2 indica que el 20% de los datos se utilizarán como conjunto de prueba, y random_state=42 asegura que la división de datos sea consistente cada vez que se ejecute el código con el mismo valor de random_state.

Para un algoritmo de regresión lineal, se pueden calcular las siguientes métricas:

- Coeficiente de Determinación (R^2): El coeficiente de determinación es una medida estadística que indica qué tan bien se ajustan los datos a la línea de regresión. Es un valor entre 0 y 1, donde 1 indica un ajuste perfecto y 0 indica que el modelo no explica nada de la variabilidad de los datos.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- Error Cuadrático Medio (ECM): El ECM es la medida promedio de los errores al cuadrado entre los valores predichos por el modelo y los valores reales. Un ECM más bajo indica un mejor ajuste del modelo a los datos.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Error Absoluto Medio (EAM): El EAM es la medida promedio de los errores absolutos entre los valores predichos y los valores reales. Proporciona una idea de la magnitud de los errores en el modelo.

$$EAM = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Coeficientes de Regresión (pendiente e intercepto): Los coeficientes de regresión indican la relación entre las variables independientes y la variable dependiente. El coeficiente de pendiente indica cuánto cambia la variable dependiente en promedio cuando la variable independiente aumenta en una unidad, mientras que el coeficiente de intercepto indica el valor esperado de la variable dependiente cuando todas las variables independientes son cero.

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2} \quad \beta_0 = \bar{y} - \beta_1\bar{x}$$

  Donde β_0 es el intercepto de la línea de regresión, β_1 es el coeficiente de regresión (pendiente), y es la media de los valores de la variable independiente y x es la media de los valores de la variable dependiente.

Regresión múltiple con Random Forest

La regresión múltiple con Random Forest combina dos conceptos clave en el aprendizaje automático: la regresión múltiple y el algoritmo Random Forest.

Regresión Múltiple: La regresión múltiple es una técnica utilizada para predecir el valor de una variable dependiente (también conocida como variable de respuesta) basándose en dos o más variables independientes (predictoras). En lugar de predecir solo una variable, como en la regresión lineal simple, la regresión múltiple permite modelar relaciones más complejas entre múltiples variables.

La ecuación general de la regresión múltiple es:

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + ... + b_n * x_n + \varepsilon$$

Donde:

- y es la variable dependiente que queremos predecir.
- $x_1, x_2, ..., x_n$ son las variables independientes.
- $b_0, b_1, b_2, ..., b_n$ son los coeficientes que representan la contribución de cada variable independiente a la variable dependiente.
- ε es el término de error, que representa la diferencia entre la predicción del modelo y el valor real.

Random Forest: Random Forest es un algoritmo de aprendizaje automático supervisado que se utiliza tanto para tareas de clasificación como de regresión. Se basa en el concepto de un conjunto de árboles de decisión.

- Árbol de decisión: Es una estructura de árbol donde cada nodo interno representa una característica (o atributo), cada rama representa una decisión basada en esa característica, y cada hoja representa el resultado de la decisión (una clase en problemas de clasificación o un valor en problemas de regresión).
- Random Forest: Consiste en una colección de árboles de decisión. Cada árbol se construye utilizando una muestra aleatoria de los datos de entrenamiento y un subconjunto aleatorio de las características. Luego, para realizar una predicción, se promedian las predicciones de todos los árboles en el bosque (para problemas de regresión) o se realiza una votación entre los árboles (para problemas de clasificación).

Cuando se aplica Random Forest a un problema de regresión múltiple, cada árbol del bosque se ajusta utilizando un subconjunto aleatorio de las características y las predicciones de todos los árboles se promedian para obtener la predicción final. Esto puede capturar relaciones no lineales y complejas entre las variables predictoras y la variable de respuesta.

En resumen, la regresión múltiple con Random Forest es una técnica poderosa para predecir variables continuas, utilizando múltiples variables predictoras, aprovechando la capacidad de Random Forest para manejar relaciones no lineales y complejas en los datos.

Para un algoritmo de regresión múltiple con Random Forest, Para un algoritmo de regresión lineal, se pueden calcular las siguientes métricas:

- Coeficiente de Determinación (R^2): El coeficiente de determinación es una medida estadística que indica qué tan bien se ajustan los datos a la línea de regresión. Es un valor entre 0 y 1, donde 1 indica un ajuste perfecto y 0 indica que el modelo no explica nada de la variabilidad de los datos.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- Error Cuadrático Medio (ECM): El ECM es la medida promedio de los errores al cuadrado entre los valores predichos por el modelo y los valores reales. Un ECM más bajo indica un mejor ajuste del modelo a los datos.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Error Absoluto Medio (EAM): El EAM es la medida promedio de los errores absolutos entre los valores predichos y los valores reales. Proporciona una idea de la magnitud de los errores en el modelo.

$$EAM = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Coeficientes de Regresión (pendiente e intercepto): Los coeficientes de regresión indican la relación entre las variables independientes y la variable dependiente. El coeficiente de pendiente indica cuánto cambia la variable dependiente en promedio cuando la variable independiente aumenta en una unidad, mientras que el coeficiente de intercepto indica el valor esperado de la variable dependiente cuando todas las variables independientes son cero.

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \beta_0 = \bar{y} - \beta_1\bar{x}$$

Donde β_0 es el intercepto de la línea de regresión, β_1 es el coeficiente de regresión (pendiente), y es la media de los valores de la variable independiente y x es la media de los valores de la variable dependiente.

- Vector importancia de las características: Vector con todos los parámetros de entrada que indica la importancia de cada variable a la hora de producir cambios significativos en el resultado final de cada iteración.

$$\text{Importancia}_i = \frac{\text{Suma de la disminución de la impureza de las divisiones que usan la característica } i}{\text{Número total de árboles en el bosque}}$$

Un algoritmo de regresión múltiple busca modelar la relación entre múltiples variables independientes (predictoras) y una variable dependiente (objetivo) continua. El objetivo principal es predecir o explicar el valor de la variable dependiente en función de las variables independientes. Algunos de los objetivos específicos que se buscan lograr con un algoritmo de regresión múltiple incluyen:

- Predicción: Utilizar el modelo entrenado para predecir el valor de la variable dependiente para nuevas observaciones basadas en los valores de las variables independientes.
- Identificación de variables importantes: Determinar qué variables independientes tienen una mayor influencia en la variable dependiente y cuáles son menos significativas. Esto puede ser útil para comprender los factores clave que afectan el fenómeno estudiado.
- Evaluación de relaciones: Analizar la relación entre las variables independientes y la variable dependiente, incluyendo la dirección y la magnitud de la relación. Esto puede proporcionar información sobre cómo las diferentes variables afectan el resultado.
- Control y optimización: En algunos casos, los modelos de regresión múltiple se utilizan para controlar u optimizar un proceso. Por ejemplo, en ingeniería o manufactura, se pueden usar para optimizar la producción minimizando costos o maximizando la eficiencia.
- Inferencia estadística: Utilizar el modelo para realizar inferencias sobre la población subyacente a partir de la cual se obtuvieron los datos de muestra. Esto puede incluir pruebas de hipótesis sobre los coeficientes de regresión para determinar si son estadísticamente significativos.

En resumen, un algoritmo de regresión múltiple se utiliza para modelar y comprender la relación entre múltiples variables, lo que permite hacer predicciones, identificar factores importantes, realizar inferencias estadísticas y optimizar procesos, entre otros objetivos.

‘Cross Validation’ o Validación Cruzada

La validación cruzada, también conocida como ‘cross-validation’ en inglés, es una técnica utilizada en el aprendizaje automático (inteligencia artificial) para evaluar el rendimiento de un modelo de manera robusta y evitar el sobreajuste. La idea principal detrás de la validación cruzada es dividir el conjunto de datos en varios subconjuntos más pequeños (llamados "folds"), y luego entrenar y evaluar el modelo varias veces, utilizando diferentes combinaciones de estos subconjuntos como datos de entrenamiento y prueba. Esto proporciona una evaluación más sólida del rendimiento del modelo, ya que se promedian los resultados de múltiples iteraciones.

El proceso típico de validación cruzada sigue estos pasos:

- Dividir los datos: Se divide el conjunto de datos en K subconjuntos (folds) aproximadamente iguales.
- Entrenar y evaluar: Se entrena el modelo K veces. En cada iteración, un fold diferente se utiliza como conjunto de prueba y los K-1 folds restantes se utilizan como conjunto de entrenamiento. Luego, se evalúa el modelo utilizando el conjunto de prueba.
- Promedio de resultados: Se calcula el rendimiento del modelo para cada iteración (por ejemplo, precisión, puntaje F1, etc.) y se promedian los resultados para obtener una estimación general del rendimiento del modelo.

Los métodos de validación cruzada más comunes incluyen la validación cruzada de k-fold (donde K es un número predefinido), la validación cruzada de Leave-One-Out (LOO), y la validación cruzada estratificada, entre otros.

La validación cruzada en el ámbito de la inteligencia artificial tiene varios objetivos principales, así como diferentes tipos que se adaptan a diferentes situaciones y conjuntos de datos. Aquí hay una explicación más detallada:

Objetivos de la Validación Cruzada:

- Estimación del rendimiento del modelo: La validación cruzada proporciona una estimación más precisa y confiable del rendimiento de un modelo en datos no vistos. Esto es crucial para comprender cómo se generaliza el modelo más allá de los datos de entrenamiento.
- Prevención del sobreajuste: Al evaluar el modelo en múltiples conjuntos de datos de prueba, la validación cruzada ayuda a identificar si el modelo está sobreajustando los datos de entrenamiento y si su rendimiento es consistente en diferentes particiones de los datos.
- Selección de hiperparámetros: La validación cruzada se utiliza para seleccionar los mejores hiperparámetros para un modelo. Al probar diferentes combinaciones de hiperparámetros en diferentes divisiones de los datos, se puede encontrar la configuración que maximice el rendimiento del modelo.

Tipos de Validación Cruzada:

- Validación cruzada de k-fold: Divide el conjunto de datos en k subconjuntos (folds) de tamaño aproximadamente igual. Luego, se entrena el modelo k veces, utilizando cada fold como conjunto de prueba una vez y los k-1 folds restantes como conjunto de entrenamiento en cada iteración.
- Validación cruzada de Leave-One-Out (LOO): Es una variante de la validación cruzada k-fold donde k es igual al número total de muestras en el conjunto de datos. En cada iteración, se entrena el modelo con todos los datos excepto uno, que se utiliza como conjunto de prueba. Este proceso se repite hasta que todas las muestras han sido utilizadas como conjuntos de prueba.
- Validación cruzada estratificada: Es similar a la validación cruzada k-fold, pero garantiza que las proporciones de las clases en cada fold sean lo más similares posible a las del conjunto de datos original. Esto es útil cuando se trabaja con conjuntos de datos desbalanceados.

- Validación cruzada aleatoria: En lugar de dividir los datos en k folds secuenciales, los datos se dividen aleatoriamente en cada iteración. Esto puede ser útil cuando la distribución de los datos no es uniforme o cuando se desea una mayor aleatoriedad en la selección de los subconjuntos de entrenamiento y prueba.

Cada tipo de validación cruzada tiene sus propias ventajas y desventajas, y la elección del método adecuado depende del tamaño del conjunto de datos, la naturaleza de los datos y los objetivos específicos del modelo.

Algunas de las métricas se evaluación del modelo que se pueden obtener de un algoritmo de clasificación son:

- Matriz de confusión: La matriz de confusión es una tabla que describe el rendimiento del modelo de clasificación en términos de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN). Proporciona información detallada sobre el rendimiento del modelo para cada clase de salida.
- Precisión o exactitud (Precision): La precisión es la proporción de verdaderos positivos (TP) respecto a todos los valores positivos predichos por el modelo. La precisión es útil cuando el costo de los falsos positivos es alto.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Recall (Sensibilidad o True Positive Rate): Recall es la proporción de verdaderos positivos (TP) respecto a todos los valores positivos reales. Recall es útil cuando el costo de los falsos negativos es alto.

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

- F1-Score: El F1-Score es la media armónica de precisión y recall. Es útil cuando se desea una métrica única que combine precisión y recall.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- ROC Curve (Curva ROC): La curva ROC es una representación gráfica del rendimiento del modelo de clasificación en diferentes umbrales de decisión. Se traza la tasa de verdaderos positivos (TPR) contra la tasa de falsos positivos (FPR) para varios umbrales de clasificación. El área bajo la curva ROC (AUC-ROC) es una medida de la capacidad de discriminación del modelo.
- Curva PR (Precision-Recall): Similar a la curva ROC, la curva PR muestra la relación entre precisión y recall para diferentes umbrales de decisión. Es especialmente útil cuando las clases están desbalanceadas.
- FPR (False Positive Rate): Es la proporción de negativos que se clasificaron incorrectamente como positivos. Se calcula como el cociente entre los falsos positivos y la suma de los verdaderos negativos y los falsos positivos.

$$\text{FPR} = \frac{FP}{FP + TN}$$

- TPR (True Positive Rate): También conocido como sensibilidad o recall, es la proporción de positivos que se clasificaron correctamente como positivos. Se calcula como el cociente entre los verdaderos positivos y la suma de los verdaderos positivos y los falsos negativos.

$$\mathrm{TPR} = \frac{TP}{TP + FN}$$

- AUC-ROC (Area Under the Receiver Operating Characteristic Curve): Es el área bajo la curva ROC (Receiver Operating Characteristic). La curva ROC es una representación gráfica de la sensibilidad frente a la tasa de falsos positivos para diferentes umbrales de clasificación. AUC-ROC proporciona una medida agregada del rendimiento del modelo en todos los umbrales de clasificación.

En el contexto de la validación cruzada, estas métricas se calcularían para cada iteración de entrenamiento y prueba y luego se promediarían para obtener una evaluación general del rendimiento del modelo.

La validación cruzada es una técnica fundamental en el desarrollo de modelos de aprendizaje automático, ya que proporciona una evaluación más confiable del rendimiento del modelo y ayuda a identificar problemas de sobreajuste.

'Tree Decision Classifier' o Árbol de decisión

Un clasificador de árbol de decisiones es un algoritmo popular de aprendizaje automático utilizado para tareas de clasificación. Funciona al dividir recursivamente los datos en subconjuntos basados en los valores de las características de entrada. En cada paso, el algoritmo selecciona la característica que mejor separa los datos en los subconjuntos más puros posibles en términos de la variable objetivo (la que se está prediciendo). El proceso continúa hasta que se cumple un criterio de detención, como alcanzar una profundidad máxima, tener un número mínimo de muestras por hoja o cuando dividir más no proporciona una mejora significativa.

Un esquema básico del proceso sería el siguiente:

- Selección de la mejor característica: El algoritmo examina cada característica y selecciona la que mejor separa los datos en clases distintas. La medida utilizada para determinar la característica "mejor" varía (por ejemplo, impureza de Gini, entropía).
- División de los datos: Una vez seleccionada la mejor característica, los datos se dividen en subconjuntos basados en los posibles valores de esa característica.
- Partición recursiva: Los pasos anteriores se repiten recursivamente para cada subconjunto hasta que se cumple un criterio de detención. Esto puede implicar alcanzar una profundidad máxima, tener un número mínimo de muestras por hoja o cuando dividir más no proporciona una mejora significativa en la pureza.
- Nodos hoja: En algún momento, el proceso resulta en nodos hoja que representan las etiquetas de clase finales o probabilidades.

Los árboles de decisión son atractivos porque son fáciles de entender y visualizar. Sin embargo, tienden a sobreajustarse a los datos de entrenamiento si no se controlan adecuadamente. Técnicas como la poda (eliminación de ramas que no proporcionan un poder predictivo significativo) y el uso de métodos de conjunto como Bosques Aleatorios o Reforzamiento del Gradiente pueden mitigar este problema.

Las métricas empleadas son las mismas que para el algoritmo de 'cross validation' al tratarse de un problema de clasificación.

Con todo esto, en este proyecto se construyen 4 casos de análisis para distintos algoritmos de inteligencia artificial: regresión lineal, regresión múltiple con Random Forest, clasificador de árbol de decisión con 'TreeDecisionClassifier' y 'Cross Validation' con la variante k-fold.

9. Optimización de la simulación con IA

En este apartado, se pretende construir y analizar los 3 tipos diferentes de algoritmos de inteligencia artificial introducidos en el punto anterior y aplicarlos de manera práctica sobre los conjuntos de datos obtenidos a través de la ejecución del código que simula el software de motor.

Para ello, a partir del archivo construido expresamente para proceder a la implementación de los algoritmos de inteligencia artificial, que se llama 'datos_2024-03-21_ia.csv' y que será genérico para todos los algoritmos, se alimenta a dichos algoritmos con la combinación de los valores de los parámetros de entrada y la clasificación de la iteración completa a través de la etiqueta y cuyo criterio se define en la función de clasificación de la puntuación de cada iteración.

Regresión lineal

1. Primero, se importan las bibliotecas necesarias, incluyendo pandas para la manipulación de datos, numpy para operaciones numéricas, y varias clases y funciones de scikit-learn, como train_test_split para dividir los datos, LinearRegression para el modelo de regresión lineal, y diversas métricas de evaluación de modelos.
2. La función aplicar_modelo_regresion() toma como argumento un nombre de archivo CSV y carga los datos desde ese archivo utilizando pandas.
3. Se separan las características (X) de las etiquetas (y). Las características son todas las columnas excepto la última, y la etiqueta es la última columna.
4. Se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split(). Aquí, se asigna el 80% de los datos para entrenamiento y el 20% para prueba, y se establece una semilla aleatoria para reproducibilidad.
5. Se crea una instancia del modelo de regresión lineal y se ajusta a los datos de entrenamiento utilizando el método fit().
6. Se calcula la puntuación del modelo de regresión utilizando el conjunto de prueba.
7. Se realizan predicciones sobre el conjunto de prueba utilizando el modelo entrenado.
8. Se calculan el Error Cuadrático Medio (ECM) y el Error Absoluto Medio (EAM) para evaluar la precisión del modelo.
9. Se obtienen los coeficientes de la regresión lineal y se calcula el coeficiente de determinación R^2 para evaluar la bondad del ajuste del modelo.
10. La función devuelve varios valores, incluida la puntuación de la regresión, las predicciones, los conjuntos de entrenamiento y prueba, las métricas de evaluación y los coeficientes de la regresión lineal.
11. Se aplica la función aplicar_modelo_regresion() al archivo especificado y se asignan los valores de retorno a variables.
12. Finalmente, se imprimen en la consola la puntuación de la regresión lineal, los parámetros de entrada, las predicciones, el ECM, el EAM, los coeficientes de la regresión lineal y el coeficiente de determinación R^2.

```python
# Algoritmo de Regresión Lineal
!pip install --upgrade scikit-learn
import pandas as pd
import numpy as np
import datetime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, r2_score, mean_squared_error, mean_absolute_error
from datetime import datetime

fecha_actual = datetime.now().strftime('%Y-%m-%d')

archivo = 'datos_2024-03-21_ia.csv'
# archivo = f'datos_{fecha_actual}_ia.csv'

def aplicar_modelo_regresion(csv_file):
    # Cargar datos desde el archivo CSV
    datos = pd.read_csv(csv_file)

    # Dividir los datos en características (X) y etiquetas (y) para la regresión
    X_regresion = datos.iloc[:, :-1]  # Selecciona todas las filas y todas las columnas excepto la última
    y_regresion = datos.iloc[:, -1] # Selecciona todas las filas y la última columna

    # Dividir los datos en conjuntos de entrenamiento y prueba para regresión
    X_train_regresion, X_test_regresion, y_train_regresion, y_test_regresion = train_test_split(X_regresion, y_regresion, test_si
    # Inicializar y entrenar el modelo de regresión lineal
    modelo_regresion = LinearRegression()
    modelo_regresion.fit(X_train_regresion, y_train_regresion)

    # Evaluar el rendimiento del modelo de regresión
    puntuacion_regresion = modelo_regresion.score(X_test_regresion, y_test_regresion)

    # Realizar predicciones en el conjunto de prueba
    predicciones = modelo_regresion.predict(X_test_regresion)

    # Calcular el Error Cuadrático Medio (ECM)
    ecm = mean_squared_error(y_test_regresion, predicciones)

    # Calcular el Error Absoluto Medio (EAM)
    eam = mean_absolute_error(y_test_regresion, predicciones)

    # Coeficientes de Regresión (pendiente e intercepto)
    pendiente = modelo_regresion.coef_
    intercepto = modelo_regresion.intercept_

    # Coeficiente de determinación R2
    r2 = r2_score(y_test_regresion, predicciones)

    return puntuacion_regresion, predicciones, y_train_regresion, y_test_regresion, X_train_regresion, X_test_regresion, ecm, eam

# Aplicar modelo regresión
puntuacion_regresion, predicciones, y_train_regresion, y_test_regresion, X_train_regresion, X_test_regresion, ecm, eam, pendiente
```

Figura 34: Algoritmo de Regresión Lineal a partir de los datos obtenidos con las distintas iteraciones del código anterior

En cuanto a los resultados obtenidos:

```
Parámetros de entrada:
               mf         M           t            p           LHV  \
33553  1414.001695  0.383604  280.707835  86951.332464  4.321065e+07
9427   1398.527311  0.398328  294.290529  92858.695586  4.479874e+07
199    1406.303814  0.407429  289.655398  93037.682268  4.418715e+07
12447  1391.583369  0.407243  293.965061  88029.575582  4.683256e+07
39489  1417.688897  0.380657  291.562878  93609.878981  4.516809e+07
...            ...       ...         ...           ...           ...
28567  1414.967096  0.389748  293.087783  91413.058937  4.406109e+07
25079  1408.359290  0.392048  275.982285  90755.106945  4.404879e+07
18707  1380.635855  0.391960  292.168284  87443.890839  4.049172e+07
15200  1423.690020  0.404516  282.784230  86815.866319  4.209907e+07
5857   1427.293881  0.406134  293.227800  93253.489637  4.605053e+07

            bpr    eta_fn    eta_c1    eta_c2    eta_t1    eta_t2     eta_n  \
33553  9.249570  0.927810  0.845763  0.928619  0.894917  0.904826  0.991352
9427   9.328672  0.923422  0.897460  0.851762  0.854439  0.883608  0.946371
199    9.327974  0.904805  0.856921  0.859659  0.912500  0.889077  0.990569
12447  9.391599  0.890220  0.900687  0.885229  0.917003  0.863506  0.928407
39489  9.910586  0.890167  0.860487  0.912581  0.904824  0.859870  0.926455
...         ...       ...       ...       ...       ...       ...       ...
28567  9.828796  0.961811  0.860656  0.915177  0.928644  0.897342  0.991998
25079  9.809950  0.887997  0.882583  0.910390  0.860845  0.875761  0.985797
18707  9.200019  0.946463  0.855667  0.906650  0.911382  0.847492  0.905400
15200  9.760081  0.963508  0.876207  0.848054  0.850104  0.930165  0.907582
5857   9.279352  0.879251  0.878362  0.921112  0.853493  0.853699  0.922828

          eta_cc      PI_i     PI_fn     PI_c1      PI_c2     PI_cc
33553  0.993503  0.982931  1.506864  1.519560   9.601037  1.014735
9427   0.942153  0.982591  1.479337  1.568122  10.035366  0.946533
199    0.987945  0.942848  1.469071  1.574265   9.523862  0.993623
12447  0.992468  1.025895  1.490787  1.583123  10.302864  1.027542
39489  0.991998  0.944793  1.616752  1.566297   9.756668  0.975476
...         ...       ...       ...       ...        ...       ...
28567  0.967391  0.935653  1.555429  1.583074  10.095759  0.937994
25079  0.994137  0.941742  1.552014  1.516373   9.632460  1.027653
18707  0.923141  1.025558  1.501905  1.477920   9.589392  1.022564
15200  0.919069  0.936123  1.514570  1.482772  10.255433  0.968266
5857   0.988854  1.018513  1.546088  1.594737   9.647571  0.975291

[10000 rows x 18 columns]
Predicciones de la regresión lineal:
 [6.84256523 6.55855238 7.48572586 ... 6.12701394 5.49525181 7.00841703]
Error Cuadrático Medio (ECM): 0.6760703514256191
Error Absoluto Medio (EAM): 0.5970618571448968
Coeficiente de Pendiente: [ 3.66156680e-03  1.86069473e+00  6.49757888e-02  8.62332186e-06
  2.32005689e-08 -5.55382079e-01 -4.95067068e+00  1.26097046e+01
  3.52000682e+00  8.10029683e+00  1.37079559e+01 -1.19834305e-02
  7.65961465e+00 -2.23397752e+00 -2.39515530e+00  1.81542655e+00
 -3.02723582e-01  8.73968092e+00]
Coeficiente de Intercepto: -53.409457532484275
Coeficiente de determinación (R^2) de las predicciones: 0.541639849728933
```

Figura 35: Resultados y métricas correspondientes a la aplicación del modelo de Regresión Lineal a los datos obtenidos a partir de las iteraciones del código anterior

Predicciones del modelo de regresión múltiple (Random Forest): Son las predicciones realizadas por el modelo de regresión múltiple utilizando el algoritmo Regresión Lineal. Cada número en esta lista corresponde a la predicción para una observación en el conjunto de prueba.

Error Cuadrático Medio (ECM): Este valor, 0.676 en este caso, representa la magnitud promedio de los errores al cuadrado entre las predicciones del modelo y los valores reales. Cuanto más bajo sea el ECM, mejor será el ajuste del modelo a los datos. En este caso, un ECM de 0.676 indica que, en promedio, las predicciones del modelo están desviadas de los valores reales en una cantidad cuadrada de aproximadamente 0.676 unidades.

Error Absoluto Medio (EAM): Este valor, 0.597 en este caso, representa la magnitud promedio de los errores absolutos entre las predicciones del modelo y los valores reales. Similar al ECM, cuanto más bajo sea el EAM, mejor será el ajuste del modelo a los datos. Un EAM de 0.597 indica que, en promedio, las predicciones del modelo están desviadas de los valores reales en aproximadamente 0.597 unidades.

Coeficiente de Pendiente: Estos son los coeficientes asociados a cada una de las características utilizadas en el modelo de regresión lineal. Cada coeficiente representa el cambio esperado en la variable de respuesta por unidad de cambio en la correspondiente variable predictora, manteniendo todas las demás variables constantes.

Coeficiente de Intercepto: Este valor, -53.4094 en este caso, representa el valor esperado de la variable de respuesta cuando todas las variables predictoras son cero. Es el punto donde la línea de regresión corta el eje y.

Coeficiente de determinación (R^2) de las predicciones: Este valor, 0.5416 en este caso, indica la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Cuanto más cercano sea R^2 a 1, mejor será el modelo en la explicación de la variabilidad de los datos. En este caso, un R^2 de 0.5416 indica que aproximadamente el 54.16% de la variabilidad en la variable de respuesta puede ser explicada por el modelo.

Regresión múltiple con Random Forest

1. Primero, se importan las bibliotecas necesarias, incluyendo pandas para la manipulación de datos, numpy para operaciones numéricas, y varias clases y funciones de scikit-learn, como train_test_split para dividir los datos, RandomForestRegressor para el modelo de regresión con Random Forest, y diversas métricas de evaluación de modelos.
2. La función aplicar_modelo_regresion() toma como argumento un nombre de archivo CSV y carga los datos desde ese archivo utilizando pandas.
3. Se separan las características (X) de las etiquetas (y). Las características son todas las columnas excepto la última, y la etiqueta es la última columna.
4. Se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split(). Aquí, se asigna el 80% de los datos para entrenamiento y el 20% para prueba, y se establece una semilla aleatoria para reproducibilidad.
5. Se crea una instancia del modelo de regresión Random Forest y se ajusta a los datos de entrenamiento utilizando el método fit().
6. Se calcula la puntuación del modelo de regresión utilizando el conjunto de prueba.
7. Se realizan predicciones sobre el conjunto de prueba utilizando el modelo entrenado.
8. Se calculan el Error Cuadrático Medio (ECM) y el Error Absoluto Medio (EAM) para evaluar la precisión del modelo.
9. Se obtienen las importancias de las características, que indican la contribución relativa de cada característica a la predicción.
10. Se calcula el coeficiente de determinación R^2 para evaluar la bondad del ajuste del modelo.
11. La función devuelve varios valores, incluida la puntuación de la regresión, las predicciones, los conjuntos de entrenamiento y prueba, las métricas de evaluación, las importancias de las características, el coeficiente de determinación R^2 y el modelo de regresión.
12. Se aplica la función aplicar_modelo_regresion() al archivo especificado y se asignan los valores de retorno a variables.
13. Finalmente, se imprimen en la consola la puntuación del modelo de regresión, los parámetros de entrada, las predicciones, el ECM, el EAM, el vector importancia de las características y el coeficiente de determinación R^2.

```python
# Algortimo de Regresión Múltiple con Random Forest
!pip install --upgrade scikit-learn
import pandas as pd
import numpy as np
import datetime
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from datetime import datetime

fecha_actual = datetime.now().strftime('%Y-%m-%d')

archivo = 'datos_2024-03-21_ia.csv'
# archivo = f'datos_{fecha_actual}_ia.csv'

def aplicar_modelo_regresion(csv_file):
    # Cargar datos desde el archivo CSV
    datos = pd.read_csv(csv_file)

    # Dividir los datos en características (X) y etiquetas (y) para la regresión
    X_regresion = datos.iloc[:, :-1]  # Selecciona todas las filas y todas las columnas excepto la última
    y_regresion = datos.iloc[:, -1] # Selecciona todas las filas y la última columna

    # Dividir los datos en conjuntos de entrenamiento y prueba para regresión
    X_train_regresion, X_test_regresion, y_train_regresion, y_test_regresion = train_test_split(X_regresion, y_regresion, test_s

    # Inicializar y entrenar el modelo de regresión múltiple con Random Forest
    modelo_regresion = RandomForestRegressor(random_state=42)
    modelo_regresion.fit(X_train_regresion, y_train_regresion)

    # Evaluar el rendimiento del modelo de regresión
    puntuacion_regresion = modelo_regresion.score(X_test_regresion, y_test_regresion)

    # Realizar predicciones en el conjunto de prueba
    predicciones = modelo_regresion.predict(X_test_regresion)

    # Calcular el Error Cuadrático Medio (ECM)
    ecm = mean_squared_error(y_test_regresion, predicciones)

    # Calcular el Error Absoluto Medio (EAM)
    eam = mean_absolute_error(y_test_regresion, predicciones)

    # Características más importantes (importancia de las características)
    importancias_caracteristicas = modelo_regresion.feature_importances_

    # Coeficiente de determinación R2
    r2 = r2_score(y_test_regresion, predicciones)

    return puntuacion_regresion, predicciones, y_train_regresion, y_test_regresion, X_train_regresion, X_test_regresion, ecm, ear

# Aplicar modelo regresión
puntuacion_regresion, predicciones, y_train_regresion, y_test_regresion, X_train_regresion, X_test_regresion, ecm, eam, importan
```

Figura 36: Algoritmo de Regresión Múltiple con Random Forest a partir de los datos obtenidos con las distintas iteraciones del código anterior

En cuanto a los resultados obtenidos:

```
Parámetros de entrada:
                mf         M           t            p           LHV  \
33553  1414.001695  0.383604  280.707835  86951.332464  4.321065e+07
9427   1398.527311  0.398328  294.290529  92858.695586  4.479874e+07
199    1406.303814  0.407429  289.655398  93037.682268  4.418715e+07
12447  1391.583369  0.407243  293.965061  88029.575582  4.683256e+07
39489  1417.688897  0.380657  291.562878  93609.878981  4.516809e+07
...            ...       ...         ...           ...           ...
28567  1414.967096  0.389748  293.087783  91413.058937  4.406109e+07
25079  1408.359290  0.392048  275.982285  90755.106945  4.404879e+07
18707  1380.635855  0.391960  292.168284  87443.890839  4.049172e+07
15200  1423.690020  0.404516  282.784230  86815.866319  4.209907e+07
5857   1427.293881  0.406134  293.227800  93253.489637  4.605053e+07

            bpr    eta_fn    eta_c1    eta_c2    eta_t1    eta_t2     eta_n  \
33553  9.249570  0.927810  0.845763  0.928619  0.894917  0.904826  0.991352
9427   9.328672  0.923422  0.897460  0.851762  0.854439  0.883608  0.946371
199    9.327974  0.904805  0.856921  0.859659  0.912500  0.889077  0.990569
12447  9.391599  0.890220  0.900687  0.885229  0.917003  0.863506  0.928407
39489  9.910586  0.890167  0.860487  0.912581  0.904824  0.859870  0.926455
...         ...       ...       ...       ...       ...       ...       ...
28567  9.828796  0.961811  0.860656  0.915177  0.928644  0.897342  0.991998
25079  9.809950  0.887997  0.882583  0.910390  0.860845  0.875761  0.985797
18707  9.200019  0.946463  0.855667  0.906650  0.911382  0.847492  0.905400
15200  9.760081  0.963508  0.876207  0.848054  0.850104  0.930165  0.907582
5857   9.279352  0.879251  0.878362  0.921112  0.853493  0.853699  0.922828

         eta_cc      PI_i     PI_fn     PI_c1      PI_c2     PI_cc
33553  0.993503  0.982931  1.506864  1.519560   9.601037  1.014735
9427   0.942153  0.982591  1.479337  1.568122  10.035366  0.946533
199    0.987945  0.942848  1.469071  1.574265   9.523862  0.993623
12447  0.992468  1.025895  1.490787  1.583123  10.302864  1.027542
39489  0.991998  0.944793  1.616752  1.566297   9.756668  0.975476
...         ...       ...       ...       ...        ...       ...
28567  0.967391  0.935653  1.555429  1.583074  10.095759  0.937994
25079  0.994137  0.941742  1.552014  1.516373   9.632460  1.027653
18707  0.923141  1.025558  1.501905  1.477920   9.589392  1.022564
15200  0.919069  0.936123  1.514570  1.482772  10.255433  0.968266
5857   0.988854  1.018513  1.546088  1.594737   9.647571  0.975291

[10000 rows x 18 columns]
Predicciones del modelo de regresión múltiple (Random Forest):
 [7.2215767 6.1084901 7.4656609 ... 6.242034  5.1968387 6.5761914]
Error Cuadrático Medio (ECM): 0.49224449123256525
Error Absoluto Medio (EAM): 0.5134608174199998
Importancia de las características: [0.02839739 0.01692227 0.21587042 0.04420566 0.02003433 0.03383023
 0.03556584 0.09026163 0.0341673  0.06171396 0.10402292 0.01875796
 0.07165061 0.04365723 0.05165117 0.03178892 0.02780718 0.06969499]
Coeficiente de determinación (R^2) de las predicciones: 0.6662695553862241
```

Figura 37: Resultados y métricas correspondientes a la aplicación del modelo de Regresión Múltiple con Random Forest a los datos obtenidos a partir de las iteraciones del código anterior

Predicciones del modelo de regresión múltiple (Random Forest): Son las predicciones realizadas por el modelo de regresión múltiple utilizando el algoritmo Random Forest. Cada número en esta lista corresponde a la predicción para una observación en el conjunto de prueba.

Error Cuadrático Medio (ECM): Este valor, 0.4922 en este caso, representa la magnitud promedio de los errores al cuadrado entre las predicciones del modelo y los valores reales. Cuanto más bajo sea el ECM, mejor será el ajuste del modelo a los datos.

Error Absoluto Medio (EAM): Este valor, 0.5134 en este caso, representa la magnitud promedio de los errores absolutos entre las predicciones del modelo y los valores reales. Similar al ECM, cuanto más bajo sea el EAM, mejor será el ajuste del modelo a los datos.

Importancia de las características: Esta lista muestra la importancia relativa de cada característica en el modelo de Random Forest. Cuanto mayor sea el valor, más importante es la característica en la predicción del modelo. Estos valores pueden ayudar a comprender qué características tienen más peso en las predicciones del modelo.

Coeficiente de determinación (R^2) de las predicciones: Este valor, 0.666 en este caso, indica la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Cuanto más cercano sea R^2 a 1, mejor será el modelo en la explicación de la variabilidad de los datos. En este caso, un R^2 de 0.666 indica que aproximadamente el 66.6% de la variabilidad en la variable de respuesta puede ser explicada por el modelo.

Árbol de decisión

1. Cargar las bibliotecas necesarias: Se importan las bibliotecas necesarias para el análisis de datos y el modelado de machine learning, incluyendo pandas, numpy, sklearn.datasets, sklearn.model_selection, sklearn.tree, sklearn.metrics y datetime.
2. Definir el archivo de datos: Se especifica el nombre del archivo CSV que contiene los datos. En este caso, se usa 'datos_2024-03-28_ia_clasif_sp.csv'.
3. Cargar y preprocesar los datos: Se utiliza pd.read_csv() para cargar los datos desde el archivo CSV especificado. Se separan las características (X) de las etiquetas (y) utilizando la función iloc. Se divide el conjunto de datos en conjuntos de entrenamiento y prueba usando train_test_split() de sklearn.model_selection.
4. Inicializar y entrenar el modelo: Se inicializa un clasificador de árbol de decisiones utilizando DecisionTreeClassifier() de sklearn.tree. Se entrena el modelo utilizando los datos de entrenamiento con fit().
5. Realizar predicciones: Se utilizan los datos de prueba para hacer predicciones utilizando predict().
6. Evaluar el modelo: Se calculan varias métricas de evaluación del modelo, incluyendo precisión, recall y F1-score utilizando funciones de sklearn.metrics como accuracy_score, precision_score, recall_score y f1_score.
7. Mostrar resultados: Se imprimen los resultados obtenidos, incluyendo los parámetros de entrada (datos de prueba), las predicciones del modelo, así como las métricas de evaluación como precisión, recall y F1-score.

```python
# Tree decision classifier
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from datetime import datetime

fecha_actual = datetime.now().strftime('%Y-%m-%d')

archivo = 'datos_2024-03-28_ia_clasif_sp.csv'
# archivo = f'datos_{fecha_actual}_ia_clasif.csv'

def aplicar_arbol_clasificacion(csv_file):
    # Cargar datos desde el archivo CSV
    datos = pd.read_csv(csv_file)

    # Cargar el conjunto de datos
    X = datos.iloc[:, :-1]  # Todas las filas, todas las columnas excepto la última
    y = datos.iloc[:, -1]   # Todas las filas, sólo la última columna

    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Inicializar el clasificador de árbol de decisión
    clf = DecisionTreeClassifier()

    # Entrenar el modelo utilizando el conjunto de entrenamiento
    clf.fit(X_train, y_train)

    # Predecir las etiquetas para el conjunto de prueba
    y_pred = clf.predict(X_test)

    # Calcular la precisión del modelo
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    return accuracy, precision, recall, f1, y_pred, X_test, X_train, y_test, y_train

# Aplicar árbol de clasificacion
accuracy, precision, recall, f1, y_pred, X_test, X_train, y_test, y_train = aplicar_arbol_clasificacion(archivo)

# Mostrar las métricas
print("Parámetros de entrada:\n", X_test)
print("Predicciones del modelo:\n", y_pred)
print("Precisión del modelo:", accuracy)
print("Precisión:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Figura 38: Algoritmo de Clasificador de Árbol de Decisión a partir de los datos obtenidos con las distintas iteraciones del código anterior

En cuanto a los resultados obtenidos:

```
Parámetros de entrada:
              mf         M           t            p           LHV  \
33553  1495.473247  0.404898  269.515803  86895.113708  3.882611e+07
9427   1379.803811  0.409706  272.986233  89573.357044  5.142768e+07
199    1401.492289  0.407816  284.003243  90265.123747  4.858284e+07
12447  1492.235119  0.383037  285.446241  91770.776088  3.863053e+07
39489  1442.429107  0.407353  278.745767  92485.656002  4.422168e+07
...            ...       ...         ...           ...           ...
28567  1379.731864  0.387255  288.793804  89239.329170  3.837462e+07
25079  1387.076474  0.419306  275.892758  90312.115806  4.694525e+07
18707  1395.743562  0.382982  268.585185  91520.934533  5.081666e+07
15200  1424.176057  0.408424  289.656590  86794.378865  3.745536e+07
5857   1504.863354  0.404263  274.637920  87025.056179  3.844021e+07

             bpr    eta_fn    eta_c1    eta_c2    eta_t1    eta_t2     eta_n  \
33553  9.966221  0.950799  0.898759  0.930234  0.933021  0.857928  0.964555
9427   9.228469  0.952705  0.912302  0.898302  0.854147  0.847183  0.907844
199    9.247297  0.934588  0.856341  0.933120  0.871134  0.930625  0.949175
12447  9.377963  0.953667  0.849195  0.853023  0.888090  0.883974  0.964306
39489  9.125960  0.951495  0.845839  0.926442  0.882984  0.932336  0.910211
...         ...       ...       ...       ...       ...       ...       ...
28567  9.796300  0.916285  0.902474  0.877602  0.906861  0.930552  0.970723
25079  9.256505  0.888163  0.909749  0.854808  0.897843  0.882324  0.990465
18707  9.342850  0.928958  0.901354  0.910656  0.854738  0.882303  0.997241
15200  9.306013  0.893484  0.909267  0.894710  0.888985  0.921609  0.967706
5857   9.823796  0.896899  0.874589  0.926232  0.908150  0.868883  0.947230

          eta_cc      PI_i     PI_fn     PI_c1      PI_c2     PI_cc
33553  0.968779  0.973652  1.571152  1.561568   9.534060  0.959050
9427   0.937180  0.981210  1.526572  1.597379   9.893199  0.987509
199    0.938515  1.024685  1.509945  1.477542  10.235187  0.986812
12447  1.012579  0.988737  1.525718  1.577424   9.882347  0.997143
39489  0.931584  1.008090  1.506122  1.495132   9.906355  0.931541
...         ...       ...       ...       ...        ...       ...
28567  1.003024  1.021589  1.593988  1.596591   9.422199  0.944036
25079  0.960093  1.008087  1.480012  1.494370   9.829605  0.978510
18707  0.969341  0.960575  1.576471  1.591747   9.965142  0.942415
15200  0.967538  1.016035  1.522620  1.520649   9.649308  0.977921
5857   0.959654  0.956909  1.611343  1.599107  10.001296  0.960463

[10000 rows x 18 columns]
Predicciones del modelo:
 ['Bien' 'Bien' 'Bien' ... 'Bien' 'Notable' 'Bien']
Precisión del modelo: 0.6372
Precisión: 0.6369978216868831
Recall: 0.6372
F1 Score: 0.6370731362292882
```

Figura 39: Resultados y métricas correspondientes a la aplicación del modelo de Clasificador de Árbol de Decisión a los datos obtenidos a partir de las iteraciones del código anterior

Se observa que se obtiene una exactitud o coeficiente F1-score del modelo del 64%, que, si bien es relativamente buena, no consigue explicar la variabilidad del modelo, y por ello se opta por probar el modelo de validación cruzada.

Cross Validation con la variante k-fold

1. Primero, se importan las funciones y clases necesarias de scikit-learn. KFold se usa para realizar la validación cruzada en k-folds, cross_val_predict se utiliza para generar predicciones durante la validación cruzada, y cross_validate es una función que permite realizar validación cruzada con métricas múltiples. f1_score es una métrica comúnmente utilizada para evaluar el rendimiento de los modelos de clasificación.
2. La función calculate_metrics calcula las métricas de evaluación del modelo, en este caso el puntaje F1. Toma un estimador est (modelo), características X y etiquetas Y como entrada, genera predicciones del modelo, calcula el puntaje F1 y devuelve un diccionario que contiene el puntaje F1.
3. La función aplicar_modelo_clasificacion toma como argumento un nombre de archivo CSV y carga los datos desde ese archivo utilizando pandas.
4. Se separan las características (X) de las etiquetas (y). Las características son todas las columnas excepto la última, y la etiqueta es la última columna.
5. Se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split(). Aquí, se asigna el 80% de los datos para entrenamiento y el 20% para prueba, y se establece una semilla aleatoria para reproducibilidad.
6. Se crea una instancia del modelo de clasificación de bosque aleatorio y se realiza validación cruzada con 10 folds. El puntaje F1 ponderado es la métrica utilizada para evaluar el rendimiento del modelo durante la validación cruzada. Se devuelve el modelo entrenado.
7. Se realizan predicciones sobre el conjunto de prueba utilizando el modelo entrenado.
8. Se calculan diversas métricas de evaluación del modelo, como la exactitud, la precisión, el recall y el puntaje F1 ponderado, utilizando las funciones de scikit-learn correspondientes.
9. La función devuelve todas las métricas calculadas, así como también los conjuntos de entrenamiento y prueba y el modelo de clasificación.
10. Se aplica la función aplicar_modelo_clasificacion() al archivo especificado y se asignan los valores de retorno a variables.
11. Finalmente, se imprimen en la consola las métricas de evaluación del modelo.

```python
# Cross Validation
from sklearn.model_selection import KFold, cross_val_predict, cross_validate
from sklearn.metrics import f1_score

def calculate_metrics(est, X, Y):
    pred = est.predict(X)
    f1 = f1_score(Y, pred, average="macro")
    cov = np.cov(Y, pred, rowvar=False)
    dicti = {"f1_score": f1}
    return dicti

fecha_actual = datetime.now().strftime('%Y-%m-%d')

archivo = 'datos_2024-03-21_ia_clasif.csv'
# archivo = f'datos_{fecha_actual}_ia_clasif.csv''

def aplicar_modelo_clasificacion(csv_file):
    # Cargar datos desde el archivo CSV
    datos = pd.read_csv(csv_file)

    # Dividir los datos en caracteristicas (X) y etiquetas (y) para la clasificación
    X_clasificacion = datos.iloc[:, :-1]  # Selecciona todas las filas y todas las columnas excepto la última
    y_clasificacion = datos.iloc[:, -1]   # Selecciona todas las filas y la última columna

    # Dividir los datos en conjuntos de entrenamiento y prueba para clasificación
    X_train_clasificacion, X_test_clasificacion, y_train_clasificacion, y_test_clasificacion = train_test_split(X_clasificacion,

    # Inicializar y entrenar el modelo de clasificación de bosque aleatorio
    modelo_clasificacion = RandomForestClassifier()

    # Realizar predicciones y evaluar el rendimiento del modelo de clasificación
    CV = cross_validate(modelo_clasificacion, X_train_clasificacion, y_train_clasificacion, cv = KFold(n_splits=10, random_state
    modelo_clasificacion=CV['estimator'][0]

    predicciones_clasificacion = modelo_clasificacion.predict(X_test_clasificacion)

    #print(CV['covariance_matrix'][0])
    exactitud = accuracy_score(y_test_clasificacion, predicciones_clasificacion)
    matriz_confusion = confusion_matrix(y_test_clasificacion, predicciones_clasificacion)
    precision = precision_score(y_test_clasificacion, predicciones_clasificacion, average='weighted', zero_division=0)
    recall = recall_score(y_test_clasificacion, predicciones_clasificacion, average='weighted')
    f1 = f1_score(y_test_clasificacion, predicciones_clasificacion, average='weighted')

    # Convertir las etiquetas a formato binario
    y_test_bin = label_binarize(y_test_clasificacion, classes=np.unique(y_test_clasificacion))
    predicciones_bin = label_binarize(predicciones_clasificacion, classes=np.unique(y_test_clasificacion))

    # Calcular la curva precision-recall y AUC-PR para cada clase individualmente
    precision, recall, _ = precision_recall_curve(y_test_bin.ravel(), predicciones_bin.ravel())
    auc_pr = auc(recall, precision)

    # Calcular la curva ROC y el AUC-ROC para cada clase individualmente
    fpr = dict()
    tpr = dict()
    auc_roc = dict()
    n_classes = y_test_bin.shape[1]
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], predicciones_bin[:, i])
        auc_roc[i] = auc(fpr[i], tpr[i])

    return predicciones_clasificacion, exactitud, precision, recall, f1, matriz_confusion, auc_roc, fpr, tpr, X_train_clasificac

# Aplicar modelo clasificación
predicciones_clasificacion, exactitud, precision, recall, f1, matriz_confusion, auc_roc, fpr, tpr, X_train_clasificacion, X_test
```

Figura 40: Algoritmo de Cross Validation a partir de los datos obtenidos con las distintas iteraciones del código anterior

En cuanto a los resultados obtenidos:

```
Parmetros de entrada:
                mf         M           t            p           LHV  \
33553  1495.473247  0.404898  269.515803  86895.113708  3.882611e+07
9427   1379.803811  0.409706  272.986233  89573.357044  5.142768e+07
199    1401.492289  0.407816  284.003243  90265.123747  4.858284e+07
12447  1492.235119  0.383037  285.446241  91770.776088  3.863053e+07
39489  1442.429107  0.407353  278.745767  92485.656002  4.422168e+07
...            ...       ...         ...           ...           ...
28567  1379.731864  0.387255  288.793804  89239.329170  3.837462e+07
25079  1387.076474  0.419306  275.892758  90312.115806  4.694525e+07
18707  1395.743562  0.382982  268.585185  91520.934533  5.081666e+07
15200  1424.176057  0.408424  289.656590  86794.378865  3.745536e+07
5857   1504.863354  0.404263  274.637920  87025.056179  3.844021e+07

            bpr    eta_fn    eta_c1    eta_c2    eta_t1    eta_t2     eta_n  \
33553  9.966221  0.950799  0.898759  0.930234  0.933021  0.857928  0.964555
9427   9.228469  0.952705  0.912302  0.898302  0.854147  0.847183  0.907844
199    9.247297  0.934588  0.856341  0.933120  0.871134  0.930625  0.949175
12447  9.377963  0.953667  0.849195  0.853023  0.888090  0.883974  0.964306
39489  9.125960  0.951495  0.845839  0.926442  0.882984  0.932336  0.910211
...         ...       ...       ...       ...       ...       ...       ...
28567  9.796300  0.916285  0.902474  0.877602  0.906861  0.930552  0.970723
25079  9.256505  0.888163  0.909749  0.854808  0.897843  0.882324  0.990465
18707  9.342850  0.928958  0.901354  0.910656  0.854738  0.882303  0.997241
15200  9.306013  0.893484  0.909267  0.894710  0.888985  0.921609  0.967706
5857   9.823796  0.896899  0.874589  0.926232  0.908150  0.868883  0.947230

         eta_cc      PI_i     PI_fn     PI_c1      PI_c2     PI_cc
33553  0.968779  0.973652  1.571152  1.561568   9.534060  0.959050
9427   0.937180  0.981210  1.526572  1.597379   9.893199  0.987509
199    0.938515  1.024685  1.509945  1.477542  10.235187  0.986812
12447  1.012579  0.988737  1.525718  1.577424   9.882347  0.997143
39489  0.931584  1.008090  1.506122  1.495132   9.906355  0.931541
...         ...       ...       ...       ...        ...       ...
28567  1.003024  1.021589  1.593988  1.596591   9.422199  0.944036
25079  0.960093  1.008087  1.480012  1.494370   9.829605  0.978510
18707  0.969341  0.960575  1.576471  1.591747   9.965142  0.942415
15200  0.967538  1.016035  1.522620  1.520649   9.649308  0.977921
5857   0.959654  0.956909  1.611343  1.599107  10.001296  0.960463

[10000 rows x 18 columns]
Predicciones de clasificación:
 ['Bien' 'Bien' 'Notable' ... 'Bien' 'Notable' 'Bien']
Exactitud: 0.7721
Precisión: [0.2    0.7721 1.    ]
Recall: [1.     0.7721 0.    ]
F1-Score: 0.7458302767970496
Matriz de Confusión:
 [[4425    1    0  701    0]
 [ 554   26    0    3    0]
 [  44   14    0   15    0]
 [ 853    0    0 3270    0]
 [   0    0    0   94    0]]
AUC-ROC: {0: 0.7826573191960525, 1: 0.521502024276012, 2: 0.5, 3: 0.8273879630657932, 4: 0.5}
Curva ROC - FPR: {0: array([0.        , 0.29776318, 1.        ]), 1: array([0.        , 0.00159286, 1.        ]), 2: array([0.,
1.]), 3: array([0.        , 0.13833589, 1.        ]), 4: array([0., 1.])}
Curva ROC - TPR: {0: array([0.        , 0.86307782, 1.        ]), 1: array([0.        , 0.04459691, 1.        ]), 2: array([0.,
1.]), 3: array([0.        , 0.79311181, 1.        ]), 4: array([0., 1.])}
```

Figura 41: Resultados y métricas correspondientes a la aplicación del modelo de Cross Validation a los datos obtenidos a partir de las iteraciones del código anterior

Predicciones de clasificación: Estas son las etiquetas predichas por el modelo para cada instancia en el conjunto de prueba.

Exactitud (Accuracy): Es la proporción de predicciones correctas sobre el total de predicciones realizadas. En este caso, la exactitud es de 0.7721, lo que indica que el 77.21% de las predicciones realizadas por el modelo fueron correctas.

Precisión (Precision): Es la proporción de verdaderos positivos sobre el total de elementos identificados como positivos por el modelo. Los valores dados son para diferentes clases. Para la primera clase, la precisión es de 0.2, para la segunda clase es de 0.7721, y para la tercera clase es de 1.0.

Recall (Recuperación o Sensibilidad): Es la proporción de verdaderos positivos sobre el total de elementos que realmente son positivos en la muestra. Al igual que la precisión, los valores dados son para diferentes clases. Para la primera clase, el recall es de 1.0, para la segunda clase es de 0.7721, y para la tercera clase es de 0.0.

F1-Score: Es la media harmónica de precisión y recall. Proporciona una única medida que equilibra ambas métricas. El valor dado es de 0.7458.

Matriz de Confusión: Proporciona un desglose de las predicciones del modelo para cada clase en comparación con las clases reales. Cada fila representa las instancias en una clase real, mientras que cada columna representa las instancias en una clase predicha. Los números diagonales representan las predicciones correctas.

AUC-ROC (Area Under the ROC Curve): Es el área bajo la curva ROC. La curva ROC representa la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR) en varios umbrales de clasificación. Un valor de AUC-ROC más cercano a 1 indica un mejor rendimiento del modelo en la clasificación. En este caso, los valores dados son para diferentes clases.

Cross Validation con la variante k-fold y muestra balanceada

Como se puede observar, la muestra está notablemente desbalanceada debido a la generación aleatoria de las variaciones que posteriormente sentarán el criterio para la calificación de cada parámetro y por ende de la iteración en sí.

Por ello, se ha querido repetir el mismo ejercicio de 'cross validation' pero asegurando una muestra homogénea y balanceada de las etiquetas de cada iteración, de modo que se pueda eliminar el factor 'mayoría' en la predicción de resultados.

Los resultados que se obtienen son los siguientes:

```
Exactitud: 0.625
Precisión: [0.2   0.625 1.   ]
Recall: [1.    0.625 0.   ]
F1-Score: 0.6203216497154195
Matriz de Confusión:
 [[22 22  6 15  1]
 [13 50  0  3  0]
 [ 1 25 46  1  0]
 [20  0  0 33 17]
 [ 0  0  0  8 69]]
AUC-ROC: {0: 0.6072261072261073, 1: 0.7966200466200466, 2: 0.8043158049786419, 3: 0.6878419452887539, 4: 0.9153246753246753}
Curva ROC - FPR: {0: array([0.        , 0.11888112, 1.        ]), 1: array([0.        , 0.16433566, 1.        ]), 2: array
([0.        , 0.02150538, 1.        ]), 3: array([0.        , 0.09574468, 1.        ]), 4: array([0.        , 0.06545455, 1.
])}
Curva ROC - TPR: {0: array([0.        , 0.33333333, 1.        ]), 1: array([0.        , 0.75757576, 1.        ]), 2: array
([0.        , 0.63013699, 1.        ]), 3: array([0.        , 0.47142857, 1.        ]), 4: array([0.        , 0.8961039, 1.
])}
```

Figura 42: Resultados y métricas correspondientes a la aplicación del modelo de Cross Validation a los datos obtenidos a partir de las iteraciones del código anterior con la muestra balanceada

Exactitud o precisión (Accuracy): Para diferentes clases, los valores de precisión son 0.2, 0.625 y 1.0 respectivamente. Estos valores indican qué tan precisa fue la clasificación del modelo para cada clase. Por ejemplo, para la segunda clase, el 62.50% de las instancias clasificadas como pertenecientes a esa clase realmente lo son.

Recall (Recuperación o Sensibilidad): Los valores de recall son 1.0, 0.625 y 0.0 para las clases respectivas. Esto indica qué tan bien el modelo pudo identificar todas las instancias relevantes en cada clase. Por ejemplo, para la primera clase, todas las instancias relevantes fueron identificadas correctamente.

F1-Score: El F1-score es de 0.620, que es la media armónica de precisión y recall. Proporciona una única medida que equilibra ambas métricas.

Matriz de Confusión: La matriz de confusión proporciona un desglose de las predicciones del modelo versus las etiquetas reales para cada clase. Muestra que hay diferentes números de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos para cada clase.

AUC-ROC (Area Under the ROC Curve): Los valores de AUC-ROC indican la capacidad discriminativa del modelo para cada clase. Valores más cercanos a 1.0 indican un mejor rendimiento. Aquí, los valores varían entre 0.656 y 0.906. Estas curvas trazan la tasa de falsos positivos frente a la tasa de verdaderos positivos para diferentes valores de umbral. Proporcionan información sobre el equilibrio entre la tasa de verdaderos positivos y la tasa de falsos positivos para diferentes clases.

Como se puede observar, los resultados son mucho más homogéneos y equilibrados, si bien se pierde exactitud al ser una muestra más pequeña y por tanto que supone un peor y menor entrenamiento para el algoritmo de inteligencia artificial.

A continuación, se va a explicar en detalle cómo se construye la matriz de confusión para el caso de 'cross validation' a partir de la muestra balanceada vista en este análisis:

Predicción/Test	*Sobresaliente*	*Notable*	*Bien*	*Deficiente*	*Muy deficiente*
Sobresaliente	*69*	*17*	*1*	*0*	*0*
Notable	*8*	*33*	*15*	*3*	*1*
Bien	*0*	*20*	*22*	*13*	*1*
Deficiente	*0*	*0*	*22*	*50*	*25*
Muy deficiente	*0*	*0*	*6*	*0*	*46*
	77	*70*	*66*	*66*	*73*

Tabla 13: Matriz de confusión del algoritmo de cross validation con la muestra balanceada

Los valores señalados en amarillo y que conforman la diagonal principal de la matriz 5x5 son los que el algoritmo ha predicho de manera correcta, mientras que los valores sin resaltar y que se encuentran fuera de la diagonal principal de la matriz son los que el algoritmo no ha conseguido predecir de manera correcta.

De este modo, al conseguir 220 aciertos de un total de 352 casos de prueba o 'test' para el algoritmo en cuestión, se obtiene una exactitud del 63%.

En las columnas se encuentran los totales de las iteraciones que se corresponden con cada calificación dentro del conjunto de prueba para el algoritmo, mientras que en las filas se encuentran los totales de las iteraciones cuya predicción ha sido dicha calificación.

Por tanto:

Hay 69 Sobresaliente clasificados como Sobresaliente
Hay 8 Sobresaliente clasificados como Notable
Hay 0 Sobresaliente clasificados como Bien
Hay 0 Sobresaliente clasificados como Deficiente
Hay 0 Sobresaliente clasificados como Muy deficiente

Hay 17 Notable clasificados como Sobresaliente
Hay 33 Notable clasificados como Notable
Hay 20 Notable clasificados como Bien
Hay 0 Notable clasificados como Deficiente
Hay 0 Notable clasificados como Muy deficiente
Hay 1 Bien clasificados como Sobresaliente
Hay 15 Bien clasificados como Notable
Hay 22 Bien clasificados como Bien
Hay 22 Bien clasificados como Deficiente
Hay 6 Bien clasificados como Muy deficiente
Hay 0 Deficiente clasificados como Sobresaliente
Hay 3 Deficiente clasificados como Notable
Hay 13 Deficiente clasificados como Bien
Hay 50 Deficiente clasificados como Deficiente
Hay 0 Deficiente clasificados como Muy deficiente
Hay 0 Muy deficiente clasificados como Sobresaliente
Hay 1 Muy deficiente clasificados como Notable
Hay 1 Muy deficiente clasificados como Bien
Hay 25 Muy deficiente clasificados como Deficiente
Hay 46 Muy deficiente clasificados como Muy deficiente

Como remate del análisis y visualización de los resultados obtenidos a través de los diferentes algoritmos de aprendizaje automático vistos en este apartado, se muestran las gráficas típicas de este tipo de problemas para comparar la precisión del modelo, es decir, los valores predichos contra los valores reales, para el caso de 'cross-validation' con la variante k-fold:

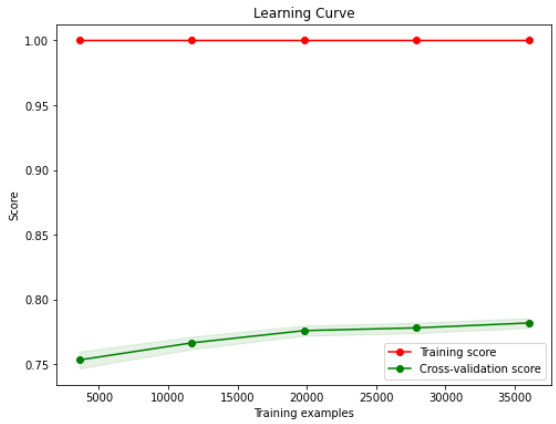

Figura 43: Learning curve del algoritmo de cross validation

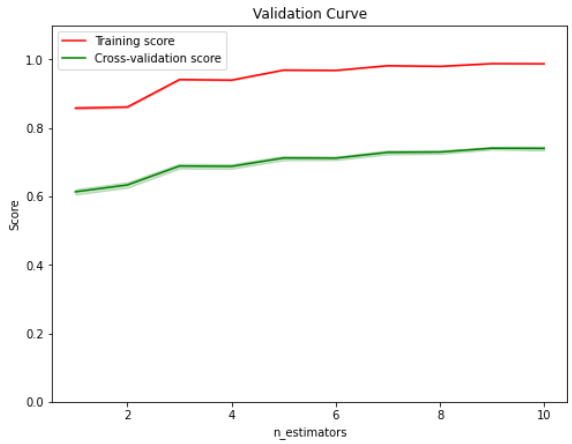

Figura 44: Validation curve del algoritmo de cross validation

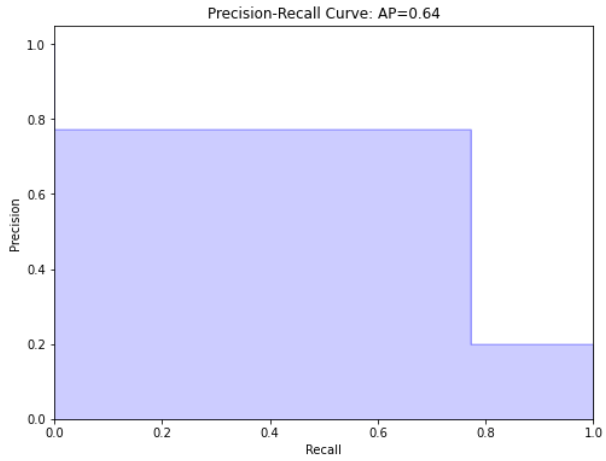

Figura 45: Curva de precisión-'recall' del algoritmo de cross validation

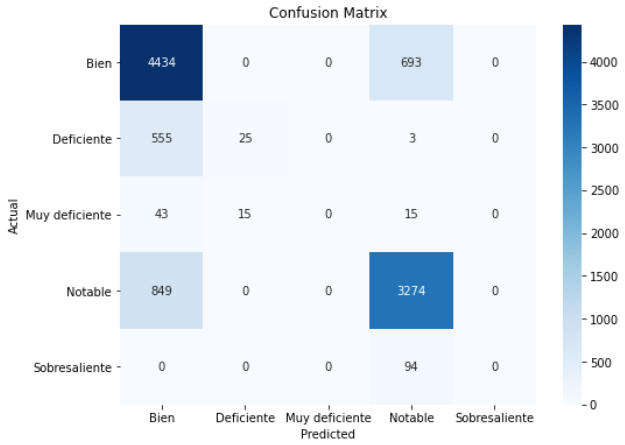

Figura 46: Matriz de confusión del algoritmo de cross validation

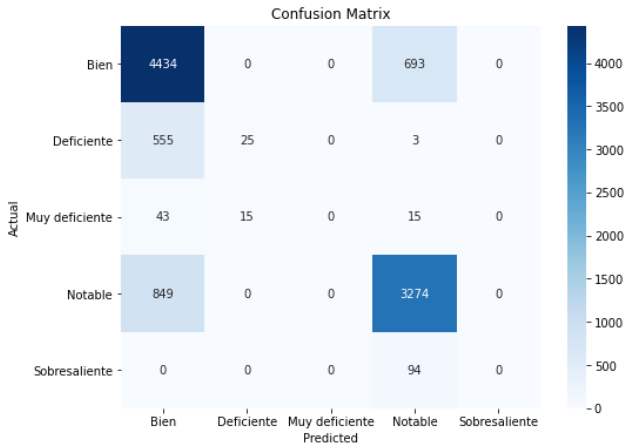

Figura 47: Matriz de confusión del algoritmo de cross validation con la muestra balanceada

Por último, en el apartado de conclusiones se expondrán las ventajas e inconvenientes en cuanto al análisis, con los distintos algoritmos de inteligencia artificial, así como una comparativa directa entre una muestra mayor pero desbalanceada o una menor pero balanceada, y se indicarán los siguientes pasos que sería conveniente dar para mejorar las métricas que definen la calidad de cualquier algoritmo de inteligencia artificial, en este caso un sistema de calificación.

10. Caso práctico

El objetivo de este apartado es utilizar los modelos construidos anteriormente, en este caso el de validación cruzada, para poder saber de manera rápida pero confiable la calificación de la iteración a partir de diferentes conjuntos de datos de entrada y todo el entrenamiento previo realizado sobre el modelo de inteligencia artificial, en este caso el de validación cruzada.

```
# Caso práctico con cross validation
data_input = [[1464.4861084193162,
               0.41183813194866475,
               275.17218144869094,
               88861.44839990883,
               49867345.747895196,
               9.18734373793585,
               0.8891906399527036,
               0.9125056627675807,
               0.8988425002073532,
               0.8485667841857824,
               0.8867645443776289,
               0.9527267002445362,
               0.9585781111815392,
               1.0196851933517495,
               1.5580635906267963,
               1.5443299302479616,
               10.012196902335651,
               0.9739539714459835]]
ans = modelo_clasificacion.predict(data_input)
print(ans)
```

```
['Notable']
```

Figura 48: Resultado predictivo de un caso práctico con un dataset cualquiera de entrada a través del modelo de Cross Validation

Como se puede observar, se alimenta al código que contiene el algoritmo de validación cruzada con 'dataset' o conjunto de datos de entrada ya definido, que en este caso son los que se definieron previamente como 'datos_in' y son los que se modifican aleatoriamente para buscar un número de casos de prueba suficiente.

A continuación, se aplica la lógica del algoritmo sobre ese conjunto de datos de entrada y se obtiene la calificación de la iteración o, lo que es lo mismo, la calificación de cada uno de esos parámetros de entrada y los correspondientes parámetros de salida que se generan a través del código que simula el software, a raíz de todo el entrenamiento previo que se ha realizado sobre el propio algoritmo.

En este caso, para el conjunto de datos de entrada indicado, se obtiene una calificación de 'Notable'. Al realizar el testeo con la validación cruzada, se comprueba que efectivamente esta debe ser la calificación correspondiente a este 'dataset'.

11. Conclusiones

Este artículo resume el estado de investigación y aplicación del DT en la industria de la aviación. En primer lugar, se resume el proceso de desarrollo del concepto de DT y se aclara los conceptos de DT e ingeniería de diseño, modelo de sistemas digitales, hilo digital, IoT y CPS. El artículo revisa la historia del DT en la industria de la aviación por parte de diversas organizaciones autorizadas, incluyendo la NASA, AFRL, el Departamento de Defensa, AFOR y NCR, y sobre esa base, resume las características del DT en la industria de la aviación. Se revisa el estado de aplicación del DT en la industria de la aviación a lo largo de todo el ciclo de vida, incluyendo el diseño de optimización, ensamblaje, fabricación, operación y mantenimiento, UAV y otros. Sobre esta base, se concluye que el DT en la aviación se utiliza principalmente en la fabricación y el mantenimiento. Se resumen las tecnologías clave del DT en la industria de la aviación. La fusión de datos es fundamental, mientras que la modelización de alta fidelidad multidimensional y 'multiscale' es esencial, la fusión de la Nueva Tecnología de la Información es el motor y la Interfaz Hombre-Máquina es necesaria. Se discuten las tendencias futuras de investigación y direcciones de desarrollo, incluyendo el control de riesgos de gemelos de proceso digitales, la investigación de DT interactivos y el aprendizaje de datos.

A pesar de que el DT se ha convertido en un tema de investigación candente, la industria de la aviación DT todavía tiene muchos problemas que deben resolverse urgentemente para mejorar la viabilidad de su aplicación en ingeniería.

En los últimos años, con el rápido desarrollo de las tecnologías de información y comunicación de nueva generación, como el Internet de las cosas (IoT), el big data, Internet móvil y la inteligencia artificial (IA), ha llegado la era del big data. Al mismo tiempo, como tecnología clave de los sistemas ciberfísicos (CPS), el gemelo digital ha recibido una amplia atención tanto de académicos como de empresas nacionales e internacionales. Por lo tanto, es importante aplicar las tecnologías de gemelo digital y 'big data' en el campo de la fabricación para promover la implementación de la fabricación inteligente en la planta de ensamblaje.

Así pues, en este proyecto se ha implementado satisfactoriamente una primera aproximación a un gemelo digital de operación. Con todo lo anterior, se ha creado finalmente un sistema global interconectado que permite probar el funcionamiento de un proceso, la detección de posibles errores y la implementación de distintas modificaciones, todo ello de manera virtual.

Este hecho abre la puerta a la implementación futura del proceso real con elevadas garantías de éxito. Además, se han implementado diferentes técnicas de machine learning que permiten, en base a las diferentes combinaciones de los datos de salida o 'outputs' del software, optimizar de manera eficiente la selección de los parámetros necesarios para obtener la simulación una vez se conoce el objetivo que se persigue y se ha entrenado suficientemente al algoritmo.

La optimización de diferentes simulaciones con el software Gasturb utilizando algoritmos de inteligencia artificial para evaluar y mejorar el rendimiento de un turbofán es un enfoque prometedor en la ingeniería aeroespacial.

La aplicación de algoritmos de clasificación o regresión a conjuntos de datos obtenidos de un software que simula el comportamiento de un motor turbofan de doble flujo presenta una amplia gama de posibles áreas de investigación.

Algunas de ellas se muestran a continuación:

- Mejora de la precisión del modelo: Investigar y desarrollar métodos para mejorar la precisión de los modelos de clasificación y regresión aplicados a los datos mediante la exploración de algoritmos de aprendizaje automático más avanzados, técnicas de preprocesamiento de datos más sofisticadas y la optimización de hiperparámetros.
- Modelado de fallas y diagnóstico de estado: Investigar cómo los algoritmos de clasificación pueden ser utilizados para identificar y predecir fallas en el motor turbofán. Esto podría implicar la construcción de modelos que clasifiquen el estado del motor como "normal" o "anormal" y la identificación de patrones de datos que indiquen posibles problemas.
- Predicción de rendimiento y eficiencia: Explorar cómo los algoritmos de regresión pueden utilizarse para predecir el rendimiento y la eficiencia del motor turbofán en diferentes condiciones de operación. Esto podría incluir la predicción de parámetros como el empuje, el consumo de combustible y la temperatura de los gases de escape.
- Optimización del diseño y control del motor: Utilizar algoritmos de regresión para optimizar el diseño y el control del motor turbofán. Esto podría implicar la construcción de modelos que relacionen los parámetros de diseño del motor con su rendimiento y eficiencia, lo que permitiría a los ingenieros explorar diferentes configuraciones de diseño y estrategias de control.
- Análisis de sensibilidad y robustez: Realizar análisis de sensibilidad y robustez utilizando modelos de regresión para comprender cómo diferentes factores y parámetros afectan el rendimiento y la confiabilidad del motor turbofán. Esto podría ayudar a identificar áreas críticas de mejora y a diseñar sistemas más robustos y resistentes a fallos.

Por otro lado, en cuanto a la ejecución de distintos algoritmos de inteligencia artificial sobre la muestra extraída en base a las distintas iteraciones que simulan el funcionamiento de un motor turbofán de doble flujo, se consigue primero acelerar la propia extracción de datos a través del software y segundo construir un criterio sólido y fiable para avanzar o predecir la calidad del 'dataset' particular con el que se quiera trabajar sin necesidad de iniciar el propio software y posteriormente evaluar su condición.

No obstante, como en cualquier proyecto, existen mejoras a realizar para garantizar por un lado la extracción homogénea, rápida y congruente de una muestra suficiente para posteriormente alimentar los algoritmos de inteligencia artificial y, por otro lado, para conseguir mejores métricas de evaluación de los distintos algoritmos, pues el entorno profesional es muy exigente en cuanto a la implantación de modelos que garanticen, y no aproximen, la veracidad de las predicciones para poder incorporarlas a la toma de decisiones.

Así, como conclusión para este proyecto, se proponen algunas mejoras que seguro terminarían de pulir los métodos empleados y por ende las conclusiones obtenidas de su posterior análisis:

- Conseguir una muestra homogénea basada en 'datasets' completamente fidedignos derivados de distintas pruebas bien en laboratorio o bien en algún software cuidadosamente ejecutado en cada caso de prueba en lugar de 'datasets' generados de manera aleatoria y en grandes cantidades.
- Seguir aumentado el 'F1-score' o la exactitud de este modelo a través de la simplificación de la lógica que lo respalda para conseguir que las aproximaciones se conviertan en predicciones completamente reales y fidedignas con aplicación directa en diferentes ámbitos de la toma de decisiones en el entorno profesional.

- Conseguir que el F1-Score aumente para el análisis con el algoritmo de 'cross validation' y la muestra balanceada a pesar de que se cuente con una muestra menor, pues la mayor variabilidad del dato debería ayudar en la correcta predicción de la clasificación de cada iteración.
- Discriminar o eliminar de la matriz de confusión aquellas predicciones erróneas que no se correspondan con el error más inmediato en cuanto a la asignación de la calificación definitiva pues es evidente que se debe a un claro error de interpretación del modelo.

En conclusión, si se incorporan las mejoras indicadas en el párrafo anterior, la optimización de simulaciones con software Gasturb utilizando algoritmos de inteligencia artificial ofrece un enfoque prometedor para mejorar las prestaciones de los motores de turbofán. Este enfoque tiene el potencial de aumentar la eficiencia, reducir los costos operativos y minimizar el impacto ambiental de la aviación, lo que beneficia tanto a la industria como al medio ambiente.

12. Futuras investigaciones

A pesar de que el DT ha logrado un excelente desempeño en la industria de la aviación, aún existe un gran potencial de mejora y optimización, especialmente en lo que respecta a su aplicación práctica.

En consecuencia, se presentan algunas tendencias de investigación y posibles direcciones:

- Control de riesgos en gemelos digitales de procesos: La investigación en la industria de la aviación sobre el DT abarca todo el ciclo de vida, lo que mejora la capacidad de las aeronaves para operar de manera eficiente y segura, pero también aumenta el riesgo y la dificultad de control en el gemelo digital de procesos. Una vez que se produce un riesgo, puede conllevar pérdidas materiales e incluso poner en peligro la seguridad de la vida. Por lo tanto, en el proceso de transformación digital de las aeronaves, es necesario prestar atención a la investigación de riesgos en el gemelo digital de procesos, reducir los factores inciertos en el proceso gemelo, mejorar la robustez del sistema y lograr una seguridad de la aviación inteligente.
- Investigación interactiva en DT: Las aeronaves DT no pueden existir sin la intervención de las personas, desde la investigación y desarrollo hasta su retiro, y el comportamiento humano es más subjetivo e impredecible. Por lo tanto, para lograr la interacción entre el mundo físico y virtual de las aeronaves DT, es necesario resolver el problema de la interacción entre humanos y máquinas en el marco de las aeronaves DT. Se debe considerar de manera integral la información, el cálculo y el control, y construir un modelo abstracto de las aeronaves DT con el ser humano en el ciclo, para lograr la coordinación óptima entre el ser humano y la aeronave y mejorar la capacidad de seguridad en vuelo.
- Aprendizaje de datos: El aprendizaje automático, el aprendizaje profundo y el aprendizaje por refuerzo tienen un buen desempeño en la clasificación, la agrupación y la generación de tareas, lo que puede mejorar la adaptabilidad del gemelo digital de la aeronave, de modo que el gemelo digital de la aeronave pueda evolucionar, percibir, recordar y resolver problemas. A la inversa, el modelo de DT puede acelerar la fase de entrenamiento del aprendizaje mediante la creación de conjuntos de datos de entrenamiento adecuados y la simulación de etiquetado automático de cadenas de herramientas, mejorando las capacidades de diagnóstico, fabricación y toma de decisiones de las aeronaves.
- Modelado de fenómenos complejos: Investigar y modelar fenómenos complejos dentro del motor turbofán que aún no se hayan abordado completamente. Esto podría incluir fenómenos de flujo multifásico, interacciones entre componentes del motor, y efectos dinámicos no lineales que podrían tener un impacto significativo en el rendimiento y la eficiencia del motor.
- Análisis de incertidumbre y variabilidad: Explorar cómo los algoritmos de clasificación y regresión pueden utilizarse para analizar la incertidumbre y la variabilidad en los datos de simulación del motor turbofán. Esto podría implicar la construcción de modelos que capturen la variabilidad inherente en los procesos de fabricación, las condiciones de operación y otros factores que afectan el comportamiento del motor.
- Integración de datos de diferentes fuentes: Investigar cómo integrar datos de diferentes fuentes, como mediciones experimentales, simulaciones numéricas y datos de campo, para mejorar la precisión y robustez de los modelos de clasificación y regresión aplicados al motor turbofán. Esto podría requerir el desarrollo de técnicas de fusión de datos y de modelado que aprovechen la información complementaria de cada fuente de datos.

- Aprendizaje automático interpretativo: Explorar enfoques de aprendizaje automático interpretativo que permitan comprender mejor los modelos de clasificación y regresión aplicados al motor turbofán. Esto podría incluir el desarrollo de métodos para visualizar y explicar las decisiones tomadas por el modelo, así como para identificar patrones y relaciones subyacentes en los datos que puedan no ser evidentes a simple vista.
- Optimización multiobjetivo: Investigar enfoques de optimización multiobjetivo que permitan encontrar soluciones óptimas para múltiples objetivos concurrentes en el diseño, operación y mantenimiento del motor turbofán. Esto podría implicar la formulación de problemas de optimización con múltiples objetivos y la aplicación de algoritmos de optimización avanzados para encontrar soluciones que equilibren estos objetivos de manera efectiva.

En resumen, hay muchas áreas de investigación aún por explorar en la aplicación de algoritmos de clasificación y regresión a conjuntos de datos de simulaciones de motores turbofán de doble flujo. Estas investigaciones podrían abrir nuevas oportunidades para mejorar la comprensión, el diseño y el rendimiento de estos sistemas críticos en la industria aeroespacial.

13. Referencias

[1] R. Schmidt, M. Möhring y R.-C. Härting, «Industry 4.0 -Potentials for Creating Smart Products: Empirical Research Results» de BIS 2015 18th International Conference on Business Information Systems, Poznan, Polonia, 2015.

[2] Deloitte, «www.deloitte.com,» 24 10 2014. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/ch/Documents/manufacturing/ch-en-manufacturing-industry4-0-24102014.pdf.

[3] M. N. Sishi y A. Telukdarie, «Implementation of Industry 4.0 technologies in the mining industry: A case study» de 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 2017.

[4] M. Russmann, P. Gerbert, M. Waldner, P. Engel, M. Harnisch y J. Justus, «www.bcg.com,» BCG, 09 04 2015. [En línea]. Available: https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.

[5] General Electric, «ge.com,» General Electric, [En línea]. Available: https://www.ge.com/digital/applications/digital-twin.

[6] IBM, «ibm.com,» IBM, [En línea]. Available: https://www.ibm.com/topics/what-is-a-digital-twin. [Último acceso: 10 07 2021].

[7] Siemens, «www.plm.automation.siemens.com,» Siemens, [En línea]. Available: https://www.plm.automation.siemens.com/global/es/our -story/glossary/digital-twin/24465.

[8] K. M. Alam, A. Sopena y A. El Saddik, «Design and Development of a Cloud Based Cyber-Physical Architecture for the Internet of Things» de 2015 IEEE International Symposium on Multimedia (ISM), Miami, USA, 2015.

[9] Chatalouf, G. (29 de Mayo de 2020). Dassault Systemes. Webinar Digital Twins.

[10] Kepware. (Junio de 2020). Plataforma KepserverEX. Obtenido de https://www.kepserverexopc.com/

[11] Lagándara, V. d. (2017). Diseño y programación de varias estaciones de trabajo mediante el uso de PLC y softwares específicos. Valladolid: Universidad de Valladolid.

[12] Mandado Pérez, E., Marcos Acevedo, J., Fernández Silva, C., & Armesto Quiroga, J. (2011). Autómatas programables y sistemas de automatización. Barcelona: Marcombo.

[13] NetToPlcSim. (Junio de 2020). Network extension for Plcsim. Obtenido de http://nettoplcsim.sourceforge.net/index.html

[14] Peón, C. d. (2019). Los gemelos digitales en la Industria 4.0. Valladolid: Universidad de Valladolid.

[15] Sánchez, J. M. (2018). Diseño de un sistema de control distribuido usando Factory IO y Codesys V3. Sevilla: Universidad de Sevilla.

[16] Serrano, A. S. (2017). Aplicacion del software Wonderware a simuladores industriales de procesos. Valladolid: Universidad de valladolid.

[17] Siemens. (Junio de 2020). SIMATIC PLC S7-1200. Obtenido de https://new.siemens.com/mx/es.html [10] Sirvent, S. J. (2018). Programación de un almacén automático de pallets. Valencia: Universidad de Valencia.

[18] Badea V, Zamfiroiu A, Boncea R (2018) Big Data in the Aerospace Industry. Informatica Economica 22(1):17–24

[19] Lee J, Bagheri B, Kao H (2015) A cyber-physical systems architecture for industry 4.0-based manufacturing systems.

[20] Lasi H, Fettke P, Kemper, HG et al (2014) Industry 4.0.

[21] Bus Inf Syst Eng 6:239–242. https:// doi. org/ 10. 1007/s12599- 014- 0334-4

[22] Ling L (2018) China's manufacturing locus in 2025: with acomparison of 'Made-in-China 2025' and 'Industry 4.0.' Technol Forecasting Soc Change 135:66–74

[23] Qi Q, Tuegel E (2018) Digital twin and big data towards Smart manufacturing and industry 4.0: 360 degree comparison". IEEE Access 6:3585–3593

[24] Mabkhot M, Al-Ahmari A, Salah B, Alkhalefah H (2018) Requirements of the smart factory system: a survey and perspective. Machines 6(23). https:// doi. org/ 10. 3390/ machi nes60 20023

[25] Zhou M, Yan J, Feng D (2019) Digital twin framework and itsapplication to power grid online analysis. CSEE J Power Energy 5(3):391–398

[26]. Tao F, Zhang H, Liu A, Nee A (2019) Digital twin in industry: state-of-the-Art. IEEE Trans Ind Informatics 15:2405–2415

[27] Kritzinger W, Karner M, Traar G, Henjes J, Sihn W (2018) Digital Twin in manufacturing: a categorical literature review and classification. IFAC- PapersOnLine 51(11):1016–1022

[28] Li L, Aslam S, Wileman A, Perinpanayagam S (2021) Digital twin in aerospace industry:a gentle introduction. IEEE Access 10:9543–9562

[29] Tuegel E, Ingraffea A, Eason T, Spottswood S (2011) Reengineering aircraft structural life prediction using a Digital Twin. Int J Aerosp Eng. https:// doi. org/ 10. 1155/ 2011/ 154798

[30] Melesse T, Pasquale RS (2020) Digital twin models in industrial operations: a systematic literature review. Procedia Manufacturing 42:267–272

[31] Grieves M, Vickers J (2017) Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. Transdisciplinary perspectives on complex systems, Springer, Cham, pp 85–113

[32] Boschert S, Rosen R (2016) Digital twin—the simulation aspect. Mechatronic Futures, Switzerland, Springer, Cham, pp. 59–74. https:// doi. org/ 10. 1007/ 978-3- 319- 32156-1_5

[33] Shafto M, Conroy M, Doyle R, Glaessgen E, Kemp C, LeMoigne J, Wang L (2010) NASA technology roadmap: modeling, simulation, information technology & processing roadmap technology area 11. 2019–04–13

[34] Tuegel E (2012) The airframe digital twin: some challenges to realization. In 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ ASME/AHS adaptive structures conference 14th.

14. Anexos

En este apartado, se pretende mostrar en detalle por un lado cómo se construye el código de Python que permite el modelado de los datos, así como su extracción y su procesamiento, y por otro cómo se construye el algoritmo de inteligencia artificial que permite el aprendizaje en base a los datos de salida del código y la estimación de nuevos resultados, así como el descubrimiento de las relaciones existentes entre los distintos parámetros y que definen las prestaciones de la operativa de un motor, en este caso un turbofán de doble flujo.

1. El primer paso es importar todos los módulos y bibliotecas a los que será necesario hacer referencia para el correcto funcionamiento del código de aquí en adelante.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import random
import datetime
import csv
from datetime import datetime
from huracan.engine import shaft
from huracan.thermo.fluids import gas, fuel
from huracan.components import inlet, fan, compressor, combustion_chamber, turbine, nozzle
```

2. El segundo paso es definir la función que va a calificar de manera automática y en función del output del código que simula el software Gasturb las distintas iteraciones.

```
#Calificación iteraciones
def calcular_puntuacion(parametro, valor_iteracion, valor_optimo, rango_min, rango_max):
#Calcula la puntuación de un parámetro en una iteración en función de su cercanía al valor óptimo.
    if valor_iteracion < rango_min or valor_iteracion > rango_max:
        return 0  # Valor fuera de rango, puntuación mínima (0)
    else:
        diferencia = abs(valor_optimo - valor_iteracion)
        rango_valores = rango_max - rango_min
        proporcion = diferencia / rango_valores
        puntuacion = 10 - (proporcion * 10)
        return max(0, round(puntuacion, 0))  # La puntuación mínima es 0
```

3. El tercer paso es definir las listas donde se van a ir generando y guardado tanto los datos de salida y entrada del código que simula el software Gasturb como las calificaciones de las distintas iteraciones y sus parámetros.

```
#Crear una lista para almacenar todas las puntuaciones de cada iteración y sus parámetros
puntuaciones_parametros = {}
puntuaciones_parametros_out = {}
puntuaciones_iteraciones = []
puntuaciones_iteraciones_out = []
puntuaciones_iteraciones_TOTAL = []
```

4. El cuarto paso es definir los valores iniciales de cada parámetro sobre los que se va a aplicar una variación aleatoria para conseguir distintos resultados en distintas iteraciones.

```
# Define los parámetros fijos
eta_gearbox=0.975

# Define los valores del caso de prueba
variables = {'mf': 1440,
             'M': 0.4,
             't': 281.65,
             'p': 89874,
             'bpr': 9.6,
             'PI_i': 0.98,
             'eta_fn': 0.92,
             'PI_fn': 1.54,
             'eta_c1': 0.89,
             'PI_c1': 1.54,
             'eta_c2': 0.89,
             'PI_c2': 9.86,
             'eta_cc': 0.965,
             'PI_cc': 0.98,
             'eta_t1': 0.89,
             'eta_t2': 0.89,
             'eta_n': 0.95,
             'LHV': 43e6}
```

5. El quinto paso es definir el número de iteraciones que se quiere aplicar y por tanto se considera suficiente para posteriormente entrenar con ciertas garantías al algoritmo de inteligencia artificial, así como la variación porcentual máxima que se quiere aplicar a los datos de los parámetros mostrados en el punto anterior. Esta variación porcentual será distinta para cada variable, para realizar mayores variaciones cuanto más grande sean los valores de dichas variables y obtener así resultados más diferentes.

```
# Número de iteraciones
num_iteraciones = 10000

# Rango de variación
variacion_base = 0.05 # Variación base para parámetros 1 < x < 100000 -> 5%
variacion_minima = 0.01  # Variación mínima para parámetros x < 1 -> 1%
variacion_maxima = 0.1  # Variación máxima para parámetros x > 100000 -> 10%
```

6. El sexto paso es definir los rangos de todos los datos, tanto los de entrada como los de salida, dentro de los cuales se acepta la variación de los parámetros, así como su valor óptimo, que servirá de referencia para calificar los parámetros en cada una de las iteraciones.

```
# Valores óptimos de los parámetros
valores_optimos = {'t': (300, 250, 350),
                   'p': (101325, 85000, 130000),
                   'bpr': (9, 5, 12),
                   'eta_fn': (0.9, 0.85, 1.0),
                   'eta_c1': (0.95, 0.80, 1.0),
                   'eta_c2': (0.95, 0.80, 1.0),
                   'eta_cc': (0.98, 0.90, 1.1),
                   'eta_t1': (0.96, 0.80, 1.0),
                   'eta_t2': (0.96, 0.80, 1.0),
                   'eta_n':(0.95, 0.9, 1.0),
                   'mf': (1500, 1000, 2000),
                   'PI_i':(1.25, 0.5, 2),
                   'PI_fn': (1.25, 0.5, 2),
                   'PI_c1': (5, 1, 10),
                   'PI_c2': (8.5, 5, 12),
                   'PI_cc': (1.03, 0.9, 1.1),
                   'M': (0.8, 0, 0.85),
                   'LHV': (44e6, 40e6, 46e6),
                   'eta_gearbox': (0.975, 0.95, 1)}

valores_optimos_out = {'i.t0': (300, 250, 350),
                       'i.p0': (95000, 80000, 110000),
                       'fn.t0': (300, 250, 375),
                       'fn.p0': (150000, 100000, 200000),
                       'c1.t0': (450, 300, 600),
                       'c1.p0': (300000, 200000, 400000),
                       'c2.t0': (850, 600, 900),
                       'c2.p0': (2250000, 1500000, 3000000),
                       'cc.t0': (1500, 1200, 1900),
                       'cc.p0': (2250000, 1500000, 3000000),
                       't1.t0': (1200, 1000, 1400),
                       't1.p0': (600000, 400000, 800000),
                       't2.t0': (850, 600, 1000),
                       't2.p0': (70000, 30000, 100000),
                       'n.t0': (450, 300, 600),
                       'n.p0': (150000, 100000, 200000)}
```

7. El séptimo paso es generar el código que permite obtener la variación porcentual aleatoria sobre los datos de entrada y su posterior calificación en función primero de si están dentro o no del rango permitido y segundo sobre su proximidad a los valores óptimos indicados.

```
# Iterar sobre cada variable de entrada y actualizar su valor
for iteracion in range(num_iteraciones):
    print(f'Iteración {iteracion+1}:')
    variables_random = dict()
    for var in variables:
        # Calcular la variación aleatoria dentro del rango especificado
        cambio_porcentual_aleatorio = random.uniform(-variacion, variacion)
        # Aplicar la variación aleatoria al valor actual
        variable_random = variables[var] * (1 + cambio_porcentual_aleatorio)
        # Mostrar las variables actualizadas
        print(f'{var}: {variable_random}')
        # Constuir un diccionario
        variables_random[var] = variable_random
        #Obtener los valores óptimos para el parámetro actual
        valor_optimo, rango_min, rango_max = valores_optimos[var]
        # Calcular la puntuación para el parámetro actual
        puntuacion_parametro = calcular_puntuacion(var, variable_random, valor_optimo, rango_min, rango_max)
        puntuaciones_parametros[var] = puntuacion_parametro
```

8. El octavo paso es general el código que permite generar los datos de salida en función de los datos de entrada indicados en el punto anterior a través de la simulación del software Gasturb, haciendo referencia a los distintos módulos y submódulos donde se define toda la física y termodinámica necesaria para entender el funcionamiento del motor en cuestión.

```
# Crear instancias con los valores de las variables actualizadas
f = fuel(LHV=round(variables_random['LHV'],0))
g = gas(mf=variables_random['mf'],
        cp=lambda T: 1150 if T > 600 else 1000,
        k=lambda T: 1.33 if T > 600 else 1.4,
        m=variables_random['M'],
        t_0=variables_random['t'],
        p_0=variables_random['p'])
i = inlet(PI=variables_random['PI_i'])
fn = fan(eta=variables_random['eta_fn'], PI=variables_random['PI_fn'])
c1 = compressor(eta=variables_random['eta_c1'], PI=variables_random['PI_c1'])
c2 = compressor(eta=variables_random['eta_c2'], PI=variables_random['PI_c2'])
cc = combustion_chamber(fuel=f, eta=variables_random['eta_cc'], PI=variables_random['PI_cc'])
t1 = turbine(eta=variables_random['eta_t1'])
t2 = turbine(eta=variables_random['eta_t2'])
n = nozzle(eta=variables_random['eta_n'])
shaft1 = shaft(fn, c1, t2, eta=variables_random['eta_c1'], eta_gearbox=eta_gearbox)
shaft2 = shaft(c2, t1, eta=variables_random['eta_c2'])
stream = g - i - fn
s1core, s1bypass = stream * (variables_random['bpr'] / (variables_random['bpr'] + 1))
s1core - c1 - c2 - cc - t1 - t2
s2 = s1core - s1bypass
s2 - n
stream.run()
```

9. El noveno paso es construir las listas donde se van a ir guardando tantos los datos de entrada aplicando una variación aleatoria, ‘datos_in’, como los datos de salida aplicando la lógica del software Gasturb y por tanto los que definen las prestaciones del motor, ‘datos_out’.

```
# Crear los DataFrames
datos_in = [variables_random['mf'],
            variables_random['M'],
            variables_random['t'],
            variables_random['p'],
            variables_random['LHV'],
            variables_random['bpr'],
            variables_random['eta_fn'],
            variables_random['eta_c1'],
            variables_random['eta_c2'],
            variables_random['eta_t1'],
            variables_random['eta_t2'],
            variables_random['eta_n'],
            variables_random['eta_cc'],
            variables_random['PI_i'],
            variables_random['PI_fn'],
            variables_random['PI_c1'],
            variables_random['PI_c2'],
            variables_random['PI_cc']]

datos_out = [i.t0, i.p0, fn.t0, fn.p0, c1.t0, c1.p0, c2.t0, c2.p0, cc.t0, cc.p0, t1.t0, t1.p0, t2.t0, t2.p0, n.t0, n.p0]
```

10. El décimo paso es, de igual modo que se hizo con los datos de entrada, generar el código que permite la calificación de los datos de salida en función de la calidad del dato según el rango aceptable y su proximidad al valor óptimo, aprovechando la función definida previamente.

```
# Obtener puntuaciones parámetros de salida
for var_out in valores_out.keys():
    #Obtener los valores óptimos para el parámetro OUT actual
    valor_optimo_out, rango_min_out, rango_max_out = valores_optimos_out[var_out]
    # Calcular la puntuación para el parámetro OUT actual
    puntuacion_parametro_out = calcular_puntuacion(var_out, valores_out[var_out], valor_optimo_out, rango_min_out, rango_max_
    puntuaciones_parametros_out[var_out] = puntuacion_parametro_out
```

11. El undécimo paso es generar los dataframes necesarios tanto con los datos de entrada como los de salida, así como las calificaciones de cada parámetro y el general de la iteración, para poder exportarlos a un ‘csv’ con identificadores.

```
# Crear un DataFrame con los datos de salida
nombres_columnas_out = ['T0_inlet [K]',
                        'p0_inlet [Pa]',
                        'T0_fan [K]',
                        'p0_fan [Pa]',
                        'T0_compressor1 [K]',
                        'p0_compressor1 [Pa]',
                        'T0_compressor2 [K]',
                        'p0_compressor2 [Pa]',
                        'T0_combustionchamber [K]',
                        'p0_combustionchamber [Pa]',
                        'T0_turbine1 [K]',
                        'p0_turbine1 [Pa]',
                        'T0_turbine2 [K]',
                        'p0_turbine2 [Pa]',
                        'T0_nozzle [K]',
                        'p0_nozzle [Pa]']

df1 = pd.DataFrame(datos_out, columns=nombres_columnas_out)

# Crear un DataFrame con los datos de entrada
nombres_columnas_in = ['mf',
                       'M',
                       't',
                       'p',
                       'LHV',
                       'bpr',
                       'eta_fn',
                       'eta_c1',
                       'eta_c2',
                       'eta_t1',
                       'eta_t2',
                       'eta_n',
                       'eta_cc',
                       'PI_i',
                       'PI_fn',
                       'PI_c1',
                       'PI_c2',
                       'PI_cc']

df2 = pd.DataFrame(datos_in, columns=nombres_columnas_in)
```

```
# Crear los DataFrames de puntuaciones de parámetros de salida
puntuaciones_in_df = pd.DataFrame([puntuaciones_parametros])
puntuaciones_out_df = pd.DataFrame([puntuaciones_parametros_out])
puntuacion_total_df = pd.DataFrame([puntuaciones_iteraciones_TOTAL])
```

12. El duodécimo paso es generar la lógica que califica cada parámetro, ya sea de entrada o de salida, y posteriormente la lógica que califica cada iteración.

```
# Calcular las puntuaciones de cada parámetro y la puntuación total promedio de la iteración
if 0 in puntuaciones_parametros.values():
    puntuaciones_iteraciones.append(0)
else:
    puntuacion_iteracion = round(sum(puntuaciones_parametros.values()) / len(puntuaciones_parametros),3)
    print('Puntuación iteración IN: ',puntuacion_iteracion)
    puntuaciones_iteraciones.append(puntuacion_iteracion)

if 0 in puntuaciones_parametros_out.values():
    puntuaciones_iteraciones_out.append(0)
else:
    puntuacion_iteracion_out = round(sum(puntuaciones_parametros_out.values()) / len(puntuaciones_parametros_out),3)
    print('Puntuación iteración OUT: ',puntuacion_iteracion_out)
    puntuaciones_iteraciones_out.append(puntuacion_iteracion_out)

if 0 in puntuaciones_parametros.values() or 0 in puntuaciones_parametros_out.values():
    puntuaciones_iteraciones_TOTAL.append(0)
else:
    suma_puntuaciones_parametros_in = sum(puntuaciones_parametros.values())
    suma_puntuaciones_parametros_out = sum(puntuaciones_parametros_out.values())
    num_parametros = len(puntuaciones_parametros) + len(puntuaciones_parametros_out)
    puntuacion_iteracion_TOTAL = round((suma_puntuaciones_parametros_in + suma_puntuaciones_parametros_out)/num_parametros,3)
    puntuaciones_iteraciones_TOTAL.append(puntuacion_iteracion_TOTAL)
```

13. El decimotercer paso es construir las listas sobre las que se van a generar y guardar de manera ordenada y coherente todos los conjuntos de datos necesarios para extraerlos a archivos ‘csv’ y posteriormente entrenar al algoritmo de inteligencia artificial.

```
# Crear los DataFrames de puntuaciones de parámetros de salida
puntuaciones_in_df = pd.DataFrame([puntuaciones_parametros])
puntuaciones_out_df = pd.DataFrame([puntuaciones_parametros_out])
puntuacion_total_df = pd.DataFrame([puntuaciones_iteraciones_TOTAL])

# Crear los DataFrames donde se van agrupando las iteraciones
df1_aux = pd.concat([df1_aux, df1])
df2_aux = pd.concat([df2_aux, df2])
puntuaciones_in_df_aux = pd.concat([puntuaciones_in_df_aux, puntuaciones_in_df])
puntuaciones_out_df_aux = pd.concat([puntuaciones_out_df_aux, puntuaciones_out_df])
```

14. El decimocuarto paso es generar la función que escale las puntuaciones de cada iteración para que dentro de la muestra exista toda la variedad de puntuaciones desde la peor con un 1 hasta la mejor con un 10 para poder entrenar de mejor manera al algoritmo de inteligencia artificial que se construiría posteriormente.

```
# Crear función para normalizar las puntuaciones del 1 al 10 de las iteraciones
def escalar_puntuaciones(puntuaciones):
    # Descartar el valor mínimo si es igual a cero
    puntuaciones_sin_cero = [valor for valor in puntuaciones if valor != 0]
    # Encontrar el valor mínimo y máximo de las puntuaciones
    min_puntuacion = min(puntuaciones_sin_cero)
    max_puntuacion = max(puntuaciones_sin_cero)
    # Calcular el rango de las puntuaciones
    rango_puntuaciones = max_puntuacion - min_puntuacion
    # Crear una lista para almacenar las nuevas puntuaciones escaladas
    puntuaciones_escaladas = []
    # Bucle de las iteraciones
    for puntuacion in puntuaciones:
        if puntuacion == 0:
            puntuaciones_escaladas.append(1)
        else:
            # Aplicar la fórmula de escalado min-max
            nueva_puntuacion = ((puntuacion - min_puntuacion) / rango_puntuaciones) * 9 + 1
            # Redondear la nueva puntuación al entero más cercano
            puntuacion_escalada = round(nueva_puntuacion, 5)
            puntuaciones_escaladas.append(puntuacion_escalada)
    return puntuaciones_escaladas
```

15. Se crea una función para calificar la puntuación normalizada de la iteración con etiquetas:

```
# Agregado para el algoritmo de ia
df_ia = pd.concat([df_ia, df2_aux, puntuacion_total_norm_df_transp], axis=1)
df_ia.columns = nombres_columnas_in + ['Puntuación iteración']

# Definir la función para clasificar según la puntuación
def clasificar_puntuacion(puntuacion):
    if puntuacion >= 9:
        return 'Sobresaliente'
    elif puntuacion >= 7 and puntuacion <9:
        return 'Notable'
    elif puntuacion >= 5 and puntuacion <7:
        return 'Bien'
    elif puntuacion >= 3 and puntuacion < 5:
        return 'Deficiente'
    else:
        return 'Muy deficiente'
```

16. Se construye el código que permite clasificar las puntuaciones normalizadas obtenidas tras todo el procesamiento anterior según el criterio indicado en la función del punto 16:

```
# Iterar sobre las filas de df_ia y clasificar la columna adicional de df_ia_clasif
df_ia_clasif = pd.DataFrame()
for index, row in df_ia.iterrows():
    puntuacion = row['Puntuación iteración']  # Obtener la puntuación de la fila actual
    etiqueta = clasificar_puntuacion(puntuacion)  # Clasificar la puntuación
    df_ia_clasif.at[index, 'Clasificación'] = etiqueta  # Asignar la etiqueta a la columna 20 de df_ia_clasif

# Constuir df_ia_clasif añadiendo la clasificación
df_ia_clasif = pd.concat([df_ia, df_ia_clasif], axis=1)
```

17. El decimoquinto y último paso es generar el código que permite la extracción a archivos ‘csv’ de todos los conjuntos de datos previamente construidos y ordenados.

```
# Escribir los DataFrame en archivos Excel
df1_aux.to_csv(ruta_completa_in, index=False)
df2_aux.to_csv(ruta_completa_in, index=False)
puntuaciones_in_df_aux.to_csv(ruta_completa_marks_in, index=False)
puntuaciones_out_df_aux.to_csv(ruta_completa_marks_out, index=False)
puntuacion_total_df.to_csv(ruta_completa_marks, index=False)
puntuacion_total_norm_df_transp.to_csv(ruta_completa_marks_norm, index=False)
df_ia.to_csv(ruta_completa_ia, index=False)
df_ia_clasif.to_csv(ruta_completa_ia_clasif, index=False)
```