Universidad Europea

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Diseño e implementación de solución para la detección de fallas en sistemas basados en sensores mediante machine learning, inteligencia artificial explicable y agentes de IA generativa.

Ingeniero Juan Bautista Acuña

Dirigido por

Dr. Álvaro Calle Cordón

CURSO 2024-25

TÍTULO: Diseño e implementación de solución para la detección de fallas en sistemas basados en sensores mediante machine learning, inteligencia artificial explicable y agentes de IA generativa.

AUTOR: Ingeniero Juan Bautista Acuña

TITULACIÓN: Máster Universitario en Análisis de Datos Masivos (Big Data)

DIRECTOR/ES DEL PROYECTO: Dr. Álvaro Calle Cordón

FECHA: Abril 2026

Resumen

El mantenimiento predictivo representa una de las aplicaciones más relevantes de la inteligencia artificial (IA) en la industria moderna, permitiendo anticipar fallos en equipos para reducir costos operativos. Sin embargo, los modelos de aprendizaje automático (ML) utilizados habitualmente en este contexto suelen funcionar generando predicciones que los usuarios no pueden validar ni cuestionar para entenderlo mejor o hasta encontrar fallas. Este trabajo propone el diseño y la implementación de una solución end-to-end para la detección temprana y diagnóstico de fallos en sistemas basados en sensores, integrando ML, IA Explicable (XAI) y modelos de lenguaje de gran escala (LLM).

El trabajo se aplica sobre el dataset "APS Failure at Scania Trucks", el cual cuenta con 60.000 registros, provenientes de 170 sensores simultáneos, con fuerte desbalance de clases de la variable objetivo (59:1) y un esquema de costos asimétrico. Se entrenan y comparan dos modelos: Random Forest (RF) y XGBoost, ambos con estrategia de class weighting para abordar el desbalance. El umbral de decisión se optimiza mediante la fórmula de Elkan, minimizando el costo operativo real. Los resultados muestran que RF, con umbral optimizado mediante el costo operativo real, logra el menor costo total (9.700 unidades frente a 13.440 de XGBoost), detectando el 98% de las fallas reales.

Posteriormente, para que el sistema pueda dar explicaciones de los sensores que influyen en la predicción, se aplica SHAP (SHapley Additive exPlanations). Para transformar estos resultados a lenguaje natural, se integran con Claude mediante la API (Interfaz de Programación de Aplicaciones) de Anthropic, generando diagnósticos adaptados a tres perfiles de usuario: operador, ingeniero y gerente; variando el vocabulario y la forma de dar la misma información. El despliegue se realiza mediante una API REST desarrollada en FastAPI y un dashboard con un chat en Streamlit, accesible para cualquier persona sin conocimientos técnicos en ML.

Los resultados demuestran que es posible construir sistemas de mantenimiento predictivo que no solo detecten fallos, sino que los expliquen y comuniquen de forma comprensible para todos los niveles de una organización industrial.

Abstract

Predictive maintenance represents one of the most relevant applications of artificial intelligence (AI) in modern industry, enabling the anticipation of equipment failures to reduce operational costs. However, the machine learning (ML) models commonly used in this context tend to generate predictions that users cannot validate or question in order to better understand them or identify potential flaws. This work proposes the design and implementation of an end-to-end solution for early fault detection and diagnosis in sensor-based systems, integrating machine learning (ML), explainable artificial intelligence (XAI), and large language models (LLM).

The work is applied to the "APS Failure at Scania Trucks" dataset, which contains 60,000 records from 170 simultaneous sensors, with a strong class imbalance in the target variable (59:1) and an asymmetric cost structure. Two models are trained and compared: Random Forest (RF) and XGBoost, both using a class weighting strategy to address the imbalance. The decision threshold is optimized using Elkan's formula, minimizing real operational cost. Results show that RF, with an optimized threshold, achieves the lowest total cost (9,700 units versus 13,440 for XGBoost), detecting 98% of actual failures.

Subsequently, to enable the system to explain which sensors influence each prediction, SHAP (SHapley Additive exPlanations) is applied. To transform these results into natural language, they are integrated with Claude via the Anthropic API (Application Programming Interface), generating diagnostics adapted to three user profiles: operator, engineer, and manager, varying the vocabulary and type of information provided. The system is deployed through a REST API built with FastAPI and an interactive dashboard with a chat module in Streamlit, accessible to any user without technical ML knowledge.

The results demonstrate that it is possible to build predictive maintenance systems that not only detect failures, but also explain and communicate them in a comprehensible way for all levels of an industrial organization.

Índice general

Índice de figuras

Índice de tablas

Glosario de acrónimos

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

APS: Air Pressure System (Sistema de Presión de Aire).

AUC-ROC: Area Under the Curve – Receiver Operating Characteristic (Área Bajo la Curva de la Característica Operativa del Receptor).

DL: Deep Learning (Aprendizaje Profundo).

FN: Falso Negativo: el modelo predice ausencia de falla cuando en realidad existe.

FP: Falso Positivo: el modelo predice falla cuando en realidad no existe.

GenAI: Generative Artificial Intelligence (Inteligencia Artificial Generativa).

HTTP: Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto).

IA: Inteligencia Artificial.

IIoT: Industrial Internet of Things (Internet Industrial de las Cosas).

IoT: Internet of Things (Internet de las Cosas).

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

LLM: Large Language Model (Modelo de Lenguaje de Gran Escala).

LSTM: Long Short-Term Memory (Red Neuronal de Memoria a Largo y Corto Plazo).

ML: Machine Learning (Aprendizaje Automático).

REST: Representational State Transfer (Transferencia de Estado Representacional).

RF: Random Forest (Bosque Aleatorio).

SHAP: SHapley Additive exPlanations (Método de explicabilidad basado en la teoría de juegos de Shapley).

TN: True Negative (Verdadero Negativo).

TP: True Positive (Verdadero Positivo).

UCI: University of California, Irvine.

XAI: Explainable Artificial Intelligence (Inteligencia Artificial Explicable).

XGBoost: Extreme Gradient Boosting.

Capítulo 1. Introducción

En las empresas manufactureras, las máquinas están destinadas a funcionar la mayor cantidad de tiempo posible para sacar el mayor provecho de sus inversiones iniciales. Esto a veces genera conflictos entre el área de producción y el área de mantenimiento, donde los primeros desean producir más mientras que los segundos prefieren detener la producción para la reparación lo antes posible y así evitar graves problemas y costos de reparación. Pero más allá de lo que desea el personal de una empresa, el cuidado del presupuesto o "budget" es el factor más limitante y determinante al momento de tomar decisiones. Es por ello que las empresas implementan sensores y sistemas que permitan recolectar datos de sus procesos en tiempo real para cuidar sus activos y reducir costos. Pero no basta solo con recolectar datos, dado que un dato por sí solo no sirve de nada si no se transforma en información. El segundo paso que las empresas comenzaron a implementar es el desarrollo de estrategias basadas en IA (Inteligencia Artificial), usando técnicas como ML (Machine Learning), DL (Deep Learning), etc. para tomar decisiones más rápidamente que un operador humano y así evitar fallos, aumentar el tiempo de operación de sus líneas y disminuir los costos asociados [1].

El equipo de mantenimiento tradicionalmente ha trabajado mediante dos tipos de mantenimiento. Un mantenimiento pensado en evitar una rotura y otro pensado en una reparación rápida para continuar con la operación sin importar los costos. El primero se llama *preventivo* y el segundo se llama *correctivo*. De ambos, el que siempre se desea evitar es el correctivo, ya que genera paradas no planificadas (no deseado en alta temporada) y costos de reparación muy elevados. Podría pensarse que la solución es el mantenimiento preventivo, porque previene la falla y por ende es menos costoso. En parte esto es cierto, es menos costoso, pero aún tiene gastos ocultos elevados ya que se intervienen equipos basados en el tiempo de uso, no en datos reales del sistema. Esto tiene involucrado paradas de producción innecesarias y cambios de repuestos que estén en buenas condiciones, sumado al costo de mano de obra propia o contratada. Pero, ¿qué sucedería si existiera una forma de saber el estado real de una pieza, basado en datos medidos en el lugar y en tiempo real? Podría decirse que sería la mejor opción. En las últimas décadas ha surgido el concepto de *mantenimiento predictivo*, cuyo objetivo es anticipar fallos mediante el análisis de datos actuales y mejor aún mediante modelos analíticos [2].

El mantenimiento predictivo se basa en datos históricos para detectar patrones de funcionamiento (train dataset). A partir de allí, se monitorean datos nuevos (test dataset), que pueden ser en tiempo real, para detectar algún comportamiento diferente al aprendido en la fase de entrenamiento y así catalogarlo como comportamiento anómalo: una señal de posible deterioro en algún componente mecánico, módulo o sensor. Gracias al avance del aprendizaje automático y del análisis de grandes volúmenes de datos, este enfoque se ha convertido en una de las aplicaciones más relevantes de la IA en la industria moderna. Modelos de ML permiten detectar relaciones entre muchas variables del sistema, lineales o no, difíciles de detectar inclusive por la persona más experimentada de la empresa.

En industrias modernas, llamadas 4.0, las máquinas ya cuentan con la suficiente cantidad de sensores y sistemas para registrar continuamente todas las variables que pueden afectar las variables objetivo del proceso. Pero más sensores instalados conlleva más datos y por lo tanto exige más capacidad y velocidad de procesamiento. Este tipo de información suele representarse como series temporales multivariantes, en las que múltiples variables evolucionan a lo largo del tiempo y presentan relaciones entre sí. Una vez almacenados los datos, analizarlos correctamente es el principal desafío de la predicción de fallos, ya que los problemas pueden manifestarse como cambios sutiles en la relación entre variables o solo en el comportamiento temporal de una de ellas. Realizar un etiquetado manual de cada registro como "sospechoso" trae aparejado un costo elevado en tiempo y recursos, sumado a que, en algunos casos, los propios cambios son imperceptibles y los registros de falla son escasos. Es posible entrenar modelos con datos de funcionamiento normales; cuando aparece una anomalía, el error de reconstrucción y predicción se dispara, delatando el problema [3].

Diversas publicaciones demuestran que las técnicas de aprendizaje automático eficaces para abordar este tipo de problemas son: modelos Random Forest, Gradient Boosting o Redes Neuronales. Estas técnicas han sido aplicadas con éxito en la detección de anomalías y en la predicción de fallos en sistemas industriales, lo que permite construir predictores capaces de identificar patrones asociados a fallos futuros a partir de datos históricos del sistema. Entonces se podría decir que el problema de detección de comportamientos anómalos estaría resuelto, pero se presenta un nuevo desafío: la interpretabilidad de los resultados de los modelos, especialmente en entornos donde las decisiones todavía están a cargo de operadores humanos.

El inconveniente de estas predicciones o procesos es que no son explicados, ya que muchos de ellos, basados en ML, funcionan como "cajas negras". Es por ello que para abordar este problema han surgido técnicas de XAI (IA Explicable) que permiten analizar la contribución de cada variable a la predicción del modelo. Entre estas técnicas se encuentran los métodos basados en valores de Shapley, como SHAP (SHapley Additive exPlanations), ampliamente utilizado para interpretar modelos complejos y entender el impacto de cada variable en la predicción final del modelo [4].

El uso de este tipo de técnicas para entender el modelo es especialmente importante en aplicaciones de mantenimiento predictivo. Si bien detectar un fallo con anticipación es fundamental, también lo es comprender qué parte del proceso (variable o sensor) está contribuyendo a esa predicción. Esta información permite a los ingenieros identificar las causas potenciales, aprender, priorizar acciones de mantenimiento y mejorar la toma de decisiones ante futuros fallos. Todo ello pone de manifiesto la necesidad de desarrollar soluciones que combinen predicción, explicabilidad y comunicación comprensible de los resultados a las personas [5].

En los últimos años, desde la expansión de los modelos de lenguaje a partir de 2022, se ha producido un rápido y variado avance en el desarrollo de modelos de GenAI (IA Generativa) y LLM (Modelo de Lenguaje de Gran Escala). Estos modelos demuestran una increíble capacidad para procesar información brindada por los modelos de ML, traducir sus resultados a lenguaje natural y asistir a las personas en tareas de análisis, diagnóstico y toma de decisiones. Diferentes

investigaciones han comenzado a explorar la integración de modelos de lenguaje con sistemas de monitoreo y mantenimiento, con el objetivo de transformar los resultados de modelos analíticos en explicaciones simples, comprensibles y adaptables para cada tipo de usuario. En este contexto surge la necesidad de desarrollar soluciones que integren estas tecnologías dentro de un mismo sistema analítico.

El presente trabajo propone el diseño e implementación de una solución end-to-end basada en IA para la detección de fallos en sistemas basados en sensores. El enfoque propuesto integra diferentes componentes dentro de un mismo pipeline analítico, incluyendo el procesamiento de datos, el entrenamiento de modelos de ML, la explicación de variables mediante técnicas de XAI y la generación de diagnósticos interpretables utilizando modelos de lenguaje. Con el objetivo de facilitar su uso en entornos reales, el sistema se despliega mediante una API (Application Programming Interface) REST (Representational State Transfer) y un dashboard interactivo, permitiendo que todos los trabajadores de una misma empresa puedan consultar y entender predicciones y diagnósticos de forma accesible y operativa.

Para validar el enfoque propuesto, el trabajo se desarrolla sobre un caso de uso concreto basado en “APS Failure at Scania Trucks” dataset, un conjunto de datos ampliamente utilizado en la investigación sobre mantenimiento predictivo, que contiene información proveniente de sensores de sistemas mecánicos cuyo objetivo es predecir si una falla en un camión está relacionada con el APS (Sistema de Presión de Aire) o con otro componente del vehículo. Este dataset permite abordar el problema de la detección de fallos a partir de múltiples variables del sistema y evaluar la capacidad de distintos modelos de ML para predecir anomalías o fallos futuros.

El trabajo se inicia con el preprocesamiento del dataset, sobre el cual se entrenan y comparan modelos de Random Forest y XGBoost. A continuación, se aplica SHAP para identificar qué variables del sistema influyen más en cada predicción, y esos resultados se trasladan a un modelo de lenguaje que genera un diagnóstico comprensible para el usuario. De esta forma, el sistema no solo identifica posibles fallos, sino que también proporciona una explicación comprensible de los factores que han llevado a dicha predicción. Además del análisis realizado sobre el dataset Scania, el enfoque propuesto busca ser lo suficientemente generalizable como para aplicarse a otros conjuntos de datos industriales. En definitiva, el objetivo no se limita a la detección de fallos, sino que comprende también la generación de diagnósticos comprensibles que orienten las decisiones de los equipos de mantenimiento.

Capítulo 2. Objetivos

2.1 Objetivos generales

Diseñar e implementar una solución end-to-end basada en IA para la detección, diagnóstico y entendimiento de fallos en sistemas basados en sensores, integrando modelos de ML, técnicas de IA explicable y modelos de lenguaje para la generación de explicaciones interpretables.

2.2 Objetivos específicos

1. Desarrollar un pipeline de procesamiento y modelado de datos utilizando el dataset "APS Failure at Scania Trucks" para la detección de fallos mediante modelos de ML: Random Forest como modelo base y comparando su rendimiento con modelos basados en gradient boosting como XGBoost, con una arquitectura suficientemente generalizable para su aplicación a otros conjuntos de datos industriales basados en sensores.

2. Analizar la interpretabilidad de los modelos mediante técnicas de XAI, aplicando métodos basados en SHAP para identificar las variables más relevantes en la predicción de fallos y comprender la influencia de cada sensor en el comportamiento del modelo.

3. Integrar modelos de lenguaje para generar diagnósticos interpretables a partir de los resultados del modelo, desarrollando un mecanismo basado en LLMs que permita generar explicaciones en lenguaje natural y facilitar la interpretación de los resultados por parte de los usuarios.

4. Implementar la solución desarrollada mediante una API REST y un dashboard interactivo, que permita a operadores e ingenieros de mantenimiento consultar predicciones, explicaciones y diagnósticos generados por el sistema de forma accesible y operativa.

2.3 Beneficios del proyecto

Uno de los principales beneficios del proyecto es que no se limita solo a predecir fallos, sino que se dedica a explicar por qué ocurren. Esto marca una diferencia concreta para los equipos de mantenimiento, que hasta ahora reciben alertas del sistema, pero deben interpretar ellos mismos qué sensor falló y qué acción tomar.

La aplicación de SHAP sobre los modelos entrenados permite identificar las variables del sistema que más influyen en cada predicción, convirtiendo una salida numérica en información accionable. A esto se suma la integración con un modelo de lenguaje, que traduce esos resultados en lenguaje natural comprensible y responde cualquier tipo de duda, adaptando las respuestas para cada persona de acuerdo a su formación.

Desde el punto de vista del despliegue, la implementación mediante API REST y un dashboard, permite que la solución pueda integrarse en ambientes de trabajo reales. Finalmente, al estar diseñado sobre un dataset específico pero con una arquitectura generalizable, el trabajo puede servir de base para implementaciones similares en otros sectores industriales basados en sensores.

Capítulo 3. Marco teórico

3.1 Industria 4.0 y el surgimiento de sistemas inteligentes

Las industrias 4.0 se caracterizan por automatizar y conectar máquinas y sensores con el mundo digital (una interfaz de software, por ejemplo) priorizando la eficiencia. Los sensores que transmiten datos en tiempo real se conectan a una red, que a su vez se comunica con otros dispositivos (Edge, Cloud, otros sensores), sus datos son procesados e interpretados por un sistema mayor para comprender mejor el comportamiento de los equipos y anticipar posibles problemas, lo que permite a los usuarios tomar decisiones. A todo este sistema interconectado se lo llama IoT (Internet de las Cosas).

El mundo se encuentra en constante evolución, y el mantenimiento predictivo no está ajeno a esto. Una de las evoluciones más significativas de la industria 4.0 hacia la 5.0 es la aplicación de IA al IoT (AIoT): Métodos de ML Clásico (Random Forest, SVM, KNN, Gradient Boosting y LightGBM), Aprendizaje por refuerzo (DQN, Multi-Agent RL), GenAI y también enfoques híbridos, como de combinación de modelos físicos con IA [6].

Un ejemplo representativo de esta transición es el trabajo presentado en [7], donde se implementó un sistema de mantenimiento predictivo en el Puerto de Limassol, Chipre. Anteriormente, el mantenimiento de los equipos de manejo de contenedores se realizaba con base en horas de operación, un enfoque propio del mantenimiento preventivo. Mediante la integración de sensores IoT, comunicación en tiempo real y modelos de aprendizaje automático, dentro del marco AIoT, el sistema adquirió la capacidad de anticipar fallas antes de que ocurran, reduciendo significativamente los tiempos de inactividad. Este caso ilustra cómo la Industria 4.0 dotó a las máquinas de inteligencia operativa, sentando las bases sobre las cuales la Industria 5.0 busca ahora incorporar valor agregado [7].

3.2 Mantenimiento predictivo basado en datos

Tradicionalmente, las estrategias de mantenimiento industrial se dividían en dos grandes categorías: mantenimiento correctivo y mantenimiento preventivo. El mantenimiento correctivo se realizaba después de que una máquina fallara, lo que generalmente implicaba altos costos debido a la reparación y a tiempos de inactividad inesperados. Por otro lado, el mantenimiento preventivo consiste en realizar revisiones periódicas de los equipos, independientemente de su estado real.

Si bien estas estrategias han sido utilizadas durante décadas, presentan importantes limitaciones. El mantenimiento correctivo puede generar interrupciones inesperadas en la producción, mientras que el mantenimiento preventivo puede provocar intervenciones innecesarias que incrementan los costos operativos.

El mantenimiento predictivo (Predictive Maintenance, PdM) surge como una alternativa más eficiente. En este enfoque, se instalan sensores en las máquinas para recopilar datos de funcionamiento, como temperatura, vibraciones, presión o consumo energético. Estos datos

son almacenados en una base de datos para luego ser analizados mediante técnicas de aprendizaje automático, con el objetivo de estudiar el comportamiento y detectar patrones que indiquen el deterioro de un componente o la proximidad de una falla. De esta manera, las organizaciones pueden intervenir de manera eficiente, optimizando el uso de recursos y reduciendo significativamente los tiempos de inactividad. [8]

3.3 Inteligencia artificial aplicada al mantenimiento

Dos publicaciones recientes presentan aplicaciones reales desde ángulos complementarios: una en un entorno portuario [7] y otra en maquinaria textil [8]. Estas demuestran el estado del arte de la aplicación de ML en industrias y explican la estrecha relación del avance del mantenimiento predictivo con las técnicas de ML y DL.

Ambas publicaciones mencionadas anteriormente siguen casi los mismos pasos: sensores, protocolo MQTT, base de datos (InfluxDB, MongoDB), modelo de ML (Random Forest, XGBoost y AdaBoost). Ambas basadas en datos de ciclo real, preprocesamiento con normalización z-score o equivalente, y validación con métricas estándar (accuracy, F1, precision, recall). Lo que diferencia a ambos estudios es el tipo de industria, lo que demuestra la versatilidad de los modelos.

Ambos estudios mencionan limitaciones que poseen sus modelos. El primero señala la escasez de registros de fallas como su principal restricción; mientras que el segundo reconoce que su modelo es de caja negra, limitación que se retoma en secciones posteriores al analizar la evolución hacia sistemas más adaptativos.

Adicionalmente a estos dos estudios, un estudio de revisión reciente [9] examina el rol de la GenAI en el mantenimiento predictivo, proponiendo: GANs y modelos de difusión para generar datos sintéticos de falla, y propone LLMs con grafos de conocimiento (knowledge graphs) para la interpretabilidad.

Estos modelos permiten analizar grandes cantidades de datos provenientes de sensores y detectar patrones complejos que serían difíciles de identificar mediante métodos tradicionales. A partir de estos datos, los algoritmos pueden aprender a distinguir entre comportamientos normales y señales que podrían indicar un problema futuro.

Sin embargo, a medida que estos modelos se vuelven más complejos, surge un nuevo desafío: comprender cómo toman sus decisiones.

3.4 El problema de la caja negra en la inteligencia artificial

Cuando se habla de caja negra en el ámbito industrial, se hace referencia a modelos de IA avanzados, como las redes neuronales profundas, que ofrecen una gran capacidad para detectar patrones en los datos, pero resultan difíciles de entender exactamente cómo llegan a una determinada predicción.

Para abordar este tema se analizan dos publicaciones recientes. En primer lugar, se toma la publicación de Cummins et al. [10] donde se analizan 102 artículos sobre mantenimiento

predictivo explicable (XPM) abarcando industrias como manufactura, energía eólica, transporte, y más, donde señalan que las explicaciones de los modelos rara vez se diseñan pensando en audiencias específicas y que falta evaluación humana de las mismas. Identifican el desafío general de la confianza del usuario como una debilidad del campo. En segundo lugar, se toma la publicación de Walker et al. [11], quienes se basan específicamente en la industria nuclear, y toman el problema de Cummins et al. y responden directamente con tres contribuciones prácticas: una interfaz de visualización centrada en el usuario desarrollando una visualización adaptada a diferentes niveles de usuario (científico de datos, operador, gerente), un marco de "confiar pero verificar" (*trust but verify*), y una evaluación empírica con personal real de la industria nuclear.

Este problema es particularmente relevante en aplicaciones industriales, donde las decisiones tomadas por los algoritmos pueden tener consecuencias importantes para la seguridad, la producción y los costos operativos.

En el contexto del mantenimiento predictivo, los operadores y técnicos necesitan comprender por qué un sistema está recomendando realizar una intervención en una máquina. Sin esta explicación, puede resultar difícil confiar plenamente en las recomendaciones generadas por los modelos de IA.

Además, investigaciones recientes han demostrado que existe un equilibrio entre el rendimiento de los modelos y su capacidad de explicación. En general, los modelos más complejos tienden a ser más precisos, pero también más difíciles de interpretar, mientras que los modelos más simples suelen ser más fáciles de explicar, aunque pueden ofrecer menor precisión [10], [11].

Es precisamente para abordar este desafío que surge el campo de la XAI aplicada al mantenimiento predictivo, el cual se analiza en la siguiente sección.

3.5 Explainable AI en mantenimiento predictivo

Los modelos de IA actuales, llamados cajas negras, dan una buena predicción, pero no explican nada. El paper de Pashami et al. [12] argumenta que esto hace que el sistema sea mucho menos útil en la práctica, porque las decisiones de mantenimiento son complejas e involucran a muchas personas con intereses y conocimientos diferentes. El paper divide el mantenimiento predictivo en dos grandes grupos:

- Diagnóstico: analiza lo que está pasando ahora mismo.
- Pronóstico: mira hacia el futuro.

El paper subraya que estas tareas no pueden tratarse de forma aislada. Si las tareas de diagnóstico no son comprensibles, las de pronóstico tampoco lo serán. Un mantenimiento es efectivo cuando:

- Se sabe qué componente falló y por qué.
- Se conoce la gravedad del problema y qué procesos se ven afectados.
- Se diseña un plan de acción concreto y eficiente.

- Se corrige la falla para que no vuelva a ocurrir.

Ninguna de estas cuatro necesidades puede resolverse solo con el modelo de IA, ni solo con el experto humano. Requieren colaboración entre ambos, y ahí es donde las explicaciones se vuelven indispensables.

Para obtener la información necesaria, la literatura propone principalmente tres momentos en los que se puede incorporar explicabilidad:

- Premoldeado: entender y explorar los datos antes de aplicar el modelo.
- Durante el modelado: usar modelos como árboles de decisión, regresión lineal, modelos causales o modelos híbridos.
- Posmodelado: agregar una capa de explicabilidad encima de modelos ya entrenados. Aquí entran técnicas como: SHAP (mide cuánto contribuye cada variable a una predicción), técnica extendida específicamente para series temporales en entornos IoT industriales por de la Peña et al. [4], LIME (crea un modelo simple local alrededor de una predicción para explicarla), explicaciones contrafactuales (responden "¿qué tendría que cambiar para que el resultado fuera diferente?") y explicaciones por ejemplos (muestran casos similares para ilustrar una predicción).

Los autores concluyen que, aunque XAI es muy importante para el mantenimiento predictivo, las técnicas actuales todavía están lejos de cubrir todas las necesidades reales. La mayoría de los trabajos existentes se limitan a calcular importancia de variables, lo cual rara vez es suficiente en la práctica.

3.6 Detección de anomalías en sistemas industriales

Una de las decisiones más difíciles de tomar dentro del mantenimiento predictivo es la definición de anomalía en el proceso. Una anomalía puede definirse como un comportamiento que se desvía del funcionamiento normal de un sistema. A continuación, se describen tres investigaciones basadas en diferentes formas de implementar una detección de anomalías.

La investigación de D. Meli [13] propone usar un *descubrimiento causal* para detectar anomalías. Su idea principal es aprender, mediante grafo causal del sistema, cómo se relacionan las variables en condiciones normales de funcionamiento, luego monitorear en tiempo real si ese patrón se rompe, y de hacerlo, activar una alarma. Su método se destaca por requerir menos tiempo de entrenamiento que el DL y por su capacidad de señalar exactamente qué conexión causal se rompió.

Por otro lado, investigaciones realizadas por E. Birihanu et al. [14] se basan en que, en condiciones normales, ciertos parámetros del sistema siempre se mueven de manera relacionada, es decir que existen *correlaciones estadísticas* entre pares de sensores. Por lo tanto, cuando ocurre un ataque o falla, estas correlaciones cambian. El modelo combina un autoencoder LSTM para determinar el tamaño óptimo de la ventana de tiempo, luego calcula la correlación de Pearson entre todas las variables de cada ventana, y con estos datos entrena un modelo de distribución gaussiana multivariada. Cuando los nuevos datos no encajan en esa

distribución, se detecta como anomalía. Finalmente, para hacer el método interpretable, aplican SHAP, que permite identificar cuáles variables fueron las más responsables de la anomalía detectada, actuando como análisis de causa raíz.

El tercer y último caso de investigación, por K. K. Kim et al. [15], propone capturar las *interdependencias secuenciales*, es decir, representar el comportamiento normal del sistema mediante secuencias visuales de series temporales, llamadas shapelets. Su razonamiento se basa en que los sistemas de control industrial operan en bucles cíclicos, por lo que el comportamiento normal tiene patrones de secuencia reconocibles que involucran múltiples variables simultáneamente. Usa un modelo Transformer con mecanismo de atención para identificar automáticamente qué segmentos de tiempo son más representativos del funcionamiento normal, extrayendo esos segmentos como shapelets. Luego entrena un clasificador de Random Forest que compara la distancia euclidiana entre los datos nuevos y los shapelets: si la distancia es grande, hay una anomalía. La explicabilidad viene de dos fuentes: visualmente se puede comparar el patrón anómalo con el shapelet normal, y los nodos del árbol de decisión del Random Forest revelan qué variables y a qué distancia marcaron la diferencia.

Los tres trabajos mencionados comparten un objetivo fundamental: detectar anomalías en sistemas industriales de manera que los operadores humanos puedan entender por qué se activó una alarma, no solo saber que algo falló. En conjunto, reflejan una tendencia clara en la investigación: alejarse de modelos más poderosos hacia métodos que sacrifican algo de rendimiento bruto a cambio de ser comprensibles, eficientes y confiables para quienes deben tomar decisiones en entornos industriales críticos.

3.7 ML, GenAI y sistemas multi-agente en el mantenimiento predictivo

A medida que los sistemas industriales se vuelven más complejos, aparece la necesidad de desarrollar arquitecturas capaces de analizar datos y tomar decisiones de manera autónoma. En esta sección se describen tres métodos: desde un mantenimiento predictivo funcional pero rígido, hacia uno adaptativo y explicable. Cada uno responde a las limitaciones del anterior, y juntos reflejan cómo el campo está evolucionando desde soluciones puntuales hacia sistemas que aprenden continuamente de la operación industrial.

El primer caso concreto y buen punto de partida para hablar del tema es el trabajo de Elkateb et al.[8] que implementa un sistema efectivo (92% de precisión), pero estático, ya que se entrena una vez y no se adapta a cuando las condiciones cambian. Es un sistema IoT con sensores físicos y un modelo AdaBoost para clasificar las paradas de máquinas.

Aumentando la complejidad del sistema, la literatura reciente de Li X et al. [9] diferencia a la IA tradicional, que aprende a predecir a partir de datos etiquetados, de la GenAI, que aprende la distribución subyacente de los datos y puede generar información nueva. Esto último es clave en el mantenimiento predictivo, ya que como se mencionó anteriormente, los datos etiquetados que reflejan una anomalía son escasos. El survey identifica beneficios adicionales en la aplicación de GenAI, relacionados con el uso de LLM, lo cual permite a los usuarios hacer consultas en lenguaje natural, crear guías paso a paso adaptadas para cada tipo de usuario combinada con

documentación técnica de la industria, entre otros. La publicación también menciona la importancia de limitar la GenAI para evitar que genere información incorrecta.

El tercer caso, publicado por Saleh et al. [16], introduce el concepto de *multi-agentes* al considerar que los modelos tradicionales son estáticos.

Distribuye agentes especializados en tres niveles computacionales: una capa Edge para la extracción de características de los sensores en tiempo real y baja latencia, una capa Fog para detección de anomalías mediante votación por consenso entre múltiples modelos, y una capa cloud para optimizar continuamente las políticas del sistema usando un algoritmo de aprendizaje por refuerzo, y genera explicaciones en lenguaje natural. Esta capacidad de razonamiento autónomo mediante LLMs es analizada en profundidad por Di Maggio [5], quien examina el estado del arte, los desafíos y las perspectivas futuras de los agentes basados en LLMs aplicados específicamente al mantenimiento predictivo.

3.8 Síntesis del marco teórico

Los papers [1], [2] y [10] son el punto de partida de la necesidad de pasar de un mantenimiento del tipo correctivo al predictivo. Presentan a la IA como una solución, pero plantean otro problema: ¿Es correcto confiar en una decisión que toma una máquina de forma autónoma? ¿Sin consultar a los especialistas de la compañía?

La solución tecnológica base aparece en los papers [6], [7] y [8] que muestran casos de éxito al combinar un sistema IoT con sensores y algoritmos de ML.

El detalle técnico de cómo un algoritmo detecta que algo está mal, es dado en los papers [3], [14] y [15], mostrando cada uno enfoques distintos (redes profundas, correlaciones, secuencias de operación), pero todos apuntando al mismo objetivo.

La explicabilidad (XAI) es dada por los papers [4], [11], [12] y [13], explicando que para que el personal que va a ejecutar la solución confíe en el resultado del algoritmo, necesita entender por qué el modelo dice lo que dice.

Por último, la visión de futuro, viene dada por los papers [5], [9] y [16]: agentes autónomos que piensan y se comunican en lenguaje natural, modelos generativos que crean datos sintéticos para compensar la escasez de datos de falla reales, y redes de muchos agentes que aprenden y se adaptan solos.

En conjunto, estos trabajos señalan que el campo está evolucionando de sistemas que detectan problemas hacia sistemas que razonan sobre ellos y actúan de forma independiente.

Capítulo 4. Metodología

4.1 Descripción del dataset

Este trabajo aplica sus modelos al dataset "APS Failure at Scania Trucks", publicado por la Universidad de California, Irvine (UCI) Machine Learning Repository [17]. El mismo cuenta con datos de sensores de camiones marca Scania y cuyo target tiene como objetivo predecir si una falla en el camión está relacionada con el APS o con otro componente del vehículo no relacionado con el APS. Por lo tanto, la variable objetivo es binaria: Positiva, si el APS falla o Negativa, si falla otro componente.

Se elige este dataset porque representa fielmente a un conjunto de datos de mantenimiento, al tener las siguientes características:

- Como se mencionó en el marco teórico, es normal que los datasets de mantenimiento tengan un marcado desbalance de clases. Este dataset posee aproximadamente 59.000 registros negativos (la falla proviene de otro componente) frente a 1.000 positivos (la falla proviene del APS). Esto da una proporción de 60:1 (muy desbalanceado).
- El dataset también define un esquema de costos asimétrico explícito: define al costo de un FN (Falso Negativo) en 500 unidades, mientras que al costo de un FP (Falso Positivo) en solo 10 unidades. Eso tiene sentido operativo ya que una rotura en ruta es mucho más cara que una revisión de mantenimiento en falso.
  Para entender mejor el concepto de costo, es necesario entender los conceptos de FN y FP. En clasificación binaria, cuando el modelo se equivoca puede hacerlo de dos maneras:
  FN: el camión tiene una falla del APS real, pero el modelo predice que no la tiene. Consecuencia: el camión sigue operando sin mantenimiento, la falla puede agravarse y generar una rotura costosa o un accidente. El dataset penaliza esto con 500 unidades.
  FP: el camión no tiene una falla del APS, pero el modelo predice que sí la tiene. Consecuencia: se envía el camión a revisión innecesariamente, perdiendo tiempo y generando un costo de mano de obra. El dataset penaliza esto con 10 unidades.
  La proporción 500:10 significa que equivocarse dejando pasar una falla real es 50 veces más costoso que generar una alarma innecesaria. Este esquema refleja la realidad operativa: no detectar una falla tiene consecuencias mucho más graves que generar una alerta incorrecta.

4.2 Preprocesamiento de datos

Después de realizar el análisis de valores faltantes del dataset, se observa que el mismo presenta una proporción elevada de valores faltantes en varias de sus columnas, en algunos casos superando el 50% de los registros. Normalmente, cuando la cantidad de valores faltantes es baja, es práctica común de los analistas de datos eliminarlos; pero hacerlo en este caso, reduciría significativamente el tamaño del conjunto de datos lo cual podría introducir sesgos en el análisis. Es por ello que se opta por una estrategia de imputación, basados en la literatura revisada sobre

manejo de valores faltantes en ML, que muestra que la imputación es preferible a la eliminación cuando la proporción de datos faltantes es alta [18].

Debido a la presencia de valores extremos, muy frecuente en datos de sensores industriales, se considera la elección más robusta reemplazar cada dato faltante por la mediana de los valores de esa misma variable.

Seguidamente se aplica sobre todas las variables continuas la normalización mediante StandardScaler. Esto se hace con el objetivo de asegurar que las diferencias de escala entre los diferentes sensores (columnas) no afecten el comportamiento de los modelos.

4.3 Modelado y manejo del desbalance

El desbalance de clases forma parte de los desafíos centrales de este trabajo, debido a que aplicar un modelo a datos desbalanceados daría un accuracy alto, ya que es fácil predecir algo que ocurre muchas veces. Esto pasa porque la métrica de accuracy trata todos los errores por igual: equivocarse en un caso de falla cuenta lo mismo que equivocarse en un caso normal. Pero en mantenimiento predictivo eso no es así: el costo de los dos tipos de error es radicalmente diferente, como quedó establecido en la sección anterior con la matriz de costos del Scania. Un buen modelo es el que logra detectar los casos que ocurren pocas veces [19]. En un contexto de mantenimiento predictivo esto es especialmente problemático, porque la clase minoritaria, la falla del APS, es precisamente la que más interesa detectar.

Para abordar este problema se adopta la estrategia de asignación de pesos por clase (class weighting), disponible nativamente en los algoritmos Random Forest y XGBoost. Lo que hace esta estrategia es asignarles un peso mayor a los errores cometidos sobre la clase minoritaria. En la práctica, con una proporción de 60:1, cada error en un caso de falla del APS vale aproximadamente 60 veces más que un error en un caso normal durante el entrenamiento. De esta forma el modelo “es forzado a entender” que equivocarse en una falla es mucho más costoso y presta más atención a esa clase, aunque tenga pocos ejemplos.

Otra estrategia que podría ser utilizada es SMOTE - Synthetic Minority Over-sampling Technique (Técnica de Sobremuestreo Sintético de la Clase Minoritaria) que, en lugar de penalizar, genera registros sintéticos nuevos de la clase minoritaria creando un dataset artificialmente balanceado con más registros de fallas. Son dos problemas los que se presentan al aplicar esta estrategia. El primero tiene que ver con que los casos escasos de falla en el dataset de Scania tienen características muy específicas. Generar datos sintéticos puede producir observaciones que no corresponden a ninguna falla real, introduciendo ruido al entrenamiento. El segundo problema es que si estos datos generados son pocos y similares entre sí, SMOTE tiende a generar variantes muy parecidas a los originales, aportando poca información nueva [20].

El class weighting evita todo esto porque no inventa datos: simplemente le dice al modelo que preste más atención a los ejemplos que ya tiene.

Una vez resuelto el desbalance de clases mediante class weighting, con los datos limpios, se entrenan y comparan dos modelos: Random Forest como modelo base y XGBoost como modelo

de comparación basado en gradient boosting. Ambos son particularmente robustos frente a datos que fueron imputados, porque al estar basados en árboles de decisión no asumen ninguna distribución estadística sobre los datos. Ambos se evalúan mediante validación cruzada estratificada, que preserva la proporción de clases en cada fold.

4.4 Ajuste del umbral de decisión

Cuando el clasificador recibe un registro nuevo de sensores, devuelve una probabilidad estimada de que ese registro corresponda o no a una falla del APS. Después, aplica un umbral para tomar la decisión final. Los clasificadores estándar toman decisiones usando un umbral de probabilidad de 0,5: si la probabilidad estimada de falla supera ese valor, el modelo predice falla del APS, de lo contrario, la falla corresponde a otro sistema.

Sin embargo, cuando los costos de error son asimétricos, como sucede en este dataset, un umbral de 0,5 deja de ser óptimo. Elkan [20] demuestra teóricamente que el umbral óptimo de decisión se puede calcular directamente a partir de la matriz de costos del problema. Aplicado al dataset Scania, quien define explícitamente en su documentación que el costo de un FN es 500 ($c_{FN} = 500$) y el costo de un FP es 10 ($c_{FP} = 10$), el umbral óptimo teórico resulta significativamente menor a 0,5, lo que implica que el modelo debe ser más sensible a la clase positiva de lo que lo sería con la configuración por defecto.

Según el paper de Elkan [20], la fórmula matemática para calcular el umbral óptimo a partir de una matriz de costos es:

$$p* = \frac{(c_{FP} - c_{TN})}{c_{FP} - c_{TN} + c_{FN} - c_{TP}} \qquad \textit{Ecuación 1}$$

Aplicando esa fórmula con los valores del Scania y asumiendo que los aciertos no tienen costo ($c_{TN} = 0, c_{TP} = 0$):

$$p* = \frac{10}{10 + 500} = \frac{10}{510} = 0{,}0196 \approx 0{,}02 \qquad \textit{Ecuación 2}$$

Eso significa que, si el modelo estima una probabilidad de falla mayor al 2%, ya debería clasificar el registro como falla del APS.

Este cálculo indica un punto de partida matemático, pero en la práctica, este valor puede no ser óptimo porque Elkan asume que las probabilidades que devuelve el modelo están perfectamente calibradas, es decir, que cuando Random Forest dice 0,15 de probabilidad de falla, esa probabilidad refleja con exactitud la realidad. Pero en la práctica, los modelos basados en árboles tienden a dar probabilidades mal calibradas. También, el umbral óptimo puede variar dependiendo de cómo el modelo específico aprendió de los datos.

Es por ello que se realiza una validación empírica mediante la prueba de distintos valores de umbral sobre el conjunto de validación, mediante el costo total:

$$costo\ total = 500 * FN + 10 * FP \qquad \textit{Ecuación 3}$$

Se elige el que dé menor costo real. Finalmente, se visualiza la curva de costo total en función del umbral para justificar la elección de forma transparente.

4.5 Métricas de evaluación

Debido al fuerte desbalance que posee el dataset de Scania, la métrica *accuracy* no es una métrica informativa debido a que un modelo que clasifique todos los registros como negativo (no resulta en una falla del APS) obtendría una accuracy superior al 98% sin detectar ninguna falla real.

En su lugar, se tienen en cuenta las siguientes métricas:

- Costo total: es la métrica principal definida por el propio dataset (Ecuación 3)
- *Precision*: mide cuántas de las alarmas que genera el modelo son fallas reales, es decir, cuántas alertas valen la pena.

$$Precision = \frac{TP}{TP + FP} \qquad \textit{Ecuación 4}$$

- *Recall*: mide cuántas de las fallas reales el modelo logra detectar.

$$Recall = \frac{TP}{TP + FN} \qquad \text{Ecuación 5}$$

- F1-score: balancea *precision* y *recall.*

$$F1 = 2 * \frac{precision * recall}{precision + recall} \qquad \textit{Ecuación 6}$$

- AUC-ROC: se usa como referencia general del poder discriminativo del modelo.

Saito y Rehmsmeier [21] demuestran que en datasets desbalanceados el AUC-ROC puede ser engañoso ya que mide qué tan bien el modelo separa las dos clases evaluando la tasa de verdaderos positivos (recall) contra la tasa de FP a distintos umbrales. El problema es que la tasa de FP se calcula dividiendo por el total de negativos, el cual es muy elevado (59000 registros). Es decir que el AUC-ROC es insensible a los errores sobre la clase mayoritaria cuando ésta es muy grande, y puede mostrar un valor alto, aunque el modelo esté detectando muy pocas fallas reales.

Para compensar, se utiliza el análisis de la curva *precision-recall*. La precision y el recall se calculan solo sobre los positivos, sin que el gran volumen de negativos pueda distorsionar el

resultado. Cuando un modelo funciona mal sobre la clase minoritaria, la curva Precision-Recall lo muestra inmediatamente con valores bajos, algo que el AUC-ROC podría ocultar.

4.6 Explicabilidad con SHAP

Una vez seleccionado el mejor modelo, se aplica SHAP para analizar la contribución de cada variable a cada predicción individual. Existen otras técnicas de XAI como LIME. Pero lo que diferencia a SHAP sobre LIME es que SHAP está fundamentada en teoría de juegos y garantiza que la suma de las contribuciones individuales de todos los sensores es exactamente igual a la diferencia entre la predicción del modelo y su valor base. El valor base es una predicción promedio que el modelo tiene al inicio (antes de analizar ningún sensor en particular) sobre todo el dataset de entrenamiento. Es el punto de partida. Cuando el modelo analiza un registro en concreto, predice una probabilidad de falla. SHAP descompone la diferencia entre el valor base y la probabilidad calculada para ese registro y le asigna a cada sensor una fracción de responsabilidad, indicando la magnitud de influencia de cada sensor para esa predicción específica.

Lo que SHAP garantiza es que si se suman las contribuciones de todos los sensores, el resultado es exactamente igual a esa diferencia de 0,433. No es una aproximación ni una estimación, es una igualdad exacta. Eso significa que el 100% de la predicción queda explicado, sin que ninguna parte quede sin atribuir a alguna variable [4].

A diferencia de SHAP, LIME construye un modelo simple alrededor de una predicción para aproximar el comportamiento local del modelo, pero esa aproximación puede ser inexacta. Si se suman las contribuciones que calcula LIME no necesariamente se obtiene la predicción original, lo cual hace que se pierda confianza y trazabilidad. Esto permite comunicar al usuario, con certeza, qué sensores y en qué proporción influyeron en la probabilidad.

4.7 Integración con Claude y generación de diagnósticos

Los resultados del modelo y los valores SHAP de cada predicción se envían a Claude mediante la API de Anthropic.

El usuario puede interactuar con el sistema en dos modos complementarios:

- Diagnóstico automático: por cada predicción se construye un prompt estructurado que incluye el resultado del modelo, la probabilidad estimada, y los principales sensores SHAP con su dirección de influencia. A partir de esta información, Claude genera un párrafo de diagnóstico en lenguaje natural, adaptado al nivel de conocimientos del usuario (operador, ingeniero o gerente).
- Chat: Posteriormente al diagnóstico automático, si el usuario decide profundizar sobre algo en específico, puede hacer preguntas sobre la predicción activa y Claude responde manteniendo el contexto del caso cargado en la sesión.

En ambos modos, el prompt incluye una instrucción explícita para que el modelo no genere información que no esté respaldada por los datos de la predicción, siguiendo la recomendación

de Li et al. sobre la importancia de limitar las alucinaciones en GenAI aplicada a mantenimiento industrial [9].

4.8 Despliegue

La solución se despliega mediante una API REST desarrollada en FastAPI, que expone endpoints para predicción individual, predicción en *batch*, generación de diagnóstico y chat. El dashboard interactivo desarrollado en Streamlit integra las tres funcionalidades en una interfaz única, permitiendo que operadores e ingenieros de mantenimiento consulten predicciones, explicaciones y diagnósticos de forma accesible sin necesidad de conocimientos técnicos en ML.

El código fuente completo de la solución se encuentra disponible en el siguiente repositorio: https://github.com/bautistaacuna/tfm-mantenimiento-predictivo-acuna.git

Capítulo 5. Resultados

5.1 Análisis exploratorio y preprocesamiento

El dataset utilizado en este trabajo de fin de master fue el "APS Failure at Scania Trucks". El mismo cuenta con 60.000 registros de entrenamiento y 16.000 registros de test (26.7%), distribuidos en 171 columnas o features, de las cuales 170 corresponden a variables de sensores y una a la variable objetivo que indica si la falla es atribuible al APS o no. Una vez finalizado el análisis de distribución de las clases, se confirma que efectivamente existe el desbalance predicho en la sección metodología: 59.000 (98,33%) registros pertenecen a la clase negativa (no es una falla del APS) y 1.000 (1,67%) a la clase positiva (es una falla del APS), resultando en una relación de desbalance de 59:1 (Figura 1).

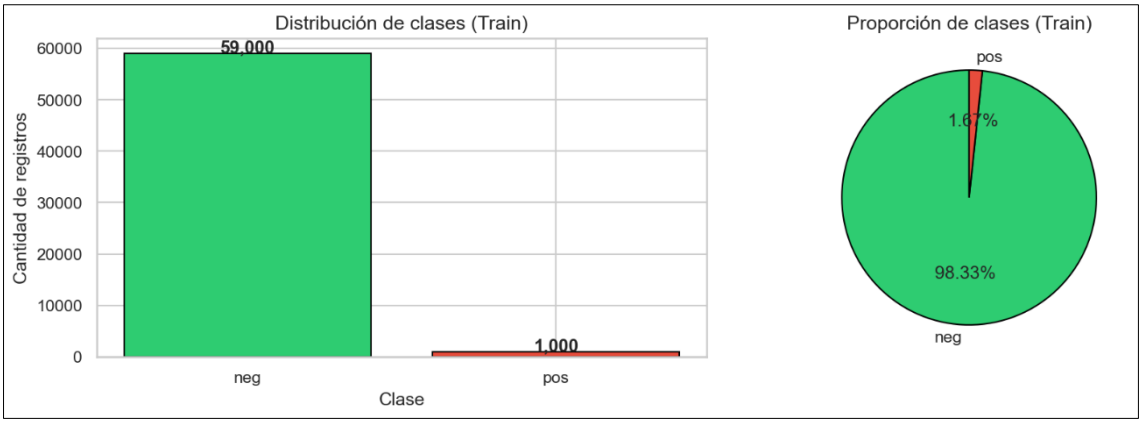

Figura 1. Distribución de clases en el conjunto de entrenamiento (relación 59:1).

Adicionalmente, el análisis de valores faltantes (Figura 2) revela que 169 de los 170 features presentan al menos un valor faltante, de los cuales:

- 8 superan el 50% de registros ausentes
- 16 se encuentran en el rango del 20% al 50%
- 145 restantes presentan menos del 20% de valores faltantes

Las variables con mayor proporción de valores faltantes fueron br_000 (82,11%), bq_000 (81,20%) y bp_000 (79,57%). Con estos datos, una eliminación de estas variables no sería la mejor opción ya que reduciría significativamente el conjunto de datos e introduciría sesgos en el análisis. Por lo tanto, se opta por una estrategia de *imputación por mediana*, calculada exclusivamente sobre el conjunto de entrenamiento para evitar *data leakage* o fuga de datos. Este error se refiere a que si se calculara la mediana usando todos los datos (train + test) y luego se usara esa mediana para imputar en todo el dataset, el modelo habría "visto" información del test durante el entrenamiento. Esta acción invalida la evaluación, porque los datos nuevos deberían ser completamente desconocidos. Es por ello que se calcula la mediana solo sobre el conjunto train y se aplica tanto a todo el dataset, simulando de esta forma lo que ocurre en un

entorno real: el modelo aprende únicamente de los datos históricos y posteriormente se evalúa sobre datos que no vio.

Explicado esto, después de aplicar la imputación, el conjunto de datos quedó completamente libre de valores faltantes. Posteriormente, es necesario realizar una normalización debido a que los sensores tienen diferentes rangos de valores y unidades de medida, por lo que una variable con valores numéricamente más elevados podría influir desproporcionadamente en el modelo respecto a otra de menor escala. Se elige para normalizar StandardScaler frente a otras alternativas debido a que se precisan métodos que no sean sensibles a valores extremos, frecuentes en datos de sensores industriales, lo que podría distorsionar la escala del resto de las variables. Se aplica *StandardScaler* sobre todas las variables continuas, obteniendo una media de 0 y una desviación estándar de 1 en todas las columnas, lo que garantiza que las diferencias de escala entre sensores no afecten el comportamiento de los modelos.

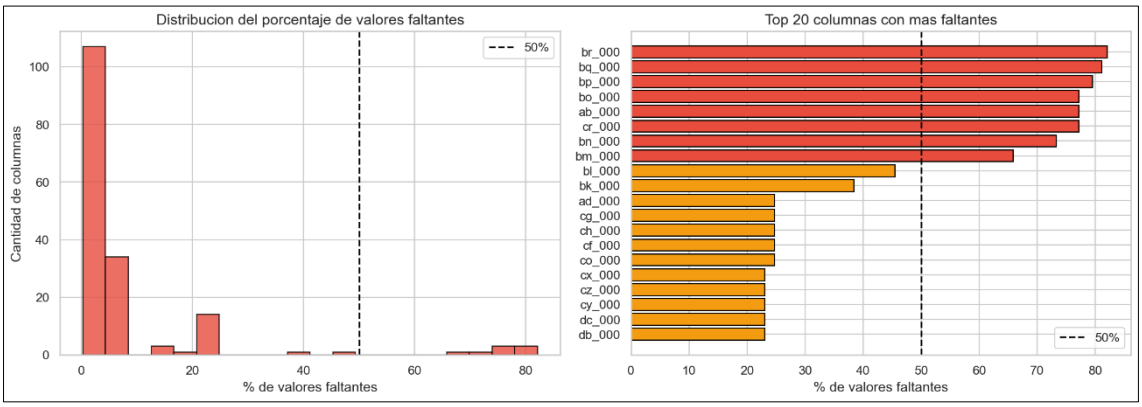

Figura 2. Análisis de valores faltantes en el conjunto de entrenamiento.

5.2 Modelado y manejo del desbalance

Tal como se vio en la bibliografía, los modelos basados en árboles de decisión son robustos para modelado de clasificación binaria ya que no asumen ninguna distribución estadística y manejan bien los datasets con muchas variables, como es el caso del dataset de Scania. Por lo tanto, se aborda el problema mediante el entrenamiento y comparación de métricas entre los modelos: *Random Forest* como modelo base y *XGBoost* como modelo de comparación basado en gradient boosting. Ambos modelos fueron configurados con la estrategia de *class weighting* para hacer frente al fuerte desbalance que existe entre la clase positiva (falla del APS) y la negativa (falla otro sistema), el cual es de 59:1.

Para implementarlo en el modelo, para el caso de Random Forest, se configura el parámetro class_*weight='balanced'*; mientras que para XGBoost se calcula el parámetro *scale_pos_weight* dividiendo la cantidad de registros negativos (59000) entre los positivos (1000), obteniendo un valor de 59,0. Esto está alineado con la documentación oficial de XGBoost y la bibliografía consultada [20]. Esta estrategia le indica al modelo que las consecuencias de cometer un error de clasificación no tienen el mismo peso, sino que equivocarse mediante un FN (predice que falla otro sistema, pero en realidad falla del APS) tiene un peso 59 veces mayor que un FP

(predice que falla el APS, pero en realidad falla otro sistema), forzándolo a prestar mayor atención a la clase minoritaria.

Ambos modelos fueron evaluados mediante validación cruzada estratificada de 5 folds, que preserva la proporción de clases en cada partición. Los resultados obtenidos se presentan en la Tabla 1.

Métrica	Random Forest	XGBoost
F1	0,7015 ± 0,0315	0,8119 ± 0,0175
Recall	0,5720 ± 0,0396	0,8100 ± 0,0114
Precision	0,9092 ± 0,0129	0,8144 ± 0,0309
AUC-ROC	0,9819 ± 0,0046	0,9868 ± 0,0022

Tabla 1. Comparación de métricas en validación cruzada estratificada (5 folds).

Los resultados de la Tabla 1 muestran que, como era de esperar, XGBoost supera a Random Forest en las métricas más relevantes. La métrica más importante en el área de mantenimiento es el Recall, ya que sería el caso de determinar que no es una falla del APS cuando realmente lo es. Implicaría tener que retirar el camión de circulación para llevarlo a reparar, y muchas veces no es solo uno, sino miles de unidades que ya están circulando por la calle. La diferencia más significativa se observa en el Recall, donde XGBoost alcanza 0,81 frente a 0,572 de Random Forest, lo que significa que XGBoost detecta el 81% de las fallas reales mientras que Random Forest solo detecta el 57%. El F1-score de XGBoost (0,8119) también supera al de Random Forest (0,7015), con una menor variabilidad entre folds, lo que indica mayor estabilidad. Ambos modelos obtienen valores de AUC-ROC superiores a 0,98, reflejando una excelente capacidad discriminativa general. La comparación visual de ambos modelos se presenta en la Figura 3.

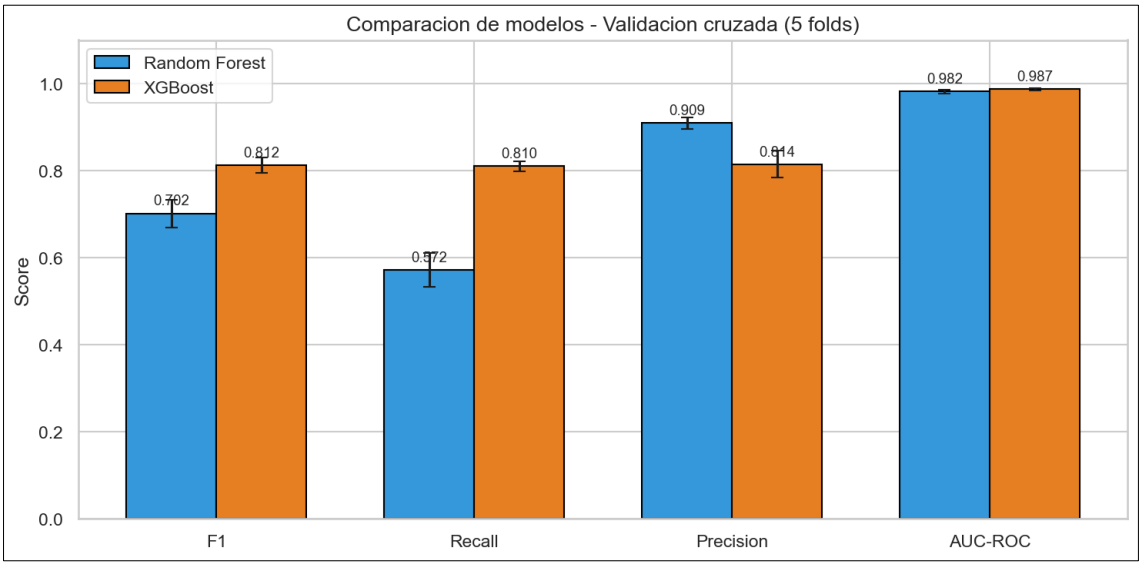

Figura 3. Comparación de métricas en validación cruzada entre Random Forest y XGBoost.

5.3 Ajuste del umbral de decisión

Una vez que se entrenaron y se compararon ambos modelos mediante validación cruzada estratificada de 5 folds que garantiza que la proporción de clases (59:1) se preserve en cada fold, lo cual es especialmente importante en datasets desbalanceados, se procede al ajuste del *umbral de decisión*. Por defecto, los clasificadores utilizan un umbral de 0,5 para determinar si una predicción corresponde a un resultado positivo o no. Sin embargo, como se describe en la sección de metodología, cuando los costos de error son asimétricos como el caso de nuestro dataset, este umbral deja de ser óptimo y es necesario realizar un cálculo adicional.

Aplicando la fórmula propuesta por Elkan [20] :

$$p* = \frac{(c_{FP} - c_{TN})}{c_{FP} - c_{TN} + c_{FN} - c_{TP}} \qquad \textit{Ecuación 7}$$

Donde:

$p*$ es el umbral óptimo de decisión.

c_{FN} es el costo de un FN: predice que no es falla del APS, pero sí lo es. Error grave, costo 500.

c_{FP} es el costo de un FP: predice falla del APS, pero no lo es. Error leve, costo 10.

c_{TN} es el costo de un TN (verdadero negativo): predice que no es falla del APS y realmente no lo es. Acierto, costo 0.

c_{TP} es el costo de un TP (verdadero positivo): predice falla del APS y realmente es falla del APS. Acierto, costo 0.

Aplicando los valores de la matriz de costos del dataset Scania ($c_{FN} = 500$, $c_{FP} = 10$, $c_{TN} = 0$, $c_{TP} = 0$), se obtiene un umbral teórico de 0,0196 (≈ 0,02), es decir un valor próximo al 2%. Este valor es el nuevo umbral, quiere decir que, cuando el modelo estime una probabilidad de falla del APS mayor al 2%, debe clasificar el registro como una falla del APS. Este cálculo matemático es solo un cálculo estimativo teórico el cual debe ser verificado de forma práctica. Para ello se utiliza la *función de costo total*, que para el dataset se define de la siguiente forma:

$$funcion\ de\ costo\ total = 500 * FN + 10 * FP \qquad \textit{Ecuación 8}$$

Donde:

- 500 es el costo unitario de un FN, definido explícitamente en la documentación del dataset Scania con este valor.
- FN es la cantidad de camiones con falla del APS real que el modelo no detectó que varía según el umbral de decisión utilizado.

Por lo tanto, el componente $500 * FN$ de la ecuación representa el costo generado por no detectar fallas reales para un determinado umbral.

- 10 es el costo unitario de un FP, definido en la documentación del dataset Scania con este valor.
- FP es la cantidad de camiones sin falla del APS que el modelo clasificó incorrectamente como falla que también varía según el umbral de decisión utilizado.

Por lo tanto, el componente $10 * FP$ de la ecuación es el costo generado por las falsas alarmas para un determinado umbral de decisión.

La suma de ambos términos da como resultado *el costo total que pagaría la empresa por los errores del modelo*, medido en unidades. En la práctica se evalúan 200 valores de umbrales de decisión diferentes sobre el conjunto de test, se calcula su FN y FP correspondiente, para usar la función de coste total y finalmente se selecciona el mínimo de ellos. La curva de costo total en función del umbral de decisión se presenta a continuación en la Figura 4.

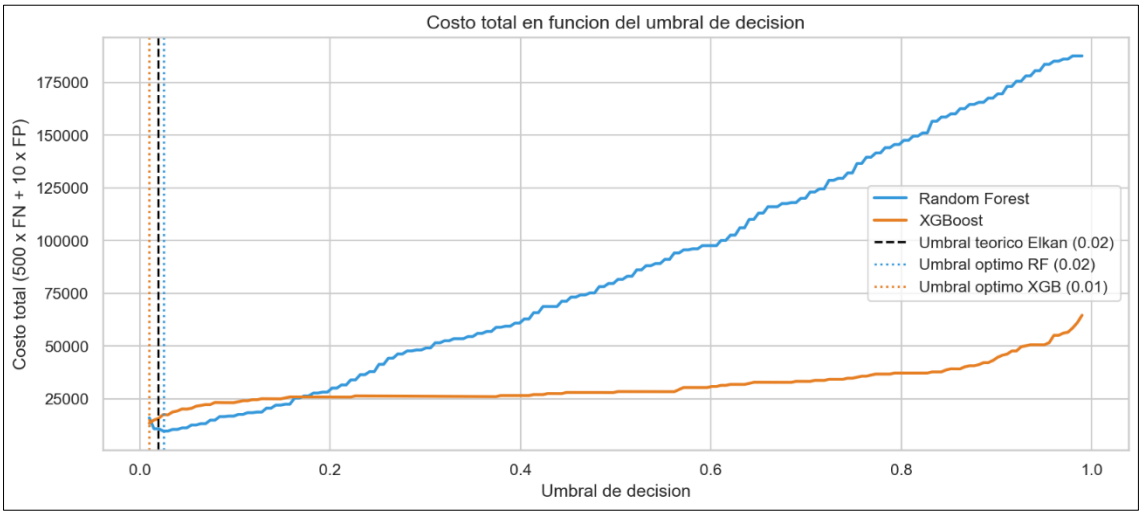

Figura 4. Costo total en función del umbral de decisión para Random Forest y XGBoost.

Se observa a partir de Figura 4, que los umbrales empíricos óptimos obtenidos fueron: 0,0248 para Random Forest y 0,0100 para XGBoost, ambos muy próximos al umbral teórico de Elkan de 0,0196. Esto permite validar de forma práctica lo obtenido de forma teórica.

A continuación, la Tabla 2 presenta de forma comparativa el desempeño de ambos modelos, tanto para el umbral por defecto (0,5) como para el umbral óptimo obtenido de forma empírica, sobre el conjunto de test.

Modelo	Umbral	FN	FP	Costo total
Random Forest	0,5	159	15	79.650
XGBoost	0,5	55	43	27.930
Random Forest	0,0248	7	620	9.700
XGBoost	0,01	21	294	13.440

Tabla 2. Comparación de resultados con umbral por defecto (0,5) y umbral óptimo empírico sobre el conjunto de test.

Los resultados demuestran que, con el umbral optimizado, Random Forest logra el menor costo total (9.700 unidades) superando a XGBoost (13.440 unidades) a pesar de haber obtenido métricas inferiores en la validación cruzada. Esto demuestra que la elección del mejor modelo no es universal, sino que depende de qué se quiere optimizar y en qué contexto se aplica.

5.4 Métricas de evaluación

A continuación, se avanza con la evaluación y comparación del desempeño real de ambos modelos sobre el conjunto de test. Para ello se calculan las métricas definidas en la sección metodología para cada combinación de modelo y umbral de decisión que se detallan en la siguiente tabla. Los resultados completos se presentan en la Tabla 3.

Modelo	Umbral	Costo total	F1	Recall	Precision	AUC-ROC
Random Forest	0,5	79.650	0,7129	0,576	0,9351	0,9918
XGBoost	0,5	27.930	0,8672	0,8533	0,8815	0,9948
Random Forest	0,0248	9.700	0,54	0,9813	0,3725	0,9918
XGBoost	0,01	13.440	0,6921	0,944	0,5463	0,9948

Tabla 3. Métricas de evaluación sobre el conjunto de test con umbral por defecto y umbral óptimo para cada modelo.

Los resultados permiten observar que:

Para el umbral óptimo, el Recall de ambos modelos aumenta considerablemente a costa de una reducción en la Precisión:

- Para Random Forest el Recall alcanza 0,9813, lo que significa que el modelo detecta el 98% de las fallas reales, dejando pasar únicamente 7 casos de los 375 positivos presentes en el conjunto de test, con un costo total de 9.700 unidades.
- Para XGBoost el Recall alcanza 0,9440, lo que indica que detecta el 94% de las fallas reales, dejando pasar 21 casos de los 375 positivos presentes en el conjunto de test, con un costo total de 13.440 unidades.

El AUC-ROC es superior a 0,99 en todos los casos. Se sabe que esta métrica no depende del umbral de decisión, motivo por el cual permanece igual. Esto se debe a que esta métrica evalúa la capacidad discriminativa general del modelo. Esto confirma lo señalado por Saito y Rehmsmeier [21] respecto a que el AUC-ROC puede resultar engañoso en datasets desbalanceados sin reflejar las diferencias reales en el costo operativo en todas las configuraciones de la tabla.

Las matrices de confusión, y las curvas Precision-Recall y ROC de ambos modelos se presentan a continuación en la Figura 5.

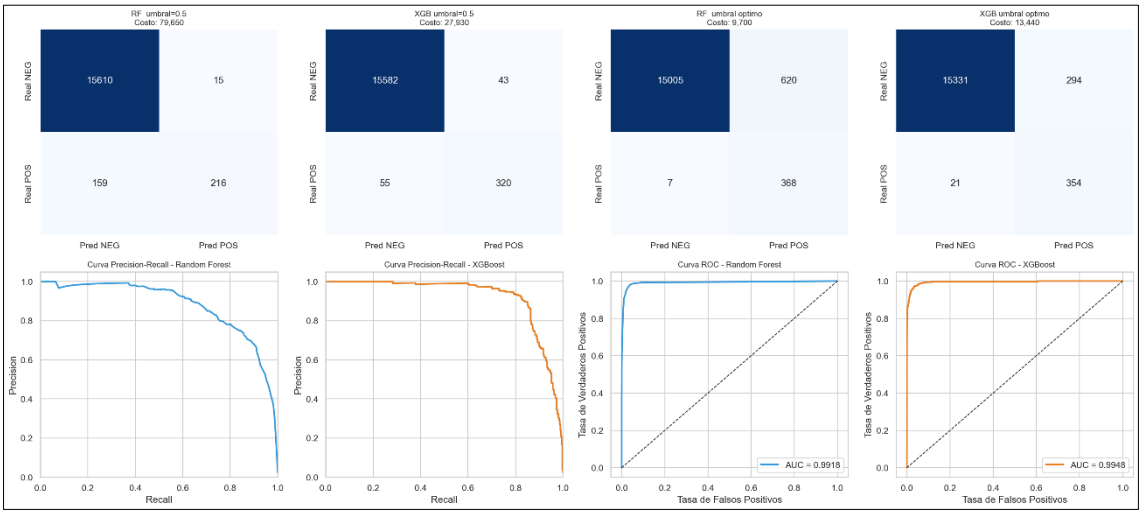

Figura 5. Matrices de confusión y curvas de evaluación para Random Forest y XGBoost con umbral por defecto y umbral óptimo.

5.5 Explicabilidad con SHAP

Continuando con las fases del proyecto, una vez evaluados ambos modelos, se aplica una técnica de XAI para entender cómo contribuye cada variable a la decisión final del modelo. La técnica elegida se llama *SHAP.* Para realizar su análisis se toma una muestra representativa de 2.000 registros del conjunto de test, seleccionados aleatoriamente. Para ello se utiliza el algoritmo explainer específico para modelos basados en árboles de decisión llamado *TreeExplainer*. Este es el más eficiente para modelos de árboles porque aprovecha la estructura interna del árbol para calcular los valores SHAP de forma exacta y muy rápida, sin necesidad de aproximaciones.

Importancia global de variables

La *importancia global de variables* es una medida que indica cuánto influye cada variable, en este caso sensor, en la decisión final del modelo considerando todo el conjunto de datos, no un caso en particular. Este es calculado como *el promedio del valor absoluto de los valores SHAP de cada variable sobre todos los registros analizados*, lo que permite identificar los sensores que mayor influencia tienen en las predicciones de cada modelo. Está claro que se usa el valor absoluto porque tanto las contribuciones positivas (falla del APS) como las negativas (no falla del APS) son igualmente importantes para entender la influencia de un sensor. Los resultados se presentan a continuación en la Figura 6.

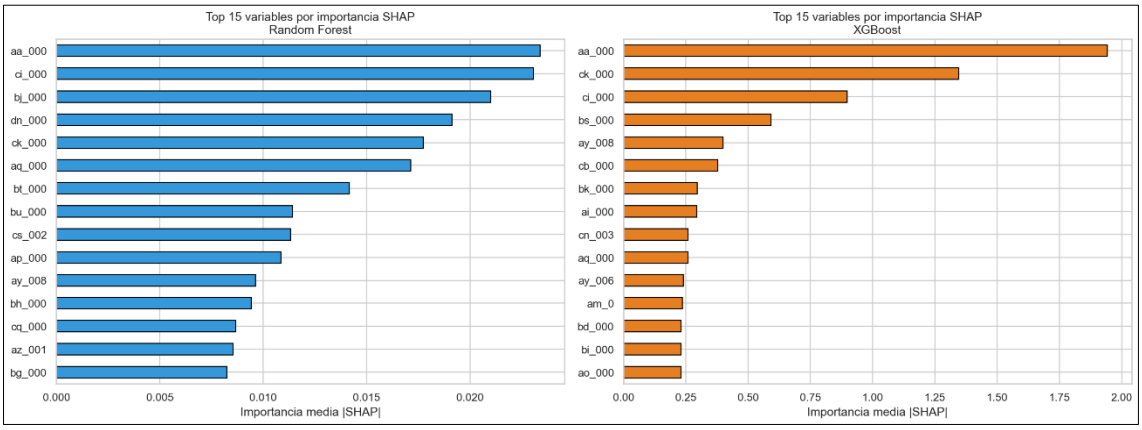

Figura 6. Importancia global de variables según valores SHAP para ambos modelos.

En ambos modelos:

- El sensor *aa_000* es la variable más influyente, seguido por *ci_000* y *ck_000*. Que sean coincidentes entre dos modelos entrenados con algoritmos distintos es un buen indicio, ya que otorga consistencia y robustez a los resultados.

Lo que los diferencia es que:

- En XGBoost la importancia está más concentrada en las primeras variables, con aa_000.
- En Random Forest la importancia se distribuye de forma más uniforme entre las variables del top 15.

Dirección de influencia de las variables

El apartado anterior determina la magnitud de la influencia de cada variable a la predicción final del modelo. Este apartado se encarga de determinar la dirección de influencia de dicha magnitud. Para poder visualizarlo, se utiliza el *beeswarm plot* presentado en la Figura 7, que complementa el análisis de importancia global añadiendo información sobre la dirección en que cada variable influye en la predicción, donde:

- Cada punto representa un registro del conjunto de datos.
- El eje vertical lista las variables ordenadas de mayor a menor importancia global.
- El eje horizontal muestra el valor SHAP de cada punto: cuanto más a la derecha, más empuja hacia falla del APS; cuanto más a la izquierda, más aleja de la falla.
- El color de cada punto indica el valor de la variable en ese registro: rojo significa valor alto del sensor y azul significa valor bajo.

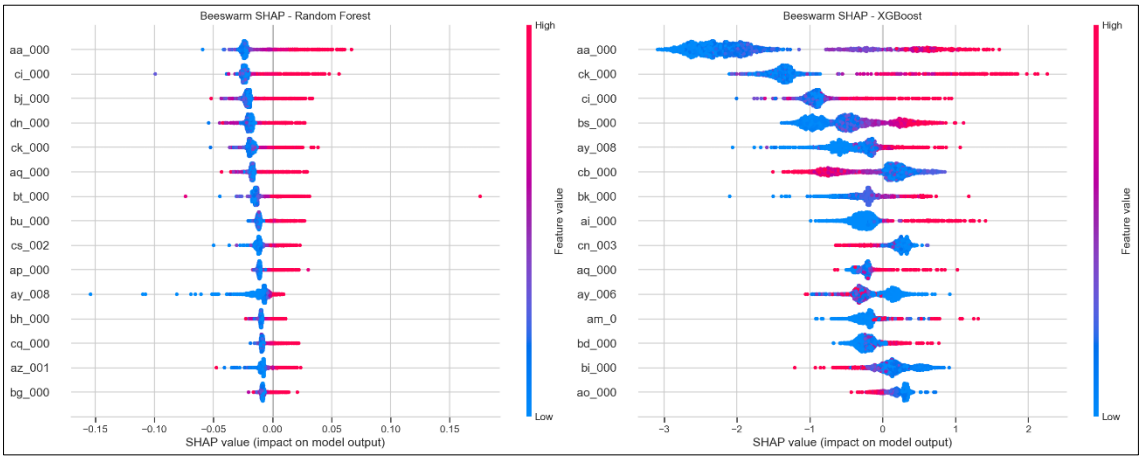

Figura 7. Beeswarm plot de valores SHAP para Random Forest y XGBoost. El color indica el valor de la variable (rojo = alto, azul = bajo) y la posición horizontal indica la dirección e intensidad de su contribución a la predicción.

Se puede observar que, en concordancia con lo visto anteriormente, en ambos modelos se ven valores altos del sensor *aa_000* (en color rojo), los cuales son superiores a los valores bajos (representados en azul) que la reducen, dando un resultante de predicción hacia la falla del APS. Este patrón se repite de forma consistente en los sensores *ci_000 y ck_000*, lo que permite concluir que lecturas elevadas en estos tres sensores son una señal característica de falla del APS.

Explicación de predicciones individuales

En la sección anterior, se muestra mediante el beeswarm el comportamiento global del modelo sobre todos los registros. A partir de ahora, se busca mediante el *waterfall plot* (gráfico de cascada) explicar la predicción de un caso individual concreto, mostrando paso a paso cómo cada variable contribuye a llevar la predicción desde el valor base hasta el valor final.

Para ilustrar la capacidad explicativa del sistema para un caso individual, se generaron *waterfall plots* para un caso positivo confirmado: el índice 37 con una probabilidad de falla de 0,9995. Los resultados se presentan en la **Error! Reference source not found.** para el modelo XGBoost y Figura 9 para el modelo Random Forest, donde:

- El punto de partida *E[f(X)]=0,093* es el valor base, mejor dicho, la predicción promedio del modelo sobre todo el conjunto de entrenamiento, lo que predice el modelo antes de analizar un sensor en particular.
- Cada barra representa cuánto contribuye cada variable específica a la decisión final:
    - Las barras rojas desplazan la predicción hacia un valor positivo, es decir hacia falla del APS.
    - Las barras azules empujan la predicción hacia un valor negativo, es decir que la falla proviene de otro componente del vehículo no relacionado con el APS.
- El valor final *f(x)* es la predicción del modelo para ese caso concreto, en este caso el índice 37, resultado de sumar todas las contribuciones al valor base.

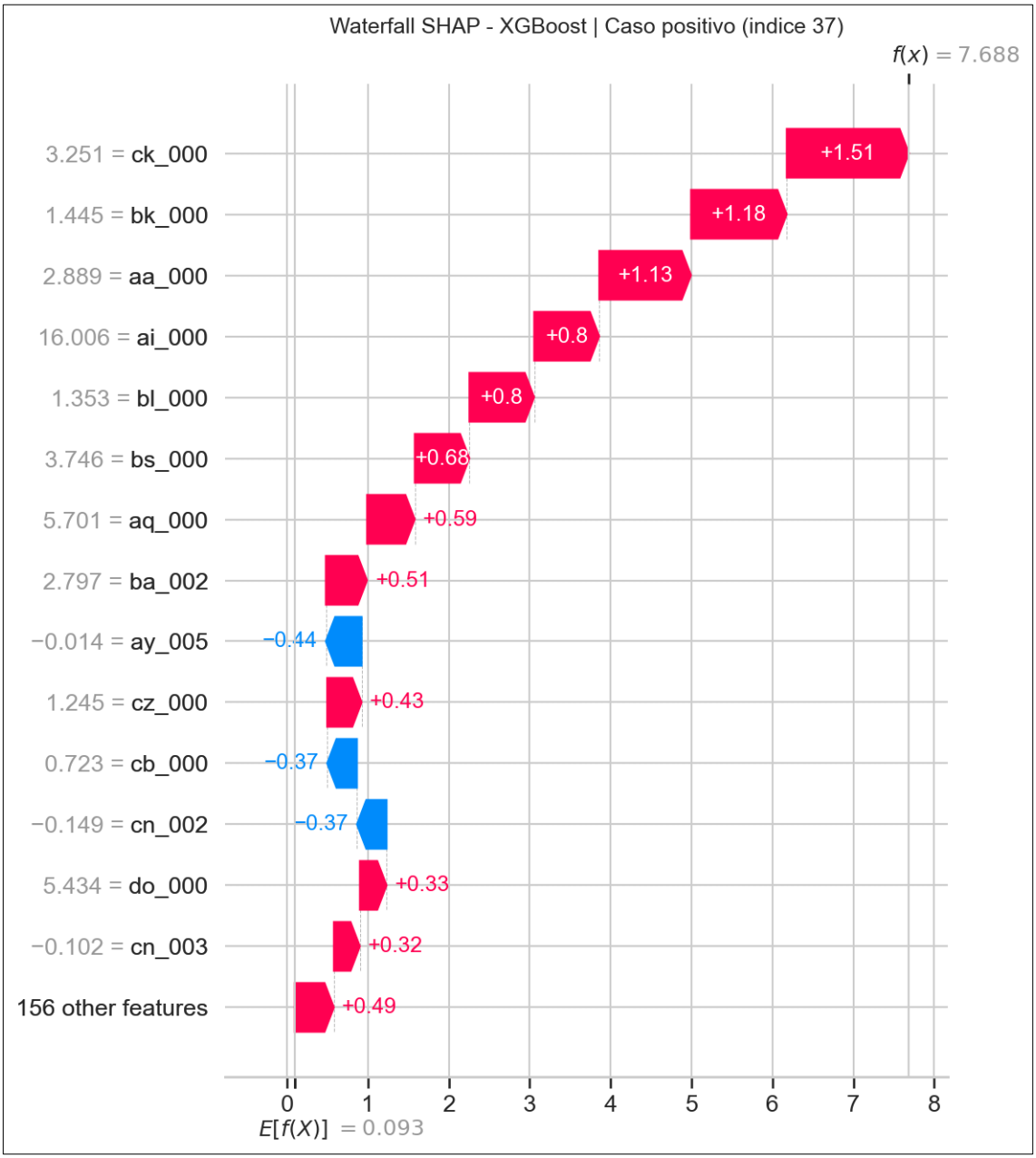

Figura 8. Waterfall plot de valores SHAP para XGBoost sobre el caso positivo confirmado índice 37 (probabilidad de falla APS = 0,9995).

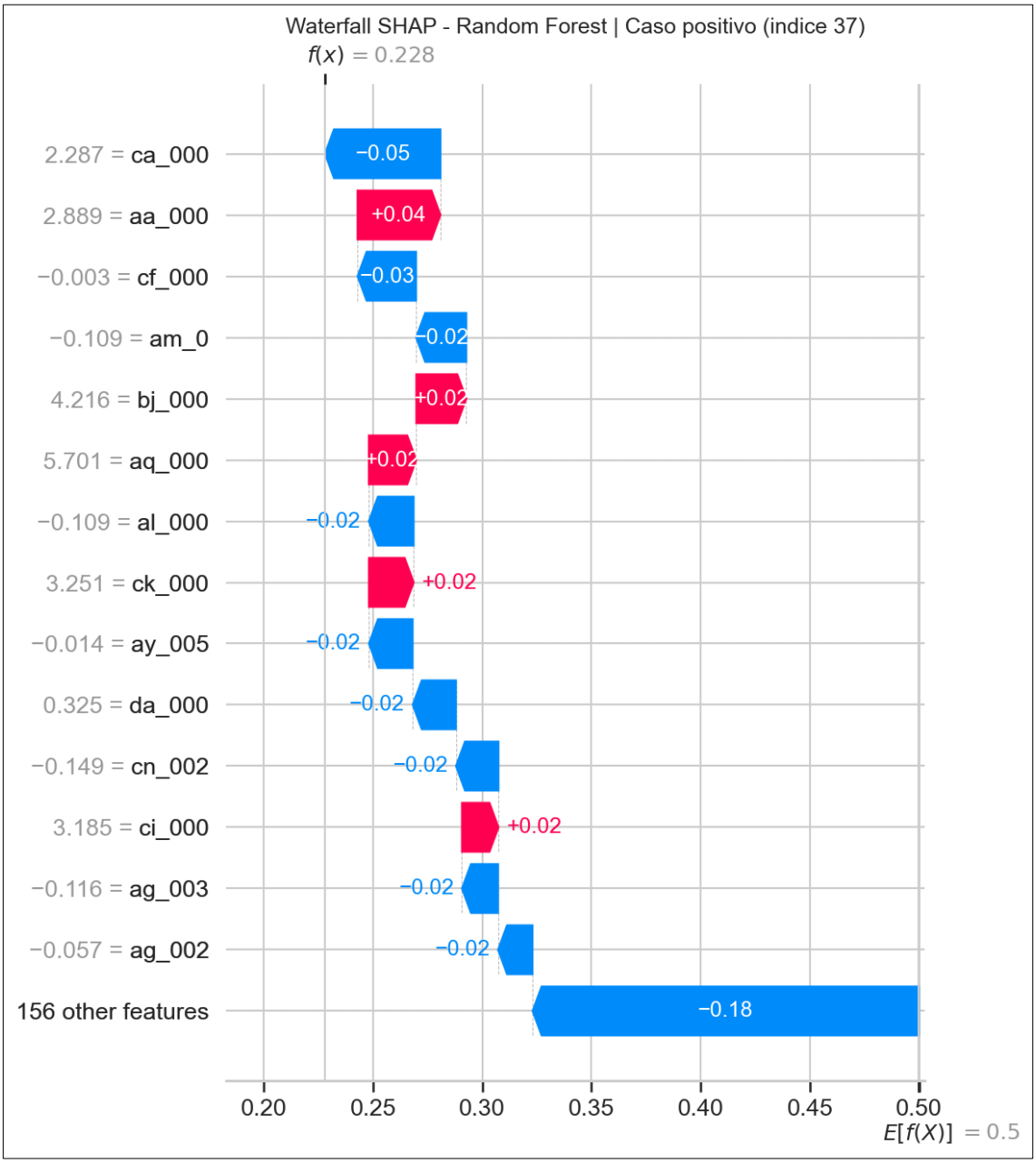

Figura 9. Waterfall plot de valores SHAP para Random Forest sobre el caso positivo confirmado índice 37 (probabilidad de falla APS = 0,9995).

Se observa para el caso anterior, con índice 37, que:

- XGBoost parte de un valor base de 0,093 y llega a una predicción de 7,688, con los sensores ck_000 (+1,51), bk_000 (+1,18) y aa_000 (+1,13) contribuyendo de forma positiva y significativa hacia la predicción de falla del APS.
- Random Forest parte de un valor base de 0,5 y llega a 0,228, con influencia entre un mayor número de sensores con contribuciones de menor magnitud que en XGBoost. Los que más influyen son *aa_000* (+0,04) el de mayor impacto positivo y *ca_000* (-0,05) el

de mayor impacto negativo. Esta diferencia de escala entre ambos modelos es característica de sus arquitecturas internas debido a cómo cada modelo construye sus predicciones internamente.

5.6 Diagnósticos generados por Claude

Avanzando hacia las fases finales del proyecto, se encuentra la integración de los resultados del modelo y los valores *SHAP* con la IA de *Claude* mediante la *API de Anthropic*. Esto tiene como objetivo crear diagnósticos en lenguaje natural adaptando su complejidad y vocabulario a cada perfil del usuario. Para validar el funcionamiento, se generaron diagnósticos para tres casos positivos confirmados del conjunto de test, cuyos índices son: 37, 44 y 61. Se generan diagnósticos utilizando los tres perfiles de usuario definidos en la metodología: operador, ingeniero y gerente.

A continuación, se presenta el diagnóstico generado para el índice 37, cuya probabilidad de falla del APS es de 0,9995, con los sensores ck_000, bk_000, aa_000, ai_000 y bl_000 como los más influyentes según SHAP.

Perfil operador:

"El APS presenta una falla confirmada con certeza muy alta (99,95%). El modelo detectó anomalías críticas en cinco sensores principales. Los sensores ck_000, bk_000 y aa_000 mostraron las lecturas más anormales y ejercieron mayor influencia en la detección de falla. Los sensores ai_000 y bl_000 también contribuyeron significativamente. Acción recomendada: detener la operación del vehículo de inmediato y realizar inspección visual del APS. Verifique fugas de aire, conexiones sueltas y estado de compresores. No reanude operaciones hasta validar que el sistema funciona correctamente."

Perfil ingeniero:

"El modelo de ML ha detectado una falla en el APS con una certeza del 99,95%. Los cinco sensores que más influyeron en esta detección son: ck_000 (SHAP=1,51), bk_000 (SHAP=1,18), aa_000 (SHAP=1,13), ai_000 (SHAP=0,80) y bl_000 (SHAP=0,80). Acción recomendada: programar inspección inmediata del APS, priorizando los componentes monitoreados por los sensores ck_000 y bk_000 dado su mayor impacto en la predicción. Se recomienda revisar válvulas, compresores y líneas de presión asociadas."

Perfil gerente:

"Se ha detectado una falla crítica en el APS con 99,95% de confiabilidad. Acción recomendada: programar mantenimiento preventivo urgente dentro de las próximas 24-48 horas. Se recomienda inspeccionar los componentes relacionados con los cinco sensores identificados, verificar fugas de aire, estado de válvulas y compresores. Esta intervención preventiva evitará paradas imprevistas costosas y riesgos de seguridad operacional."

Todos los perfiles transmiten la misma información técnica, pero la forma en la que es transmitida debe ser diferente. Entre ellos cambia el nivel de detalle y el vocabulario ya que el

operador recibe instrucciones concretas y simples orientadas a la acción inmediata, el ingeniero obtiene los valores SHAP específicos para priorizar la inspección técnica orientado a una solución definitiva, y el gerente recibe una visión orientada al impacto operacional y económico (tiempo de vehículo detenido, costos implicados, y *stakeholders*).

Si el usuario considera escaso el informe o precisa información específica, el sistema permite realizar consultas en lenguaje natural mediante un módulo de chat integrado. Para ejemplificar, se plantea la pregunta: '¿Cuál es el sensor más crítico y qué se debería revisar primero?'. El sistema respondió priorizando el sensor ck_000 por su mayor valor SHAP (1,5106) y listando los pasos de inspección en orden de criticidad, manteniendo en todo momento el contexto de la predicción activa y sin generar información adicional no respaldada por los datos del modelo.

5.7 Despliegue del sistema

Con todo listo para ser mostrado al usuario, la solución desarrollada fue desplegada mediante dos componentes complementarios: una *API REST* desarrollada en FastAPI y un dashboard interactivo desarrollado en *Streamlit*, ambos integrados en un mismo entorno de ejecución local.

API REST (FastAPI)

La API REST es el componente del sistema que permite la comunicación entre el dashboard y los modelos entrenados mediante peticiones HTTP. Expone cuatro endpoints, es decir, puntos de acceso, identificados por su método HTTP (GET para consultas, POST para envío de datos) y su ruta:

- GET /health para verificar el estado del servicio,
- POST /predecir para recibir los valores de los sensores y devolver la predicción junto con los valores SHAP más influyentes,
- POST /diagnostico para generar un diagnóstico en lenguaje natural adaptado al perfil del usuario, y
- POST /chat para realizar consultas en lenguaje natural sobre una predicción activa manteniendo el contexto del caso analizado.

Dashboard interactivo (Streamlit)

Streamlit es una librería de Python de código abierto que permite crear aplicaciones web interactivas de forma rápida y sencilla. Fue utilizado para construir un dashboard interactivo que integra todas las funcionalidades del sistema en una interfaz visual accesible sin necesidad de conocimientos técnicos en ML para que todos los empleados de una compañía puedan utilizarlo. A continuación, se muestra en la Figura 10 la interfaz inicial del dashboard antes de ejecutar un análisis.

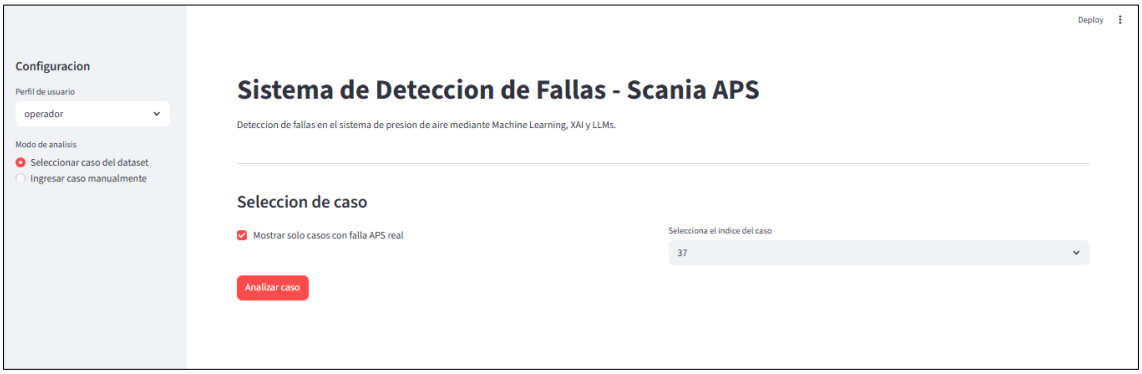

Figura 10. Interfaz inicial del dashboard interactivo antes de ejecutar un análisis.

La interfaz se organiza de la siguiente forma:

- Un panel lateral izquierdo donde se encuentra:
  la configuración: donde el usuario selecciona su perfil (operador, ingeniero o gerente) y el modo de análisis (selección de un caso del dataset o ingreso manual de valores de sensores).
- Un panel central donde se presenta la sección del caso: el usuario puede filtrar únicamente los casos con falla en el APS real confirmada mediante un checkbox y seleccionar el índice del caso a analizar mediante un desplegable. Finalmente, un botón para ejecutar el análisis haciendo clic en "Analizar caso".

En este ejemplo, una vez ejecutado el análisis sobre el caso índice 37, el sistema presenta los resultados en tres bloques diferenciados, tal como se muestra en la siguiente Figura 11.

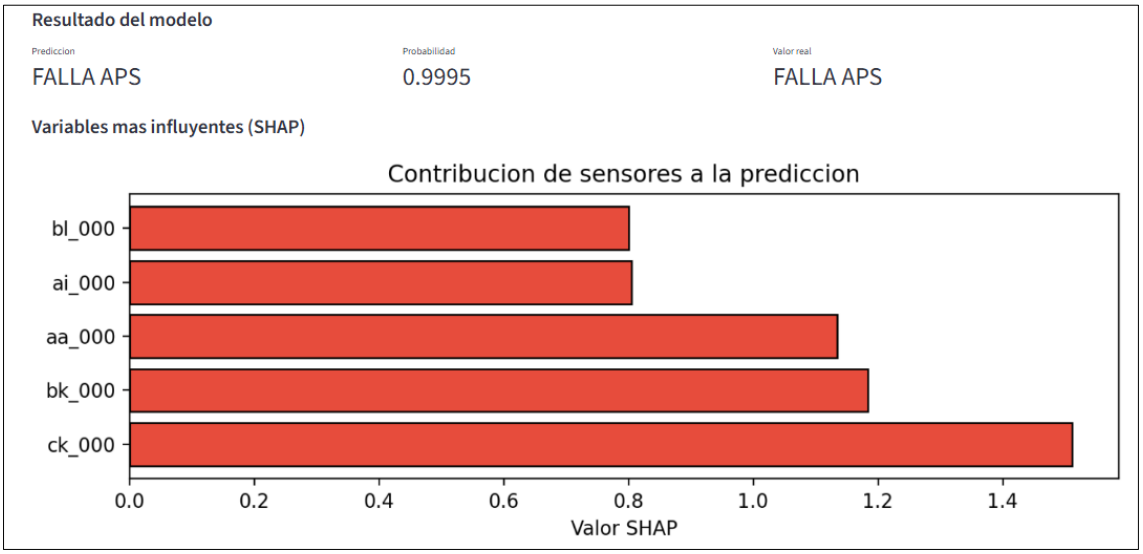

Figura 11. Resultado del modelo y contribución de sensores según SHAP para el caso índice 37.

Los resultados están divididos en:

- Un primer bloque correspondiente al resultado del modelo, donde se presentan la predicción (FALLA el APS), la probabilidad estimada (0,9995) y el valor real del caso (FALLA el APS), confirmando que la predicción es correcta.
- Un segundo bloque que presenta el gráfico de barras horizontales con los cinco sensores más influyentes según SHAP, donde se puede observar que *ck_000* es el sensor con mayor contribución a la predicción de falla (SHAP=1,51), seguido por *bk_000* (SHAP=1,18) y *aa_000* (SHAP=1,13).
  Todos los sensores presentan valores SHAP positivos, lo que indica que sus lecturas actuales aumentan el riesgo de falla del APS.

Seguidamente, el usuario puede solicitar la generación del diagnóstico en lenguaje natural haciendo clic en el botón "Generar diagnóstico con Claude". La Figura 12 a continuación muestra el diagnóstico generado para el perfil “operador” sobre el caso de falla identificado por el índice 37.

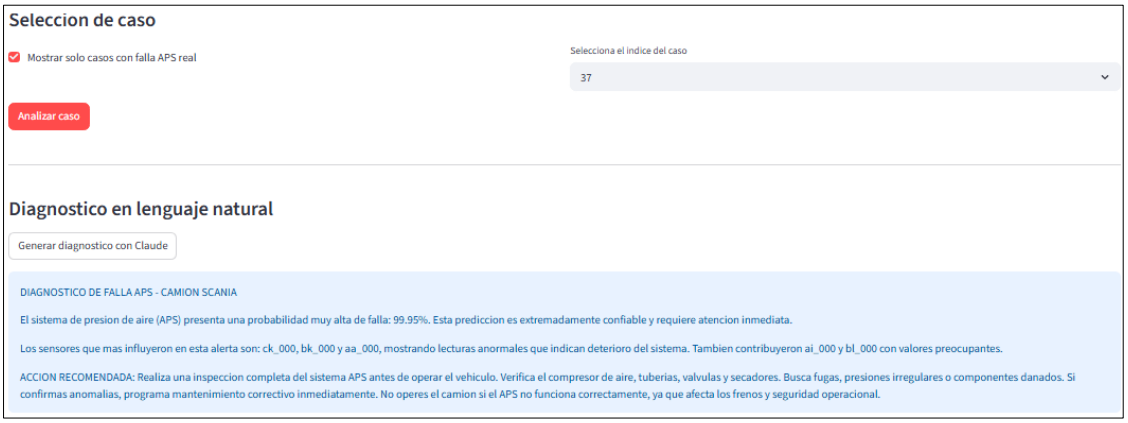

Figura 12. Diagnóstico generado en lenguaje natural por Claude para el caso índice 37, perfil operador.

El texto generado es un texto estructurado que principalmente identifica si la falla proviene o no del APS, posteriormente menciona los sensores más críticos y finalmente proporciona una acción concreta de mantenimiento, adaptando el vocabulario y el nivel de detalle de acuerdo con el perfil seleccionado. Para el ejemplo de la Figura 12, el diagnóstico indica que el APS presenta una probabilidad muy alta de falla del 99,95%, señala los sensores ck_000, bk_000 y aa_000 como los más críticos y recomienda realizar una inspección completa del APS antes de operar el vehículo.

Por último, en el caso de que el usuario precise realizar una consulta adicional, el sistema permite recibir consultas adicionales en lenguaje natural mediante el módulo de chat integrado, que mantiene el contexto de la predicción activa, es decir que solo formula respuestas teniendo en cuenta las predicciones del modelo. A modo de ejemplo, la Figura 13 muestra la respuesta generada ante la pregunta "¿Cuál es el sensor más crítico y qué debería revisar primero?".

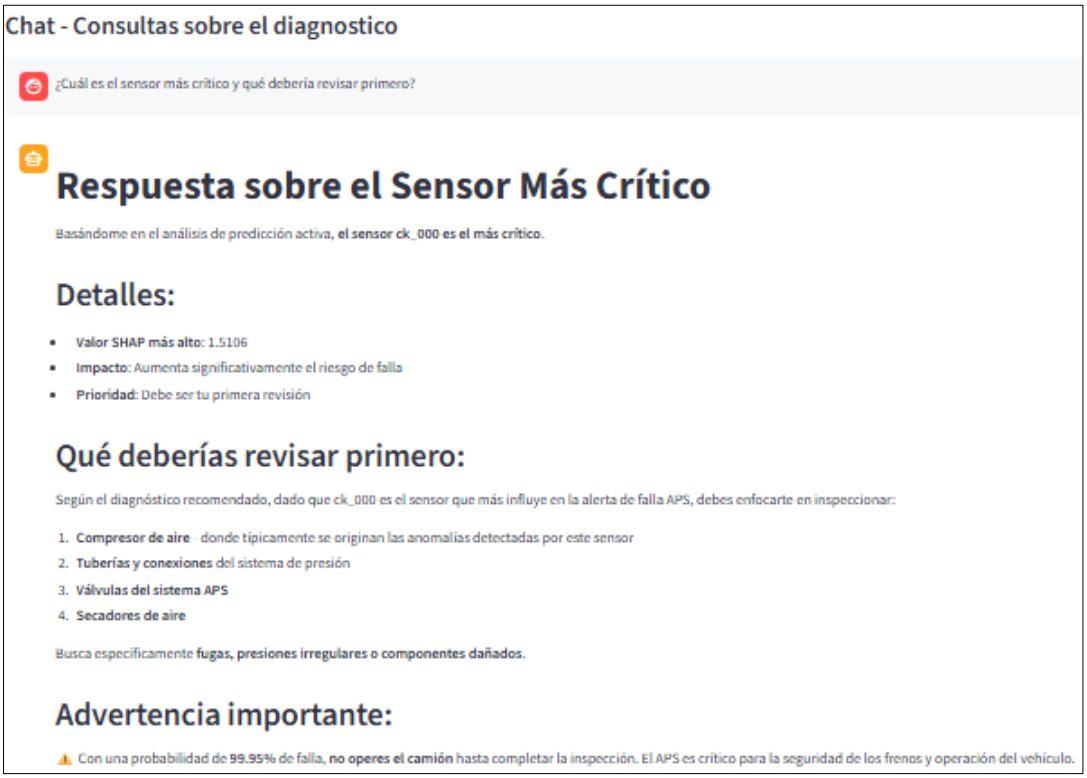

Figura 13. Respuesta del módulo de chat ante la consulta sobre el sensor más crítico del caso índice 37.

En su respuesta, el sistema identifica al sensor *ck_000* como el más crítico por su valor SHAP de 1,5106. También detalla los componentes específicos que se deben inspeccionar en orden de prioridad (compresor de aire, tuberías y conexiones, válvulas del APS y secadores de aire), y finalmente incluye una advertencia operacional indicando que no se debe operar el camión hasta completar la inspección.

Capítulo 6. Discusión

6.1 Preprocesamiento y el dataset

El análisis exploratorio del dataset confirmó las características propias de un conjunto de datos de mantenimiento industrial real:

- un marcado desbalance de clases (relación 59:1),
- una elevada proporción de valores faltantes (en 169 de 170 variables) y
- la presencia de valores extremos en los datos de sensores.

Estas características, lejos de ser un obstáculo, representan fielmente las condiciones que se encuentran en entornos industriales reales, lo que otorga validez y relevancia práctica a los resultados obtenidos.

Imputación por mediana

La decisión de imputar por mediana en lugar de eliminar las variables con alto porcentaje de valores faltantes resultó ser la mejor decisión, ya que permitió conservar las 170 variables predictoras sin introducir sesgos en el análisis. Eliminar las 8 variables con más del 50% de valores faltantes hubiera supuesto descartar información potencialmente relevante sobre el comportamiento del APS, y como se observó en el análisis SHAP posterior, algunas de las variables más influyentes en las predicciones presentaban valores faltantes significativos.

6.2 Modelado y desbalance

Los resultados obtenidos en la validación cruzada confirmaron que XGBoost supera a Random Forest en las métricas más relevantes para este problema, especialmente en *recall* (0,81 frente a 0,57). Esta diferencia se explica por la naturaleza del algoritmo de gradient boosting, que construye los árboles de forma secuencial corrigiendo los errores del modelo anterior, lo que lo hace especialmente efectivo para aprender patrones de la clase minoritaria cuando se combina con la estrategia de *class weighting*.

Sin embargo, el hallazgo más relevante de esta etapa es que al ajustar el umbral de decisión, Random Forest logra el menor costo total sobre el conjunto de test (9.700 unidades) superando a XGBoost (13.440 unidades), a pesar de haber obtenido métricas inferiores en la validación cruzada. Este resultado ilustra una limitación importante de las métricas estándar de ML: el modelo con mejor F1 o mejor Recall en validación cruzada no es necesariamente el que minimiza el costo operativo real cuando los errores tienen costos asimétricos. Esto es consistente con lo planteado por Elkan [20], quien argumenta que en problemas con costos asimétricos la optimización del umbral de decisión es tan importante como la selección del algoritmo.

La validación empírica del umbral teórico de Elkan también mostró resultados relevantes. Los umbrales óptimos empíricos obtenidos (0,0248 para Random Forest y 0,0100 para XGBoost) se encuentran muy próximos al umbral teórico de 0,0196, lo que valida la fórmula de Elkan como punto de partida sólido para problemas con costos asimétricos, aunque confirma también la

necesidad de una validación empírica posterior dado que las probabilidades de los modelos basados en árboles no siempre están perfectamente calibradas.

6.3 Explicabilidad con SHAP

La aplicación de SHAP sobre ambos modelos permitió identificar un conjunto consistente de sensores críticos, con *aa_000*, *ci_000* y *ck_000* posicionándose entre las variables (sensores) más influyentes en ambos modelos. Esta consistencia entre dos algoritmos entrenados de forma independiente refuerza la confianza en que dichos sensores efectivamente capturan información relevante sobre el estado del APS, y no son simplemente artefactos de un modelo particular.

El *beeswarm plot* (Figura 7) reveló que los valores elevados de estos sensores están sistemáticamente asociados a predicciones de falla del APS, lo que proporciona a los equipos de mantenimiento una hipótesis concreta sobre qué componentes del sistema inspeccionar prioritariamente. Este tipo de información, que va más allá de una simple alerta de falla, es precisamente lo que Cummins et al. [10] identifican como una de las principales carencias de los sistemas de mantenimiento predictivo actuales: la falta de explicaciones diseñadas para audiencias específicas y orientadas a la acción.

La diferencia de escala observada entre los valores SHAP de XGBoost y Random Forest, donde XGBoost presenta valores considerablemente más elevados, es una característica inherente a las arquitecturas internas de cada modelo y no afecta la interpretabilidad de los resultados. Lo relevante es la ordenación relativa de los sensores y la dirección de su influencia, que resultan consistentes entre ambos modelos.

6.4 Integración con Claude y los diagnósticos generados

La integración con *Claude* mediante la API de *Anthropic* demostró ser efectiva para traducir los resultados técnicos del modelo en diagnósticos comprensibles y accionables para distintos perfiles de usuario. Los diagnósticos generados para los tres perfiles (operador, ingeniero y gerente) mantuvieron en todos los casos la información técnica, pero adaptaron el vocabulario, el nivel de detalle y el enfoque de la recomendación al perfil destinatario, sin generar información adicional no respaldada por los datos obtenidos mediante los modelos.

Este resultado es especialmente relevante en el contexto de lo planteado por Walker et al. [11], quienes destacan la importancia de adaptar las explicaciones de los modelos de IA a diferentes niveles de usuario como condición necesaria para generar confianza en los sistemas de mantenimiento predictivo. El sistema desarrollado en este trabajo implementa precisamente ese principio de forma automática, sin requerir que el equipo de mantenimiento tenga conocimientos técnicos en ML para interpretar los resultados.

La instrucción explícita incluida en el prompt para que Claude no genere información adicional no respaldada por los datos de la predicción demostró ser efectiva, siguiendo la recomendación de Li et al. [9] sobre la importancia de limitar las alucinaciones en GenAI aplicada a

mantenimiento industrial. En ninguno de los diagnósticos generados se observó información inventada o inconsistente con los valores SHAP proporcionados.

6.5 Limitaciones del sistema

A pesar de los resultados obtenidos, el sistema presenta algunas limitaciones que deben ser consideradas. En primer lugar, el modelo fue entrenado y evaluado sobre el dataset Scania, por lo que su aplicación directa a otros sistemas industriales requeriría un proceso de reentrenamiento con datos específicos de cada contexto. Si bien la arquitectura del pipeline fue diseñada para ser generalizable, los modelos entrenados no lo son.

En segundo lugar, el análisis SHAP se realizó sobre una muestra de 2.000 registros del conjunto de test por razones de eficiencia computacional, lo que podría introducir pequeñas variaciones en los valores de importancia global respecto a un análisis sobre el conjunto completo.

En tercer lugar, el sistema depende de la disponibilidad de la API de Anthropic para la generación de diagnósticos y el módulo de chat, lo que introduce una dependencia externa que debe ser considerada en un despliegue productivo real.

Finalmente, el dashboard desarrollado opera en un entorno local, por lo que un despliegue en producción requeriría una infraestructura adicional de servidores y mecanismos de autenticación para garantizar la seguridad del acceso.

Capítulo 7. Conclusiones

El presente trabajo tuvo como objetivo diseñar e implementar una solución end-to-end basada en IA para la detección, diagnóstico y entendimiento de fallos en sistemas basados en sensores, integrando modelos de ML, técnicas de XAI y modelos de lenguaje para la generación de explicaciones interpretables. A continuación, se presentan las conclusiones obtenidas en relación a cada uno de los objetivos específicos planteados.

Pipeline de procesamiento y modelado

Se desarrolló exitosamente un pipeline completo de procesamiento y modelado sobre el dataset "APS Failure at Scania Trucks", que abarca desde la carga y exploración del dataset hasta el entrenamiento, evaluación y comparación de modelos. El pipeline demostró ser robusto frente a las condiciones propias de un dataset de mantenimiento industrial real: marcado desbalance de clases (relación 59:1), elevada proporción de valores faltantes (169 de 170 variables) y presencia de valores extremos en los datos de sensores. La estrategia de imputación por mediana y normalización mediante *StandardScaler* permitió preparar el dataset de forma adecuada para el modelado, mientras que la estrategia de *class weighting* permitió abordar el desbalance de clases sin necesidad de generar datos sintéticos.

La comparación entre Random Forest y XGBoost demostró que XGBoost supera a Random Forest en las métricas estándar de validación cruzada, especialmente en Recall (0,81 frente a 0,57). Sin embargo, el hallazgo más relevante de esta etapa fue que al ajustar el umbral de decisión mediante la fórmula de Elkan, Random Forest logra el menor costo total operativo (9.700 unidades frente a 13.440 de XGBoost), demostrando que la elección del mejor modelo no puede basarse únicamente en las métricas estándar, sino que debe considerar el contexto operativo y la matriz de costos del problema.

Análisis de interpretabilidad mediante SHAP

La aplicación de *SHAP* sobre ambos modelos permitió identificar un conjunto consistente de sensores críticos, con aa_000, ci_000 y ck_000 posicionándose entre las variables más influyentes en ambos modelos de forma independiente. Esta consistencia refuerza la confianza en que dichos sensores efectivamente capturan información relevante sobre el estado del APS y no son artefactos de un modelo particular.

El análisis de la dirección de influencia mediante el *beeswarm plot* reveló que valores elevados de estos sensores están sistemáticamente asociados a predicciones de falla del APS, proporcionando a los equipos de mantenimiento hipótesis concretas sobre qué componentes inspeccionar prioritariamente. Los *waterfall plots* demostraron la capacidad del sistema para explicar predicciones individuales de forma detallada, identificando qué sensores fueron determinantes en cada caso concreto y en qué magnitud y dirección influyeron en la predicción final.

Integración con modelos de lenguaje

La integración con *Claude* mediante la *API* de *Anthropic* demostró ser efectiva para traducir los resultados técnicos del modelo en diagnósticos comprensibles y accionables para distintos perfiles de usuario. Los diagnósticos generados para los perfiles de operador, ingeniero y gerente mantuvieron en todos los casos la información técnica de base, adaptando el vocabulario, el nivel de detalle y el enfoque de la recomendación al destinatario sin generar información adicional no respaldada por los datos del modelo.

El módulo de chat integrado demostró ser capaz de mantener el contexto de la predicción activa y responder consultas específicas en lenguaje natural, priorizando la información según los valores SHAP del caso analizado. Este resultado confirma que la combinación de ML, XAI y modelos de lenguaje permite construir sistemas de mantenimiento predictivo que no solo detectan fallos, sino que también los explican y comunican de forma comprensible para cualquier nivel de la organización.

Despliegue del sistema

La solución fue desplegada exitosamente mediante una *API REST* desarrollada en *FastAPI* y un dashboard interactivo desarrollado en *Streamlit*, integrando en una única interfaz las funcionalidades de predicción, explicabilidad y diagnóstico en lenguaje natural. El sistema demostró ser funcional y accesible para usuarios sin conocimientos técnicos en ML, cumpliendo con el objetivo de facilitar su uso en entornos operativos reales.

Reflexión final

Los resultados obtenidos en este trabajo demuestran que es posible construir sistemas de mantenimiento predictivo que vayan más allá de la simple detección de fallos, incorporando explicabilidad y comunicación comprensible de los resultados como componentes fundamentales del sistema. La integración de ML, XAI y modelos de lenguaje en un mismo pipeline analítico representa un paso hacia sistemas de mantenimiento predictivo más transparentes, confiables y accesibles para todos los niveles de una organización industrial.

Como líneas de trabajo futuro se identifican las siguientes oportunidades de mejora:

- La incorporación de documentación técnica específica (planos eléctricos, mecánicos o neumáticos) para enriquecer los diagnósticos generados por el modelo de lenguaje.
- La extensión del pipeline a otros datasets industriales para validar la generalización del enfoque.
- La implementación de mecanismos de reentrenamiento automático del modelo ante la llegada de nuevos datos.
- El despliegue del sistema en una infraestructura cloud que permita su uso en producción por múltiples usuarios de forma simultánea.

Declaración de uso de Inteligencia Artificial

En la elaboración del presente trabajo se ha utilizado apoyo de herramientas de Inteligencia Artificial en distintas fases, diferenciando claramente su rol como parte del sistema desarrollado y como asistencia en la revisión del documento.

Integración técnica en el sistema

Se ha utilizado la herramienta Claude (Anthropic), mediante su API, como componente funcional del sistema desarrollado.

Su función consiste en la generación de diagnósticos en lenguaje natural adaptados a distintos perfiles de usuario (operador, ingeniero y gerente), a partir de los resultados del modelo de machine learning y los valores SHAP obtenidos.

Esta integración forma parte del diseño e implementación del sistema propuesto y se encuentra debidamente documentada en las secciones 4.7 y 5.6 del presente trabajo.

Apoyo en la revisión del documento

Se ha utilizado la herramienta Claude (Anthropic), a través de su interfaz web (claude.ai), como asistencia en tareas de revisión del documento, incluyendo:

- Corrección ortográfica y gramatical
- Verificación de coherencia en el uso de acrónimos
- Ajustes de formato en expresiones matemáticas

En ningún caso estas herramientas han sido utilizadas para la generación autónoma del contenido académico, análisis, resultados o conclusiones del trabajo, siendo estos desarrollados íntegramente por el autor.

Bibliografía

[1] A. Ucar, M. Karakose, and N. Kırımça, "Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends," Jan. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app14020898.

[2] Y. Mahale, S. Kolhar, and A. S. More, "A comprehensive review on artificial intelligence driven predictive maintenance in vehicles: technologies, challenges and future research directions," Apr. 01, 2025, *Springer Nature*. doi: 10.1007/s42452-025-06681-3.

[3] W. Liu *et al.*, "Unsupervised Deep Anomaly Detection for Industrial Multivariate Time Series Data," *Applied Sciences (Switzerland)*, vol. 14, no. 2, Jan. 2024, doi: 10.3390/app14020774.

[4] M. F. de la Peña, Á. L. P. Gómez, and L. F. Maimó, "ShaTS: A Shapley-based Explainability Method for Time Series Artificial Intelligence Models applied to Anomaly Detection in Industrial Internet of Things," Jun. 2025, [Online]. Available: http://arxiv.org/abs/2506.01450

[5] L. G. Di Maggio, "Toward Autonomous LLM-Based AI Agents for Predictive Maintenance: State of the Art, Challenges, and Future Perspectives," Nov. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app152111515.

[6] T. Bitam, A. Yahiaoui, D. E. Boubiche, R. Martínez-Peláez, H. Toral-Cruz, and P. Velarde-Alvarado, "Artificial Intelligence of Things for Next-Generation Predictive Maintenance," Dec. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/s25247636.

[7] S. Aslam *et al.*, "Machine Learning-Based Predictive Maintenance at Smart Ports Using IoT Sensor Data," *Sensors*, vol. 25, no. 13, Jul. 2025, doi: 10.3390/s25133923.

[8] S. Elkateb, A. Métwalli, A. Shendy, and A. E. B. Abu-Elanien, "Machine learning and IoT – Based predictive maintenance approach for industrial applications," *Alexandria Engineering Journal*, vol. 88, pp. 298–309, Feb. 2024, doi: 10.1016/j.aej.2023.12.065.

[9] X. Li *et al.*, "Generative AI for Predictive Maintenance for Manufacturing: A Survey," *Procedia CIRP*, vol. 139, pp. 35–40, 2026, doi: 10.1016/j.procir.2025.09.012.

[10] L. Cummins *et al.*, "Explainable Predictive Maintenance: A Survey of Current Methods, Challenges and Opportunities," *IEEE Access*, vol. 12, pp. 57574–57602, 2024, doi: 10.1109/ACCESS.2024.3391130.

[11] "INL/RPT-23-74159 Revision 0 Explainable Artificial Intelligence Technology for Predictive Maintenance," 2023. [Online]. Available: http://www.lwrs.gov

[12] S. Pashami *et al.*, "Explainable Predictive Maintenance," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.05120

[13] D. Meli, “Explainable Online Unsupervised Anomaly Detection for Cyber-Physical Systems via Causal Discovery from Time Series,” Jul. 2024, [Online]. Available: http://arxiv.org/abs/2404.09871

[14] E. Birihanu and I. Lendák, “Explainable correlation-based anomaly detection for Industrial Control Systems,” *Front. Artif. Intell.*, vol. 7, 2024, doi: 10.3389/frai.2024.1508821.

[15] K. K. Kim, J. S. Kim, and I. C. Euom, “Explainable Anomaly Detection Based on Operational Sequences in Industrial Control Systems,” *IEEE Access*, vol. 13, pp. 66170–66187, 2025, doi: 10.1109/ACCESS.2025.3560260.

[16] R. Saleh, K. P. Dinh, B. Villányi, and T.-S. Hy, “Self-Evolving Multi-Agent Network for Industrial IoT Predictive Maintenance,” Feb. 2026, [Online]. Available: http://arxiv.org/abs/2602.16738

[17] UCI Machine Learning Repository, “APS Failure at Scania Trucks,” UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks.

[18] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, “A survey on missing data in machine learning,” *J. Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00516-9.

[19] B. Krawczyk, “Learning from imbalanced data: open challenges and future directions,” Nov. 01, 2016, *Springer Verlag*. doi: 10.1007/s13748-016-0094-0.

[20] C. Elkan, “The Foundations of Cost-Sensitive Learning,” Seattle, Aug. 2001. Accessed: Apr. 07, 2026. [Online]. Available: https://www.ijcai.org/Proceedings/01/Papers/039.pdf

[21] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PLoS One*, vol. 10, no. 3, Mar. 2015, doi: 10.1371/journal.pone.0118432.