MANTENIMIENTO PREDICTIVO MEDIANTE ANÁLISIS DE DATOS Y MACHINE LEARNING USANDO XGBOOST

Ing. Alpaca Rendón, Jesús A.

Ing. Del Carpio Maraza Heberth E.

Ing. Vargas Franco, Mauricio S.

Resumen Ejecutivo

El mantenimiento predictivo es una estrategia que permite anticipar fallos en los equipos industriales antes de que ocurran, lo que ayuda a reducir el tiempo de inactividad y los costos asociados con las reparaciones inesperadas. A diferencia de los enfoques tradicionales de mantenimiento correctivo y preventivo, el mantenimiento predictivo utiliza datos históricos y en tiempo real para predecir cuándo es probable que ocurra una falla. Con el avance de la tecnología y la disponibilidad de grandes volúmenes de datos, se han desarrollado técnicas avanzadas de machine learning que mejoran la precisión de estas predicciones. Este artículo explora el uso del algoritmo XGBoost para desarrollar un modelo de mantenimiento predictivo efectivo, destacando su capacidad para manejar datos complejos y mejorar la eficiencia operativa en entornos industriales.

MANTENIMIENTO PREDICTIVO MEDIANTE ANÁLISIS DE DATOS Y MACHINE LEARNING USANDO XGBOOST

1. INTRODUCCIÓN

En la actualidad, la eficiencia y la continuidad operativa de las maquinarias y equipos son cruciales para el éxito de cualquier industria. El mantenimiento preventivo, una estrategia que implica la inspección y reparación regular de los equipos antes de que se produzcan fallas, juega un papel vital en la garantía de esta eficiencia. Esta metodología no solo ayuda a evitar costosos tiempos de inactividad no planificados, sino que también prolonga la vida útil de los equipos, mejora la seguridad operativa y optimiza el uso de los recursos.

Sin embargo, el mantenimiento preventivo tradicional puede no ser suficiente en un entorno industrial cada vez más complejo y exigente. Es aquí donde los sistemas de mantenimiento predictivo basados en análisis de datos y técnicas de machine learning entran en juego. Estos sistemas permiten una monitorización continua y en tiempo real de las condiciones operativas de los equipos, prediciendo fallas potenciales antes de que ocurran. Al integrar sensores avanzados, recopilación de datos y algoritmos predictivos, se pueden identificar patrones y anomalías que indican un posible deterioro del rendimiento de los equipos.

Según Rojas (2022), las empresas básicas enfrentaron múltiples accidentes debido a la falta de mantenimiento y de equipos de seguridad. Esto pasa en muchas empresas y genera bastante preocupación entre los trabajadores puesto que pueden ser el desencadenante de muchos accidentes laborales.

Según Merca2 (2024), el número de muertes por accidentes laborales en España aumentó un 22% en 2024 en comparación con el año anterior. La publicación menciona que son 3 las razones principales de los accidentes, estos son: infartos o da;os cerebrales, el cual es el de mayor detección, sin embargo el que ocupa el segundo lugar son los accidentes laborales como amputaciones o lesiones por atrapamientos, estos en su gran mayoría son por 2 razones: incumplimiento de protocolos de seguridad y falta de mantenimiento adecuado a las máquinas. Sin duda es un problema que pueden tener consecuencias en las vidas de las personas, y sabiendo la cantidad de máquinas que se procesan es difícil tener un control para cada una de las máquinas, por lo que un proceso que analice y procese de forma automatizada podría ayudar a reducir la probabilidad de accidentes y consecuencias que pueda tener el no mantenimiento de máquinas.

Implementar un sistema de mantenimiento predictivo ofrece múltiples ventajas. Primero, permite una planificación más precisa y proactiva del mantenimiento, reduciendo significativamente las interrupciones no planificadas y los costos asociados a reparaciones de emergencia. Segundo, mejora la eficiencia operativa al minimizar el tiempo de inactividad y optimizar el uso de recursos y personal. Tercero, incrementa la seguridad al identificar problemas potenciales antes de que se conviertan en fallos críticos que puedan poner en riesgo a los trabajadores y al entorno.

En este contexto, el presente trabajo se enfoca en el desarrollo e implementación de un sistema de mantenimiento predictivo mediante el análisis de datos y técnicas de machine learning.

2. TRABAJOS RELACIONADOS

En esta sección, revisamos como otros procesos de automatización y generación de algunos frameworks utilizando modelos de Machine Learning (ML) pueden brindar resultados aceptables para mantenimiento preventivo de máquinas y otros casos de uso similares:

2.1. Modelo de Machine Learning basado en mantenimiento predictivo para puentes que cruzan ríos. (Zhao, Tan, Zang, 2023)

El paper presentado nos explica cómo se utilizó un modelo de ML para el mantenimiento predictivo acorde a políticas de puentes que cruzan ríos o rutas de agua, para ello nos muestra que se tiene claro el motivo principal de las fallas es debido al desgaste del material, caso que en muchas ocasiones, el clima por el calentamiento global, ha ido incrementando el caudal de los ríos, generando mayores desgastes en las bases de los puentes. El paper nos muestra y explica la data inicial que se utilizó para el entrenamiento del modelo y las pruebas que se realizaron en un total de 40 puentes de Francia, los cuales dieron resultados positivos donde un ingeniero junior pudo validar que el algoritmo estaba siguiendo correctamente los resultados. Para el proyecto el artículo menciona el uso de 3 modelos a utilizar, siendo el de XG Boost el que brindó los mejores resultados con la misma data. La data estaba compuesta por datos de inspección de reportes de la entidad SNCF que destaca 2 componentes claves para detectar el riesgo en el análisis: evento de fluidos(lluvias, elevación de caudal, etc) y vulnerabilidad. De la data se seleccionaron 18 variables a considerar como entrada las cuales son de tipo categórico y numérico de 75 puentes de los últimos 5 años. Los 3 algoritmos que se utilizaron fueron: support vector machine (SVM) (Cortes & Vapnik, 1995), random forest (RF) (Leo Breiman, 2001) y extreme gradient boosting (XGBoost) (Chen & Guestrin, 2016). Los resultados fueron catalogados como buenos por los ingenieros junior del proyecto y siendo probado además en 40 puentes de Francia mostrados como resultado final del artículo.

2.2. Modelo de Machine Learning y Pipeline basado en mantenimiento predictivo de Impresoras 3D de Polímero, (Leite et al. 2023).:

Para el artículo indicado se nos brinda un trabajo que aplica un modelo de Machine Learning con regresión, en este caso se busca predecir el mantenimiento para impresoras 3D, enfocado específicamente en el proceso del FDM de Cetona de Poliestireno. El artículo menciona que el elemento que más se desgasta o que es la raíz de la mayor cantidad de fallas en caso no se de mantenimiento, es la boquilla de impresión, esta tiene un sistema el cual tiene indicadores y sensores en la impresora que van a generar data al momento de imprimir, como la fecha, el tiempo, etc.

Los datos requirieron en este caso de transformación de valores, que fueron emparejados por el ID de la impresora y el tipo de boquilla, una vez fueron limpiados y transformados los datos, se procedió a pasarlos por el modelo de regresión para detectar el mantenimiento de la impresora, o mejor dicho en la boquilla de la impresora. Para la fase de experimentación, se utilizó 114 boquillas donde 56 de ellas estaban dañadas, se ejecutaron aproximadamente 6000 trabajos con archivos e información del trabajo, también se

consideraron los logs, lo cual dio resultados favorables para la detección de mantenimiento preventivo. En este caso el artículo no realizó pruebas con otros modelos, utilizó directamente modelo de regresión y solo testeo con el proceso de impresión de FDM con cetona de poliestireno, no se utilizó otros materiales para ello.

2.3. Mantenimiento predictivo con MATLAB

Este ebook proporcionado por MathWorks basado en su producto MATLAB, nos guía a iniciarnos en el desarrollo de algoritmos de mantenimeinto predictivo, a través de ejemplos, tutoriales, terminología y acceso al software de prueba.

Detalla tres partes, Introducción al mantenimiento predictivo, donde menciona que se aprenderá como el mantenimiento predictivo difiere de las estrategias de mantenimiento reactivo y preventivo, permitiéndonos descubrir los flujos de trabajo de mantenimiento predictivo, como la adquisición y el preprocesamiento de datos, la extracción de características y el entrenamiento de modelos de Machine Learning. La segunda parte nos muestra como extraer las características más relevantes de los datos y el entrenamiento de modelos de Machine Learning a partir de dichas características para clasificar diferentes tipos de fallos. Y finalmente la tercera parte, cálculo de la vida útil restante con MATLAB, nos muestra diferentes modelos de estimación para estimar la vida útil restante (RUL) de su sistema, similitud, supervivencia y degradación, nos explica el modelo RUL adecuado para diferentes sistemas.

2.4. Sistema de mantenimiento continuo para una programación óptima basada en la monitorización de máquinas en tiempo real. (Costa, FJO 2018)

El resumen de esta disertación nos menciona que, en la actualidad, las actividades de mantenimiento son las que más llaman la atención de las empresas debido al aumento de los costes que suponen las paradas repentinas de las máquinas y, en consecuencia, la parada de los procesos productivos. Estas paradas se producen en su mayoría por desgastes de sus componentes que dan lugar a averías en las máquinas y es necesario realizar un seguimiento minucioso de los procesos de fabricación. En base a ello, y para aumentar la eficiencia de la línea de producción, se hace necesario monitorizar de forma continua el rendimiento de las máquinas y, junto con todos los datos históricos de mantenimiento, crear estrategias para minimizar las fases y los costes de mantenimiento. Estas estrategias pueden residir en la predicción de unos periodos de tiempo adecuados para realizar las operaciones de mantenimiento y, en base a ello, agrupar un conjunto de máquinas para realizar las actividades de mantenimiento entre turnos de día y de noche. Esto supone una dificultad principalmente por la creciente complejidad de la programación y planificación de las actividades de una línea de producción, siendo necesario minimizar el impacto de las actividades de mantenimiento basándose en la predicción de fallos en todo el plan ya existente.

3. MARCO TEÓRICO

3.1. Mantenimiento predictivo

El mantenimiento predictivo se basa en el monitoreo continuo de la condición de los equipos mediante la recopilación de datos de sensores, registros de mantenimiento y condiciones operativas. El objetivo es anticipar las fallas antes de que ocurran, lo que permite realizar intervenciones de mantenimiento en el momento más oportuno. utilizando datos en tiempo real y análisis avanzados. A diferencia del mantenimiento preventivo, que se basa en intervalos de tiempo fijos, el mantenimiento predictivo permite intervenciones más precisas, reduciendo los costos y el tiempo de inactividad (Jardine, Lin, & Banjevic, 2006).

3.1.1. Fundamentos del Mantenimiento Predictivo

El mantenimiento predictivo se basa en la recolección continua de datos de condición de los activos, como vibraciones, temperatura, ruido y datos operativos, que son analizados para identificar patrones o anomalías que preceden a un fallo (Mobley, 2002). Este enfoque se sustenta en la teoría de la fiabilidad y en técnicas de análisis de datos que permiten establecer correlaciones entre las condiciones de operación y la probabilidad de fallo.

Según la investigación de Heng et al. (2009), el mantenimiento predictivo ofrece una mejor gestión del ciclo de vida de los activos al utilizar modelos estadísticos y algoritmos de machine learning para predecir fallos. Esta técnica es particularmente eficaz en industrias donde el tiempo de inactividad no programado puede resultar en pérdidas significativas, como en el sector energético o manufacturero.

3.1.2. Técnicas de Machine Learning en Mantenimiento Predictivo

En los últimos años, las técnicas de machine learning han ganado popularidad en el campo del mantenimiento predictivo debido a su capacidad para manejar grandes volúmenes de datos y su precisión en la predicción de fallos (Zhao et al., 2017). Los modelos de aprendizaje automático, como las redes neuronales, los árboles de decisión y los modelos de boosting, permiten identificar patrones complejos que pueden pasar desapercibidos con métodos estadísticos tradicionales.

XGBoost, en particular, ha demostrado ser eficaz en el mantenimiento predictivo debido a su capacidad para manejar datos desbalanceados y su eficiencia computacional (Chen & Guestrin, 2016). XGBoost utiliza una técnica de boosting de gradiente, que optimiza los modelos mediante la combinación de varios árboles de decisión débiles para formar un modelo fuerte, lo que mejora significativamente la precisión de las predicciones.

3.1.3. Relevancia del Análisis de Datos

El análisis de datos es fundamental para el mantenimiento predictivo, ya que permite extraer información valiosa de grandes volúmenes de datos históricos y en tiempo real (Kumar et al., 2004). A través de técnicas de preprocesamiento, como la normalización y la reducción de dimensionalidad, los datos se preparan para ser utilizados en modelos de machine learning.

Investigaciones previas han demostrado que la calidad del análisis de datos es crucial para el éxito del mantenimiento predictivo. Por ejemplo, estudios como el de Lee et al. (2014)

destacan que un análisis adecuado de los datos de condición de los equipos puede mejorar significativamente la precisión de las predicciones de fallos y, por ende, optimizar la planificación del mantenimiento.

3.1.4. Comparación con Otros Enfoques de Mantenimiento

El mantenimiento predictivo se diferencia del mantenimiento preventivo y correctivo en su enfoque basado en datos. Mientras que el mantenimiento preventivo se realiza en intervalos regulares basados en la experiencia o recomendaciones del fabricante, el mantenimiento predictivo utiliza datos en tiempo real para determinar el momento exacto en que debe realizarse la intervención (Tsang, 2002).

Los estudios de Jardine et al. (2006) y Mobley (2002) muestran que, aunque el mantenimiento preventivo puede ser efectivo en ciertos contextos, no es tan eficiente como el mantenimiento predictivo en términos de reducción de costos y maximización del tiempo de funcionamiento del equipo.

3.1.5. Beneficios y Desafíos del Mantenimiento Predictivo

El mantenimiento predictivo ofrece numerosos beneficios, incluyendo la reducción de costos de mantenimiento, la minimización del tiempo de inactividad y la prolongación de la vida útil de los equipos (Jardine, Lin, & Banjevic, 2006). Sin embargo, también presenta desafíos significativos, como la necesidad de infraestructura tecnológica avanzada y la gestión de grandes volúmenes de datos (Zhao et al., 2017).

Un desafío clave es la precisión de las predicciones. Aunque los modelos de machine learning, como XGBoost, han mejorado considerablemente la capacidad de predecir fallos, la efectividad del mantenimiento predictivo depende en gran medida de la calidad y cantidad de datos disponibles, así como de la capacidad para interpretar correctamente los resultados del modelo (Heng et al., 2009).

3.2. Análisis de Datos

El análisis de datos es un proceso fundamental en la toma de decisiones basado en datos, que implica la inspección, limpieza, transformación y modelado de datos con el objetivo de descubrir información útil, llegar a conclusiones y apoyar la toma de decisiones (Provost & Fawcett, 2013). Este proceso es esencial en disciplinas como la estadística, la informática y la investigación científica, donde los datos se utilizan para inferir patrones y tendencias que no son evidentes a simple vista.

3.2.1. Componentes del Análisis de Datos

El análisis de datos generalmente se descompone en varias etapas clave:

- **Recolección de datos**: Esta etapa implica la obtención de datos de diversas fuentes, que pueden incluir encuestas, bases de datos, sensores, y transacciones. La calidad de los datos recolectados es crucial, ya que influye directamente en la fiabilidad de los análisis posteriores (Chen, Chiang, & Storey, 2012).
- **Limpieza de datos**: En esta etapa, se identifican y corrigen los errores y valores atípicos en los datos. El proceso de limpieza de datos incluye la eliminación de

duplicados, la corrección de errores de entrada, y el manejo de valores faltantes. La calidad de los datos limpios es vital para evitar sesgos en los análisis (Dasu & Johnson, 2003).

- **Transformación de datos**: Una vez que los datos están limpios, es posible que necesiten ser transformados o normalizados para que sean adecuados para el análisis. Esto puede incluir la normalización, la agregación, la discretización o la transformación de variables categóricas en numéricas (Kantardzic, 2011).
- **Modelado de datos**: En esta fase, se aplican métodos estadísticos o algoritmos de machine learning para modelar los datos y extraer patrones. Este paso puede incluir técnicas de regresión, clasificación, clustering o reducción de dimensionalidad, dependiendo del objetivo del análisis (Hastie, Tibshirani, & Friedman, 2009).
- **Interpretación y visualización**: Finalmente, los resultados del análisis se interpretan y se presentan de manera que sean comprensibles para los tomadores de decisiones. La visualización de datos es una herramienta crucial en esta etapa, ya que permite a los usuarios comprender rápidamente patrones complejos en los datos (Few, 2006).

3.2.2. Técnicas de Análisis de Datos

El análisis de datos puede implicar una amplia gama de técnicas, dependiendo del tipo de datos y los objetivos del análisis:

- **Análisis descriptivo**: Se utiliza para describir las características básicas de los datos. Incluye medidas de tendencia central como la media, mediana y moda, así como medidas de dispersión como la desviación estándar y el rango (Hair et al., 2010).
- **Análisis exploratorio de datos (EDA)**: Esta técnica se utiliza para descubrir patrones, detectar anomalías y verificar suposiciones mediante el uso de gráficos y estadísticas resumidas. Es un enfoque esencialmente inductivo que ayuda a los analistas a entender mejor los datos antes de aplicar modelos predictivos (Tukey, 1977).
- **Análisis predictivo**: Utiliza modelos estadísticos y de machine learning para hacer predicciones sobre datos futuros o no observados. Los algoritmos como la regresión lineal, los árboles de decisión y los modelos de boosting como XGBoost se utilizan comúnmente en este tipo de análisis (Hastie, Tibshirani, & Friedman, 2009).
- **Análisis prescriptivo**: Va un paso más allá del análisis predictivo al sugerir posibles acciones basadas en los resultados del análisis. Involucra la optimización y la simulación para encontrar la mejor solución a un problema dado (Shmueli et al., 2010).

3.2.3. Importancia del Análisis de Datos en la Toma de Decisiones

El análisis de datos proporciona una base sólida para la toma de decisiones en diversos campos, incluyendo negocios, ingeniería, ciencias de la salud, y más. Al transformar datos en información útil, el análisis de datos permite a las organizaciones optimizar operaciones, mejorar la eficiencia y anticipar tendencias futuras (Davenport & Harris, 2007).

Por ejemplo, en la industria, el análisis de datos se utiliza para el mantenimiento predictivo, donde los datos de sensores se analizan para predecir fallos en equipos antes de que ocurran. En la salud, los datos de pacientes se analizan para predecir enfermedades y mejorar el cuidado del paciente (Provost & Fawcett, 2013).

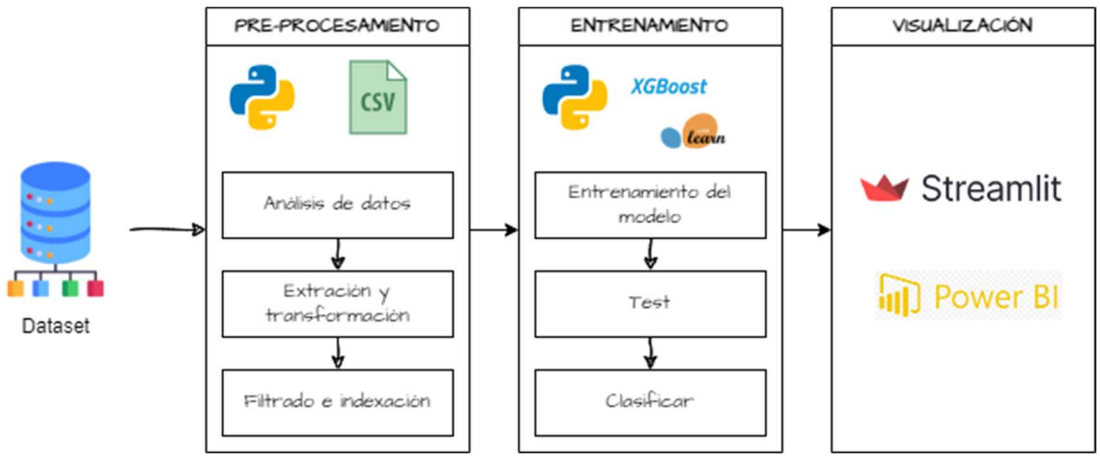
3.2.4. Desafíos en el Análisis de Datos

A pesar de sus beneficios, el análisis de datos también presenta desafíos significativos. Entre ellos se incluyen:

- **Manejo de grandes volúmenes de datos**: Con el aumento de la generación de datos, manejar y procesar grandes volúmenes de datos se ha convertido en un desafío. El *Big Data* ha llevado a la necesidad de nuevas tecnologías y enfoques para gestionar y analizar datos a gran escala (Chen, Chiang, & Storey, 2012).
- **Calidad de los datos**: La calidad de los datos sigue siendo un desafío crítico. Datos incompletos, inexactos o desactualizados pueden llevar a conclusiones erróneas, lo que resalta la importancia de las etapas de limpieza y preprocesamiento (Dasu & Johnson, 2003).

4. VISIÓN GENERAL DEL SISTEMA

El propósito de desarrollar este sistema, es generar un dashboard que muestra las predicciones correctamente de nuevas máquinas con patrones similares a los que el modelo de ML puede detectar, principalmente se han definido x clasificaciones dependiendo lo indicado por el modelo:



Se busca que este modelo pueda tener un resultado visual ya sea en Power Bi, StreamLit o con Ploty mediante el modelo entrenado exportado, podemos utilizarlo para que brinde el resultado. El lenguaje principal será Python ya que es muy destacado en el campo de la Inteligencia Artificial y el análisis de datos, actualmente se siguen creando librerías que mejoran y facilitan el trabajo de modelos de ML con este lenguaje.

Zhu, Wang, y Jin (2024) exploraron un enfoque de control de procesos adaptativo en la manufactura aditiva utilizando aprendizaje profundo por refuerzo. Este fue implementado principalmente para Python con open access para todos los desarrolladores, y así como esta

librería, existen muchas otras. Entre ellas tenemos algunas de las mas conocidas que también se utilizaron para esta implementación como numpy, sns, plot y las librerías de modelos con scikit-learn.

Para el modelo de entrenamiento, se opto por utilizar el modelo de XG Boost, a continuación se brinda más detalles del modelo.

4.1. XGBoost.

XGBoost, abreviatura de *eXtreme Gradient Boosting*, es un algoritmo de machine learning basado en árboles de decisión que utiliza la técnica de boosting para mejorar la precisión de las predicciones. Fue desarrollado por Tianqi Chen en 2016 y rápidamente se convirtió en una herramienta popular en competencias de ciencia de datos debido a su eficiencia y rendimiento superior en comparación con otros algoritmos (Chen & Guestrin, 2016).

4.1.1. Fundamentos del Boosting

El boosting es una técnica de ensamble que combina varios modelos débiles, generalmente árboles de decisión, para crear un modelo más fuerte. Cada árbol se construye de manera secuencial, y cada uno corrige los errores cometidos por el árbol anterior. El objetivo es minimizar una función de pérdida que mide la diferencia entre las predicciones del modelo y los valores reales (Friedman, 2001).

XGBoost optimiza este proceso utilizando un enfoque de boosting de gradiente, en el cual los árboles de decisión se ajustan para corregir los errores residuales de manera iterativa. Este método es particularmente eficaz para problemas de clasificación y regresión, donde la precisión del modelo es crucial.

Los **árboles potenciados por gradientes** (gradient boosted trees) existen desde hace tiempo y hay mucho material sobre el tema. Explicaremos el tema de una manera sencilla utilizando el articulo original y utilizando los principios del aprendizaje supervisado.

XGBoost es utilizado para problemas de aprendizaje supervisado, donde utilizamos los datos de entrenamiento (con múltiples funciones) $\boldsymbol{x_i}$ para predecir una variable objetivo $\boldsymbol{y_i}$. Antes de aprender sobre los árboles específicamente, comencemos repasando los elementos básicos del aprendizaje supervisado. El **modelo** se refiere a la estructura matemática mediante la cual se realiza la predicción. La salida $\boldsymbol{y_i}$ se calcula a partir de la entrada $\boldsymbol{x_i}$. El valor de predicción tiene diferentes interpretaciones como regresión o como clasificación. Los **parámetros ($\boldsymbol{\theta}$)** son la parte desconocida que necesitamos aprender de los datos. En los problemas de regresión lineal, los parámetros son los coeficientes.

Con elecciones juiciosas para $\boldsymbol{y_i}$, podemos expresar una variedad de tareas, como regresión, clasificación y ranking. La tarea de entrenar el modelo consiste en encontrar los mejores parámetros $\boldsymbol{\theta}$, que mejor se ajusten a los datos de entrenamiento de entrada $\boldsymbol{x_i}$ y etiquetas $\boldsymbol{y_i}$. Para entrenar el modelo, necesitamos definir la **función objetivo** $obj(\theta)$. para medir qué tan bien se ajusta el modelo a los datos de entrenamiento. Una característica destacada de las

funciones objetivo es que constan de dos partes: **función pérdida de entrenamiento** $\mathbf{L}(\boldsymbol{\theta})$ mide cuán predictivo es nuestro modelo con respecto a los datos de entrenamiento (Chen & Guestrin, 2016), y término de **regularización** $\Omega(\boldsymbol{\theta})$ controla la complejidad del modelo, lo que nos ayuda a evitar el sobreajuste (Tibshirani, 1996).

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

Esto suena un poco abstracto, así que pondremos el ejemplo que se pone en el artículo original.

Consideremos el problema en la Imagen 1. Problema de ajuste. Se le pide que ajuste visualmente una función escalonada dados los puntos de datos de entrada en la esquina superior izquierda de la imagen. ¿Cuál de las tres soluciones cree que se ajusta mejor?

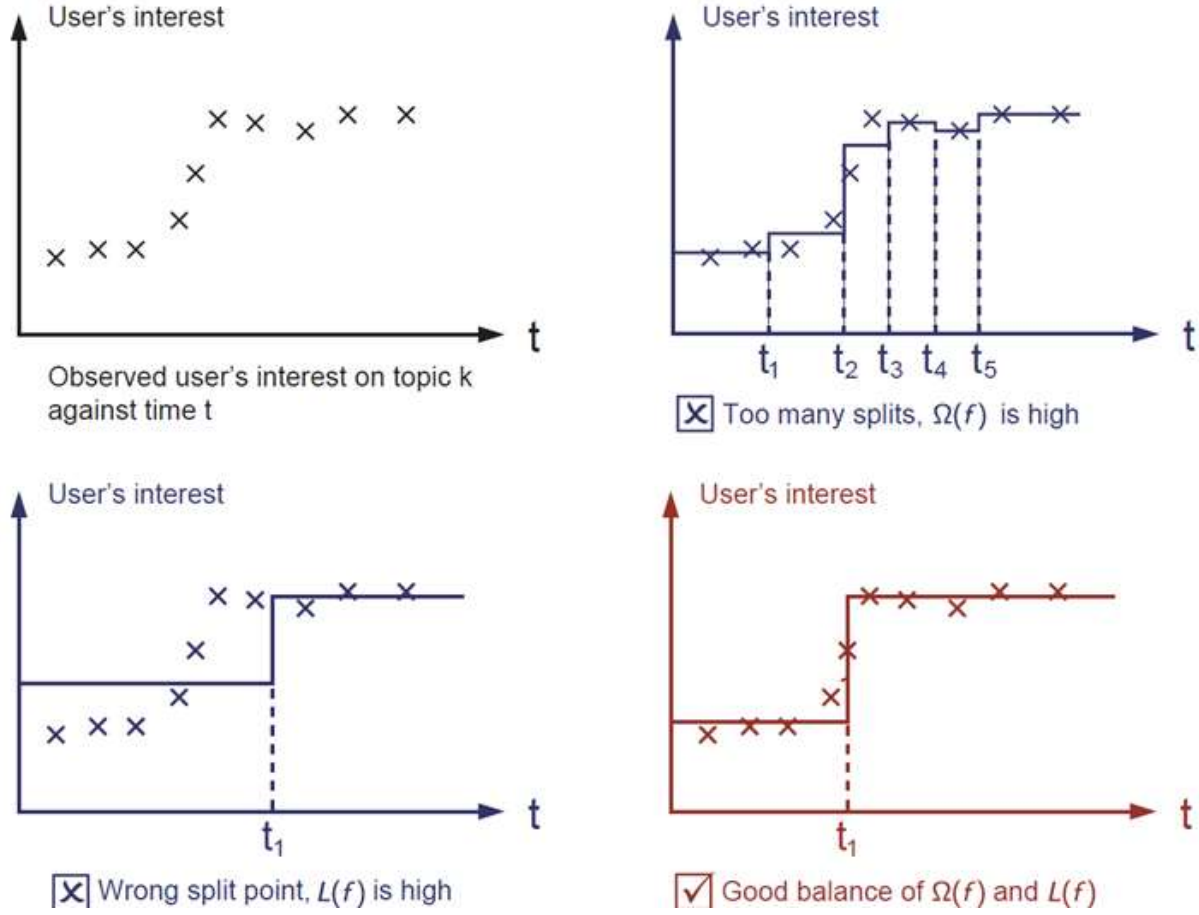


Imagen 1. Problema de ajuste

La respuesta correcta está marcada en rojo. El principio general es que queremos un modelo simple y predictivo. El equilibrio entre los dos también se conoce como equilibrio entre sesgo y varianza en el aprendizaje automático.

El modelo de conjunto de árboles consta de un conjunto de árboles de clasificación y regresión (CART). Aquí hay un ejemplo simple de un CART que clasifica si a alguien le gustará un hipotético juego de computadora X que mostramos en la Imagen 2.

Clasificamos a los miembros de una familia en diferentes hojas y les asignamos la puntuación correspondiente. Un CART es un poco diferente de los árboles de decisión, en los que la hoja

solo contiene valores de decisión. En CART, se asocia una puntuación real a cada una de las hojas, lo que nos brinda interpretaciones más ricas que van más allá de la clasificación. Esto también permite un enfoque unificado y basado en principios para la optimización.

Por lo general, un solo árbol no es lo suficientemente sólido como para ser utilizado en la práctica. Lo que se utiliza en realidad es el modelo de conjunto, que suma las predicciones de varios árboles.

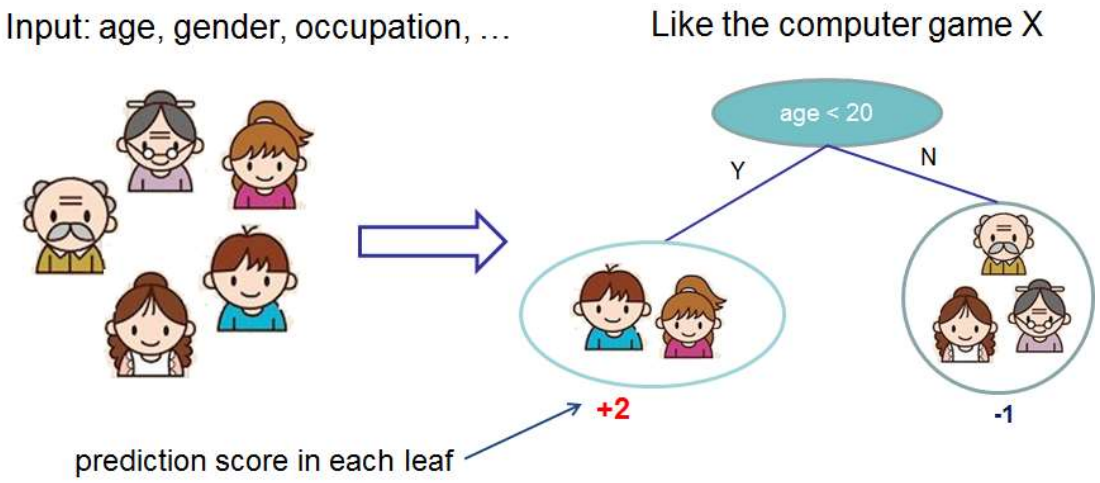


Imagen 2. Predicción con un árbol.

A continuación en la **¡Error! No se encuentra el origen de la referencia.**. Las puntuaciones de predicción de cada árbol individual se suman para obtener la puntuación final. Si observa el ejemplo, un hecho importante es que los dos árboles intentan complementarse entre sí.

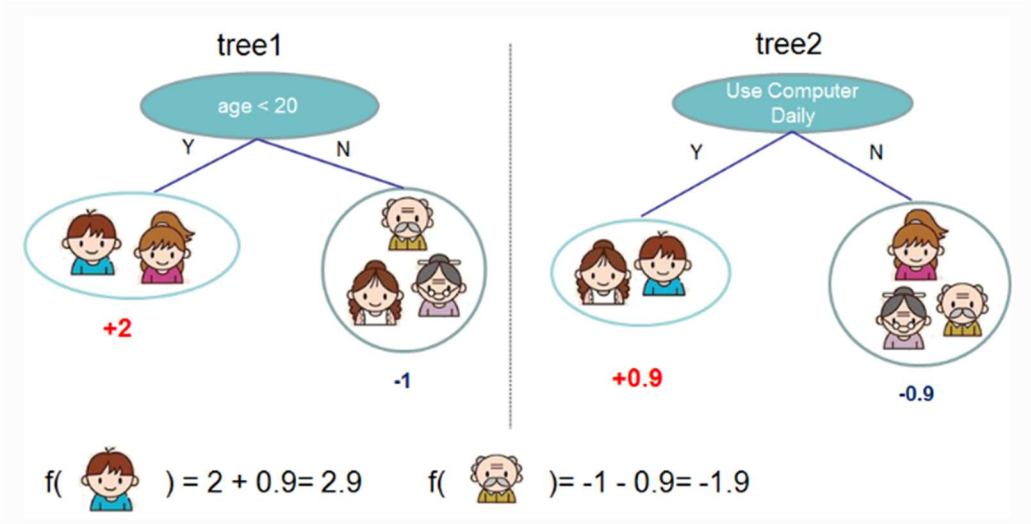


Imagen 3 Predicción de un conjunto de dos árboles

Matemáticamente, podemos escribir nuestro modelo en la forma

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)\,, f_k \in F$$

Dónde *K* es el número de árboles, f_k es una función en el espacio funcional F, el cual es el conjunto de todos los CART posibles.

Por lo tanto, los bosques aleatorios y los árboles potenciados son realmente los mismos modelos; la diferencia surge de cómo los entrenamos. Esto significa que, si escribe un servicio predictivo para conjuntos de árboles, solo necesita escribir uno y debería funcionar tanto para bosques aleatorios como para árboles potenciados por gradiente.

El entrenamiento se inicia definiendo una función objetivo y optimizarla.

Puedes descubrir que lo que necesitamos aprender son esas funciones f_i, cada una de las cuales contiene la estructura del árbol y las puntuaciones de las hojas. Aprender la estructura del árbol es mucho más difícil que el problema de optimización tradicional, en el que simplemente se puede tomar el gradiente. Es imposible aprender todos los árboles a la vez. En cambio, utilizamos una estrategia aditiva, arreglamos lo que hemos aprendido y añadimos un árbol nuevo a la vez, Escribimos el valor de predicción en el paso ***t*** como $\hat{y}_i^t$.

Entonces tenemos:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

Y usaremos el árbol que optimice nuestra función objetivo, usando el error cuadrático medio

$$\text{obj}^t = \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t) + \text{ constant}$$

En la anterior fórmula se tomo la expansión de Taylor de la función de perdida.

Donde $\boldsymbol{g_i}$ y $\boldsymbol{h_i}$ se definen como

$$\begin{aligned}
g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \\
h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})
\end{aligned}$$

Eliminando todas las constantes la función objetivo en el paso ***t*** se convierte en

$$\sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \omega(f_t)$$

A partir de aquí hallaremos **el término de regularización**, para lo cual necesitamos definir la complejidad del árbol $\boldsymbol{\omega(f)}$ para lo cual definimos el árbol ***f(x)***

$$f_t(x) = \omega_{q(x)}, \omega \in R^T, q: R^d \rightarrow \{1,2,..,T\}$$

ω es el vector de puntuaciones en las hojas, ***q*** es una función que asigna cada punto de datos a la hoja correspondiente y ***T*** es el número total de hojas

$$\omega(f) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2$$

La fórmula anterior es la definición de complejidad del XGBoost, aunque hay muchas formas de definirla.

Después de reformular el modelo del árbol, la función objetivo en el ***t-ésimo*** árbol queda como

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j}g_i)w_j + \frac{1}{2}(\sum_{i\in I_j}h_i + \lambda)w_j^2] + \gamma T\end{aligned}$$

Donde $I_j = \{i|\ q(x_i) = j\}$, es el conjunto de índices de puntos de datos asignados a la ***j-ésima*** hoja, comprimiendo $G_{ji} = \sum_{i\in Ij} g_i$ y $H_j = \sum_{i\in Ij} h_i$

$$\text{obj}^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T$$

En esta ecuación ***ωj*** son independientes entre si, la forma $G_{j\omega j} + \frac{1}{2}(H_j + \lambda)\omega_j^2$, es cuadratico y el mejor ***ωj*** para una estructura dada q(x) y la mejor reducción objetiva que podemos obtener es:

$$\begin{aligned}w_j^* &= -\frac{G_j}{H_j + \lambda} \\ \text{obj}^* &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T\end{aligned}$$

La última ecuación mide que tan buena es la estructura de un árbol q(x).

Si todo esto suena un poco complicado, observemos la Imagen 4. Puntuación o score del árbol. y veamos cómo se pueden calcular los puntajes. Básicamente, para una estructura de árbol dada, enviamos las estadísticas g_i y h_i, a las hojas a las que pertenecen, sume las estadísticas y use la fórmula para calcular qué tan bueno es el árbol. Esta puntuación es como la medida de impureza en un árbol de decisión, excepto que también tiene en cuenta la complejidad del modelo.

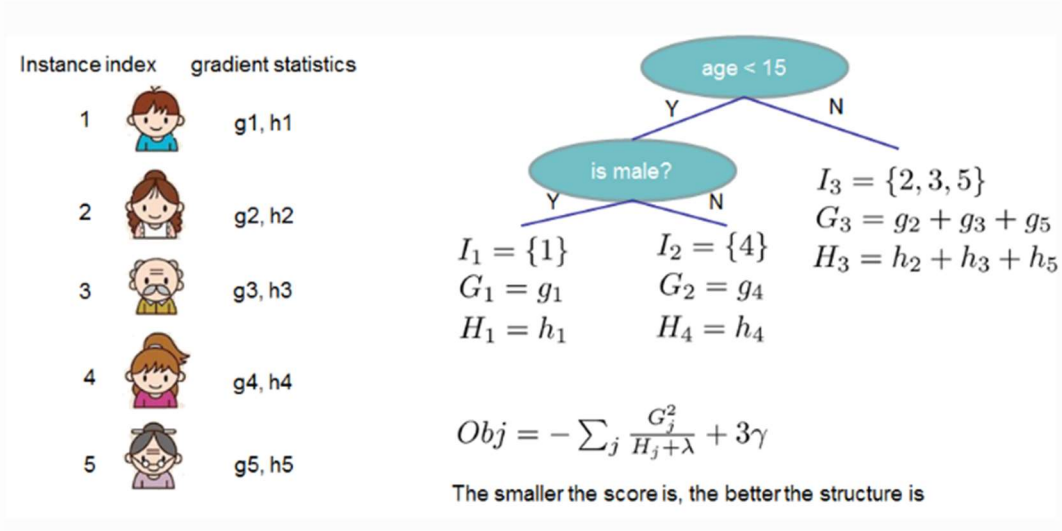


Imagen 4. Puntuación o score del árbol.

Ahora que tenemos una forma de medir la calidad de un árbol, lo ideal sería enumerar todos los árboles posibles y elegir el mejor. En la práctica, esto es inviable, por lo que intentaremos optimizar un nivel del árbol a la vez. En concreto, intentaremos dividir una hoja en dos hojas y la puntuación que obtenga será:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

Esta ecuación la podemos descomponer en,

- La puntuación en la nueva hoja izquierda.
- La puntuación en la nueva hoja derecha.
- La puntuación en la hoja original.
- La regulación en la hoja adicional.

Podemos observar que, si la ganancia es menor que γ, sería mejor no agregar esa rama, a esto llamaremos poda en los modelos basados en árboles

En el caso de datos con valores reales, normalmente queremos buscar una división óptima. Para hacerlo de forma eficiente, colocamos todas las instancias en orden, como en la siguiente Imagen 5: Split de ganancias. Un escaneo de izquierda a derecha es suficiente para calcular la puntuación de estructura de todas las posibles soluciones de división, y podemos encontrar la mejor división de manera eficiente.

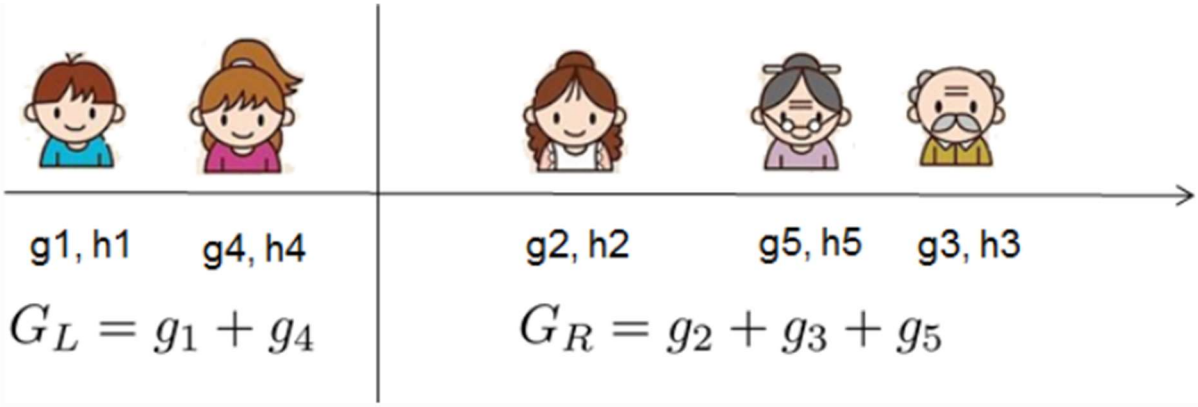


Imagen 5: Split de ganancias.

Componentes Clave de XGBoost

- Función de pérdida: XGBoost minimiza una función de pérdida que puede ser personalizada según el problema específico. En clasificación binaria, comúnmente se utiliza la entropía cruzada, mientras que en problemas de regresión se emplea el error cuadrático medio (Chen & Guestrin, 2016).
- Regularización: XGBoost incorpora términos de regularización L1 (Lasso) y L2 (Ridge) en su función de objetivo, lo que ayuda a prevenir el sobreajuste al penalizar la complejidad del modelo. Esto hace que el modelo sea más robusto al trabajar con datos ruidosos o sobreajustados (Tibshirani, 1996).
- Paralelización: Una de las características distintivas de XGBoost es su capacidad para realizar cálculos en paralelo, lo que acelera significativamente el proceso de entrenamiento. A diferencia de otros algoritmos de boosting que construyen árboles de manera secuencial, XGBoost optimiza el tiempo de ejecución utilizando técnicas de paralelización durante el cálculo de las mejores divisiones en los árboles de decisión (Chen & Guestrin, 2016).
- Manejo de datos faltantes: XGBoost trata los datos faltantes de manera eficiente, estimando de manera automática el mejor camino en el árbol para los valores faltantes durante el proceso de construcción de los árboles (Chen & Guestrin, 2016). Esto evita la necesidad de imputar valores faltantes antes de entrenar el modelo.

4.1.2. Ventajas de XGBoost

XGBoost ofrece varias ventajas que lo hacen destacar en comparación con otros algoritmos de machine learning:

- Eficiencia computacional: XGBoost está diseñado para ser extremadamente rápido y eficiente en el uso de recursos computacionales. La implementación del algoritmo en C++ y su capacidad para realizar cálculos en paralelo lo hacen ideal para aplicaciones que manejan grandes volúmenes de datos (Chen & Guestrin, 2016).
- Precisión en la predicción: Gracias a su capacidad para manejar términos de regularización y corregir errores iterativos de manera eficiente, XGBoost logra una alta precisión en una amplia variedad de tareas predictivas (Chen & Guestrin, 2016).
- Flexibilidad: XGBoost permite personalizar tanto la función de pérdida como los parámetros del modelo, lo que le otorga flexibilidad para adaptarse a diferentes tipos de

problemas de machine learning, incluyendo clasificación, regresión y ranking (Chen & Guestrin, 2016).

4.1.3. Aplicaciones de XGBoost

XGBoost se ha utilizado en una amplia gama de aplicaciones, desde la detección de fraudes en transacciones financieras hasta la predicción de enfermedades en el ámbito de la salud (Nielsen, 2016). En particular, ha sido utilizado con éxito en el mantenimiento predictivo para predecir fallos en equipos industriales, donde su capacidad para manejar datos desbalanceados y su alta precisión lo convierten en una herramienta valiosa (Zhao et al., 2017).

En competencias de ciencia de datos como Kaggle, XGBoost ha sido el algoritmo preferido, por muchos equipos debido a su combinación de precisión y velocidad. Su capacidad para manejar conjuntos de datos grandes y complejos lo hace particularmente adecuado para problemas que requieren predicciones precisas en tiempo real.

4.1.4. Limitaciones de XGBoost

A pesar de sus numerosas ventajas, XGBoost no está exento de limitaciones:

- **Complejidad del modelo:** Dado que XGBoost puede generar modelos muy complejos, existe el riesgo de sobreajuste si no se ajustan adecuadamente los parámetros de regularización (Chen & Guestrin, 2016).
- **Requiere ajuste de hiperparámetros**: Para obtener el mejor rendimiento de XGBoost, es necesario un ajuste cuidadoso de los hiperparámetros, lo que puede requerir una gran cantidad de tiempo y recursos computacionales (Nielsen, 2016).
- **Limitaciones en interpretabilidad:** Al ser un modelo de ensemble que combina múltiples árboles, XGBoost puede ser difícil de interpretar en comparación con modelos más simples como árboles de decisión individuales o regresiones lineales (Lundberg & Lee, 2017).

4.1. DATASET

El dataset seleccionado para este trabajo es brindado por la página Kaggle el cual trata de una colección de diferentes archivos .CSV que constan de lo siguiente:

MACHINE.CSV:

Este dataset contiene 47 máquinas donde tiene un identificador único para las maquinas, un modelo especifico que puede repetirse en el dataset y la edad de la máquina que están en años.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   machineID  47 non-null     int64
 1   model      47 non-null     object
 2   age        47 non-null     int64
dtypes: int64(2), object(1)

memory usage: 1.2+ KB
```

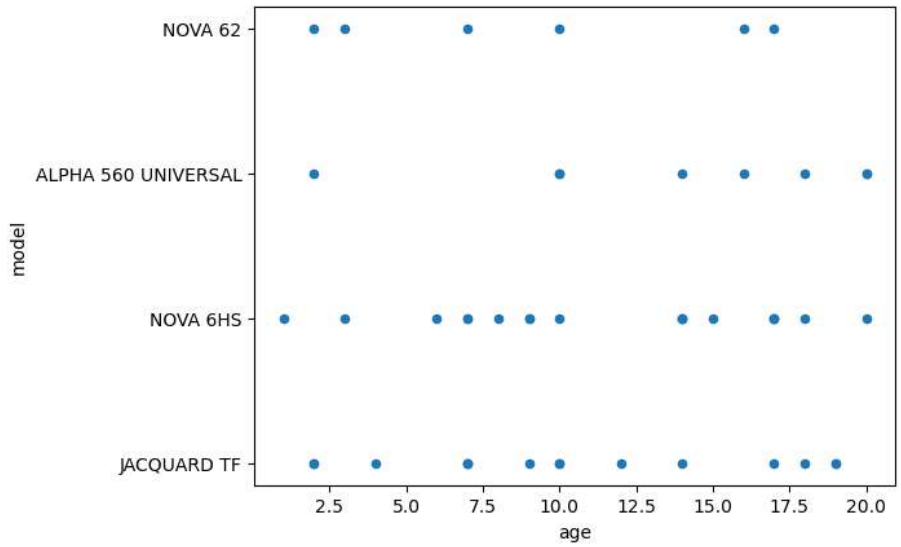


Imagen 6: Tabla Machines, Modelo VS. Antigüedad.

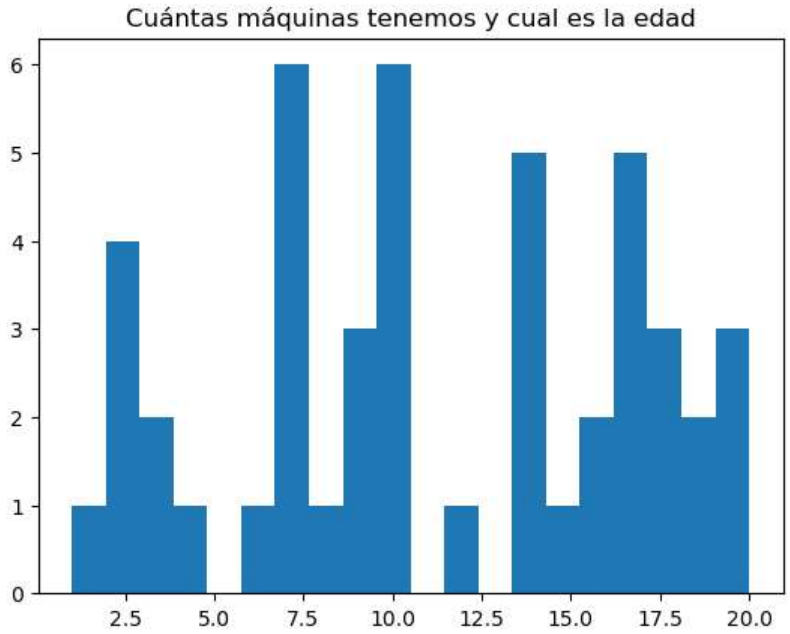


Imagen 7: Cantidad de máquinas VS. Antigüedad.

MANTTO.CSV:

Este dataset contiene el registro de los mantenimientos realizados a las maquinas, aquí podemos encontrar la fecha (el día) en el que se realizó dicho mantenimiento, el id de la maquina y el componente al cual se le aplico el mantenimiento.

Se tiene un total de 1557 registros de mantenimiento del 2020 al 2021.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1557 entries, 0 to 1556
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
```

```
---  ------     --------------  -----
 0   datetime   1557 non-null   datetime64[ns]
 1   machineID  1557 non-null   int64
 2   comp       1557 non-null   object
dtypes: datetime64[ns](1), int64(1), object(1)

memory usage: 36.6+ KB
```

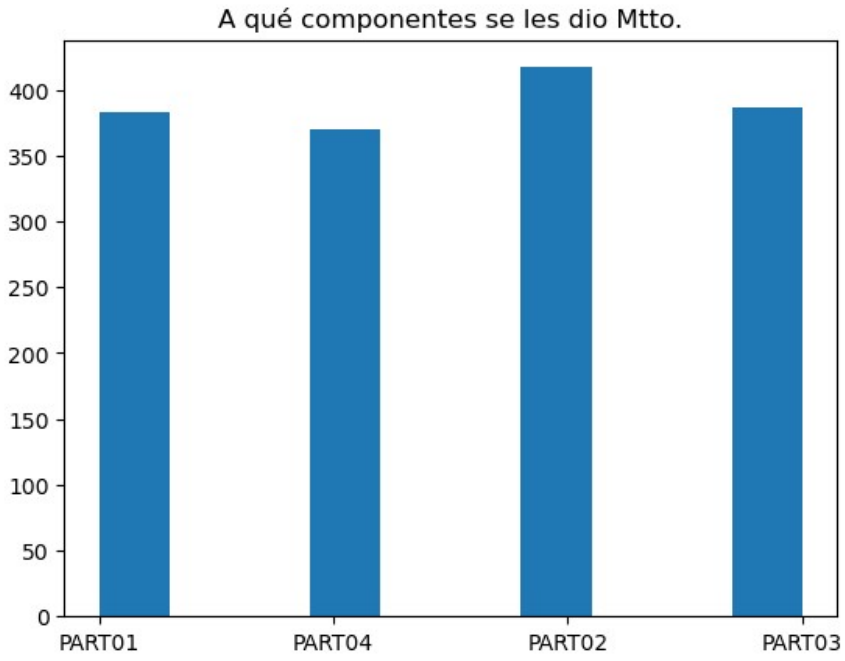


Imagen 8: Veces es las que una parte fue notificada con falla.

FAILURE.CSV:

Este registro contiene 354 filas donde menciona la fecha, el id de la maquina y que componente fallo.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 354 entries, 0 to 353
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   datetime   354 non-null    datetime64[ns]
 1   machineID  354 non-null    int64
 2   failure    354 non-null    object
dtypes: datetime64[ns](1), int64(1), object(1)

memory usage: 8.4+ KB
```

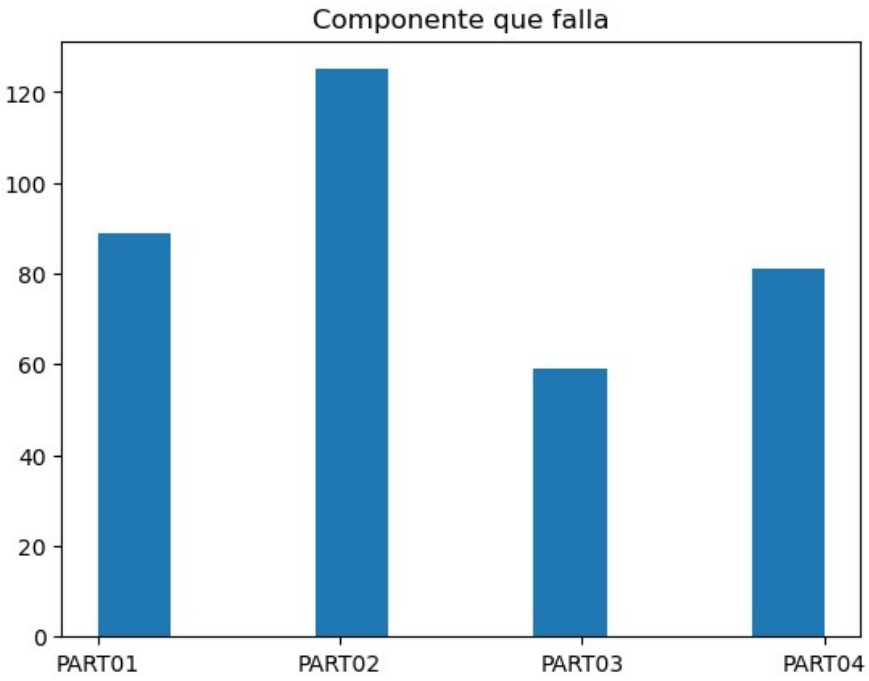


Imagen 9: Componentes con fallas.

ERROR.CSV:

Aquí podemos encontrar lo más grave que puede ocurrirle a la máquina, puesto que este dataset indica que maquina llego a fallar y en que día fallo, adicionalmente indica un error Id para tener un seguimiento sobre el error mapeado o detectado de la máquina, consta de 1830 registros.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1830 entries, 0 to 1829
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   datetime   1830 non-null   datetime64[ns]
 1   machineID  1830 non-null   int64
 2   errorID    1830 non-null   object
dtypes: datetime64[ns](1), int64(1), object(1)

memory usage: 43.0+ KB
```

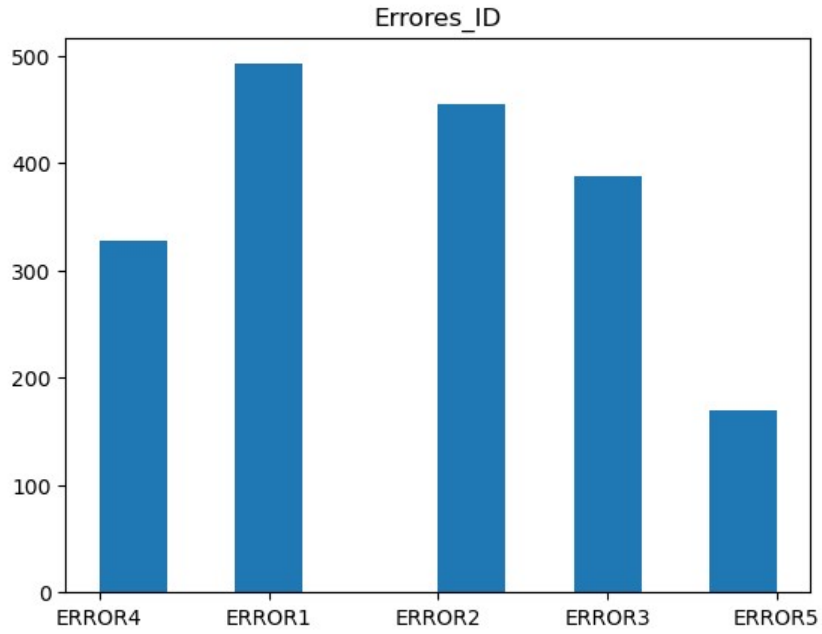


Imagen 10: Cantidad de fallas.

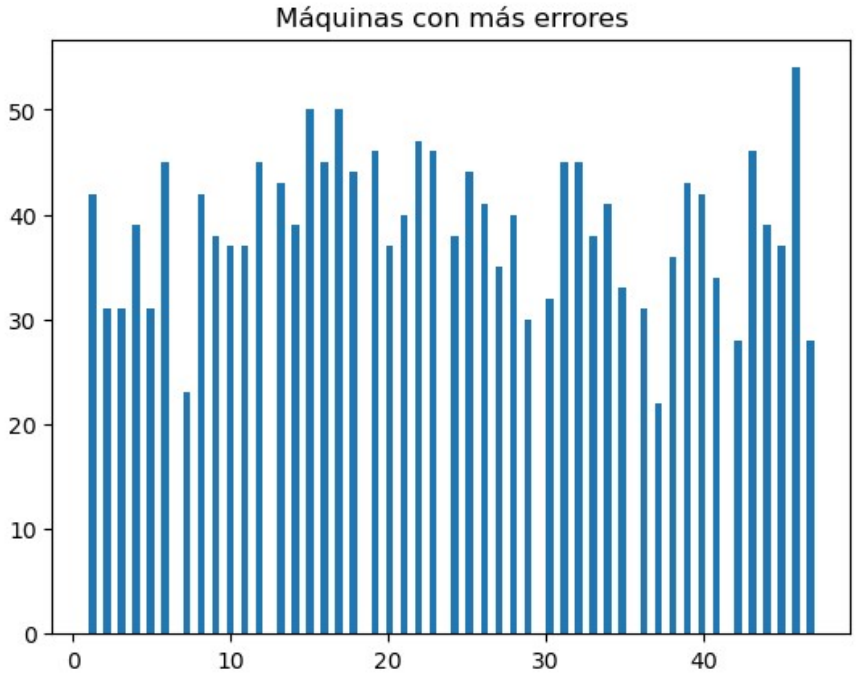


Imagen 11: Errores por máquina.

SENSOR.CSV:

Este se podría considerar uno de los datasets más completos de la base de datos puesto que contiene un registro inmenso de los datos telemétricos del equipo, como son el voltage, la rotacion, la presion y vibracion de las maquinas, lo cual son datos que ayudarían a identificar

posibles fallas, además de ello contiene el id de la maquina y la fecha y hora de los datos obtenidos, tiene datos desde el 2021 al 2022 siendo un total de 464335 registros.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 464333 entries, 0 to 464332
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   datetime   464333 non-null  datetime64[ns]
 1   machineID  464333 non-null  int64
 2   volt       464333 non-null  float64
 3   rotate     464333 non-null  float64
 4   pressure   464333 non-null  float64
 5   vibration  464333 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(1)

memory usage: 21.3 MB
```

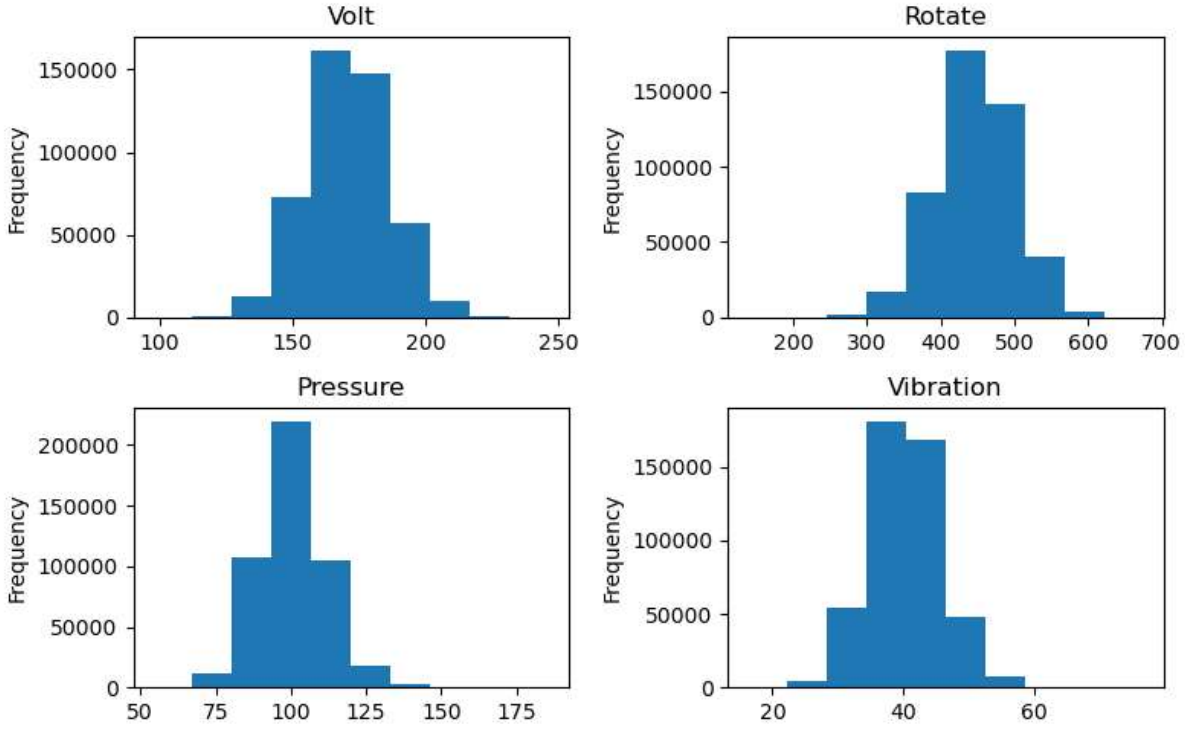


Imagen 12: Histograma de Sensor.

4.2. ANALISIS DE TAREAS

La planificación de la aplicación consta de diferentes etapas las cuales serán consideradas en orden para el correcto desarrollo, además de brindar la importancia adecuada a cada punto, es necesario indicar que muchos de los procesos de desarrollos de modelos predictivos son similares en cuanto a tareas, variando únicamente la complejidad del dataset y el modelo elegido.

T1 Identificación de componentes críticos: Para implementar un sistema de mantenimiento predictivo, es esencial identificar los componentes del equipo que son más críticos para las operaciones. El análisis de modos y efectos de fallos (FMEA) se utiliza para determinar qué componentes deben ser monitoreados de cerca, ya que su fallo podría causar interrupciones significativas en la producción.

T2 Recolección y selección de datos: Los datos para el análisis provienen de múltiples fuentes, como sensores en tiempo real, registros de mantenimiento anteriores y datos operativos. Es crucial garantizar que estos datos sean precisos y representen adecuadamente las condiciones de los equipos. El preprocesamiento de datos, que incluye la limpieza y transformación, es un paso esencial para asegurar que el modelo XGBoost funcione de manera óptima.

T3 Preprocesamiento de datos: Antes de entrenar el modelo, los datos deben ser preparados mediante técnicas como la eliminación de valores atípicos, el tratamiento de datos faltantes y la normalización. Además, se puede aplicar la reducción de dimensionalidad, como el Análisis de Componentes Principales (PCA), para simplificar el conjunto de datos sin perder información relevante.

T4 Identificación de patrones de fallo: El análisis de los datos históricos permite identificar patrones que preceden a los fallos de los equipos. Estos patrones se utilizan para entrenar el modelo de XGBoost, que luego se valida y ajusta para mejorar su precisión en la predicción de fallos futuros.

5. PROPUESTA ALGORITMO (BACKEND + VISUALIZACIÓN (FRONTEND)

Algoritmos (Backend):

- **Selección del algoritmo**: XGBoost se elige por su capacidad para manejar grandes volúmenes de datos y por su eficacia en la predicción de fallos con alta precisión. A diferencia de otros algoritmos, XGBoost maneja eficazmente los datos desbalanceados y ofrece un rendimiento computacional superior.
- **Entrenamiento y validación del modelo**: El modelo se entrena utilizando datos históricos y se valida con un conjunto separado de datos para evaluar su rendimiento. Se utilizan técnicas como la validación cruzada y el ajuste de hiperparámetros para optimizar el modelo, asegurando que sea preciso y eficiente.

Visualización (Frontend):

- **Herramientas de visualización**: La visualización de los resultados de las predicciones es crucial para que los ingenieros y técnicos tomen decisiones informadas. Se propone el uso de herramientas como D3.js y Plotly para crear dashboards interactivos que muestran las predicciones de fallos y el estado de los equipos en tiempo real.

- **Diseño de dashboards**: Los dashboards deben ser intuitivos y accesibles, permitiendo a los usuarios visualizar rápidamente las tendencias, recibir alertas sobre fallos inminentes y monitorizar la salud del equipo de manera eficiente.

6. DISEÑO VISUAL

Interfaz de usuario (UI): Se diseña una interfaz de usuario que presente la información de manera clara y fácil de interpretar. Esta interfaz incluye paneles de control, gráficos interactivos y opciones de personalización para diferentes usuarios, como ingenieros de mantenimiento y gerentes de operaciones.

Experiencia de usuario (UX): Se prioriza una experiencia de usuario que sea fluida y que permita a los usuarios navegar por la plataforma de manera eficiente. Esto incluye la facilidad para interpretar los datos presentados y la capacidad para actuar rápidamente ante las predicciones de fallos.

Prototipo de diseño: Se presentan mockups y wireframes que ilustran cómo los usuarios interactuaron con la plataforma. Estos prototipos muestran escenarios de uso en los que los usuarios revisan el estado de los equipos y toman decisiones de mantenimiento basadas en las predicciones generadas por el modelo XGBoost.

7. RESULTADOS: CASOS DE ESTUDIO

El paso previo al modelado es hacer la ingeniería de características, en casos de problemas de series de tiempo, este proceso es vital para asegurar el éxito del modelo, en este caso se transformaron columnas de telemetry a sus medias móviles y desviaciones estándar en ventanas de 3 horas y 24 horas, de ésta última sólo se agarra el primer dato cada 3 horas para que no nos quedemos con récords "*none*", además, aplicamos "*One Hot Encoders*" a las columnas que lo requieran y tomamos el número de días desde que se hizo mantenimiento a alguna pieza, el siguiente paso es que a la columna target que es "*failures*" le aplicamos un "*label encoder",* por último eliminamos data que no nos sirve como *"MachineId"* y *"Datetime"*. Al final de la ingeniería de datos nos quedamos con las siguientes columnas:

'voltmean_3h', 'rotatemean_3h', 'pressuremean_3h', 'vibrationmean_3h', 'voltsd_3h', 'rotatesd_3h', 'pressuresd_3h', 'vibrationsd_3h', 'voltmean_24h', 'rotatemean_24h', 'pressuremean_24h', 'vibrationmean_24h', 'voltsd_24h', 'rotatesd_24h', 'pressuresd_24h', 'vibrationsd_24h', 'error1_count', 'error2_count','error3_count', 'error4_count', 'error5_count', 'Part_1', 'Part_2', 'Part_3', 'Part_4', 'age', 'model_ALPHA 560 UNIVERSAL','model_JACQUARD TF', 'model_NOVA 62', 'model_NOVA 6HS'. Donde: 'voltmean_3h', 'rotatemean_3h', 'pressuremean_3h', 'vibrationmean_3h', representa la media móvil en la ventana de 3 horas de las "*features*" del dataset "*telemetry"*. Las siguientes "*features"*: 'voltsd_3h', 'rotatesd_3h', 'pressuresd_3h', 'vibrationsd_3h', representan la desviación estándar en una ventana de 3 horas. Las siguientes "*features"*: 'voltmean_24h', 'rotatemean_24h', 'pressuremean_24h', 'vibrationmean_24h', representan la media móvil de la ventana de 24 horas. Las siguientes "*features"*: 'voltsd_24h', 'rotatesd_24h', 'pressuresd_24h', 'vibrationsd_24h', representan la desviación estándar de la ventana de 24 horas. Las siguientes columnas: 'error1_count', 'error2_count','error3_count', 'error4_count', 'error5_count',

representan el "*One Hot Encode*" de el cálculo de errores cada 24 horas por cada máquina. Las siguientes columnas: 'Part_1', 'Part_2', 'Part_3', 'Part_4', representan la cantidad de días que pasaron desde la última vez que se aplicó un mantenimiento a cada pieza o parte. La columna 'age' se refiere a la edad de la máquina. 'model_ALPHA 560 UNIVERSAL', 'model_JACQUARD TF', 'model_NOVA 62', 'model_NOVA 6HS', representan los modelos de las máquinas después de aplicar un "*One Hot Encode*".

Dividimos el dataset en dos, el *training dataset* toma los récords con las fechas más antiguas, y el *testing dataset* toma los récords con las fechas más recientes, esto porque en problemas de series temporales agarrar fechas aleatorias pueden arruinar la eficiencia del modelo ya que la temporalidad es vital para su éxito, y porque la data más antigua se usa para predecir errores futuros, como en cualquier modelo de predicción. En el dataset tenemos un problema de "class imbalance", sin embargo, se continuó para ver si este "class imbalance" era realmente molesto para clasificar los resultados. Donde 4 es None, 0 es Part_1, 1 es Part_2, 2 es Part_3, 3 es Part_4, cada parte que falla de la máquina.

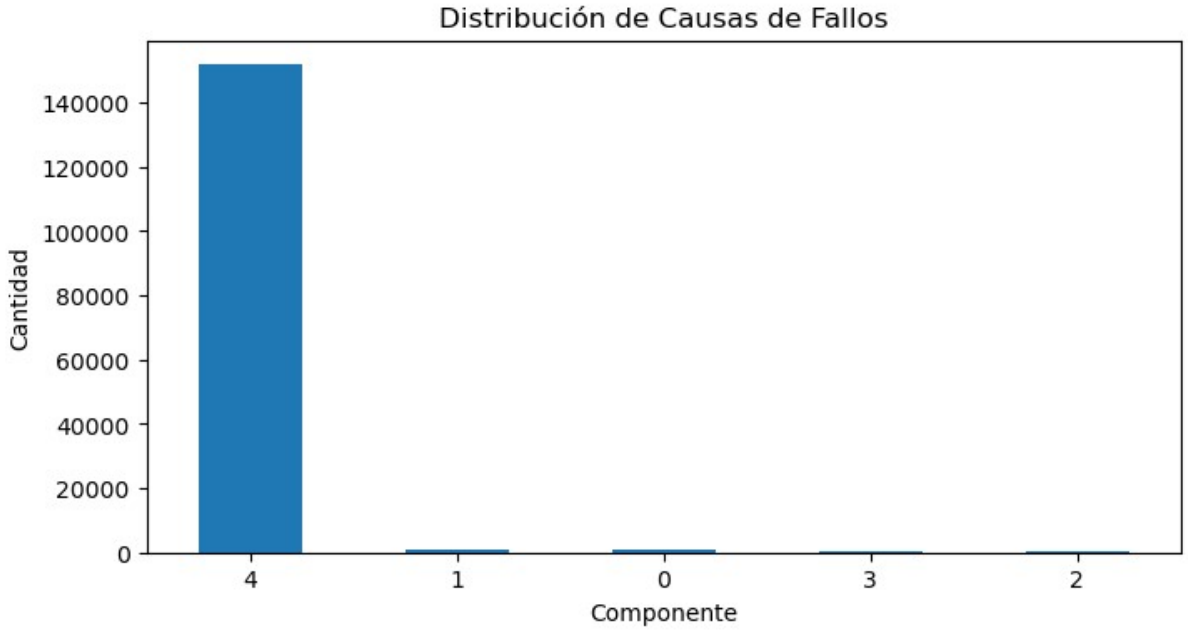


Imagen 13: Distribución de causas de fallos.

En la parte de Modelado, usamos la librería "xgboost", de la cual obtenemos un algoritmo XGBoost , se sabe que el XGBoost es un algoritmo muy eficiente con mucha precisión y muy usado recientemente. Ya existen investigaciones donde se usó este algoritmo como en la investigación: "Sistema de Mantenimiento Continuo para una programación óptima basada en la monitorización de máquinas en tiempo real" (Costa, FJO 2018), y, también, implementaciones en entornos reales como: "XGBoost: A Scalable Tree Boosting System" (Chen, T., & Guestrin, C. 2016).

El siguiente paso es pasar la data por el algoritmo, con lo que obtenemos un modelo basado en el XGBoost que predice muy correctamente a pesar del exagerado "class imbalance" que se tiene. Para que sepamos qué tan bien predecía el modelo, usamos las siguientes métricas: accuracy, precision, recall y f1-score, y una matriz de confusión. Las siguientes tablas e

imágenes representan los resultados de las métricas obtenidas. La primera tabla representa las métricas por cada “target label” se nota claramente que existe una buena predicción del modelo.

	precision	recall	f1-score	support
0	0.92	0.97	0.95	161
1	0.91	0.98	0.94	233
2	1.00	0.89	0.94	95
3	0.98	0.98	0.98	116
4	1.00	1.00	1.00	38512
accuracy			1.00	39117
macro avg	0.96	0.97	0.96	39117
weighted avg	1.00	1.00	1.00	39117

Imagen 14: Métricas de precisión

La Imagen 15: Matriz de confusión. muestra una matriz de confusión que nos otorga de forma visual el buen funcionamiento predictivo de nuestro modelo, no hay muchas fallas en ningún caso.

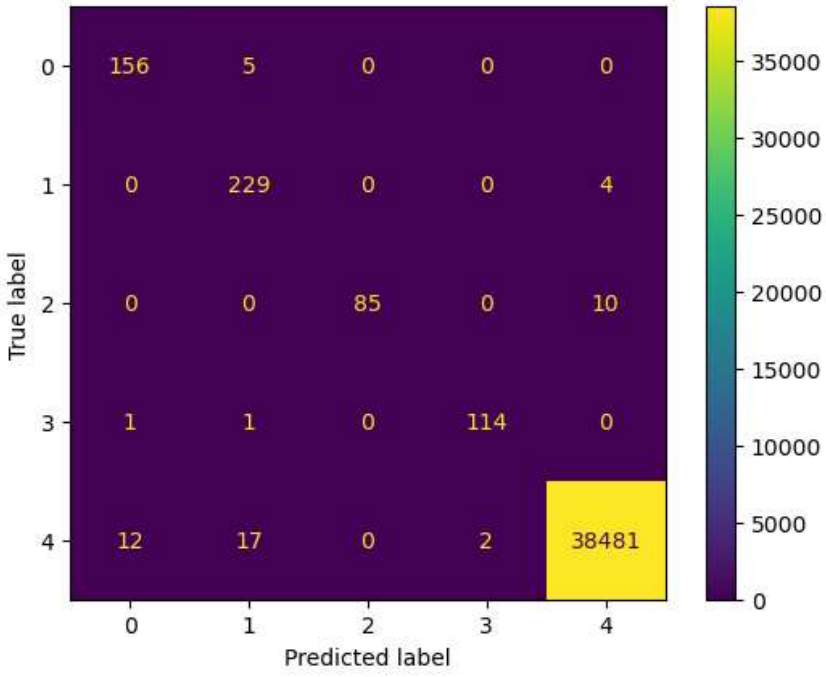


Imagen 15: Matriz de confusión.

Por último, obtenemos la gráfica de explicación del modelo (Imagen 16: Importancia de las características del modelo.) , en el cuál vamos a observar cuáles son las variables que predicen mejor.

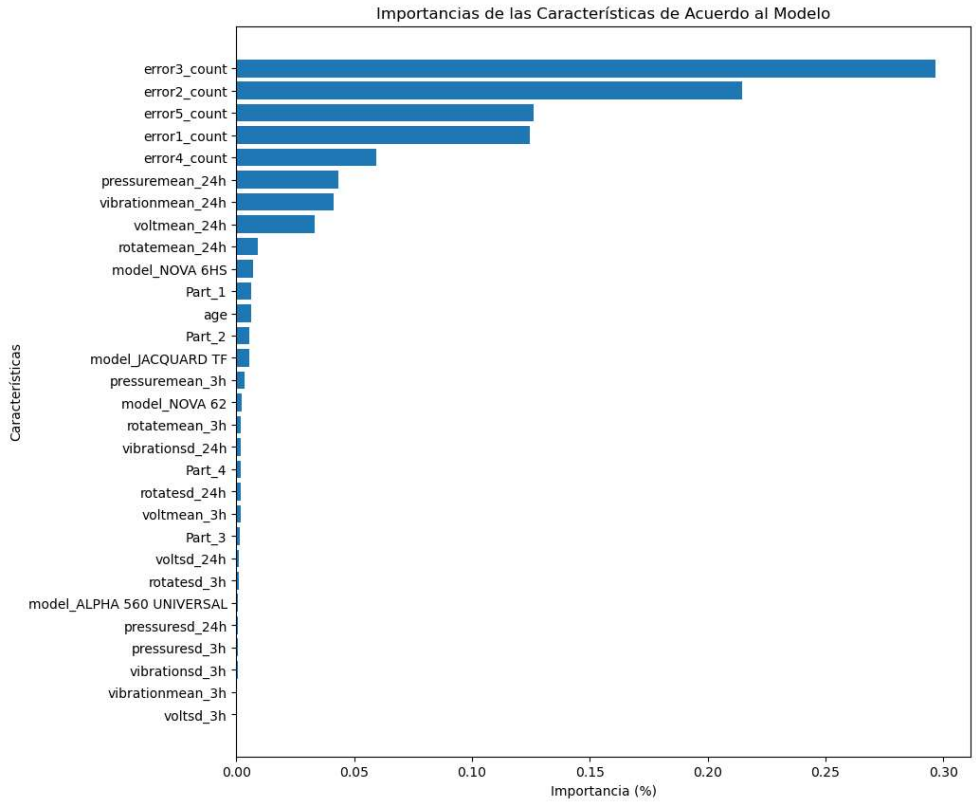


Imagen 16: Importancia de las características del modelo.

En este trabajo también se aplicaron otros algoritmos para modelar y para hacer comparaciones, los algoritmos son LogisticRegression y RandomForestClassifier, dos algoritmos extraídos de la librería de Sklearn, los resultados de ambos fueron inferiores al XGBoost, sin embargo, el RandomForestClassifier fue el que tuvo resultados muy cercanos al XGBoost, en el caso del LogisticRegression los resultados fueron muy malos. En las siguientes tablas e imágenes se presentan los resultados de ambos modelos, la primera tabla e imagen representan las métricas y la matriz de confusión del modelo con el algoritmo RandomForestClassifier, el resto representan las métricas y la matriz de confusión del modelo con el algoritmo LogisticRegression.

	precision	recall	f1-score	support
0	0.91	0.89	0.90	161
1	0.82	0.99	0.90	233
2	1.00	0.81	0.90	95
3	1.00	0.98	0.99	116
4	1.00	1.00	1.00	38512
accuracy			1.00	39117
macro avg	0.95	0.94	0.94	39117
weighted avg	1.00	1.00	1.00	39117

Imagen 17: Métricas de precisión Random Forest Classifier.

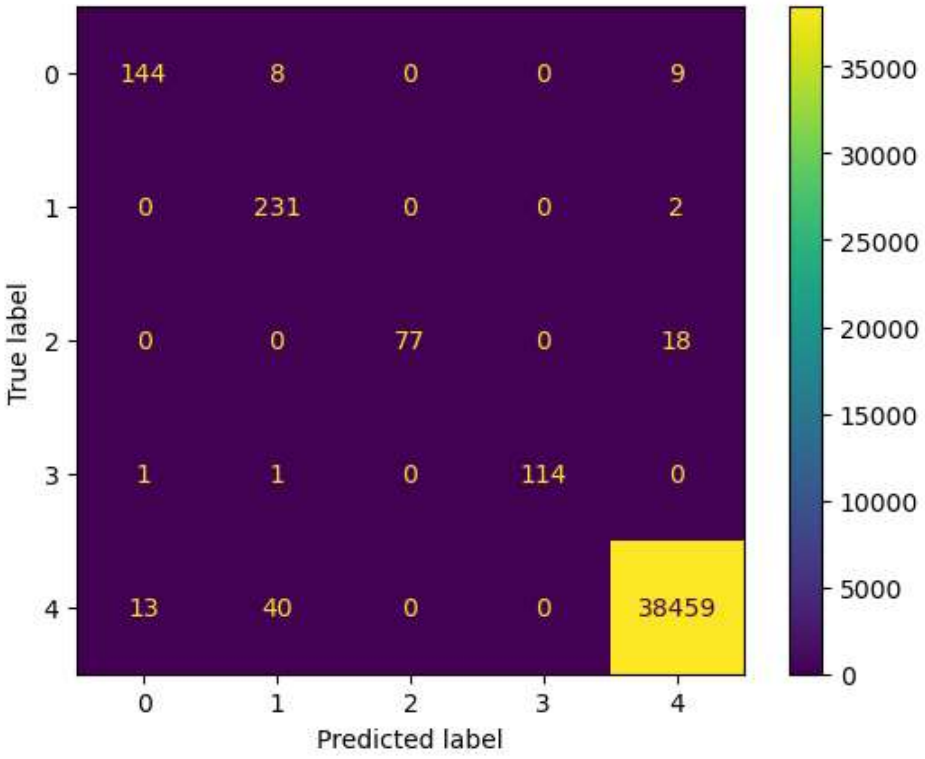


Imagen 18: Matriz de confusión de Random Forest Classifier.

	precision	recall	f1-score	support
0	0.31	0.14	0.19	161
1	0.19	0.09	0.13	233
2	0.09	0.05	0.07	95
3	0.00	0.00	0.00	116
4	0.99	0.99	0.99	38512
accuracy			0.98	39117
macro avg	0.32	0.26	0.27	39117
weighted avg	0.97	0.98	0.98	39117

Imagen 19: Métricas de precisión Logistic Regression.

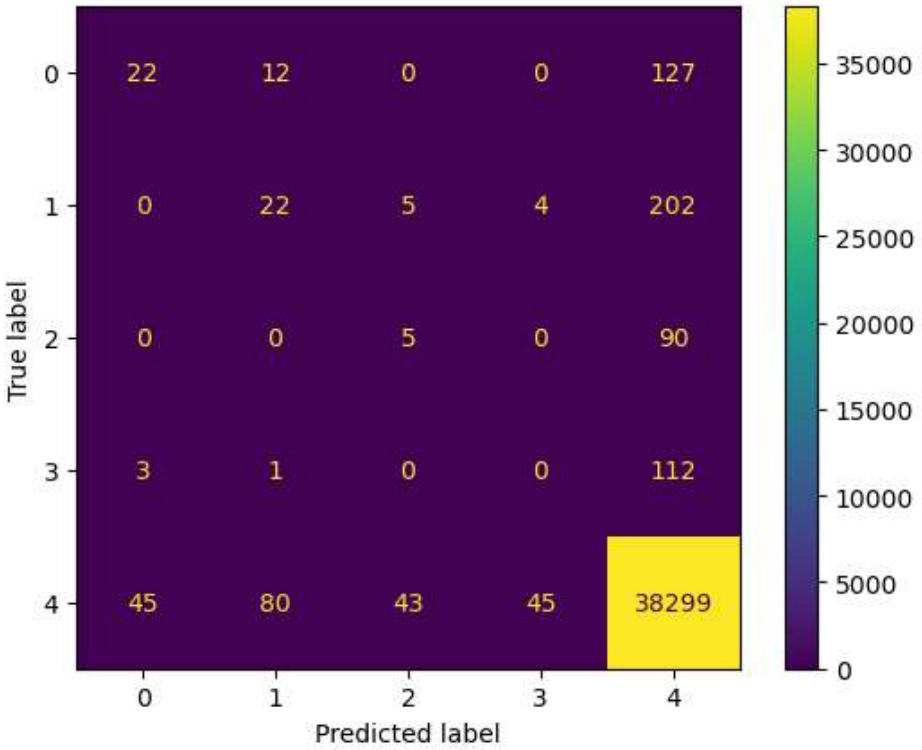


Imagen 20:Matriz de confusión Logistic Regression.

Para la parte de la visualización, se creó un dashboard utilizando StreamLit y Plotly con el fin de mostrar de manera gráfica y profesional los resultados, además de generar una interfaz dinámica donde podemos ingresar diferentes valores de manera manual y validar si con esos datos sería necesario un mantenimiento a la máquina o no, es decir, estamos utilizando el modelo entrenado con XGBoost para poder obtener una respuesta y verificar si el modelo es óptimo o no.

Imagen 21: App de predicción.

Modificando valores en las sliders de la izquierda, podemos obtener un resultado en la pantalla principal indicando si sería parte o no de los mantenimientos, además de indicarnos cual seria la parte que requiere el mantenimiento. Recordemos que en el análisis del modelo y la data, verificamos que los atributos error_count tienen mayor impacto en las decisiones del modelo, por lo que si modificamos más esos atributos tenemos la posibilidad de que exista algún cambio en la predicción, en la imagen podemos observar que con los valores por defecto nos detecta un mantenimiento requerido en la parte 4 de la maquinaria.

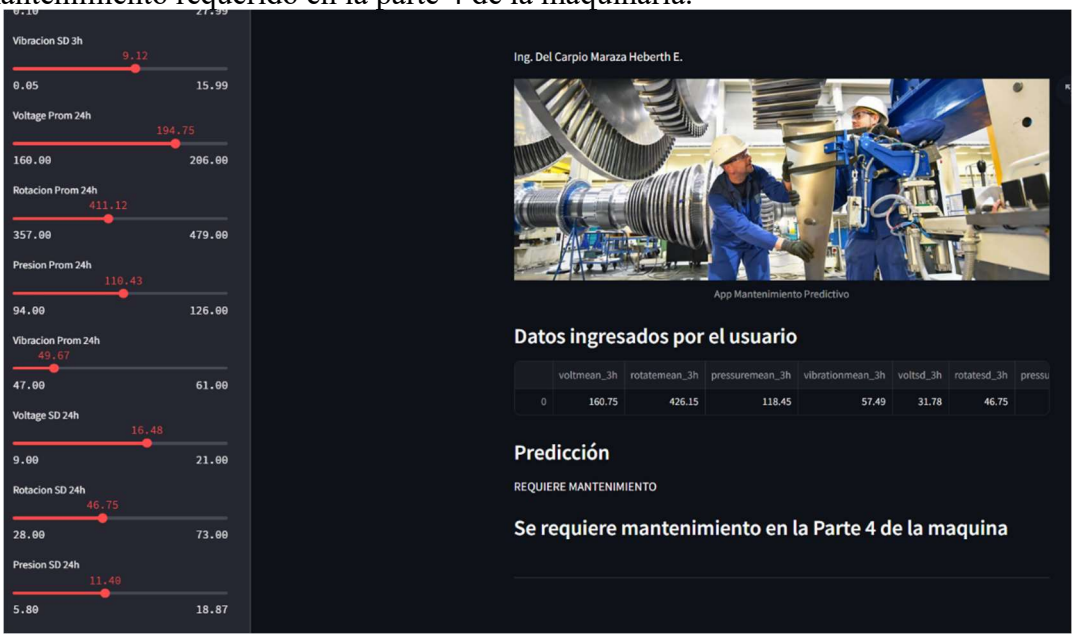


Imagen 22: App de predicción.

Si cambiamos y jugamos con los atributos dependiendo los datos de ingreso podemos obtener otro resultado.

Imagen 23: App de predicción.

Mostrando en este nuevo caso el mantenimiento para la Parte 1, si validamos con datos que según el registro no requiere mantenimiento y colocamos los mismo datos en la interfaz, este nos dará como resultado que no requiere mantenimiento.

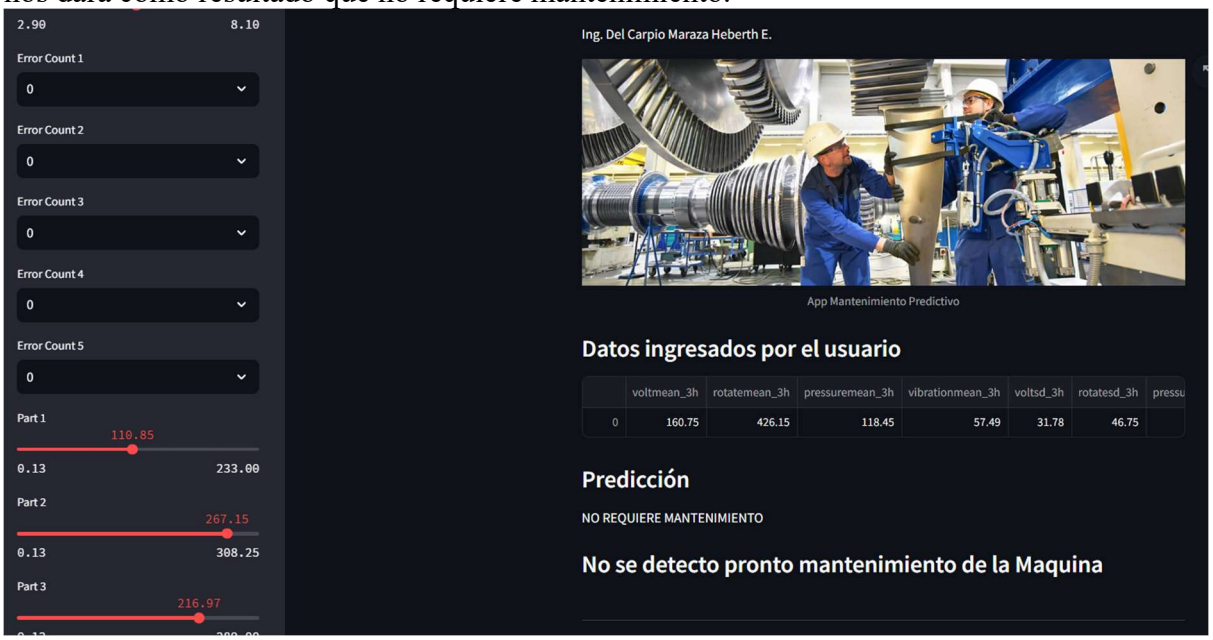


Imagen 24:App de predicción.

Adicionalmente se agregó graficas de medición de KPIS como opción adicional en la interfaz.

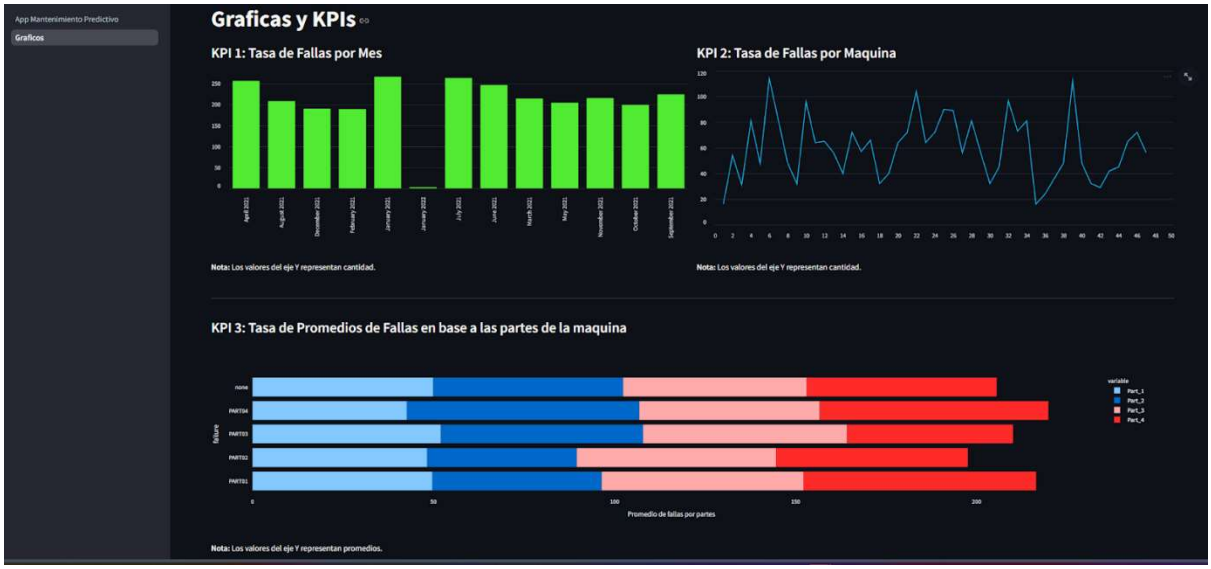


Imagen 25: App de predicción.

8. DISCUSIÓN

El modelo XGBoost fue el que mejor efectividad obtuvo de todos los algoritmos. La calidad de los datos es buena, pero se tuvo que aplicar ingeniería de características para que ésta data pueda ser útil a la hora de procesarla en el modelo, la cantidad es suficiente para el caso, y la precisión que tenemos es muy buena, tanto las métricas como la matriz de confusión muestran que el modelo performa de muy buena manera.

Uso del modelo XGBoost en un entorno real: Un trabajo de mantenimiento predictivo cuesta mucho dinero, y lograr volverlo más eficiente con ayuda de herramientas de machine learning, como la presentada en este proyecto, tiene un potencial de reducir costos y mejorar la eficiencia operativa en la empresa.

Limitaciones del estudio: El estudio se limitó a tener un dataset completo, pero solo de datos entre el 2020 y 2021, se podría poner a prueba con diferentes escenarios y otros datasets similares para otras empresas que requieran tener una predicción sobre los mantenimientos, si bien esto conlleva mucho más tiempo de coordinación y ejecución, es una limitante que se definió en el actual trabajo.

Sugerencias para futuros trabajos: Como trabajos a futuro se podría implementar resultados para una API y que se haga del consumo de otros sistemas o frontends más adecuados y más presentables, así como también automatizaciones como alertas por correo cuando se ingrese nueva información y requiera un nuevo mantenimiento, esto para mantener alertas al personal directamente relacionado al mantenimiento y superiores.

9. CONCLUSIONES

Resumen de hallazgos: Después de aplicar el modelo y obtener las métricas de eficiencia y la matriz de confusión, podemos decir que el modelo basado en el algoritmo XGBoost debe ser eficaz en la predicción de fallos de maquinarias en entornos industriales. Para esto, sin embargo, debemos contar con herramientas aparte como sensores que nos permitan obtener la data necesaria para poder aplicar el modelo y hacer las predicciones para que se haga un mantenimiento predictivo.

Contribución al campo: Se observa según los resultados y las métricas del modelo, podemos tener claro que es una ayuda en el tema de auditar y tener un control interno mucho más eficiente, subrayando la novedad y eficacia del uso de XGBoost en este contexto específico, podemos tener la seguridad de que el modelo predeciría de una manera mas específica el mantenimiento y evitar posibles accidentes causados por maquinaria. El modelo obtuvo entre 85 y 90% de eficiencia en predicciones, lo cual se considera óptimo para los resultados. Posiblemente la generación de una interfaz más dinámica y completa pueda permitir la generación de mejor control de un sistema de predicción y atraer la atención de diferentes interesados en el proyecto.

Impacto potencial: Se puede usar este modelo o crear uno nuevo de acuerdo a las necesidades de la empresa que puede aumentar el ahorro de costos, la mejora de la seguridad y el aumento de la eficiencia operativa en la organización.

Cierre y perspectivas futuras: Se concluye la eficiencia del modelo como se mencionó anteriormente, sin embargo, a futuro y para brindar una solución que ayude a las empresas de manera correcta, se requiere generar todo un sistema de backend y frontend más exacto, intuitivo y amigable al usuario, esto permitirá ofrecer una solución de software viable en el mercado. En el futuro se puede agregar más pre procesos de data, como la aplicación de un PCA que puede ayudar a que el modelo prediga mejor o que un algoritmo menos intensivo en poder computacional como el Logistic Regression sea más eficiente, al igual que aplicar redes neuronales para obtener mejores resultados. Y por último, añadir más sensores a las máquinas para obtener más información de ellas que puedan ayudar a predecir cuándo una falla podría ocurrir.

Referencias

- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
- Heng, A., Zhang, S., Tan, A. C., & Mathew, J. (2009). Rotating machinery prognostics: State of the art, challenges and opportunities. *Mechanical Systems and Signal Processing*, 23(3), 724-739.
- Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7), 1483-1510.
- Kumar, D., Crocker, J., & Knezevic, J. (2004). Reliability and Maintenance Modeling. *IEEE Transactions on Reliability*, 53(2), 213-220.
- Lee, J., Ni, J., Djurdjanovic, D., Qiu, H., & Liao, H. (2014). Intelligent prognostics tools and e-maintenance. *Computers in Industry*, 57(6), 476-489.
- Mobley, R. K. (2002). *An Introduction to Predictive Maintenance*. Butterworth-Heinemann.
- Tsang, A. H. C. (2002). Strategic dimensions of maintenance management. *Journal of Quality in Maintenance Engineering*, 8(1), 7-39.
- Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2017). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115, 213-237.
- Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. MIS Quarterly, 36(4), 1165-1188.
- Dasu, T., & Johnson, T. (2003). Exploratory Data Mining and Data Cleaning. Wiley-Interscience.
- Davenport, T. H., & Harris, J. G. (2007). Competing on Analytics: The New Science of Winning. Harvard Business Review Press.
- Few, S. (2006). Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media.
- Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2010). Multivariate Data Analysis (7th ed.). Pearson Education.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.
- Kantardzic, M. (2011). Data Mining: Concepts, Models, Methods, and Algorithms (2nd ed.). Wiley-IEEE Press.
- Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking. O'Reilly Media.
- Shmueli, G., Patel, N. R., & Bruce, P. C. (2010). Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner (2nd ed.). Wiley.
- Tukey, J. W. (1977). Exploratory Data Analysis. Addison-Wesley.
- Zarsky, T. Z. (2016). Incompatible: The GDPR in the Age of Big Data. Seton Hall Law Review, 47(4), 995-1020.
- Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232.

- Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems, 4765-4774.
- Nielsen, D. (2016). Tree Boosting With XGBoost - Why Does XGBoost Win "Every" Machine Learning Competition? (Master's thesis). Norwegian University of Science and Technology.
- Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.
- Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2017). Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing, 115, 213-237.
- Rojas, W. (2022, diciembre 28). Empresas básicas cierran 2022 con al menos 12 accidentes por falta de mantenimiento y de equipos de seguridad. Correo del Caroní. https://correodelcaroni.com/laboral-economia/empresas-basicas-cierran-2022-con-al-menos-12-accidentes-por-falta-de-mantenimiento-y-de-equipos-de-seguridad/
- Merca2. (2024, agosto 8). España registra 390 muertes por accidentes laborales hasta julio de 2024, un 22% más que en 2023. https://www.merca2.es/2024/08/08/espana-muertes-accidentes-laborales-1877015/
- Zhao, H., Wang, Z., Tan, K., Zhang, L., & Hong, W. (2023). Experimental study on the mechanical behavior of recycled aggregate concrete under complex stress conditions. Case Studies in Construction Materials, 19, e02501. https://doi.org/10.1016/j.cscm.2023.e02501
- Leite, M., Sequeira, A. A., & Paço, A. (2023). Machine learning pipeline for predictive maintenance in polymer 3D printing. *Procedia Computer Science, 229*, 1663-1673. https://doi.org/10.1016/j.procs.2023.07.093