UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN
ELECTRÓNICA Y AUTOMATIZACIÓN
Resolución: RPC-SO-09-No.265-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGISTER

Título del proyecto:
Sistema de detección predictiva de fallas en un cortador láser mediante el algoritmo de Regresión Logística, Red Neuronal Perceptrón Multicapa y Árbol de Decisión
Línea de Investigación:
Ciencias de la ingeniería aplicadas a la producción, sociedad y desarrollo sustentable
Campo amplio de conocimiento:
Ingeniería, industria y construcción
Autor/a:
Eduardo Daniel Montero Narváez
Tutor/a:
Ing. Ernesto Cortijo.Mg.

Quito – Ecuador

2025

APROBACIÓN DEL TUTOR

Yo, **_René Ernesto Cortijo Leyva_** con C.I: **_1719010108_** en mi calidad de Tutor del proyecto de investigación titulado: **Sistema de detección predictiva de fallas en una cortadora láser mediante el algoritmo de Regresión Logística, Red Neuronal Perceptrón Multicapa y Árbol de Decisión**.

Elaborado por: **_Eduardo Daniel Montero Narváez_**, de C.I: **_0603051095_**, estudiante de la Maestría: **_Electrónica y Automatización_**, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., 21 de marzo de 2025

Firmado electrónicamente por:
RENE ERNESTO
CORTIJO LEYVA

_____

Firma

DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE

Yo, Eduardo Daniel Montero Narváez con C.I: 0603051095, autor/a del proyecto de titulación denominado: Sistema de detección predictiva de fallas en una cortadora láser mediante el algoritmo de Regresión Logística, Red Neuronal Perceptrón Multicapa y Árbol de Decisión. Previo a la obtención del título de Magister en Electrónica y Automatización.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar el respectivo trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor@ del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito D.M., 21 marzo de 2025

Firmado electrónicamente por:
EDUARDO DANIEL
MONTERO NARVAEZ

Firma

Tabla de contenidos

Índice de tablas

Índice de figuras

INFORMACIÓN GENERAL

Contextualización del tema

En la industria actual, con el ascenso de la automatización, la inteligencia artificial ha cambiado la forma de gestionar el mantenimiento, mejorando la operación de los equipos de precisión como el cortador láser. Estos dispositivos necesarios en la elaboración de productos en diversas fábricas como la metalmecánica, la automotriz, elaboración de mobiliario así como también la producción de componentes electrónicos. Su habilidad para efectuar cortes precisos en los materiales como metales, plásticos, madera ha mejorado la eficacia de la producción. Pero debido al uso continuo, la falta de planes de mantenimiento produce fallas inesperadas en componentes importantes, como los motores de paso a paso, los espejos ópticos, los ventiladores de enfriamiento, impactando a la productividad ocasionando costos elevados por reparaciones y tiempos de inactividad **(Tezanos, 2019)**.

Contar con la operación continua del cortador láser es fundamental para un trabajo óptimo de producción en las fábricas. En la actualidad, las empresas realizan estrategias de mantenimiento correctivo o preventivo programado, las mismas que no garantizan las interrupciones no planificadas. En este ámbito, diseñar un sistema predictivo basado en algoritmos de inteligencia artificial facilitara anticiparse a las fallas, ofreciendo un diagnóstico temprano de anomalías, brindando una intervención oportuna. Usando técnicas como Regresión Logística, Redes Neuronales y Árboles de Decisión, facilita la identificación de patrones en variables operativas tales como temperatura y humedad, mejorando la gestión del mantenimiento **(Tezanos, 2019)**.

La industria en que se utilizan el cortador láser involucra personales operativos, técnicos de mantenimiento y profesionales de ingeniería, quienes supervisan los procesos productivos. La adecuación de tecnologías de monitoreo inteligente en esta máquina ayudara a mejorar la toma de decisiones durante el uso de la máquina, minimizando la dependencia de inspecciones manuales y diagnósticos basados en la experiencia empírica. Este punto de vista no solo incrementará la eficiencia en la detección de fallas, sino que también contribuirá a una mayor seguridad en el ambiente de trabajo y a una reducción en el desperdicio de materiales **(Tezanos, 2019)**.

Es una estrategia, el desarrollo del sistema de mantenimiento predictivo a través de una inteligencia artificial, la cual representa un avance significativo en la modernización de la industria manufacturera. El acortar los tiempos de inactividad y optimizar los recursos técnicos, la competitividad se fortalece en las empresas garantizando una mayor calidad en los productos finales. Este proyecto propone una solución escalable, replicable que pueda implementarse en distintos ámbitos industriales lo cual consolida el uso de tecnologías avanzadas para la mejora de procesos y obteniendo un óptimo rendimiento de los equipos **(Lozano-Gutiérrez, 2016)**.

Problema de investigación

En la industria manufacturera la eficiencia y la continuidad productiva dependen del adecuado funcionamiento de las máquinas agregadas en las etapas de producción. En cuanto al cortador láser, la misma es esencial para la elaboración de piezas en diversos sectores, metalmecánica, la industria automotriz, la producción de mobiliario. Sin embargo su uso continuo da como resultado un alto riesgo de fallas en los sistemas mecánicos, ópticos y de enfriamiento, lo que genera tiempos de inactividad no planificados, desperdicio de materiales y costos de reparación elevados **(Lozano-Gutiérrez, 2016)**.

Hoy en día, el mantenimiento de esta máquina se lo hace mayormente de forma correctiva o mediante cronogramas fijos, impidiendo anticipar fallas antes de que ocurran. La ausencia de un sistema de monitoreo predictivo utilizando la inteligencia artificial limita detectar anomalías en variables operativas claves como la temperatura y la humedad, las mismas que afectan directamente al rendimiento de los motores eléctricos paso a paso, la estabilidad de los espejos ópticos y la eficacia del sistema de enfriamiento. El resultado de estas anomalías para las empresas son pérdidas económicas, una reducción en la calidad del producto final debido a fallas inesperadas en sus equipos **(Villegas , 2020)**.

La falta de herramientas tecnológicas que detecten tempranamente las fallas limita la capacidad de respuesta del personal técnico, que utiliza inspecciones manuales, visuales juntamente con la experiencia empírica para diagnosticar problemas. Esto retrasa las actividades de mantenimiento, sino que aumentan la posibilidad de cometer errores en la identificación de fallas, afectando la funcionabilidad de la maquinaria y la productividad en general **(Tezanos, 2019)**.

Si esta situación persiste, los costos operativos continuarán en aumento, la competitividad de las empresas se verá comprometida y la durabilidad del cortador láser se reducirá significativamente. La falta de un sistema predictivo también podría traducirse en una menor eficiencia energética, debido al uso incorrecto de los recursos y la necesidad de reemplazar componentes antes de lo necesario. Asimismo, el riesgo de fallas críticas en la maquinaria aumentará, afectando la seguridad en el entorno industrial y generando impactos negativos en la producción **(Tezanos, 2019)**.

Por estas razones, se hace necesario la implementación de un sistema de detección predictiva de fallas en el cortador láser, empleando algoritmos de Regresión Logística, Redes Neuronales y Árboles de Decisión **(Villegas , 2020)**.

Objetivo general

Diseñar un sistema de detección predictiva de fallas en un cortador láser mediante el algoritmo de Regresión Logística, Red Neuronal Perceptrón Multicapa y Árbol de Decisión

Objetivos específicos

- Contextualizar los fundamentos teóricos sobre la inteligencia artificial para la predicción de fallas en equipos.
- Determinar la metodología del desarrollo de los algoritmos para la detección predictiva de fallas.
- Diseñar los algoritmos de inteligencia artificial de Regresión Logística, Red Neuronal Perceptrón Multicapa y Árbol de Decisión para la detección predictiva de fallas en un cortador láser.
- Ejecutar los algoritmos en la inteligencia artificial creados para la detección de fallas en un cortador láser.
- Observar los resultados mediante pruebas en la interfaz diseñada del sistema.

Vinculación con la sociedad y beneficiarios directos:

La realización del sistema de detección predictiva de fallas en un cortador láser hará un cambio profundo en la industria manufacturera, ya que mejorara la eficiencia en los procesos productivos y reducirá desperdicios de material. Al mejorar el mantenimiento de las máquinas, se contribuye al ahorro de energía eléctrica, disminuye los costos de operación dando como resultado el aumento de la vida útil de los equipos. Además al disminuir los tiempos de inactividad, se mejorara la capacidad de respuesta ante fallas imprevistas, ganando una mejor estabilidad en la producción industrial.

En cuanto a la formación y el desarrollo profesional, este proyecto ayudará con la capacitación de operarios, técnicos del personal de mantenimiento en el uso de herramientas tecnológicas orientadas al monitoreo y diagnóstico de fallas. La información y los resultados obtenidos podrán ser utilizados para la elaboración de materiales educativos y guías para las buenas prácticas en el mantenimiento predictivo, contribuyendo al crecimiento del conocimiento en este ámbito. Como resultado se podrán generar publicaciones científicas que promuevan la difusión y el conocimiento de estas tecnologías dentro del sector industrial.

Los usuarios directos de este proyecto son los trabajadores de la producción, técnicos de mantenimiento e ingenieros de procesos, los mismos que tendrán acceso a datos actualizados que los ayudara con la elección de decisiones estratégicas en la administración del mantenimiento.

Las empresas podrán reducir costos y mejorar en la eficiencia operativa que beneficiará a las organizaciones que dependen de cortadoras láser para sus operaciones. En un ámbito más amplio la implementación de este sistema tecnológico fomentará la modernización de la industria, promoviendo un entorno productivo más inteligente, seguro y sostenible.

CAPITULO I: DESCRIPCION DEL PROYECTO

1.1 Contextualización general del estado del arte

La investigación al problema de detección predictiva de fallos en el cortador láser se basa en la creciente demanda de optimizar los procesos de mantenimiento industrial a través del desarrollo de la inteligencia artificial. La industria manufacturera se topa con retos operativos originados por errores imprevistos en equipos de gran precisión, lo que provoca costos altos y periodos de parada. Para analizar este asunto, se han buscado varias fuentes de información, como publicaciones científicas, libros, congresos internacionales en el campo de mantenimiento predictivo, aprendizaje automático y automatización industrial. La elección de literatura se ha fundamentado en estudios recientes, siendo prioritarias las investigaciones de los últimos años, así como en las bases de datos de IEEE Xplore, Springer, ScienceDirect y repositorios institucionales relevantes. Este estudio se ubica en el contexto de las tendencias presentes de la industria 4.0 en la que la aplicación de modelos de inteligencia artificial facilita la predicción de averías y se constituye un progreso importante para mejorar la administración del mantenimiento **(Lozano-Gutiérrez, 2016)**.

El análisis del problema de investigación sobre la detección predictiva de fallas en cortadoras láser se fundamenta en la creciente necesidad de mejorar los procesos de mantenimiento industrial mediante el uso de inteligencia artificial.

La industria manufacturera enfrenta desafíos operativos derivados de fallos inesperados en equipos de alta precisión, lo que genera costos elevados y tiempos de inactividad. Para abordar esta problemática, se han revisado diversas fuentes documentales, incluyendo artículos científicos, libros y conferencias internacionales en las áreas de mantenimiento predictivo, aprendizaje automático y automatización industrial. La selección de literatura se ha basado en publicaciones recientes, priorizando estudios de los últimos cinco años en bases de datos como IEEE Xplore, Springer, ScienceDirect y repositorios institucionales **(Lozano-Gutiérrez, 2016)**.

Los estudios comprenden el mantenimiento predictivo, datos en tiempo real, la utilización del aprendizaje automático para identificar irregularidades en sistemas industriales. Estudios recientes han comprobado la factibilidad de estos métodos en el sector de la manufactura. Dueñas-Ramírez et.al (2020) realizando una investigación en dos fábricas de Colombia con el objetivo de valorar la aplicación del mantenimiento predictivo mediante tecnologías de la Industria 4.0. Utilizando instrumentos digitales con los cuales recopilaron información, consiguiendo así mejorar los tiempos de intervención y disminuir los gastos operacionales. Los resultados mostraron que la utilización de sensores IoT junto con el análisis de datos de la nube posibilitaron que estas fábricas incrementaran su eficacia, obteniendo una disminución del tiempo de inactividad en un 25%, lo cual constituye un progreso considerable **(Ramírez, 2020)**.

Por otra parte, Villegas Morphy (2020) realizo un estudio en la industria petrolera venezolana, donde se implementó un modelo de inteligencia artificial para la predicción de fallas en las maquinarias de turbocompresores utilizados en la Planta Compresora MUSCAR de Petróleos de Venezuela (PDVSA). Aplicando la metodología CRISP-DM junto con redes neuronales y lógica difusa logró identificar fallas frecuentes, mejorando la toma de decisiones en la planificación del mantenimiento, alcanzando una precisión del 99.96% en la clasificación de fallas y reduciendo significativamente los tiempos de inactividad de los equipos **(Villegas , 2020)**.

De manera similar en, Prellezo Tezanos (2019) llevó a cabo un estudio en una planta de fabricación de electrodomésticos, con el objetivo de implementar un sistema de mantenimiento predictivo basado en tecnologías de la Industria 4.0. Para ello, se utilizaron sensores conectados a PLCs, un IoT Gateway con el protocolo MQTT y almacenamiento en la nube AWS para el procesamiento de datos. El resultado de la introducción de este sistema se obtuvo una disminución con tiempo de parada por el mantenimiento correctivo en un 2% del OEE, optándolo por el mantenimiento predictivo fuera del horario productivo esto mejora la disponibilidad de la maquinaria, mejorando la eficiencia del proceso productivo. Por último, Hurtado-Cortés et al. (2016) realizo un estudio en la Universidad Distrital Francisco José de Caldas en Colombia, donde estudiaron algunas técnicas de inteligencia artificial aplicadas a la detección y diagnóstico de fallas en sistemas dinámicos. Compararon métodos como redes neuronales artificiales, sistemas de lógica difusa, sistemas neurodifusos y sistemas inmunes artificiales, destacando sus características y aplicaciones en la supervisión de procesos industriales **(Tezano, 2019)**.

Estos estudios facilitan la viabilidad de aplicar modelos de inteligencia artificial en la predicción de fallas en cortadoras láser. El avance de la machine learning y la utilización de sensores industriales abren las puertas para mejorar la precisión del diagnóstico de fallas y agilizar la toma de decisiones en mantenimiento. La presente investigación busca diseñar un modelo fudamentado en regresión logística, redes neuronales y árboles de decisión para optimizar la planificación del mantenimiento, para reducir costos y mejorar la productividad. Con respecto a investigaciones similares, se han hecho trabajos que tienen relación con la presente investigación, contribuyendo al desarrollo del mantenimiento predictivo en diversos sectores. Sánchez Cocha (2024) realizo un modelo de detección de fallas en motores de corriente continua utilizando machine learning, implementando redes neuronales artificiales e herramientas de simulación como MATLAB/Simulink y Quartus. Su trabajo demostró que el algoritmo Levenberg-Marquardt dio como resultado una elevada precisión en la identificación de fallas internas y externas, dando valides a la aplicación de técnicas de inteligencia artificial en el mantenimiento de maquinarias electromecánicas. En el aporte de Reino Cárdenas (2022) el cual desarrolló un sistema basado en redes neuronales y protocolo MQTT para la predicción del desgaste en turbinas Pelton en la central hidroeléctrica Río Verde Chico, Ecuador. Usando sensores de turbidez, una Raspberry Pi 3 para la recopilación de datos, su investigación permitió minimizar fallos operativos y reducir paradas imprevistas, mejorando la eficacia del mantenimiento predictivo en sistemas industriales de alto impacto. Finalmente, Keewong Zapata et al. (2023) implementaron un modelo neuronal para predecir la rugosidad superficial en cortes realizados mediante la tecnología de chorro de agua con abrasivos, con redes neuronales multicapa, optimización de parámetros de corte, su estudio mejoró significativamente la calidad del proceso de manufactura y redujo desperdicios de material **(Cárdenas, 2022)**.

Estos estudios refuerzan la utilidad de la inteligencia artificial en el mantenimiento predictivo estableciendo un marco teórico para la aplicación de estos modelos en cortadoras láser industriales.

1.2 Proceso investigativo metodológico

Este estudio desarrolla una investigación cuantitativa, cuyo objetivo es desarrollar el sistema para la detección predictiva de fallas en el cortador láser utilizando un algoritmo de inteligencia artificial. La investigación se dirige hacia la optimización del mantenimiento industrial a través del análisis de datos, facilitando anticipar fallas para reducir tiempos de inactividad. Se adoptó un diseño experimental, ya que se probarán modelos de aprendizaje automático sobre datos simulados.

En la ejecución de este estudio, se aplicarán métodos teóricos y prácticos que permitirán el desarrollo y validación del sistema propuesto. En cuanto a lo teórico, se llevará a cabo una revisión sistemática de la bibliografía en cuanto al mantenimiento predictivo, la inteligencia artificial y las aplicaciones industriales del aprendizaje automático. Se revisaran modelos de regresión logística, redes neuronales artificiales y árboles de decisión, que han resultado ser útiles para identificar fallos en diversos sectores industriales. En cuanto a la ejecución del sistema, se llevará a cabo el diseño de estos modelos utilizando herramientas como Python, TensorFlow y Scikit-learn. Se realizarán pruebas experimentales con datos simulados de temperatura y vibración.

Para la obtención de información, se usaran diversas técnicas cualitativas y cuantitativas. Se aplicará la observación en entornos industriales para identificar patrones operacionales y condiciones de falla en el cortador láser.

La comunidad objeto de estudio está conformada por técnicos de mantenimiento y expertos en IA y mantenimiento predictivo. Para obtener la muestra, se empleó un muestreo no probabilístico por conveniencia, buscando profesionales en industrias que cuenten con un historial de mantenimiento documentado y acceso a sensores de monitoreo en sus equipos. A continuación, se presenta en la Tabla 1 la distribución de la población y la muestra seleccionada**:**

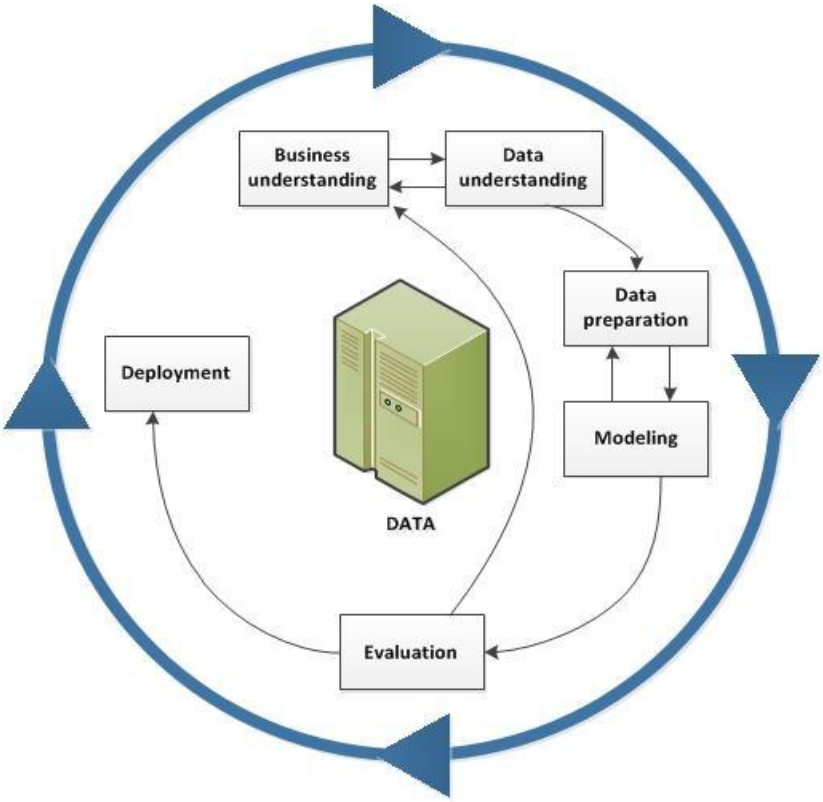
Tabla 1.

Distribución de la población y la muestra seleccionada

Categoría	**Población Total**	**Muestra Seleccionada**
Técnicos de mantenimiento	10	5
Expertos en IA y mantenimiento predictivo	5	5

Figura 1.

Ciclo de vida de minería de datos



El procedimiento de trabajo utilizado en la ejecución del proyecto se fundamentara en la aplicación del ciclo CRISP-DM (Cross-Industry Standard Process for Data Mining) mostrada en la Figura 1, utilizado en proyectos de minería de datos e inteligencia artificial. Esto incluyó las siguientes fases: comprensión del negocio, recopilación y preparación de datos, modelado de algoritmos, evaluación de resultados e implementación del sistema. Se ejecutaron pruebas Finalmente, los resultados obtenidos fueron validados mediante métricas de exactitud, precisión, recall y F1 score, dando la efectividad del modelo propuesto en la mejora del mantenimiento predictivo en la cortadora láser.

CAPÍTULO II: PROPUESTA

2.1 Fundamentos teóricos aplicados

Esta investigación se realiza con respecto a la automatización industrial, inteligencia artificial y mantenimiento predictivo, esencial para la mejora de procesos de fabricación en el sector de la manufactura. La Automatización Industrial ayuda en el monitoreo de maquinaria a través de sensores y algoritmos de aprendizaje automático, posibilitando la identificación precoz de averías en dispositivos esenciales. Por otro lado, la inteligencia artificial ofrece herramientas para el estudio de grandes cantidades de datos, mejorando la toma de decisiones fundamentada en patrones y tendencias operativas. En este caso, el mantenimiento predictivo se fundamenta en el estudio de datos operativos para anticipar posibles errores y mejorar la eficacia de los sistemas de fabricación

2.1.1 Inteligencia Artificial

Se comprende a la Inteligencia Artificial como un sistema del tipo informático que analiza e interpreta la información mediante el desarrollo de algoritmos que emulan el pensamiento humano. Con ello se intenta emular características importantes del comportamiento humano como:

- Razonar de forma similar al ser humano
- Interactuar con las personas a través del lenguaje natural
- Aprender de sus resultados y adaptarse a nuevos ambientes
- Resolver problemas de manera rápida y eficiente

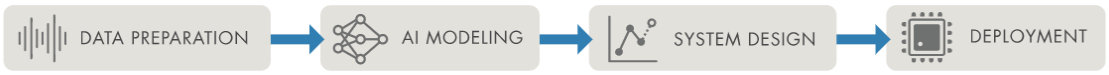
Dentro de la Inteligencia Artificial se tiene varios campos de desarrollo entre ellos:

- Aprendizaje automático
- Aprendizaje profundo

En la siguiente figura se muestra la forma de trabajo de la Inteligencia Artificial

Figura 2.

Flujo de trabajo de la Inteligencia Artificial



El proceso inicia por la toma de datos en bruto y hacerlos útiles para un modelo preciso, eficiente y significativo. La preparación de los datos requiere conocimientos especializados como experiencia en señales de voz y audio, procesamiento de imagen y video.

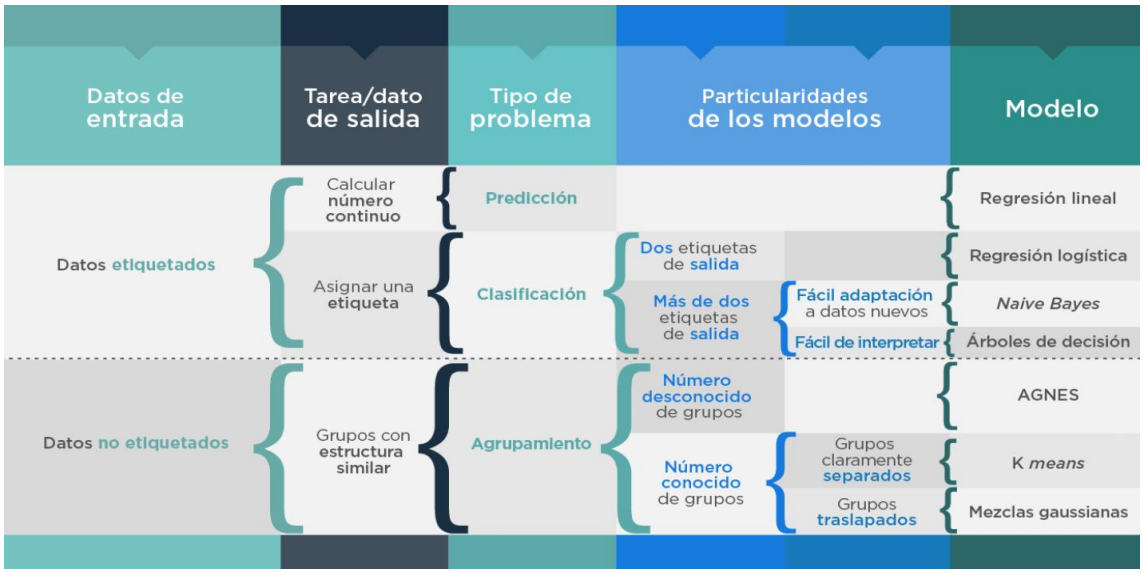
2.1.2 Machine Learning

También conocido como aprendizaje automático se basa en algoritmos matemáticos para que las máquinas puedan procesar una gran cantidad de datos, adquieran conocimiento por sí mismas y realicen predicciones.

Dependiendo de ciertas características como el tipo de entrada, el tipo de dato de salida, el problema a resolver, entre otros, se puede utilizar diferentes algoritmos. En la Figura se muestra los algoritmos que se pueden implementar.

Figura 3.

Elección del Modelo de Machine Learning



En contraposición, el mantenimiento predictivo es una táctica sofisticada que emplea información en tiempo real y modelos de inteligencia artificial para prever errores en maquinaria industrial. En el contexto de la Industria 4.0, este ha progresado con la utilización de sensores de IoT, sistemas de vigilancia en la nube e algoritmos de aprendizaje automático. Por otra parte el mantenimiento correctivo y preventivo, el mantenimiento predictivo disminuye los gastos y periodos de parada al detectar irregularidades antes de que provoquen errores críticos**.**

2.1.3 Modelos de Inteligencia Artificial

En la predicción de fallas de la cortadora láser, se usan modelos de aprendizaje automático que son Regresión Logística, Redes Neuronales Artificiales (RNA) y Árboles de Decisión. Cada algoritmo permite analizar patrones en los datos y generar predicciones sobre el estado de los equipos.

- **Regresión Logística**

Es un modelo estadístico que se utiliza para la clasificación binaria, permitiendo predecir la repetición de fallas basadas en datos operacionales históricos. Este algoritmo es útil para detectar patrones de desgaste en los componentes del cortador láser.

- **Redes Neuronales Artificiales**

Modelos de aprendizaje profundo que emulan la funcionalidad del cerebro humano para identificar relaciones no lineales en los datos. En el proyecto, se utiliza una red neuronal multicapa para analizar la interacción entre variables de esta forma predecir la aparición de fallas en motores eléctricos paso a paso y sistemas ópticos.

- **Árboles de Decisión**

Modelos de clasificación que organizan los datos en forma de nodos y ramas realizando la identificación de patrones en el comportamiento de los equipos. Su capacidad para interpretar relaciones complejas lo convierte en una herramienta clave para la detección de anomalías en los sistemas de cortadora láser.

2.1.4 Python y Sklearn

Python es un lenguaje de programación común mente utilizado en la programación de aplicaciones industriales debido a su versatilidad, facilidad de integración con diversas tecnologías y su extenso ecosistema de bibliotecas especializadas. En el proyecto, Python se emplea en dos áreas esenciales: en un diseño de interfaz gráfica de usuario (GUI) también en el diseño de modelos de inteligencia artificial para la predicción de fallas en el cortador láser.

- **Diseño de la Interfaz Gráfica de Usuario (GUI)**

Para la elaboración de la interfaz gráfica, se utilizaron bibliotecas como Tkinter y PyQt, en las cuales se desarrolla aplicaciones interactivas con un diseño intuitivo. La interfaz proporciona acceso a utilitarios como autenticación de usuario, visualización de gráficos y ejecución de predicciones.

- **Implementación de Modelos de Inteligencia Artificial**

Python brinda un ecosistema robusto para el desarrollo de algoritmos de aprendizaje automático. En el proyecto, se implementaron modelos de Regresión Logística, Árboles de Decisión y Redes Neuronales Artificiales (RNA) usando la biblioteca Scikit-learn. Estos modelos permiten predecir fallas en los componentes del cortador láser fundamentados en datos históricos de temperatura y humedad.

2.2 Descripción de la propuesta

Un sistema basado en el mantenimiento predictivo que es diseñado consta de alginas fases vinculadas que facilitan la realización de datos simulados para su procesamiento, análisis de esta forma identificar fallos en el cortador láser. Debido a que los datos empleados en este estudio son producidos de manera artificial, la meta es replicar las condiciones reales de funcionamiento de estos dispositivos para valorar el desempeño de los modelos de inteligencia artificial en un ambiente regulado.

La estructura del sistema sigue un flujo ordenado que garantiza la correcta operación de los algoritmos de aprendizaje automático y una visualización de resultados mediante una interfaz gráfica.

A continuación, se explica la funcionalidad del sistema:

- **Generación de Datos Simulados:** Debido a la falta de sensores físicos porque no se cuenta con una cortadora láser real, se generan datos sintéticos de variables como temperatura y humedad. Estos datos simulan patrones de

operación normales y fallas para alimentar los modelos predictivos.

- **Procesamiento de Datos:** Esta información simulada es preprocesada para eliminar valores atípicos normalizando así las variables de entrada. Se utilizan herramientas como Pandas y NumPy para la limpieza y organización de los datos antes de alimentar los modelos de inteligencia artificial.
- **Entrenamiento de Modelos de IA:** Se realizan modelos de aprendizaje automático, como Regresión Logística, Árboles de Decisión y Redes Neuronales Artificiales. Los modelos son entrenados con los datos simulados para identificar patrones asociados a fallas mecánicas, ópticas y electrónicas en la cortadora láser.
- **Evaluación del Modelo:** Se ejecuta la validación de los modelos mediante métricas como precisión, recall y F1-score, cuya etapa permite seleccionar el algoritmo con mejor rendimiento para la predicción de fallas en entornos simulados basado en esto evaluar su viabilidad para futuras implementaciones en entornos industriales reales.
- **Implementación en la Interfaz:** La interfaz gráfica realizada en Python con Tkinter o PyQt permite la interacción con el usuario, dando opciones de visualización de datos, ejecución de predicciones y generación de reportes sobre el estado del sistema.
- **Monitoreo de Simulación:** En el sistema se observa y controla la supervisión de los datos generados, analizando patrones de comportamiento y verificando la precisión de los modelos en la detección de condiciones de operación anómalas.
- **Predicción de fallas:** Se designara el mejor modelo de algoritmo de machine learning para realizar la predicción de 3 tipos de fallas: 1. Sobrecalentamiento de los motores eléctricos de paso a paso (Falla Mecánica), 2. Condensación en los espejos ópticos (Falla Óptica), 3. fallo del ventilador de enfriamiento (Falla

Electrónica)

- **Actividades de Mantenimiento:** Dependiendo de la falla detectada se detalla las actividades de mantenimiento que debe realizar.

a. Estructura general

En la figura se muestra un diagrama de la propuesta del sistema predictivo de fallas.

Figura 4.

Propuesta del sistema predictivo de fallas

b. Explicación del aporte

El sistema incorpora diversas funcionalidades interactivas que permiten la detección y análisis predictivo de fallas en el cortador láser, generando un entorno dinámico y automatizado para la toma de decisiones. La interactividad se basa en la integración de la interfaz gráfica intuitiva en la cual se monitorea los datos, la ejecución de modelos de inteligencia artificial en la interpretación de resultados.

Se han usados diversos recursos tecnológicos con los cuales se realizó la implementación del mantenimiento predictivo. Entre los recursos más importantes se encuentran:

Lenguaje de programación Python: Fundamental para la realización del sistema, integrando bibliotecas como Pandas, NumPy y Scikit-learn para el procesamiento de datos y modelado de IA.

En el código del programa se incluyen las librerías con las que se realizará el procesamiento de los datos (ver Figura 5).

Figura 5.

Librerías del sistema

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
```

- **Interfaz gráfica (GUI) con Tkinter/PyQt**: Permite la visualización de datos y predicciones de fallas en una plataforma accesible para el usuario final.

Se tiene la venta de Menú principal en la que se puede acceder al sistema o a la ventana de ayuda (Figura 6).

Figura 6.

Menú principal

En la ventana de Login de la interfaz gráfica el operador del sistema debe ingresar el usuario y la contraseña (ver Figura 7).

Figura 7.

Ventana Login

Si el usuario ingresa mal el usuario y la contraseña se presenta el mensaje de "Usuario o contraseña incorrectos" como se observa en la (Figura 8).

Figura 8.

Ventana Login usuario y contraseña incorrectos

Una vez ingresado correctamente el usuario y contraseña se habilita la ventana de menú en dónde se tiene la opción de ingresar a la ventana en dónde se carga el conjunto de datos que se analizará y de realizar la predicción de fallas en la cortadora láser (ver Figura 9).

Figura 9.

Ventana de Menú Principal

En la ventana de análisis de datos (ver Figura 10) se tiene la opción de cargar los datos que se van a analizar, graficar los datos de temperatura y humedad, así como también se puede observar el valor de la media, desviación estándar, máximo y mínimo del conjunto de datos que se ha cargado.

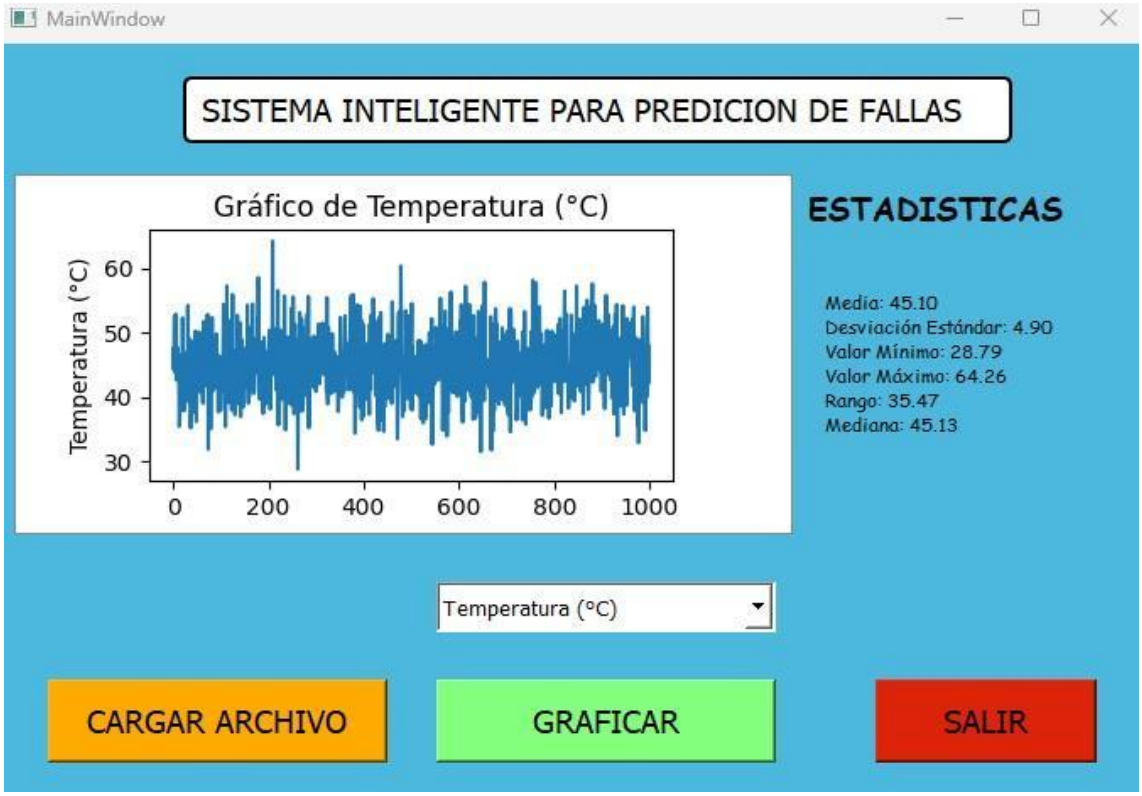
Figura 10.

Ventana de Análisis de datos – Cargar Archivo

Se selecciona la variable a graficar (Temperatura o Humedad) y se visualizan los datos (ver Figura 11).

Figura 11.

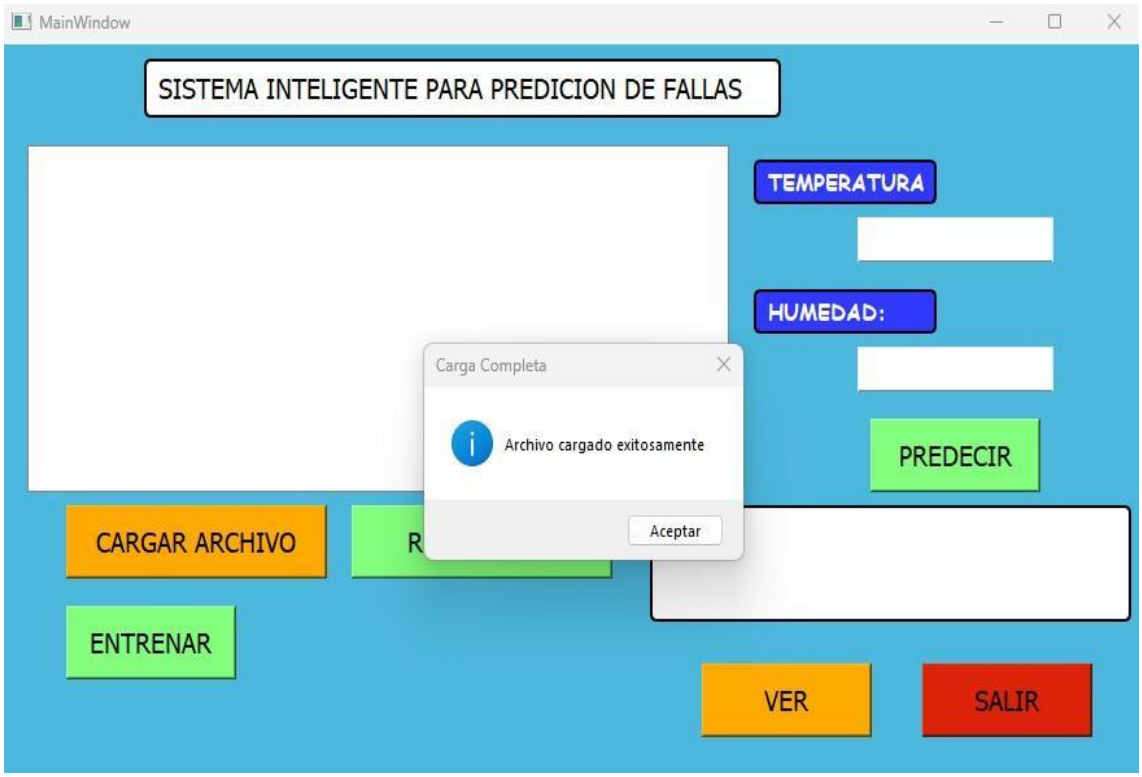
Ventana de Análisis de datos – Gráfica de Datos



En la ventana de Predicciones (ver Figura 12) se puede cargar el archivo que contiene el conjunto de datos. Se puede dar clic en la opción de entrenar modelos, en dónde se ejecutará los 3 modelos de Machine Learning: Regresión Logística, Redes Neuronales, Árbol de decisión.

Figura 12.

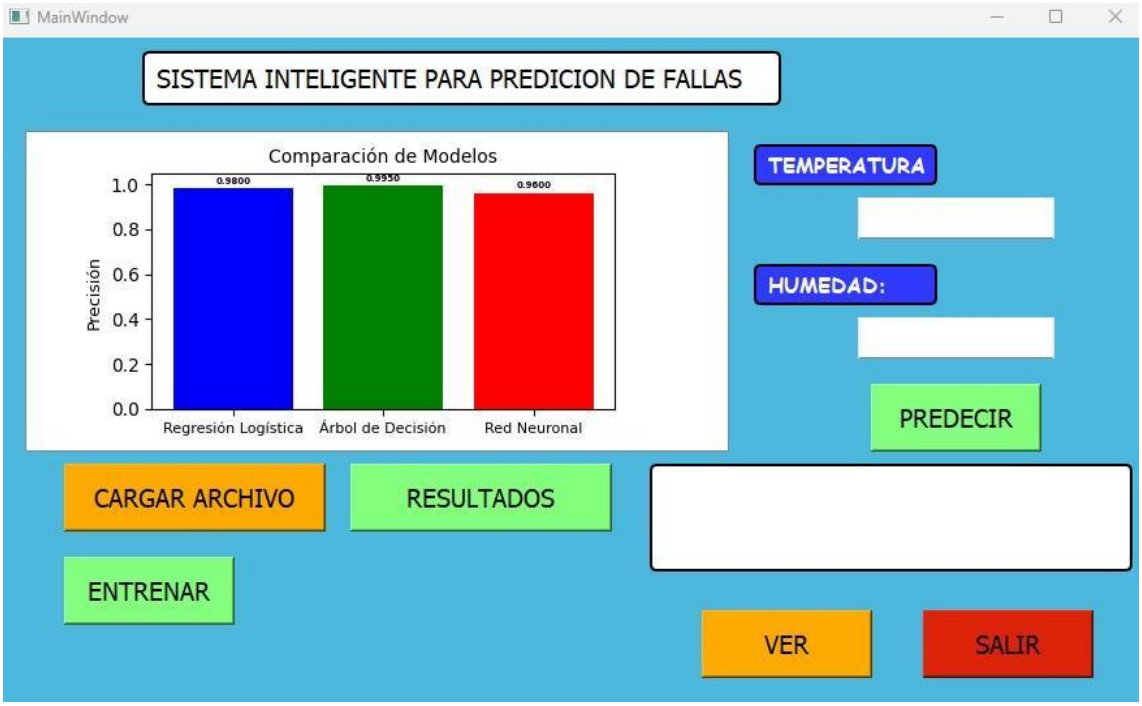
Ventana de Predicciones



Al dar clic en Entrenar Modelos se mostrarán los resultados de los modelos implementados, así como la gráfica comparativa de los 3 modelos implementados. Esta comparativa servirá para realizar la selección del mejor modelo de predicción (ver Figura 13).

Figura 13.

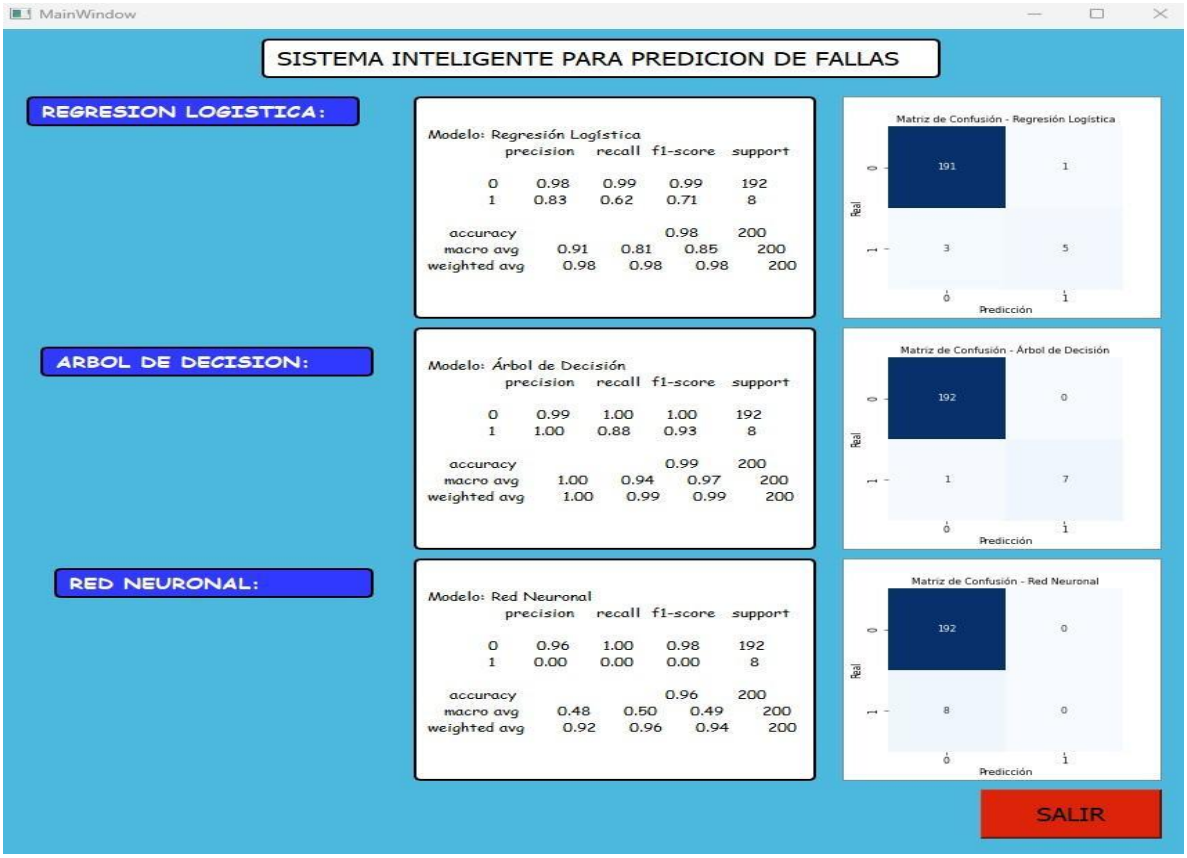
Ventana de Predicciones – Comparación de Modelos



Los modelos se evalúan en función de varias métricas y se selecciona el mejor modelo. Se visualizan los resultados de los modelos de acuerdo con la Figura 14.

Figura 14.

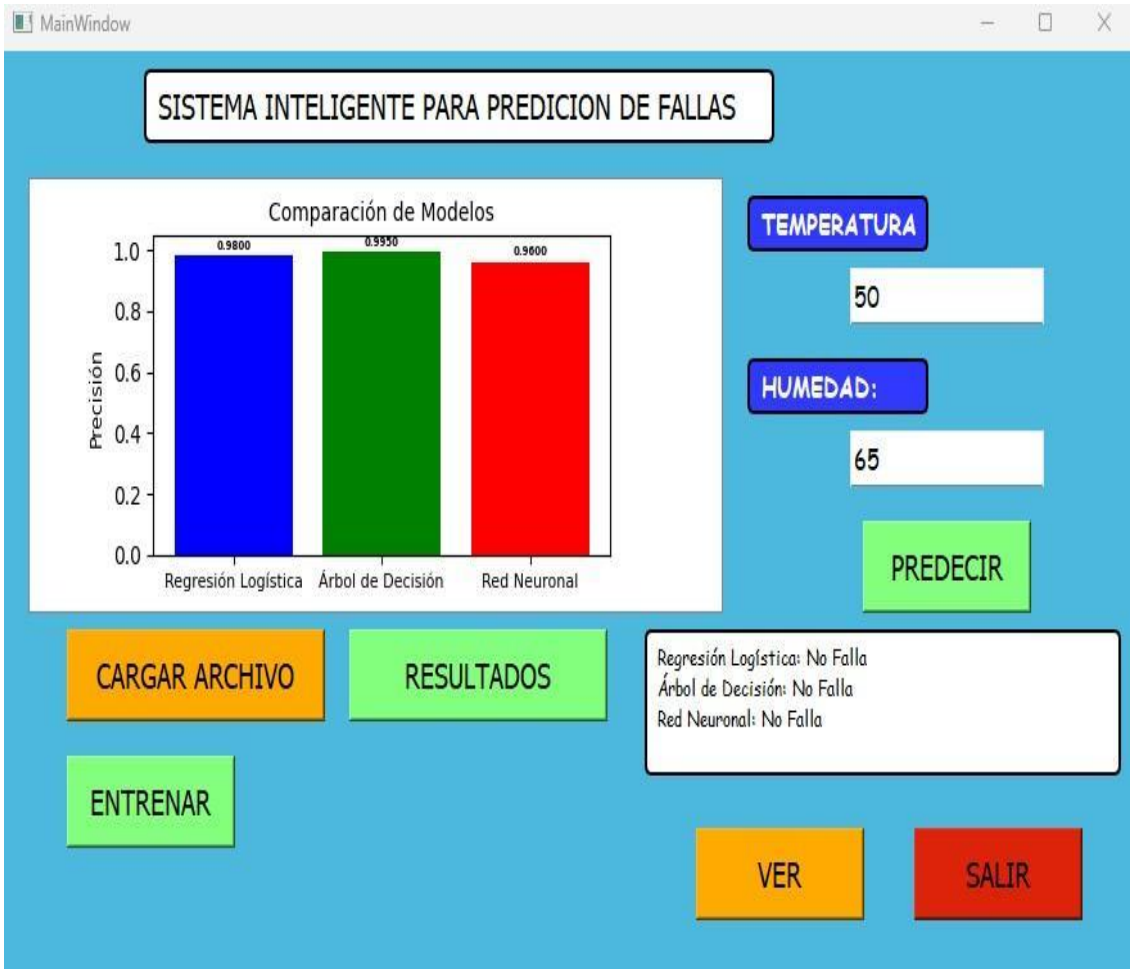
Ventana de Predicciones – Resultados de Modelos



Después se puede ingresar un valor de temperatura y otro de humedad y realizar la predicción utilizando el mejor modelo de machine learning (ver Figura 15).

Figura 15.

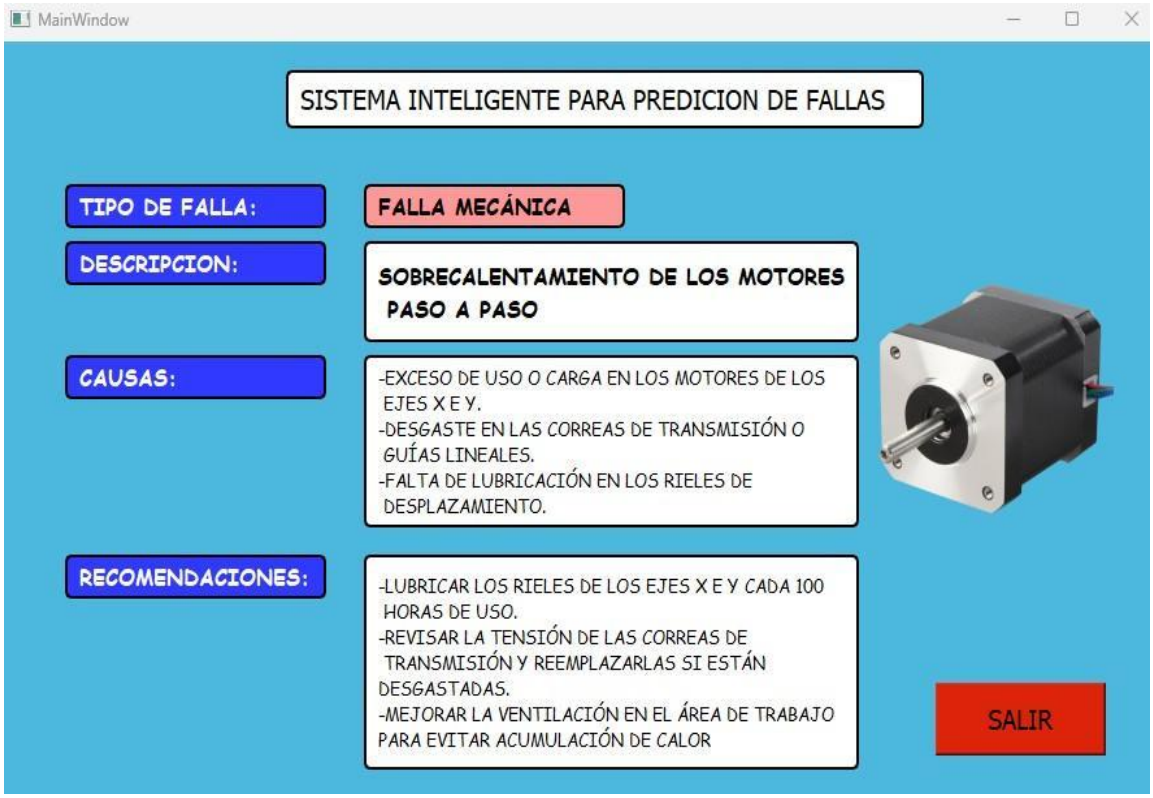
Ventana de Predicciones – Realizar Predicción



Una vez realizadas las predicciones se indica el tipo de falla y se recomienda las actividades de mantenimiento de la máquina (ver Figura 16).

Figura 16.

Ventana de Predicciones – Actividades de mantenimiento



- **Algoritmos de aprendizaje automático**: Se emplean Regresión Logística, Árboles de Decisión y Redes Neuronales para el análisis y predicción de fallas con base en datos simulados.

La generación de datos se llevó a cabo para la implementación de los algoritmos de aprendizaje automático. En esta fase inicial, mediante un estudio del funcionamiento de las cortadoras láser, se estableció que poseen sensores para registrar la temperatura de los motores y ventiladores, así como sensores que registran la humedad ambiental, que puede influir en la óptica y electrónica.

Considerando estos sensores se investigaron las fallas comunes que se presentan en la máquina asociado al comportamiento de la temperatura y humedad.

Se considera la falla de Sobrecalentamiento de los motores paso a paso a esto lo consideraremos como una falla mecánica. Las causas de esta falla son:

- Exceso de uso o carga en los motores de los ejes X e Y.

- Desgaste en las correas de transmisión o guías lineales.
- Falta de lubricación en los rieles de desplazamiento.

Se indica cómo detectan la falla mecánica los algoritmos de inteligencia artificial:

- *Regresión Logística:* Detecta si el aumento de temperatura está correlacionado con el tiempo de operación.
- *Árbol de Decisión:* Clasifica la falla según la temperatura límite de operación.
- *Red Neuronal:* Aprende patrones complejos de calentamiento y predice fallas antes de que ocurran.

Se considera también la falla de Condensación en los Espejos Ópticos a esta falla se le identificará como Falla Óptica. Las causas de esta falla son:

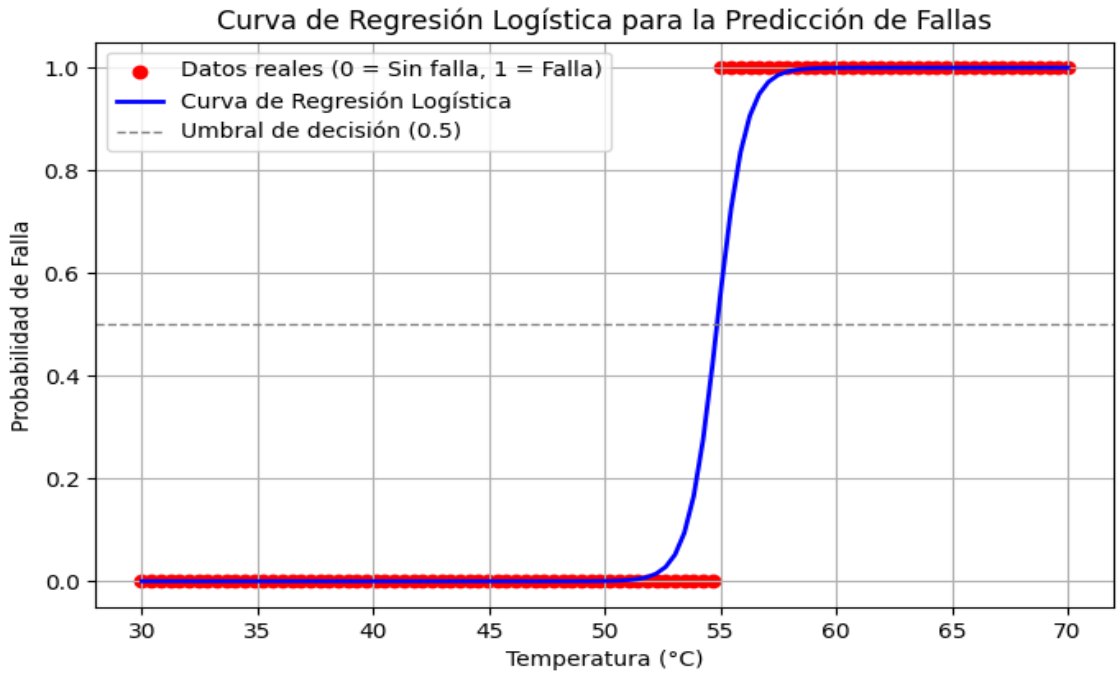
- Alta humedad ambiental (> 70%) combinada con una diferencia de temperatura en la óptica.
- Mal funcionamiento del sistema de extracción de aire.
- Uso prolongado sin pausas para enfriamiento.

Se indica cómo detectan la falla óptica los algoritmos de inteligencia artificial:

- Regresión Logística: Identifica si la humedad elevada afecta la calidad del corte.

Figura 17.

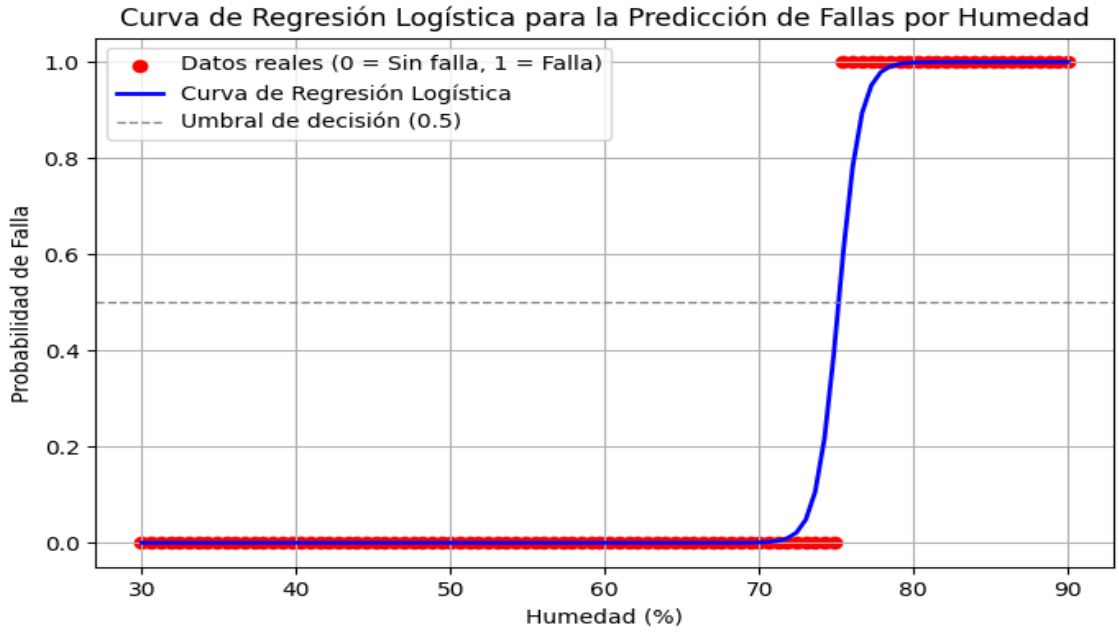
Curva Regresión Logística de la Temperatura



En el caso de la temperatura se observa en la Figura 17 que, a temperaturas inferiores a 55°C, la probabilidad de falla es prácticamente 0. A medida que la temperatura aumenta y supera los 55°C, la probabilidad de falla crece rápidamente hasta alcanzar un 100%.

Figura 18.

Curva Regresión Logística de la Humedad

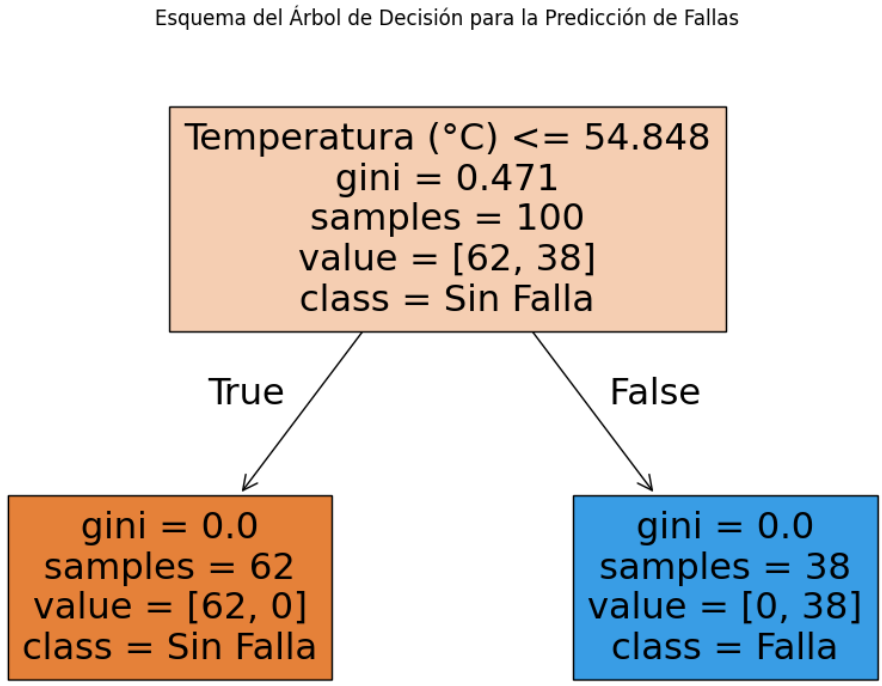


En el caso de la humedad en la Figura 18 se observa que con niveles de humedad inferiores al 75%, la probabilidad de falla es cercana a 0. A medida que la humedad supera el 75%, la probabilidad de falla aumenta bruscamente hasta alcanzar un 100%. La línea azul muestra la transición entre operación normal y fallo, evidenciando que altos niveles de humedad están asociados a fallos en la cortadora láser.

- *Árbol de Decisión:* Clasifica las condiciones críticas de temperatura y humedad.

Figura 19.

Esquema árbol de decisión temperatura



De acuerdo con la Figura 19:

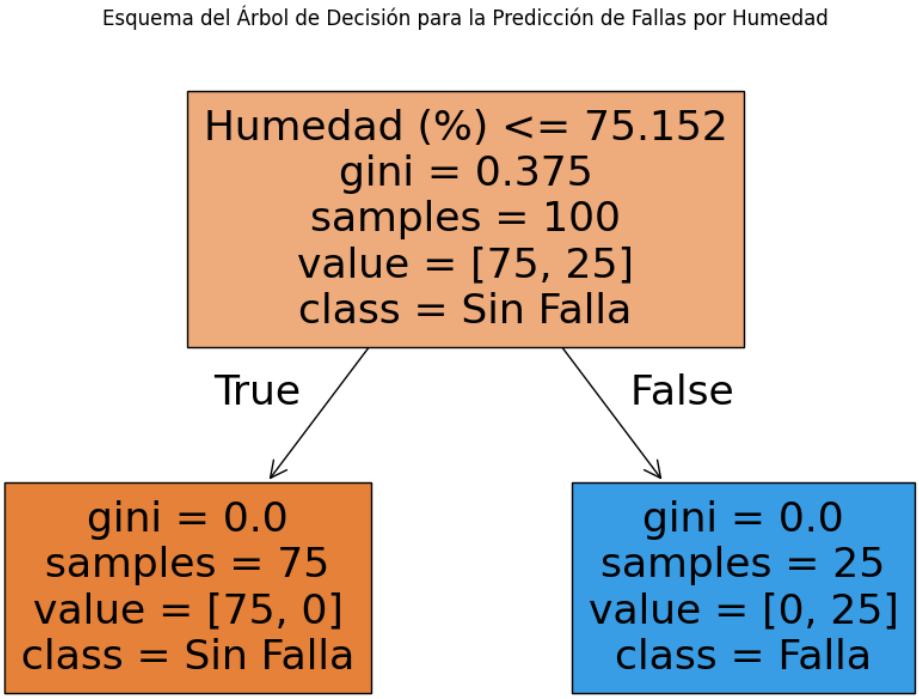
- *Criterio de decisión principal: La condición crítica en este árbol es si la temperatura es mayor a 54.848°C.*
- *Si la temperatura es menor o igual a 54.848°C, el sistema clasifica la muestra como "Sin Falla".*
- *Si la temperatura es mayor a 54.848°C, se clasifica como "Falla", lo que indica*

sobrecalentamiento en los motores paso a paso.

- *El índice de Gini en el nodo raíz es 0.471, lo que indica que la separación entre clases no es totalmente clara antes de la primera división.*
- *Los nodos hoja tienen Gini = 0.0, lo que significa que las muestras en estos nodos se clasifican sin ambigüedad.*

Figura 20.

Esquema árbol de decisión de la humedad



De acuerdo con la Figura 20

- Criterio de decisión principal: El nodo raíz indica que la condición principal para determinar si hay una falla es si la humedad es mayor a 75.152%.
- Si la humedad es menor o igual a 75.152%, el sistema clasifica la muestra como "Sin Falla".
- Si la humedad es mayor a 75.152%, la muestra es clasificada como "Falla", indicando que existe riesgo de condensación en los espejos ópticos.

- Gini = 0.375 en el nodo raíz, lo que indica que hay cierta mezcla de clases (fallas y no fallas) antes de la primera decisión.
- En los nodos hoja, el índice de Gini es 0.0, lo que significa que la clasificación en estos puntos es completamente pura (todas las muestras son de una sola clase).
- *Red Neuronal:* Aprende a predecir cuándo la condensación comienza a afectar la transmisión del láser.

También se analiza la falla del ventilador de enfriamiento a esto se considerará como Falla Electrónica. Las causas de esta falla son:

- Acumulación de polvo y residuos en los ventiladores del tubo láser y la fuente de alimentación.
- Desgaste en los rodamientos del motor del ventilador.
- Sobrecalentamiento de la fuente de alimentación por mala disipación térmica.

Las condiciones para detección de esta falla por parte de la inteligencia artificial son:

- *Regresión Logística:* Analiza si la temperatura del sistema de enfriamiento está aumentando con el tiempo.
- *Árbol de Decisión:* Detecta patrones de sobrecalentamiento y clasifica el estado del ventilador.
- *Red Neuronal:* Predice fallas antes de que la temperatura llegue a niveles críticos.

Se generan los datos de temperatura y humedad como se observa en la Figura 21.

Figura 21.
Librerías para generación de datos

```python
# Reimportar librerías después del reinicio del estado
import numpy as np
import pandas as pd
from IPython.display import display
import os


# Configurar semilla para reproducibilidad
np.random.seed(42)

# Número de muestras simuladas
n_samples = 1000

# Simulación de temperatura (°C) en motores, espejos y fuente de alimentación
temperatura = np.random.normal(loc=45, scale=5, size=n_samples)  # Promedio 45°C, desviación 5°C

# Simulación de humedad relativa (%) en el ambiente de trabajo
humedad = np.random.normal(loc=55, scale=10, size=n_samples)  # Promedio 55%, desviación 10%
```

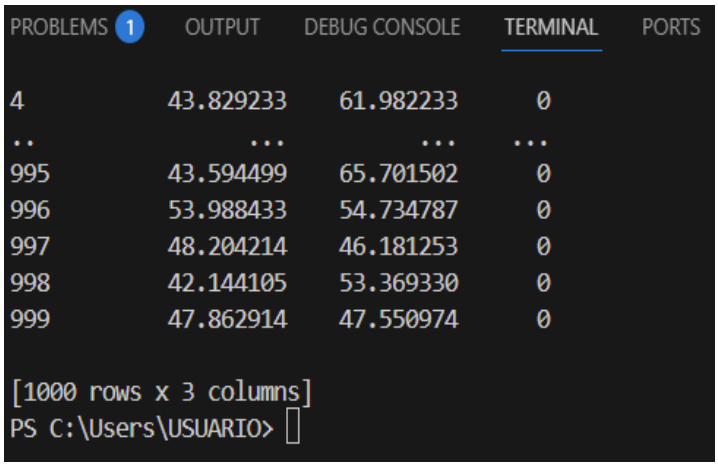
Se definen las condiciones para las fallas críticas (ver Figura 22).

Figura 22.
Simulación de fallas críticas

```python
# Simulación de fallas según condiciones críticas
# 1. Sobrecalentamiento de motores (> 60°C)
# 2. Condensación en espejos (> 65% humedad y temperatura > 50°C)
# 3. Fallo del ventilador de enfriamiento (> 55°C)

falla = ((temperatura > 60) | ((humedad > 65) & (temperatura > 50)) | (temperatura > 55)).astype(int)

# Crear DataFrame con los datos simulados
df_simulado = pd.DataFrame({
    'Temperatura (°C)': temperatura,
    'Humedad (%)': humedad,
    'Falla': falla
})

# Mostrar el DataFrame generado
display(df_simulado)

# Guardar el DataFrame en un archivo Excel llamado "datos_simulados.xlsx"
file_path = os.path.join(r
# index=False evita que se guarde el índice del DataFrame en el archivo
df_simulado.to_excel(file_path, index=False)
```

Los datos se imprimen en el terminal del software como se observa en la (Figura 23).

Figura 23.
Impresión de datos generados



Y se exportan los datos a un archivo Excel como se muestra en la (Figura 24).

Figura 24.
Excel con los datos generados

	A	B	C
1	**Temperatura (°C)**	**Humedad (%)**	**Falla**
2	47.48357077	68.99355437	0
3	44.30867849	64.24633683	0
4	48.23844269	55.5963037	0
5	52.61514928	48.53063222	0
6	43.82923313	61.98223314	0
7	43.82931522	58.93485385	0
8	52.89606408	63.9519322	0
9	48.83717365	61.35171802	0
10	42.65262807	65.49552715	0
11	47.71280022	49.64764788	0
12	42.68291154	68.17394066	0
13	42.67135123	56.97599605	0
14	46.20981136	75.75260873	0
15	35.43359878	48.10812182	0
16	36.37541084	72.35963803	0
17	42.18856235	56.97910783	0
18	39.9358444	48.48581996	0
19	46.57123666	50.16114166	0
20	40.45987962	51.79652692	0
21	37.93848149	59.24165946	0
22	52.32824384	60.22835488	0
23	43.8711185	49.26299996	0
24	45.33764102	54.75645408	0
25	37.87625907	76.42270359	0
26	42.27808638	72.2754317	0
27	45.55461295	59.3632367	0

Se cargan los datos de temperatura y humedad, y se aplican los modelos: Regresión Logística, Árbol de Decisión, Red Neuronal Perceptrón Multicapa (ver Figura 25).

Figura 25.
Modelos de Inteligencia Artificial

```
111     def load_file():
112         global data
113         file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
114         if file_path:
115             data = pd.read_csv(file_path)
116             messagebox.showinfo("Carga Completa", "Archivo cargado exitosamente")
117 
118     def train_models():
119         global models
120         X = data[['Temperatura (°C)', 'Humedad (%)']]
121         y = data['Falla']
122         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
123 
124         models = {
125             'Regresión Logística': LogisticRegression(),
126             'Árbol de Decisión': DecisionTreeClassifier(),
127             'Red Neuronal': MLPClassifier()
128         }
```

Se realiza el entrenamiento de los modelos como se muestra en la (Figura 26).

Figura 26.
Entrenamiento de los modelos

```
130         results = ""
131         ax.clear()
132         for name, model in models.items():
133             model.fit(X_train, y_train)
134             y_pred = model.predict(X_test)
135             acc = accuracy_score(y_test, y_pred)
136             report = classification_report(y_test, y_pred)
137             results += f"{name} - Precisión: {acc:.4f}\n{report}\n"
138 
139             ax.bar(name, acc)
140 
141         results_label.config(text=results)
```

c. Estrategias y/o técnicas

Se utilizaron estrategias metodológicas para construir el sistema de mantenimiento predictivo, fundamentadas en el aprendizaje supervisado y la validación experimental. Se utilizó una metodología iterativa en la cual los modelos de inteligencia artificial fueron ajustados progresivamente a partir del análisis de datos simulados. Se aplicó una técnica de prueba y error al entrenar los modelos para mejorar su precisión en la predicción de fallas.

Las herramientas tecnológicas utilizadas incorporan bibliotecas avanzadas de procesamiento de datos y modelado de inteligencia artificial. Se utilizó Python debido a su flexibilidad con un amplio ecosistema de bibliotecas especializadas. Para la gestión de datos se emplearon Pandas y NumPy, mientras que para la implementación de modelos de aprendizaje automático se utilizaron Scikit-learn y TensorFlow/Keras. La interfaz gráfica fue desarrollada con Tkinter o PyQt, permitiendo la visualización de los resultados en una plataforma intuitiva. Estas herramientas fueron seleccionadas por su eficiencia en el manejo de datos y su facilidad para la integración con sistemas de monitoreo en entornos reales.

2.3 Validación de la propuesta

En lo teórico, se llevó a cabo una revisión sistemática de la bibliografía en cuanto al mantenimiento predictivo, la inteligencia artificial y las aplicaciones industriales del aprendizaje automático. Se diseñaron modelos de regresión logística, redes neuronales artificiales y árboles de decisión, que han demostrado ser útiles para identificar fallos en diversos sectores industriales.

Tabla 2.

Descripción de perfil de validadores

N°.	Nombres y Apellidos	Años de experiencia	Titulación Académica	Cargo
1	EDISSON IVAN ALDAS SERRANO	8	MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACION INDUSTRIAL	SUPERVISOR DE INSTRUMENTACION Y CONTROL – POLIDUCTO TRES BOCAS - PASCUELAS-CUENCA DE EP PETROECUADOR.
2				
3				

Tabla 3.

Escala de evaluación del validador 1

CRITERIOS	EVALUACION SEGUN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto				✓	
Aplicabilidad				✓	
Conceptualización				✓	
Actualidad				✓	
Calidad Técnica				✓	
Factibilidad				✓	
Pertinencia				✓	

Tabla 4.

Escala de evaluación del validador 2

CRITERIOS	EVALUACION SEGUN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto					
Aplicabilidad					
Conceptualización					
Actualidad					
Calidad Técnica					
Factibilidad					
Pertinencia					

Tabla 5.

Escala de evaluación del validador 3

CRITERIOS	EVALUACION SEGUN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto					
Aplicabilidad					
Conceptualización					
Actualidad					
Calidad Técnica					
Factibilidad					
Pertinencia					

2.4 Matriz de articulación de la propuesta

La matriz condensa a la estructuración de un producto efectuado mediante los fundamentos teóricos, metodológicos, estratégicos-técnicos, tecnológicos utilizados.

Tabla 3.

Matriz de articulación

Ejes o partes principales del proyecto		Breve descripción de los resultados de cada parte	Sustento teórico que se aplicó en la construcción del proyecto	Metodologías, herramientas técnicas y tecnológicas que se emplearon
1	Definición: Variables y elementos del sistema, modelos de IA, procesamiento de datos simulados	1.1. Identificación de las variables críticas: temperatura y humedad. 1.2. Simulación de datos operativos en entornos industriales.	Inteligencia Artificial Machine Learning	Python (Pandas, Numpy) para procesamiento de datos.
2	Diseño: Estructura del sistema de mantenimiento predictivo.	2.1 Interfaz de usuario en Python 2.2. Diseño de algoritmos de Inteligencia Artificial: Regresión Logística, Árbol de Decisión, Redes Neuronales. 2.3 Selección de la mejor estrategia de predicción basada en métricas de rendimiento.	Aplicación de diseño y programación de interfaz de usuario Python.	Tkinter biblioteca de Python para generar la interfaz de usuario Sklearn para implementación de algoritmos de inteligencia artificial en Python.

3	Implementación: Desarrollo de la interfaz gráfica y ejecución del sistema.	3.1 Creación de la GUI para visualización de predicciones y monitoreo de datos. 3.2. Integración de gráficos y reportes para la interpretación de fallas.	Diseño de interfaces de usuario Desarrollo de software	Tkinter biblioteca de Python para generar la interfaz de usuario.
4	Evaluación y monitoreo: Validación del sistema mediante simulaciones y reportes	4.1 Validación del rendimiento de los modelos mediante métricas de precisión, recall y F1-score	Métodos de validación Evaluación de Modelos Predictivos Análisis de Resultados	Visualización de métricas en la interfaz gráfica creada con Python. Visualización de actividades de mantenimiento sugeridas en Python.

2.5 Análisis de resultados. Presentación y discusión.

Esta evaluación del sistema se realizará mediante la medición de las métricas clave de rendimiento de los modelos de IA.

Para evaluar cada modelo de Machine Learning se utiliza la matriz de confusión. Esta es una herramienta para evaluar el rendimiento de los modelos de clasificación en aprendizaje automático. Representa el número de predicciones correctas e incorrectas realizadas por un modelo en comparación con los valores reales de la clase objetivo. En este caso, se evaluaron tres modelos distintos para la predicción de fallas en una cortadora láser: Regresión Logística, Árbol de Decisión y Red Neuronal.

Cada matriz tiene cuatro valores principales como se muestra en la Tabla 4**.**

Tabla 4.

Valores de la matriz de confusión

Clase Real	Predicción: No Falla (0)	Predicción: Falla (1)
No Falla (0)	Verdaderos Negativos (TN)	Falsos Positivos (FP)
Falla (1)	Falsos Negativos (FN)	Verdaderos Positivos (TP)

- TN (True Negative): Casos correctamente clasificados como "Sin falla".
- TP (True Positive): Casos correctamente clasificados como "Falla".
- FP (False Positive): Casos donde el modelo predijo erróneamente una falla cuando no existía.
- FN (False Negative): Casos donde el modelo no detectó una falla real.

En la Figura 27 a Figura 29 se muestra la matriz de confusión de los algoritmos implementados.

Figura 27.

Matriz de Confusión – Regresión Logística

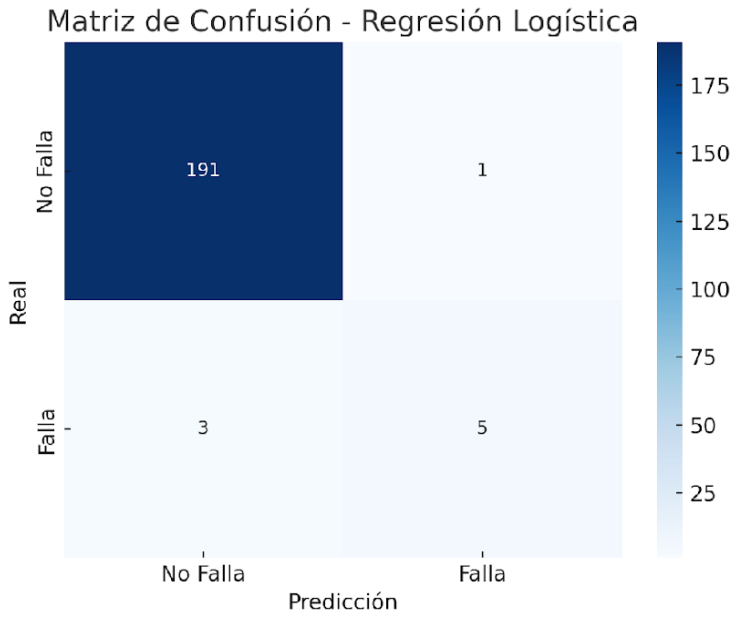


Figura 28.

Matriz de Confusión – Árbol de decisión

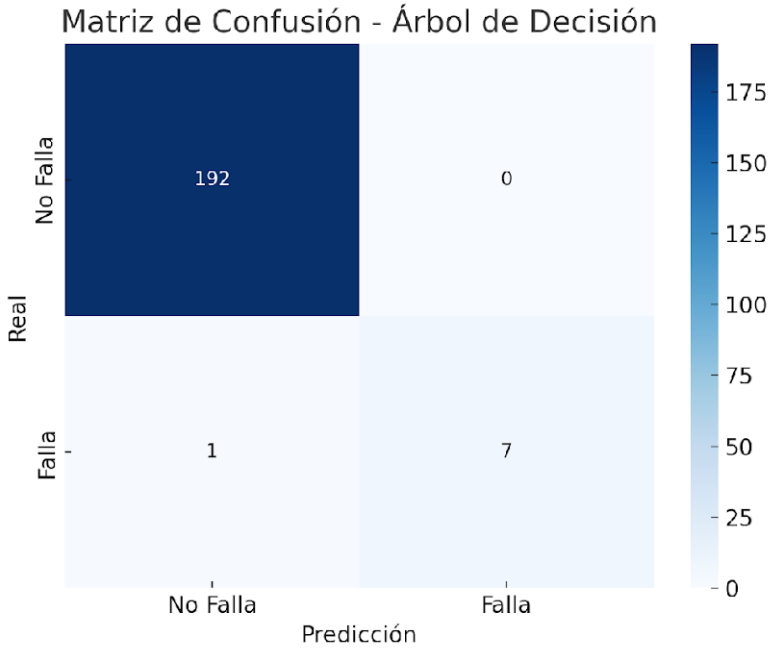
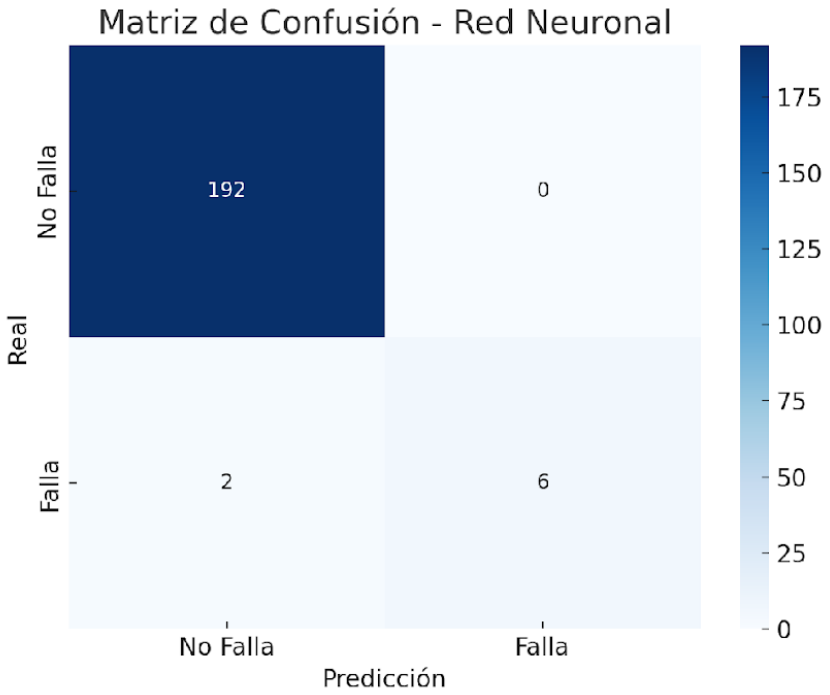


Figura 29.
Matriz de Confusión – Red Neuronal



Al evaluar los algoritmos de inteligencia artificial se obtienen los siguientes resultados:

- Regresión Logística
  - TN: 191 (correctamente identificados como "No Falla").
  - TP: 5 (correctamente identificados como "Falla").
  - FP: 1 (caso donde predijo falla, pero no existía).
  - FN: 3 (fallas reales que no fueron detectadas).
  - Análisis: El modelo tuvo un alto desempeño general, pero cometió 3 errores importantes al no detectar fallas reales (FN), lo que puede representar un riesgo en aplicaciones críticas.
- Árbol de Decisión
  - TN: 192

- TP: 7
- FP: 0 (ninguna predicción errónea de falla).
- FN: 1 (una falla real no detectada).
- Análisis: Este modelo fue el más preciso en la clasificación, con cero falsos positivos y solo una falla no detectada, lo que lo hace el modelo con mejor desempeño.
- Red Neuronal
- TN: 192
- TP: 6
- FP: 0
- FN: 2
- Análisis: También mostró alto rendimiento, con cero falsos positivos, pero con dos fallas no detectadas, lo que lo coloca ligeramente detrás del Árbol de Decisión en efectividad.

En este análisis se obtuvo mejor desempeño con el algoritmo de Árbol de Decisión, porque tiene la menor cantidad de errores en detección de fallas. La Regresión Logística también funcionó bien, pero tuvo más falsos negativos, lo que podría impactar en la confiabilidad del sistema. La Red Neuronal tuvo un buen desempeño, pero quedó ligeramente por debajo del Árbol de Decisión en la detección de fallas.

Se aplicarán métricas como precisión, recall y F1-score para determinar la efectividad del sistema en la detección predictiva de fallas.

Precisión:

- Es el porcentaje de predicciones positivas correctas sobre todas las predicciones positivas realizadas.

- Interpretación: Indica cuán preciso es el modelo al clasificar una instancia como positiva (minimización de falsos positivos).
- Rango General: 0 a 1 (0% a 100%)
    - Aceptable: Una precisión por encima de 70% se considera generalmente buena, pero en situaciones donde los falsos positivos son costosos (por ejemplo, cuando un resultado positivo inicia un tratamiento médico), se podría exigir una precisión superior a 90%.
    - Consideraciones: Alta precisión es especialmente importante en contextos donde los falsos positivos tienen consecuencias graves.

Recall (Sensibilidad o Tasa de Verdaderos Positivos):

- Es el porcentaje de verdaderos positivos correctamente identificados sobre el total de positivos reales.
- Interpretación: Indica qué tan bien el modelo captura todos los positivos reales (minimización de falsos negativos).
- Rango General: 0 a 1 (0% a 100%)
    - Aceptable: Un recall por encima de 70% es generalmente bueno. En situaciones donde es crucial capturar todos los casos positivos, un recall de 80%-90% o más es deseable.
    - Consideraciones: Un alto recall es crucial en situaciones donde los falsos negativos son inaceptables, como en el diagnóstico de enfermedades graves.

F1 Score:

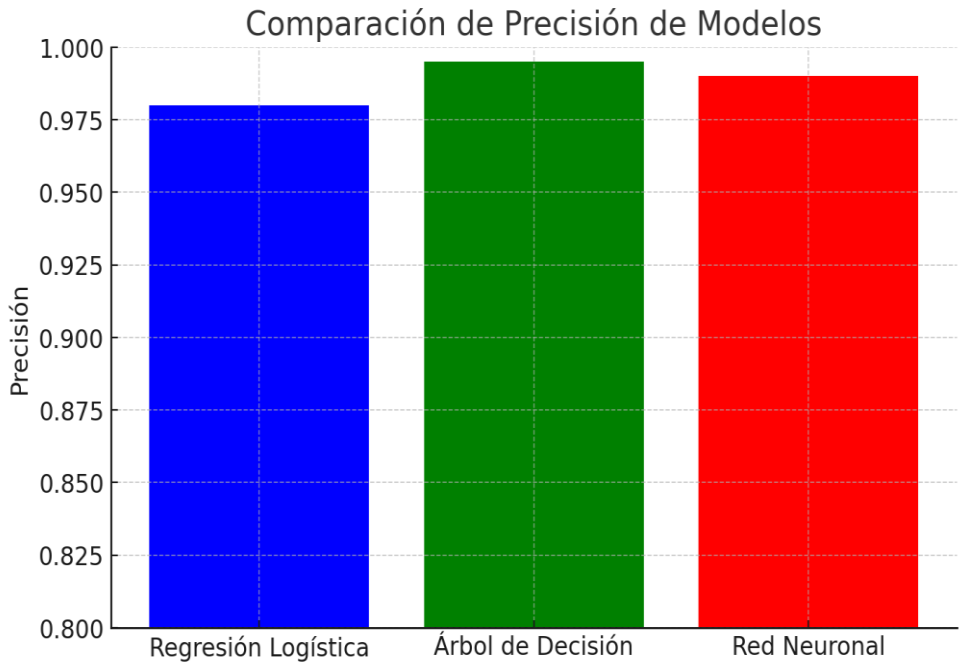
- Es la media armónica de la precisión y el recall, proporcionando un balance entre ambas métricas.

- Interpretación: Es útil cuando se necesita un balance entre precisión y recall, especialmente cuando las clases están desbalanceadas.
- Rango General: 0 a 1 (0% a 100%)
    - Aceptable: Un F1 Score superior a 0.7 es generalmente considerado bueno. Un valor cercano a 1 indica un buen balance entre precisión y recall.
    - Consideraciones: F1 Score es particularmente útil cuando hay un desbalance entre clases y se necesita considerar tanto la precisión como el recall.

Se grafica la comparación de los modelos como se muestra en la Figura 30.

Figura 30.

Comparación de los modelos



Para comparar los modelos se obtuvo las métricas de precisión, recall y f1-score que se muestran en la Tabla 5.

Tabla 5.

Métricas Comparativas de los modelos

No.	Modelo	Precisión	Recall (Case 1)	F1-Score
1	Regresión Logística	0.98	0.625	0.71428571428
2	Árbol de Decisión	0.995	0.875	0.93333333333
3	Red Neuronal	0.99	0.75	0.85714285714

Se han implementado los modelos y se concluye que el modelo de Árbol de Decisión es el mejor modelo con alta precisión y detección de fallas. La Red Neuronal también tiene buen desempeño, pero su recall es menor. La Regresión Logística es menos efectiva para detectar fallas.

Se realizaron 10 pruebas para verificar la efectividad del modelo. En la Tabla 6 se muestran los resultados.

Tabla 6.

Pruebas de Predicción

No.	Temperatura (°C)	Humedad (%)	Tipo de Falla
1	58	50	Falla Electrónica (Fallo de ventilador)
2	62	45	Falla Mecánica (Sobrecalentamiento de Motores)
3	47	70	Sin Falla

4	55	60	Sin Falla
5	50	68	Sin Falla
6	65	40	Falla Mecánica (Sobrecalentamiento de Motores)
7	42	75	Sin Falla
8	48	65	Sin Falla
9	60	55	Falla Electrónica (Fallo del ventilador)
10	52	72	Falla Óptica (Condensación en Espejos)

En la Figura 31 se muestra la prueba realizada para un valor de temperatura de 60°C y una humedad de 65%, el tipo de falla detectada es la falla electrónica y al dar clic en VER se especifica las causas y las recomendaciones (ver Figura 32).

Figura 31.

Detección de Falla Electrónica

Figura 32.

Actividades de mantenimiento para falla electrónica

Se realizó la prueba de la falla mecánica al ingresar un valor de temperatura de 75°C y un valor de humedad del 81% (ver Figura 33Figura 32). En la se especifican las actividades recomendadas para este tipo de falla (ver Figura 34).

Figura 33.

Detección de falla mecánica

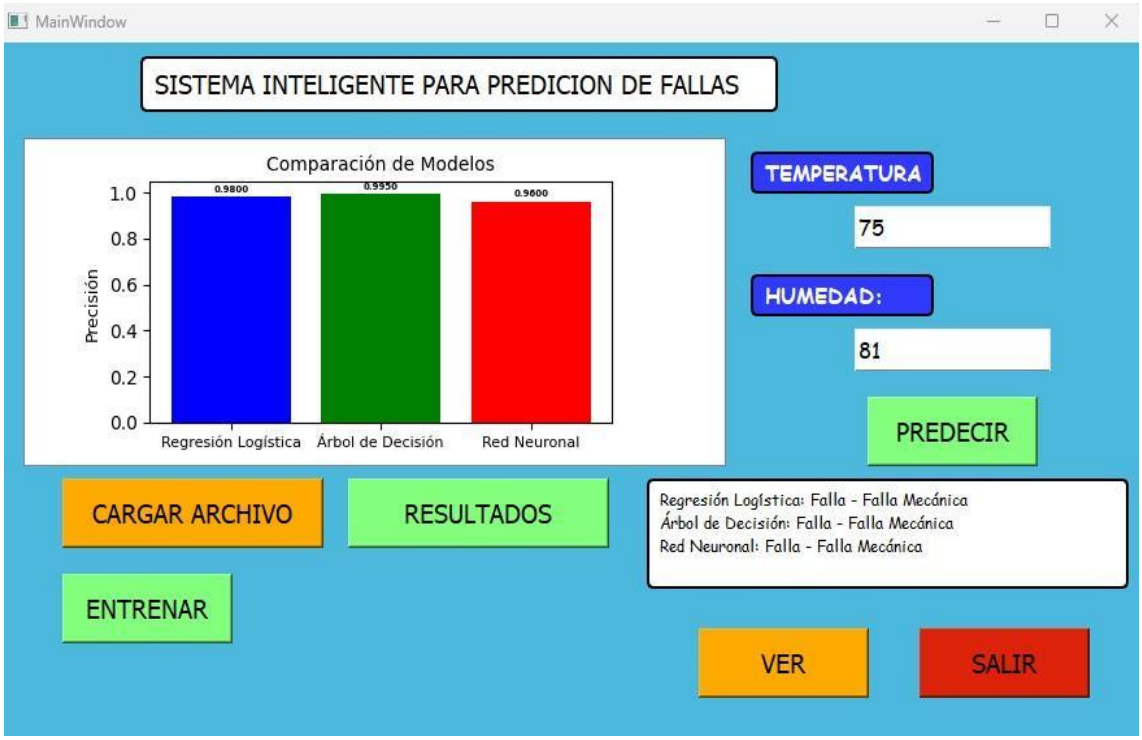
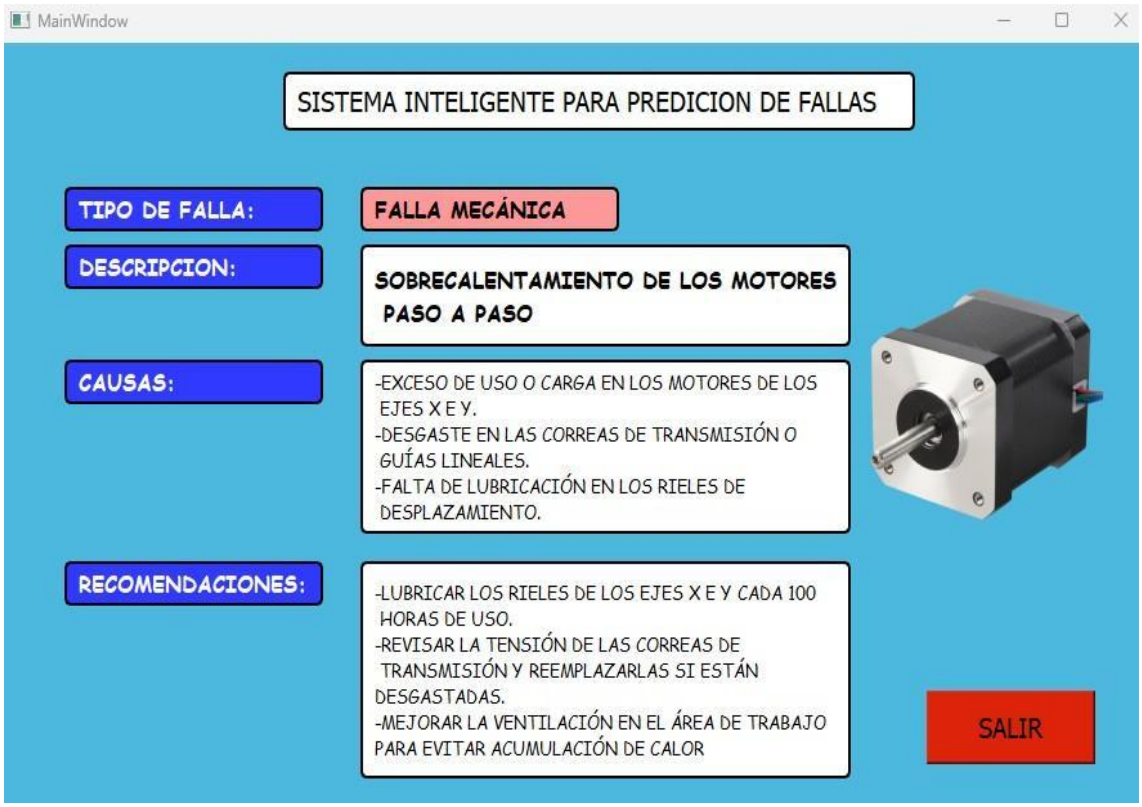


Figura 34.

Actividades para falla mecánica



Se realiza la prueba de falla óptica con un valor de temperatura de 55°C y una humedad 100% (ver Figura 35) y se especifican las actividades de mantenimiento para la falla óptica (ver

Figura **36**).

Figura 35.

Detección de falla óptica

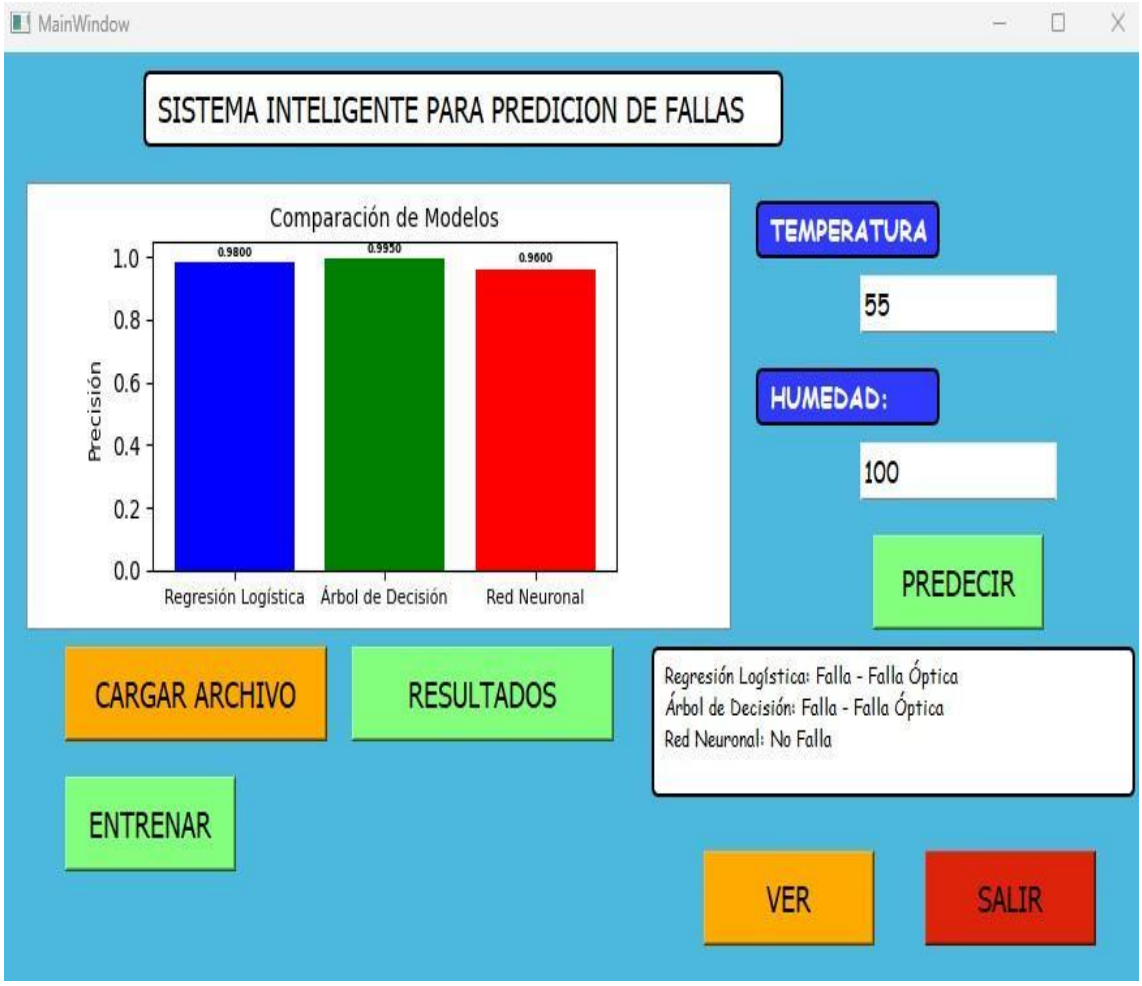


Figura 36.

Actividades para falla óptica


MainWindow
SISTEMA INTELIGENTE PARA PREDICION DE FALLAS
TIPO DE FALLA:
FALLA ÓPTICA
DESCRIPCION:
CONDENSACIÓN EN LOS ESPEJOS OPTICOS
CAUSAS:
-ALTA HUMEDAD AMBIENTAL (> 70%) COMBINADA CON UNA DIFERENCIA DE TEMPERATURA EN LA ÓPTICA.
-MAL FUNCIONAMIENTO DEL SISTEMA DE EXTRACCIÓN DE AIRE.
-USO PROLONGADO SIN PAUSAS PARA ENFRIAMIENTO.
RECOMENDACIONES:
-INSTALAR UN DESHUMIDIFICADOR EN LA SALA DE TRABAJO SI LA HUMEDAD SUPERA EL 65%.
-LIMPIAR LOS ESPEJOS Y LENTES CON ALCOHOL SOPROPÍLICO CADA 50 HORAS DE USO.
-REVISAR EL SISTEMA DE EXTRACCIÓN DE AIRE Y LIMPIARLOS FILTROS REGULARMENTE.
SALIR

CONCLUSIONES

Se realizó un robusto marco teórico acerca de la inteligencia artificial en el mantenimiento predictivo. La revisión bibliográfica posibilitó poner en contexto los progresos en los modelos de aprendizaje automático y su influencia en la industria de manufactura, demostrando que la incorporación de algoritmos de aprendizaje automático en procesos industriales puede mejorar la toma de decisiones y disminuir los periodos de parada.

Se desarrollaron tres modelos de inteligencia artificial para la predicción de fallos, evaluados con métricas de exactitud, recall y puntaje F1. El modelo de árbol de decisión obtuvo el mejor rendimiento, con una buena habilidad para predecir, con una tasa reducida de errores en la categorización de fallos.

Se realizó y comprobó una interfaz gráfica en Python que facilita a los usuarios la carga de datos, la observación de gráficos de temperatura y humedad, la funcionalidad de modelos de predicción con las alertas de las tareas de mantenimiento. El funcionamiento interactivo de la interfaz invita la implementación futura de esta tecnología en ambientes industriales, para la toma de decisiones.

Finalmente, la ejecución de la propuesta a través de ensayos en la interfaz corroboró la funcionalidad del sistema creado. La observación de los modelos evidenció que la inteligencia artificial es una herramienta útil para la identificación predictiva de fallos, con la posibilidad de ser utilizada en diversas clases de maquinaria industrial. No obstante, se reconoce que al no disponer de información tangible para incrementar la exactitud del sistema, esto abre una oportunidad de futuros estudios esta vez con sensores físicos en ambientes operativos reales.

RECOMENDACIONES

Los hallazgos observados en este proyecto, presentan las siguientes recomendaciones para futuras investigaciones, mejoras en la implementación del sistema de mantenimiento predictivo en el cortador láser. Mejoramiento del marco teórico en estudios comparativos, se recomienda analizar nuevas técnicas de inteligencia artificial como el aprendizaje profundo (Deep Learning) como modelos híbridos que combinen diferentes algoritmos para mejorar la detección de fallas. Optimización para mejora de los modelos de predicción, aunque el árbol de decisión mostró el mejor desempeño en la detección de fallas, se sugiere explorar técnicas de optimización de hiper parámetros y combinar modelos de machine learning con enfoques de análisis estadístico avanzado. También se recomienda aumentar el conjunto de datos de entrenamiento con información de fallas reales para mejorar la capacidad de generalización de los modelos. Implementación en un entorno real con sensores físicos, para mejorar la precisión del sistema, se sugiere la implementación de sensores físicos de temperatura y humedad en una cortadora láser real. La integración con dispositivos IoT permitiría el monitoreo en tiempo real proporcionando datos más confiables y optimizando el proceso de predicción de fallas mediante la combinación de datos reales y simulados. Mejoras en la interfaz gráfica y funcionalidad del sistema se recomienda expandir la interfaz gráfica desarrollada en Python para incluir funcionalidades avanzadas como alertas en tiempo real generación automática de reportes y compatibilidad con plataformas en la nube. Esto facilitaría su adopción en la industria y permitiría un acceso remoto eficiente para la supervisión de las condiciones operativas del equipo. Expansión del modelo a otros equipos industriales finalmente, se recomienda explorar la aplicación del sistema de mantenimiento predictivo en otras máquinas industriales, como robots de manufactura, fresadoras CNC y prensas hidráulicas, con el fin de evaluar su escalabilidad y adaptabilidad a diferentes procesos de producción. Estas recomendaciones buscan fortalecer la investigación en mantenimiento predictivo y facilitar la integración de inteligencia artificial en entornos industriales, promoviendo la mejora en los procesos en la reducción en el tiempo de inactividad en equipos críticos.

BIBLIOGRAFÍA

Dueñas-Ramírez, L. M., Villegas-López, G. A., Castiblanco-Tique, S., & Castaño-Restrepo, C. A. (2020). Casos de éxito en la implementación del mantenimiento predictivo mediante el uso de tecnologías de la industria 4.0 en empresas colombianas. In *Actas del Congreso Internacional de Ingeniería de Sistemas* (pp. 109-121).

Morphy, L. V. Inteligencia Artificial para Toma de Decisiones en el Mantenimiento Predictivo a equipos Turbocompresores en la Industria Petrolera.

Prellezo Tezanos, A. (2019). Utilización de técnicas de la Industria 4.0 en línea de montaje automatizada para la introducción de un plan de mantenimiento predictivo.

Hurtado-Cortés, J. E., Narváez-Moreno, J. A., & Lozano-Gutiérrez, A. (2016). Inteligencia artificial aplicada a la detección y diagnóstico de fallas en sistemas dinámicos. *Ingeniería*, 21(2), 186-207. https://doi.org/10.15446/dyna.v83n199.55612

Sánchez Cocha, J. V. (2024). Detección de fallas en un motor usando el método de machine learning. *Tesis de maestría, Universidad Israel (UISRAEL)*.

Reino Cárdenas, M. I. (2022). Sistema inteligente basado en redes neuronales y protocolo MQTT para predicción de desgaste en turbinas Pelton. *Tesis de maestría, Universidad Israel (UISRAEL)*.

Keewong Zapata, R., Kemper Valverde, N., Ruiz-Huerta, L., Caballero-Ruiz, A., Nava Sandoval, R., & Jiménez Carrión, M. (2023). Modelo neuronal para la predicción de la rugosidad superficial y optimización del proceso de corte por chorro de agua. *Libro de Ingeniería, Universidad Israel (UISRAEL).*

ANEXOS

ANEXO 1

CÓDIGO DEL ALGORITMO

```
import sys
from tkinter import filedialog, messagebox

import pandas as pd
import matplotlib.pyplot as plt
from PyQt5.QtGui import QPainter
from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QMessageBox, QLineEdit, QGraphicsScene
from PyQt5.uic import loadUi
from PyQt5.uic.properties import QtCore
from matplotlib.backends.backend_qt5agg import (FigureCanvasQTAgg as FigureCanvas)
from primera import Ui_MainWindow as Ui_Primera
from segunda import Ui_MainWindow as Ui_Segunda
from tercera import Ui_MainWindow as Ui_Tercera
from cuarta import Ui_MainWindow as Ui_Cuarta
from quinta import Ui_MainWindow as Ui_Quinta
from sexta import Ui_MainWindow as Ui_Sexta
from septima import Ui_MainWindow as Ui_Septima
from octava import Ui_MainWindow as Ui_Octava
from novena import Ui_MainWindow as Ui_Novena
from decima import Ui_MainWindow as Ui_Decima
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns


class VentanaPrincipal(QMainWindow):
```

```
29  class VentanaPrincipal(QMainWindow):
30      def __init__(self):
32          self.ui = Ui_Primera()
33          self.ui.setupUi(self)
34          self.ui.pushButton.clicked.connect(self.abrir_segunda_ventana)
35          self.ui.pushButton_2.clicked.connect(self.abrir_decima_ventana)
36          self.ui.pushButton_3.clicked.connect(self.salir_ventana)
37
38      def abrir_segunda_ventana(self):
39          self.ventana = SegundaVentana(self)  # Pasamos self para poder volver
40          self.ventana.show()
41          self.close()  # Cierra la ventana actual
42
43      def abrir_decima_ventana(self):
44          self.ventana = DecimaVentana(self)  # Pasamos self para poder volver
45          self.ventana.show()
46          self.close()  # Cierra la ventana actual
47
48      def salir_ventana(self):
49          self.close()  # Cierra la ventana actual
50
51
52  class SegundaVentana(QMainWindow):
53      def __init__(self,ventana_anterior):
54          super().__init__()
55          self.ui = Ui_Segunda()
56          self.ui.setupUi(self)
57          self.ventana_anterior = ventana_anterior  # Guardamos referencia a la ventana anterior
58          self.ui.pushButton_3.clicked.connect(self.volver_atras)  # Botón de "Atrás"
59          self.ui.pushButton.clicked.connect(self.abrir_tercera)  # Botón de "Adelante"
60          # Configurar line edit de contraseña para ocultar caracteres
61          self.ui.lineEdit_2.setEchoMode(QLineEdit.Password)
62
63      def volver_atras(self):
64          self.ventana_anterior.show()  # Muestra la ventana anterior
65          self.close()  # Cierra la ventana actual
66
67      def abrir_tercera(self):
68          usuario = self.ui.lineEdit.text()
69          contraseña = self.ui.lineEdit_2.text()
70
71          # Verificación de usuario y contraseña
72          if usuario == "admin" and contraseña == "1234":  # Cambia por tus credenciales reales
73              self.ventana = TerceraVentana(self)
74              self.ventana.show()
75              self.close()
76          else:
77              QMessageBox.warning(self, "Error", "Usuario o contraseña incorrectos")
78
79
80  class TerceraVentana(QMainWindow):
81      def __init__(self, ventana_anterior):
82          super().__init__()
83          self.ui = Ui_Tercera()
84          self.ui.setupUi(self)
```

```
        self.ventana_anterior = ventana_anterior
        self.ui.pushButton_3.clicked.connect(self.volver_atras)
        self.ui.pushButton.clicked.connect(self.abrir_cuarta_ventana)
        self.ui.pushButton_2.clicked.connect(self.abrir_quinta_ventana)

    def volver_atras(self):
        self.ventana_anterior.show()
        self.close()

    def abrir_cuarta_ventana(self):
        self.ventana = CuartaVentana(self)  # Pasamos self para poder volver
        self.ventana.show()
        self.close()  # Cierra la ventana ac

    def abrir_quinta_ventana(self):
        self.ventana = QuintaVentana(self)  # Pasamos self para poder volver
        self.ventana.show()
        self.close()  # Cierra la ventana ac

class CuartaVentana(QMainWindow):
    def __init__(self, ventana_anterior):
        super().__init__()
        self.ui = Ui_Cuarta()
        self.ui.setupUi(self)
        self.ventana_anterior = ventana_anterior
        self.df = None  # Para almacenar los datos del Excel
        self.ui.pushButton_2.clicked.connect(self.cargar_excel)
```

```
        self.ui.pushButton_2.clicked.connect(self.cargar_excel)
        self.ui.pushButton.clicked.connect(self.graficar)
        self.ui.pushButton_3.clicked.connect(self.volver_atras)  # Botón para volver

    def volver_atras(self):
        self.ventana_anterior.show()
        self.close()

    def cargar_excel(self):
        """ Permite seleccionar y cargar un archivo Excel """
        ruta, _ = QFileDialog.getOpenFileName(self, "Seleccionar archivo CSV", "", "Archivos CSV (*.csv)")
        if ruta:
            try:
                self.df = pd.read_csv(ruta)
                self.ui.comboBox.clear()
                self.ui.comboBox.addItems(self.df.columns)  # Llena el combo con los nombres de las columnas
                QMessageBox.information(self, "Éxito", "Archivo cargado correctamente.")
            except Exception as e:
                QMessageBox.critical(self, "Error", f"No se pudo cargar el archivo:\n{e}")

    def graficar(self):
        """ Grafica la variable seleccionada del ComboBox """
        if self.df is None:
            QMessageBox.warning(self, "Error", "No se ha cargado un archivo CSV.")
            return

        # Obtener la columna seleccionada en el ComboBox
```

```
        columna = self.ui.comboBox.currentText()

        if columna not in self.df.columns:
            QMessageBox.warning(self, "Error", f"La columna {columna} no existe en el archivo.")
            return

        # Crear una figura y un canvas para el gráfico
        fig, ax = plt.subplots(figsize=(4, 2))
        ax.plot(self.df[columna])  # Graficamos la columna seleccionada
        ax.set_title(f'Gráfico de {columna}')
        ax.set_xlabel('Índice')
        ax.set_ylabel(columna)

        # Calcular estadísticas principales
        media = self.df[columna].mean()
        desviacion_estandar = self.df[columna].std()
        minimo = self.df[columna].min()
        maximo = self.df[columna].max()
        rango = maximo - minimo
        mediana = self.df[columna].median()

        # Mostrar las estadísticas en el QLabel
        stats_text = (
            f"Media: {media:.2f}\n"
            f"Desviación Estándar: {desviacion_estandar:.2f}\n"
            f"Valor Mínimo: {minimo:.2f}\n"
            f"Media: {media:.2f}\n"
            f"Desviación Estándar: {desviacion_estandar:.2f}\n"
            f"Valor Mínimo: {minimo:.2f}\n"
            f"Valor Máximo: {maximo:.2f}\n"
            f"Rango: {rango:.2f}\n"
            f"Mediana: {mediana:.2f}"
        )
        self.ui.label_2.setText(stats_text)  # Aseg

        # Integrar el gráfico en el QGraphicsView
        canvas = FigureCanvas(fig)
        scene = QGraphicsScene(self)  # Creamos una escena
        scene.addWidget(canvas)  # Añadimos el canvas a la escena
        self.ui.graphicsView.setScene(scene)  # Establecemos la escena en el QGraphicsView

        canvas.draw()  # Dibujamos el gráfico

class QuintaVentana(QMainWindow):
    def __init__(self, ventana_anterior):
        super().__init__()
        self.ui = Ui_Quinta()
        self.ui.setupUi(self)
        self.ventana_anterior = ventana_anterior
        self.df = None  # Para almacenar los datos del Excel
        self.ventana_falla = None  # Para la ventana de falla
        self.tipo_falla = None  # Almacenar la falla detectada
        self.estadisticas_modelos = {}
```

```
187             self.estadisticas_modelos = {}
188             self.ui.pushButton_2.clicked.connect(self.load_file)
189             self.ui.pushButton_3.clicked.connect(self.train_models)
190             self.ui.pushButton_4.clicked.connect(self.abrir_novena_ventana)
191             self.ui.pushButton_5.clicked.connect(self.volver_atras)
192             self.ui.pushButton_6.clicked.connect(self.abrir_ventana_falla)
193             self.ui.pushButton.clicked.connect(self.predict_value)
194
195
196         def volver_atras(self):
197             self.ventana_anterior.show()  # Muestra la ventana anterior
198             self.close()  # Cierra la ventana actual
199
200         def load_file(self):
201             global data
202             file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
203             if file_path:
204                 data = pd.read_csv(file_path)
205                 messagebox.showinfo("Carga Completa", "Archivo cargado exitosamente")
206
207         def train_models(self):
208             global models
209             global X_test
210             global y_test
211             X = data[['Temperatura (°C)', 'Humedad (%)']]
212             y = data['Falla']
213             X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
214
215             models = {
216                 'Regresión Logística': LogisticRegression(),
217                 'Árbol de Decisión': DecisionTreeClassifier(),
218                 'Red Neuronal': MLPClassifier(max_iter=500)
219             }
220
221             results = "Resultados:\n"
222             self.estadisticas_modelos.clear()
223             fig, ax = plt.subplots(figsize=(4.6, 2.3))  # Ajustar tamaño del gráfico
224             accuracies = []
225
226             for name, model in models.items():
227                 model.fit(X_train, y_train)
228                 y_pred = model.predict(pd.DataFrame(X_test, columns=X.columns))  # Evita advertencias
229                 acc = accuracy_score(y_test, y_pred)
230                 report = classification_report(y_test, y_pred, zero_division=0)
231                 self.estadisticas_modelos[name] = f"Precisión: {acc:.4f}\n{report}"  # Guardar estadísticas
232                 results += f"{name} - Precisión: {acc:.4f}\n{report}\n"
233                 accuracies.append((name, acc))
234
235             # Crear gráfico de barras con diferentes colores
236             names, scores = zip(*accuracies)
237             colors = ['blue', 'green', 'red']  # Colores para cada barra
238             bars = ax.bar(names, scores, color=colors)
239
240             # Agregar valores sobre cada barra
```

```
            ax.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.02,
                    f"{score:.4f}", ha='center', fontsize=5, fontweight='bold')

        # Configuración del gráfico
        ax.set_title("Comparación de Modelos", fontsize=10)
        ax.set_ylabel("Precisión", fontsize=9)
        ax.set_xticks(range(len(names)))  # Fijar ticks antes de etiquetas
        ax.set_xticklabels(names, fontsize=8)

        # Mostrar en QGraphicsView
        canvas = FigureCanvas(fig)
        scene = QGraphicsScene()
        scene.addWidget(canvas)
        self.ui.graphicsView.setScene(scene)
        canvas.draw()

    def predict_value(self):
        temp = float(self.ui.lineEdit.text())
        hum = float(self.ui.lineEdit_2.text())

        # Crear DataFrame con nombres de columnas correctos
        input_data = pd.DataFrame([[temp, hum]], columns=['Temperatura (°C)', 'Humedad (%)'])

        results = ""
        for name, model in models.items():
            pred = model.predict(input_data)  # Ahora input_data tiene nombres de columnas
            if pred[0] == 1:  # Si la predicción es "Falla"
                falla_tipo = self.clasificar_falla(temp, hum)
                results += f"{name}: Falla - {falla_tipo}\n"
                    self.tipo_falla = falla_tipo
            else:
                results += f"{name}: No Falla\n"
                if name == "Regresión Logística":
                    self.tipo_falla = None

        # if self.tipo_falla is None:
         #    QMessageBox.information(self, "Estado", "No se ha detectado ninguna falla.")

         self.ui.label_2.setText(results)

    def clasificar_falla(self,temp, hum):
        if temp > 60:
            return "Falla Mecánica"
        elif hum > 65 and temp > 50:
            return "Falla Óptica"
        elif temp > 55:
            return "Falla Electrónica"
        else:
            return "Sin Falla"

    def abrir_ventana_falla(self):
        if self.tipo_falla:
            if "Mecánica" in self.tipo_falla:
                self.ventana_falla = SextaVentana(self)  # Ventana 6
            elif "Óptica" in self.tipo_falla:
                self.ventana_falla = SeptimaVentana(self)  # Ventana 7
            elif "Electrónica" in self.tipo_falla:
```

```
                self.ventana_falla = OctavaVentana(self)  # Ventana 8
            else:
                return

            self.ventana_falla.show()
            self.close()

    def abrir_novena_ventana(self):
        self.ventana = NovenaVentana(self,models, X_test, y_test)  # Pasamos self para poder volver
        self.ventana.show()
        self.close()  # Cierra la ventana ac

class SextaVentana(QMainWindow):
    def __init__(self, ventana_anterior):
        super().__init__()
        self.ui = Ui_Sexta()
        self.ui.setupUi(self)
        self.ventana_anterior = ventana_anterior
        self.ui.pushButton_5.clicked.connect(self.volver_atras)  # Botón de "Atrás"

    def volver_atras(self):
        self.ventana_anterior.show()  # Muestra la ventana anterior
        self.close()  # Cierra la ventana actual

class SeptimaVentana(QMainWindow):
    def __init__(self, ventana_anterior):
        super().__init__()
        self.ui = Ui_Septima()
        self.ui.setupUi(self)
```

```
        self.ventana_anterior = ventana_anterior
        self.ui.pushButton_5.clicked.connect(self.volver_atras)  # Botón de "Atrás"

    def volver_atras(self):
        self.ventana_anterior.show()  # Muestra la ventana anterior
        self.close()  # Cierra la ventana actual

class OctavaVentana(QMainWindow):
    def __init__(self, ventana_anterior):
        super().__init__()
        self.ui = Ui_Octava()
        self.ui.setupUi(self)
        self.ventana_anterior = ventana_anterior
        self.ui.pushButton_5.clicked.connect(self.volver_atras)  # Botón de "Atrás"

    def volver_atras(self):
        self.ventana_anterior.show()  # Muestra la ventana anterior
        self.close()  # Cierra la ventana actual

class NovenaVentana(QMainWindow):
    def __init__(self, ventana_anterior,models, X_test, y_test):
        super().__init__()
        self.ui = Ui_Novena()
        self.ui.setupUi(self)
        self.models = models
        self.X_test = X_test
```

```
355         self.y_test = y_test
356         self.ventana_anterior = ventana_anterior
357         self.ui.pushButton_3.clicked.connect(self.volver_atras)  # Botón de "Atrás"
358         #modelos = list(self.estadisticas_modelos.keys())
359         #self.ui.label_2.setText(self.estadisticas_modelos[modelos[0]])  # Modelo 1
360         #self.ui.label_6.setText(self.estadisticas_modelos[modelos[1]])  # Modelo 2
361         #self.ui.label_7.setText(self.estadisticas_modelos[modelos[2]])  # Modelo 3
362         self.mostrar_estadisticas()
363 
364         # Mostrar matrices de confusión en los QGraphicsView
365         self.mostrar_matrices_confusion()
366 
367     def volver_atras(self):
368         self.ventana_anterior.show()  # Muestra la ventana anterior
369         self.close()  # Cierra la ventana actual
370 
371     def mostrar_estadisticas(self):
372         labels = [self.ui.label_2, self.ui.label_6, self.ui.label_7]
373         for i, (name, model) in enumerate(self.models.items()):
374             y_pred = model.predict(self.X_test)
375             report = classification_report(self.y_test, y_pred, zero_division=0)
376             labels[i].setText(f"Modelo: {name}\n{report}")
377 
378     def mostrar_matrices_confusion(self):
379         views = [self.ui.graphicsView, self.ui.graphicsView_2, self.ui.graphicsView_3]
380         fig_size = (4, 4)  # Tamaño de cada figura
381         dpi = 55
```

```
381         dpi = 55
382 
383         for i, (name, model) in enumerate(self.models.items()):
384             y_pred = model.predict(self.X_test)
385             cm = confusion_matrix(self.y_test, y_pred)
386 
387             fig, ax = plt.subplots(figsize=fig_size, dpi=dpi)
388             sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, ax=ax)
389             # Establecer tamaño de fuente para el título y etiquetas
390             ax.set_title(f"Matriz de Confusión - {name}", fontsize=10)  # Título más pequeño
391             ax.set_xlabel("Predicción", fontsize=10)  # Etiqueta X más pequeña
392             ax.set_ylabel("Real", fontsize=10)  # Etiqueta Y más pequeña
393 
394             # Ajustar tamaño de las etiquetas de los valores dentro del mapa de calor
395             for label in (ax.get_xticklabels() + ax.get_yticklabels()):
396                 label.set_fontsize(10)
397 
398             canvas = FigureCanvas(fig)
399             scene = QGraphicsScene()
400             scene.addWidget(canvas)
401             views[i].setScene(scene)
402             canvas.draw()
403 
404 class DecimaVentana(QMainWindow):
405     def __init__(self, ventana_anterior):
406         super().__init__()
407         self.ui = Ui_Decima()
```

```
408             self.ui.setupUi(self)
409             self.ventana_anterior = ventana_anterior
410             self.ui.pushButton_3.clicked.connect(self.volver_atras)  # Botón de "Atrás"
411 
412         def volver_atras(self):
413             self.ventana_anterior.show()  # Muestra la ventana anterior
414             self.close()  # Cierra la ventana actual
415 
416     if __name__ == "__main__":
417         app = QApplication(sys.argv)
418         ventana = VentanaPrincipal()
419         ventana.show()
420         sys.exit(app.exec_())
421 
```

ANEXO 2

NAVEGACIÓN DE LA INTERFAZ