UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE DIAGNÓSTICO Y DETECCIÓN DE FALLAS EN UN SISTEMA DE AIRE ACONDICIONADO DE PRECISIÓN, MEDIANTE INTELIGENCIA ARTIFICIAL

PRESENTADO POR:
Br. AMILCAR QUISPE ASTORGA

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:
Dr. Ing. ROGER JESÚS COAQUIRA CASTILLO

CUSCO - PERÚ
2025

Universidad Nacional de San Antonio Abad del Cusco

INFORME DE ORIGINALIDAD

El que suscribe, asesor Roger Jesus Coaquira Castillo, quien aplica el software de detección de similitud al trabajo de tesis titulada: **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE DIAGNÓSTICO Y DETECCIÓN DE FALLAS EN UN SISTEMA DE AIRE ACONDICIONADO DE PRECISIÓN, MEDIANTE INTELIGENCIA ARTIFICIAL"**, presentado por: **AMILCAR QUISPE ASTORGA** con DNI Nro: **70585747**, para optar el título profesional de: **INGENIERO ELECTRONICO.**

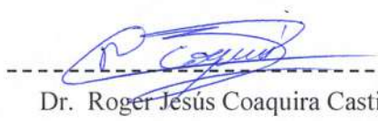
Informo que el trabajo de investigación ha sido sometido a revisión por dos veces, mediante el Software de detección de similitud, conforme al Art. 6° del *Reglamento para Uso del Sistema de Detección de Similitud de la UNSAAC* y de la evaluación de originalidad se tiene un porcentaje de 10% de similitud.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No sobrepasa el porcentaje aceptado de similitud.	X
Del 11 al 30 %	Devolver al usuario para las subsanaciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, conforme al reglamento, quien a su vez eleva el informe al Vicerrectorado de Investigación para que tome las acciones correspondientes; sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de **asesor**, firmo el presente informe en señal de conformidad y adjunto la primera página del reporte del Sistema de Detección de Similitud.

Cusco, 8 de abril de 2025



Dr. Roger Jesús Coaquira Castillo

Nro. de DNI 01333608

ORCID del Asesor: 0000-0003-3791-110X

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: https://unsaac.turnitin.com/viewer/submissions/oid:27259:447019298?locale=es-MX

Amilcar Quispe Astorga

Volumen de Tesis Amilcar Quispe Astorga reposi.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:447019298

Fecha de entrega
8 abr 2025, 12:06 p.m. GMT-5

Fecha de descarga
8 abr 2025, 12:17 p.m. GMT-5

Nombre de archivo
Volumen de Tesis Amilcar Quispe Astorga reposi.pdf

Tamaño de archivo
10.4 MB

241 Páginas

50.946 Palabras

295.566 Caracteres

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

Top Sources

7% Internet sources
1% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Dedicatoria

El presente trabajo lo dedico a Dios, por acompañarme siempre, porque todo lo que tengo y soy es gracias a Él.

A mi familia, y a todos los que lo integran, por haberme brindado el mejor ambiente de crecimiento, especialmente en mi desarrollo integral como ser humano.

Agradecimientos

En primer lugar, quiero agradecer a Dios por siempre estar presente en cada experiencia de vida, por ser mi fuente de fuerza, iluminar mi mente y poner en mi destino a personas tan maravillosas. A mi familia, principalmente a mis padres, hermana y cada integrante; a todos mis abuelos, tíos, tías, primos, primas, sobrinos y sobrinas. Con su apoyo se culminó este proyecto (tesis). Este trabajo lo ofrezco a Dios, por darme todo.

Agradezco profundamente a mi asesor, Dr. Ing. Roger Jesús Coaquira Castillo, por su guía, paciencia y gran compromiso, fue fundamental para la realización de este estudio. Al laboratorio LIECAR y a todos sus integrantes, gracias de corazón, por su apoyo en este objetivo.

A cada docente de la Escuela Profesional de Ingeniería Electrónica, quienes desempeñaron un papel crucial en mi desarrollo académico y profesional, especialmente a los docentes dictaminadores, Ing. Luis Jimenez Troncoso y Alex Jhon Quispe Mescco.

Y a todas las personas que tuve el honor de conocer formando una parte de mí mentores (Branko VC) que también me inspiraron a seguir adelante con determinación absoluta.

Resumen

Los sistemas de aire acondicionado de precisión (AAP) siempre están propensos a diferentes tipos de fallas. Para abordar este desafío, se ha desarrollado un sistema de detección y diagnóstico de fallas automático en tiempo real, que clasifica los eventos como fallado o normal, mediante el análisis de señales de estado (presión, temperatura, corriente y voltaje). Esta investigación está fundamentada en Data-Driven y modelos de aprendizaje automático donde se propone una estrategia específica para cinco tipos de fallas del sistema. Se trabajó en el sistema de aire acondicionado de precisión, de la marca Rittal, modelo SK3328.500 (enclosure top therm, cooling unit), donde se instaló sensores de presión capacitivos, de corriente de efecto hall, de voltaje de inducción electromagnética, de temperatura infrarrojo y termocupla, bajo un análisis previo en el capítulo IV.

Para la implementación del sistema se obtuvo un dataset de las señales de condición de estado del sistema AAP, inicialmente compuesto de 31,059 muestras, después de una etapa de preprocesamiento (módulo RandomUnderSampler-RUS) se obtuvo la base de datos con 20,000 muestras, las cuales incluyen eventos de funcionamiento normal y de fallas, generados en el sistema de aire acondicionado de precisión (AAP), se realizó la selección del mejor modelo de aprendizaje automático, bajo el criterio de exactitud (accuracy), se hicieron pruebas en condiciones diferidas (base de datos) y en tiempo real, el modelo Support Vector Machine alcanzo un 93%, Decisión Tree alcanzo un 93%, Gradient Boosting alcanzo un 91%, K-Nearest Neighbors alcanzo un 83%, Naive Bayes alcanzo un 77% y el modelo Random Forest sobresalió, en las pruebas diferidas alcanzando un 96% y un 95.28% de accuracy en tiempo real. Además, se hizo una prueba de validación con el mejor modelo seleccionado, en tiempo real, donde se simula un entorno real para el sistema AAP y alcanzo una tasa de accuracy del 93.49%.

Palabras clave: Aire acondicionado de precisión (AAP); Sistema de detección y diagnóstico de fallas (FDD); Raspberry Pi; Arduino Nano; Efecto hall; Infrarrojo (IR); Inteligencia artificial (IA); (SVM); (KNN); (RF); (GB); (DT); (NB); Exactitud (accuracy).

Abstract

Precision air conditioning (AAP) systems are always prone to different types of failures. To address this challenge, a real-time automatic fault detection and diagnosis system has been developed, which classifies events as faulty or normal, by analyzing status signals (pressure, temperature, current and voltage). This research is based on Data-Driven and machine learning models where a specific strategy is proposed for five types of system failures. The work was done on the precision air conditioning system, Rittal brand, model SK3328.500 (enclosure top therm, cooling unit), where capacitive pressure sensors, Hall effect current sensors, electromagnetic induction voltage sensors, infrared temperature sensors and thermocouple sensors were installed, under a previous analysis in chapter IV. For the implementation of the system, a dataset of the status condition signals of the AAP system was obtained, initially composed of 31,059 samples, after a preprocessing stage (RandomUnderSampler-RUS module) the database with 20,000 samples was obtained, which include normal and failure operating events, generated in the precision air conditioning system (AAP), the selection of the best machine learning model was made, under the accuracy criterion, tests were done in deferred conditions (database) and in real time, the Support Vector Machine model reached 93%, Decision Tree reached 93%, Gradient Boosting reached 91%, K-Nearest Neighbors reached 83%, Naive Bayes reached 77% and the Random Forest model stood out, in the deferred tests reaching 96% and 95.28% accuracy in real time. In addition, a validation test was performed with the best selected model, in real time, where a real environment is simulated for the AAP system and an accuracy rate of 93.49% was achieved.

Keywords: Precision Air Conditioning (PAC); Fault Detection and Diagnostic System (FDD); Raspberry Pi; Arduino Nano; Hall Effect; Infrared (IR); Artificial Intelligence (AI); (SVM); (KNN); (RF); (GB); (DT); (NB); Accuracy.

Introducción

Los sistemas aire acondicionado de precisión (AAP) de organizaciones privadas como públicas son susceptibles a diversos tipos de fallos. Por factores como daños mecánicos, exposición al polvo, corrosión, operación y mantenimiento inadecuados, entre otros, contribuyen a disminuir el rendimiento de estos sistemas, causando una variedad de fallos. Estos problemas pueden afectar considerablemente el confort y la calidad del aire, además de propiciar un uso ineficiente de la energía. Las aplicaciones prácticas han demostrado que las estrategias de detección y diagnóstico de fallas (FDD) oportunas y precisas pueden evitar eficazmente las fallas, reduciendo los tiempos de detección en el sistema AAP que se desarrolla bajo el enfoque de inteligencia artificial. La inspección manual y convencional de cada elemento implica un trabajo exigente que demanda amplios conocimientos mecánicos y experiencia en mantenimiento.

La finalidad de este trabajo es elaborar una solución precisa y adaptable para detectar y diagnosticar indicios de falla en un sistema de aire acondicionado de precisión en tiempo real, estableciendo una base para futuros avances en este campo. Este volumen se organizó en ocho capítulos que cubrieron diferentes aspectos del estudio.

En el Capítulo 1, se desarrolla las generalidades del estudio, resaltando la relevancia de enfrentar el desafío de la detección de fallas del sistema de aire acondicionado en tiempo real y de forma automática. Definiendo la problemática, justificación y sus objetivos.

El Capítulo 2 señala los fundamentos teóricos esenciales para entender los principios y tecnologías principales que intervienen en el sistema de detección de fallas. Se abordan aspectos como los sistemas heating, ventilation, and air conditioning (HVAC), base de datos, modelos de aprendizaje automático (algoritmos para clasificar señales), sistemas embebidos. Además, se describe las variables.

En el Capítulo 3, se enfocó en definir los procedimientos y técnicas esenciales para estructurar y validar este estudio. Se describe el tipo y nivel de investigación, el diseño experimental, alcances y las

técnicas empleadas, asegurando la confiabilidad y aplicabilidad de los resultados, destacando la importancia de las decisiones metodológicas en el éxito del proyecto.

En el Capítulo 4, se enfoca en el diseño del sistema, estableciendo de manera precisa tanto los requerimientos generales como los específicos para el software y el hardware. En esta sección se describen los subsistemas principales: el subsistema de adquisición, encargado de recoger las señales de los sensores y el subsistema de detección, que procesa estos datos mediante modelos de inteligencia artificial. Asimismo, se detalla la estructura de flujo de los módulos (etapas de código) que componen el subsistema de detección, lo que permite comprender cómo se integran y se interrelacionan las diferentes fases del procesamiento de datos.

El Capítulo 5 aborda la implementación del sistema, inicialmente se describe la unidad de análisis en evaluación y se detallan procedimientos técnicos cruciales para la ubicación e instalación instrumental. Examinando el sistema embebido conforme a los requerimientos definidos, se detalla la configuración y puesta en práctica del algoritmo necesario para cada módulo del sistema. También se desarrolla la base de datos, configurando el sistema para cada evento de falla por evaluar, luego se procede a registrar en MySQL y cada registro se filtra, etiqueta y divide la base de datos, para entrenar los modelos de aprendizaje y validarlos.

En el Capítulo 6, se centra en las pruebas y resultados conseguidos. Se expresa a detalle las evaluaciones hechas en condiciones diferidas (base de datos) y en tiempo real sobre el sistema de aire acondicionado de precisión, en los eventos de falla estudiados.

En el Capítulo 7, se interpretan los resultados del proyecto, relacionándolos con los objetivos y la literatura existente. Se evalúa las métricas de los modelos de aprendizaje automático (KNN, SVM, RF, GB, DT, NB) y el rendimiento del subsistema de adquisición, identificando fortalezas y áreas de mejora.

En el Capítulo 8 se representan mediante tablas, los costos de implementación y finalmente se describen las conclusiones y recomendaciones.

Índices

Índice de Figuras

Índice de Tablas

Abreviaciones y Acrónimos

AAP: Aire Acondicionado de Precisión (Precision Air Aconditioning en inglés).

FDD: Diagnóstico y Detección de Fallas (Fault Diagnosis and Detection en inglés).

IA: Inteligencia Artificial (Artificial Intelligence en inglés).

ADC: Conversión de Analógico a Digital (Analog-to-Digital Convert en inglés).

IR: Infra Rojo (infrared en inglés).

SVM: Máquina de Vectores Soporte (Support Vector Machine en inglés).

KNN: k-Vecinos Cercano (K-Nearest Neighbors en inglés).

RF: Bosque Aleatorio (Random Forest en inglés).

GB: Aumento de Gradiente (Gradient Boosting en inglés).

DT: Arbol de Decisión (Decision Trees en inglés).

NB: Bayes Ingenuo (Naive Bayes en inglés).

I2C: Inter-Circuito Integrado (Inter-Integrated Circuit en inglés).

USB: Bus Universal en Serie (Universal Serial Bus en inglés).

JSON: Notación de Objeto de JavaScript (JavaScript Object Notation en inglés).

CSV: Valores Separado por Comas (comma-separated values en inglés).

Accuracy: Métrica para evaluar modelos de clasificación de Machine Learning

SQL: Lenguaje de Consulta Estructurado (Structured Query Language en inglés).

OC: Sobrecarga de Refrigerante (Refrigerant Overcharge)

UC: Subcarga de Refrigerante (Refrigerant Undercharge)

RL: Restricción de línea (Line Restriction)

CA: Reducción del Flujo de Aire del Condensador (Condenser Airflow Reduction)

CE: Reducción del Flujo de Aire del Evaporador (Evaporator Airflow Reduction)

Capítulo I

Generalidades

1.1 Descripción de la Realidad Problemática

La demanda de los sistemas de Aire Acondicionados de Precisión (AAP) actuales requiere un trabajo eficiente y permanente, siendo este sistema, parte indispensable para la industria de climatización y aunque existen investigaciones con enfoques similares (por ejemplo, Wang et al., 2021), la mayoría de las soluciones implementadas en la industria de Aires Acondicionados son limitadas y se basan en evaluaciones manuales y mantenimiento preventivo periódico, a pesar de que representa un porcentaje importante de consumo de energía a nivel mundial y nuestro país no es ajeno, cabe resaltar que, este sistema AAP representa un alto porcentaje de consumo de energía en esta industria (ASHRAE, 2013).

En investigaciones más recientes, se ha señalado que los sistemas (AAP) son propensos a diversos tipos de fallos. Factores como daños mecánicos, exposición al polvo, corrosión, operación y mantenimiento inadecuados, entre otros, contribuyen a la reducción del rendimiento de estos sistemas, generando una variedad de fallos. Estos problemas pueden afectar considerablemente el confort y la calidad del aire, además de propiciar un uso ineficiente de la energía. Además, se resalta que, de acuerdo con las tendencias actuales, los métodos basados en datos están emergiendo como cruciales para el futuro de estos sistemas, de acuerdo con la revisión de antecedentes, se subraya su exactitud y precisión en el manejo de bastantes datos en tiempo real, así como su habilidad para mejorar a partir de la experiencia. Su aplicación permite la detección temprana de indicios de fallas, especialmente en datos no estructurados, como registros de sensores. La capacidad de detectar indicios de fallas puede ayudar a una planificación de mantenimiento más efectiva, no obstante, la cantidad de investigaciones que aplican este enfoque en sistemas de AAP sigue siendo limitada (Wang, et all. 2021).

Muchas organizaciones tanto privadas como públicas, que requieren de una alternativa de climatización, no disponen de un sistema integral para la detección de fallas, que sea efectivo para el mantenimiento, según el personal encargado de esta área, no tienen mecanismos adecuados para atenuar las consecuencias de una serie de problemas como; reparaciones en este sistema industrial, interrupciones de servicio, reducción de la durabilidad del equipo lo que conlleva eventualmente a pérdidas económicas y pérdida de prestigio de la empresa. Donde solo hacen evaluaciones de manera convencional y en el mejor de los casos recurren a procedimientos de un mantenimiento preventivo cada cierto periodo de tiempo, realizando mediciones con instrumentos manuales como multímetros, termómetros infrarrojos, etc. Y revisando el historial en sus tarjetas de control de cada equipo.

Es complejo asegurar el funcionamiento continuo de un sistema sin un seguimiento de los parámetros de funcionamiento, sobre todo la gran dificultad de hacer este seguimiento sin interrumpir el funcionamiento del equipo (Guerra, 2013). Por lo que resulta, indispensable tener un sistema que detecte y diagnóstique indicios de fallas, que pueda evaluar, que si los parámetros de funcionamiento (señales de presión, temperatura, voltaje y corriente) del sistema AAP están fuera del rango de tolerancia para un patrón de funcionamiento normal, sino también de manera implícita identificar cuándo y cómo, esta gran cantidad de señales no estructuradas de los sensores, varían (Zhu, Du, Chen, Jin, & Huang, 2019).

Mediante la generación controlada de las fallas más comunes y recurrentes; Sobrecarga de Refrigerante (OC), Subcarga de Refrigerante (UC), Restricción de línea (RL), Reducción del Flujo de Aire del Condensador (CA) y Reducción del Flujo de Aire del Evaporador (CE) (Ebrahimifakhar, Kabirikopaei, & Yu, 2020).

1.2 Formulación del Problema

1.2.1 Problema General

En los sistemas de Aire Acondicionado de Precisión (cooling unit), siempre existen fallas (las más comunes) que afectan su desempeño y que pueden surgir fuera de los periodos de mantenimiento, la

falta de un seguimiento constante y registro de las variables de funcionamiento (base de datos) dificulta la identificación temprana de estas fallas, ya que se realizan evaluaciones utilizando instrumentos manuales. Además, la limitada implementación de métodos automatizados para la adquisición y análisis en tiempo real limita la detección de estas fallas.

1.2.2 Problemas Específicos

Problema Específico 1. Para garantizar un funcionamiento y trabajo permanente, sobre todo, sin tener que detener el sistema, se requiere un seguimiento de las variables de funcionamiento del Aire Acondicionado de Precisión (AAP).

Problema Específico 2. No se cuenta con un registro de los datos que definen la condición de estado del Aire Acondicionado de Precisión (AAP), para situaciones específicas de fallas, más comunes.

Problema Específico 3. La ausencia de un modelo de aprendizaje con alta precisión y exactitud para la detección y clasificación en tiempo real, de eventos de falla en el sistema aire acondicionado de precisión (AAP), limita la capacidad de identificar indicios y la toma de decisiones oportunas.

1.3 Justificación

1.3.1 Justificación Tecnológica

Con este trabajo, se busca dar una herramienta que reduzca los tiempos de detección de fallas del sistema AAP y se desarrolla bajo el enfoque de inteligencia artificial donde se obtiene un modelo con estructuras de aprendizaje automático (algoritmos) con IA (machine learning) para la clasificación de las señales de estado del AAP, entrenados mediante la generación controlada de fallas más comunes (OC, UC, RL, CA y EA), evaluadas en este proyecto, que a su vez se requiere bastante conocimiento y criterio para la selección del módulo de adquisición y los sensores, más ideales, con la capacidad de resolver el problema de este proyecto, con la finalidad de conseguir un set de datos con señales que representen el estado del sistema de AAP y con este conocimiento se podría evaluar otros equipos de la industria, aplicando conceptos de IA en tipos de aprendizaje automático para la detección.

1.3.2 Justificación Práctica

En relación al mantenimiento, se busca mejorar la duración operativa del aire acondicionado de precisión. Además, se pretende reducir las interrupciones imprevistas, en base al detección y diagnóstico del indicio de fallas, con información que podría ser valiosa para los encargados del mantenimiento. Que tiene como finalidad promover el enfoque del mantenimiento predictivo automatizado (inteligencia artificial) como una filosofía y una herramienta esencial para administrar el mantenimiento en el sector de climatización.

1.3.3 Justificación Social

En el sector industrial, como instituciones del gobierno nacional demandan mayor rendimiento y alta eficiencia en los sistemas AAP actuales que a su vez estos atienden las necesidades de la I4.0 y muchas de estas empresas tiene en común el problema de no tener un mantenimiento fiablemente, estructurado y organizado. Si normalmente, surgen las fallas fuera de los alcances del periodo del plan mantenimiento y este trabajo como una herramienta podría ayudar a detectar indicios de fallas en los sistemas de AAP, en tiempo real, en base al análisis de las señales de estado (P, T°, V, I) mediante un algoritmo de aprendizaje automático (IA). Además, consecuentemente podría ayudar a disminuir costos de reparación, cambios de manera prematura, reducir tiempos fuera de servicio e incrementar la duración operativa de los sistemas que, en consecuencia, maximizar los niveles de productividad y de esta manera reducir costos.

También debido a que el servicio de mantenimiento preventivo es costoso con esta solución se requerirá con menor frecuencia. En resumen, esta información podría ayudar a los encargados de mantenimiento como una herramienta para que evalúen estrategias de mantenimiento, sobre el estado y ser implementado en otros ambientes donde se encuentren estos sistemas de AAP, esto resulta de interés para diferentes tipos de empresas e instituciones.

1.4 Objetivos

1.4.1 Objetivo General

Diseñar e implementar un sistema de diagnóstico y detección de fallas en un sistema de Aire Acondicionado de Precisión, basado en las señales de presión, temperatura, voltaje y corriente, mediante Inteligencia Artificial.

1.4.2 Objetivos Específicos

Objetivo Específico 1. Diseñar e implementar un sistema de instrumentación y adquisición de datos capaz de registrar las señales variables de condición del estado (presión, temperatura, voltaje y corriente) del sistema de aire acondicionado de precisión (AAP).

Objetivo Específico 2. Generar una base de datos con señales de la condición de estado del Aire Acondicionado de Precisión (AAP), para situaciones específicas de las fallas más comunes, mediante la generación controlada de fallas en el sistema AAP.

Objetivo Específico 3. Seleccionar el mejor modelo de aprendizaje (IA), entrenado con la base de datos, para la detección y clasificación de los eventos de falla evaluados en tiempo real, en pruebas a escala piloto de desempeño.

Capítulo II

Marco Teórico

2.1 Antecedentes

2.1.1 Trabajos Internacionales

En el trabajo de investigación realizada por (Ebrahimifakhar, Kabirikopaei, & Yu, Data-driven fault detection and diagnosis for packaged rooftop units using statistical machine learning classification methods, 2020), en Nebraska-Lincoln University, Omaha, United States, basada en datos del sistema de aire acondicionado de tipo RoofTop Units utilizando métodos de clasificación de aprendizaje automático.

La estrategia planteada aborda la tarea FDD como un problema de clasificación de múlticlase. Debido que los datos experimentales para RTU´s son raros y difíciles de obtener, se utiliza una biblioteca de datos simulada de fallas de modelos bajo condición de estado estable para entrenar y validar los algoritmos de clasificación. Así mismo, utiliza la técnica de sobremuestreo sintético de minorías para crear muestras artificiales de clase menos representadas con el propósito de balancear el conjunto de datos. Para evaluar siete tipos de fallas: refrigerant undercharge, refrigerant overcharge, compressor valve leakage, restriction line, condenser airflow reduction, evaporador airflow reduction y non-condensable gas. Donde se evaluó mediante los siguientes modelos bagging (BA), XGBoost (XGB), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), k-nearest neightbors (KNN), random forest (RF), adaBoost (AD), support vector machine (SVM) y regresión lógica (LR), basado en 15 variables como temperatura, presión y potencia del compresor. Finalmente, se utilizó un método supervisado basado en el modelo support vector machine, logrando una precisión del 96.2%. Por medio de un diagrama comparativo, se evidenció que "el rendimiento general del modelo SVM, ajustado con los parámetros óptimos, fue el mejor, mientras que el modelo LDA tiene el índice de precisión general más bajo de 76.2%". Destaco que las fallas poco comunes son complejas de evaluar por el limitado número de datos, este trabajo también examina la relevancia relativa de las variables de entrada.

En la sección de conclusiones los resultados de los métodos de clasificación BA, RF, AD y XGB revelan que TLL y Tdischg son las variables predictoras más relevantes en el proceso de detección y diagnóstico de fallas de RTU. Además, se destaca que el uso de la técnica de sobremuestreo sintético minoritario (SMOTE) puede mitigar el problema del desequilibrio de clases.

(Zhu, Du, Chen, Jin, & Huang, 2019)En el artículo de investigación, "Hybrid model based refrigerant charge fault estimation for the data center air conditioning system", se presenta un novedoso modelo híbrido con enfoque en la estimación de fallas de la carga de refrigerante basado en métodos de machine learning, como un instrumento robusto de estimación de la carga de refrigerante para los sistemas de aire acondicionado de los edificios actuales. El trabajo eficiente del hardware informático sensible puede ser influenciado por la insuficiente capacidad de enfriamiento, debido a fallas de fuga de refrigerante. Incluso para los Datacenters con menos ocupación, el funcionamiento de servidores y los equipos electrónicos pueden verse muy influenciados por la falla de fuga de refrigerante. Para el fin de asegurar la operación confiable de los servidores y otros dispositivos electrónicos, el sistema de aire acondicionado en los centros de datos debe proveer una temperatura y humedad ambiental más estrictas aún más que el aire acondicionado convencionales. La subcarga en el aire acondicionado de un centro de datos puede provocar fallos en dispositivos electrónicos clave debido a la insuficiente capacidad de refrigeración y a las altas temperaturas interiores las cuales traería grandes pérdidas económicas. En consecuencia, es necesario desarrollar sistemas de modelos robustos para estimar las intensidades de falla de carga insuficiente o sobrecarga de refrigerante para los sistemas AAP de DC.

En primer lugar, la mejora se presenta como un modelo semi empírico para estimar la falla de la subcarga.

En segundo lugar, el coeficiente de información máximo (MIC) se utiliza para analizar y seleccionar las variables de entrada adicionales que tienen una alta correlación con el residuo de predicción. Como un algoritmo basado en el marco del aumento, the gradient boosting decision tree (GBDT), se emplea

para diseñar el modelo de aprendizaje para compensar el residuo de predicción del modelo semi empírico mejorado. Además, se desarrolla el modelo híbrido que combina los modelos semi empíricos y de aprendizaje automático.

Finalmente, se compara y evalúa las capacidades de predicción y generalización de los distintos modelos bajo diferentes condiciones, utilizando los datos experimentales.

(Salazar Garcia, 2023) En su investigación," Sistema de monitorización y alerta temprana para mantenimiento preventivo de compresores en procesos industriales bajo industria 4.0", señala que en muchas organizaciones no tienen estrategias apropiadas para limitar el daño causado por la reparación de los equipos industriales cuando se averían, especialmente para los compresores y otros aparatos. Por lo tanto, es necesario que cuenten con un mecanismo que notifique con antelación sobre el periodo que las maquinas han estado en uso, para ejecutar un mantenimiento preventivo que atenué un posible desgaste o avería.

Se puede llevar a cabo el monitoreo del compresor mediante un servicio Android y con un dispositivo de adquisición de señales que se conecte a Internet (IoT) y varios sensores. Esto permitirá la adquisición de variables del compresor y los periodos de funcionamiento; la evaluación se hará en ciertos eventos específicos en función de la operación del compresor, obteniendo las variables correspondientes. Para verificar el rendimiento del equipo, se indujeron las fallas de manera manual, y el sistema trabajó de buena manera como se preveía inicialmente. Los dispositivos de medición estarán monitoreando continuamente esas variables, si se presenta una variación, el equipo IoT notificará a la aplicación del dispositivo. Los parámetros a monitorea son: T° ambiente, T° de contacto, periodos de uso, corriente y humedad relativa.

(Laughman , Armstrog, Leeb, & Armstrong, 2006)En este trabajo "Detection of rooftop cooling unit faults based on electrical measurements", usan el método (NILM) Nonintrusive Load Monitoring, únicamente usando valores o señales eléctricas de voltaje y corriente que puedan ser usados para

detectar fallas, además señala que el método Fault Diagnosis and Detection (FDD) es efectivo en costos por que está enfocado en la medida de temperatura para detectar con sensores de bajo costo, la localización de los sensores es crítico, pero con el uso de sensores de presión y potencia mejoraría la confiabilidad, toma varias fallas evaluadas en los antecedentes y los analiza a nivel eléctrico, menciona que toma 15716 muestras el 76% eléctricas 18% mecánicas 5% del circuito de refrigeración.

En el desarrollo del trabajo evalúa seis tipos de fallas, pero con enfoque de análisis eléctrico: 1 Short Cycling (hay un incremento de potencia en el estado estacionario al final del transitorio, comparado con uno con alivio de presión). 2 Refrigerant Charge - Undercharge que hay mucho vapor sobrecalentado en el evaporador y subcalentado en el condensador. Overcharge hay mucho líquido calentado en el condensador. Para ambos casos se incrementa el nivel de presión e incrementa la potencia del compresor como síntoma de subcarga y sobre carga. 3 Compressor Back Leakage (para la falla de fuga, la presión del condensador sube lentamente hasta que el vapor busca la condición de saturación). 4 Flow Blockage (consiste en bloquear en niveles 10 50 100% para reducir el flujo). 5 Load Spectra With Artificial Introduced Imbalance Faults (el ventilador del condensador y la alimentación son susceptible a fallas, debido al desequilibrio, vibraciones asociadas a variaciones periódicas de carga en el motor y estas variaciones pueden ser detectadas, este desequilibrio puede resultar una hoja del impulsador doblada, de acumulación de suciedad (filtros) objetos extraños), esta puede ser detectada por la descomposición de la señal de potencia tanto para el ventilador del condensador/alimentación. 6 Compressor Liquid Ingestion (falla poco frecuente según los antecedentes, se evalúa en un compresor)

En la parte final señala los enfoques de los métodos de detección (análisis de frecuencias, transitorios anómalos, niveles de voltaje y corriente y desbalance de las fases) y todas las fallas desarrolladas pueden ser detectadas solo con sensores de voltaje y corriente.

2.1.2 Trabajos Nacionales

(Guamán Buestán, 2019)En su proyecto de tesis “Desarrollo de un modelo basado en datos a partir de señales de vibración para la detección de fallos de un compresor reciprocante de simple efecto de doble etapa”, se centra en tres aspectos fundamentales. En primer lugar, se aborda la recopilación de datos; en segundo lugar, se diseña un procedimiento para el preprocesamiento de los valores de la vibración y finalmente, se elabora un enfoque basado en RN recurrentes Long Short Term Memory (LSTM) para la detección de fallos.

En el informe, se proporciona una explicación sobre las fallas más frecuentes en compresores reciprocantes y examina el uso de redes neuronales para su modelado. También se realiza una experimentación para capturar valores de vibración, incluyendo una descripción minuciosa de una estrategia experimental y los instrumentos empleados. Seguidamente, se lleva a cabo el preprocesamiento de los valores recogidos mediante un método diseñado se pretende incrementar la cantidad de series temporales relevantes del equipo y generar señales con una resolución más baja. Finalmente, se pone en práctica y se evalúa el modelo de diagnóstico, entrenando la red LSTM con los conjuntos óptimos de hiperparámetros determinados a través de una óptica bayesiana que restringe el espacio de búsqueda.

Este método se utiliza específicamente para identificar fallas en las válvulas de entrada y salida del compresor. Los hallazgos del estudio *revelan que el modelo más eficiente alcanza una precisión* del 93%, superando a tres enfoques clásicos en términos de rendimiento en el modelado con redes LSTM. En resumen, se utiliza Python para su etapa de procesamiento y el diseño de modelos, empleando un sensor piezoeléctrico para la señal de vibración junto con equipos de National Instruments (Guamán Buestán, 2019).

2.2 Base Teórica

2.2.1 Estudio del Sistema HVAC – AAP, Partes y Principio de Funcionamiento

HVAC (Heating, Ventilation and Air Conditioning) Es un sistema que implica la regulación de las condiciones ambientales con propósitos industriales y de confort, Bustamante Milla (2018) describe.

> El principio de funcionamiento del aire acondicionado, que está basado en principios físicos: primero el calor va de la parte más caliente al más frio, segundo los fluidos absorben calor cuando se transforman de líquido a gas, tercero los fluidos liberan calor cuando se transforman de gaseoso a líquido y cuarto el límite de ebullición de un fluido varía según la presión ejercida.

Que abarcan diversas configuraciones creadas para satisfacer los requerimientos específicos de climatización en diferentes entornos. Pueden estar clasificados por distintas criterios, como medio por donde se transfiere, directo o a través de un fluido (refrigerante, aire, agua o una combinación de estos), por su distribución (split, minisplit, compact), por sus elementos componentes (solo ventilación, solo refrigeración, solo calefacción o la combinación de estos), entre los más populares se puede distinguir a RTU(RoofTop Unit) son sistemas de aire acondicionado completos que se instalan en parte superior y son ideales para uso comercial general, VRF (Variable Refrigerant Flow) es un sistema de aire acondicionado integral que gestiona el flujo de refrigerante para varias unidades desde una unidad control, destacándose con una alta eficiencia energética y flexibilidad en el control de temperatura, AHU (Air Handling Unit) encargados de tratar el aire que circula dentro de un sistema . Estos equipos captan el aire exterior, lo filtran, lo enfrían y lo distribuyen en instalaciones de gran escala a través de conductos.

Además, el servicio de climatización equivale al 30 a 40% de la energía total requerida, solo después de los equipos informáticos. Los sistemas HVAC también se dividen como sistemas de confort y precisión, los centros de datos requieren un sistema de climatización especializado. Porque el consumo de recursos en el día no es igual en las diferentes horas (Bustamante Milla, 2018).

Aire acondicionado de precisión (AAP)

Se emplea en ambientes donde las condiciones de operación son más sensibles con requisitos de control muy precisos, como en laboratorios, salas blancas, quirófanos o datacenters. Estas instalaciones requieren un enfriamiento constante durante todo el año, las 24 horas del día, así como la capacidad de satisfacer las demandas de equipos electrónicos sensibles al calor. Asimismo, es necesario un manejo exacto de la temperatura y la humedad, junto con una mayor calidad del aire y un incremento de la habilidad de enfriamiento. En términos de funcionamiento, los sistemas de aire acondicionado suelen dedicar entre un 80% y un 100% de su capacidad al enfriamiento, mientras que el resto se destina a la remoción de humedad, lo que la diferencia de los sistemas de confort, que generalmente se centran en la carga térmica relacionada con el calor sensible (Calderon Solano, 2021). Independientemente del método de enfriamiento empleado, todos estos sistemas se centran en regular el flujo de refrigeración, la temperatura, el flujo aire y la humedad, ya sea a través de líquido o aire

2.2.2 Elementos más Importantes del Sistema de Refrigeración

Evaporador:

Encargado de captar el calor del entorno al facilitar la evaporación del refrigerante. En el contexto de un centro de datos, el evaporador se ubica en las unidades de manejo de aire (UTA) o unidades de manejo de precisión. Su función consiste en extraer el calor del aire que circula dentro del DC, enfriando de manera efectiva el aire antes de redistribuirlo.

Condensador:

Responsable de liberar el calor previamente absorbido por el evaporador al entorno exterior contribuyendo así a mantener condiciones controladas de temperatura y humedad., puede encontrarse en ubicaciones externas. Su objetivo principal radica en liberar el calor recolectado del aire interno.

Ventilador:

El componente principal encargado de la circulación del aire es el ventilador. Este elemento se encarga de impulsar el aire y asegurar un flujo constante por el sistema, facilitando que el aire

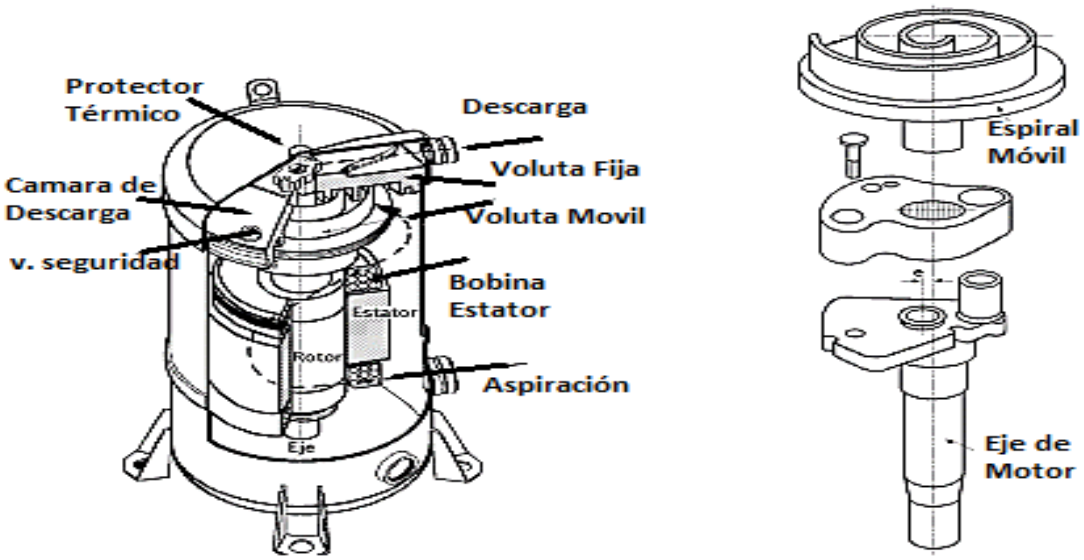
acondicionado se desplace de un lugar a otro a través de los conductos. Generalmente, los ventiladores son impulsados por un propulsor, ya sea de manera directa o mediante el uso de cadenas y poleas. En algunos casos, especialmente en entornos industriales como las plantas de energía, los ventiladores pueden ser impulsados por vapor o turbinas de gas. Y los tipos más conocidos de ventiladores son los centrífugos y los axiales (Calderon Solano, 2021).

Compresor:

Dispositivo encargado de comprimir un fluido refrigerante para aumentar su presión y temperatura. Este aumento provoca que el calor fluya desde el fluido refrigerante hacia el ambiente. Este se ocupa de recircular el fluido refrigerante para regular la temperatura requerida.

Figura 1

Compresor hermético



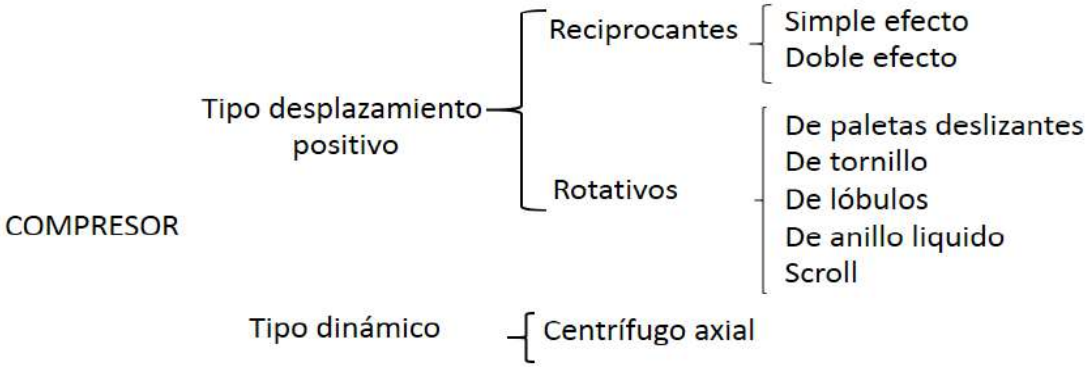
***Nota**: Se presenta las partes del compresor, adecuado de* (Fernandez DIez, 2000).

El funcionamiento conjunto del evaporador y el condensador crea un ciclo continuo. El evaporador absorbe el calor del aire interior y lo transfiere al refrigerante. El condensador, por su parte, libera este calor al entorno exterior. Este ciclo de absorción y liberación de calor permite crear y mantener un entorno controlado, ideal para el funcionamiento óptimo de los equipos sensibles.

2.2.3 Tipos de Compresores

A pesar de que todos los compresores tienen el mismo objetivo, se usan para cada situación y procedimiento específico en diferentes áreas industriales. Se dividen en dos grandes grupos por su principio de funcionamiento. Resulta que el compresor evaluado para este proyecto es de tipo desplazamiento positivo de clase rotativo.

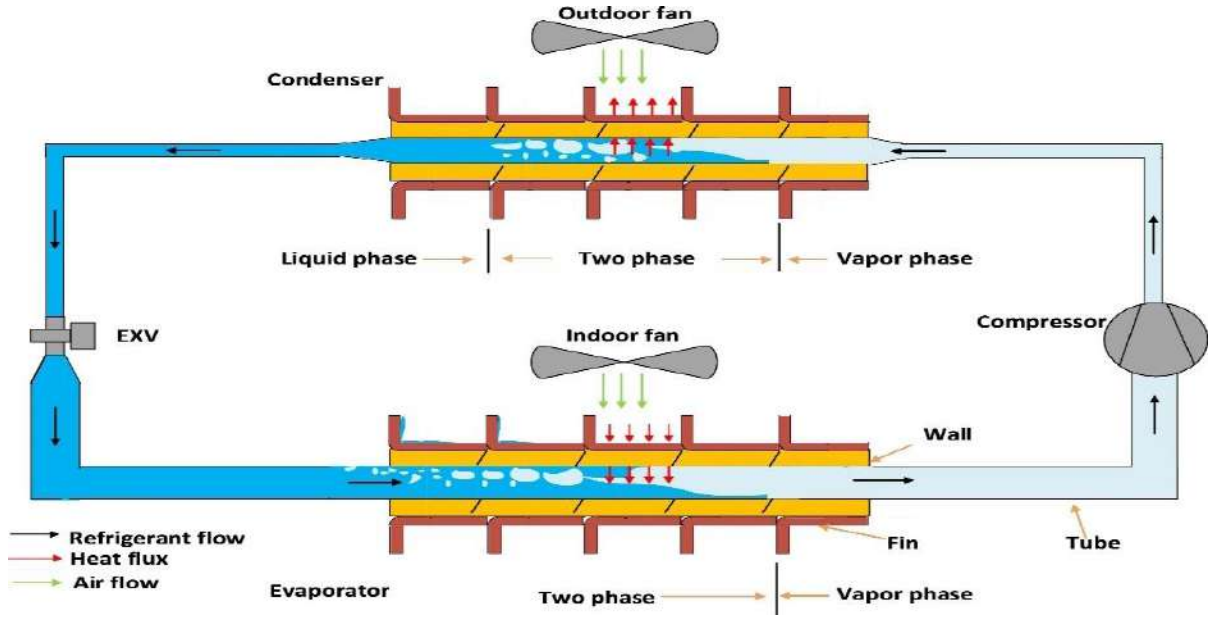
Figura 2
Tipos de compresores



***Nota**: Se clasifica mediante su principio de funcionamiento. Fuente* (Fernandez DIez, 2000).

Principio de funcionamiento, de acuerdo con la regla de compresión interna, un compresor consiste de dos elementos en forma de espiral unidos con pernos: una espiral fija y otra que es accionada por un motor. La espiral móvil se mueve sin contacto con los metales, a la vez que el aire se comprima en espacios cada vez más pequeños. La fuerza motora de la espiral rotatoria proviene de un cigüeñal de carrera corta que gira desplazándose excéntricamente al centro de su espiral fija. Esto produce una aspiración que atrae el aire a través de la abertura superior del dispositivo. Mientras se mueve al interior de la carcasa, el aire se comprime gradualmente, hasta alcanzar el puerto de salida y una válvula anti retorno. El refrigerante comprimido y presurizado se libera desde el puerto de salida del centro del conjunto mientras que la válvula anti retorno evita el retorno del refrigerante.

Figura 3
Diagrama de refrigeración



***Nota**: Se representa el proceso de refrigeración, fuente* (Chen, y otros, 2022).

Ciclo de compresión a vapor.

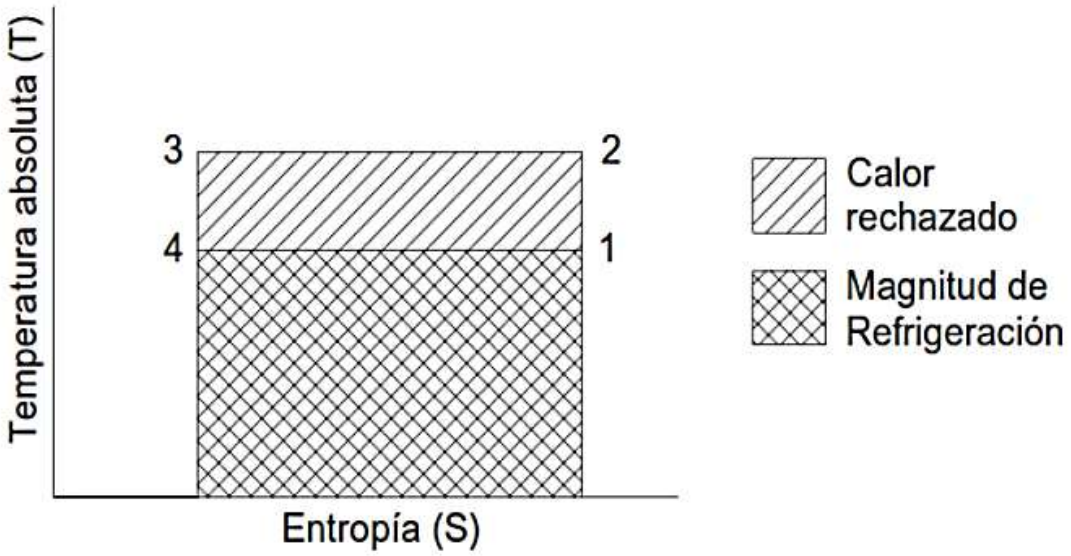
Conocido también como Ciclo de Carnot con un enfoque revolucionario para obtener refrigeración constante, guiando un refrigerante por cuatro etapas clave. Este condujo al desarrollo de la máquina térmica. Este ciclo se caracteriza por ser ideal y estar integrado por métodos reversibles, incluyendo un par de procesos isotérmicos y dos adiabáticos. Se considera que este ciclo tiene el potencial de lograr una eficiencia máxima que supera a la de cualquier ciclo real y sirve como referencia para estimar las temperaturas que permiten lograr dicha eficiencia máxima (Dongo Arrayan & Perez Olivera, 2020).

En la figura 4, se observan cuatro procesos del ciclo de Carnot:

- P1-P2 Compresión adiabática, sin fricción ni intercambio de calor (Compresor).
- P2-P3 Eliminación de calor de manera isotérmica, con el refrigerante manteniendo una temperatura constante (Condensador).
- P3-P4 Expansión adiabática, sin fricción ni intercambio de calor (Válvula de expansión).
- P4-P1 Absorción de calor a presión constante, con el refrigerante manteniendo una temperatura estable (Evaporador).

Figura 4

Diagrama de temperatura vs entropía para ciclo de Carnot.



***Nota**: Representa la magnitud del calor rechazado y la refrigeración. Fuente* (Wilbert F., 1998).

Ciclo de refrigeración estándar

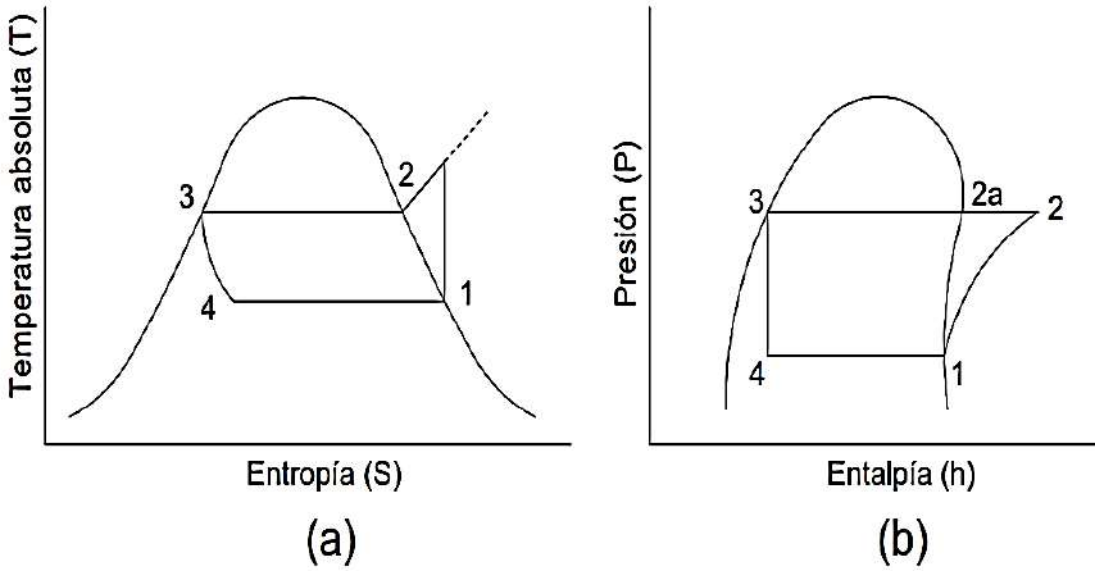
Este es un ciclo teórico que plantea *que el vapor refrigerante sale del evaporador* e ingresa al compresor como vapor saturado, a la temperatura y presión correspondientes a su vaporización. Asimismo, el líquido abandona el condensador y pasa al dispositivo de control de refrigerante como un líquido saturado, manteniendo las condiciones de presión y temperatura de condensación. Este enfoque permite simular un ciclo que se asemeja al funcionamiento real del sistema de refrigeración. Las diferencias entre este ciclo y el ciclo de Carnot se pueden identificar en las siguientes áreas (Dongo Arrayan & Perez Olivera, 2020):

- Temperaturas de condensación y evaporación no se alinean a los puntos caliente y frío.
- La expansión es isoentálpica, mas no isoentrópica
- El proceso de compresión p1-p2 ilustrado en la Figura 5, experimenta una compresión húmeda al tener el lugar completamente en la región bifásica, donde coexisten vapor y líquido.

Como resultado, el ciclo deja de tener una forma rectangular, lo que provoca un aumento en su área interna y, por ende, en la cantidad de trabajo total necesario.

Figura 5

Ciclo estándar de refrigeración, vapor-compresión



***Nota**: Comparación de* (a) diagrama temperatura-entropía y (b) diagrama presión-entropía*. Fuente* (Martínez Jimenez, 2005)*.*

Seguidamente, se explican con mayor detalle los siguientes procesos importantes en la refrigeración:

Sobrecalentamiento.

Se emplea como un proceso preventivo para disminuir posibles retornos de líquido, que puede ocurrir en la parte de entrada del compresor. Debido a que el vapor sobrecalentado tiene un volumen específico mayor, esto resulta en un aumento del flujo volumétrico que el compresor debe llevar. Además, al analizar la Figura 5, se observa que las curvas de compresión isotérmica tienen una razón decreciente conforme a que incrementa la temperatura de ingreso al compresor, lo que conlleva a un incremento en la potencia de compresión (Pc) requerida. Por lo tanto, este sobrecalentamiento afecta negativamente el coeficiente de rendimiento (COP) (Dongo Arrayan & Perez Olivera, 2020).

Subenfriamiento.

Cuando el fluido se subenfría al salir del condensador, la capacidad de enfriamiento por unidad de masa de refrigerante aumenta, lo que resulta en una reducción del caudal y un incremento de la potencia frigorífica (Qf), mejorando así la eficiencia del ciclo (Dongo Arrayan & Perez Olivera, 2020).

2.2.4 Principales Fallas

Tabla 1

Principales fallas del sistema de aire acondicionado de precisión

Falla	**Variables con las que se puede evaluar**
Fugas de en el sistema	Presión, temperatura, corriente, voltaje
Sobrecarga de refrigerante	Presión, temperatura, corriente, voltaje
Subcarga de refrigerante	Presión, temperatura, corriente, voltaje
Perdida de la eficiencia volumétrica (fuga en válvulas)	Presión, temperatura, corriente, voltaje
Condensador sucio	Presión, temperatura, corriente, voltaje
Alimentador del filtro de aire sucio (flujo bloqueado)	Presión, temperatura, corriente, voltaje
Restricción en la línea de liquido	Presión, temperatura, corriente, voltaje
Restricción en la línea de vapor	Presión, temperatura, corriente, voltaje
Arranque inundado/ golpe de liquido (análisis de transitorios anómalos)	Frecuencia, corriente, voltaje
Gas no condensable	Presión, temperatura, corriente, voltaje
Retorno de liquido	Presión, temperatura, corriente, voltaje
Ciclos cortos	Frecuencia, corriente, voltaje
Desequilibrio por carga variable (fan condensador / fan evaporador)	Frecuencia, corriente, voltaje
Ingestión de líquido al compresor	Presión, frecuencia, corriente, voltaje

Nota. Fuente: (Laughman , Armstrog, Leeb, & Armstrong, 2006).

2.3 Mantenimiento Industrial

El mantenimiento se refiere a una serie de prácticas o procedimientos destinados a prolongar la duración operativa de un sistema, garantizar su funcionamiento con un mínimo costo y asegurar un alto nivel de seguridad.

La relevancia de este mantenimiento radica en la imperiosa búsqueda de las empresas para mantener sus sistemas y maquinarias operativas de manera permanente y eficiente. Y se conocen distintos tipos, como:

A) Mantenimiento Correctivo

Hace referencia a un grupo de acciones orientadas a reparar el equipo después de que ha cesado su funcionamiento. Este mantenimiento puede ser programado o inesperado, y generalmente implica gastos considerables (Contreraz Alvarez, 2020).

B) Mantenimiento Preventivo

Se trata de un set de actividades dirigidas a disminuir la posibilidad de fallo y deterioro de un sistema. Surgió como una respuesta a las limitaciones del mantenimiento correctivo. Esta estrategia se fundamenta en la realización de inspecciones preventivas de equipos antes de que se presenten fallos, siguiendo intervalos de tiempo previamente establecidos para prevenir los fallos antes de que se produzcan. La determinación de estos intervalos se realiza mediante un análisis estadístico basado en los datos históricos de eventos disponibles. Es un mantenimiento planeado que conlleva costos considerables

C) Mantenimiento Predictivo

La definición ampliamente aceptada de mantenimiento predictivo implica supervisar de manera regular las condiciones mecánicas, operativas, de eficiencia y otros indicadores relacionados con el funcionamiento de los sistemas de procesos. Este monitoreo proporciona la información necesaria que permiten extender al máximo los intervalos entre reparaciones y reducir tanto la frecuencia como el costo de las interrupciones no programadas debido a fallos (Calderon Solano, 2021).

2.4 Sistema de Diagnóstico y Detección de Fallas

Este sistema, en general consta de dos etapas principales, el de instrumentación y el de detección.

2.4.1 Subsistema de Instrumentación

En esta etapa se contempla, los conceptos, principios, requerimientos y procedimientos. Sobre la parte instrumental del sistema, para los diferentes tipos de sensores que se necesita en este trabajo. Con el objetivo de obtener las señales de estado como presión, corriente, tensión y temperatura.

2.4.2 Subsistema de Detección

En esta etapa se contempla, los conceptos, características y requerimientos sobre el sistema de adquisición, donde se evalúa diferentes tipos de microprocesadores, microcontroladores, circuitos para acondicionar las señales de los sensores. Con la capacidad necesaria para generar un repositorio de datos. Y consecuentemente, gracias a los modelos se pueda detectar y diagnosticar las fallas.

2.4.3 Sensores y Transductores

Son dispositivos que tienen la capacidad de transformar un tipo de energía en otro. Y comúnmente, se suele confundir la distinción entre transductores y sensores; sin embargo, la diferencia entre ellos radica en su aplicación y eficiencia. Los sensores se ocupan de transformar una naturaleza de energía en otra sin considerar necesariamente la eficiencia, con el propósito de medir fenómenos físicos y generar una salida cuantificable que varía en función del fenómeno que están registrando. Por otro lado, los transductores no necesariamente convierten la energía en forma eléctrica y requieren eficiencia en su conversión, debido a que su aplicación suele ser directa en actuadores o dispositivos de visualización (Calderon Solano, 2021).

A) Sensor de Temperatura

Figura 6
Sensor MLX90614

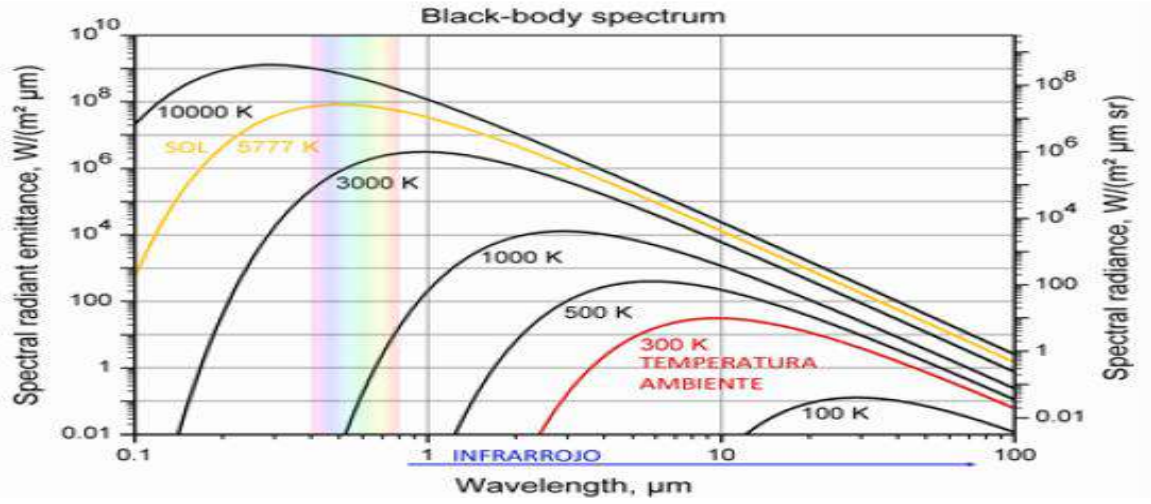
Nota: *Representa un sensor de temperatura de superficie por infrarrojo, fuente sensovant.*

Es un sensor infrarrojo para medir temperatura sin necesidad de contacto directo y abarcar un ángulo de visión de 90 grados, se puede obtener la temperatura promedio en una región específica. Estos sensores pueden ser integrados con dispositivos como Arduino o autómatas para detectar la temperatura de un objeto sin contacto. La comunicación se lleva a cabo a través del protocolo I2C, mide temperaturas en un rango de -70 ºC hasta +380 ºC. Además, integra amplificadores de bajo ruido, un conversor analógico-digital de 17 bits y una resolución de temperatura 0,02 ºC (Reyes Sosa, 2018).

El fundamento radica en la ley de Stefan-Boltzmann, que establece que cualquier cuerpo con una temperatura superior al cero absoluto (0 ºK) emite radiación, cuya distribución espectral está directamente relacionada con su temperatura. El sensor MLX90614 capta esta radiación y produce una señal eléctrica que representa la temperatura de los objetos (Reyes Sosa, 2018).

Figura 7

Espectros de objetos negros

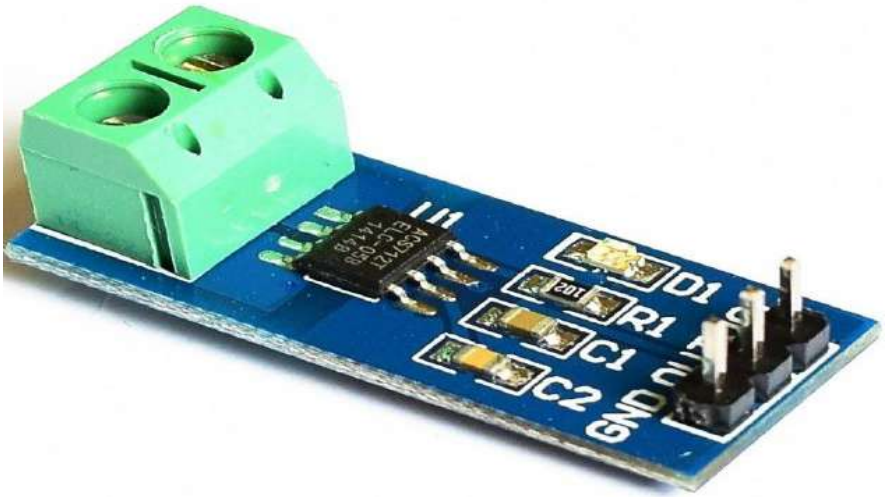


Nota: *Diagrama entre la longitud de onda y espectro de la emitancia, fuente* (Reyes Sosa, 2018).

Esto implica que los sensores con un ángulo de visión reducido son ideales para realizar mediciones específicas directamente en frente del dispositivo. En cambio, los sensores con un ángulo de visión más amplio sirven más para identificar aumentos de temperatura en zonas (Reyes Sosa, 2018).

B) Sensor de Corriente

Figura 8
Sensor de corriente



Nota: *Representa un sensor de corriente ACS712, fuente Sparkfun electronics.*

Descubierto por Edwin C. Hall en 1879, el efecto Hall describe la generación de una diferencia de voltaje en un conductor al aplicar una corriente eléctrica bajo la influencia de un campo magnético externo. Este fenómeno se manifiesta únicamente cuando el campo magnético es perpendicular al flujo de la corriente (Gutierrez & Ituralde, 2017).

Sensor Hall ACS712.

Está conformado por un dispositivo de efecto Hall lineal de alta precisión y bajo desplazamiento, acompañado de una línea conductora de cobre ubicada cerca de la superficie del chip. La corriente que atraviesa esta línea genera un campo magnético, que es detectado por el circuito integrado Hall y transformado en una tensión proporcional. Este dispositivo opera en un intervalo de voltaje de 4,5 V a 5,5 V y funciona con sistemas de alimentación de 5 V (Reyes Sosa, 2018).

Figura 9

Cuadro de características de tipos de ACS712

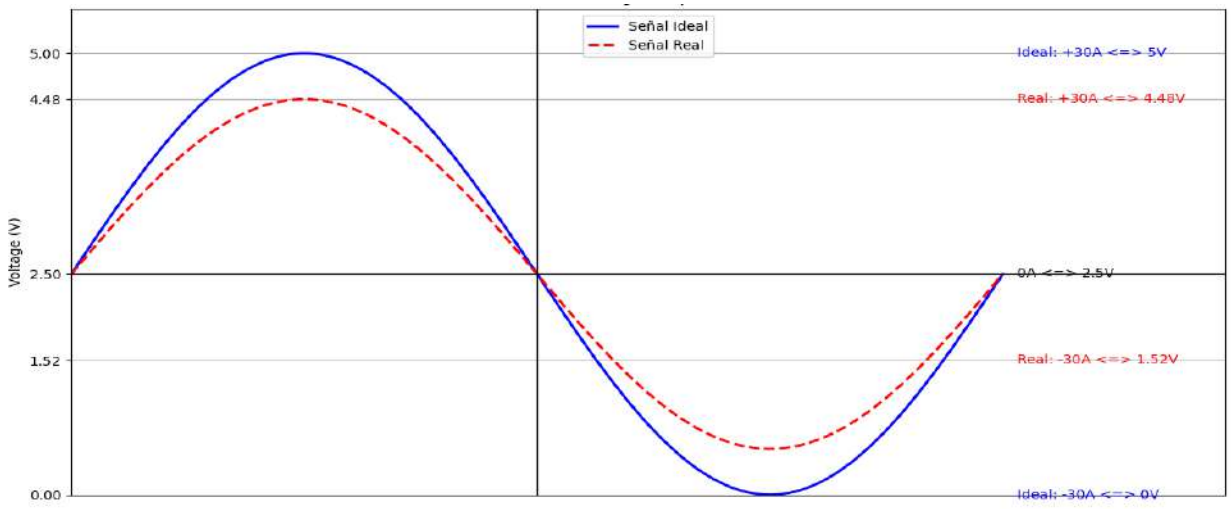
Número de Parte	Empacado	T_A $(°C)$	Rango Optimizado, $I_P(A)$	Sensibilidad (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	-40 a 85	±5	185
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	-40 a 85	±20	100
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	-40 a 85	±30	66

Nota: *Cuadro descriptivo de sensores ACS712,* fuente: (Briones Romero, 2023)

Para entender con mayor claridad cómo opera el sensor, consideremos que este dispositivo puede medir corrientes en un intervalo que va desde -30 A hasta +30 A. La salida del sensor, según las especificaciones del fabricante, siempre se encuentra en el rango de 0 a 5 voltios. En este rango, el valor de 0 A se representa por los 2.5 V, y por cada amperio adicional, el voltaje cambia en 66 mV. Es importante destacar que este sensor puede realizar mediciones tanto en corriente continua (CD) como alterna (CA). Si analizamos una forma de onda sinusoidal, el valor mínimo corresponde a -30 A, lo que significa que los 2.5 V de salida equivalen a 0 A, mientras que los +30 A se representan por los 5 V.

Figura 10

Equivalencia del voltaje de salida del sensor ACS712 30Amp-66mV/A



C) Sensor de Voltaje

Se trata de un dispositivo transformador de voltaje diseñado para proporcionar una salida activa monofásica. Incluye un circuito integrado con un amplificador operacional para corregir cualquier

desviación en la salida analógica. Este dispositivo está capacitado para medir voltajes de baja tensión, siendo adecuado para aplicaciones de hasta 220 V. Además, cuenta con un potenciómetro integrado en la placa, con un rango ajustable de 0-10 kΩ, que ofrece setear la magnitud de la señal de salida según las necesidades específicas (Briones Romero, 2023).

Puedes emplear cualquier microcontrolador que admita entrada analógica para captar el voltaje en tiempo real y realizar cálculos de energía. Para obtener resultados más precisos, es recomendable calibrar la salida del sensor utilizando un voltímetro. El sensor se alimenta con una corriente continua de 5 V y proporciona una salida analógica que guarda relación con la señal sensada. Cabe resaltar que la señal de salida tiene un desplazamiento (offset) equivalente a 2.5 V cuando no hay tensión de entrada, es decir, está conectada a 0 V (Briones Romero, 2023).

Figura 11
Sensor ZMPT-101 B

Nota: *Representa un embebido sensor de voltaje, fuente Sparkfun electronics.*

D) Protocolos de Comunicación

El bus i2C.

El estándar I2C (Inter-Integrated Circuit) fue concebido por Philips en 1982 con el propósito de facilitar la comunicación entre sus dispositivos. Con el tiempo, este estándar fue adoptado por otros fabricantes y consolidado como una norma de uso extendido (Gutierrez & Ituralde, 2017).

También conocido como TWI (Interfaz de Dos Cables) por razones de licencia, el estándar I2C ya no está sujeto a restricciones de uso desde que la patente caducó en 2006. Su operación requiere

únicamente dos cables: para la señal de reloj (CLK) y para la transmisión de datos (SDA), es sincronico, lo que lo diferencia ventajosamente del bus SPI. Sin embargo, su implementación implica una complejidad ligeramente mayor, así como la necesidad de una electrónica específica (Reyes Sosa, 2018).

En el bus, tienen una dirección única cada dispositivo que se utiliza para acceder a este individualmente. Dicha dirección puede establecerse físicamente (hardware), permitiendo normalmente ajustar los últimos 3 bits mediante interruptores, o puede configurarse completamente por algoritmo. Es esencial que cada dispositivo conectado al bus cuente con una dirección exclusiva (Reyes Sosa, 2018).

E) Transductor de Presión

Figura 12

Transductor de Presión Manométrica

Nota: *Representa al sensor de presión manométrica, fuente CAREL Industries.*

El transductor de presión convierte la señal de presión interna del sistema de aire acondicionado de precisión en una señal eléctrica. Donde un diafragma flexible se deforma al ser sometido a la presión interna del circuito cerrado. Esta deformación genera un cambio en la resistencia eléctrica de unos strain gauges dispuestos en un puente de Wheatstone, produciendo una salida analógica proporcional a la diferencia entre la presión interna y la presión atmosférica, es decir, la presión manométrica. Los sensores utilizados, con salidas analógicas típicas de 0.5 a 4.5 V, están diseñados para aplicaciones industriales HVAC/R, ofreciendo una gran estabilidad en la señal y una alta inmunidad EMC/EMI, lo que los convierten en un excelente transductor que cumple con los requisitos industriales (Fernandez DIez, 2000).

F) Transductor de Temperatura

Figura 13
Transductor de temperatura en capsulado

Nota: *Representa al sensor digital de temperatura, fuente Maxin Integrated.*

El DS18B20 es un sensor de temperatura digital altamente integrado, con un diseño que facilita la recolección de datos térmicos. Su interfaz 1-Wire permite conectar múltiples dispositivos a un solo pin, optimizando el uso de recursos microcontroladores. Con un rango de medición que abarca desde temperaturas desde -55°C hasta 125°C, este sensor es idóneo para una gran variedad de aplicaciones. Su precisión, resolución y robustez, lo convierten en una elección popular para proyectos de automatización, monitoreo ambiental y control de procesos industriales, mientras que su resistencia a interferencias electromagnéticas garantiza una operación confiable en entornos exigentes (Acondicionado, 2019).

2.5 Procesamiento de Señales

Se refiere al grupo de métodos utilizadas para medir, analizar y manipular señales. Estas señales, que suelen ser de naturaleza analógica, pueden ser procesadas de diversas formas. Una de ellas es mediante el procesamiento analógico, donde las señales analógicas se manipulan directamente a través de circuitos integrados, filtros y más sistemas similares. Sin embargo, este enfoque presenta la limitación de requerir circuitos electrónicos físicos, lo que puede resultar en dispositivos más grandes y complejos a medida que se incrementa la complejidad del procesamiento.

Por otro lado, para los de naturaleza digital, que implica la conversión de una señal analógica a una forma digital, permitiendo así su análisis como datos binarios sobre un microprocesador. En esta

forma se ajustan fácilmente por algoritmo, a diferencia del procesamiento de la naturaleza analógica, donde las operaciones están inherentemente vinculadas a circuitos electrónicos (Salazar, 2018).

2.5.1 Conversión Analógica-Digital

Este proceso implica la conversión de la naturaleza de la señal y se lleva a cabo a través de dispositivos conocidos como convertidores analógico-digitales (ADC). Consta de tres etapas principales:

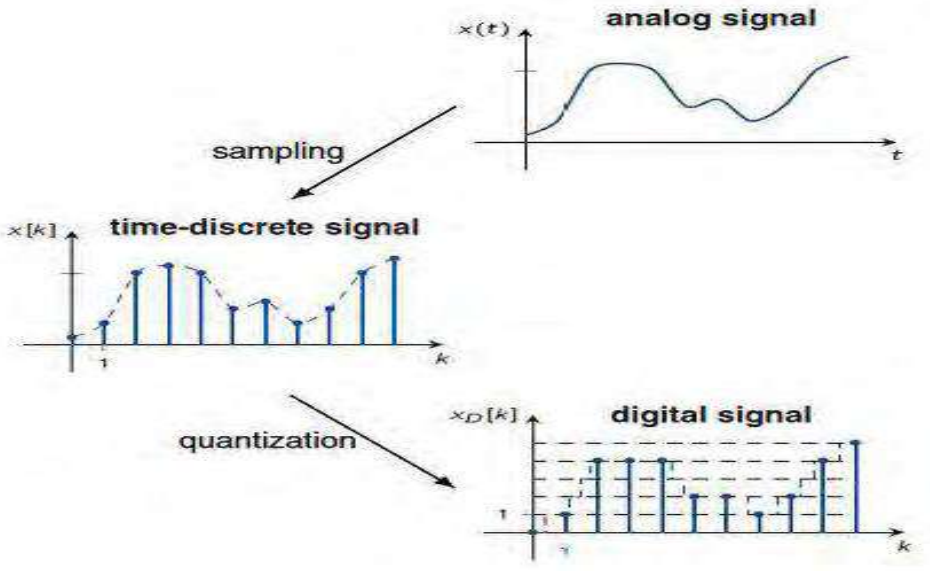
Muestreo: Esta etapa capta una serie de muestras discontinuas de una señal analógica. La cantidad de muestras captadas por segundo es denominada tasa de muestreo (Hz): a una tasa de muestreo más alta, mayor será la resolución. El número total de muestras depende de su duración y resolución. Ejemplo, una señal de duración t seg y se muestrea a una f Hz, el total de muestras= t * f.

Cuantificación: En esta parte la señal ya tiene formato discreto, que se caracteriza por estar representada en el tiempo mediante puntos separados, aunque su amplitud sigue siendo continua. El proceso de convertir esta señal discreta en una señal digital se conoce como cuantificación, el cual implica la representación del valor de amplitud de cada, en lugar de infinitos (Gutierrez & Ituralde, 2017).

Codificación: El procedimiento de codificación implica la representación de cada valor discreto de la señal digital por medio de una pila de bits binarios (Gutierrez & Ituralde, 2017).

Figura 14

Proceso de digitalización (muestreo y cuantificación)



Nota: *Representación del proceso de digitalización,* *f*uente: (Briones Romero, 2023)

2.5.2 Calibración de Sensores

Es esencial para avalar la exactitud y fiabilidad de la información recolectada esto implica ajustar y validar la respuesta del sensor para obtener mediciones precisas y coherentes en relación con el fenómeno monitoreado. Este proceso incluye realizar pruebas controladas y comparar las lecturas con un estándar conocido y confiable. Durante la calibración, es posible ajustar la sensibilidad, la linealidad y la compensación de temperatura para reducir errores sistemáticos y optimizar la exactitud de las mediciones. Es crucial llevar a cabo la calibración en condiciones ambientales representativas del entorno operativo real. La calibración periódica de los sensores es indispensable para garantizar una precisión continua y detectar cualquier variación o deterioro en su rendimiento con el tiempo.

2.6 Base de Datos

Un set de datos (dataset) se deriva del evento que se desea modelar. Este conjunto consiste en instancias identificadas por un conjunto específico de cualidades, para facilitar su interpretación y uso, los datos se estructuran en forma de tablas, organizados en columnas y filas, donde las filas representan instancias y las columnas denotan las características asociadas (Guamán Buestán, 2019).

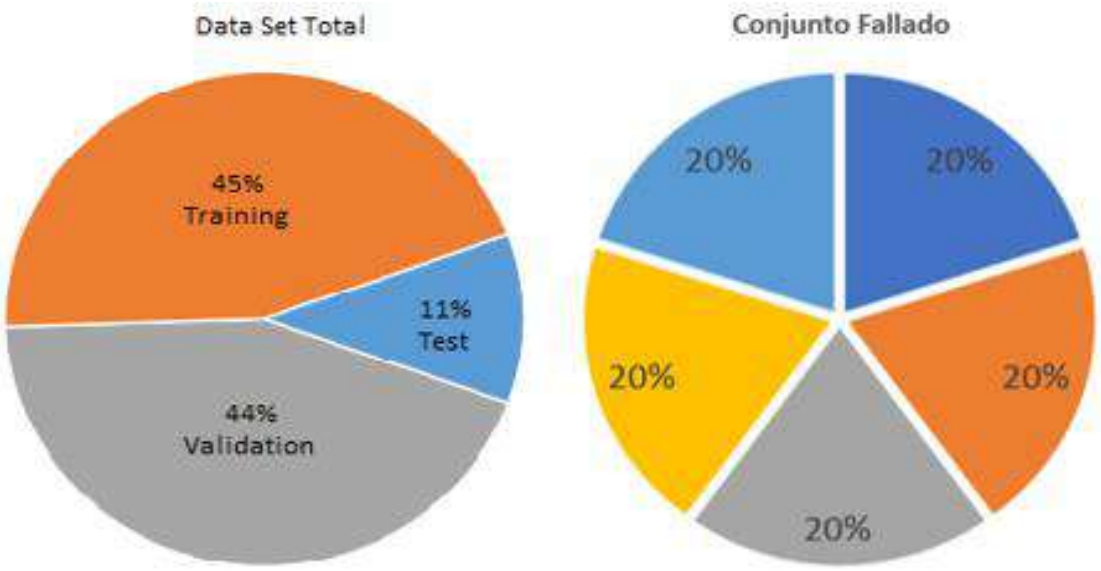
2.6.1 Generación de Data Set

Antes de comenzar el proceso de modelado, es esencial preparar los datos, lo que implica investigar las conexiones entre las diversas características y borrar posibles errores que puedan haber surgido durante la adquisición de los datos. Además, se requiere dividir los datos en subconjuntos para las etapas de construcción, validación y test. Una técnica comúnmente empleada es el "holdout", que implica la división del conjunto en subconjuntos de entrenamiento y prueba. Aunque la literatura sugiere una relación estándar de 70% de entrenamiento y 30% de prueba, la distribución puede variar según la disponibilidad de datos y los requisitos específicos de cada caso. En la situación en que ciertos modelos necesiten la optimización de hiperparámetros, se aconseja crear un subconjunto de validación adicional.

Es fundamental destacar que cada división de los datos debe ser aleatoria (al azar) para asegurar la representatividad de los subconjuntos (Guamán Buestán, 2019).

Figura 15

Representación de data sets



Nota: *Aplicación de train_test_split a la data set****,*** *f*uente: (Briones Romero, 2023)

2.6.2 Manejo de Base de Datos en MySQL en el Sistema de Adquisición

Implica la organización y almacenamiento estructurado de información digital. Estos sistemas utilizan tablas, organizadas en filas y columnas, para representar atributos específicos de los datos recolectados, tales como mediciones o eventos. SQL (Structured Query Language trabaja con datasets de datos relacionales, concediendo la realización de funciones como inserción, actualización, eliminación y consulta de datos. Las consultas SQL se emplean para extraer información relevante de los sensores, generar informes o realizar análisis. La conexión a este sistema es esencial para enlazar la plataforma y la base de datos en sí, mientras que la inserción, actualización y eliminación de datos son procesos fundamentales para agregar, modificar o eliminar información. Un manejo efectivo del conjunto de datos asegura la integridad, disponibilidad y rendimiento de los datos recolectados.

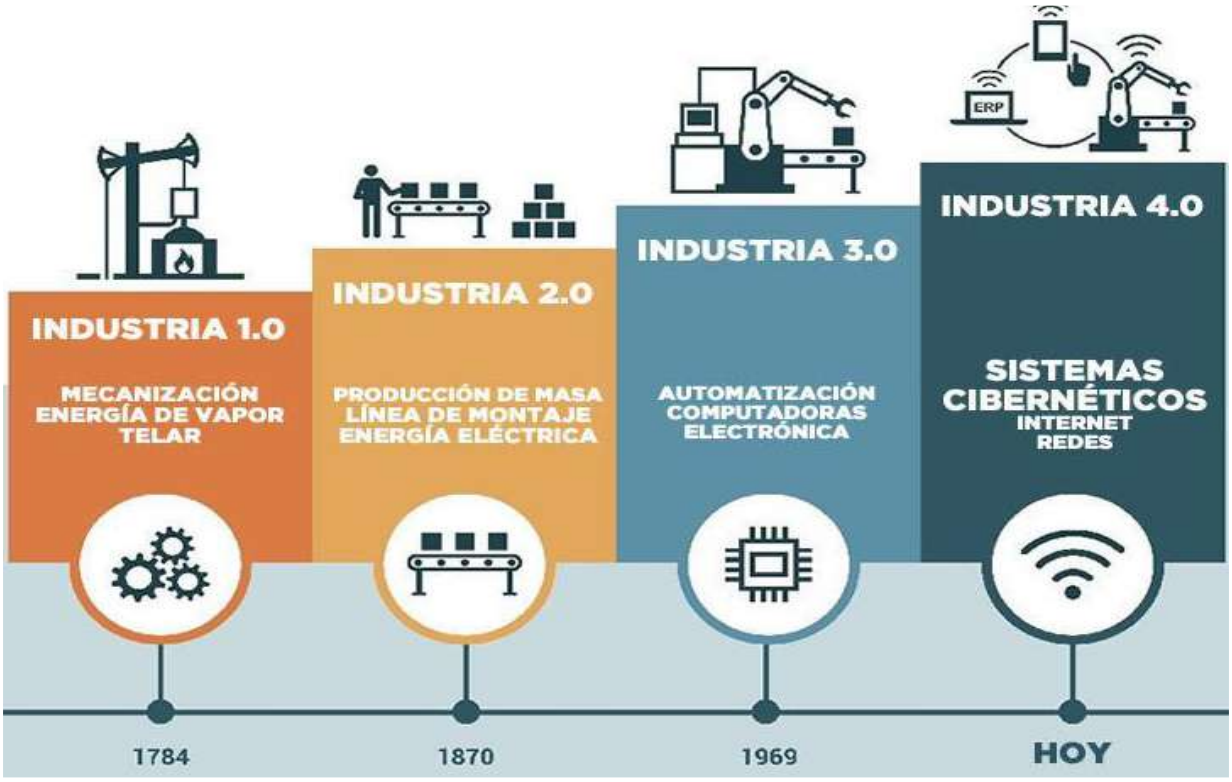
2.7 Inteligencia Artificial

Hablando del contexto de la Industria 4.0, el gobierno de Alemania inició una estrategia en Hannover Messe 2011. Esta etapa, es promovida por la transformación digital, significa un cambio cualitativo como las industrias gestionan la cadena de valor (Rojko, 2017).

Se caracteriza por la interconexión de personas, cosas y sistemas con el objetivo de ofrecer información útil en tiempo real. La aplicación de fundamentos que implica la optimización permanente de procesos mediante la depuración de desperdicios y la mejora de la eficiencia y la producción en tiempo real, ha resultado en una disminución de los gastos de manufactura. Esta tecnología ofrece una amplia variedad de soluciones para lograr aún mayores ahorros. Las empresas que la empleen de forma eficiente puedan experimentar una reducción de los costos de hasta un 40 % en los próximos años (Laguens, 2018).

Figura 16

La revolución en el tiempo



Nota: *La figura representa las características de cada revolución industrial en el tiempo, adaptada de* (Maisueche Cuadrado, 2019).

Modelos de Predicción

La evaluación predictiva es parte de la minería de datos que hace referencia al trabajo de obtener información de las señales para predecir tendencias, las relaciones entre las variables, el conocimiento inducido y patrones, pudiendo utilizarse en eventos desconocido y en cualquier momento. Se emplea la evaluación de datos para establecer funciones entre variables de situaciones pasadas con el fin de pronosticar resultados futuros. La exactitud de tales predicciones varía dependiendo del tipo de análisis empleado, así como del número y calidad de las suposiciones realizadas (Espino Timon, 2017).

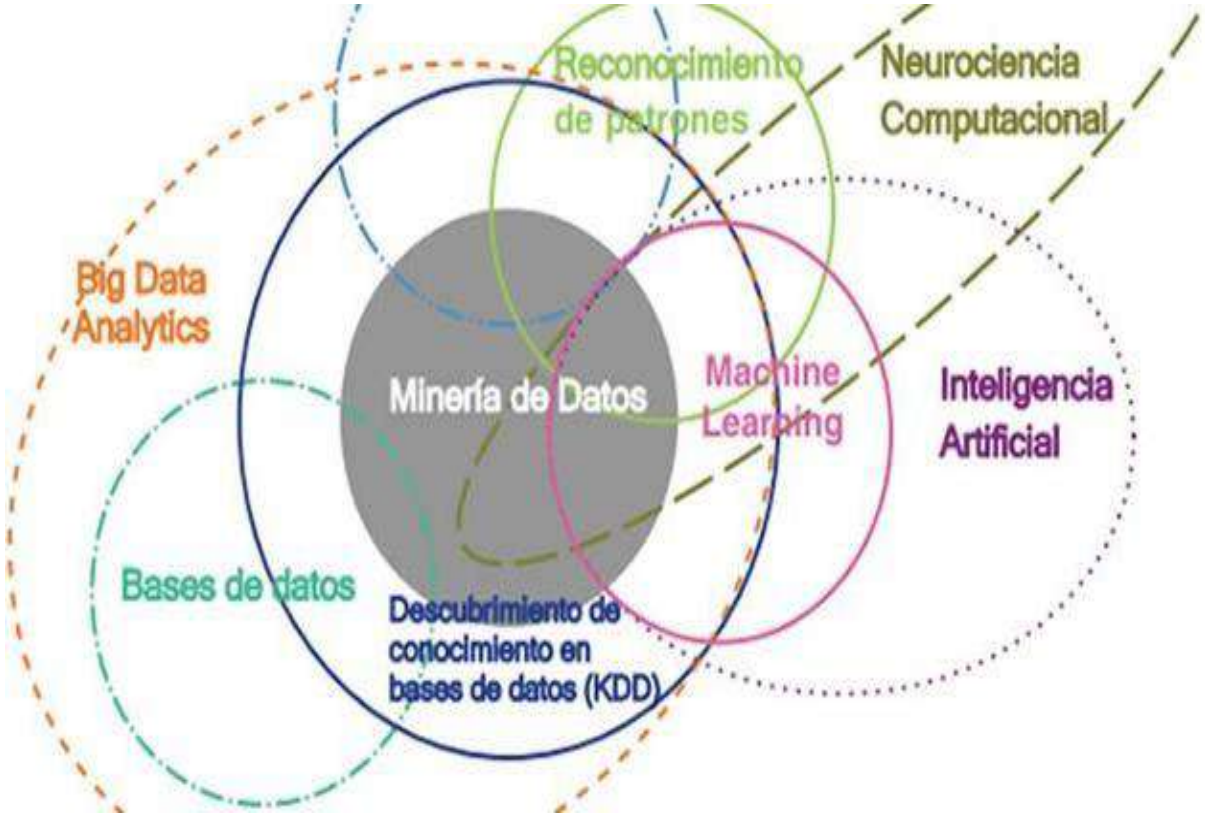
Cabe señalar que el modelo de predicción nunca dará el 100% de éxito y muchas veces estará lejos de estos resultados. A pesar de que el nivel de acierto sea bajo, es preferible utilizar un modelo para predecir (Espino Timon, 2017).

2.8 Machine Learning

Conforma una parte de la inteligencia artificial que ha estado en desarrollo durante varias décadas, pero su uso es más común para el público en general en años recientes. Según la definición de Arthur Samuel en 1959, consiste en que los procesadores sean capaces de aprender sin una programación explícita, lo que significa que pueden establecer sus propias reglas y aprender de los datos proporcionados durante el entrenamiento. Este proceso implica suministrar una gran cantidad de datos a la computadora para que aprenda y se entrene, lo que resulta en un modelo capaz de realizar tareas específicas o hacer predicciones (Guamán Buestán, 2019).

Existen diversos algoritmos diseñados para tareas específicas. Uno de los enfoques comunes es el Aprendizaje Supervisado, que implica un proceso de entrenamiento supervisado por humanos, dividido en fases de entrenamiento y evaluación (Guamán Buestán, 2019).

Figura 17
Machine learning (intercepciones)



***Nota**: La figura representa Data Mining y su interrelación con otras tecnologías (Villén, 2019).*

El aprendizaje automático se basa en técnicas matemáticas para dotar de aprendizaje a las computadoras, permitiéndoles reconocer patrones y generalizar comportamientos basados en datos históricos decisiones sin la necesidad de una intervención humana. En este proceso, los datos históricos se organizan en variables predictoras (entradas) y variables predichas (salidas), como la predicción de fallos en máquinas. Es importante comprender la terminología utilizada en este campo, como características (variables de entrada), clases (etiquetas para las variables de salida) y clasificadores (funciones que asignan etiquetas a partir de características) (Guamán Buestán, 2019).

También resulta crucial familiarizarse con los términos empleados en estos aprendizajes, una característica se refiere a una variable de entrada, mientras que la clase representa la etiqueta o identificación asignada a las variables de salida y el clasificador es una función f que relaciona el conjunto de características con una etiqueta específica, presenta dos etapas (Guamán Buestán, 2019).

La etapa de entrenamiento consiste en dar al código un set de datos que incluye tanto las entradas como las etiquetas correspondientes a las salidas esperadas (también conocidos como "datos etiquetados"). Estos datos se estructuran en un dataset, lo que permite al algoritmo aprender a reconocer los patrones presentes en ellos y a asociar las etiquetas adecuadas a esos patrones. Este conjunto de datos también se conoce como conjunto de entrenamiento. Como producto de la etapa de entrenamiento, se obtiene un modelo de Machine Learning.

En la etapa de evaluación, que se lleva a cabo después de haber obtenido el modelo en la fase anterior, se suministra al algoritmo un dataset no etiquetados que son diferentes de los usados en la etapa de entrenamiento. En esta parte, el modelo etiquetara a este nuevo conjunto de datos utilizando todo lo aprendido durante la etapa de entrenamiento. Al conocer las etiquetas reales, es factible realizar una evaluación del performance o la eficacia de este modelo. Por ejemplo, consideremos un modelo que ha sido entrenado con eventos de dos tipos etiquetas: positivas (por ejemplo, fallas) y negativas (por ejemplo, normales).

2.8.1 Ventajas de Machine Learning

Una de las mayores ventajas es su habilidad para manejar grandes volúmenes de datos de forma rápida y eficaz, lo que facilita la adquisición y el análisis constante de las señales. Esta rapidez y automatización son especialmente beneficiosas para detectar y prevenir fallas en sistemas críticos como de un aire acondicionado de precisión. Al utilizar machine learning para evaluar los valores de sensores, es posible reconocer patrones y anomalías que podrían ser indicio de problemas potenciales previo a que se vuelvan en fallas catastróficas. Además, el machine learning permite mejorar continuamente sus modelos predictivos a más datos. En resumen, el machine learning ofrece una herramienta poderosa para optimizar la monitorización y el mantenimiento de sistemas de adquisición de señales analógicas, mejorando así la fiabilidad, eficiencia y seguridad de los sistemas críticos (Maisueche Cuadrado, 2019).

2.8.2 Aplicaciones de Machine Learning

El machine learning es crucial en los sistemas de detección y diagnóstico de fallas, proporcionando numerosas aplicaciones que incrementan tanto la eficiencia como la confiabilidad en los trabajos industriales. Además del mantenimiento predictivo, donde los modelos analizan datos de sensores para anticipar posibles fallos y programar intervenciones proactivas, se aplica en el monitoreo de la calidad del producto. Aquí, los modelos pueden identificar patrones en las señales de condición de estado en tiempo real y detectar anomalías que podrían indicar un deterioro en la calidad del producto, permitiendo intervenciones tempranas para corregir problemas y asegurar la capacidad con procedimientos de calidad (Scikit learn, 2007-2024). Otro campo de aplicación importante es la optimización de procesos, donde los algoritmos pueden analizar grandes volúmenes de datos operativos para diagnosticar situaciones de mejora en la eficiencia y la productividad. Al recomendar ajustes en tiempo real, el machine learning ayuda a maximizar la producción y minimizar el desperdicio, contribuyendo así a la rentabilidad y competitividad de las operaciones industriales. En conjunto, estas aplicaciones demuestran el valor del machine learning para optimizar de mejora continua de los procesos industriales, permitiendo una toma de decisiones más informada y eficaz para enfrentar los desafíos de la producción moderna (Management Solutions, 2018).

2.8.3 Clasificación de Machine Learning

Las principales categorías en Machine Learning son:

Aprendizaje no supervisado (Unsupervised Learning)

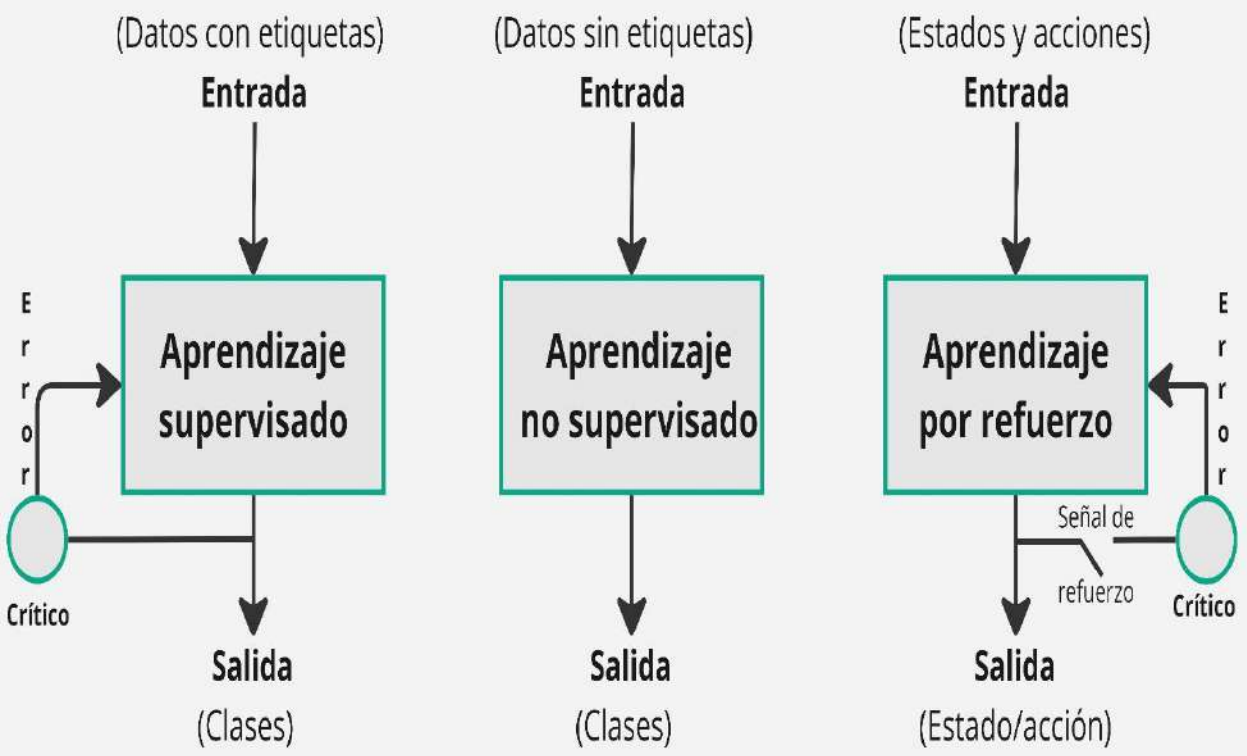
Aprendizaje Supervisado (Supervised Learning)

Aprendizaje Semi-supervizado (Semi-supervised Learning)

Aprendizaje por Reforzamiento (Reinforcment Learning)

Figura 18

Esquema de técnicas de aprendizaje más comunes



Nota: Adaptado del articulo cc-models de Ibm.

Aprendizaje Supervisado (Supervised Learning)

Requiere de datos etiquetados previamente para aprender cómo funciona. Esto se logra proporcionando al sistema un repositorio de datos previamente resueltos para que pueda aprender a abordar problemas futuros. Al examinar los valores de entrada como de salida, el modelo intenta definir una función que pueda explicar la proporción entre la salida y la entrada (Management Solutions, 2018).

Este enfoque de aprendizaje es el más ideal, de acuerdo al estilo de predicción, la naturaleza de la variable y el contexto en el que se aplicarán los resultados de la validación. Por ello, desarrollará con mayor enfoque los siguientes algoritmos que serán objeto de estudio: (Maisueche Cuadrado, 2019)

Se suelen encontrarse con 2 desafíos frecuentes durante su entrenamiento con datos, los cuales son (Management Solutions, 2018):

Sobreajuste (Overfitting): Sucede cuando un modelo ha sido entrenado con demasiada precisión en el subconjunto de entrenamiento, probablemente puede afectar negativamente su rendimiento para nuevos valores (Management Solutions, 2018).

Correlación (Correlation): Este problema aparece cuando el modelo a entrenar incluye características que están altamente interrelacionadas, como es el caso de los kilogramos y los gramos, los cuales muestran gran correlación. Hasta, una de estas variables puede influir casi por completo en el resultado (Management Solutions, 2018).

Se pueden distinguir dos categorías que abordan el Machine Learning, las cuales se expresan a continuación:

Clasificación: Proceso de anticipar etiquetas de clase, las cuales corresponden a opciones preestablecidas dentro de una lista. Este proceso se divide en clasificación binaria, que responde con opciones de no o sí, y clasificación multiclase, que involucra la asignación de más de dos clases (Contreras Alvarez, 2020).

Regresión: Se enfoca en anticipar un valor numérico, ya sea entero o decimal, en el contexto de las matemáticas y la programación. Por ejemplo, es posible estimar el ingreso semestral de una persona teniendo en cuenta variables como el nivel educativo y la ubicación geográfica (Contreras Alvarez, 2020).

Algunos de los algoritmos más comúnmente empleados son:

KNN vecinos más cercanos, Regresión logística, Árboles de decisiones (DT), Random forest (RF) y SVM.

Aprendizaje no Supervisado (Supervised Unlearning)

Este enfoque se emplea cuando no se cuenta con datos previamente etiquetados. En su lugar, se enfoca en la exploración de los datos para describir su estructura y ubicar posibles relaciones entre estos, permitiendo así agruparlos en conjuntos similares conocidos como "clusters". Actualmente es fundamental para el desarrollo del Deep Learning y se prevé que eventualmente pueda reemplazar al

Machine Learning. Entre los modelos más conocidos de este tipo de aprendizaje se encuentran KMeans y Análisis de Componentes Principales (PCA) (Contreras Alvarez, 2020).

En este proyecto, se aplican técnicas pertenecientes al "Aprendizaje Supervisado" con el fin de clasificar evento de fallas.

2.9 Métodos de Diagnóstico y Detección de Fallas

2.9.1 Métodos Basado en el Modelo

Se emplean los modelos explícitos de entrada-salida para predecir el comportamiento normal de un sistema y detectar posibles fallas con base en una pequeña cantidad de datos en tiempo real. Sin embargo, este enfoque puede resultar costoso, debido a que con lleva la ejecución de pruebas experimentales extensas y la incorporación de mediciones adicionales de entrada y salida, aumentando así la cantidad de sensores requeridos. Por consiguiente, el proceso de desarrollo de modelos robustos para este propósito se vuelven un trabajo arduo y de gran complejidad (Maisueche Cuadrado, 2019).

2.9.2 Métodos Basados en Data-driven

Desde una perspectiva contraria, los métodos de FDD basados en datos no necesitan un modelo específico del sistema, gracias a los avances recientes en sistemas de monitoreo, pueden aprovechar la abundancia de datos generados por los sensores. Además, estos enfoques basados en datos son fácilmente adaptables a sistemas con diversas estructuras. Sin embargo, el desempeño de estos métodos está sujeto fuertemente a la calidad de datos de entrenamiento y puede deteriorarse si no se comprende plenamente el funcionamiento del sistema (Maisueche Cuadrado, 2019).

2.9.3 Métodos Híbridos

Un enfoque híbrido combina lo mejor de ambos mundos al integrar modelos explícitos de entrada-salida con métodos basados en datos FDD. Al hacerlo, se aprovechan las ventajas de los modelos de entrada-salida para predecir el funcionamiento del sistema en condiciones normales de operación, al

tiempo que se utilizan los datos generados por los sensores para superar la capacidad de detección y diagnóstico de fallas (Maisueche Cuadrado, 2019). Este enfoque permite enfrentar lo complejo de los sistemas y estructuras variables, al tiempo que se optimiza el rendimiento del sistema de monitoreo y diagnóstico. Sin embargo, es crucial garantizar la precisión y fiabilidad de la información de entrenamiento y mantener un entendimiento profundo del funcionamiento del sistema para lograr resultados óptimos (Maisueche Cuadrado, 2019).

2.10 Clasificación Supervisada de Señales y Etapas

Tiene la función de distinguir entre distintos eventos, identificando de manera autónoma un tipo de evento frente a otro.

2.10.1 Adquisición e Identificación de los Datos

La fase de adquisición e clasificación requiere recopilar datos crudos en el área de interés y organizarlos para distinguir la información relevante. Durante esta etapa, se construye un pull de datos que representa las fallas que se pretenden etiquetar. En el proceso de marcado, una persona añade información crucial que será utilizada para el entrenamiento del modelo, lo que da lugar a que estos clasificadores sean denominados "supervisados" (Guamán Buestán, 2019).

Cuando se trata del tamaño adecuado de un conjunto de datos, no hay una cantidad exacta mínima o máxima de muestras establecida para garantizar la efectividad del entrenamiento. *A pesar de que se sugiere iniciar con un mínimo de 30 muestras, en la práctica* esta cantidad puede cambiar dependiendo de distintos factores, como el tipo y la variedad de los eventos, *además de la calidad de las muestras. Si las fallas a detectar presentan una alta diversidad* y se cuenta con un número limitado de muestras, o solo hay muestras de algunas variedades, podría ocurrir lo que se conoce como "Underfitting". Esto significa que el modelo evaluado no consigue aprender patrones ni generalizar el conocimiento. Por otro caso, si se cuentan con muchas de muestras, pero todas pertenecen a situaciones específicas, el algoritmo puede aprender a reconocer solo esas variedades particulares. En este caso,

cuando se enfrenta a una nueva variedad, el modelo puede fallar en clasificarla correctamente, lo cual se conoce como "Overfitting" (Guamán Buestán, 2019).

2.10.2 Pre-procesamiento de la Datos

Constituye una fase inicial previo de cualquier proceso o entrenamiento. Su propósito radica en transformar los datos de manera que resulten más manejables en las partes subsiguientes. Durante esta fase, se utilizan diferentes métodos como el filtrado, la depuración de datos y la reducción del volumen de información.

2.10.3 Extracción de Características y Definición del Training Dataset

Una vez finalizado el preprocesamiento y disponiendo de un conjunto de señales con sus etiquetas, todavía no es posible comenzar el entrenamiento (Guamán Buestán, 2019). Esto se debe a que el grupo de datos inicial está compuesto por segmentos de señales digitales, que principalmente son bits sin contenido informativo relevante. Por lo tanto, antes del entrenamiento del clasificador, es importante extraer la información pertinente de estas señales y setearlas en un grupo de datos, llamado "Training Dataset" la información relevante obtenida se denomina "características" o "features" (Guamán Buestán, 2019).

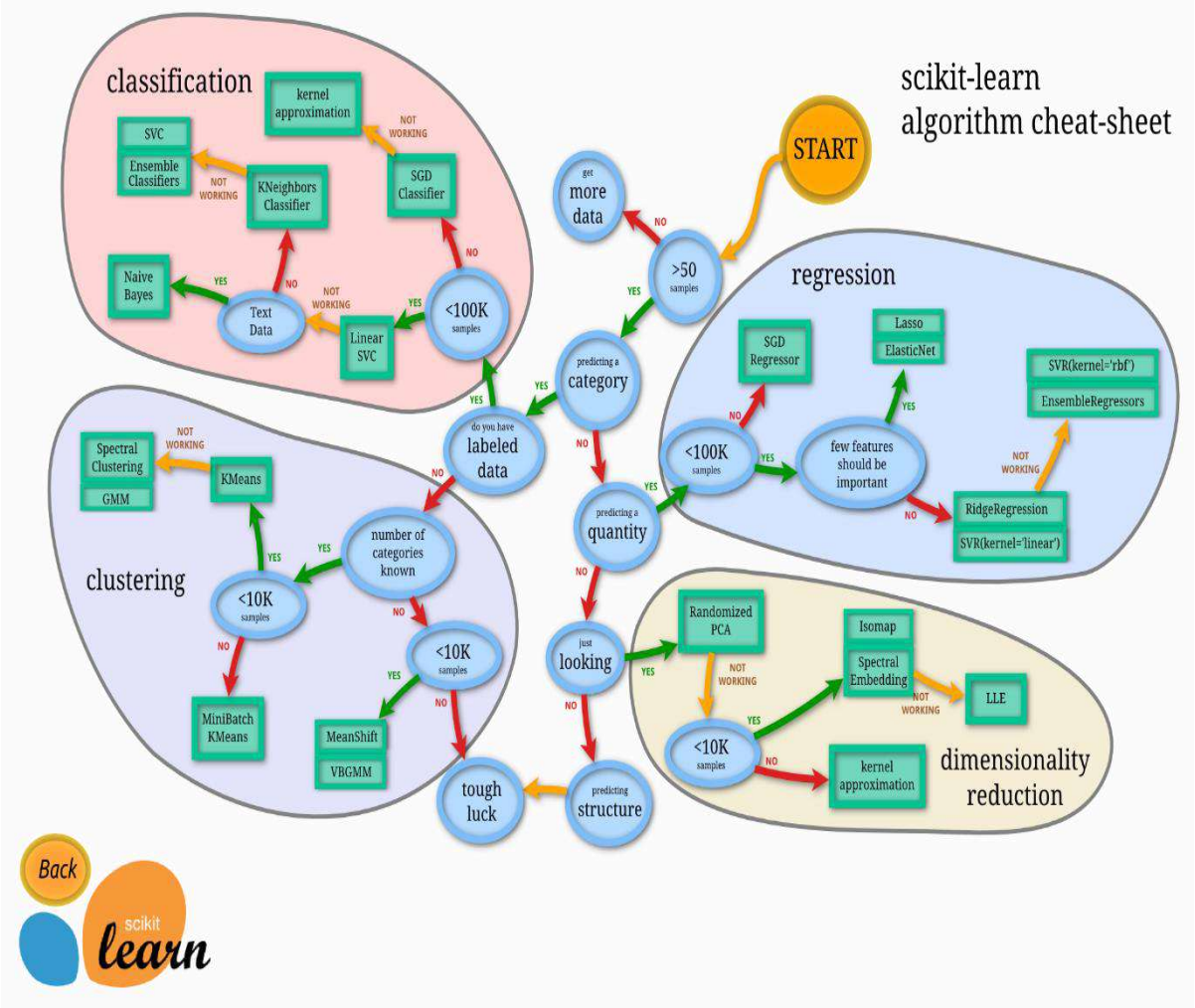
En el contexto del procesamiento de señales, una característica se define como un grupo de valores numéricos que encapsulan datos relevantes acerca de una señal particular. Un conjunto de características relevantes para una aplicación en particular se conoce como un "patrón de características". Este es esencial para identificar el evento específico de interés. Una tarea crucial en el desarrollo de un clasificador es determinar qué características formarán un patrón capaz de distinguir los eventos de interés de otros (Benites Condori, 2023).

2.11 Algoritmo de Machine Learning y Entrenamiento

La selección del modelo es de gran importancia y debe basarse en la tarea específica que se desea llevar a cabo. En el instructivo de la biblioteca Scikit-learn, se proporciona un diagrama claro y educativo que puede ayudar en la elección del algoritmo adecuado figura 19.

Figura 19

Esquema de aplicaciones de Scikit-learn



***Nota**: Diagrama de modelos de aprendizaje por su aplicación típica, fuente librería Scikit-learn.*

El proceso de entrenamiento de un modelo implica la selección de un algoritmo de ML apropiado. Entre los modelos más comúnmente utilizados se encuentran el método de los K-Nearest-Neighbor, la Support Vector Machine y el Desición Tree. Como se diagrama en la figura 19, estos últimos algoritmos

son particularmente adecuados para proyectos de detección. Es importante destacar que todo modelo presenta hiperparámetros ajustables, conocidos como hiperparámetros, que pueden condicionar la etapa de aprendizaje del modelo. La elección de estos parámetros Puede influir notablemente en la exactitud de los resultados. En el desarrollo siguiente, se detalla con una breve explicación de cada uno de estos algoritmos (Scikit learn, 2007-2024).

2.11.1 K Vecinos más Cercanos (K-Nearest-Neighbor)

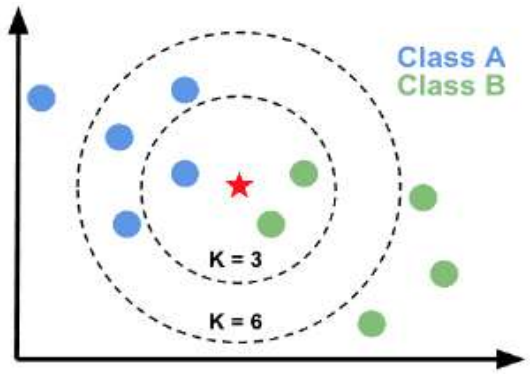
Es un método estadístico aplicable en la tarea de regresión como en la de clasificación dentro de un espacio multidimensional. Su utilidad abarca datos numéricos, discretos y categóricos. De un inicio no se preconcibe ninguna suposición sobre el esquema de donde se saca la muestra de modelado. que consiste en un subconjunto de entrenamiento compuesto por valores positivos y negativos. Para clasificar una nueva muestra, se calcula la proximidad al punto más proximo dentro del set de entrenamiento. La clasificación se basa en la proximidad de este punto. En este tipo de clasificador, se toman en cuenta los k puntos más próximos y la clasificación se define según la señal predominante de la mayoría (Gallegos Sánchez & Huachín Herrera, 2018).

El aprendizaje consiste en el almacenamiento de instancias de las cuales se conocen sus valores a futuro. Esto implica construir un modelo de aproximación f a partir de ejemplos existentes en una database histórica (conjunto de entrenamiento) para comparar con cada nuevo ejemplo y así predecir sus valores futuros. En vez de crear una única aproximación para el bloque entero de ejemplos donde se quiere predecir sus valores futuros, esta técnica genera aproximaciones locales a través de subgrupos del bloque de entrenamiento. Estos subgrupos están conformados por los k vecinos más cercanos al caso con el objetivo de prever su comportamiento en el futuro. Se fundamenta en la media de los valores futuros de los k vecinos, asignando distintos pesos que indican la relevancia de cada vecino, donde aquel que esté más cerca tendrá una mayor ponderación que el más lejano (Gallegos Sánchez & Huachín Herrera, 2018).

La selección del valor de K es fundamental y puede resultar decisiva para predecir correctamente una muestra específica. Por ejemplo, en la figura 20, los elementos blue y green representan los datos de entrenamiento, y el objetivo es determinar la clase de la estrella. Si se selecciona K=3, se asignará a la clase green, mientras que con K=6 se asignará a la clase blue. (Scikit learn, 2007-2024).

Figura 20

Espacio donde se muestra los puntos de entrenamiento



***Nota**: Representación de los puntos de entrenamiento correspondientes a dos (blue y green) y el punto, cuya clase se quiere predecir, fuente https://www.jcchouinard.com/k-nearest-neighbors/.*

Además de determinar el valor de K en el algoritmo (K-Nearest-Neighbor), También es fundamental determinar si todos los vecinos en el grupo de K más cercanos tienen un voto equitativo en la predicción de la clase, o si aquellos vecinos más cercanos tendrán un voto que posea más peso en la decisión final. Este parámetro ajustable, conocido como "weights", puede adoptar dos valores distintos: "uniform" o "distance" (Scikit learn, 2007-2024).

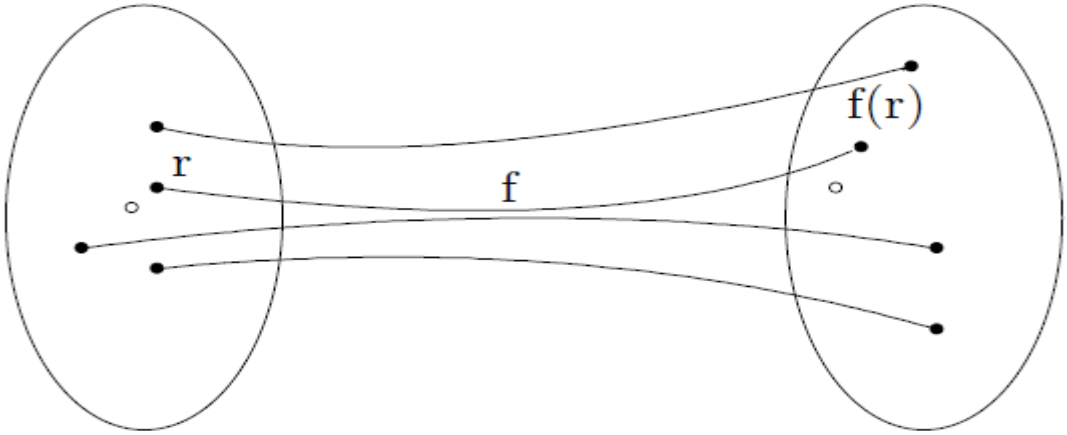
Otro hiperparámetro importante es la técnica empleada para medir la proximidad entre puntos. Hay varias opciones disponibles para este fin. La más utilizada es la distancia Euclidiana, que corresponde a la línea recta que enlaza dos puntos en un espacio de n dimensiones. Otra opción es la distancia de Manhattan, que simula el espacio entre dos puntos siguiendo un patrón en forma de cuadrícula. La distancia Minkowski, por su parte, es una generalización de las distancias Euclidiana (r = 2) y Manhattan (r = 1).

$$D.Euclidea: D(X,Y) = \sqrt{\sum_{i=1}^{n} |X_i - Y_i|^2}$$
$$D.Manhattan: D(X,Y) = \sum_{i=1}^{n} |X_i - Y_i| \quad (1)$$
$$D.Minkowski\ D(X,Y) = (\sum_{i=1}^{n} |X_i - Y_i|^r)^{1/r}$$

X y Y representan los dos puntos que se busca hallar la separación y n es la dimensionalidad de los puntos.

Figura 21

Aproximación local



Nota: *Representación de la idea geométrica para un único vecino, fuente* (Gallegos Sánchez & Huachín Herrera, 2018).

$$X_{fut} = [X_{t+1}, \dots, X_{t+n}](puntos\ de\ \mathbb{R}n)$$
$$X_{pas} = [X_t, X_{t-1}, X_{t-2}, \dots, X_{t-(m-1)}]\ puntos\ de\ \mathbb{R}m) \quad (2)$$

La predicción de n valores futuros de una serie temporal X_t:

$$X_{fut} = \frac{\sum_{i=1}^{vecinos}(X_{i-1} - X_i)}{\#\ vecinos} + X_{presente} \quad (3)$$

Si: X_i es el vecino del punto de $X_{presente}$ más alejado (está entre los valores pasados, X_{pas} organizados desde el más próximo hasta el más distante.

La función distancia es:

$$d^2(q,z) = \sum_{i=1}^{m}(q_i - z_i)^2 \quad (4)$$

El número de vecinos, determinar el K óptimo para que el error de una serie temporal fuera ínfima; se intenta, reducir la función objetivo cuadrática:

$$\sum_{j=1}^{p}(\hat{X}_{fut} - X_{fut})^2 \quad (5)$$

Acotación de error, predecir el error absoluto viene dado así:

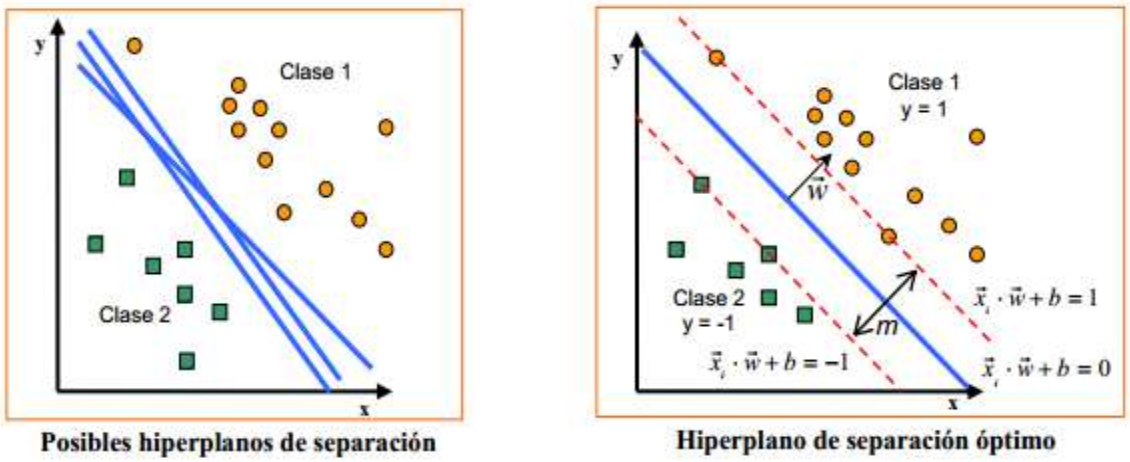
$$EA = \frac{\sum_{i=1}^{radio}\frac{(X_{fut+i} - X_{presente+i})}{X_{presente+i}}}{radio} \quad (6)$$

2.11.2 Maquina de Vectores de Soporte (Support Vector Machine)

Los vectores de soporte con el objetivo de los clasificadores discriminatorios es crear un plano en un espacio con N dimensiones (cantidad de variables independientes) con la intensión de extremizar la división de los datos pertenecientes a dos clases distintas. Este enfoque es para realizar clasificaciones binarias y evaluaciones de regresión, basadas en kernel. El hiperplano es una frontera que se usa para separar los datos y asignarles una clase diferente en cada lado (Gandhi, 2018).

Figura 22

Representación de Support Vector Machine



Nota: *Se ve el hiperplano elegido de entre todos los otros posibles, adaptado de RPubs by RStudio.*

Tanto para valores lineales como no lineales, junto con su importante precisión, aunque con el sacrificio de un bajo consumo de potencia computacional (Gandhi, 2018).

A continuación, los fundamentos teóricos del algoritmo. Consideremos un grupo de muestras de entrenamiento con etiquetas, como se muestra en la figura 22, donde cada punto es caracterizado por dos variables: (x1, y1), ..., (xi, yi), donde xi representa un punto en un espacio N-dimensional (xi ∈ ℜN) y yi es su etiqueta. Dado que SVM se enfoca en la clasificación binaria, generalmente se asigna yi = +1 para clases positivas y yi = -1 para clases negativas. En este contexto, el hiperplano óptimo debe ser una relación capaz de predecir el valor de yi para una xi desconocida. Si los datos de entrenamiento son linealmente separables, el hiperplano óptimo se define así (Scikit learn, 2007-2024):

$$(w)^T x_i + b \geq 1 \; para \; todo \; x_i \in clase \; positiva$$
$$(w)^T x_i + b \leq -1 \; para \; todo \; x_i \in clase \; negativa \qquad (7)$$

La regla de decisión se expresa de la siguiente manera, donde x expresa la variable de entrada, w es un vector que denota los pesos, y b es una constante simple.

$$f(x) = signo[w.x + b] \qquad (8)$$

La optimización se puede abordar mediante la introducción de un Lagrangiano de la siguiente manera:

$$L(w, b, \propto) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \propto_i \left((X_i.w) + b\right) - 1 \qquad \propto = (\propto_i, \ldots., \propto_N) \qquad (9)$$

Los multiplicadores de Lagrange, representados por &i ≥ 0, constituyen el vector α, mientras que N indica el número general de muestras en el grupo de entrenamiento (Scikit learn, 2007-2024):

$$\sum_{i=1}^{N} \propto_i Y_i = 0$$
$$w = \sum_{i=1}^{N} \propto_i Y_i X_i = 0 \qquad (10)$$

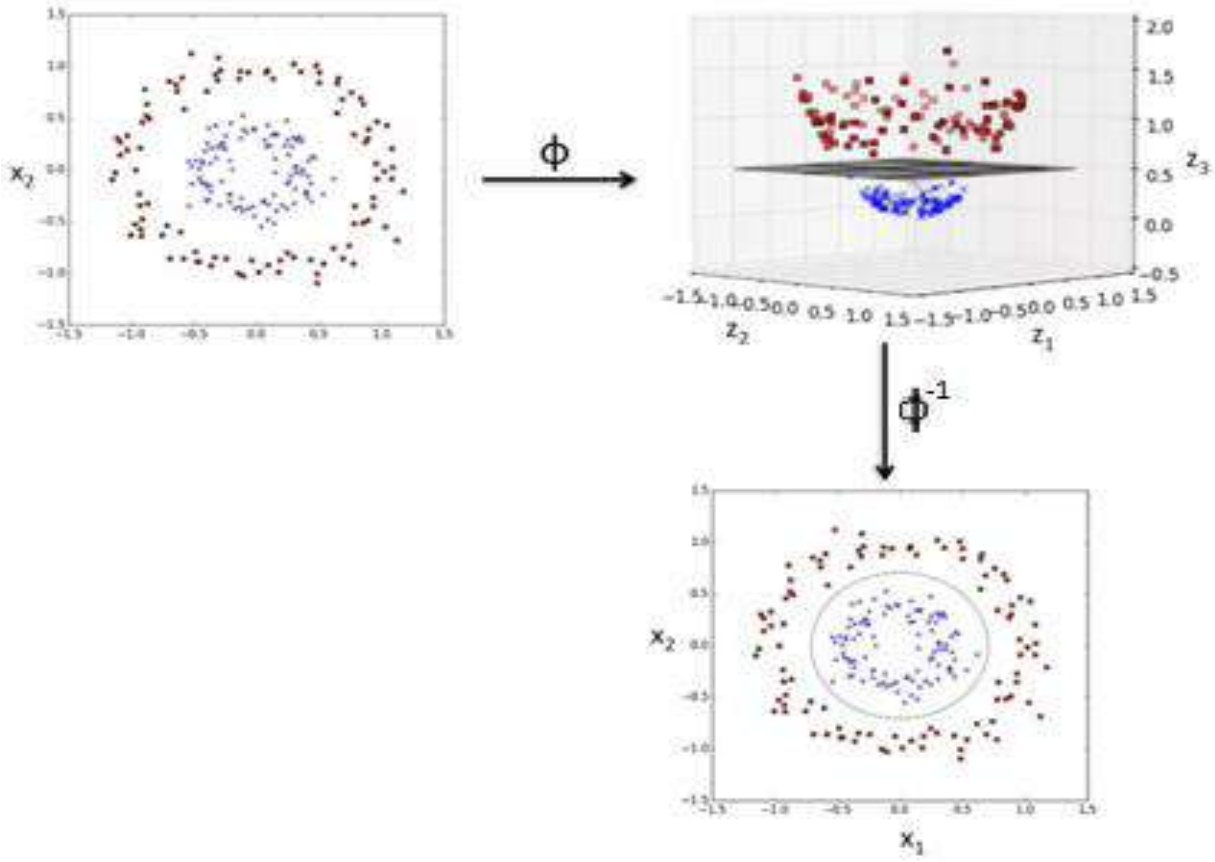
$$f(x) = signo[\sum_{i=1}^{N} \propto_i Y_i (X_i.X_j) + b] \qquad (11)$$

Donde X_i son los puntos de entrenamiento, X_j es el nuevo punto que se desea clasificar.

Es crucial comprender la definición de Kernel en relación con el mdoelo de SVM. Imaginemos que los datos están organizados como se representan en la primera gráfica de la figura 23. En esta disposición, las clases están claramente separadas, lo que debería facilitar la clasificación. Sin embargo, encontrar el hiperplano óptimo resulta imposible porque no se puede trazar una línea recta que divida los datos en dos partes distintas. Para tales casos, se emplea "truco del kernel". Esta técnica modifica el espacio en el que se encuentran los puntos, permitiendo la introducción de una nueva dimensión que facilita la división de los tipos mediante un hiperplano. En la segunda imagen de la figura 23 se muestra este hiperplano en el plano transformado, mientras que la tercera imagen ilustra cómo se proyecta el hiperplano en el plano bidimensional (Scikit learn, 2007-2024).

Figura 23

Proceso kernel

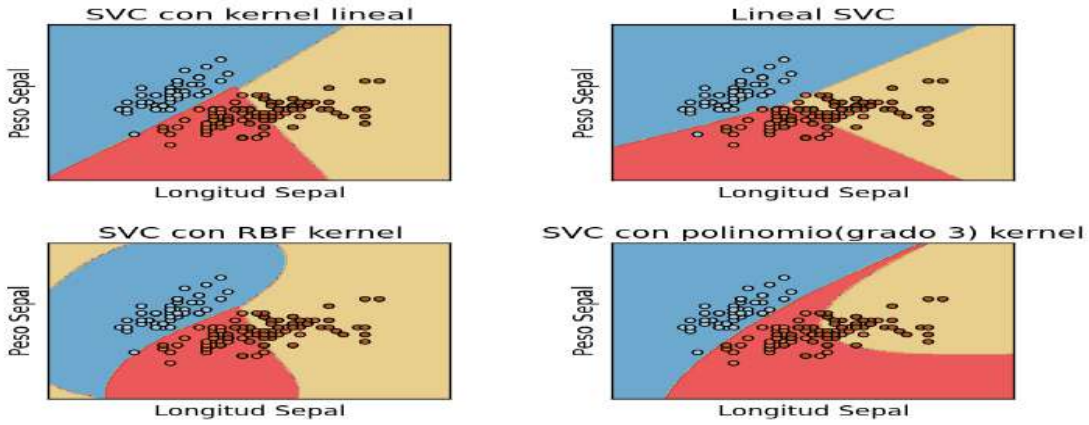


Nota: *Representación de la transformación del espacio donde se encuentran los puntos de entrenamiento mediante kernel, fuente* https://keepcoding.io/blog/importancia-de-los-kernels-para-los-svm/.

Los diversos tipos de kernel comúnmente empleados se pueden visualizar en la figura 24, donde se presentan los kernels de base radial (RBF), lineal y polinómico.

Figura 24

Tipos de kernels de SVM

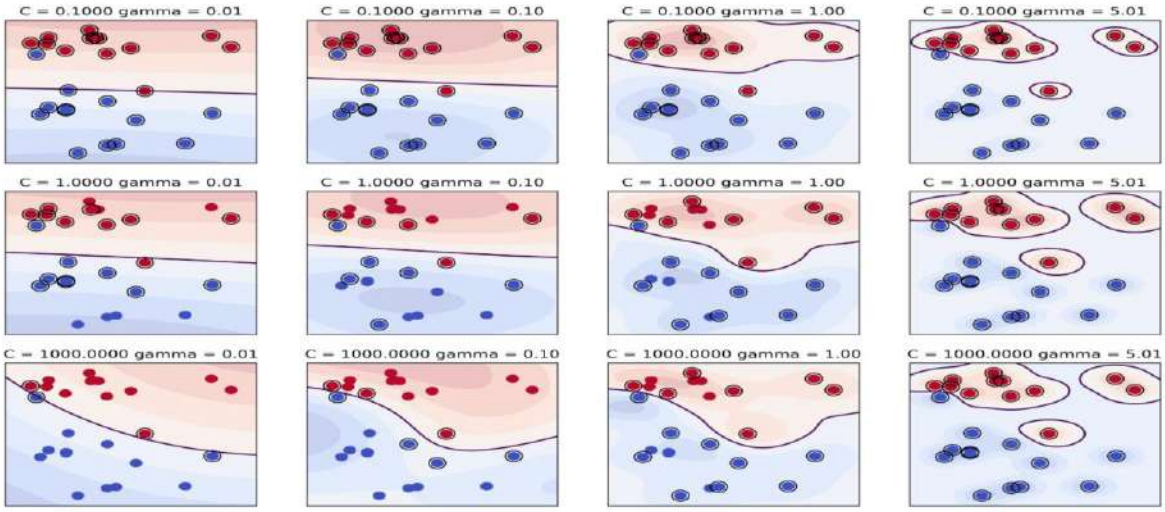


Nota*: Representación de la aplicación de distintos kernels en una data de entrenamiento compuesta por 3 clases, fuente librería Scikit-learn*

Los hiperparámetros asociados al algoritmo SVM están directamente relacionados con el kernel utilizado. Por consiguiente, los dos hiperparámetros que requieren ajuste son gamma y C. El parámetro gamma determina la extensión de la importancia de cada muestra dentro del conjunto de entrenamiento; valores pequeños de gamma sugieren una influencia más extensa, mientras que valores grandes indican una influencia más concentrada. Por otro lado, el parámetro C controla la cantidad de clasificaciones erróneas permitidas por el algoritmo al calcular el hiperplano que separa las clases. En general, se recomienda que el valor de C sea moderado, ya que valores bajos pueden favorecer con una mayor generalización de las clases, aunque a costa de una menor cantidad de clasificaciones erróneas. En la figura 25 se aprecia que, a pesar de que las técnicas para valores altos de C aparentemente producen una división más precisa, el modelo ideal podría ser aquel que admita algunas clasificaciones incorrectas

(como se muestra en el caso de C = 1 y gamma = 0.1), dado que estas clasificaciones incorrectas podrían corresponder a valores atípicos que no expresan fielmente sus clases pertinente y, por ende, no deberían sugerir significativamente en la determinación del hiperplano (Scikit learn, 2007-2024).

Figura 25
Hiperparámetros de SVM



***Nota**: Representación de una separación de dos clases bajo distintos valores de C y gamma, fuente https://amueller.github.io/aml/02-supervised-learning/07-support-vector-machines.html.*

Elegir adecuadamente los valores de los hiperparámetros; C y puede influir de manera considerable en el desempeño del modelo, por lo que es crucial seleccionar los valores más apropiados.

2.11.3 Random Forest

Esta técnica se basa en el uso de árboles de decisión, combinando múltiples árboles independientes y realizando un promedio de sus resultados. El proceso comienza tomando una muestra inicial que se introduce en un árbol de decisión, donde se somete a pruebas binarias en cada nodo, conocidas como Splits. El objetivo es encontrar una solución en la última etapa del proceso. Esta metodología es especialmente útil para resolver problemas complejos, ya que se separa en pequeños problemas más sencillos (Maisueche Cuadrado, 2019).

Su ecuación representativa es:

$$\theta_k = argmax_{\theta j \epsilon \tau j} I_j \quad (12)$$

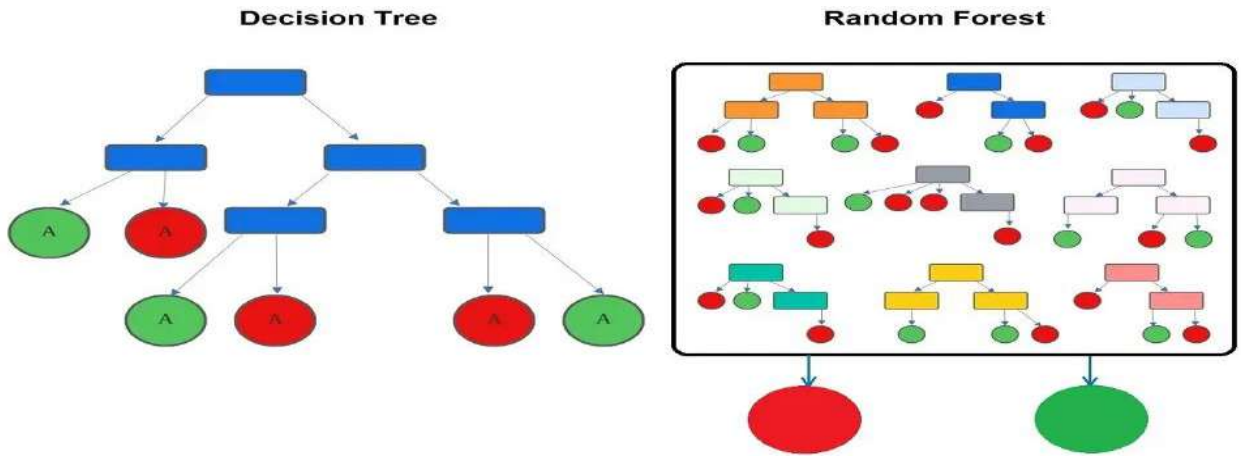
La relación de ganancia está definida con la ecuación.

$$I_j = H(j) - \sum_{i \epsilon 1,2} \frac{|S_j^i|}{|S_j|} H(S_j^i) \quad (13)$$

Considerando el grupo de muestras presentadas al nodo que se va a dividir como S y s' figurando dos subconjuntos que surgen de la escisión, esta relación tiene la capacidad de evaluar la entropía del grupo, La naturaleza de esta medición puede variar.

Hay dos enfoques para evaluar la importancia de las variables en la muestra, que son la Disminución de la Presión Media y Media de Gini. De acuerdo con varios autores, el clasificador RF es especialmente eficaz para predecir datos no estructurados, como los obtenidos a través de redes de sensores (Contreras Alvarez, 2020).

Figura 26
Arboles aleatorios



***Nota**: Comparación entre árboles aleatorios y árbol de decisión. fuente:https://towardsdatascience.com/ decision-tree-and-random-forest-from-scratch-4c12b351fe5e*

2.11.4 Gradient Boosting

La técnica forma parte de los métodos de boosting, un enfoque en el que varios modelos débiles (usualmente árboles de decisión de baja profundidad) se integran secuencialmente para crear un modelo fuerte. El concepto sobre GB, es que cada nuevo modelo busca mitigar los errores realizados por los

modelos anteriores, ajustando las predicciones aplicando un gradiente descendente sobre una función de pérdida (Scikit learn, 2007-2024).

La finalidad del modelo es minimizar una función de pérdida $L\big(y, F(x)\big)$ donde y es la variable objetivo y $F(x)$ representa la predicción del modelo. La idea principal es construir un modelo aditivo compuesto por varios modelos base ${h_m}^{(x)}$ de la siguiente forma (Scikit learn, 2007-2024):

$$F(x) = \sum_{m=1}^{M} \propto_m h_m(x) \qquad (14)$$

Donde:

- F(x) es la predicción total del modelo.
- $h_m(x)$ representa cada modelo débil (por lo general, árboles de decisión).
- $\propto_m$ es el peso asociado al modelo m.
- M es el número total de modelos.

Cada nuevo modelo base se entrena para minimizar los residuos (errores de predicción) de la etapa anterior. El modelo aplica el gradiente descendente para optimizar la función de pérdida, calculando los gradientes de los errores con respecto a las predicciones actuales (Gandhi, 2018).

El principio de la optimización que guía Gradient Boosting es el ajuste de los errores de predicción a través de una técnica de gradiente descendente sobre la función de pérdida. Esto implica que cada modelo posterior se ajusta para reducir los errores que dejó el modelo anterior (Scikit learn, 2007-2024).

- En cada iteración m, se calcula el residuo o el gradiente r_i de la función de pérdida:

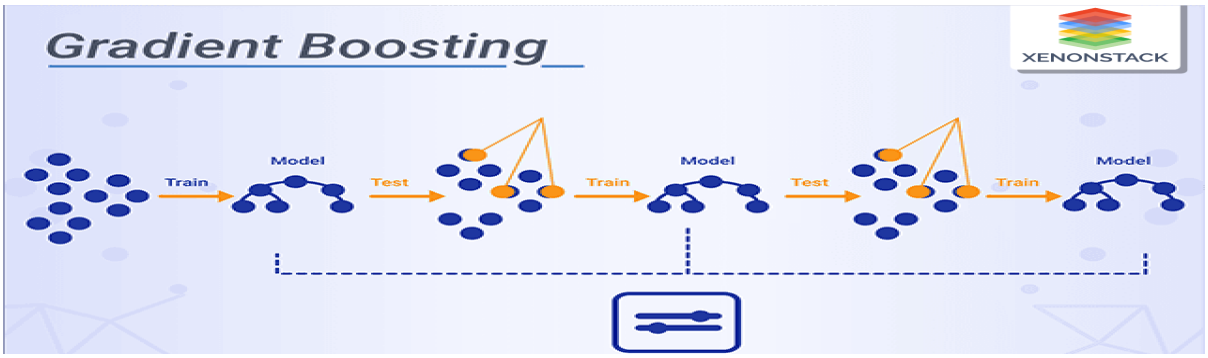
$${r_i}^{(m)} = -[\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}] \qquad (15)$$

- Luego, se entrena un nuevo árbol de decisión sobre los residuos ${r_i}^{(m)}$ para aprender los errores.
- El modelo final se actualiza en cada iteración sumando el modelo actual al anterior:

$$F_m(x) = F_{m-1}(x) + \propto_m h_m(x) \qquad (16)$$

Donde $\propto_m$ es una tasa de aprendizaje que maneja cuánto contribuye cada modelo al final.

Figura 27
Gradiente de boosting



***Nota**: Comparación del aumento de gradiente con otros modelos. fuente:* https://www.xenonstack.Com/glossary/gradient-boosting

2.11.5 Decision Tree

Sirve para problemas de clasificación y de regresión. Su esquema es similar a un diagrama de flujo, donde los nodos internos expresan atributos del set de datos, las ramas expresan los posibles valores de esas características, y los nodos hoja contienen las decisiones finales o predicciones. La idea central es segmentar el espacio de las características en subconjuntos más pequeños hasta que uno sea lo suficientemente homogéneo para tomar una decisión (Maisueche Cuadrado, 2019). Y un árbol de decisión consta de (Scikit learn, 2007-2024):

- Nodo raíz**:** El primer nodo que representa la característica más importante para dividir los datos.
- Nodos internos**:** Divisiones sucesivas de las características que conducen a la siguiente rama.
- Nodos hoja**:** El punto final de una rama, que simboliza una predicción específica o categoría.

En la construcción del árbol, se seleccionan las cualidades que mejor dividen los datos en cada paso. Las medidas más comunes para seleccionar la característica óptima son:

Gini Impurity (Impureza de Gini)**:**

Esta medida evalúa la pureza de un nodo. El valor de Gini se calcula como:

$$Gini(D) = 1 - \sum_{i=1}^{n} {p_i}^2 \qquad (17)$$

Donde pi es la proporción de elementos de la clase i en el conjunto D. Un Gini cercano a 0 señala que los elementos en el nodo pertenecen mayormente a una única clase.

Entropía (para algoritmos basados en ID3):

La entropía calcula el grado de desorden o impureza en los datos:

$$Entropía(D) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (18)$$

Un nivel de entropía más bajo indica que los datos están más ordenados.

Ganancia de Información (Information Gain):

La ganancia de información evalúa cuánto se reduce la impureza al segmentar los datos en función de un atributo particular.:

$$Ganancia(D, A) = Entropía(D) - \sum_{v \in A} \frac{|D_v|}{|D|} . Entropía(D_v) \quad (19)$$

Donde A es una característica y D_v es el subconjunto de datos donde la característica A toma el valor v.

Principio de Evaluación y Funcionamiento

El funcionamiento se fundamenta en la elección de la mejor división en cada nodo. La característica con la mayor ganancia de información (o la menor impureza de Gini) es la que se elige para dividir los datos. Este proceso continúa de manera recursiva hasta que se alcanzan ciertas condiciones de parada, como (Román, 2019):

Todos los datos en un nodo pertenecen a la misma clase.

Se alcanza una profundidad máxima predeterminada.

No hay suficientes datos para continuar dividiendo.

Después de construir el árbol, las predicciones se realizan siguiendo las ramas del árbol desde la raíz hasta los nodos hoja, comparando los valores de las características en cada nodo.

2.11.6 Naive Bayes

Basado en la aplicación del Teorema de Bayes asumiendo una independencia significativa entre las variables. Es especialmente popular para trabajos de clasificación. La palabra "naive" (ingenuo) hace referencia al hecho de que el modelo asume que todas las características son independientes entre sí, lo cual casi imposible, sin embargo, en muchos casos esta suposición facilita los cálculos sin afectar de manera notable el desempeño del modelo (Management Solutions, 2018).

Utiliza el Teorema de Bayes para hallar la probabilidad condicional de que una muestra x pertenezca a una clase específica C_k :

$$P(C_k \mid X) = \frac{P(x \mid C_k).P(C_k)}{P(x)} \quad (20)$$

$P(C_k \mid X)$ es la probabilidad posterior de que X pertenezca a la clase C_k .

$P(x \mid C_k)$ es la probabilidad de observar los datos X si pertenecen a la clase C_k (probabilidad condicional).

$P(C_k)$ es la probabilidad a priori de la clase C_k .

$P(x)$ es la probabilidad de los datos observados x.

El modelo Naive Bayes asume que todas las características X= (x1, x2,..., xn) son independientes entre sí, lo que simplifica la ecuación:

$$P(x \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k) \quad (21)$$

Por lo tanto, la probabilidad posterior se convierte en:

$$P(C_k \mid X) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \quad (22)$$

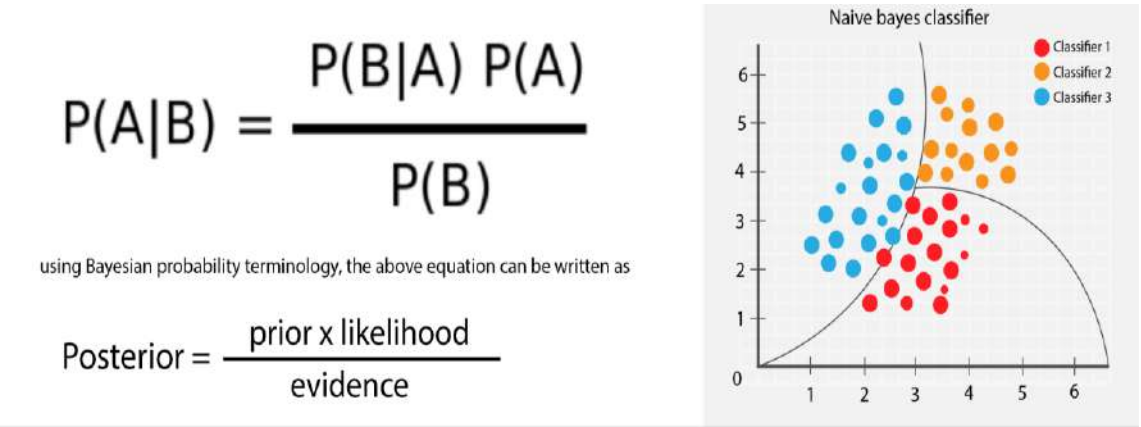
En la práctica, se omite P(X) porque es una constante para todas las clases y no afecta la clasificación final.

Específicamente, La probabilidad condicional para una característica xi dado que pertenece a la clase Ck , de Gaussian Naive Bayes, se calcula como:

$$P(\mathrm{xi} \mid \mathrm{Ck}) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(\mathrm{xi} - \mu\mathrm{k})^2}{2\sigma_k^2}\right) \quad (23)$$

Donde $\mu\mathrm{k}$ es la media y σ_k^2 es la varianza de la característica en la clase Ck .

Figura 28
Naive bayes



***Nota**: Relación entre un clasificador bayesiano y un clasificador probabilístico. fuente:* https://thatware.co/naive-bayes/

2.12 Métricas de evaluación

La fase final consiste en una evaluación exhaustiva, que puede realizarse mediante la determinación de la precisión, la exactitud o mediante el uso de la validación cruzada. Que señala si el clasificador cumple con los estándares deseados. Si no es satisfactorio, se pueden aplicar diferentes ajustes, como recopilar más datos, implementar técnicas adicionales para limpiar los valores, probar nuevas características o la optimización de los parámetros de los algoritmos de entrenamiento. Estos ajustes se deben realizar de manera iterativa hasta lograr el modelo más eficiente. La figura 29 ofrece una visión general del proceso completo para diseñar un clasificador supervisado de fallas (Román, V., 2019).

Matriz de Confusión

Es una de las técnicas más fundamentales para analizar la eficacia y exactitud del modelo en proceso de entrenamiento. Es particularmente útil en situaciones de clasificación que involucran múltiples clases o categorías. Por ejemplo, en una situación donde se desea detectar si un sistema presenta eventos de fallas o no, se asignan etiquetas a la variable objetivo: "1" para señalar un evento con fallas y "0" para señalar un evento de funcionamiento normal (Guamán Buestán, 2019).

Figura 29
Matriz de confusión

***Nota**: Representación de valores de falsos positivos y falsos negativos del modelo aprendizaje. Fuente, https://aprendeia.com/evaluandoelerrorenlosmodelosdeclasificacionmachinelearning/#google_vignette*

Verdaderos Positivos (VP): Indican los casos correctos en los que el modelo identifica un evento como "positivo" y este realmente lo es.

Verdaderos Negativos (VN): Corresponden a las clasificaciones correctas en las que el modelo identifica un evento como "negativo" y esta evaluación es acertada.

Falsos Negativos (FN): Representan errores donde el modelo clasifica un evento como "negativo", aunque en realidad sea positivo.

Falsos Positivos (FP): Son errores en los que el modelo identifica un evento como "positivo", cuando en realidad este es negativo.

Estos parámetros constituyen lo que se denomina matriz de confusión. Basados en esta, es posible hallar diversas métricas para evaluar la efectividad del modelo, tales como:

$$
\begin{aligned}
Accuracy(\text{Exactitud}) &= \frac{VP + VN}{VP + VN + FP + FN} \\
Precision(\text{Precisión}) &= \frac{VP}{VP + FP} \\
Sensitivity(\text{Sensibilidad}) &= \frac{VP}{VP + FN} \\
Specificity(\text{Especificidad}) &= \frac{VN}{VN + FP}
\end{aligned}
\qquad (24)
$$

La Curva ROC (Receiver Operating Characteristic) es una expresión visual que permite analizar la eficacia de un modelo clasificador al mostrar la relación entre la Tasa de Verdaderos Positivos (sensibilidad) y la Tasa de Falsos Positivos. Al modificar el umbral de decisión del modelo, la curva visualiza cómo cambia su rendimiento para diferenciar entre clases positivas y negativas (Guamán Buestán, 2019). El Área Bajo la Curva ROC (AUC) determina la habilidad del modelo para diferenciar entre las clases. correctamente entre clases; su valor oscila entre 0 y 1, donde un AUC cercano a 1 indica un excelente desempeño del modelo, mientras que un AUC de 0.5 sugiere una capacidad discriminativa similar a una elección al azar (De los Rios Tomala, 2019).

Validación Cruzada de K-iteraciones

Una técnica alternativa de evaluación, a diferencia de la técnica mencionada anteriormente, esta metodología no divide una sección arbitraria de los datos para las pruebas. En cambio, separa los datos en K divisiones distintas de entrenamiento/evaluación, donde 1/K de los valores se utiliza en la evaluación y el resto en el entrenamiento. Esta práctica se itera K veces (K-iteraciones), garantizando que en cada iteración se utilice una sección distinta de los valores para la evaluación. Posteriormente, se calculan las métricas de rendimiento, como la exactitud (Accuracy), en cada iteración. Al final, se promedian estos resultados para conseguir una estimación global de la eficacia del modelo (Guamán Buestán, 2019).

2.13 Sistema Embebido

En el campo de los sistemas embebidos, resulta crucial la habilidad para interactuar con señales físicas como presión, temperatura, corriente, entre otras, a fin de garantizar un funcionamiento óptimo y efectivo. Estos sistemas son concebidos para operar en tiempo real y satisfacer requerimientos específicos, lo que demanda la capacidad de procesar datos provenientes de sensores físicos de forma ágil y precisa. Incluir un procesador que pueda interactuar directamente con estas señales físicas en la misma placa base del sistema embebido simplifica la captación, preprocesamiento y análisis de los datos en tiempo real, lo que a su vez optimiza la suficiencia de respuesta y eficacia del sistema en su conjunto. Además, esta integración reduce la necesidad de componentes externos, lo que disminuye la complejidad y los costos, resultando en sistemas más compactos, eficientes y rentables (ESP32, 2024).

En la actualidad, los sistemas basados en inteligencia artificial se han vuelto fundamentales para procedimientos FDD en sistemas como aires acondicionados. Estas soluciones proporcionan un equilibrio entre exactitud y eficiencia, lo que las hace fundamentales para asegurar el rendimiento óptimo de los equipos. Además, con el avance tecnológico, se están desarrollando algoritmos cada vez más sofisticados que pueden ejecutarse en tiempo real y en plataformas integradas, Esto facilita la identificación de fallas de manera rápida y precisa en los sistemas de aire acondicionado (Arduino.cc, 2024).

Dentro del marco de este proyecto de investigación, se seleccionaron dos sistemas embebidos, como una unidad de procesamiento y otra de adquisición, que satisfacen los criterios de costo computacional establecidos. Según antecedentes que usan estos sistemas de adquisición y la disponibilidad en el mercado (Raspberry y Arduino), como las mejores opciones.

2.14 Herramientas Computacionales

2.14.1 Python

Es un entorno polifacético, interpretado y de alto nivel que experimento un crecimiento notable en su adopción últimamente. Proporciona una variedad de funcionalidades y características que lo hacen idóneo para diversas aplicaciones, incluidas las relacionadas con sistemas embebidos (Python, 2024).

Una ventaja destacada de Python es su portabilidad y compatibilidad con múltiples plataformas y SO, como Linux, Windows y Mac OS X. Esto garantiza que los sistemas embebidos desarrollados con Python puedan ejecutarse en una variedad de dispositivos y entornos, lo que aumenta su accesibilidad y versatilidad. Además, Python cuenta con una biblioteca estándar extensa y una gran variedad de paquetes y frameworks disponibles que facilitan la evolución del software para sistemas embebidos. Desde aplicaciones científicas hasta comunicación en red, Python ofrece herramientas y recursos que simplifican el proceso de desarrollo y aceleran la implementación de funcionalidades clave en los sistemas embebidos (Python, 2024).

2.14.2 Scikit-Learn:

Libreria de código abierto aplicado al aprendizaje automático en Python, utilizada para llevar a cabo diversas tareas vinculadas con data mining y machine learning. Proporciona una gran variedad de funciones para aprendizaje supervisado y no supervisado, junto con módulos para el preprocesamiento de datos, la evaluación de modelos y la selección de características. Se destaca por su facilidad de uso, su documentación exhaustiva y su eficacia en la aplicación de modelos de machine learning. Scikit-learn encuentra aplicaciones en múltiples áreas, como la clasificación de textos, el reconocimiento de imágenes, la detección de anomalías, la regresión y la agrupación. Además, se integra fácilmente con otras frameworks, librerias populares de Python, como Pandas y NumPy, lo que simplifica su uso en flujos de trabajo existentes. Su comunidad activa de desarrolladores y su amplia base de usuarios la transforman en una vía preferida para los que pretenden implementar soluciones de aprendizaje automático en Python (Scikit learn, 2007-2024).

2.14.3 Mysql:

Es un sistema de gestión de bases de datos relacional de fuente abierta que tiene una alta demanda, tanto en entornos web como corporativos. Destacado por su fiabilidad, rapidez y facilidad de uso, ha sido ampliamente adoptada por desarrolladores y organizaciones. Sus características abarcan desde la compatibilidad con diversas plataformas hasta el soporte para múltiples tipos de datos, pasando por transacciones ACID, replicación y escalabilidad. Además, ofrece herramientas para la gestión, copia de seguridad y optimización de recursos de los conjuntos de datos. La integración fluida de MySQL con lenguajes como Python, PHP y Java lo hace accesible en diversos entornos. Asimismo, Dispone de una comunidad proactiva que ofrece muchos recursos, incluyendo información extensa y foros de usuarios.

2.15 Variables

2.15.1 Variables Independientes

Señales de la condición de estado:

- Temperatura (T°) de contacto del armazón del compresor (T).
- Corriente (I) de trabajo del compresor (A).
- Voltaje (V) del compresor (V).
- Presión (P) en la línea de baja presión (Pl).
- Presión (P) en la línea de alta presión (Ph).
- Presión (P) de succión del compresor (Ps).
- Presión (P) de descarga del compresor (Pd).
- Potencia de consumo (W) del compresor (W).
- Temperatura (T°) en la salida del condensador (Tc).
- Temperatura (T°) en la salida del evaporador (Te).

2.15.2 Variable Dependiente

Detección las fallas.

Capítulo III

Metodología

La metodología es un pilar fundamental en cualquier trabajo de investigación, ya que establece el conjunto de procedimientos y técnicas que guiarán el desarrollo del estudio. Es crucial para estructurar y se verifica la aplicación de modelos de aprendizaje automático y el empleo de tecnologías para la recopilación y el procesamiento de datos. Este capítulo detalla el tipo y nivel de investigación, el diseño experimental empleado, y las técnicas utilizadas para garantizar que los resultados conseguidos sean confiables y aplicables, subrayando la importancia de cada decisión metodológica en el éxito del proyecto.

3.1 Tipo de Investigación

Esta línea de estudio tiene propósito cuantitativo debido a que el sistema toma en cuenta la adquisición y análisis de las señales de presión, temperatura, corriente y voltaje del sistema de AAP, se genera una base de datos con situaciones específicas de fallas y se realizan pruebas a escala piloto, mediante modelos de aprendizaje automático (IA) se busca detectar la presencia de posibles fallas en el sistema de AAP y técnicas estadísticas para la validación y evaluación del rendimiento del sistema.

3.2 Nivel de Investigación

Este proyecto se identifica como un estudio explicativo, ya que no solo busca medir las variables (presión, temperatura, corriente y voltaje) y especificar sus características, sino que también tiene como objetivo evaluar las causas subyacentes de los eventos de falla más comunes evaluados. A través del análisis de los patrones obtenidos en los datos, durante los ciclos de refrigeración, se busca obtener un modelo con el que se puede detectar los eventos de fallas.

3.3 Diseño de Investigación

En el estudio se manipula el valor de las variables presión, temperatura, corriente y voltaje (causas), con el propósito de examinar las repercusiones de dichas modificaciones y observar cómo

afectan a otra variable (variable dependiente). Estas pruebas en condiciones controladas a escala piloto refuerzan que es un estudio experimental, para probar sobre la capacidad del sistema de IA de detectar fallas y por lo previsto es un estudio de clase cuasiexperimental, porque se manipulan las condiciones del sistema para generar fallas, pero no se puede controlar todas las variables del entorno.

3.4 Alcance de Investigación

- En este estudio sobre el aire acondicionado (cooling unit), se evaluó con enfoque a nivel sistema y se diseñó un sistema que adquiera las señales de presión, temperatura, voltaje y corriente.
- En este estudio se usó alcances de la industria 4.0 (IA, ML, Big Data) y modelos de aprendizaje del tipo supervisado (por ser ampliamente usados para la detección de fallas), donde se empleó un subconjunto de datos training (generada), para instruir a los modelos. También se evaluaron los índices de la exactitud (accuracy).
- En este trabajo se detecta si un evento tiene un funcionamiento con falla o esta normal, basado en el nivel de las señales de estado P, T°, V, I. (si se sitúan o no fuera del rango de tolerancia para un patrón de funcionamiento normal del sistema), mediante los modelos entrenados, con una generación controlada de fallas (finalmente variando el valor de la presión, la temperatura, el voltaje y la corriente).
- El sistema de detección cuenta con una unidad de procesamiento con capacidad suficiente para obtener una base de datos y procesar las señales para clasificarlas.

3.5 Limitaciones de Investigación

- El estudio se realiza sobre el sistema de aire acondicionado de precisión (AAP), de la marca Rittal, modelo SK3328.500 (enclosure, Top Therm wall-mounted), es una unidad de enfriamiento (cooling unit)
- Se evaluó mediante técnicas de aprendizaje supervisado (KNN, SVM, RF, GB, DT y NB).
- Se evaluó solo las señales de P, T°, I, V durante el periodo necesario para obtener el conjunto data set para el entrenamiento, la validación y prueba.

3.6 Unidad de Análisis

Son los datos sensoriales recolectados del sistema de aire acondicionado de precisión (AAP). Estos datos incluyen lecturas de corriente, voltaje, temperatura y presión, los cuales son procesados por los modelos de aprendizaje automático para definir si el sistema opera en condiciones normales o presenta fallas.

3.7 Técnicas e Instrumentos de Recolección de Datos

La investigación fue de carácter experimental y las técnicas incluyen la monitorización continua de las variables críticas del sistema de aire acondicionado mediante sensores seleccionados como ACS712 para corriente, ZMPT/101 para voltaje, MLX90614 para temperatura del compresor, SPKT0043P para presión y DS18B20 para la temperatura. Los datos se recopilan utilizando una Raspberry Pi y un Arduino Nano, que están conectados por USB y utilizan protocolos de comunicación I2C y señales analógicas.

3.8 Viabilidad y Factibilidad

Este estudio resultó viable y factible por diferentes factores como; la disponibilidad del equipo de aire acondicionado de precisión (AAP), la disponibilidad de los componentes tecnológicos necesarios, como sensores y plataformas de procesamiento (Raspberry Pi y Arduino Nano) y la posibilidad de generar un repositorio de datos para fallas específicas (UC la subcarga de refrigerante, OC sobrecarga de refrigerante, RL restricción de la línea de líquido, CA reducción de flujo de aire del condensador y EA reducción de flujo de aire del evaporador). Además, la implementación de los modelos de inteligencia artificial mediante Python y la biblioteca Scikit-learn es accesible y respaldada por una amplia documentación, lo que facilita la integración y optimización del sistema propuesto.

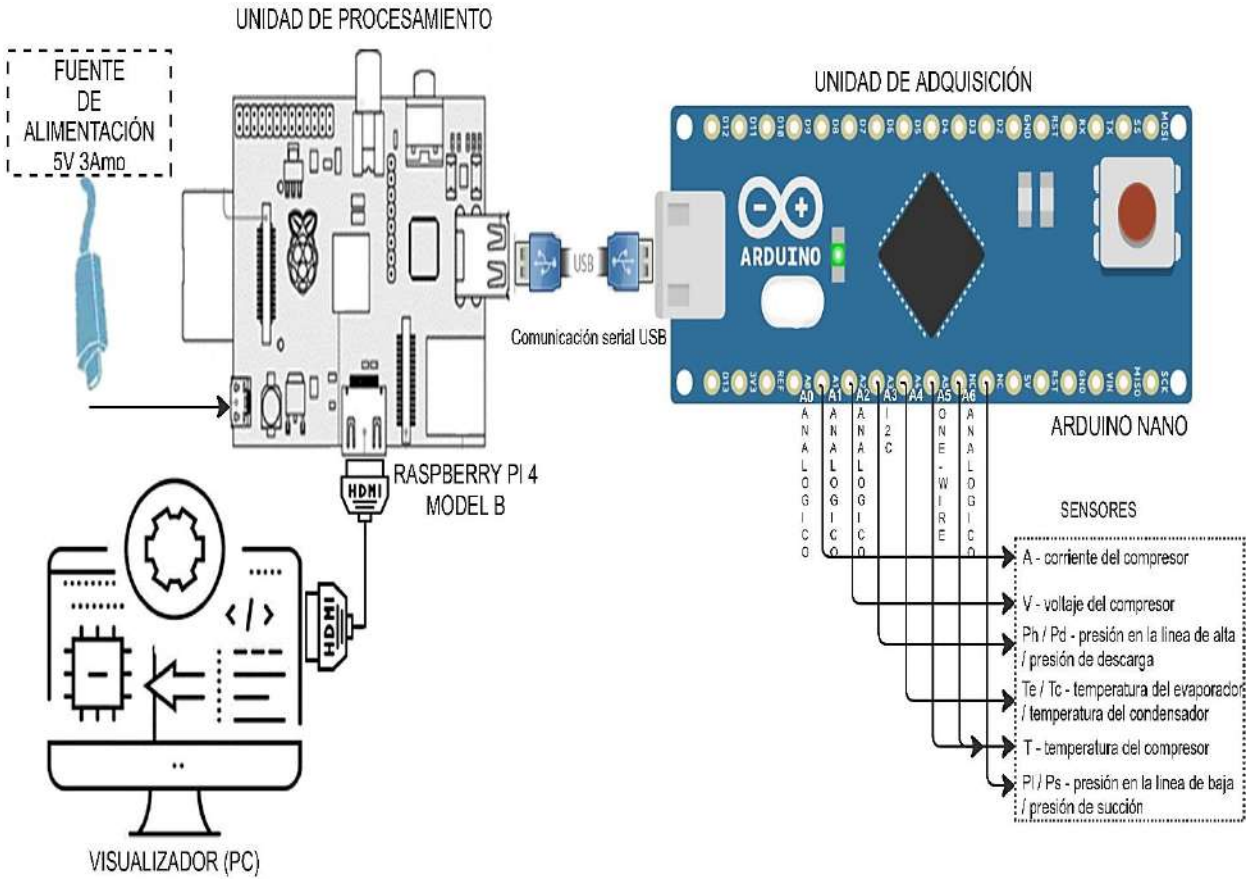
Capítulo IV

Diseño del Sistema de Diagnóstico y Detección de Fallas

Es la etapa inicial del desarrollo de una solución tecnológica automatizada para la detección de fallas en un sistema de aire acondicionado de precisión. Que cuenta principalmente con dos subsistemas y cada uno de estos tienen módulos. Que a su vez involucra la selección en la parte instrumental (de sensores), en la parte de detección (se evalúa algoritmos, microprocesadores, microcontroladores y bases de datos), incluyendo la definición de las características de detección, pruebas y resultados.

Figura 30

Diagrama general del sistema de Diagnóstico y detección de fallas



***Nota**: Representación esquemática de un sistema de diagnóstico y detección de fallas, adaptado de la fuente: https://www.dreamstime.com/ stock-illustration-arduino-electronic-elements-components-prototype-applications-image69038231*

4.1 Consideraciones Previas del Diseño

En esta sección, se establece los requerimientos, para detectar y diagnosticar eventos de fallas del aire acondicionado de precisión (AAP), mediante un subsistema adquisición de señales de manera automatizada.

4.1.1 Requerimientos Generales

Rapidez de detección: Es importante que el sistema tenga la capacidad de detectar las fallas en tiempo real. El análisis de la base de datos generada reveló que, en promedio, cada 8 a 10 min se capturan entre 300 y 600 muestras. Esta rapidez de muestreo es indispensable, ejemplo al iniciar el compresor se registran corrientes de 24 A que, si el intervalo de muestreo fuera mayor, podrían no ser detectados.

Precisión de detección: El sistema debe garantizar una exactitud notable en el reconocimiento de fallas, minimizando la incidencia de resultados falsos tanto positivos como negativos.

Robustez: El sistema debe ser resiliente ante posibles variaciones en las condiciones del entorno, aun cuando su aplicación se dé en ambientes controlados como los centros de datos.

Escalabilidad y replicabilidad: Es esencial que el sistema sea escalable, permitiendo la adición de actualización para integrar nuevas soluciones y funcionalidades con el tiempo. Asimismo, debe ser replicable en una variedad de sistemas HVAC para garantizar su utilidad generalizada.

4.1.2 Requerimientos de Diseño (software)

Sistema operativo: El algoritmo debe ser diseñado para ser ejecutado en un sistema operativo específico, como Linux, Microsoft o MacOS, asegurando su compatibilidad total.

Adquisición y procesamiento de datos: El algoritmo debe poder adquirir valores en tiempo real desde los sensores y procesarlos utilizando modelos de aprendizaje automático.

Algoritmos de detección y diagnóstico de fallas: Se requiere que el modelo integre algoritmos para el procedimiento FDD, los cuales analizan y procesan los datos provenientes de los sensores para determinar cuándo los niveles de las señales se encuentran fuera del rango óptimo de funcionamiento.

4.1.3 Requerimientos de Implementación (hardware)

Unidad de procesamiento: Un procesador especializado, diseñado para manejar un estudio de datos en tiempo real y que soporte la ejecución de software enfocado en detección y diagnóstico.

Unidad de adquisición: El sistema requerirá un microcontrolador que pueda adquirir las señales de los sensores de características analógicas y pueda interactuar con la unidad de procesamiento.

Sensores de instrumentación: Este trabajo tiene por objetivo analizar señales de naturaleza de presión, corriente, voltaje y temperatura del sistema. Mediante sensores comerciales e industriales.

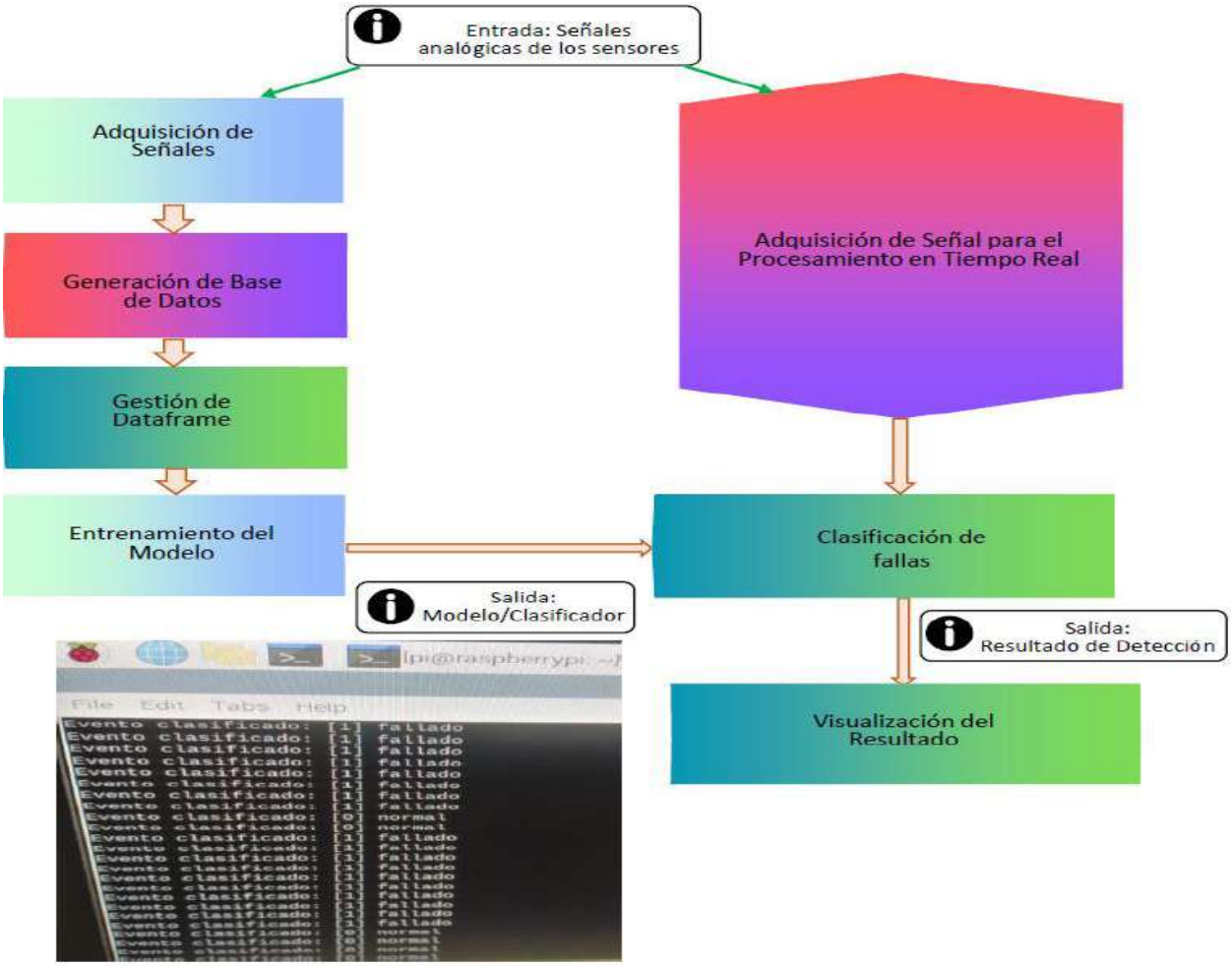
Finalmente se seleccionaron 10 características para formar el conjunto de características, esto basado en la experiencia y las recomendaciones de los antecedentes, que en gran mayoría frecuentan usar las mismas variables (Chen, y otros, 2022), pero para este estudio, se seleccionaron 1 la presión (Pl) en la línea de baja presión, 2 presión (Ph) en la línea de alta presión, 3 presión (Ps) de succión y 4 presión (Pd) de descarga, con estas 4 medidas de presión, se busca evaluar su comportamiento en las principales etapas del sistema, donde se registra valores de presión bajas y altas, dentro del mismo sistema para eventos de falla y las del compresor, 5 temperatura (T°) en la salida de evaporador, 6 temperatura (T°) en la salida de condensador (Zhu, Du, Chen, Jin, & Huang, 2019), 7 temperatura (T°) de contacto del armazón del compresor, con esta medida se busca evaluar el comportamiento frente a eventos de falla, bajo el principio de que la generación de calor es un subproducto inevitable del funcionamiento de la mayoría de los sistemas, este fenómeno puede ser generado por una variedad de factores, como condiciones operativas subóptimas, que producen un incremento excesivo de la temperatura (Ebrahimifakhar, Kabirikopaei, & Yu, 2020), 8 corriente (I) de trabajo del compresor, 9 voltaje (V) del compresor y 10 potencia (W) del compresor, con estas últimas medidas se busca evaluar la variación del consumo de potencia, en situaciones de eventos de falla del sistema, como se evalúa en otras investigaciones (Laughman , Armstrog, Leeb, & Armstrong, 2006).

4.2 Lógica del Sistema

El enfoque del sistema se basa en una metodología basada en datos, orientado para la detección y diagnóstico de fallas al tiempo. En lo siguiente, se expone una descripción general de las 7 partes del sistema propuesto.

Figura 31

Módulos del sistema de diagnóstico y detección de fallas

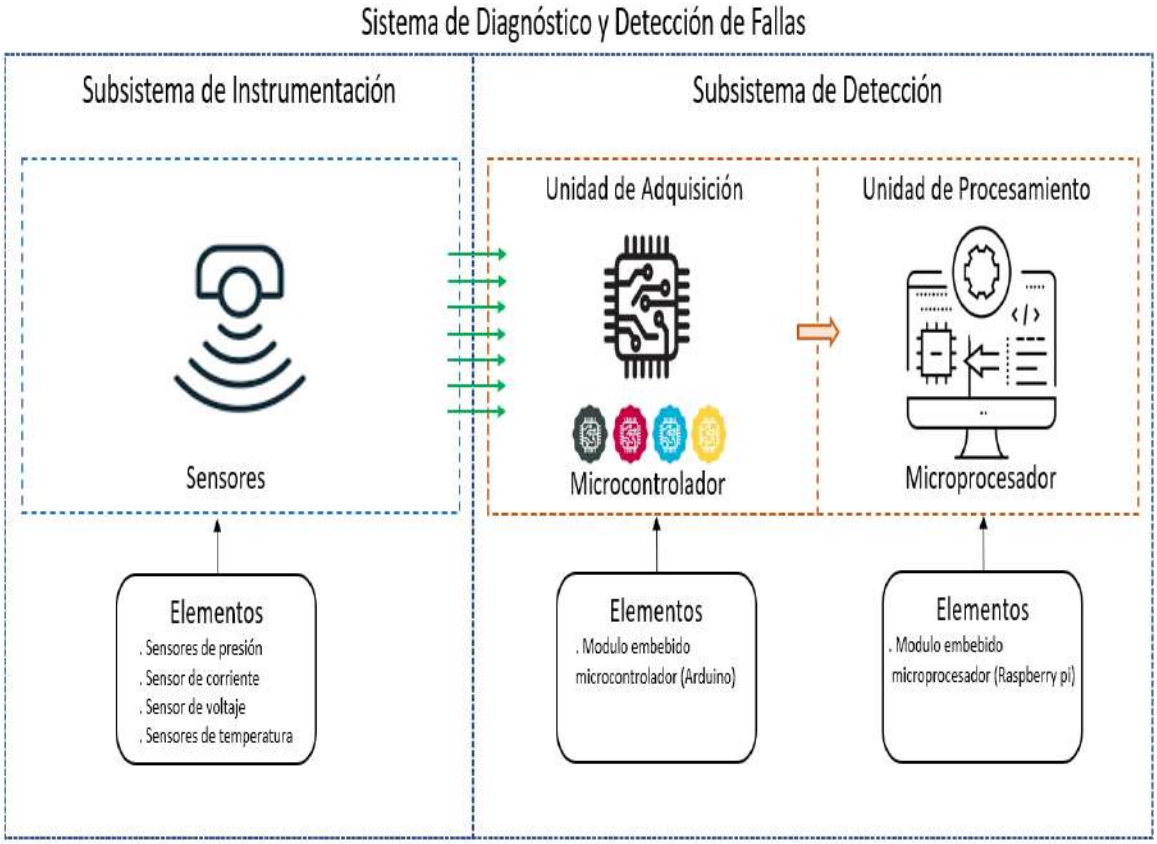


4.3 Formato de la Data de la Señal

De manera anticipada, en la etapa adquisición se estructura la forma de cómo se almacenan los datos, ya que prevé usar una opción de set de datos estructurado SQL, para que se pueda manipular de manera sencilla en diferentes formatos como json, csv y dataframe de pandas. Además, contiene la data de las señales de tipo float en cada columna configurada.

4.4 Sistemas de Diagnóstico y Detección de Fallas

Figura 32
Subsistemas del sistema de diagnóstico y detección de fallas



***Nota**: Adaptado de fuente: https://www.dreamstime.com/illustration /microcontrollerprograming.html*

Este sistema se puede dividir en dos subsistemas principales, que evalúan la etapa relacionada al elemento hardware y software, respectivamente.

4.4.1 Subsistemas de Instrumentación

Este subsistema es el responsable de evaluar todos los componentes instrumentales; características, naturaleza, principios de funcionamiento, requerimientos, entre otros aspectos, de cada sensor que se contempla usar para este estudio y los protocolos de comunicación requeridos para cada caso específico. También abarca la etapa de acondicionamiento de las señales, calibración de sensores, para buscar la confiabilidad y la exactitud de los datos captados.

4.4.2 Subsistemas de Detección

Este subsistema es una parte esencial del sistema, dedicada a la obtención y análisis al tiempo de las señales generadas por el aire acondicionado (AAP) con el propósito de detectar y diagnosticar fallas específicas. Utiliza tecnología especializada para analizar de manera continua las señales de presión, corriente, voltaje y temperatura. Su objetivo principal radica en la clasificación de estas señales evaluadas, determinando si reflejan un funcionamiento normal o indican la presencia de fallos.

Los componentes esenciales del subsistema de detección pueden comprender lo siguiente:

Unidad de adquisición: El microcontrolador tiene una función central en este subsistema, encargándose de obtener las señales provenientes de los sensores con características analógicas y facilitar la interacción con la unidad core.

Unidad de procesamiento especializado: El core procesador juega un rol crucial en el subsistema de detección al encargarse del procesamiento y análisis de los valores en tiempo real para identificar posibles fallos en AAP.

Algoritmos de evaluación de datos: Estos algoritmos se encargan de procesar la información recabada por el sistema de detección, aplicando modelos de inteligencia artificial para detectar posibles fallas en el sistema AAP.

4.5 Selección de Elementos para el Diseño del Sistema

La selección de los componentes puede ser clasificado según su funcionalidad, dependiendo del rol que cumplan y pueden clasificarse así.

4.5.1 Selección de la Unidad de Procesamiento

Para seleccionar el microprocesador más idóneo que cumpla con el costo computacional para el sistema de este proyecto que se busca evaluar, se comparó tres alternativas: Raspberry Pi 4B 4GB, Raspberry Pi 4B-8GB y Raspberry Pi 5B-4GB, en la Tabla 2 se detallan las cualidades de cada unidad de procesamiento con el objetivo de seleccionar la más apropiada.

Tabla 2

Comparación de las unidades de procesamiento

	RASPBERRY Pi 4B *4GB*	**RASPBERRY Pi 4B-8GB**	**RASPBERRY Pi 5B-4GB**
CPU	Broadcom BCM2711 ARM Cortex-A72	Broadcom BCM2711 ARM Cortex-A72	Broadcom BCM2712 ARM Cortex-A76
CPU Frecuencia	64bit 1.5GHz Quad-core	64bit 1.5GHz Quad-core	64bit 2.4GHz Quad-core
GPU	Broadcom VideoCore VI a 500MHz	Broadcom VideoCore VI a 500MHz	Broadcom VideoCore VII a 800MHz
RAM	4GB LPDDR4	8GB LPDDR4	4GB LPDDR4X
Wifi	2.4G/5G 802.11.b/g/n/ac	2.4G/5G 802.11.b/g/n/ac	2.4G/5G 802.11.b/g/n/ac
Bluetooh	5.0, BLE	5.0, BLE	5.0, BLE
Internet	Gigabit Ethernet (RJ45)	Gigabit Ethernet (RJ45)	Gigabit Ethernet (RJ45)
POE powered	Si	Si	Si
Display	Micro HDMI*2 support 4k60	Micro HDMI*2 support 4k60	Micro HDMI*2 support 4k60
IO	40 Pin	40 Pin	40 Pin
USB	2*USB 3.0 2*USB 2.0	2*USB 3.0 2*USB 2.0	2*USB 3.0 2*USB 2.0
Precio promedio	S/. 400.00	S/. 550.00	S/. 800.00

El Raspberry Pi 4B 4GB viene equipado con un procesador Broadcom BCM2711 ARM Cortex-A72 de 64 bits, que opera a una frecuencia de 1.5GHz en Quad-core. Incorpora una GPU Broadcom VideoCore VI de 500MHz y 4GB de memoria RAM LPDDR4. Ofrece conectividad Wi-Fi dual (2.4G/5G) 802.11.b/g/n/ac, Bluetooth 5.0 y Ethernet Gigabit, garantizando una amplia variedad de opciones de conexión. Además, cuenta con puertos USB 3.0 y USB 2.0, soporte para visualización en 4k60 a través de dos puertos Micro HDMI, y es compatible con PoE. Su precio promedio es de S/. 400.00.

En contraste, el Raspberry Pi 4B-8GB comparte características similares al modelo de 4GB, pero se distingue por su memoria RAM LPDDR4 de 8GB. Esta capacidad adicional de memoria puede ser beneficiosa para ejecutar aplicaciones más exigentes, aunque su precio promedio es más elevado, llegando a los S/. 550.00 (Raspberry Foundation, 2020).

Por su parte, el Raspberry Pi 5B-4GB presenta un procesador Broadcom BCM2712 ARM Cortex-A76 de 64 bits, que funciona a una frecuencia de 2.4GHz en Quad-core, junto con una GPU Broadcom

VideoCore VII de 800MHz. Aunque ofrece un rendimiento ligeramente superior a los modelos anteriores, su precio promedio es el más alto, alcanzando los S/. 800.00 (Raspberry Foundation, 2020).

En vista de ello, para la unidad de procesamiento, tras evaluar las opciones, el microprocesador Raspberry Pi 4B 4GB es la elección más adecuada para el sistema. Ofrece un equilibrio óptimo entre rendimiento, capacidad de memoria y precio. Gracias a su potente procesador, conectividad versátil, escalabilidad flexible y capacidad para manejar aplicaciones exigentes lo hacen ideal para nuestras necesidades de este proyecto que se busca evaluar. En resumen, el Raspberry Pi 4B 4GB proporciona la mejor relación calidad-precio y rendimiento para nuestro proyecto.

4.5.2 Selección de la Unidad de Adquisición

Se comparó tres alternativas disponibles: ESP-32, Arduino Uno y Arduino Nano.

Tabla 3

Comparación de las unidades de adquisición

	ESP-32 DEVKITV1	**ARDUINO UNO**	**ARDUINO NANO**
Microcontrolador	Microprocesador Xtensa Dual Core LX6	ATMega328	ATMega328P
Memoria Flash	4MB	32KB	32KB
SRAM	520MB	2KB	2KB
EEPROM	No disponible	Un KB	Un KB
Velocidad del reloj	Hasta 240 MHz	16MHZ	16MHZ
Tensión de funcionamiento	3.3V DC	5V DC	5V DC
Tensión de entrada	3.3V DC	7V-12V DC	7V-12V DC
Pines digitales IO	36	14	14
Pines de entrada analógica	18	6	8
Resolución de entradas ADC	12 bits	10 bits	10 bits
I2C	2	1	1
UARTs	3	1	1
SPI	4	1	1
CAN	SI	NO	NO
PWM	16	6	6
WI-FI	802.11 b/g/n	NO	NO
BLUETOOTH	v4.2 BR/EDR y BLE	NO	NO
Precio promedio	S/. 45.00	S/. 60.00	S/. 30.00

El ESP-32 está equipado con un microprocesador Xtensa Dual Core LX6 de 32 bits, que presenta una memoria Flash de 4MB y una SRAM de 520MB. A diferencia de los microcontroladores Arduino, el ESP-32 no cuenta con EEPROM. Su velocidad de reloj alcanza hasta 240MHz y opera con una tensión de 3.3V DC. Ofrece una amplia variedad de pines digitales IO (36) y pines de entrada analógica (18), junto con soporte para I2C (2), UARTs (3), SPI (4) y CAN. Adicionalmente, dispone de conectividad Wi-Fi (802.11 b/g/n) y Bluetooth (v4.2 BR/EDR y BLE). Su costo promedio es de S/. 45.00 (ESP32, 2024).

En contraste, el ARDUINO UNO, reconocido como uno de los microcontroladores más populares, está basado en el chip ATMega328P. Posee una memoria Flash de 32KB, una SRAM de 2KB y una EEPROM de I KB. Su velocidad de reloj es de 16MHz y opera con una tensión de 5VDC. Dispone de 14 pines digitales IO, 6 pines de entrada analógica y soporte para I2C (1), UART (1) y SPI (1). A diferencia del ESP-32, no cuenta con conectividad Wi-Fi ni Bluetooth. El ARDUINO UNO tiene un precio promedio de S/. 60.00.

Por último, el ARDUINO NANO representa una versión compacta del ARDUINO UNO, con especificaciones similares. Basado en el chip ATMega328, ofrece una memoria Flash de 32 KB, una SRAM de 2 KB y una EEPROM de un KB. Su velocidad de reloj es de 16MHz y opera con una tensión de 5V DC. Dispone de 14 pines digitales IO, 8 pines de entrada analógica y soporte para I2C (1), UARTs (1) y SPI (1). Al igual que el ARDUINO UNO, carece de conectividad Wi-Fi y Bluetooth. Su precio promedio de S/. 30.00.

En consecuencia, de lo anterior, para la unidad de adquisición, después de un análisis exhaustivo, el microcontrolador ARDUINO NANO es la opción más adecuada para el proyecto. Aunque sus especificaciones son ligeramente inferiores en comparación con el esp-32, su amplia disponibilidad y robustez lo vuelven en la elección óptima para las necesidades de detección y diagnóstico de fallos en un sistema AAP, ya que el esp-32, es más útil en proyectos con comunicaciones inalámbricas, que para este estudio no es el caso y respecto al arduino uno, el arduino nano es una versión más optima debido a la cantidad de sensores que se requiere adquirir. En resumen, el ARDUINO NANO ofrece un equilibrio óptimo entre funcionalidad y costo para nuestro proyecto (ESP32, 2024).

4.5.3 Selección de Sensores

4.5.3.1 Sensor de Corriente. Para seleccionar el sensor ideal para el sistema de este trabajo que se busca evaluar, se comparó las propiedades de los sensores de corriente disponibles: ACS712, INA219 y SCT-013, en la Tabla 4 se describen las características de cada sensor de corriente.

Tabla 4

Comparación de los sensores de corriente

	ACS712	**INA219**	**SCT-013**
Tipo	Efecto Hall	ADC	Transformador CT
Tipo de corriente	AC/DC	DC	AC
Rango de corriente	Hasta 30 A AC/DC	Hasta 26 A DC	Hasta 100 A AC
Sensibilidad	66 mV/A	40 mV/A	100 mV/A
Precisión	±1.5%	±1%	±1%
Interfaz	Salida analógica	I2C/SPI	Salida analógica
Alimentación	5 V DC	3.3 V/5 V DC	No requiere
Aplicaciones típicas	Control de motores, monitorización de corriente	Monitorización de corriente y voltaje en sistemas de bajo consumo	Monitorización de corriente en sistemas de alta corriente
Ventajas	Fácil de usar, bajo ruido, bajo consumo	Alta precisión, interfaz digital, protección de sobrecarga	No invasivo, fácil de usar, adecuado para alta corriente
Precio promedio	S/. 15.00	S/. 25.00	S/. 50.00

El sensor ACS712 es un dispositivo de efecto Hall que mide tanto corriente alterna como corriente continua, con un rango de hasta 30 A. Presenta una sensibilidad de 66 mV/A y una precisión de ±1.5%. Utiliza una interfaz de salida analógica y requiere una alimentación de 5 V DC. Por su facilidad de uso, bajo nivel de ruido y consumo reducido, es óptimo para usos de control de motores y supervisión de corriente. Además, su precio promedio es de S/. 15.00, (alternativa económica).

En contraste, el sensor INA219 emplea un convertidor analógico-digital (ADC) y puede medir corriente hasta 26 A en corriente continua. Ofrece una precisión mayor de ±1% y dispone de una interfaz digital (I2C/SPI). Con un precio promedio de S/. 25.00, es ideal para monitorear corriente y voltaje en sistemas de bajo consumo, gracias a su alta precisión y protección contra sobrecargas.

Por último, el sensor SCT--013 es un transformador de corriente (CT) capaz de medir corriente hasta 100 A en corriente alterna. Con una sensibilidad de 100 mV/A y una precisión de ±1%, su interfaz es analógica y no requiere alimentación externa. Es idóneo para vigilar corriente en sistemas de alta intensidad, aunque su precio promedio es de S/. 50.00.

Por lo tanto, el módulo ACS712 se considera la opción más adecuada para la medición de corriente. Su elección se fundamenta en su capacidad para registrar altos niveles de corriente (dado que el compresor experimenta una corriente de arranque superior a 22 A), su versatilidad para medir tanto corriente alterna como continua y su amplio uso en sistemas de adquisición industrial, según la literatura revisada. Aunque presenta ciertos errores de offset, estos son aceptables dentro del alcance de este estudio. La corriente que circula por la línea genera un campo magnético, el cual es detectado por el sensor de efecto Hall integrado y convertido en un voltaje proporcional que expresa el nivel de corriente.

4.5.3.2 Sensor de Temperatura para el Compresor. Para determinar el sensor más indicado, se examinó las características de los sensores de temperatura sin contacto disponibles: MLX90614, TMP007 y D6T-1A-01, en la Tabla 5 se describen las características de cada sensor de temperatura.

Tabla 5

Comparación de los sensores de temperatura

	MLX90614	**TMP007**	**D6T-1A-01**
Tipo	Infrarrojo	Analógico	Analógico
Rango de temperatura	-40°C a +125°C	-40°C a +125°C	-40°C a +125°C
Precisión	±0.5°C	±1°C	±2°C
Interfaz	I2C	I2C	I2C
Alimentación	3.3 V/5 V DC	2.5 V/5.5 V DC	4.5 V a 5.5 V
Consumo de corriente	<1 mA	240 µA	5 mA
Tiempo de respuesta	~500 ms	33 ms	350 ms
Aplicaciones típicas	Termómetros sin contacto, control de temperatura	Medición de temperatura de objetos y superficies	Detección de presencia humana, monitoreo térmico
Ventajas	Sin contacto, alta precisión, interfaz digital	Tamaño compacto, bajo consumo, termopila integrada	Matriz de sensores, detección de múltiples zonas
Precio promedio	S/. 99.00	S/. 150.00	S/. 250.00

El módulo MLX90614 es un dispositivo infrarrojo que puede medir el rango de temperatura desde -40°C hasta +125°C. Ofrece una precisión de ±0.5°C y utiliza una interfaz digital I2C. Con una alimentación de 3.3 V/5 V DC y un consumo de corriente inferior a 1 mA, es ideal para trabajos que necesitan una alta precisión y sin contacto, como termómetros sin contacto y control de temperatura. Aunque su precio promedio es de S/. 99.00, su alta precisión y facilidad de uso se vuelve una alternativa valiosa.

Por otro lado, el sensor TMP007 es un dispositivo de medición infrarroja con un rango operativo de -40°C a +125°C, aunque con una precisión inferior de ±1°C. También emplea una interfaz I2C, pero su consumo de energía es mayor (240 µA) en comparación con el MLX90614. A pesar de que su tiempo de respuesta es más rápido (33 ms), su costo promedio de S/. 150.00 y su menor precisión lo posicionan como una alternativa secundaria para esta aplicación específica.

A diferencia de otras opciones, el sensor infrarrojo D6T-1A-01 de Omron opera en un rango térmico más restringido, desde -10°C hasta +60°C, con una precisión de ±1.5°C. Si bien puede detectar múltiples zonas de temperatura, su alto consumo eléctrico (5 mA), su rango operativo reducido y su precio de S/. 250.00 lo convierten en una opción menos óptima para la supervisión continua del compresor en sistemas de aire acondicionado.

Por esta razón, el sensor de temperatura MLX90614 es la elección más adecuada. Su alta precisión, funcionamiento sin contacto, su aplicación sin contacto, su amplio uso en los sistemas de adquisición de equipos industriales según los antecedentes debido a que mide la temperatura de una superficie mas no de un punto específico, mejor nivel de precisión y a pesar de que en la literatura señalan que tiene errores en temperaturas bajas se torna como elección óptima para el proceso FDD del aire acondicionado de precisión, porque las temperaturas que mide del compresor son mayores a 15°C.

4.5.3.3 Sensor de Voltaje. Para la elección del sensor de voltaje ideal para el sistema de este proyecto que se busca evaluar, se comparó tres alternativas disponibles: ZMPT-I0IB, ZMPT102 y ZMPT107.

Posteriormente, en la Tabla 6 se señalan las características de cada sensor de voltaje con el objetivo de seleccionar la más apropiada.

Tabla 6

Comparación de los sensores de voltaje

	ZMPT101B	**ZMPT102**	**ZMPT107**
Tipo	AC	AC	AC
Rango de voltaje	0-250 V AC	0-250 V AC	0-250 V AC
Frecuencia	50/60 Hz	50/60 Hz	50/60 Hz
Salida	Analógica	Analógica	Analógica
Sensibilidad	Ajustable	Ajustable	Ajustable
Precisión	Alta	Alta	Alta
Alimentación	5 V DC	5 V DC	5 V DC
Consumo de corriente	Bajo	Bajo	Bajo
Aplicaciones típicas	Medición de voltaje AC, monitoreo de voltaje en sistemas eléctricos	Medición de voltaje AC, monitoreo de voltaje en sistemas eléctricos	Medición de voltaje AC, monitoreo de voltaje en sistemas eléctricos
Precio promedio	S/. 20.00	S/. 30.00	S/. 50.00

El módulo ZMPT102 guarda similitudes con el 101 en varios aspectos. Ambos son sensores de voltaje de corriente alterna (AC) con capacidad para medir en un rango de 0-250 V AC y operan a una frecuencia de 50/60 Hz. Tanto el ZMPT102 como el sensor de la figura 11 ofrecen una salida analógica, permiten ajustar la sensibilidad, poseen alta precisión y presentan un consumo de corriente bajo. Sin embargo, el ZMPT102 tiene un precio promedio ligeramente superior de S/. 30.00.

Por otro lado, el sensor ZMPT107 también es un sensor de voltaje de corriente alterna (AC) con un rango de medición de 0-250 V AC y una frecuencia de operación de 50/60 Hz. Al igual que los otros sensores, proporciona una salida analógica, posibilita ajustar la sensibilidad, presenta alta precisión y un bajo consumo de corriente. No obstante, su precio promedio es más elevado, alcanzando los S/. 50.00.

En vista de lo anterior, para el sensor de voltaje; luego de analizar las opciones, se concluye que el módulo ZMPT/101B resulta la alternativa más idónea para el sistema en evaluación. A pesar de compartir similitudes con los otros sensores, como un rango de medición adecuado, alta precisión y bajo

consumo de corriente y disponibilidad en el mercado. Principalmente su habilidad de ajuste de sensibilidad y versatilidad para adecuarse a diversas condiciones de voltaje lo vuelven en una opción versátil y adecuada para la obtención de voltaje del aire acondicionado, también la diferencia de costo lo hace más elegible porque en el diseño de un sistema es importante el aspecto económico. En resumen, el sensor seleccionado ofrece la combinación ideal de funcionalidad, precisión y costo.

4.5.3.4 Sensor de Presión. Para la selección del sensor de presión más idóneo para el sistema de este proyecto que se busca evaluar, se comparó tres alternativas: el SPKT0043P, el BMP280 y el MPXV7002DP.

Seguidamente, en la Tabla 7 se resume las características de cada sensor de presión para tomar una decisión informada.

Tabla 7

Comparación de los sensores de presión

	SPKT0043P	**BMP280**	**MPXV7002DP**
Tipo	Industrial	Digital	Diferencial
Rango de presión	6 Mpa	300-1100 hPa	-2 kPa a 2 kPa
Precisión	±0.8% F.S.	±1 hPa	±1.5 % FS
Interfaz	Analógica	I2C y SPI	Analógica
Consumo de energía	Variable	Bajo	Variable
Tamaño	Variable	Pequeño y compacto	Pequeño
Alimentación	5V	3.3 V	5 V
Aplicaciones típicas	Sistemas industriales	Meteorología, GPS	Climatización, HVAC
Precio promedio	S/. 300.00	S/. 15.00	S/. 100.00

El sensor SPKT0043P, clasificado como industrial, ha sido diseñado para aplicaciones que demandan mediciones precisas de presión en entornos industriales. Con un amplio rango de presión de hasta 6 Mpa y una precisión de ±0.8% F.S., este sensor opera con una interfaz analógica y requiere una alimentación de 5V. Aunque su consumo energético puede variar y su tamaño dependerá del modelo, es

idóneo para aplicaciones industriales que requieran mediciones precisas de presión. No obstante, su precio promedio es relativamente alto, alcanzando los S/. 300.00.

En contraste, el sensor BMP280 es de tipo digital y está principalmente destinado a aplicaciones meteorológicas y de posicionamiento global (GPS). Con un rango de presión más limitado de 300-1100 hPa y una precisión de ±1 hPa, este sensor ofrece una interfaz digital compatible con I2C y SPI, con un consumo de energía bajo de 3.3 V. Su diseño compacto resulta útil donde se requiere un espacio reducido, y su precio promedio es de aproximadamente S/. 15.00.

Por último, el sensor MPXV7002DP, diferencial, está diseñado para aplicaciones de climatización y sistemas HVAC. Con un rango de presión de -2 kPa a 2 kPa y una precisión de ±1.5% FS, este sensor opera con una interfaz analógica y requiere una alimentación de 5V. Aunque su consumo de energía varía, su tamaño compacto lo hace versátil. Sin embargo, su precio promedio es más alto, llegando a los S/. 100.00.

Por consiguiente, para el sensor de presión; tras analizar las opciones, se considera que el tipo de sensor SPKT0043P es la elección más apropiada, ya que el sistema evaluado es un circuito herméticamente cerrado, que cuenta con dos principales etapas de alta y baja presión, que dispone de válvulas de servicio para poner sensores de presión. A pesar de su precio relativamente alto, su amplio rango de presión y alta precisión lo hacen ideal para nuestras necesidades de detección y diagnóstico de fallas. Su diseño industrial asegura una mayor robustez, confiabilidad en ambientes de operación exigentes y su amplio uso en los sistemas de aire acondicionados de precisión y sistemas HVAC industriales según los fabricantes debido a que mide la presión del sistema directamente. En resumen, el sensor SPKT0043P ofrece la combinación perfecta de rendimiento y confiabilidad para nuestro proyecto.

4.5.3.5 Sensor de Temperatura para el Evaporador y Condensador. Para determinar el sensor más adecuado de estos elementos críticos para la detección de fallas como la reducción del flujo de aire en el condensador y la reducción del flujo de aire en el evaporador, se examinó las características de los sensores: DS18B20, LM35 y TMP36. En seguida, en la Tabla 8 se expresa las características de cada sensor.

Tabla 8

Comparación de los sensores de temperatura

	DS18B20	LM35	TMP36
Tipo	Digital	Analógico	Analógico
Rango de temperatura	-55°C a +125°C	-45°C a +149°C	-40°C a +125°C
Precisión	±0.5°C	±0.4°C	±2°C
Interfaz	1-Wire	Analógica	Salida analógica
Alimentación	3V/5 V DC	5 a 30 V	2.7 V a 5.5 V
Consumo de corriente	1 mA	2 mA	50 µA
Tiempo de respuesta	~750 ms	Rápido	750 ms
Aplicaciones típicas	Control de temperatura en sistemas de bajo consumo	Control de temperatura en sistemas de bajo consumo	Monitoreo de temperatura en sistemas de bajo consumo
Ventajas	Preciso, fácil de usar, interfaz digital, económico, encapsulado a prueba de agua	Bajo costo, fácil de usar, salida lineal	Fácil de usar, bajo consumo de corriente, amplio rango de temperatura
Precio promedio	S/. 10.00	S/. 5.00	S/. 10.00

El sensor DSB18B20 es un dispositivo digital que también mide un rango de temperatura desde -55°C hasta +125°C, con una precisión de ±0.5°C. Utiliza una interfaz digital de 1-Wire y requiere una alimentación de 3 V a 5.5 V, con un consumo de corriente de aproximadamente 1 mA. Con un precio promedio de S/. 10.00, es una opción precisa y económica para la evaluación de temperatura en sistemas de bajo consumo.

Por otro lado, el LM35 es un sensor de temperatura analógico con rango de operación que va desde -45.5°C a 148.8°C, con una precisión típica de ±0.4°C a temperatura ambiente. La interfaz del LM34 consiste en una salida de voltaje lineal, donde cada incremento de 10 mV representa 1°C. Con un precio promedio de S/. 5.00, es una opción económica, aunque su uso está más orientado a entornos controlados debido a su susceptibilidad a ruido eléctrico.

En contraste, el sensor TMP36 es un dispositivo analógico que puede calcular temperaturas en el rango de -40°C a +125°C, con una precisión de ±2°C. Su interfaz es una salida analógica y requiere una

alimentación de 2.7 V a 5.5 V, con un consumo de corriente de aproximadamente 50 µA. Aunque tiene un precio promedio similar al DSB18B20 de S/. 10.00, su precisión es menor y su aplicación se centra en el monitoreo de temperatura en sistemas de bajo consumo.

Por consiguiente, en este estudio resulta fundamental medir la temperatura en diversos puntos del sistema de refrigeración, siendo el sensor DS18B20 una opción óptima para esta tarea. Su interfaz digital facilita la conexión de múltiples sensores a un único pin del microcontrolador. Además, su elevada precisión asegura mediciones confiables incluso en entornos de baja temperatura. Su encapsulado resistente al agua, ideal para condiciones de condensación en el sistema evaluado, y su inmunidad al ruido eléctrico, contribuyen a la integridad de los datos en aplicaciones industriales.

4.5.4 Elección del Sistema de Base de Datos

Se realizó un estudio comparativo de opciones para almacenar y gestionar la información recopilada. Las opciones evaluadas incluyen InfluxDB, MySQL (o MariaDB) y SQLite.

Tabla 9

Comparación de sistema de base de datos

	INFLUX-DB	**MYSQL MARIADB**	**SQLite**
Tipo de base de datos	orientada a series temporales	Relacional	Relacional
Escalabilidad	Moderada	Alta	Moderada
Lenguaje de consulta	InfluxQL (similar a SQL)	SQL	SQL
Soporte ACID	NO	SI	SI
Escalabilidad horizontal	NO	SI	NO
Modelo de datos	datos con timestamp)	Esquema fijo	Esquema fijo
Comunidad y soporte	Comunidad activa, documentación adecuada	Amplia comunidad y documentación extensa	Amplia comunidad y documentación
Flexibilidad en esquema	Alta	Baja	Alta
Velocidad de lectura/escritura	Moderada	Alta	Alta
Tamaño máximo de base de datos	Depende de la configuración y recursos	Ilimitado	2 terabytes

InfluxDB se destaca por ser una base de datos orientada a series temporales, lo que la hace idónea para gestionar datos que evolucionan con el tiempo, como los registros de sensores en un sistema de aire acondicionado. Aunque ofrece una escalabilidad moderada y una alta flexibilidad en la estructura de datos, no es completamente compatible con el estándar ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). A pesar de ello, cuenta con una comunidad activa y una documentación exhaustiva. Su capacidad de escalar horizontalmente es limitada y su rendimiento de lectura/escritura es moderado.

Por otro lado, MySQL (o MariaDB) es una base de datos relacional reconocida por su alta escalabilidad y su amplia base de usuarios. Utiliza SQL como lenguaje de consulta y ofrece soporte completo para el estándar ACID, asegurando la consistencia y la integridad de los datos. Asimismo, brinda una flexibilidad moderada en la estructura de datos y una velocidad de lectura/escritura elevada. Además, su capacidad máxima de almacenamiento es prácticamente ilimitada, lo que la hace adecuada para sistemas que generan grandes volúmenes de datos.

Por último, SQLite es otra opción de base de datos relacional que comparte algunas características con MySQL en cuanto a flexibilidad y rendimiento de lectura/escritura. Sin embargo, su capacidad de escalamiento y su límite máximo de almacenamiento están limitados a 2 terabytes, lo que podría ser una restricción en sistemas con una gran acumulación de datos a lo largo del tiempo.

Tras un análisis exhaustivo, se ha concluido que MySQL es la opción más idónea para el sistema de detección y diagnóstico de fallos. Su alta escalabilidad, soporte completo para ACID y extensa comunidad de usuarios la convierten en la solución ideal para gestionar eficaz y fiablemente los datos en nuestro proyecto. En resumen, MySQL ofrece un equilibrio óptimo entre rendimiento, confiabilidad y flexibilidad para las necesidades específicas.

4.5.5 Elección de Refrigerante

Los refrigerantes son sustancias químicas que se usan en los sistemas de aire acodiciados para transferir el calor de un lugar a otro. Existen una gran variedad de tipos de refrigerantes que se usan en

los sistemas AAP. Algunos de los refrigerantes más comunes incluyen el “R-22, R-410A, R-404A, R-407C, R-134ª” y el nuevo estándar, R-32.

Seguidamente, en la Tabla 10 se describen las características de cada tipo de refrigerante con el objetivo de seleccionar la más apropiada.

Tabla 10

Comparación de tipos de refrigerantes

	R134a	**R410a**	**R22**
Tipo de refrigerante	HFC hidroclorofluoro carbono	HFC hidroclorofluoro carbono	HCFC hidroclorofluoro carbono
Potencial de calentamiento global	1400	2088	3490
Potencial de destrucción del ozono (POD)	0	0	0.05
Presión de trabajo	Baja	Alta	Media
Eficiencia energética	Media	Alta	Baja
Inflamabilidad	No inflamable	No inflamable	Levemente inflamable
Toxicidad	Baja	Baja	Moderada
Aplicaciones	Refrigeración comercial, aire acondicionado de presición y automotriz	Aire acondicionado residencial y comercial	Refrigeración comercial e industrial (antiguamente)

Pero por recomendación del fabricante del sistema de aire acondicionado en evaluación, y características del sistema, recomienda usar el R-134a.

4.5.6 Elección del Lenguaje de Programación

Durante la etapa de diseño es crucial seleccionar un lenguaje de programación apropiado para desarrollar el modelo clasificador. En este contexto, se optó por utilizar Python como el principal lenguaje de programación por sus características.

Python es altamente adaptable y fácil de utilizar, lo que lo convertirá en una elección idónea para la elaboración de programas complejos requeridos para la implementación de algoritmos de aprendizaje.

Algunas de las características más notables de Python en este ámbito incluirán:

- Su versatilidad permite abordar una gran variedad de tareas de programación, así mismo el desarrollo de sistemas de detección y diagnóstico de fallos.

- La orientación a objetos de Python, lo que simplifica la integración con otros sistemas y facilita la gestión de la complejidad del proyecto.

- La sintaxis clara y legible de Python, hace que la escritura y comprensión del código sea accesible, resultando especialmente beneficioso en estudios de evaluación de datos.

- La disponibilidad de una extensa variedad de librerías y frameworks especializados para el estudio de datos y el aprendizaje automático, como Scikit-learn, NumPy, Matplotlib, Iphyton y Pandas. Estas bibliotecas proporcionan funciones y algoritmos predefinidos que simplificarán la etapa de ejecución y acelerarán la implementación de modelos ML (Scikit learn, 2007-2024).

En conclusión, Python se presenta como la opción idónea para el trabajo de detección y diagnóstico de fallos, ya que proporciona un grupo entero de instrumentos y bibliotecas para el uso de algoritmos, como SVM, KNN, RF, DT, NB, GB etc. Lo que permitirá clasificar con precisión si el aire acondicionado está funcionando correctamente o presenta alguna anomalía. Su flexibilidad, facilidad de uso y gran comunidad de usuarios respaldan su elección como la opción óptima para este proyecto (Python, 2024).

4.6 Diseño del Subsistema de Instrumentación

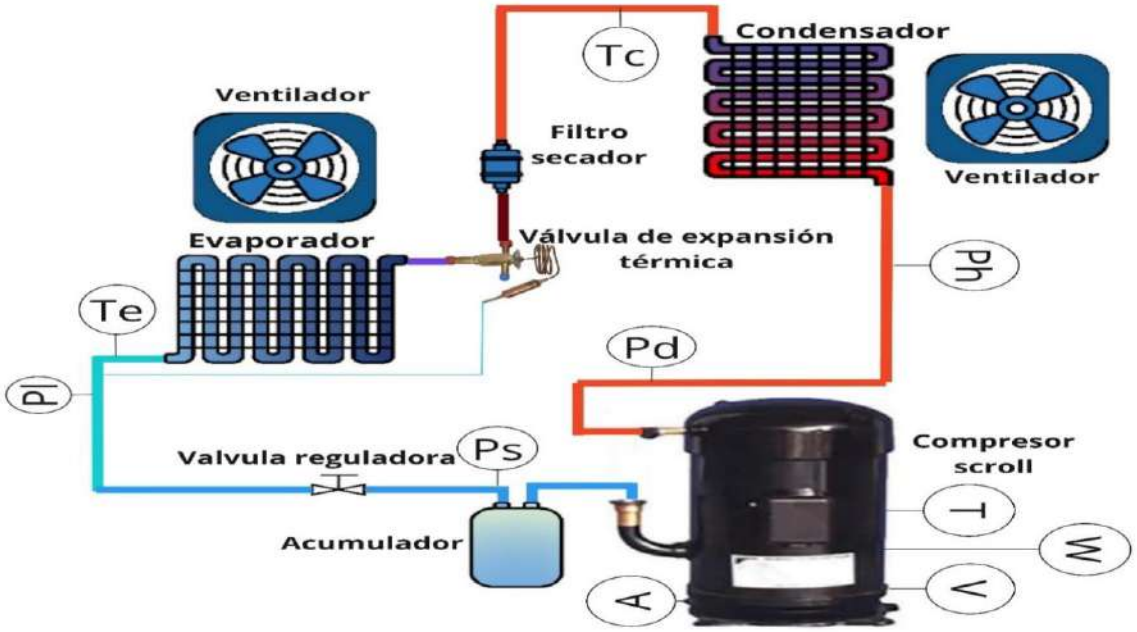
Constituye el primer paso en el desarrollo de una solución tecnológica para detectar fallas en el sistema AAP e involucra la selección, diseño y ubicación de los sensores y protocolos de comunicación.

4.6.1 Diseño y Ubicación los Sensores en el Sistema AAP

El esquema ilustrado en la Figura 33 representa la disposición estratégica de los sensores clave dentro del sistema AAP, encargados de registrar tanto variables directas como indirectas, con el fin de alcanzar los objetivos del estudio.

Figura 33

Diagrama esquemático de instrumentación y sensores en el sistema AAP



Nota: *Representación del d*iseño y ubicación de sensores de P, T, I y V en el sistema AAP.

Tabla 11

Estadísticas de las variables (reglas) para el diagnóstico del clasificador

Variable de Entrada	Unidad	promedio	Mínimo	Máximo
Tcomp (T)	°C	35	15	60
Icomp (A)	A	3	2.5	23
Vcomp (V)	V	222	215	229
Plowpres (Pl)	kPa	40	5	65
Phightpres (Ph)	kPa	100	65	190
Psuc (Ps)	kPa	40	1	65
Pdesc (Pd)	kPa	100	65	190
Tevap (Te)	°C	25	-3	19
Tcond (Tc)	°C	25	18	35

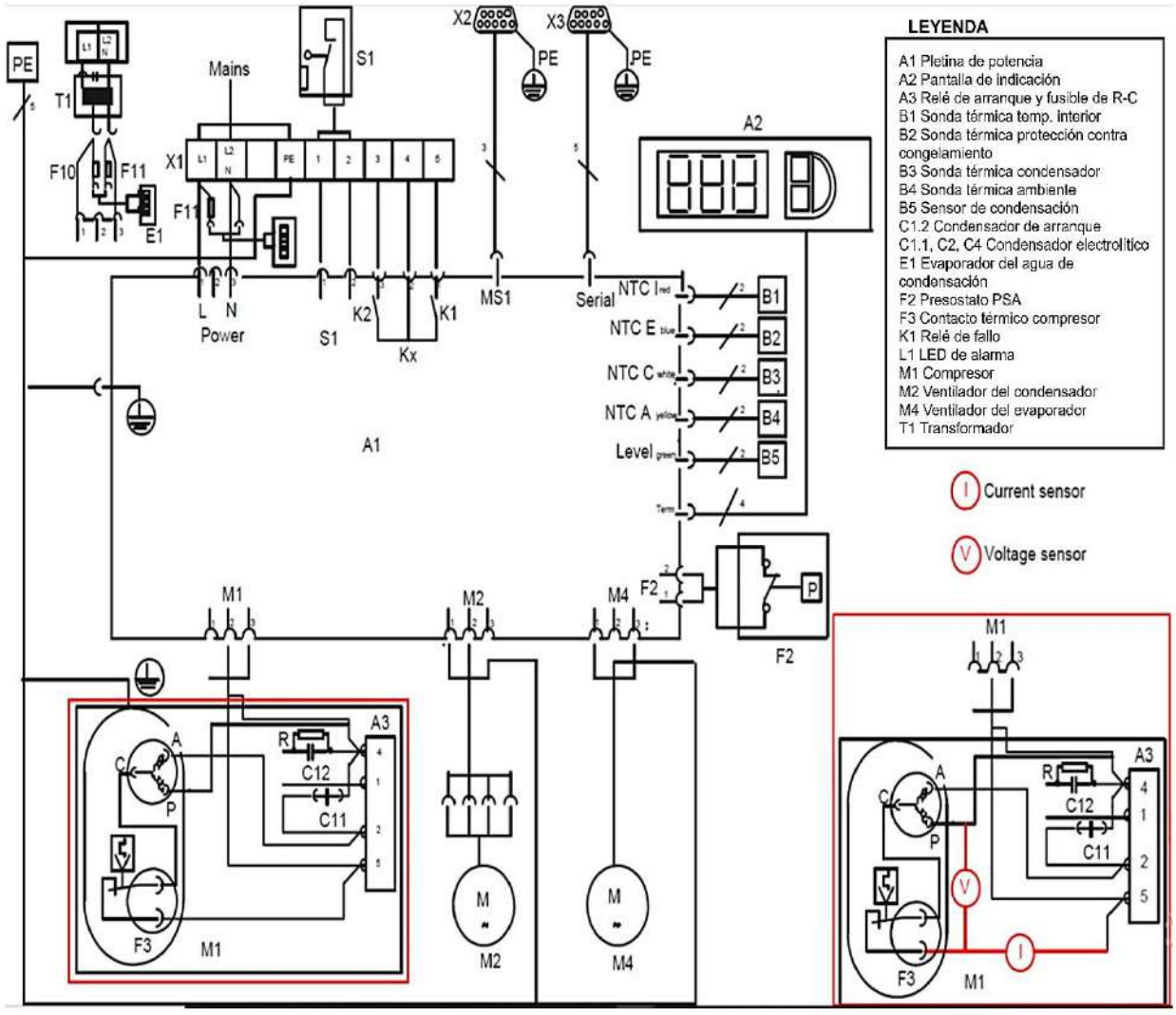
En la Tabla 11 se detallan los valores estadísticos clave (media, máximo y mínimo) de cada variable, extraídos de la base de datos generada. Esta base contenía inicialmente más de 31,000 muestras de 10 variables diferentes, considerando cinco tipos de fallas. Con el objetivo de mejorar la representatividad de cada condición, se llevó a cabo un balanceo de datos, reduciendo el conjunto a 20,000 muestras. Para el cálculo de estos valores descriptivos, se utilizaron funciones de Python como `describe()` de pandas y métodos de NumPy (`df[columna].mean()`, `df.min()`, `df.max()`), lo que

permitió automatizar la obtención de resultados. Esta información es crucial para definir los rangos normales de operación y establecer los límites utilizados en las reglas de diagnóstico del sistema.

Para proporcionar una representación más detallada de la ubicación de los sensores eléctricos, que no pueden ser visualizados con precisión en el diagrama de instrumentación de la figura 33, se emplea la figura 34. En esta se muestra el esquema de conexiones eléctricas del sistema AAP de la marca Rittal. En dicho diagrama, el sensor de corriente está instalado en la línea correspondiente al borne común del compresor, mientras que el sensor de voltaje se sitúa entre las líneas de los bornes común y de trabajo.

Figura 34

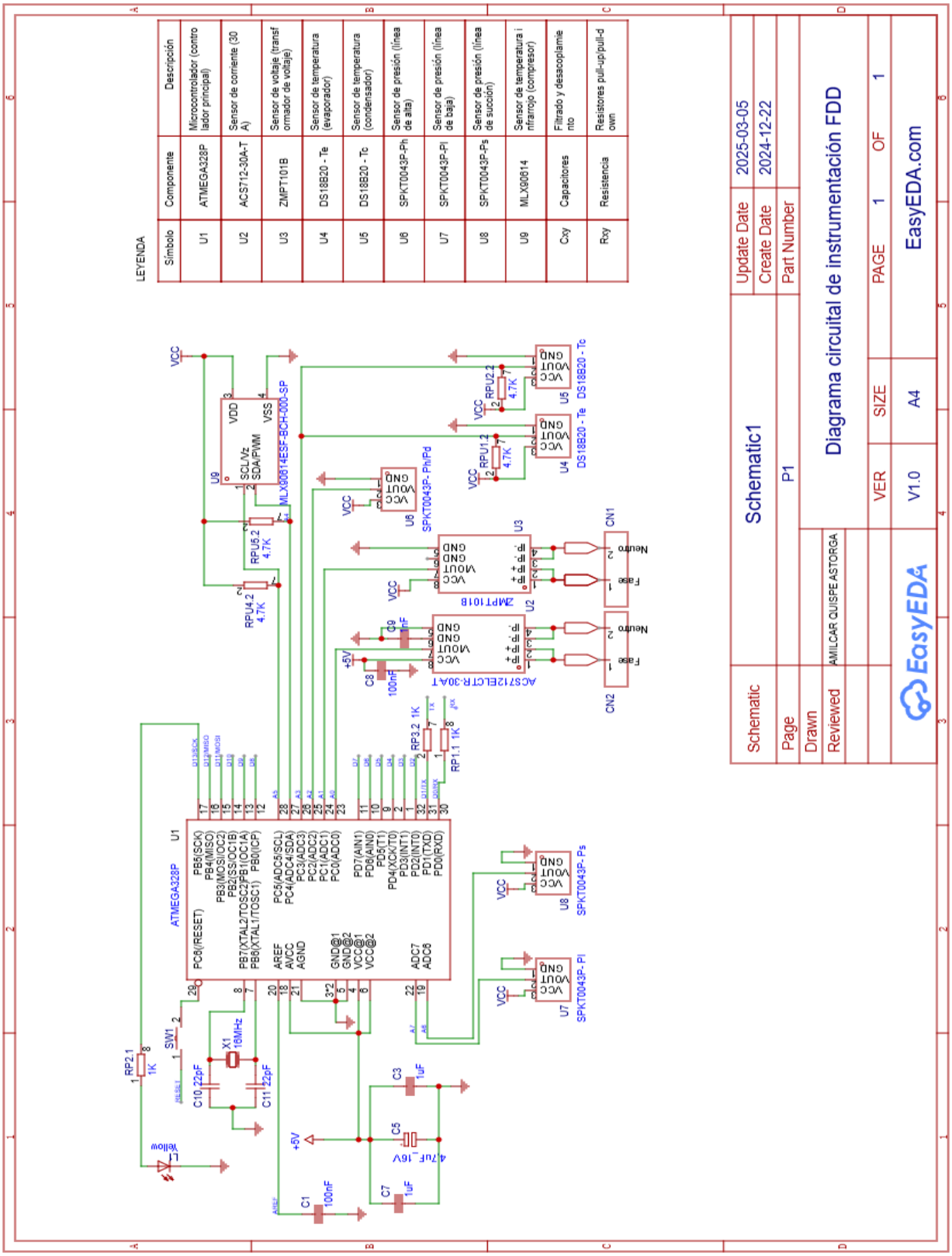
Diagrama específico de instrumentación de naturaleza eléctrica en el sistema AAP



Nota: Diseño y ubicación de sensores de corriente y voltaje en el sistema AAP*, fuente adaptada Rittal.*

Figura 35

Diagrama esquemático circuital de instrumentación (módulo adquisición)



En la figura 35 se representa el diagrama circuital, específicamente de la etapa de adquisición, etapa donde interactúa el microcontrolador Arduino nano y los sensores seleccionados, en la sección 4.5.

4.7 Diseño del Subsistema de Detección

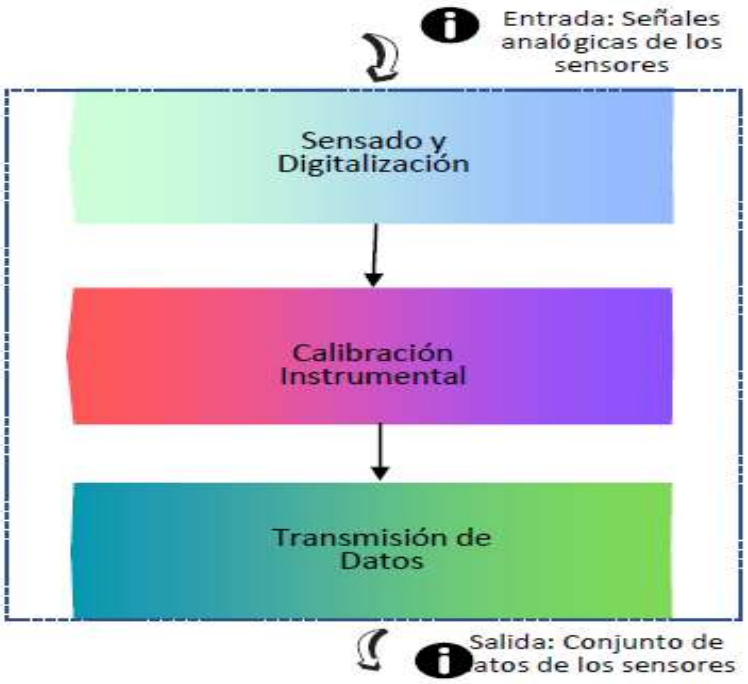
El diseño del subsistema de detección constituye una etapa principal en el desarrollo de una herramienta tecnológica para clasificar las fallas del sistema AAP. Comprende la elección y configuración de algoritmos, así como la definición de la arquitectura del subsistema y los criterios de diagnóstico.

Consta de 7 módulos: el módulo de adquisición de señales, de generación de base de datos, de gestión de dataframe, de entrenamiento del modelo, de adquisición de señal para procesamiento en tiempo real, de clasificación de fallas y de visualización del resultado. La arquitectura del sistema completo, se basa en dos etapas que pueden compartir funcionalidades.

4.7.1 Módulo de Adquisición de Señales

Este módulo se estructuro en torno a tres etapas fundamentales: Sensado y digitalización de la señal, calibración instrumental y trasmisión de datos. La figura 36 detalla el diseño.

Figura 36
Arquitectura del módulo adquisición de señales



4.7.1.1 Sensado y Digitalización de la Señal

Estos procesos se desarrollan entre los sensores y la unidad de adquisición.

Para el caso del sensor de corriente, se tomó las señales de corriente alterna del compresor (220v, 60 Hz) y se transforman en niveles de intensidad de corriente, por su naturaleza analógica, se requiere una etapa conversor de analógico a digital.

Para el caso del sensor de voltaje, se tomó las señales de voltaje alterna del compresor (220v, 60 Hz) y se transforman en niveles de intensidad de voltaje, por su naturaleza analógica, se requiere una etapa conversor de analógico a digital.

Para el caso del sensor de presión, se tomó las señales físicas de presión, en los diferentes puntos del sistema AAP y se transforman en señales eléctricas analógicas, por su naturaleza analógica, se requiere una etapa conversor de analógico a digital.

En la situación de los sensores de temperatura, se tomó las señales físicas de temperatura, uno del compresor (220v, 60 Hz) y se transforman en señales digitales, la circuitería del sensor infrarrojo realiza la adquisición, el procesamiento y la conversión de la señal infrarroja en una lectura de temperatura, dos los niveles de temperatura en la parte de la salida del condensador y en la salida del evaporador, mediante sensores de tipo termocupla, que sean a prueba de agua (protegidos mediante capsula de metal).

De acuerdo con las especificaciones del digitalizador, este opera con un reloj de frecuencia de 16 MHz y con 10 bits de profundidad. La tasa de muestreo define la resolución temporal, aunque en este caso la resolución se expresa en bits. Con una profundidad de 10 bits, el digitalizador puede capturar hasta 2^10 (1024) valores discretos (Arduino.cc, 2024).

También es crucial considerar el elemento de alimentación para los sensores. Dado que los sensores están integrados con el Arduino Nano, recibirán energía de la misma fuente que alimenta al Arduino Nano. Para un funcionamiento óptimo, la alimentación recomendada para el Arduino Nano es de 5V-12V en corriente continua (consultar anexo 03).

Aunque existen filtros digitales, como los de paso bajo o algoritmos como RUSboost, diseñados para eliminar el ruido y suavizar las señales, la decisión de no aplicarlos en el procesamiento de señales responde a un enfoque metodológico intencional, basado en criterios técnicos y experimentales. Se busca preservar la integridad de la información crítica necesaria para detectar condiciones incipientes de falla, ya que un suavizado excesivo podría atenuar o eliminar variaciones sutiles que son esenciales para el diagnóstico. En su lugar, se adoptó una estrategia de discriminación selectiva que descarta únicamente los valores extremos estadísticamente improbables, conservando así la información relevante.

seleccionados. Para la medición de corriente, se ha elegido el sensor de efecto Hall ACS712-30A, a pesar de su susceptibilidad al ruido debido a su naturaleza de tipo Hall y al sistema en evaluación. Esta decisión se justifica por sus ventajas frente a sensores que utilizan resistencias Shunt, las cuales generan caídas de tensión significativas y presentan riesgos de corrientes de retorno que podrían dañar tanto el convertidor analógico-digital (ADC) como la plataforma Arduino. En contraste, el ACS712 reduce las caídas de tensión, lo que permite realizar mediciones más seguras y confiables.

Como parte de los criterios de diseño, se definió que el sistema debe tener la capacidad de medir corrientes de hasta ±30 A, coincidiendo con el rango de operación del sensor. Este rango asegura un monitoreo adecuado de las corrientes generadas en el sistema de aire acondicionado. En la figura 8 se ilustra el módulo del sensor ACS712-30A empleado, que incorpora los componentes sugeridos en la hoja de datos correspondiente.

El sensor ACS712 cuenta con una sensibilidad nominal de 66 mV/A y un voltaje de offset de aproximadamente 2.5 V. Esto significa que, al detectar su valor máximo de 30 A, genera una señal proporcional de 30 A × 66 mV/A = 1.98 V. Por lo tanto, la salida del sensor para 30 A es 2.5 V + 1.98 V = 4.48 V, mientras que, para corrientes negativas equivalentes, la salida sería 2.5 V - 1.98 V = 1.52 V. En resumen, el rango de salida del sensor oscila entre aproximadamente 1.5 V y 4.5 V. Dado que el ADC del Arduino admite entradas de 0 a 5 V, no es necesario realizar una amplificación adicional de la señal.

Cálculos asociados, para obtener el valor de la corriente efectiva Irms, primero se calcula el voltaje de pico a pico Vpp de la señal de salida, luego V_{RMS} y finalmente se determina I_{RMS}

$$V_{PP} = \frac{(V_{max} - V_{min}).5.0}{1024.0}$$

$$V_{RMS} = \frac{V_{pp}.0.707}{2.0} \qquad (25)$$

$$I_{RMS} = \frac{(V_{RMS} - 2.5V).1000}{66}$$

Donde V_{max} Y V_{min} son los valores máximo y mínimo de la señal digitalizada por el ADC. El uso de un condensador C en la salida 10 nF según la hoja de datos ayuda a filtrar y reducir el ruido.

Los cálculos se encuentran implementados en el código del sistema a través de la función `obtenerCorrienteRMS`, como en el Anexo 09. Cabe destacar que el módulo ACS712 incorpora los componentes esenciales para su operación, por lo que no se requiere un circuito adicional.

Para la señal de voltaje se utiliza el sensor ZMPT, diseñado específicamente para la medición de voltajes de corriente alterna mediante un transformador integrado. Este sensor destaca por su capacidad para proporcionar señales proporcionales y aisladas, mejorando la seguridad del sistema.

El proceso de medición de voltaje incorpora un acondicionamiento básico, basado en un divisor de voltaje que emplea un potenciómetro multivuelta de 100 kΩ. Este potenciómetro regula la ganancia del amplificador operacional interno, permitiendo controlar la sensibilidad. Opcionalmente, se puede añadir un filtro RC, formado por una resistencia de 1 kΩ y un condensador de 100 nF, para minimizar el ruido. Teniendo en cuenta el voltaje de offset de Vcc/2 (2.5 V), se ajusta el potenciómetro mientras se supervisa la salida hasta obtener el valor deseado, garantizando que no haya saturación en la señal.

El sensor elegido tiene una sensibilidad ajustable mediante un potenciómetro integrado. En este proyecto, la sensibilidad se calibró a 325.25 mV/V lo que garantiza mediciones exactas en el rango de funcionamiento del sistema de aire acondicionado, la función general $V_{out} = Sensibilidad * V_{in.}$

El voltaje RMS se calcula directamente mediante las funciones integradas en la librería del sensor seleccionado. Este enfoque utiliza el promedio cuadrático (RMS) de la señal en un intervalo específico, asegurando mediciones precisas y consistentes. En la figura 11 se muestra el módulo del sensor seleccionado, el cual incorpora los componentes sugeridos por el fabricante.

Para la medición de presión se seleccionaron cuatro sensores SPKT00403P, diseñados para medir presiones en rangos de 0 a 250 psi. Estos sensores generan señales analógicas proporcionales a la presión.

Cada sensor presenta un voltaje de salida descrito por la ecuación:

$$\mathrm{V_{out}} = \frac{analogRead * 5.0}{1024.0}$$

$$\mathrm{P_{kpa}} = \frac{(\mathrm{V_{out}} - 0.454) * 250}{6.894744} \qquad (26)$$

Donde 0.454 V corresponde al voltaje de compensación (OFFSET) en condiciones de presión nula.

La primera ecuación $\mathrm{V_{out}}$, es del voltaje de salida y en la ecuación $\mathrm{P_{kpa}}$ se elimina el voltaje de compensación, también se puede convertir a otras unidades utilizando relaciones (1 kPa=0.145038 psi).

Esta medición cuenta con un acondicionamiento interno que incluye amplificación, compensación de temperatura y calibración de fábrica. En su diseño, incorpora un diafragma de silicio con un puente Wheatstone integrado, formado por cuatro resistencias piezoresistivas dispuestas en puente. Además, cuenta con un amplificador interno cuya ganancia está calibrada de fábrica, junto con un circuito de compensación de offset y un sensor de temperatura integrado que garantiza la compensación térmica.

Para que los sensores MLX90614 y DS18B20 funcionen adecuadamente, es necesario utilizar resistencias pull-up, tal como lo indican sus especificaciones técnicas. En el caso del MLX90614, que emplea el protocolo de comunicación I2C, se conecta una resistencia pull-up de 4.7 kΩ a las líneas SDA y SCL. Mientras tanto, el sensor DS18B20, que opera bajo el protocolo One-Wire, requiere una resistencia pull-up de 4.7 kΩ entre su línea de datos y la fuente de alimentación (Vcc). Esta configuración asegura que la línea de datos se mantenga en un nivel lógico alto cuando no hay actividad de transmisión. Los valores de resistencia seleccionados siguen las recomendaciones del fabricante.

4.7.1.2 Calibración Instrumental

Este proceso, es una etapa muy importante, luego de haber sensado y digitalizado las señales de los sensores, ya tenemos valores de corriente en amperios, voltaje en voltios, presión en kPa y temperatura en grados Celsius, pero estos valores no necesariamente son los correctos.

Considerando que la calibración instrumental de sensores representa un proceso esencial en el ámbito de la instrumentación y la medición, imprescindible para garantizar la exactitud y la fiabilidad de las métricas conseguidas. Este proceso se centra en asegurar que los sensores o instrumentos de medición produzcan lecturas exactas y repetibles de acuerdo con estándares reconocidos y aceptados. Para lograrlo, se seleccionan estándares de referencia confiables y trazables, como equipos de medición certificados por laboratorios acreditados, que sirven como punto de comparación para las lecturas del sensor en cuestión. La calibración conlleva un conjunto de pasos que van desde la preparación adecuada del equipo hasta la ejecución del procedimiento de calibración, que puede incluir la aplicación de estímulos conocidos al sensor y la comparación de sus respuestas con los valores esperados. En caso de detectar desviaciones o errores durante la calibración, se realizan ajustes y correcciones en el sensor para mejorar su precisión. La calibración de sensores es de vital importancia que trasciende campos de diversas disciplinas, siendo esencial tanto en la industria como en la investigación, ya que sensores calibrados adecuadamente garantizan mediciones precisas y confiables, fundamentales para las decisiones y la garantía de la calidad en diversos procesos y sistemas (Gutierrez & Ituralde, 2017).

Para este trabajo se calibro a nivel software (recomendación de los fabricantes) y a nivel hardware mediante la comparación de las mediciones con instrumentos certificados, ver anexos 28, 30, 32, 34, 36, donde se representa hojas de procedimiento y datos de calibración para cada tipo sensor.

4.7.1.3 Transmisión de Datos

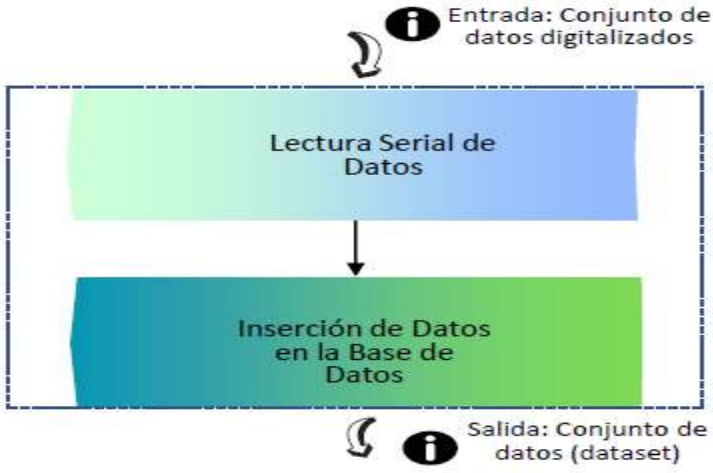
Mediante la comunicación serial, en el contexto de sistemas de adquisición de señales, representa un método fundamental para la transferencia de datos entre dispositivos electrónicos, haciendo posible

la relación y el intercambio secuencial de información, bit a bit, a través de un solo canal de transmisión. En el presente trabajo, se empleó esta forma de comunicación para establecer una conexión fiable y eficiente entre el microcontrolador Arduino Nano, encargado de la captura de datos suministrados por los sensores, y el elemento de procesamiento Raspberry Pi. En este sistema, el Arduino Nano actúa como la unidad de adquisición de señales, mientras que la Raspberry Pi funge del elemento de procesamiento principal. La conexión entre ambos dispositivos se ejecutó por medio de un enlace serial por el puerto USB, lo que habilita la transferencia de información de manera bidireccional y en tiempo real. Esta comunicación serial otorga una alternativa robusta y escalable para la integración de múltiples sensores con la unidad de procesamiento, permitiendo la recolección y el estudio eficiente de datos en aplicaciones de monitoreo, control y diagnóstico de sistemas como el aire acondicionado de precisión abordado en este trabajo. Además, la comunicación serial ofrece la flexibilidad necesaria para adaptarse a diferentes entornos y requisitos de diseño, lo transforma en una alternativa óptima para la implementación del sistema de adquisición de señales en un amplio espectro de campos industriales y científicas. Es claro que se puede usar otras formas de comunicación, con las de naturaleza inalámbrica, pero para los objetivos y limitaciones del trabajo se elige el tipo serial.

4.7.2 Módulo de Generación de Base de Datos

Figura 37

Arquitectura del módulo de generación de base de datos



4.7.2.1 Lectura Serial de datos

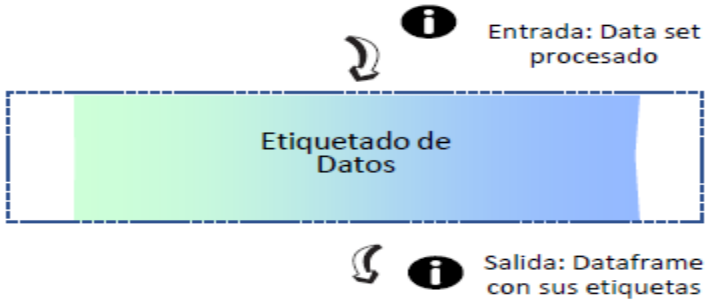
En esta fase se lleva a cabo la lectura de datos, donde se procede a obtener los datos del puerto serial línea por línea. Para lograr esto, se configura la conexión con el puerto serial, el cual se detectó automáticamente mediante comandos.

4.7.2.2 Inserción de Datos en la Base de Datos

En esta parte se establecio la conexión con la base de datos, básicamente se autentica todos los accesos necesarios (usuarios, contraseñas y la base de datos), convertimos la estructura de datos de cada línea recibida en un diccionario JSON, para su mejor interpretación luego insertamos la información ya en la base de datos mysql, mediante comandos de lenguaje sql (query), al final, desconectamos la conexión a la base de datos (Oracle, 2024).

4.7.3 Módulo de Gestión de Data Frame

Figura 38
Arquitectura del módulo de gestión de data frame



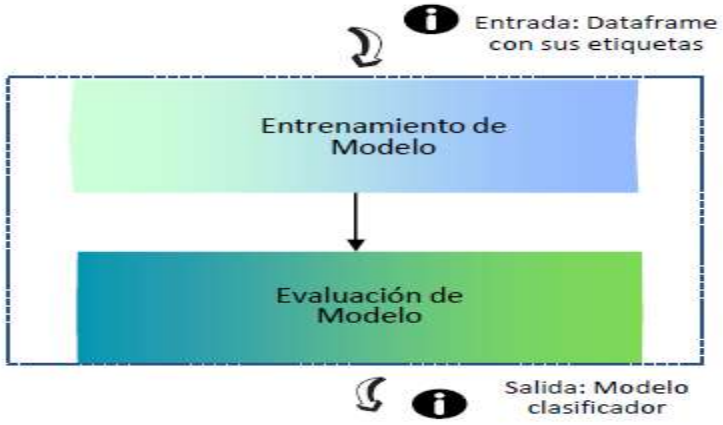
4.7.3.1 Etiquetado de Datos

Para gestionar de forma eficiente la base de datos, se debe convertir a un dataframe mediante pandas librería de Python. Para este trabajo se clasifica los datos en dos categorías: la primera, denominada "fallas", incluyo todas las señales asociadas a cinco eventos específicos; la segunda, etiquetada como "normal", comprende los eventos, que no son fallas. Durante la creación del dataframe, hay que tener presente la cantidad de muestras para un funcionamiento normal y con fallas debe ser igual o similar, así lograremos una distribución equilibrada de muestras en el conjunto de entrenamiento.

4.7.4 Módulo de Entrenamiento del Modelo

La arquitectura del módulo propuesto, representada en la figura 39, se basa en dos procesos principales: entrenamiento y evaluación del modelo.

Figura 39
Arquitectura del módulo de entrenamiento del modelo



4.7.4.1 Entrenamiento del Modelo

Para esta etapa crucial, es esencial tener un grupo de datos de entrenamiento debidamente preparado, donde cada muestra esté correctamente etiquetada. Durante la realización de este estudio, se han identificado dos categorías: "fallas" y "normal". Aunque estas etiquetas podrían permanecer como cadenas de texto dentro del conjunto de datos, se recomienda transformarlas en valores numéricos por razones de practicidad y eficiencia en futuras operaciones. Por lo tanto, se ha decidido asignar el valor entero 1 a la etiqueta "fallas" o evento "negativo", y el valor 0 a la etiqueta "normal" o evento "positivo". Este enfoque garantiza una representación más clara y uniforme de las etiquetas en el set de datos de entrenamiento, lo que simplifica su procesamiento y análisis. Así, el grupo de datos de entrenamiento adopto una estructura coherente y está listo para su posterior análisis.

Una vez que se haya finalizado la preparación del set de datos de entrenamiento, se procedió a entrenar los diferentes modelos utilizando seis algoritmos distintos: el algoritmo de "K vecinos más cercanos" (K-Nearest-Neighbor), el de random forest (RF), el de Support Vector Machine (SMV), el de Gradient Boosting (GB), el de Decision Trees (DT) y otro basado en el algoritmo Naive Bayes (NB). Como

se explicó en la sección 2.11 estos algoritmos tienen sus propios hiperparámetros, que ofrecen la posibilidad de configurar el comportamiento de los algoritmos y, por tanto, optimizar el rendimiento de los modelos (Zhu, Du, Chen, Jin, & Huang, 2019) (Scikit learn, 2007-2024).

4.7.4.2 Evaluación del Modelo

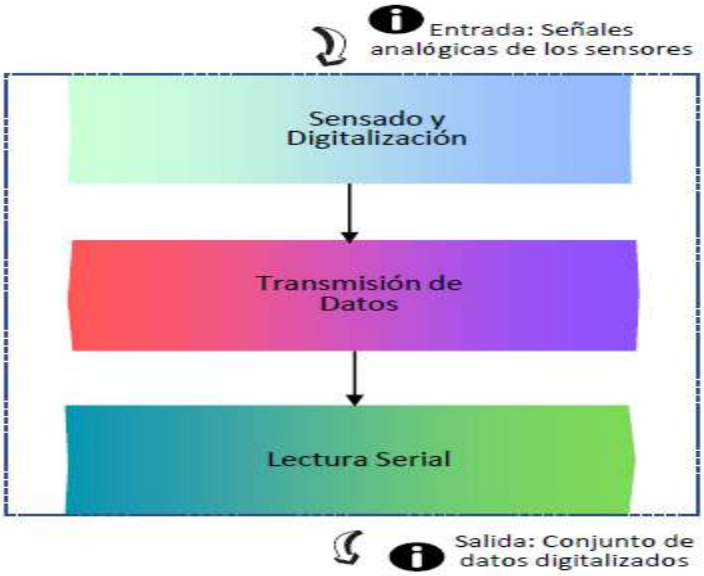
Al finalizar esta etapa del módulo, obtuvo un modelo clasificador que requiere una evaluación de su efectividad. Existen diferentes técnicas para evaluar los algoritmos y para este estudio se utilizará la validación cruzada de K iteraciones. Inicialmente se ha seleccionado un valor de K=10 para esta validación en promedio, un valor comúnmente empleado en la literatura. Además, se realizaron pruebas con diversas combinaciones de valores de hiperparámetros para el entrenamiento, tanto con los modelos SVM, KNN, RF, GB, DT y NB con el propósito de determinar la lista de hiperparámetros que sugiera el modelo de Aprendizaje Automático más eficiente. Al finalizar el entrenamiento, tendremos un clasificador de alto rendimiento, sometido a distintos procesos de evaluación y validación (Salazar Garcia, 2023).

4.7.5 Módulo de Adquisición de Señal para Procesamiento en Tiempo Real

La arquitectura del módulo propuesto, representada en la figura 40, se basa en tres procesos principales: sensado y digitalización, transmisión de datos y lectura serial.

Figura 40

Arquitectura del módulo adquisición de señal para procesamiento en tiempo real



4.7.5.1 Sensado y Digitalización

Este segmento del módulo sigue un proceso idéntico al descrito en la sección 4.7.1.1, empleando el mismo procedimiento y arrojando resultados similares. En esta etapa, se obtuvo las señales digitales provenientes de los sensores, las cuales son adquiridas por el Arduino Nano.

4.7.5.2 Transmisión de Datos

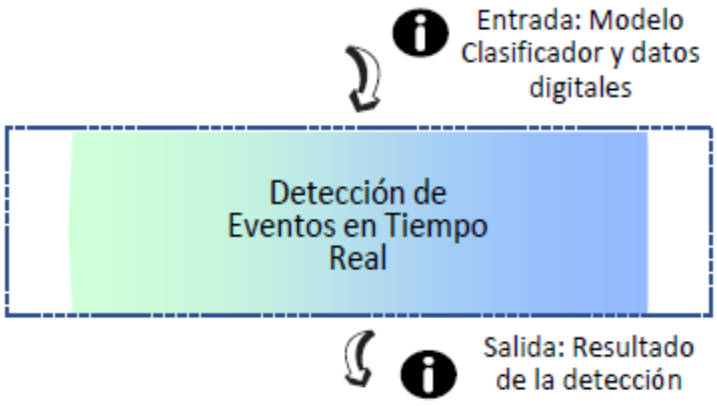
De forma similar en esta etapa del trabajo, se empleó la comunicación serial para establecer una conexión fiable y eficiente al microcontrolador Arduino Nano, encargado de la captación de señales que vienen de los sensores, y la unidad de procesamiento Raspberry Pi, como se detalló en la sección 4.7.1.3.

4.7.5.3 Lectura Serial de Datos

En esta fase, se llevó a cabo la lectura de datos, siguiendo un funcionamiento análogo al presentado en la sección 4.7.2.1. Principalmente, los datos se leen línea por línea desde el puerto serial, para lo cual se configura la conexión con el puerto serial, el cual es detectado automáticamente mediante comandos, estos datos se analizarán posteriormente en el módulo clasificador.

4.7.6 Módulo de Clasificación de Fallas

Figura 41
Arquitectura del módulo de detección



4.7.6.1 Detección de Eventos en Tiempo Real

En esta etapa del módulo, se usó del algoritmo de aprendizaje supervisado seleccionado descrito previamente en la sección 4.7.4.1.

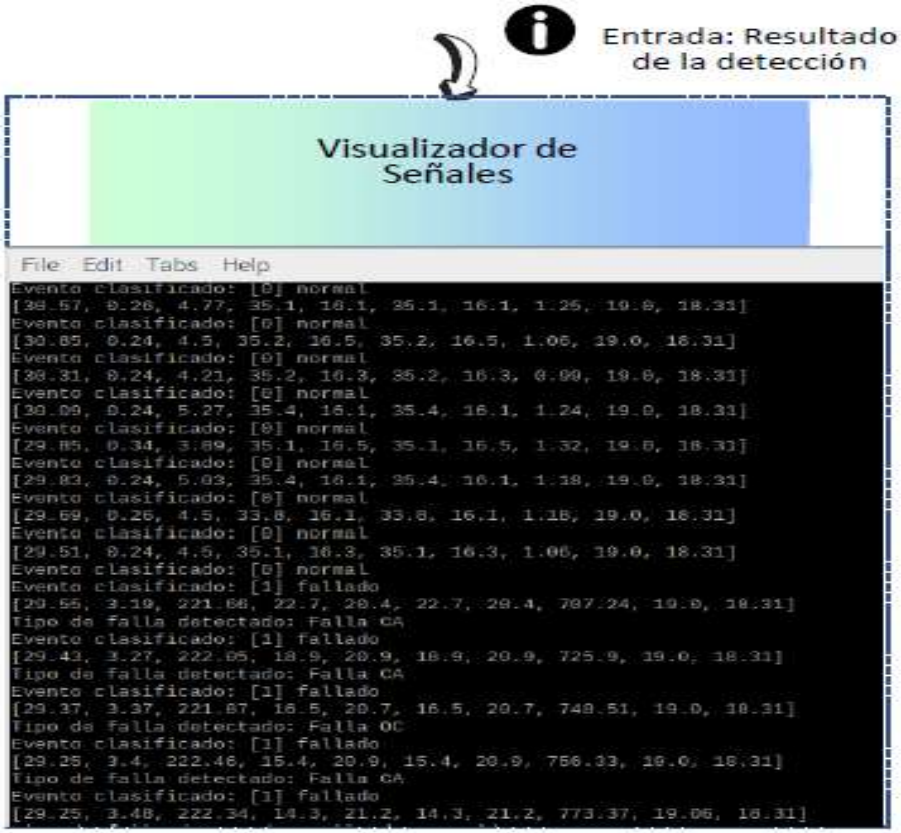
El proceso es simple, se importa el archivo.pkl que contiene los parámetros óptimos y llevar a cabo la detección basada en la lectura serial de datos obtenida por la unidad de adquisición. En este estudio el resultado del detector es una etiqueta binaria: 1 para 'falla' y 0 para 'normal', según la asignación o convención establecida durante el entrenamiento (ver sección 4.7.3.1). Y si es un evento fallado se diagnostica que tipo de falla es, según las reglas de diagnóstico del clasificador.

4.7.7 Módulo de Visualización de Resultados

Una vez obtenida la detección, la fase conclusiva del módulo implica guardar dicho resultado, en la base de datos (para una evaluación posterior de métricas de evaluación). Dicha detección se guarda en su representación numérica (1 o 0) si [1] representa un evento fallado y si [0] representa un evento normal. Y si es un evento fallado muestra que tipo de falla se detecta.

4.7.7.1 Visualizador de Señales

Figura 42
Arquitectura del módulo de visualización de resultados



Capítulo V

Implementación del Sistema de Diagnóstico y Detección de Fallas

En esta etapa el diseño se materializa a través del ensamblaje real de todos los elementos. Los requerimientos del diseño se traducen en software de programación y configuración de hardware. Se instalan sensores para captar señales de distintas naturalezas, como presión, voltaje, corriente y temperatura. También abarca la verificación rigurosa del funcionamiento del sistema a través de pruebas en eventos específicos, permitiendo ajustar y refinar tanto el software como el hardware según lo requerido. Por último, se integran todos los elementos, se realizan los ajustes necesarios y se ponen en operación, logrando que el sistema FDD funcione según lo diseñado, identificando eventos de fallas.

5.1 Descripción de la Unidad de Análisis y Experimentación

Es un sistema de aire acondicionado, de la serie SK 3328.500 COOLING UNIT, un enfriador de expansión directa (mediante refrigerante R134A) de la marca RITTAL, equipado con una válvula de expansión termostática y un compresor de tipo scroll, está diseñado principalmente para refrigerar el aire dentro de los armarios, cuidando así los componentes sensibles a los cambios de temperatura., con una potencia nominal de refrigeración 2Kw/h (6.880 BTU/h) (capacidad de enfriamiento del sistema).

Figura 43
Unidad de análisis, sistema de aire acondicionado

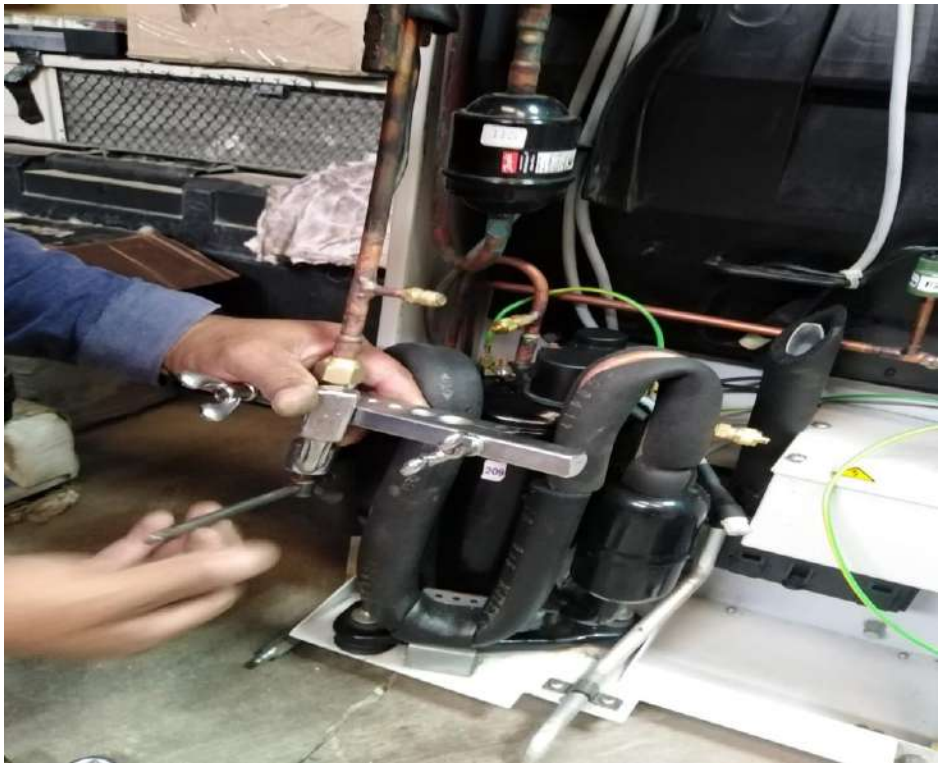
Nota: *Representación de las principales partes de análisis d*el sistema AAP.

5.2 Implementación del Subsistema de Instrumentación

5.2.1 Procedimientos Previos para Implementar el Subsistema de Instrumentación

Después de la etapa de diseño y su posterior elección de los sensores, un procediendo imprescindible, es la ubicación e instalación de las válvulas de servicio, así se evidencia en la figura 44.

Figura 44
Procedimientos previos para la instalación instrumental



Nota: *Se retiro el refrigerante del sistema AAP, previamente, interrumpiendo el circuito de refrigeración*.

También se instaló un mecanismo (válvula de paso), para posteriormente realizar la generación de fallas específicas, objetivos de este trabajo, como se puede observar en el Anexo 38, detalladamente.

Hasta esta etapa se comprende, que ya se instaló todo lo previo necesario en este subsistema de instrumentación. Luego de haber interrumpido en diferentes partes el sistema de refrigeración (siendo este un circuito cerrado o hermético), después se tiene que volver a hermetizar dicho circuito como en su estado inicial. Para realizar procesos estandarizados, como para la instalación inicial de un sistema de

refrigeración (objetivo es cargar refrigerante en medidas optimas, recomendado por el fabricante), Conforme a lo ilustrado en la figura 45, el trabajo de extracción de todo el aire y humedad del circuito de refrigeración mediante una bomba de vacío. Para asegurar un funcionamiento seguro del sistema, y ahorra recién podemos cargar el refrigerante elegido en la etapa de diseño.

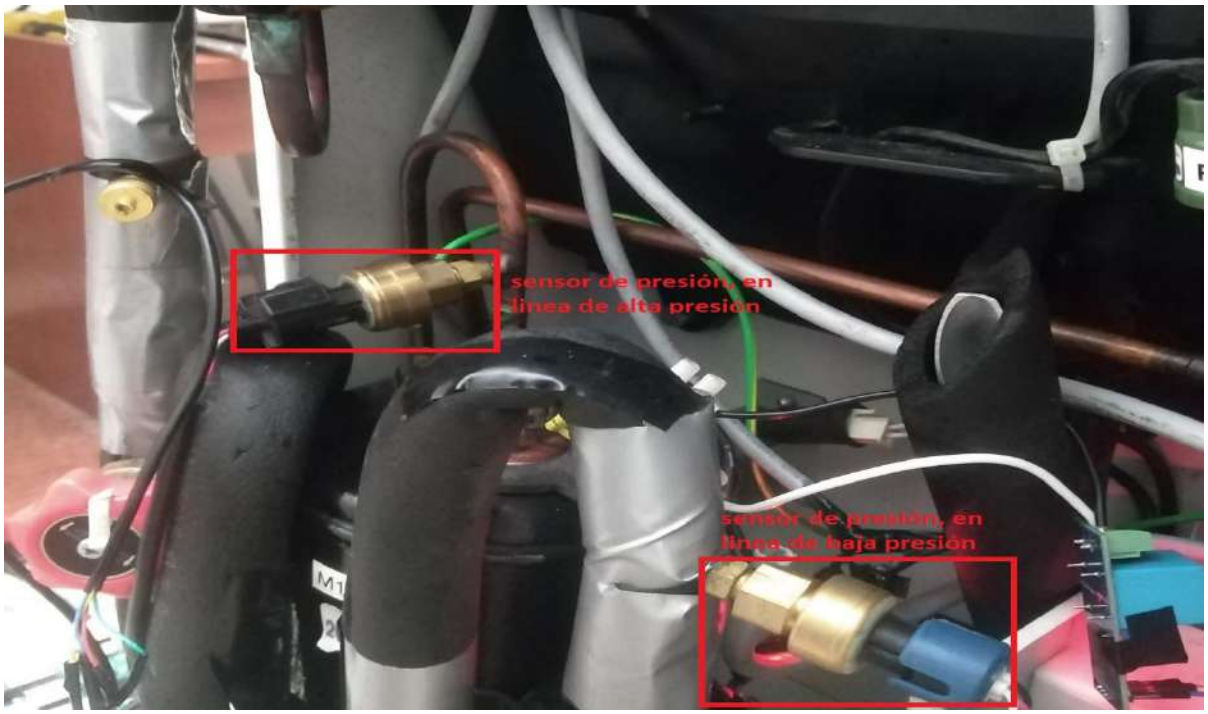
Figura 45
Procedimiento de eliminación de elementos del circuito mediante bomba de vacío

5.2.2 Ubicación e Instalación de los Sensores del Subsistema de Instrumentación

En esta etapa se instaló los sensores de presión, sobre las válvulas de servicio ya ubicadas en el circuito de refrigeración. Como se evidencia en la figura 46.

Figura 46
Ubicación de sensores de presión a la salida del evaporador y condensador

Figura 47
Ubicación de sensores de presión – compresor



Nota: *Se ubico e instalo los sensores de presión en la línea de baja presión y de alta presión.*

También se instalan los otros sensores, ya seleccionados en la etapa de, según se aprecia en la figura 48 donde se esquematizo la ubicación del sensor de temperatura, este sensor de naturaleza infrarrojo, se posiciono a las ¾ partes desde la base del compresor, debido a que a esa altura se encuentra internamente la parte más importante del compresor (rotor y cámara de compresión), además, mediante un instrumento de temperatura se observó que en esta parte se concentra el mayor nivel de temperatura.

Figura 48

Instalación y ubicación del sensor de temperatura en el compresor

Nota: *Se ubico e instalo el sensor de MLX90614, de naturaleza infrarrojo.*

De acuerdo con la figura 49 se observa la instalación del sensor de temperatura, seleccionado previamente en la etapa de diseño, como la opción más óptima para evaluar este sistema.

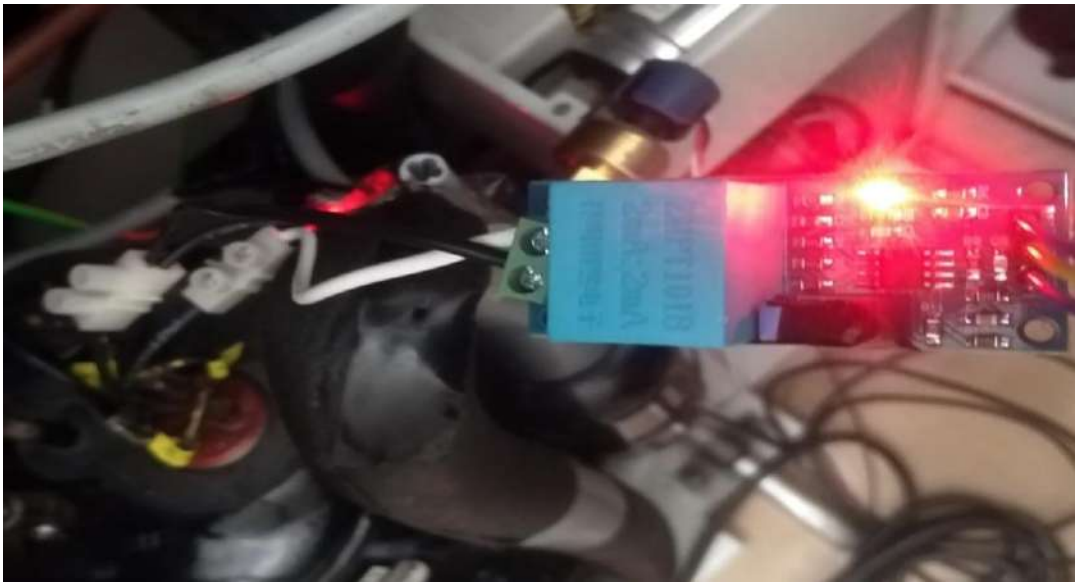
Figura 49

Instalación y ubicación del sensor de temperatura

Nota: *Se ubico e instalo el sensor de temperatura DS18B20 (evaporador y condensador) de principio PTC.*

De acuerdo con la figura 50 siguiente se observa la instalación del sensor de voltaje, seleccionado previamente en la etapa de diseño, como la opción más óptima para evaluar este sistema.

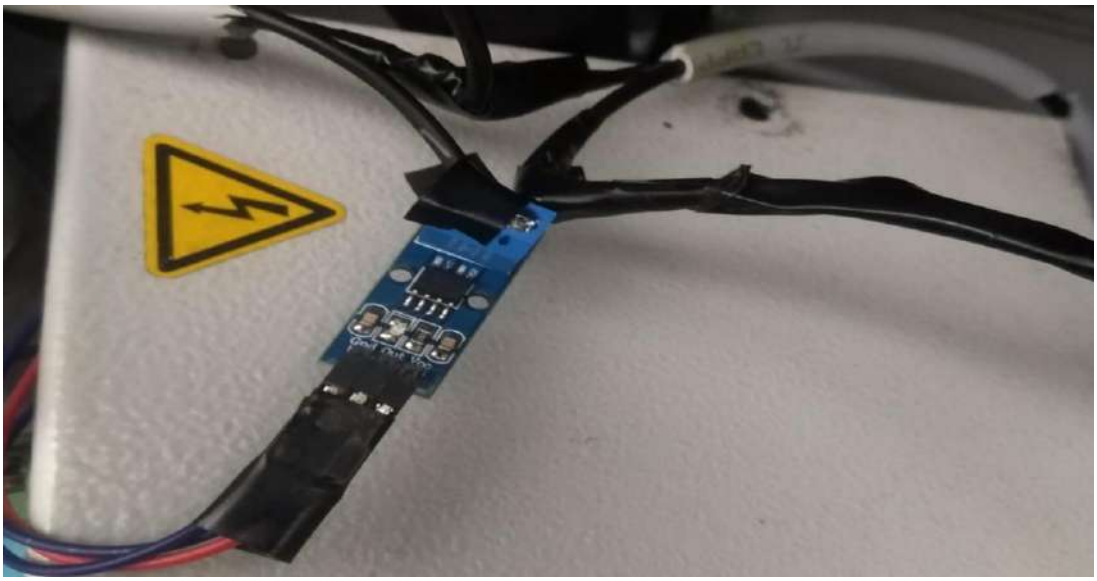
Figura 50
Instalación y ubicación del sensor de voltaje



Nota: *Se ubico e instalo el sensor de voltaje seleccionado, de naturaleza de corriente alterna.*

En la figura 51 se observa la instalación del sensor de corriente, seleccionado previamente en la etapa de diseño, como la opción más óptima para evaluar este sistema.

Figura 51
Instalación y ubicación del sensor de corriente



Nota: *Se ubico e instalo el sensor de corriente ACS712 con capacidad de 20 Amp como máximo.*

5.3 Implementación del Subsistema de Detección

5.3.1 Implementación del Módulo de Adquisición de Señales

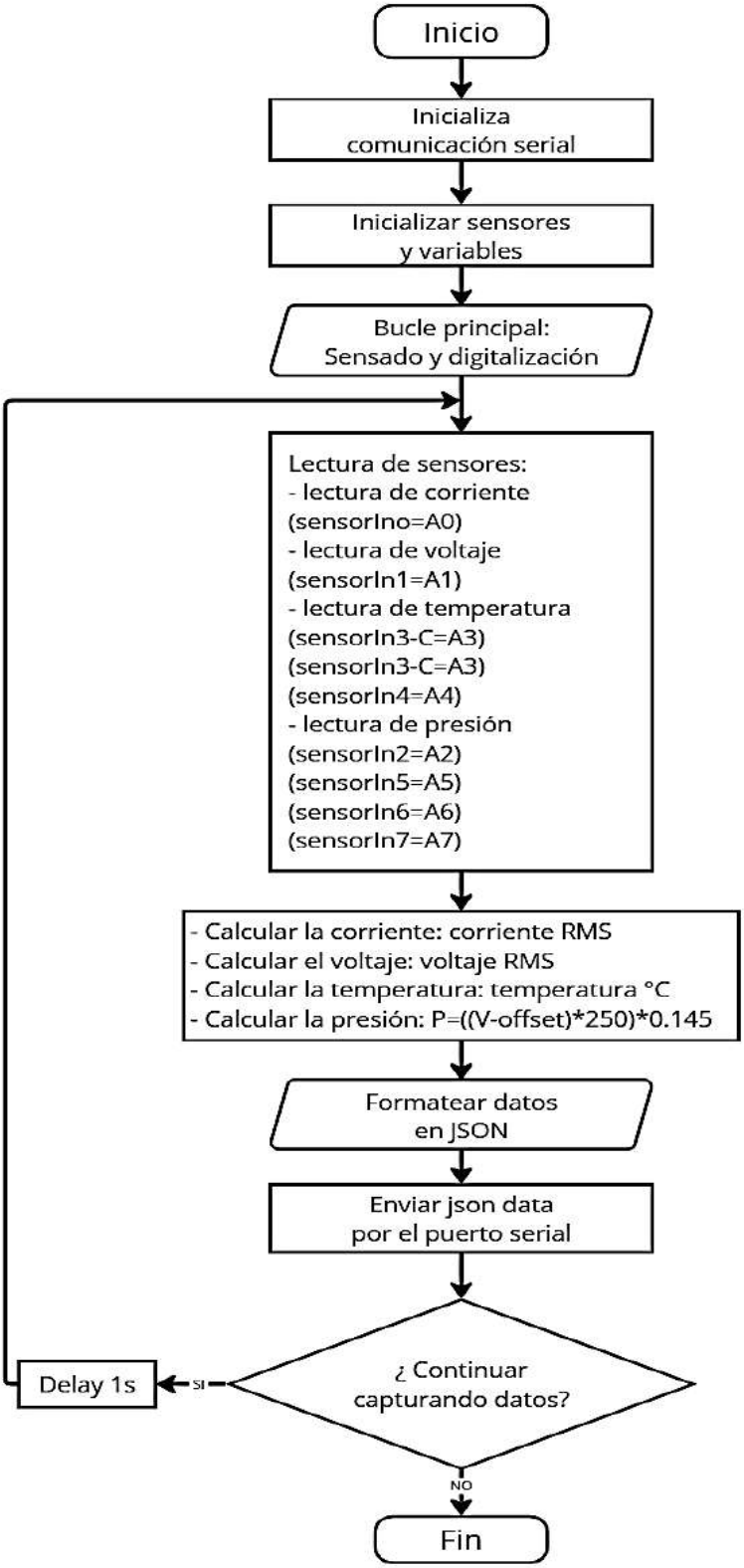
5.3.1.1 Implementación de Etapa: Sensado y Digitalización

Hasta esta etapa ya se tiene todos los sensores ubicados e instalados, sobre el sistema en evaluación, por tanto para esta etapa de la implementación de este subsistema de detección, se utilizó la unidad de adquisición seleccionado en la etapa de diseño, que básicamente es un microcontrolador con características necesarias para poder captar las señales (analógicas) de los sensores ya instalados y dicha unidad seleccionada también tiene la capacidad de convertir las señales analógicas a digitales, porque ya tiene embebido este módulo para un nivel lógico de 5v DC I/O. Con esto también se logra digitalizar todas las señales analizadas.

Por tanto, se implementó un conjunto de instrucciones, que pueden realizar la tarea especifica en este caso de captar las señales en los pines de entrada del microprocesador de forma automática y continua. De la siguiente forma, primero declaramos las variables con el valor de los pines (A0, A1, A2, A3, A4, A5, A6 Y A7), también declaramos constantes (como valores de sensibilidad) proporcionada en las hojas de datos de cada sensor, luego inicializamos la comunicación serial, en la parte principal del código obtenemos el valor de estas variables mediante relaciones recomendadas por los fabricantes, por último, retornamos estos valores en formato json, para convertirlos posteriormente en archivos csv, se puede observar con mayor detalle en el anexo 9 código 1 Sensado y digitalización. Adicionalmente, se esquematiza el flujo de las acciones y decisiones que conforman los algoritmos de cada etapa, estructurando la información de manera clara y secuencial mediante diagramas de flujo, por ejemplo, así como en la figura 52 Diagrama de flujo del módulo de adquisición de señales.

Figura 52

Diagrama de flujo del módulo de adquisición de señales



5.3.1.2 Implementación de Etapa: Calibración Instrumental

En la sección anterior ya se obtuvo el valor de las señales (de temperatura, presión, corriente y voltaje), que se visualice en el monitor serial. Pero no se tiene certeza si estos valores son los correctos, como se mencionó anteriormente sobre la calibración instrumental “representa un proceso esencial en el ámbito de la instrumentación y la medición, indispensable para garantizar la exactitud y la confiabilidad de los valores conseguidos. Este proceso se centra en garantizar que los sensores de medición produzcan lecturas exactas y repetibles de acuerdo con estándares reconocidos y aceptados. Para lograrlo, se seleccionaron cuidadosamente estándares de referencia confiables, como equipos de medición certificados, que sirven como punto de comparación para las lecturas del sensor en cuestión. En caso de detectar desviaciones o errores durante la calibración, se realizan ajustes y correcciones en el sensor para mejorar su precisión” (Gutierrez & Ituralde, 2017).

Para este trabajo se calibro a nivel software (recomendación de los fabricantes) y a nivel hardware mediante la comparación de las mediciones con instrumentos certificados (Termómetro IR que se pueden observar de manera detallada en el Anexo 29, Pinza amperimétrica que se pueden ver de manera detallada en el Anexo 33, Multímetro digital que se pueden visualizar de manera detallada en el Anexo 35, Manómetro analógico de presión que se pueden ver de manera detallada en el Anexo 37)

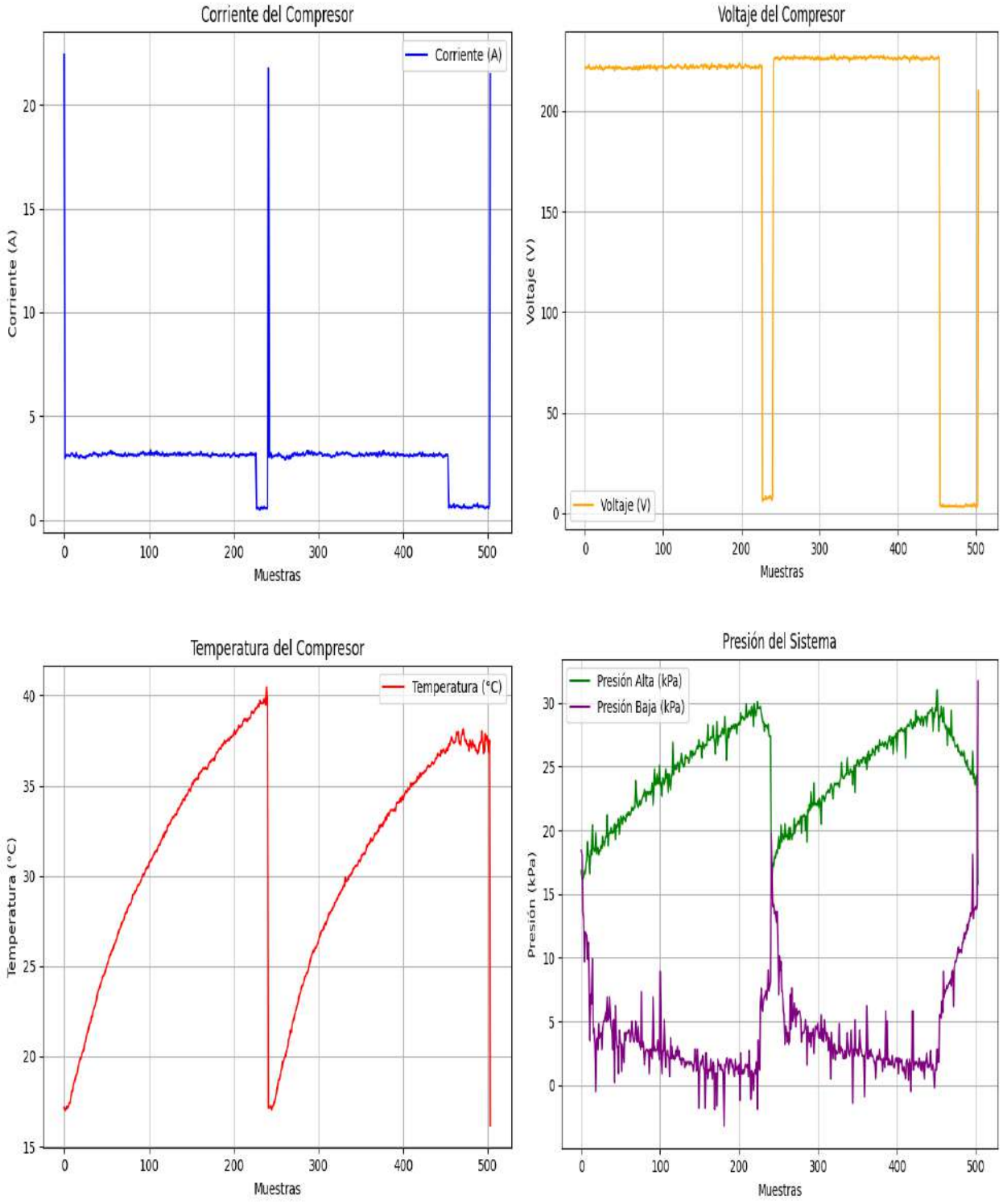
5.3.1.3 Implementación de etapa: Transmisión de Datos

En esta parte se representa él envió de los datos, que ya pasaron por los anteriores procedimientos, según la selección desarrollada en el capítulo IV diseño, en la que el tipo de comunicación más idóneo para este trabajo resultó mediante un conexionado cableado (conexión USB).

La figura 53 muestra las curvas de las variables más importantes, registradas para este estudio, durante dos ciclos de refrigeración, que tiene una duración de 2 a 10 minutos aproximadamente. En este intervalo, se recopilan entre 200 y 500 muestras, para comprender mejor el comportamiento del sistema.

Figura 53

Curva de las variables más importantes del estudio



Nota: *Se representa las curvas de las variables, muestreadas cada segundo.*

5.3.2 Implementación del Módulo Generación de Base de Datos

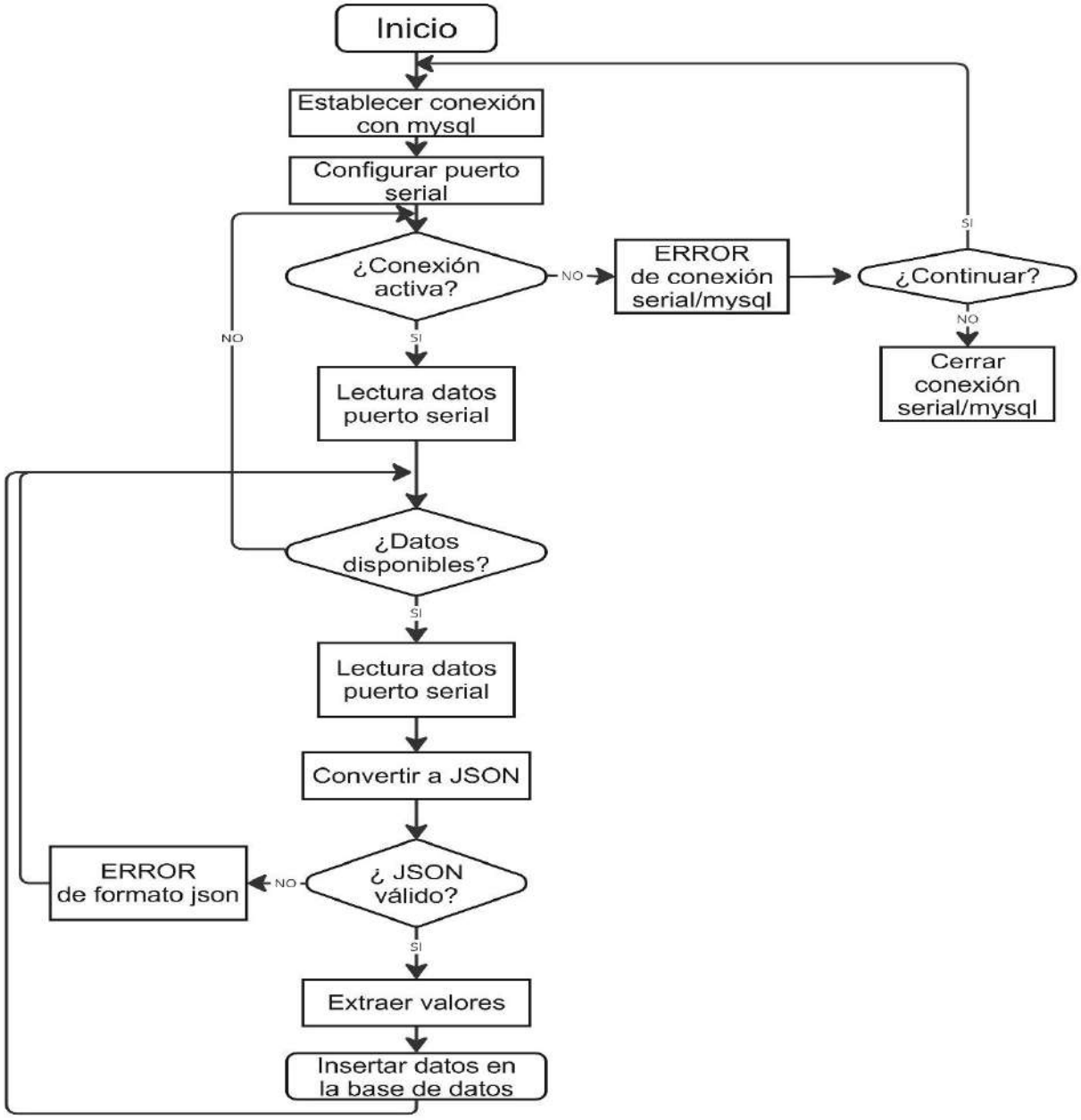
En esta etapa de implementación, entró en funcionamiento la unidad de procesamiento, que en el capítulo de diseño se eligió, que básicamente es un microprocesador, que también se puede entender como un computador embebido del tamaño de un smartphone, con todas las características de un computador, con la ventaja de tener dimensiones como para instalar en cualquier ambiente y la facultad de implementar una base de datos, en este caso particular se opta por MySQL (mariaDB).

Por tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar la tarea especifica de generar un set de datos de forma automática y continua. De la forma siguiente, primero se debe conseguir una conexión con el repositorio de datos, introduciendo datos de usuario, nombre de la base de datos y password, de forma predeterminada. Pero previamente se debe crear este repositorio de datos, la tabla donde se almacenará, el formato de la tabla con detalles de los valores de los datos que serán introducidos mediante líneas de comando del lenguaje SQL, luego se debe establecer el conexionado del puerto serial. En la parte principal del algoritmo, se empieza con la lectura de los datos del puerto serial, luego estos datos los pasamos a formato json, por último, pasamos a introducir en la base de datos, pero también tenemos que especificar el nombre las columnas, que debe ser idéntico a los encabezados establecidos anteriormente. Esta última etapa debe ser repetitiva y continua. Como se puede observar en el anexo 10: Código 2 *Generación de la base de datos en MYSQL, con más detalles.*

Luego se requiere que la información del repositorio de datos almacenados, se extraigan en un archivo CSV. Por lo tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea especifica de forma automática, primero se debe lograr un vínculo con la base de datos introduciendo información de vinculación, de manera predeterminada, luego mediante una consulta query “select*from nameoftable” obtenemos este archivo, mas también es necesario, indicar una ruta para poder guardar este archivo CSV. Por último, se le da los permisos necesarios de lectura y escritura al archivo exportado, para más detalle observar en el anexo 11: Código 3 *Exportar base de datos.csv.*

Figura 54

Diagrama de flujo del módulo de generación de base de datos



5.3.3 Implementación del Módulo de Gestión de Data Frame

En esta etapa, se tiene por objetivo convertir el archivo dataset obtenido anteriormente en un elemento dataframe para el entrenamiento, mediante herramientas como librerías para gestionar el dataframe (pandas). Por lo tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea especifica de forma automática. De la siguiente forma, primero se debe

importar estas librerías, luego se define una función para cargar los archivos desde el CSV, se empieza con la lectura a la ruta donde se encuentra el archivo y luego se transforma en un dataframe, se agrega una columna para etiquetar, llamada “Clase”. Por último, se retorna este archivo dataframe listo para el entrenamiento, como se puede observar en el Anexo 12: Código 4 *Gestión de data frame.*

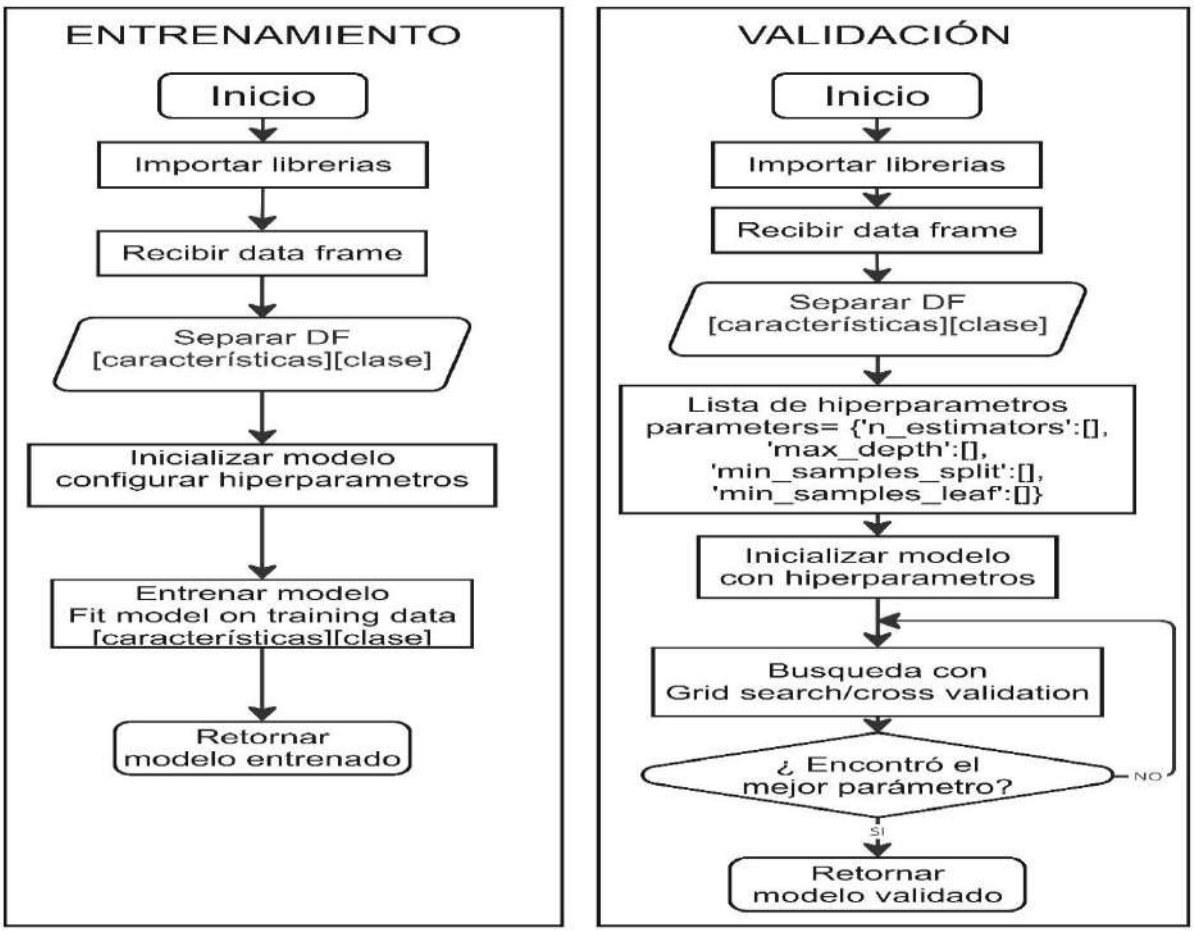
5.3.4 Implementación del Módulo de Entrenamiento del Modelo

5.3.4.1 Implementación de Etapa: Entrenamiento del Modelo

Hasta esta etapa ya se tiene el archivo dataframe para el entrenamiento, que este filtrado, etiquetado y procesado. Se tiene por objetivo adiestrar el modelo SVM, desde un training dataframe (procesado con pandas), cuyos parámetros de ajuste son los hiperparámetros de entrenamiento. Por lo tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea especifica de forma automática. De la siguiente forma, primero se debe importar la librería sklearn, luego se define una función para el entrenamiento del modelo con SVM, se empieza con la declaración de variable el primero para un dataframe solo para vectores de las características y otro un dataframe solo para la columna clase y luego mediante funciones como “svm.SVC” de sklearn, se ejecuta el entrenamiento del modelo. Por último, se retorna un archivo de tipo SVC que representa un modelo SVM, como se puede observar en el Anexo 13: Código 5 Entrenamiento con support vector machine SVM.

Como se explicó en el párrafo anterior, el proceso del entrenamiento de un modelo, específicamente del modelo SVM, este mismo proceso se repite para los otros cinco modelos más, K-Neighbors, Random Forest, Gradient Boosting, Decision Tree y Naive Bayes, de forma similar, con sus hiperparámetros y funciones particulares de cada modelo evaluado, que además el código de entrenamiento de cada modelo, está representado en la sección de anexos 14, 15, 16, 17, 18.

Figura 55
Diagrama de flujo del módulo de entrenamiento del modelo



5.3.4.2 Implementación de Etapa: Evaluación del Modelo

En esta etapa se buscó comparar el rendimiento de diferentes modelos entrenados como el SVM, evaluando su desempeño para diferentes configuraciones de hiperparámetros utilizando validación cruzada, para buscar el modelo más sobresaliente entrenado con el algoritmo SVM. Por lo tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea específica de forma automática. De la siguiente forma, primero se debe importar la librería sklearn, luego se define una función para la calificación de los modelos entrenados con el algoritmo SVM, se empieza con la declaración de variable el primero para un dataframe solo para vectores de las características y otro dataframe solo para la columna clase, enlistamos valores de hiperparámetros para luego mediante la función como “GridSearchCV” de sklearn, se realiza un barrido completo con los valores de la lista de hiperparámetros. Por último, se devuelve el resultado del ordenamiento de los hiperparámetros en

función de su rendimiento, determinado a través de múltiples iteraciones de validación cruzada, como se puede visualizar en el Anexo 19: Código *Validación del modelo SVM, mediante lista de hiper parámetros.*

Como se explicó en el párrafo anterior, el proceso de evaluación de un modelo, específicamente del modelo SVM, este mismo proceso se repite para los otros cinco modelos más, K-Neighbors, Random Forest, Gradient Boosting, Decision Tree y Naive Bayes, de forma similar, con su lista de hiperparámetros particulares de cada modelo evaluado, que además el código de entrenamiento de cada modelo, está representado en la sección de anexos 20, 21, 22, 23, 24.

5.3.5 Implementación del Módulo de Acondicionamiento de Señal para Procesamiento en Tiempo Real

Esta etapa se puede considerar, como una prueba en tiempo real, para su desarrollo usa módulos y etapas similares, los que se diseñaron y desarrollaron anteriormente.

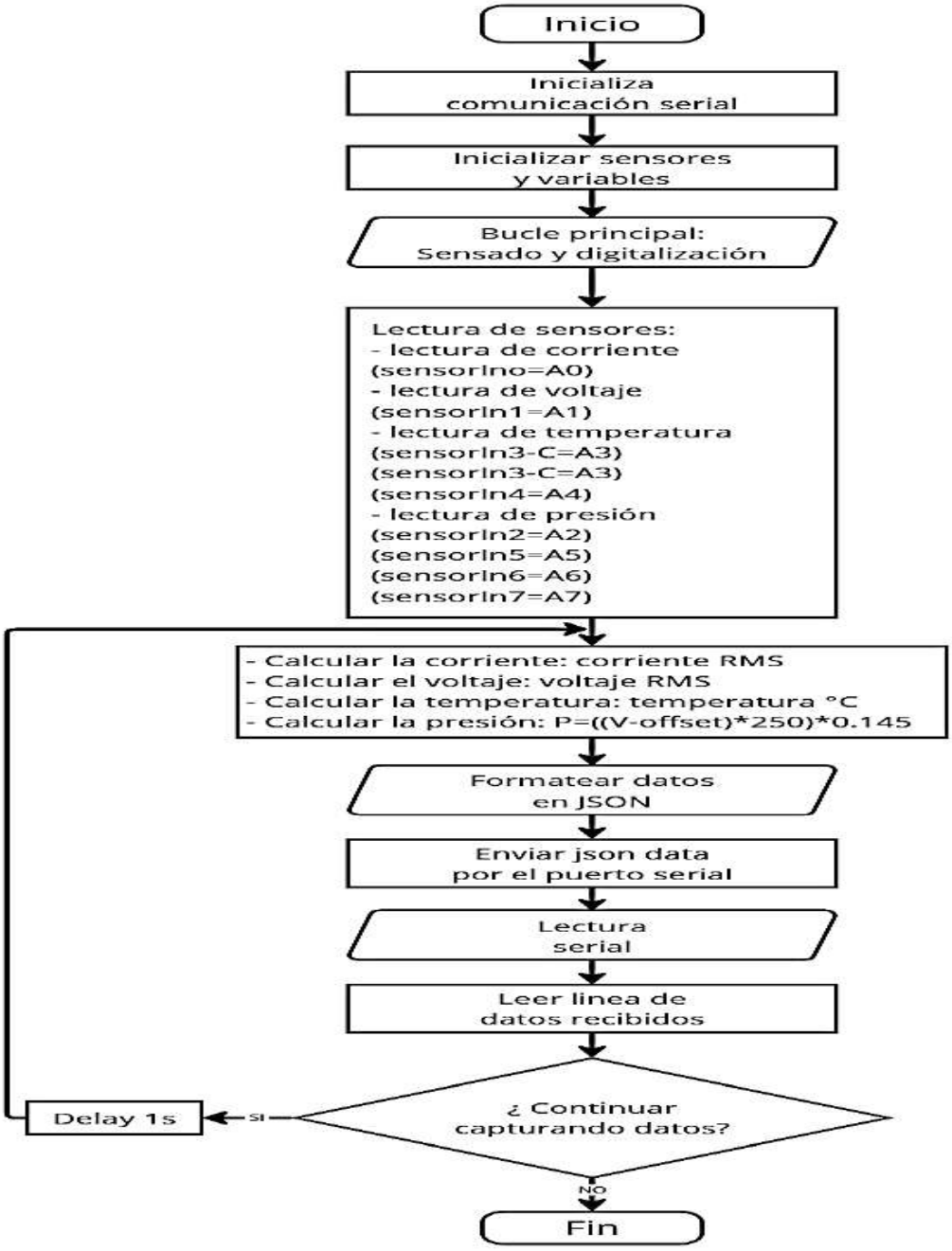
5.3.5.1. Implementación de Etapa: Sensado y Digitalización, esta etapa, es idéntica a lo desarrollado en la parte de implementación del módulo de adquisición, en la sección 5.3.1.1. Tanto a nivel físico como a nivel software.

5.3.5.2. Implementación de Etapa: Transmisión de Datos, esta etapa, es similar a lo desarrollado en la parte de implementación del módulo de adquisición, en la sección 5.3.1.3. También se decide usar comunicación física (USB Universal Serial Bus).

5.3.5.3. Implementación de Etapa: Vector de Características, esta parte se puede considerar como una sub etapa, de estos módulos desarrollados, cuya finalidad es hallar el vector de características, mediante una lectura serial de los datos. Por lo tanto, también se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea específica de forma automática. De la siguiente forma, primero define una función para la obtención del vector de características desde la lectura serial, se empieza con la lectura de datos desde la unidad de adquisición (microcontrolador), luego se decodifica los datos y elimina características adicionales y por último convertir la cadena de datos en una lista (float), como se puede observar en el anexo 25.

Figura 56

Diagrama de flujo para adquirir la señal para su procesamiento en tiempo real



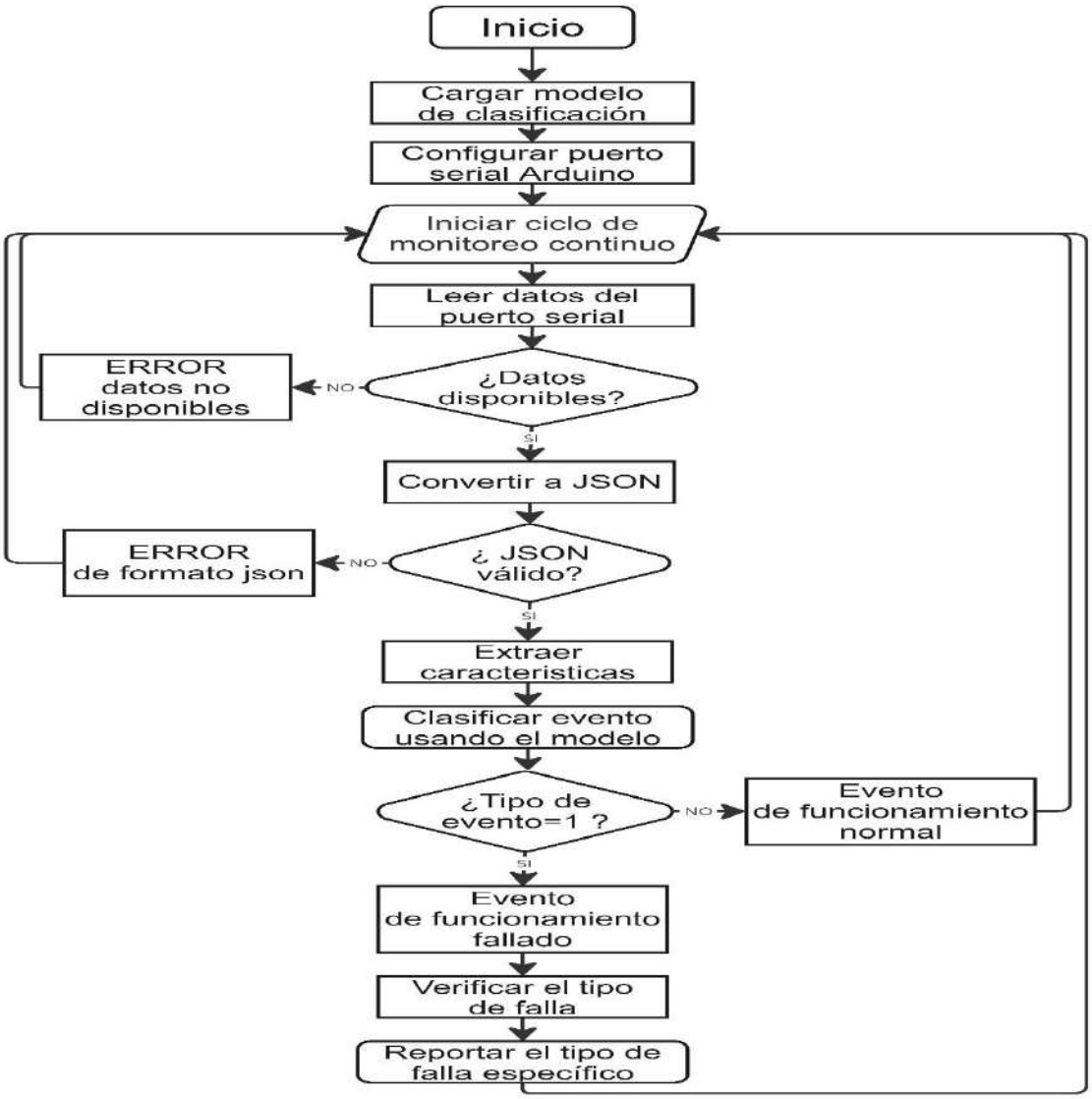
5.3.6 Implementación del Módulo de Clasificación de Eventos

Esta es una de las etapas de la parte final del sistema desarrollado, hasta esta etapa ya se tiene el modelo con mejor rendimiento y un vector de características del procesamiento en tiempo real desarrolladas en la etapa anterior. Se tuvo por objetivo clasificar el evento de falla. Por lo tanto, también

se requiere implementar un conjunto de instrucciones, que puedan realizar esta tarea especifica de forma automática. De la siguiente forma, primero se debe importar la librería sklearn, luego se define una función para cargar el modelo, esta requiere la ruta de archivo donde se encuentra el modelo, también se tiene que definir la función clasificación de eventos, para esta función necesitamos el modelo y el vector de características, por último, también se debe definir la función donde se obtenga el vector de características y clasifique el evento, como se desarrolló en la sección 5.3.5.3. Implementación de etapa: Vector de Características, como se puede observar en el anexo 26: Código 18 *Detección de eventos.*

Figura 57

Diagrama de flujo del módulo de detección



Y para diagnosticar luego de detectar que es un evento fallado, se usó las reglas de diagnóstico del clasificador, esto requiere de manera adicional una etapa de código condicional, que se observa en el anexo 26, Es necesario desarrollar una función adicional, con la finalidad de exportar e importar los modelos de ML de python en formato (*.pkl), para esto se define la ruta donde se almacena el modelo, además, se especifica los permisos de sobre estos modelos (lectura y escritura) y por último retorna el modelo, como en el anexo 27: Código 19 *Exportación de modelos.pkl, con más detalles*.

5.3.7 Implementación del Módulo de Visualización del Resultado

Es una parte esencial del proyecto. Después de desarrollar las diversas fases del sistema, se requiere una pantalla (monitor) conectada mediante un cable HDMI al sistema de detección, que consiste en un Arduino Nano y una Raspberry Pi. Este enlace facilita la visualización al tiempo de la respuesta serial generada por el sistema de detección y en el caso de un evento de falla se diagnostica que tipo falla es. Con la integración del monitor, se pudo observar y analizar las señales de estado procesadas por el sistema, que detectan eventos como normales o fallados, basándose en parámetros como presión, temperatura, corriente y voltaje. Este módulo de visualización no solo facilita el monitoreo inmediato de la operación del sistema AAP, sino que también ayudó a validar la eficacia del modelo de inteligencia artificial desarrollado, demostrando el performance del sistema en eventos reales y diferidas. Esta habilidad de visualización es crucial para la operación eficiente del sistema y proporciona una herramienta poderosa para futuras mejoras y evaluaciones en el área de la clasificación de fallas.

5.4 Guía de Adquisición de la Base de Datos

5.4.1 Configuración de la Unidad de Análisis

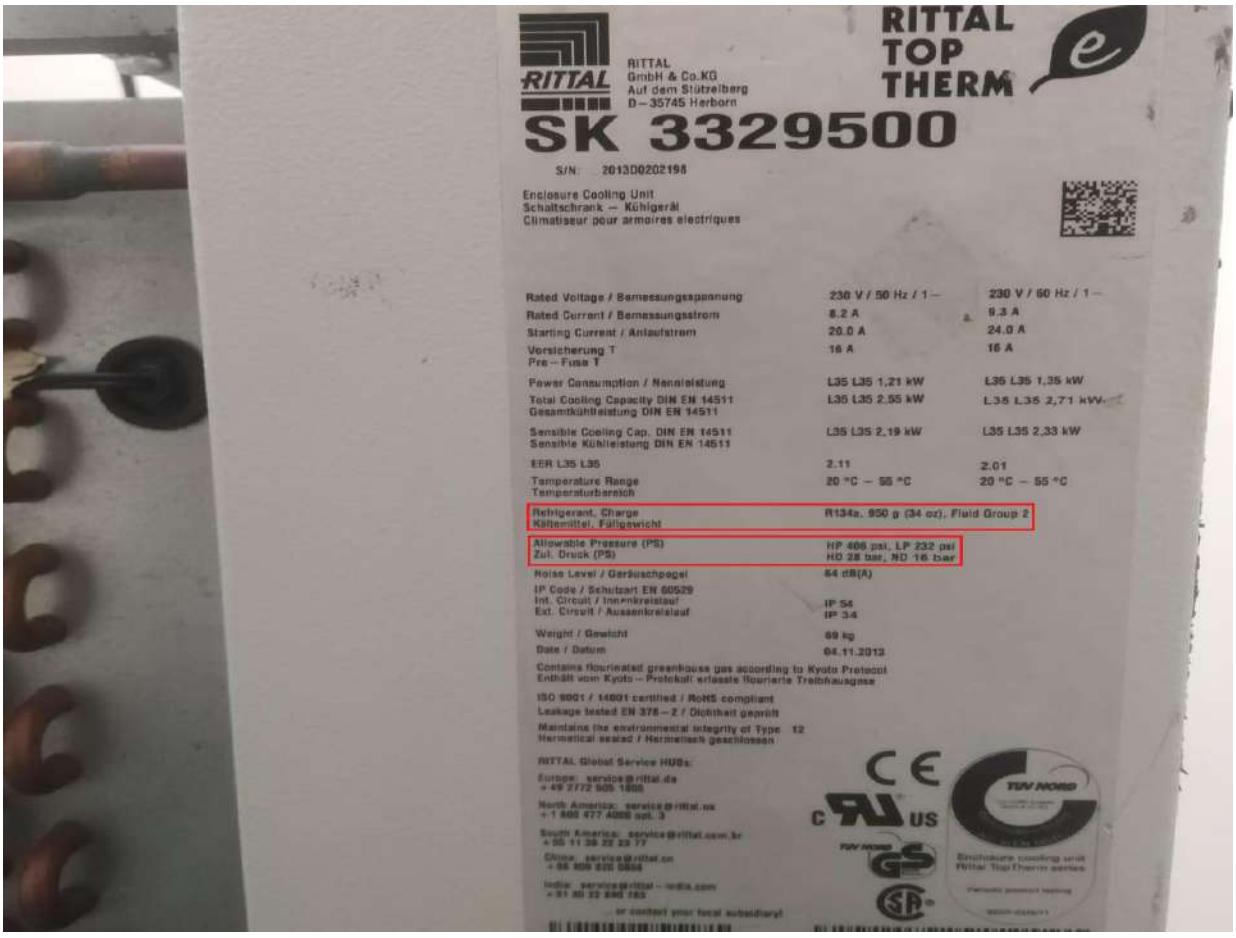
La sobrecarga de refrigerante suele ocurrir debido a un mantenimiento inadecuado del equipo, se busca que el lado de baja presión sea lo más bajo posible en estos sistemas; sin embargo, una cantidad excesiva de refrigerante incrementará la presión y la temperatura operativa del sistema, disminuyendo

así su capacidad de enfriamiento. Es posible que el sistema aún pueda cambiar de estado, pero a una temperatura muy alta de la prevista, no se enfriaría tan eficientemente como con la carga adecuada.

El proceso de generación controlada de fallas, de este caso, se desarrolla de esta forma, en la unidad de experimentación, se agrega el refrigerante R134a, a un nivel superior, al recomendado por el fabricante para su funcionamiento normal como se evidencia en la figura 58, mediante un kit de manómetros donde se observa el incremento de presión, específicamente se incrementa el nivel de refrigerante a 130% o 1235 g de refrigerante, luego de unos minutos, se estabiliza el nivel de presión tanto en el de alta y baja presión (Zhu, Du, Chen, Jin, & Huang, 2019).

Figura 58

Características del sistema AAP RITTAL SK 3329500 – Enclosure Cooling Unit



Nota: *Se representa específicamente valores para un funcionamiento normal, 950 g - R134A - RITTAL.*

Figura 59

Evento refrigerant overcharge (OC)

Nota: *Se representa el sobre nivel de presión, mediante relojes manómetro.*

La subcarga de refrigerante generalmente se produce debido a una fuga o un mantenimiento incorrecto del equipo. En estos sistemas, es crucial que el lado de baja presión mantenga una presión determinada para un rendimiento óptimo; no obstante, una cantidad insuficiente de refrigerante disminuirá la presión y la temperatura operativa del sistema, reduciendo conciderablemente su capacidad de enfriamiento. Aunque el sistema puede continuar operando, lo hará a una temperatura inferior de lo previsto, lo cual reducirá su eficiencia de enfriamiento (Wang, 2021).

Para generar de manera controlada una falla por subcarga, se desarrolla de manera similar con la sobrecarga, como en la figura 59, específicamente se reduce la cantidad de refrigerante R134a a un nivel inferior al recomendado por el fabricante figura 58, alcanzando solo el 65% de la carga normal o 618 g de refrigerante aproximadamente. Mediante un conjunto de manómetros, se observa la disminución de presión, que se estabiliza tras unos minutos a la vez en la parte de hight como en el de low presión, permitiendo evaluar la magnitud de la subcarga en la eficiencia del sistema.

Figura 60

Evento refrigerant undercharge (UC)

Nota: *Se representa el sub del nivel de presión, mediante relojes manómetro específicos para R134a.*

En el caso de la falla restricción de la línea de líquido, factores cómo dobladuras o filtro/secador sucio en la línea de líquido, las formas de influencia de la falla, aceleran la acumulación de depósitos corrosivos y microorganismos que pueden obstruir las tuberías y provocar la perforación de equipos debido a la corrosión.

Para generar de manera controlada una falla restricción de línea de líquido del circuito de aire acondicionado, se implementa mediante el uso de una válvula para imponer la pérdida de presión.

Figura 61

Evento liquid-line restriction (RL)

Para el caso de la falla reducción de flujo de aire del condensador (CA), puede deberse al ensuciamiento del serpentín o flujo de aire insuficiente en el condensador y problemas de diseño del ventilador o del sistema de distribución, es equivalente a operar con un condensador de menor capacidad y conduce a temperaturas y presiones de condensación más altas que en condiciones normales.

Para generar de manera controlada una falla (condenser airflow reduction), en el sistema AAP, se desarrolla bloqueando una parte de la superficie con malla plástica para aumentar la resistencia al flujo.

Figura 62

Evento condenser airflow reduction (CA)

Y para el caso de la falla reducción de flujo de aire del evaporador (EA), se desarrolla de manera similar a la falla reducción de flujo de aire del condensador.

Para generar de manera controlada una falla (evaporador airflow reduction), en el sistema AAP, también se desarrolla bloqueando la superficie del filtro de salida del evaporador, específicamente en este estudio se realizan usando malla rachel, para incrementar la resistencia al flujo del aire.

Figura 63

Evento evaporator airflow reduction (EA)

Tabla 12

Resumen de fallas y descripción

N°	Tipos de fallas	Descripción
1	Subcarga de refrigerante (UC)	El nivel de la carga de refrigerante tiene menor nivel al recomendado (65% - 618 g).
2	Sobrecarga de refrigerante (OC)	El nivel de la carga de refrigerante tiene mayor nivel al recomendado (130% - 1235 g).
3	Restricción de la línea de líquido (RL)	Se implementa mediante una válvula reguladora (ubicada por la entrada del compresor) para imponer la pérdida de presión.
4	Reducción de flujo de aire del condensador (CA)	Se implementa bloqueando una parte de la superficie (intercambiador) del condensador con una malla, para reducir el flujo de aire en el lado del condensador.
5	Reducción de flujo de aire del evaporador (EA)	Se implementa bloqueando una parte de la superficie del filtro de salida del evaporador con una malla, para reducir el flujo de aire en el lado del evaporador.

5.4.2 Software Utilizado para la Implementación de Base de Datos

Además, se emplearon varios programas de código abierto para la generación de la base de datos, su almacenamiento, y otros propósitos relacionados con el análisis y procesamiento de datos.

Después de acondicionar los módulos de hardware, se procedió con la implementación de los pseudocódigos y la configuración inicial del hardware Raspberry Pi 4B 4GB. Este procedimiento abarca la instalación del SO, así como la configuración de las bibliotecas. Para ello, se comienzo descargando el archivo necesario, que en este caso es la versión Debian. GNU/Linux 11 (Bullseye) aarch64. Es importante tener un conocimiento del sistema operativo GNU/Linux, dado que el proceso implica el uso de comandos en la terminal para instalar los paquetes requeridos y configurar el sistema, se utilizó Balena Etcher para grabar la imagen del SO, en una tarjeta de memoria SD de 32 GB o superior. Después de grabar la imagen, se arranca el sistema operativo en el Raspberry Pi utilizando la tarjeta SD. Es recomendable crear un usuario "root" y asignarle una contraseña. Una vez que el SO ha iniciado, es necesario actualizar el sistema ejecutando los comandos "sudo apt update" y "sudo apt upgrade". Una vez finalizada la actualización, se procede a reiniciar el dispositivo.

Con esta configuración, el Raspberry Pi 4B 4GB está listo, aunque, solo se ha instalado el sistema operativo básico y se han llevado a cabo los seteos iniciales.

Figura 64
Sistema operativo Linux Debian

Para la siguiente etapa es importante, la instalación del software para el manejo de la base de datos, Mariadb (MySQL) elegido como la mejor opción en el capítulo IV Diseño; se empieza con la actualización del instalador de aplicativos, mediante comando sudo apt, como se muestra en el anexo 39.

Luego, se procede con la instalación mediante comando, sudo apt install mariabd-server, según se aprecia en el anexo 40 y 41.

También es indispensable un editor de código (IDE entorno de desarrollo integrado), para poder implementar los pseudocódigos, se procede con la instalación de Geany: Geany ya se encuentra preinstalado en la versión 1.37.1, conforme a lo representado en la figura 65.

Figura 65

Entorno de desarrollo integrado Geany

5.5 Base de Datos

Esta parte se enfoca en la inserción de diversas variables (temperatura, corriente, voltaje, presión) en el repositorio de datos, tal como se detalló en la sección de adquisición de datos en el capítulo V.

La base de datos se generó, aproximadamente en un periodo de tiempo de 3 meses, primero registramos para un funcionamiento normal, que se debe entender como un ciclo de funcionamiento de refrigeración, donde cada parte del aire acondicionado, como el compresor, los ventiladores de la etapa

del condensador, del evaporador y otros elementos, entran en funcionamiento, bajo una secuencia característica. En el transcurso del día se induce al funcionamiento, elevando la temperatura de sensor del sistema de forma manual (este proceso representa, un caso hipotético de sobre calentamiento de los equipos de ese ambiente), aproximadamente unas 3 a 4 veces al día, el primero a las 9:00 am, luego a las 13:00 pm, luego a las 17:00 pm y por último a las 21:00 pm, cabe aclarar que el sistema está funcionando todo el día. De forma similar se procede con la captura de datos para los eventos de falla estudiados, como se explica en la parte 5.4 Guía de adquisición de la base de datos, de este capítulo.

5.5.1 Preprocesamiento de la Información de la Base de Datos

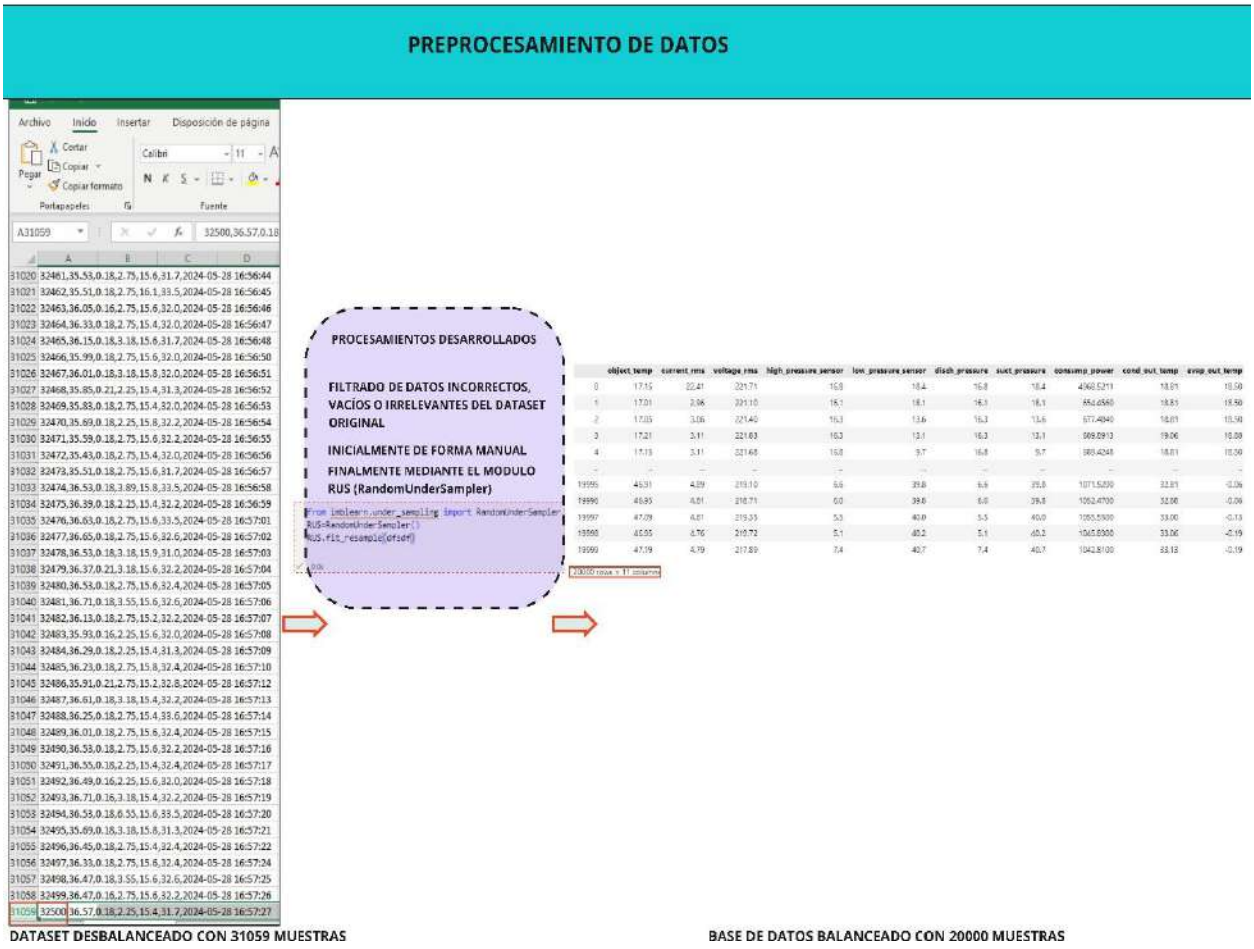
Este preprocesamiento es esencial para el repositorio de datos generado. Si nuestro archivo de base de datos en formato CSV contiene inicialmente más de 31,000 muestras. La depuración de datos incorrectos, vacíos, inconsistentes o irrelevantes es fundamental para evitar que el modelo aprenda patrones inexactos o sesgados, lo cual podría reducir significativamente su efectividad y precisión.

El problema de los conjuntos de datos sesgados o desbalanceado se puede abordar en términos de nivel de datos, ajustando la distribución mediante técnicas de muestreo. Los procedimientos más conocidos son el sobremuestreo y el submuestreo.

En primer lugar, el sobremuestreo se utiliza para incrementar el número de ejemplos del tipo minoritario. En contraste, los métodos de submuestreo se emplean para disminuir el número de situaciones en el tipo mayoritario.

En este estudio se submuestrea los datos del tipo mayoritario para equilibrar el conjunto de datos gradualmente. Primero de forma manual, las muestra que están lejos del ciclo de refrigeración (en función del tiempo) y finalmente usando el módulo RUS “RandomUnderSampler” de la librería “imblearn.under_sampling”, mediante “RUS.fit_resample()”, permitiendo así que el sistema detecte fallas de manera más confiable y exacta.

Figura 66
Etapas de preprocesamiento de la base de datos



Nota: *Se representa la preparación del repositorio de datos, para posterior uso en el entrenamiento.*

5.5.2 Etiquetado de Datos

El proceso de etiquetado de datos se basa en la generación controlada de condiciones de falla inducidas experimentalmente (sección 5.4.1 y Tabla 12). En una primera etapa, se recopilaron 31,059 muestras, las cuales fueron sometidas a un proceso de preprocesamiento y balanceo, reduciendo el conjunto final a 20,000 registros. De estos, las primeras 10,000 representan condiciones normales de operación del sistema, mientras que los otros 10,000 se distribuyen en bloques de 2,000 muestras, cada uno correspondiente a una de las cinco condiciones de falla: Sobrecarga de refrigerante (OC), Restricción de línea de líquido (RL), Subcarga de refrigerante (UC), Reducción del flujo de aire del condensador (CA) y Reducción del flujo de aire del evaporador (EA). Para cada caso, se ejecutaron múltiples ciclos de operación hasta alcanzar estabilidad térmica y funcional, garantizando que las mediciones de presión, corriente, voltaje y temperatura reflejaran fielmente el estado inducido. El etiquetado se realizó de

manera semiautomatizada mediante scripts en Python, los cuales vincularon los timestamps de los registros con los logs de las intervenciones experimentales, asegurando una correspondencia exacta entre cada muestra y la condición operativa simulada. Como resultado, se obtuvo una base de datos de 20,000 muestras, distribuidas entre las seis condiciones y caracterizadas por 10 variables, constituye el fundamento para el entrenamiento, validación y prueba de los algoritmos de clasificación de este estudio.

Figura 67

Dataframe post - filtrado

	object_temp	current_rms	voltage_rms	high_pressure_sensor	low_pressure_sensor	disch_pressure	suct_pressure	consump_power	cond_out_temp	evap_out_temp
0	17.15	22.41	221.71	16.8	18.4	16.8	18.4	4968.5211	18.81	18.50
1	17.01	2.96	221.10	16.1	18.1	16.1	18.1	654.4560	18.81	18.50
2	17.05	3.06	221.40	16.3	13.6	16.3	13.6	677.4840	18.81	18.50
3	17.21	3.11	221.83	16.3	13.1	16.3	13.1	689.8913	19.06	18.88
4	17.15	3.11	221.68	16.8	9.7	16.8	9.7	689.4248	18.81	18.50
...	...	...	...	...	...	...	...	...	...	...
19995	46.91	4.89	219.10	6.6	39.8	6.6	39.8	1071.5200	32.81	-0.06
19996	46.95	4.81	218.71	6.0	39.8	6.0	39.8	1052.4700	32.88	-0.06
19997	47.09	4.81	219.35	5.5	40.0	5.5	40.0	1055.5500	33.00	-0.13
19998	46.95	4.76	219.72	5.1	40.2	5.1	40.2	1045.8300	33.06	-0.19
19999	47.19	4.79	217.89	7.4	40.7	7.4	40.7	1042.8100	33.13	-0.19

Nota: *Se representa al dataframe procesado, en la que contiene 20000 muestras para 7 columnas.*

Mediante una función de la librería pandas, “concat”, se puede unir dataframes, en este caso de forma horizontal, que está definida con el parámetro axis=1, como se define en la figura 68.

Figura 68

Función concat de pandas, para unir dataframes.

```
dataframe_sdf=pd.concat([dfsdf1,dfclass], axis=1)
```

Si se observa la figura 69, se nota que, el dataframe incluye una columna denominada “clase”, la cual representa el etiquetado de nuestra base de datos. Es importante mencionar que los modelos de aprendizaje automático comprenden mejor los valores numéricos que los datos en formato de cadena de texto (strings). Por esta razón, es más práctico utilizar “1” para representar la etiqueta “falla” y “0” para la etiqueta “normal”.

Figura 69

Dataframe post - etiquetado total

	object_temp	current_rms	voltage_rms	high_pressure_sensor	low_pressure_sensor	disch_pressure	suct_pressure	consump_power	cond_out_temp	evap_out_temp	clase
0	17.15	22.41	221.71	16.8	18.4	16.8	18.4	4968.5211	18.81	18.50	0
1	17.01	2.96	221.10	16.1	18.1	16.1	18.1	654.4560	18.81	18.50	0
2	17.05	3.06	221.40	16.3	13.6	16.3	13.6	677.4840	18.81	18.50	0
3	17.21	3.11	221.83	16.3	13.1	16.3	13.1	689.8913	19.06	18.88	0
4	17.15	3.11	221.68	16.8	9.7	16.8	9.7	689.4248	18.81	18.50	0
...	...	...	...	...	...	...	...	...	...	...	...
19995	46.91	4.89	219.10	6.6	39.8	6.6	39.8	1071.5200	32.81	-0.06	1
19996	46.95	4.81	218.71	6.0	39.8	6.0	39.8	1052.4700	32.88	-0.06	1
19997	47.09	4.81	219.35	5.5	40.0	5.5	40.0	1055.5500	33.00	-0.13	1
19998	46.95	4.76	219.72	5.1	40.2	5.1	40.2	1045.8300	33.06	-0.19	1
19999	47.19	4.79	217.89	7.4	40.7	7.4	40.7	1042.8100	33.13	-0.19	1

20000 rows × 11 columns

Nota: *Se representa al dataframe procesado final, en la que contiene 20000 muestras para 11 columnas.*

5.5.3 División de la Base de Datos

Esta fase de división del repositorio de datos es importante para preparar los datos de manera adecuada para el entrenamiento, validación y evaluación de modelos. Utilizando la función "train_test_split" de la biblioteca "sklearn.model_selection", el repositorio de datos se divide en tres subconjuntos: entrenamiento, validación y prueba. Este enfoque asegura una distribución aleatoria y representativa de los datos originales en cada subconjunto. Con el subconjunto de entrenamiento para ajustar las características del modelo, el subconjunto de validación sirve para afinar los hiperparámetros y evitar el sobreajuste, y el subconjunto de prueba se emplea para calificar el performance final del modelo. Este procedimiento garantiza que el modelo mantenga un grado alto de exactitud y fiabilidad en situaciones reales, como evidencia la figura 69, primero se le asigna un 50% al conjunto train, 50% al resto y de este resto el 88% para el subconjunto validation y 12% para el grupo test, esto se resume como 50% para el train, 44% para validation y 6% para el test. En cambio, la figura 70, se evidencia las características y cantidades reales de esta división (Guamán Buestán, 2019). (Scikit learn, 2007-2024)

Figura 70

Proceso división de la base de datos en training, validation y test.

```
from sklearn.model_selection import train_test_split
train, intermediate = train_test_split(dataframe_sdf, test_size=0.5, random_state=42)
validation, test = train_test_split(intermediate, test_size=0.12, random_state=42)
```

```
train.info()
```
✓ 3.3s

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 9434 to 15795
Data columns (total 11 columns):
 #    Column                  Non-Null Count   Dtype
---   ------                  --------------   -----
 0    object_temp             10000 non-null   float64
 1    current_rms             10000 non-null   float64
 2    voltage_rms             10000 non-null   float64
 3    high_pressure_sensor    10000 non-null   float64
 4    low_pressure_sensor     10000 non-null   float64
 5    disch_pressure          10000 non-null   float64
 6    suct_pressure           10000 non-null   float64
 7    consump_power           10000 non-null   float64
 8    cond_out_temp           10000 non-null   float64
 9    evap_out_temp           10000 non-null   float64
 10   clase                   10000 non-null   int64
```

```
validation.info()
```
✓ 0.2s

```
<class 'pandas.core.frame.DataFrame'>
Index: 8800 entries, 411 to 16798
Data columns (total 11 columns):
 #    Column                  Non-Null Count   Dtype
---   ------                  --------------   -----
 0    object_temp             8800 non-null    float64
 1    current_rms             8800 non-null    float64
 2    voltage_rms             8800 non-null    float64
 3    high_pressure_sensor    8800 non-null    float64
 4    low_pressure_sensor     8800 non-null    float64
 5    disch_pressure          8800 non-null    float64
 6    suct_pressure           8800 non-null    float64
 7    consump_power           8800 non-null    float64
 8    cond_out_temp           8800 non-null    float64
 9    evap_out_temp           8800 non-null    float64
 10   clase                   8800 non-null    int64
```

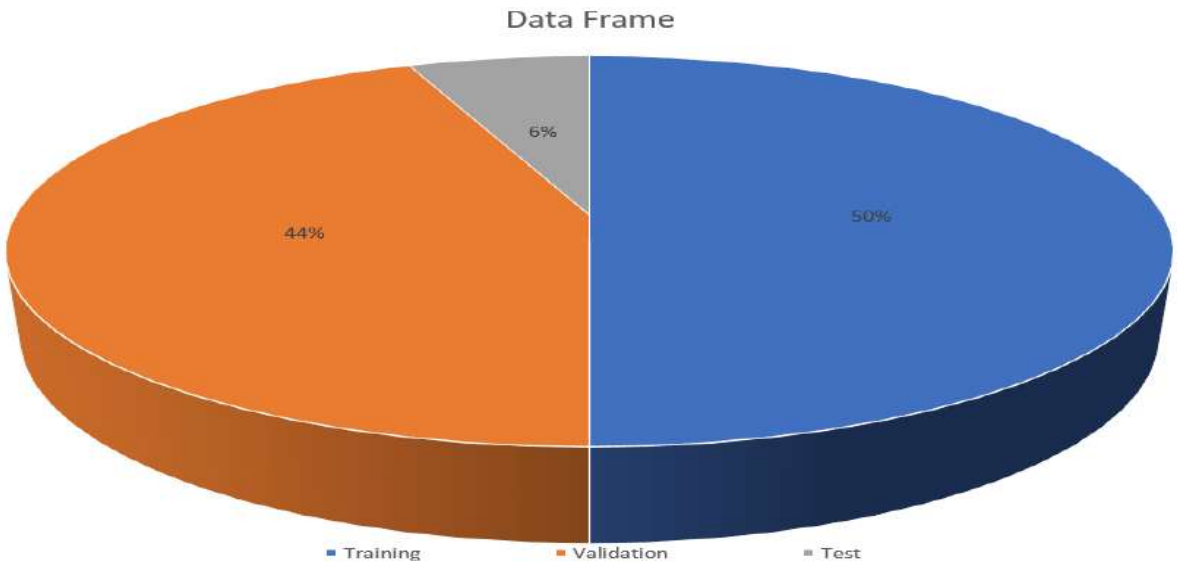
```
test.info()
```
✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
Index: 1200 entries, 7884 to 10862
Data columns (total 11 columns):
 #    Column                  Non-Null Count   Dtype
---   ------                  --------------   -----
 0    object_temp             1200 non-null    float64
 1    current_rms             1200 non-null    float64
 2    voltage_rms             1200 non-null    float64
 3    high_pressure_sensor    1200 non-null    float64
 4    low_pressure_sensor     1200 non-null    float64
 5    disch_pressure          1200 non-null    float64
 6    suct_pressure           1200 non-null    float64
 7    consump_power           1200 non-null    float64
 8    cond_out_temp           1200 non-null    float64
 9    evap_out_temp           1200 non-null    float64
 10   clase                   1200 non-null    int64
```

Como un extracto resumido, de esta etapa de base de datos, se expresa mediante gráficos circulares, la figura 71 expresa la subdivisión del grupo de datos ya procesado, en tres grupos; training, validation y test, con sus respetivos porcentajes, esta división se realiza en base, a los antecedentes similares sobre estos estudios.

Figura 71

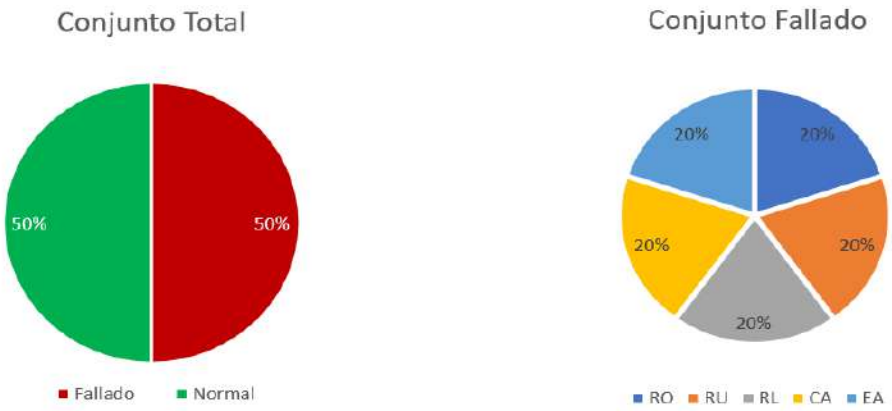
Diagrama circular de la visión de los conjuntos training, validation y test



De forma más detallada, se representa la subdivisión que comprende la base de datos generada para este proyecto, primero el conjunto total está compuesto por un 50% de muestras de evento de falla y 50% de muestras de evento normal, luego se representa de forma específica como está compuesto el conjunto de eventos de falla, donde el evento sub carga (UC Under Charge), representa el 20% de este conjunto, el evento sobre carga (OC Over Charge), representa el 20% del conjunto, el evento restricción de línea de líquido (RL Liquid-line Restriction), representa el 20% del conjunto, el evento reducción del flujo de aire del condensador (CA condenser airflow reduction), representa el 20% del conjunto y el evento reducción del flujo de aire del evaporador (EA evaporator airflow reduction), representa el 20% del conjunto. También de forma detalla mediante una tabla de 6x2 se representa, en detalle estos porcentajes.

Figura 72

Diagrama representativo de cada conjunto de datos



Conjuntos	Normal	RU	RO	RL	CA	EA
Muestras	10000	2000	2000	2000	2000	2000
Porcentajes	50%	10%	10%	10%	10%	10%

Nota: *Se representa de manera específica el porcentaje que representan cada conjunto de estos datos.*

5.6 Entrenamiento de los Modelos de Aprendizaje

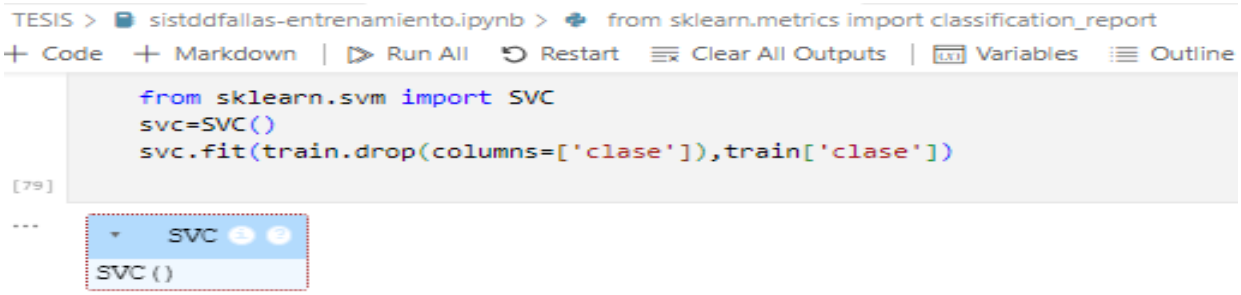
En este desarrollo, se describe el procedimiento para crear el modelo de aprendizaje supervisado destinado a clasificar eventos de fallas en el aire acondicionado. El entrenamiento de estos modelos se realizó proporcionando a los algoritmos, información de entrenamiento, que fueron previamente etiquetados y divididos en conjuntos de entrenamiento, validación y prueba. Esto permitió a los algoritmos aprender a identificar los vínculos y patrones presentes en los datos.

Se experimenta con múltiples algoritmos para optimizar la clasificación y el entrenamiento. Se utilizó el método de los k vecinos más cercanos (KNN), esta técnica es esencialmente simple y adecuada cuando las muestras están bien etiquetadas, como en este estudio específico, ofreciendo resultados satisfactorios para este tipo de detección. También, se empleó la técnica de máquinas de vectores de soporte (SVM), conocida por su habilidad para hacer predicciones con alta exactitud y rapidez en comparación con otros modelos, se implementó la técnica de bosques aleatorios (Random Forest), que

es altamente efectiva para esta clase de conjunto de datos, gradient boosting (GB), decision tree (DT) y por último naive bayes (NB) (Scikit learn, 2007-2024).

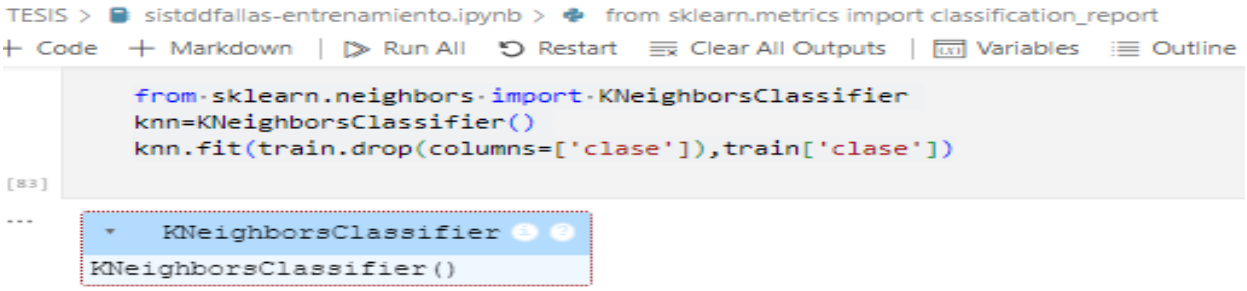
Usando la biblioteca “sklearn”, se entrenan diversos modelos, para el modelo SVC, se optimizan hiperparámetros como el parámetro de regularización y el kernel, a fin de mejorar la exactitud del clasificador y buscar resultados más confiables, conforme a lo ilustrado en la figura 73, se asigna a una variable la función SVC y mediante la función “fit” se ajusta los mejores parámetros para la data introducida.

Figura 73
Algoritmo de entrenamiento del modelo SVC



En el caso de KNN, se ajusta el número de vecinos para equilibrar las características entre el sesgo y la varianza, asegurando que el modelo generalice correctamente, como en la figura 74, se asigna a una variable la función KNeighborsClassifier y mediante la función “fit” se ajusta los mejores parámetros.

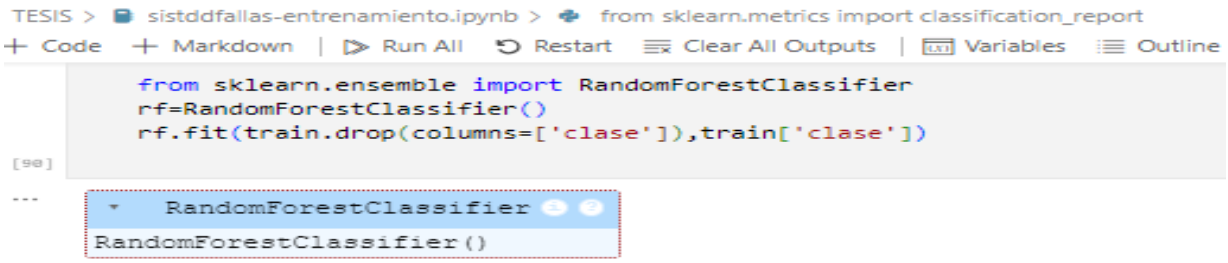
Figura 74
Algoritmo de entrenamiento del modelo KNeighborsClassifier



Para Random Forest, se cuantifican los árboles del bosque y se determina la profundidad máxima de cada árbol, entre otros hiperparámetros, con la finalidad de incrementar la exactitud y la capacidad

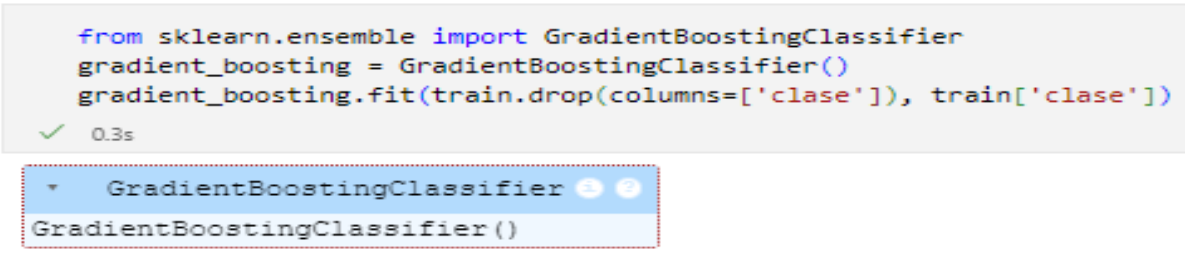
del modelo para manejar datos de alta dimensionalidad, como en la figura 75, se asigna a una variable la función RandomForestClassifier y mediante la función “fit” se ajusta los mejores.

Figura 75
Algoritmo de entrenamiento del modelo RandomForestClassifier



Usando la biblioteca “sklearn”, se entrena el modelo Gradient Boosting, optimizando hiperparámetros como el número de estimadores y la tasa de aprendizaje, para maximizar la precisión del clasificador y obtener un modelo robusto, conforme a lo ilustrado en la figura 76. Se asigna a una variable la función GradientBoostingClassifier y mediante la función “fit” se ajusta el modelo a la data introducida, logrando una mejor adaptación en problemas complejos al reducir gradualmente el error.

Figura 76
Algoritmo de entrenamiento del modelo GradientBoostingClassifier



Para el modelo Decision Tree, se ajustan hiperparámetros como la profundidad máxima del árbol y el criterio de división, equilibrando la habilidad del modelo para ubicar patrones complejos y evitando el sobreajuste, como se muestra en la figura 77. Se asigna a una variable la función DecisionTreeClassifier y mediante la función “fit” se entrena el árbol de decisión en el repositorio de datos proporcionado, logrando clasificaciones más interpretables y ajustadas.

Figura 77
Algoritmo de entrenamiento del modelo DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(df_subprueba.drop(columns=['clase']), df_subprueba['clase'])
```

0.0s

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

En el caso del modelo Naive Bayes, se elige el clasificador más idóneo según las cualidades de los datos (por ejemplo, GaussianNB para datos con distribución normal), asegurando un ajuste rápido y eficiente, como se detalla en la figura 78. Se asigna a una variable la función GaussianNB y mediante la función “fit” entrenamos el modelo, aprovechando su habilida para manejar muchos datos y su efectividad en clasificaciones con independencia condicional entre variables.

Figura 78

Algoritmo de entrenamiento del modelo GaussianNB

```
from sklearn.naive_bayes import GaussianNB
naive_bayes = GaussianNB()
naive_bayes.fit(train.drop(columns=['clase']), train['clase'])
```

0.0s

```
GaussianNB
GaussianNB()
```

Tras el entrenamiento de estos algoritmos, se generan archivos en formato pickle (.pkl), un formato nativo de Python utilizado para serializar y deserializar tipos de datos. Estos archivos contienen los pesos y los parámetros de los modelos entrenados.

5.7 Validación de los Modelos de Aprendizaje

El ajuste de hiperparámetros es una tarea laboriosa y repetitiva, ya que requiere probar numerosas combinaciones de valores. Para simplificar esta evaluación, se empleó una funcionalidad de la biblioteca Scikit Learn, que permite experimentar con diversas combinaciones de hiperparámetros especificados en una lista. Como resultado, se obtuvo la combinación de hiperparámetros que optimizaron el modelo, evaluado mediante validación cruzada.

El procedimiento de validación utilizando GridSearchCV se realizó con el grupo de datos de validación. GridSearchCV divide estos datos en varios subconjuntos, analizando cada combinación de

hiperparámetros y destacando la que ofrece los mejores resultados. Este enfoque no solo ajusta los modelos con precisión, sino que también previene el sobreajuste, asegurando que los modelos generalicen adecuadamente con datos nuevos. Al concluir el proceso de validación, se consigue el modelo con los parámetros óptimos, listo para evaluar en el grupo de datos de prueba y posteriormente aplicado en escenarios reales para clasificar fallas en el sistema de aire acondicionado.

Figura 79

Módulos del sistema de diagnóstico y detección de fallas

5.8 Software Utilizado

Tabla 13

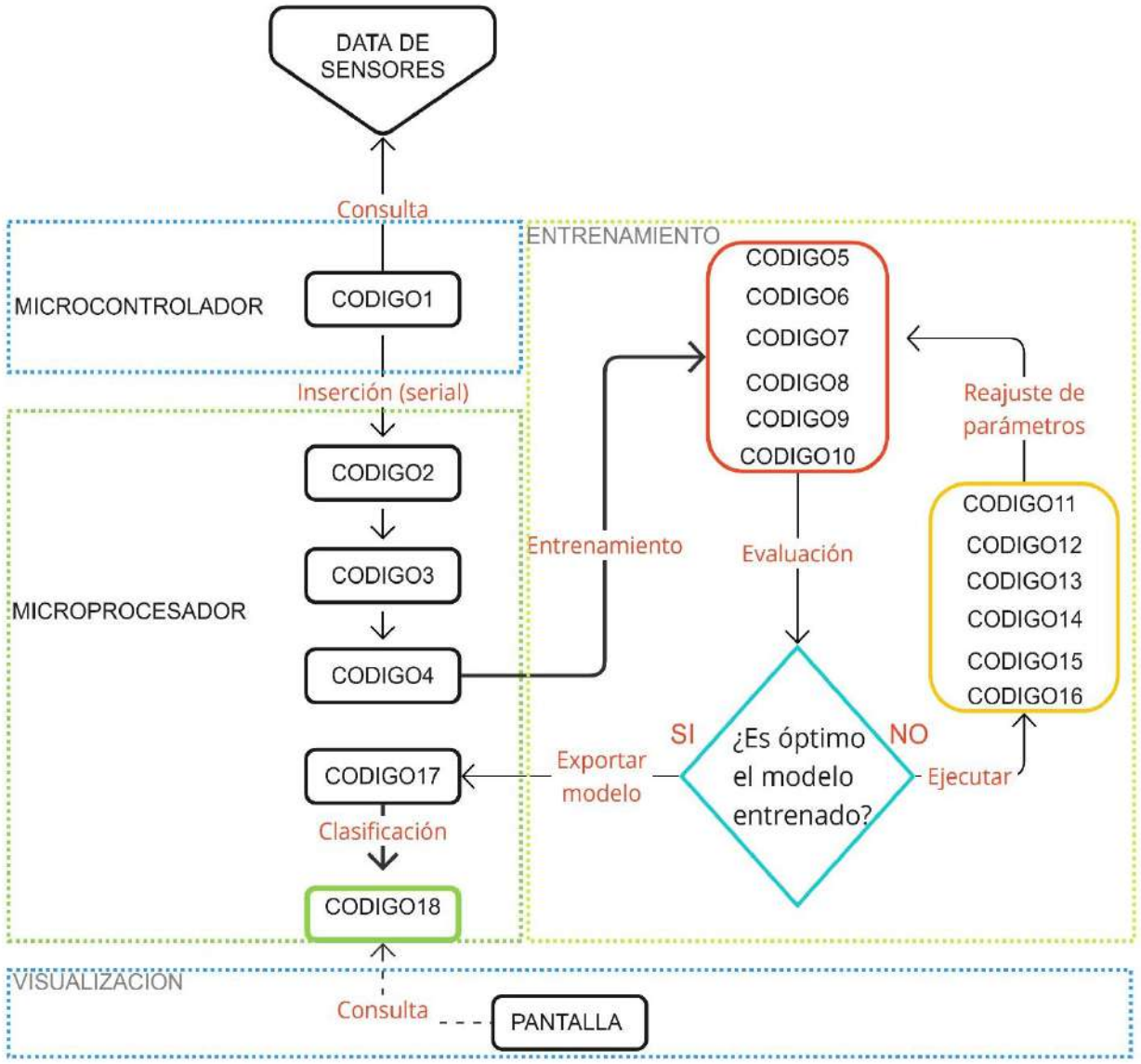
Resumen de softwares utilizados

Librería	Versión	Descripción
Visual Studio Code	1.89.1	Editor de código versátil que te permite escribir scripts y programas en Python para procesar datos, entrenar modelos de aprendizaje automático, analizar resultados y crear interfaces gráficas para interactuar con el sistema de diagnóstico.
Geany	1.37.1	Editor de código ligero ideal para desarrollar scripts y programas en Python sencillos, editar archivos de configuración y automatizar tareas repetitivas en la investigación. Además, es un editor por defecto que viene en el microprocesador seleccionado.
Arduino Ide	2.3.2	Un entorno de programación para microcontroladores Arduino, que te permite programar sensores para recopilar datos del sistema de aire acondicionado y establecer comunicación entre el sistema de adquisición y el de diagnóstico.
Pyserial	3.5	Se presenta como un puente que conecta los dispositivos serie, brindando una interfaz simple y versátil para interactuar con los sensores, microcontroladores y microprocesadores. Su facilidad de uso, flexibilidad, gran gama de funcionalidades y amplia comunidad la convierten en una herramienta esencial para desarrolladores que trabajan en automatización, adquisición de datos, comunicación con microcontroladores.
Pandas	2.2.0	Provee herramientas para analizar y manipular datos de sensores, limpiarlos, explorarlos, identificar patrones y prepararlos para entrenar modelos de aprendizaje automático.
Numpy	1.26.3	Provee herramientas de soporte para creación de objetos de tipo array multidimensionales (como vectores y matrices), también para realizar cálculos matemáticos sobre datos de sensores, normalizarlos, escalarlos, transformarlos, calcular características y métricas, e implementar algoritmos de aprendizaje automático.
Scikit-learn	1.4.1	Provee de simples y eficientes herramientas para el análisis de predicción de data, para implementar diversos algoritmos de aprendizaje automático como árboles de decisión, SVM, RF y KNN para clasificar fallas, evaluar su rendimiento y seleccionar el modelo más adecuado para el sistema diagnosticado.
Scipy	1.12.0	Provee de algoritmos y herramientas para realizar análisis estadísticos de datos de sensores, implementar algoritmos de optimización para mejorar el rendimiento del sistema de diagnóstico y crear visualizaciones de datos avanzadas para comprender mejor los resultados.
MySQL	15.1	Un sistema de bases de datos para almacenar datos de sensores y resultados del diagnóstico, recuperarlos para entrenar y evaluar modelos, e implementar el sistema de monitoreo y visualización de datos en tiempo real.

5.9 Diagramas de Interacción de Códigos

Figura 80

Diagrama de flujo de la interacción de códigos



Capítulo VI

Pruebas y Resultados

Como etapa final, se describen las pruebas y resultados del sistema de detección y diagnóstico de fallas en el aire acondicionado de precisión, desarrollado con métodos de machine learning y aprendizaje supervisado, con un enfoque en la evaluación exhaustiva, integral del sistema.

Se llevaron a cabo dos formas de pruebas. Inicialmente, se realizaron pruebas utilizando condiciones diferidas, de fallas del sistema AAP, utilizando una base de datos de fallas, generada a partir de las señales de estado del sistema evaluado de forma controlada. Estas pruebas se ejecutaron en un ordenador para evaluar la exactitud del sistema para detectar de fallas. Posterior al estudio del performance del sistema en condición diferida, se prosiguió con la ejecución de pruebas en tiempo real, que conllevo instalar el sistema en un aire acondicionado de precisión operativo bajo condiciones reales, para evaluar su capacidad de detectar y diagnosticar fallos en un entorno de funcionamiento práctico.

6.1 Pruebas y Resultados de las Etapas de este Proyecto

6.1.1 Sistema de Instrumentación y Detección de Fallas del Sistema AAP

Se realizaron pruebas de este sistema de instrumentación y detección de eventos de fallas en conjunto también de forma individual, componente por componente; como se observa en la figura 81, se realizaron en el Raspberry pi 4b, el Arduino nano y los diferentes tipos de sensores (P, T°, V, I).

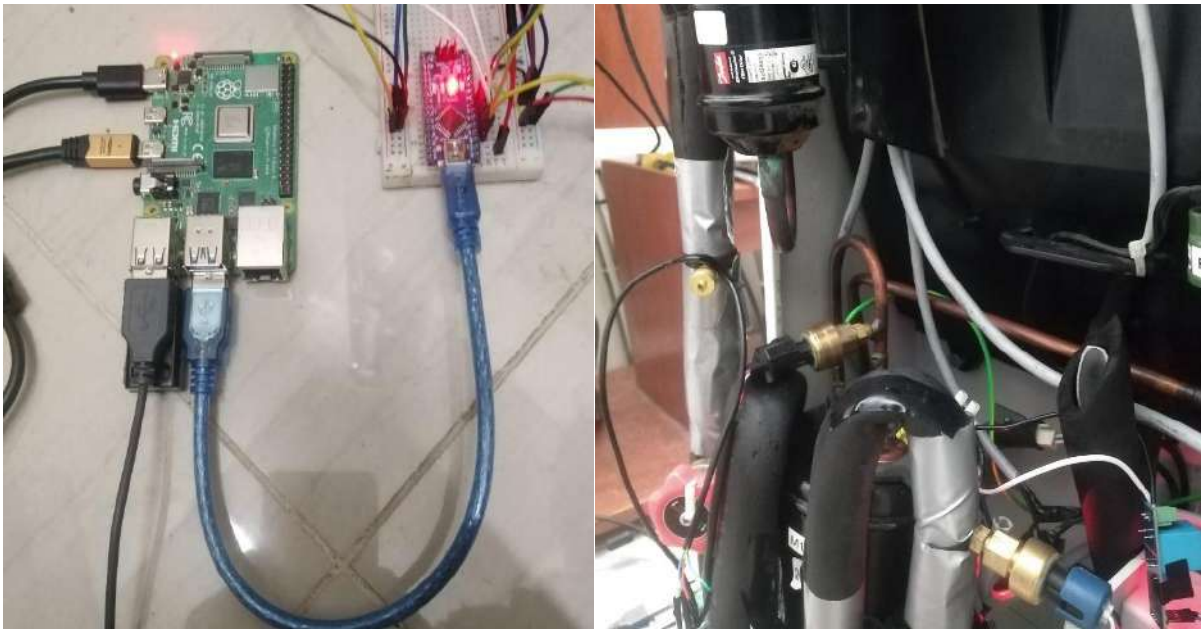
EL procesador central, requiere de un suministro de energía de 5.1V/3A, que sirve como alimentación para la unidad de adquisición, mediante USB, que a su vez alimenta a la etapa instrumental (distintos sensores P, T°, V, I).

Para evaluar el sistema de detección, se utilizó el mismo código representado en pseudocódigo para el sensado y la digitalización, para realizar la tarea especifica de captar las señales en los pines de entrada del microprocesador de forma automática y continua. De la siguiente forma, primero se declara las variables con el valor de los pines A0, A1, A2, A3, A4, A5, A6, también declaramos constantes (como

valores de sensibilidad) proporcionada en las hojas de datos de cada sensor, luego inicializamos la comunicación serial, en la parte principal del código obtenemos el valor de estas variables mediante relaciones recomendadas por los fabricantes, por último, se retorna estos valores en formato json, para convertirlos posteriormente en archivos csv.

Figura 81

Prueba del sistema de instrumentación y detección de fallas del sistema AAP



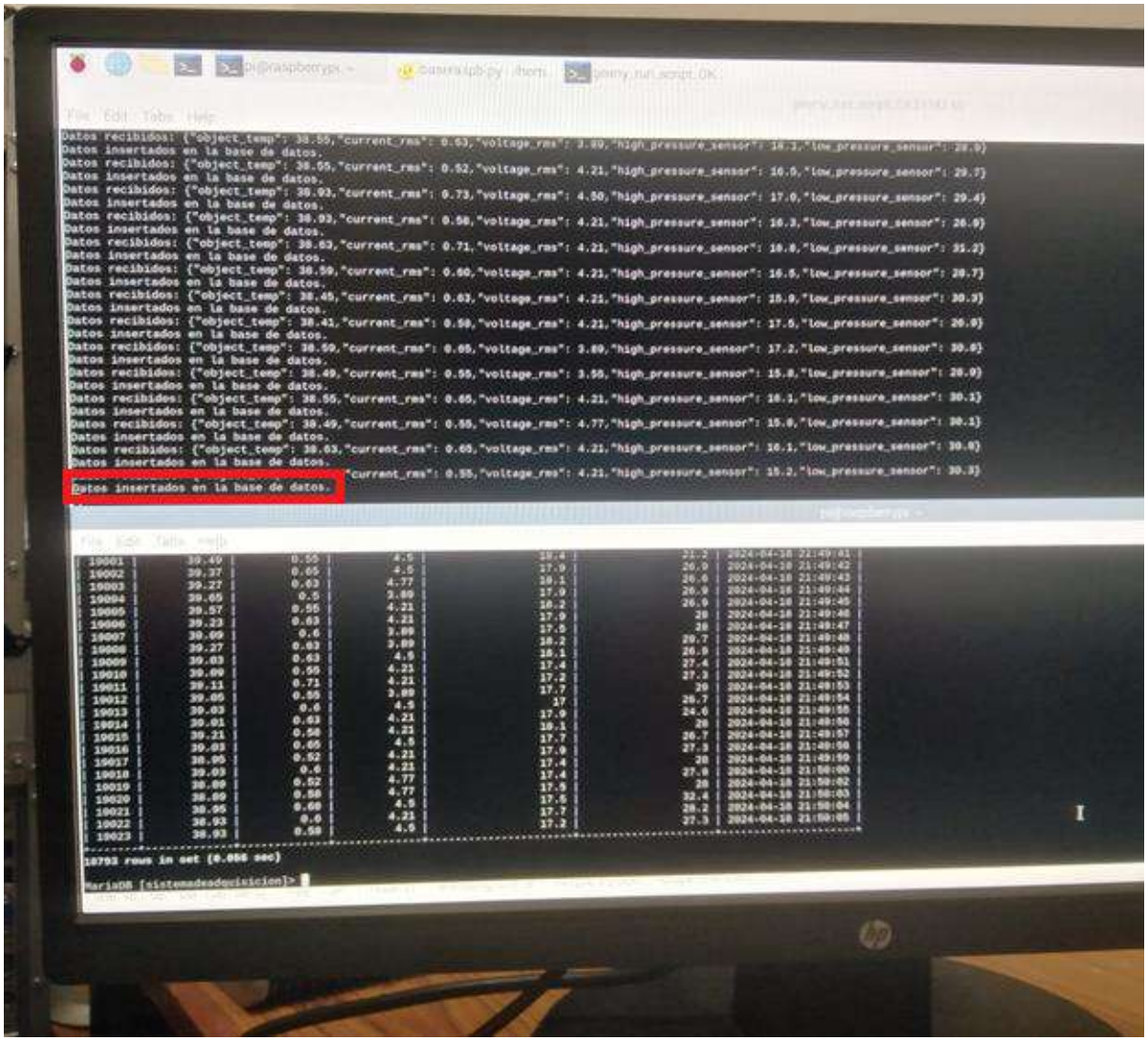
6.1.2 Base de Datos de Fallas del Sistema AAP

Se realizaron pruebas buscando cumplir el segundo objetivo, de obtener la base de datos con señales de condición de estado para situaciones con y sin fallas; se realiza en el embebido Raspberry pi 4b, principalmente con la plataforma MariaDB (MySQL) para evaluar esta etapa, se utilizó el mismo código, con el fin de automatizar la generación continua de un repositorio de datos. De la forma siguiente se vincula a la base de datos introduciendo información de usuario, nombre del repositorio de datos y contraseña, de forma predeterminada. Se crear este repositorio de datos, la tabla donde se almacena, el formato de la tabla con detalles de los valores de los datos que serán introducidos, mediante líneas de comando del lenguaje SQL, luego se establece la comunicación del puerto serial. En la parte principal del algoritmo, se empieza con la lectura de los datos del puerto serial, luego estos datos los pasamos a

formato json, por último, se pasa a introducir en la base de datos, especificando el nombre las columnas, que debe ser idéntico a los encabezados establecidos anteriormente. Esta última etapa debe ser repetitiva y continua. En la figura 82, se visualiza como los datos son insertados en MySQL.

Figura 82

Prueba de inserción de base de datos

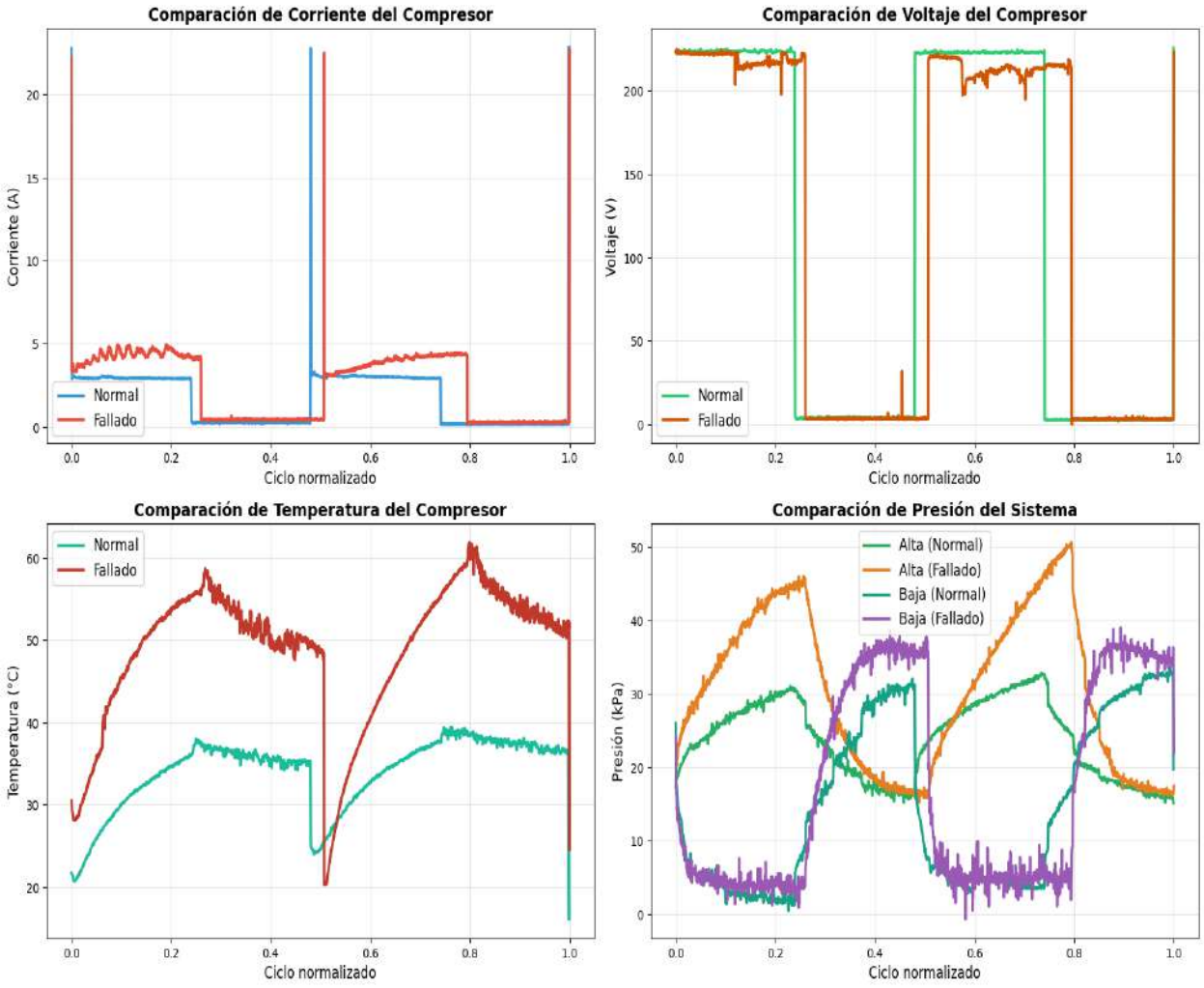


La Figura 83 compara dos ciclos completos de operación del sistema del compresor: uno en condiciones normales (representado por líneas azules/verdes) y otro con una falla por subcarga de refrigerante (indicado por líneas rojas/naranjas). Las señales muestran patrones claramente diferenciados, característicos de esta falla crítica. Como se aprecia en los gráficos, la subcarga de refrigerante provoca niveles de voltaje elevados, mientras que la corriente se mantiene inusualmente

baja, un patrón que indica que el compresor está operando sin la carga adecuada. Además, la temperatura experimenta un aumento sostenido, lo que evidencia un sobrecalentamiento del circuito debido a la insuficiente cantidad de refrigerante para disipar el calor generado. Aunque la presión puede parecer normal en términos de valores absolutos, su comportamiento dinámico se desvía del patrón estándar. Cabe destacar que el ciclo de funcionamiento con falla es notablemente más largo que el normal, lo que confirma la ineficiencia del sistema para alcanzar las temperaturas deseadas. Este problema de comprimir adecuadamente y lograr una refrigeración efectiva resulta en tiempos de operación prolongados que, eventualmente, pueden causar daños permanentes en el compresor y sus componentes asociados.

Figura 83

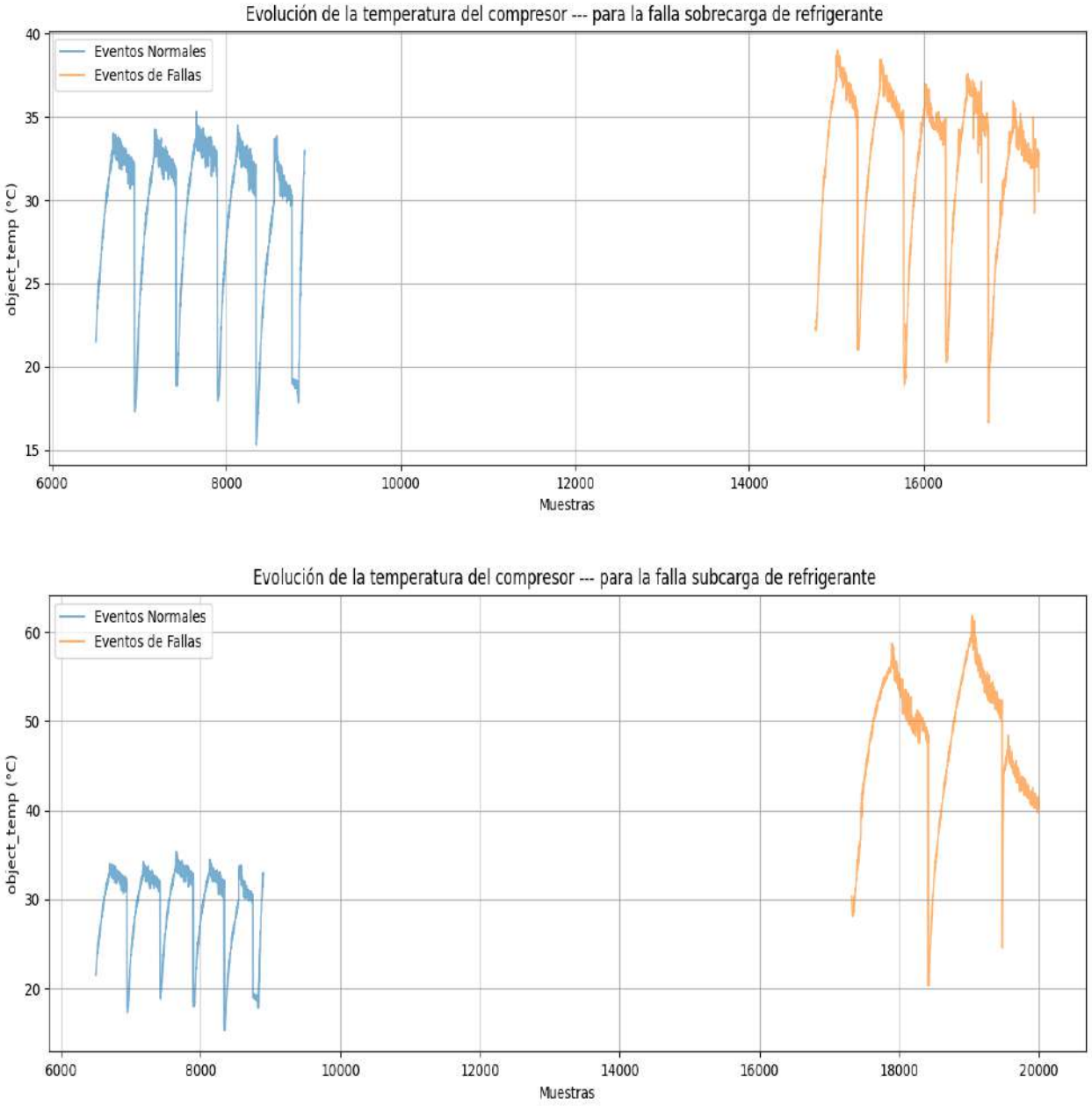
Comparación entre evento normal y de falla

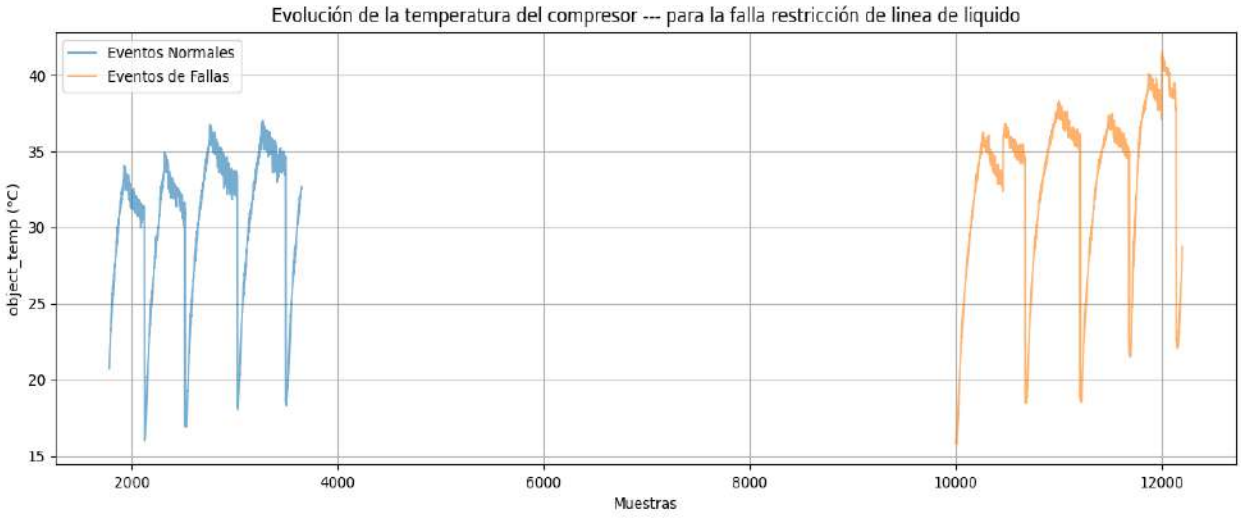


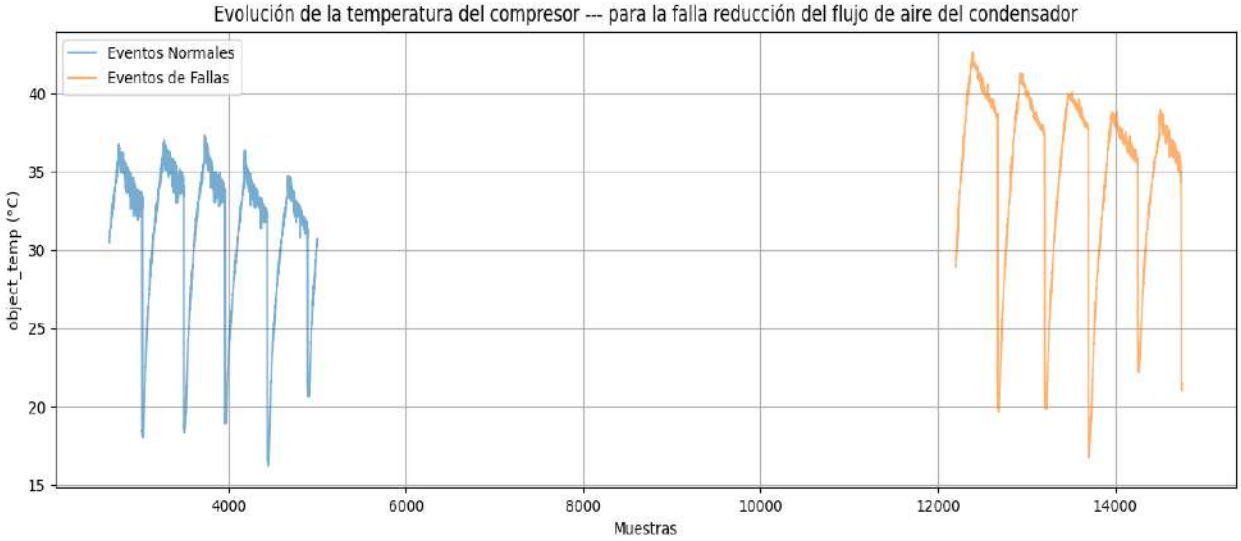
Las curvas comparativas de la figura 84 ilustran cómo evolucionan las características en los cinco tipos de fallas evaluados. La señal azul representa el funcionamiento normal y la señal naranja los eventos de falla. En función de la temperatura del compresor (°C), seleccionada como variable principal por su mejor capacidad para evidenciar la diferencia entre operación normal y estados de falla. Principalmente se nota el incremento diferenciado en el nivel de temperatura de cada patrón de cada falla.

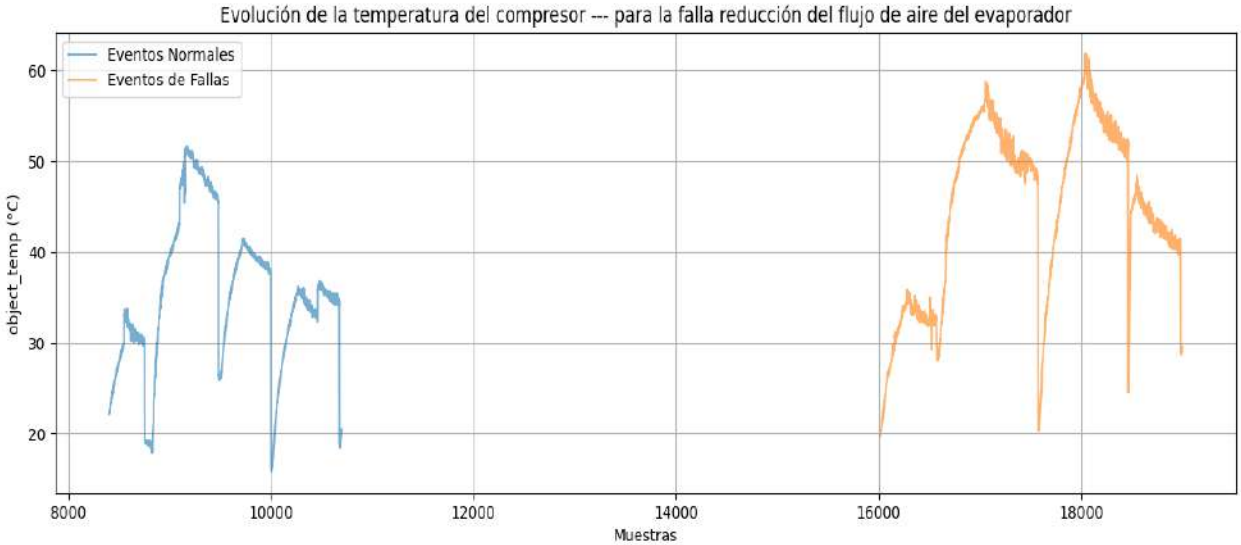
Figura 84

Representación del cambio de características de cada tipo de falla evaluada




Evolución de la temperatura del compresor --- para la falla restricción de linea de liquido
Eventos Normales
Eventos de Fallas
object_temp (°C)
Muestras


Evolución de la temperatura del compresor --- para la falla reducción del flujo de aire del condensador
Eventos Normales
Eventos de Fallas
object_temp (°C)
Muestras


Evolución de la temperatura del compresor --- para la falla reducción del flujo de aire del evaporador
Eventos Normales
Eventos de Fallas
object_temp (°C)
Muestras

6.2 Pruebas y Resultados de los Modelos de Clasificación en Condición Diferida

Los diversos modelos de aprendizaje fueron puestos a prueba y sus desempeños se muestran en esta sección. Se seleccionaron los modelos Support Vector Machines, K-Nearest Neighbors, Random Forest, Gradient Boosting, Decision Tree y Naive Bayes. Las pruebas se llevaron a cabo sobre el aire acondicionado de precisión piloto, de la marca Rittal, modelo SK3328.500 (enclosure top therm, cooling unit), para las fallas específicas evaluadas y ejecutadas según el lineamiento de los antecedentes.

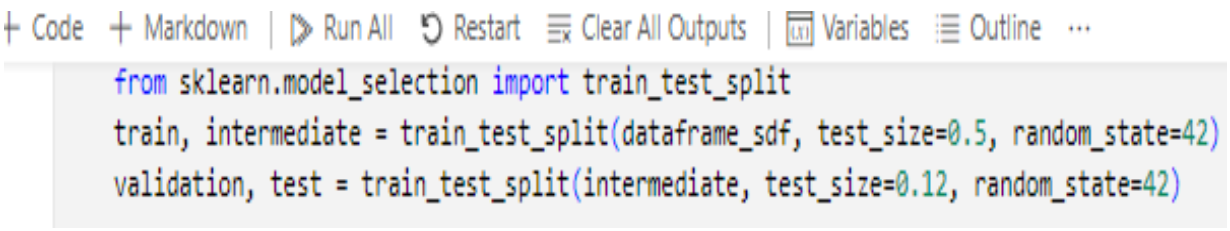
6.2.1 Pruebas y Resultados del Modelo Support Vector Machines (SVM)

6.2.1.1 Pruebas en el Entorno de la Computadora

El desarrollo y las pruebas del sistema se ejecutaron en la misma máquina, durante estas pruebas, se evaluó cómo el sistema detecta las fallas del aire acondicionado al analizar las señales de estado y los patrones asociados a cada evento de falla evaluado.

Las pruebas para los modelos de entrenamiento, empieza con la etapa de dividir la base de datos total, en tres subconjuntos denominados; train, validation y test. Basado en la recomendación de los antecedentes y literatura especializada, que mencionan diferentes niveles de porcentaje de división. Esta parte se desarrolló empleando la función train_test_split de la librería sklearn, como se observa en la figura 85, los porcentajes de división de los conjuntos y que son utilizado por todos los modelos evaluados.

Figura 85
Fase de división de subconjuntos



Se buscó el modelo de Machine Learning más exacto, mediante el algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se representa la exactitud (accuracy) del modelo SVM, cabe destacar que esta métrica se determina, reemplazando los valores de la ecuación 24, la cantidad de verdaderospositivos (VP), verdaderosnegativos (VN), falsosnegativos (FN) y falsospositivos

(FP) en la figura 86, se evidencia un 96% de accuracy. Este porcentaje de accuracy se utilizó como indicador principal para analizar los patrones de comportamiento y elegir el modelo más apropiado.

Figura 86

Reporte de clasificación de modelo SVM etapa entrenamiento

	precision	recall	f1-score	support
0	0.97	0.98	0.97	7467
1	0.93	0.90	0.92	2533
accuracy			0.96	10000
macro avg	0.95	0.94	0.94	10000
weighted avg	0.96	0.96	0.96	10000

Luego del proceso de entrenamiento se evaluó las diversas combinaciones de propiedades para mejorar este valor, conforme se presenta en la figura 87, la evaluación de la combinación de hiperparámetros, mediante la función GridSearchCV, donde se especificó mediante una lista valores de "C", "gamma" y también se usó el proceso cross validation (CV), donde la variable de regularización (C), controla el margen y Gamma diseña el margen de separación dentro del espacio de características y regula la configuración de la función de decisión.

Figura 87

Proceso de validación para el modelo SVM

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {'C':[400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000, 1100, 1150],
              'gamma':[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,
                       0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09,
                       0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009,]}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
svc = svm.SVC()
clf = GridSearchCV(svc, parameters, cv=8, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa una parte del algoritmo para la etapa de validation, mediante una lista definida de los hiperparámetros con los que se pretende validar el modelo SVM.*

Además, en la figura 88, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy del proceso de validación, como se puede observar, llegando a 95.05% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 88
Reporte de mejores parámetros modelo SVM etapa validación

```
SVC(C=900, gamma=0.01)
{'C': 900, 'gamma': 0.01}
0.9505681818181819
```

6.2.1.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento Support Vector Machines (SVM), evaluado en el conjunto test dividido previamente. Como se muestra en la figura 89, nos resulta un nivel de porcentaje bueno del 93% de accuracy.

Figura 89
Reporte de clasificación de modelo SVM etapa testeo

```
from sklearn.metrics import classification_report
print(classification_report(test['clase'], svc.predict(test.drop(columns=['clase']))))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       889
           1       0.90      0.83      0.86       311

    accuracy                           0.93      1200
   macro avg       0.92      0.90      0.91      1200
weighted avg       0.93      0.93      0.93      1200
```

6.2.2 Pruebas y Resultados del Modelo K-Nearest Neighbors (KNN)

6.2.2.1 Pruebas en el Entorno de la Computadora

También de forma similar, el desarrollo y las pruebas del sistema se llevaron a cabo en la misma máquina. Donde se examinó la eficiencia del sistema para detectar las fallas del aire acondicionado, mediante el análisis de las señales de condición de estado y los patrones que se forman para cada evento de falla evaluado.

Se buscó el modelo más exacto, mediante el algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se representa la exactitud del modelo KNN, en la figura 90, es igual al 83% de accuracy.

Figura 90

Reporte de clasificación de modelo KNN etapa entrenamiento

	precision	recall	f1-score	support
0	0.82	1.00	0.90	7467
1	0.97	0.36	0.52	2533
accuracy			0.83	10000
macro avg	0.90	0.68	0.71	10000
weighted avg	0.86	0.83	0.80	10000

Como del proceso de entrenamiento se obtuvo, 83% de accuracy, este valor es bueno, pero aún es bajo, por tanto, se evaluó las de diversas combinaciones de propiedades para mejorar este valor, tal como se representa en la figura 91, la evaluación de la combinación de hiperparámetros, mediante la función GridSearchCV, donde se especificó mediante una lista valores de “n-neighbors”, “weights”, “metric” y también se usa el proceso cross validation (CV).

Figura 91

Proceso de validación para el modelo KNN

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {'n_neighbors':[1, 3, 5, 7, 9, 25, 35, 45, 55, 65, 75, 85, 95],
              'weights':['uniform', 'distance'],
              'metric': ['euclidean', 'manhattan', 'minkowski']}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
knn = neighbors.KNeighborsClassifier()
clf = GridSearchCV(knn, parameters, cv=iteraciones, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa una parte del algoritmo para la etapa de validation, mediante una lista definida de los hiperparámetros con los que se pretende validar el modelo KNN.*

En la figura 92, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy, del proceso de validación, como se puede observar mejora, llegando a más del 88% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 92
Reporte de mejores parámetros modelo KNN etapa validación

```
KNeighborsClassifier(metric='manhattan', n_neighbors=3, weights='distance')
{'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
```

6.2.2.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento K-Nearest Neighbors, evaluado en el conjunto test dividido previamente. Como se representa en la figura 93, resultó un nivel de porcentaje bueno del 83% de accuracy.

Figura 93
Reporte de clasificación de modelo KNN etapa testeo

```
from sklearn.metrics import classification_report
print(classification_report(test['clase'], knn.predict(test.drop(columns=['clase']))))
```

	precision	recall	f1-score	support
0	0.82	1.00	0.90	889
1	0.98	0.36	0.53	311
accuracy			0.83	1200
macro avg	0.90	0.68	0.71	1200
weighted avg	0.86	0.83	0.80	1200

6.2.3 Pruebas y Resultados del Modelo Random Forest (RF)

6.2.3.1 Pruebas en el Entorno de la Computadora

También de forma similar, el desarrollo y las pruebas del sistema se llevaron a cabo en la misma máquina. Donde se examinó la eficiencia del sistema para detectar las fallas del aire acondicionado,

mediante el análisis de las señales de condición de estado y los patrones que se forman para cada evento de falla evaluado.

Se buscó un modelo más exacto, mediante el algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se observa la accuracy del modelo RF, en la figura 94.

Figura 94

Reporte de clasificación de modelo RF etapa entrenamiento

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7467
1	1.00	1.00	1.00	2533
accuracy			1.00	10000
macro avg	1.00	1.00	1.00	10000
weighted avg	1.00	1.00	1.00	10000

Luego del proceso de entrenamiento se obtuvo, se evaluó las de diversas combinaciones de propiedades para mejorar este valor, según se observa en la figura 95, el diagnóstico de la combinación de hiperparámetros, mediante la función GridSearchCV, donde se especifica mediante una lista valores de “n-estimators”, “max_depth”, “min_samples_split”, “min_samples_leaf” y también se usó el proceso cross validation.

Figura 95

Proceso de validación para el modelo RF

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {'n_estimators': [100, 150, 200, 250, 300, 350, 400, 450, 500],
              'max_depth': [None, 10, 15, 20, 25, 30, 35, 40, 45, 50],
              'min_samples_split': [0,5, 1, 2, 3, 4, 5],
              'min_samples_leaf': [0,5, 1, 2, 3, 4, 5]}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
rf = RandomForestClassifier()
clf = GridSearchCV(rf, parameters, cv=10, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa la validation de modelo RF, mediante una lista de hiperparámetros.*

El modelo optimizado alcanzó una exactitud del 95.28%, destacándose como el más eficiente de los métodos probados en esta investigación. Esto evidencia la habilida del modelo RF para manejar la complejidad y la variabilidad inherentes en este tipo de sistemas AAP, proporcionando resultados robustos y confiables en la detección de fallas.

En la figura 96, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy, del proceso de validación, como se puede observar mejora, llegando a más del 95.28% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 96

Reporte de mejores parámetros modelo RF etapa validación

```
RandomForestClassifier(max_depth=25, n_estimators=150)
{'max_depth': 25, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
0.9528409090909091
```

6.2.3.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento Random Forest (RF), evaluado en el conjunto test dividido previamente. Como se representa en la figura 97, resultó un nivel de porcentaje sobresaliente del 96% de accuracy, que es similar al obtenido de la validación.

Figura 97

Reporte de clasificación de modelo RF etapa testeo

```
from sklearn.metrics import classification_report
print(classification_report(test['clase'], rf.predict(test.drop(columns=['clase']))))
```

✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       889
           1       0.97      0.88      0.92       311

    accuracy                           0.96      1200
   macro avg       0.96      0.93      0.95      1200
weighted avg       0.96      0.96      0.96      1200
```

6.2.4 Pruebas y Resultados del Modelo Gradient Boosting (GB)

6.2.4.1 Pruebas en el Entorno de la Computadora

También de forma similar, el desarrollo y las pruebas del sistema se llevaron a cabo en la misma máquina. Donde se examinó la eficiencia del sistema para detectar las fallas del aire acondicionado, mediante el análisis de las señales de condición de estado y los patrones que se forman para cada evento de falla evaluado.

Se buscó un modelo más exacto, mediante del algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se visualiza la exactitud del modelo GB, en la figura 98, es igual al 93% de accuracy.

Figura 98
Reporte de clasificación de modelo GB etapa entrenamiento

	precision	recall	f1-score	support
0	0.93	0.98	0.95	7467
1	0.93	0.78	0.85	2533
accuracy			0.93	10000
macro avg	0.93	0.88	0.90	10000
weighted avg	0.93	0.93	0.93	10000

Luego del proceso de entrenamiento, se evaluó las diversas combinaciones de propiedades para mejorar este modelo, según se observa en la figura 99, Para Gradient Boosting, se optimiza los hiperparámetros como la tasa de aprendizaje y el número de estimadores para maximizar la precisión del clasificador y obtener un modelo robusto, mediante la función GridSearchCV y también se usó el proceso cross validation.

Figura 99
Proceso de validación para el modelo GB

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {
    'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [3, 4, 5, 6],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
gb = GradientBoostingClassifier()
clf = GridSearchCV(gb, parameters, cv=iteraciones, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa la validation de modelo GB, mediante una lista de hiperparámetros.*

En la figura 100, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy, del proceso de validación, como se puede observar mejora, llegando a 94.94% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 100

Reporte de mejores parámetros modelo GB etapa validación

```
GradientBoostingClassifier(learning_rate=0.3, max_depth=6, min_samples_leaf=2,
                           min_samples_split=5, n_estimators=150)
{'learning_rate': 0.3, 'max_depth': 6, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 150}
0.9494318181818182
```

6.2.4.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento Gradient Boosting (GB), evaluado en el conjunto test dividido previamente. Como se representa en la figura 101, resultó un valor de porcentaje bueno del 91% de accuracy.

Figura 101
Reporte de clasificación de modelo RF etapa testeo

```
from sklearn.metrics import classification_report
predicciones_gb = gradient_boosting.predict(test.drop(columns=['clase']))
print(classification_report(test['clase'], predicciones_gb))
```

✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       889
           1       0.91      0.74      0.82       311

    accuracy                           0.91      1200
   macro avg       0.91      0.86      0.88      1200
weighted avg       0.91      0.91      0.91      1200
```

6.2.5 Pruebas y Resultados del Modelo Decision Tree (DT)

6.2.5.1 Pruebas en el Entorno de la Computadora

También de forma similar, el desarrollo y las pruebas del sistema se llevaron a cabo en la misma máquina. Donde se examinó la eficiencia del sistema para detectar las fallas del aire acondicionado, mediante el análisis de las señales de condición de estado y los patrones que se forman para cada evento de falla evaluado.

Se buscó un modelo más exacto, mediante del algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se representa la accuracy del modelo RF, en la figura 102.

Figura 102
Reporte de clasificación de modelo DT etapa entrenamiento

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7467
           1       1.00      1.00      1.00      2533

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000
```

Luego del proceso de entrenamiento se obtuvo, se evaluó las diversas combinaciones de propiedades para mejorar este modelo, según se observa en la figura 103, Para Decisión Tree, los

hiperparámetros que se optimizan son la profundidad máxima del árbol y el criterio de división, equilibrando la capacidad del modelo, mediante la función GridSearchCV y también se usó el proceso cross validation.

Figura 103
Proceso de validación para el modelo DT

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {'criterion': ['gini', 'entropy'],
              'max_depth': [None, 10, 20, 30],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
dt = DecisionTreeClassifier()
clf = GridSearchCV(dt, parameters, cv=iteraciones, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa la validation de modelo DT, mediante una lista de hiperparámetros.*

El modelo optimizado alcanzó una exactitud (accuracy) del 93.12%, destacándose como el más eficiente de los métodos probados en esta investigación. Esto evidencia la capacidad del modelo Decision Tree para controlar la complejidad y la variabilidad inherentes en este tipo de sistemas AAP, proporcionando resultados robustos y confiables en la detección de fallas.

En la figura 104, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy, del proceso de validación, como se puede observar mejora, llegando a más del 93.12% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 104
Reporte de mejores parámetros modelo RF etapa validación

```
DecisionTreeClassifier()
{'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
0.93125
```

6.2.5.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento Decision Tree (DT), evaluado en el conjunto test dividido previamente. Como se observa en la figura 105, resultó un nivel de porcentaje destacable del 93% de accuracy, que es similar al obtenido de la validación.

Figura 105

Reporte de clasificación de modelo DT etapa testeo

```
from sklearn.metrics import classification_report
predicciones_dt = decision_tree.predict(test.drop(columns=['clase']))
print(classification_report(test['clase'], predicciones_dt))
```

✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       889
           1       0.89      0.85      0.87       311

    accuracy                           0.93      1200
   macro avg       0.92      0.91      0.91      1200
weighted avg       0.93      0.93      0.93      1200
```

6.2.6 Pruebas y Resultados del Modelo Naive Bayes (NB)

6.2.6.1 Pruebas en el Entorno de la Computadora

También de forma similar, el desarrollo y las pruebas del sistema se llevaron a cabo en la misma máquina. Donde se examinó la eficiencia del sistema para detectar las fallas del aire acondicionado, mediante el análisis de las señales de condición de estado y los patrones que se forman para cada evento de falla evaluado.

Se buscó un modelo más exacto, mediante el algoritmo descrito, en la sección (5.6 Entrenamiento de los modelos de aprendizaje). Donde se resalta la accuracy del modelo NB, en la figura 106, es igual al 75% de accuracy.

Figura 106

Reporte de clasificación de modelo NB etapa entrenamiento

	precision	recall	f1-score	support
0	1.00	0.67	0.80	7467
1	0.50	0.99	0.67	2533
accuracy			0.75	10000
macro avg	0.75	0.83	0.73	10000
weighted avg	0.87	0.75	0.77	10000

Luego del proceso de entrenamiento se obtuvo, se evaluó las diversas combinaciones de propiedades para mejorar este modelo, según se observa en la figura 107, Para Naive Bayes, la optimización de los hiperparámetros está en relación a las características de los datos (por ejemplo, GaussianNB para datos con distribución normal), asegurando un ajuste rápido y eficiente, mediante la función GridSearchCV y también se usó el proceso cross validation.

Figura 107

Proceso de validación para el modelo NB

```
# Lista de hiperparámetros que el algoritmo GridSearchCV se encargará de combinar
parameters = {'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]}

# Prueba de todas las combinaciones posibles de la lista de hiperparámetros
nb = GaussianNB()
clf = GridSearchCV(nb, parameters, cv=iteraciones, return_train_score=False)

# Entrenamiento de distintos modelos bajo un training dataset definido
clf.fit(dataframe_vectores_caracteristicas, dataframe_columna_clase)
```

Nota: *Se representa la validation de modelo NB, mediante una lista de hiperparámetros.*

El modelo optimizado alcanzó una exactitud del 74.92%, destacándose como el más eficiente de los métodos probados en esta investigación. Esto demuestra la habilidad del modelo Naive Bayes para gestionar la complejidad y la variabilidad inherentes a la detección de fallas en este tipo de sistemas AAP.

En la figura 108, se representa el mejor estimador, los mejores parámetros y el mejor score del porcentaje de accuracy, del proceso de validación, como se puede observar mejora, llegando a más del 74.92% de accuracy, pero con los valores del conjunto de validation dividido previamente.

Figura 108
Reporte de mejores parámetros modelo NB etapa validación

```
GaussianNB(var_smoothing=0.1)
{'var_smoothing': 0.1}
0.7492045454545455
```

6.2.6.2 Resultados en el Entorno de la Computadora

El resultado de las pruebas sobre el modelo de entrenamiento Naive Bayes (NB), evaluado en el conjunto test dividido previamente. Como se grafica en la figura 109, resultó un nivel de porcentaje bajo del 77% de accuracy, que es similar al obtenido de la validación.

Figura 109
Reporte de clasificación de modelo RF etapa testeo

```
predicciones_nb = naive_bayes.predict(test.drop(columns=['clase']))
print(classification_report(test['clase'], predicciones_nb))
```

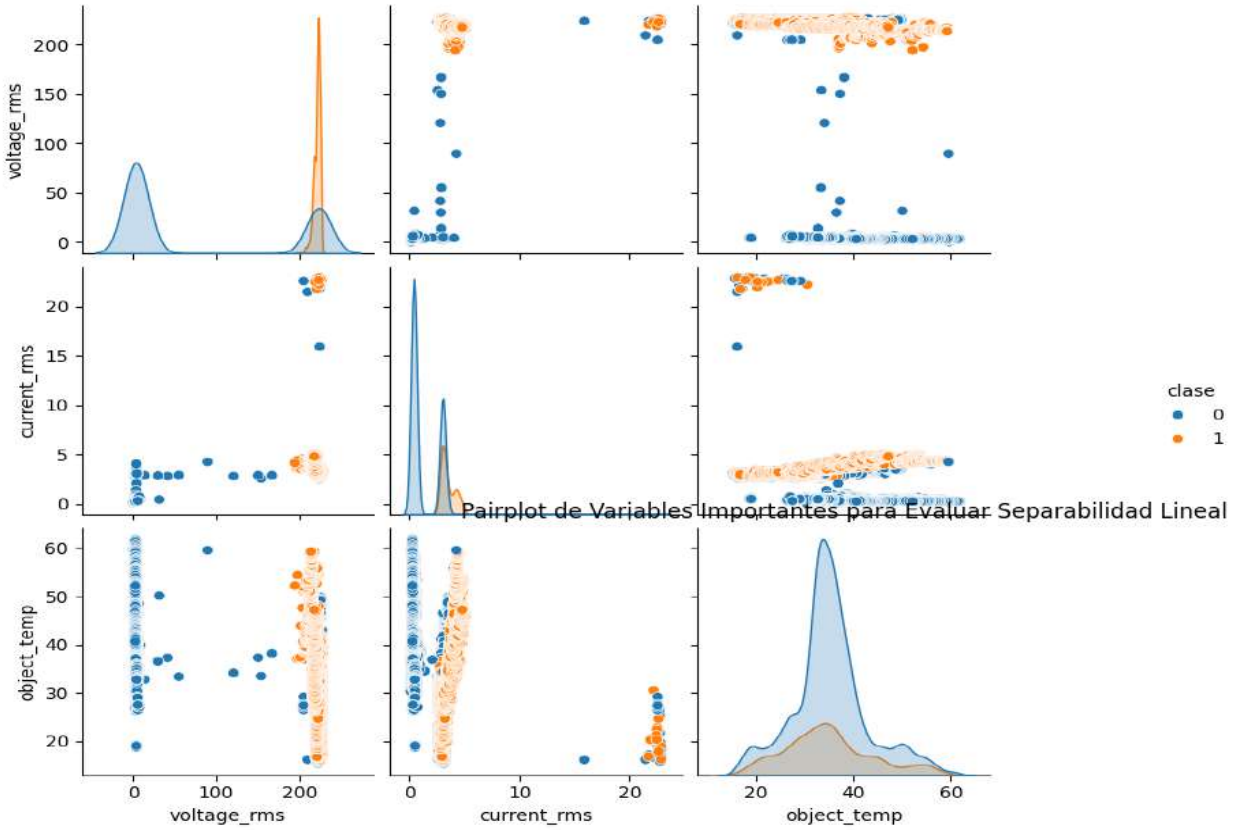
✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.99      0.69      0.81       889
           1       0.53      0.99      0.69       311

    accuracy                           0.77      1200
   macro avg       0.76      0.84      0.75      1200
weighted avg       0.87      0.77      0.78      1200
```

Hasta esta etapa, ya se puede observar cuales son los mejores modelos, por tanto, es importante analizar los datos, entender bajo que principios, criterios se seleccionan, como el tipo de kernel (datos no lineales), la elección de la distancia depende del tipo (peso) de dato y si son de la misma escala o cuan dispersos son. Para los otros modelos, el criterio de impurezas, basado en la cantidad del repositorio de datos y la finalidad de que nuestros modelos sean rápidos y no complejos. Todo este análisis sirve como base para elegir el hiperparámetro más ideal, como se aprecia en las figuras 110 y 111.

Figura 110

Evaluación de la separabilidad lineal



Nota: *Se representa la distribución de los datos, 3 variables más importantes en función de sus etiquetas.*

Figura 111

Importancia de características entre Gini Y Entropía

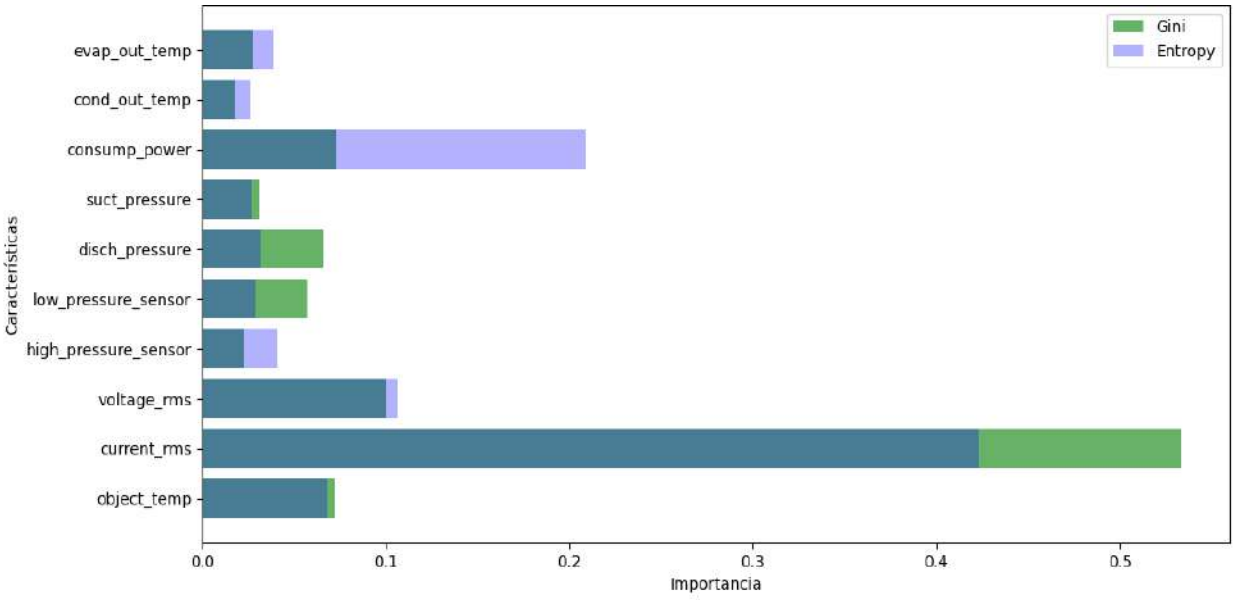
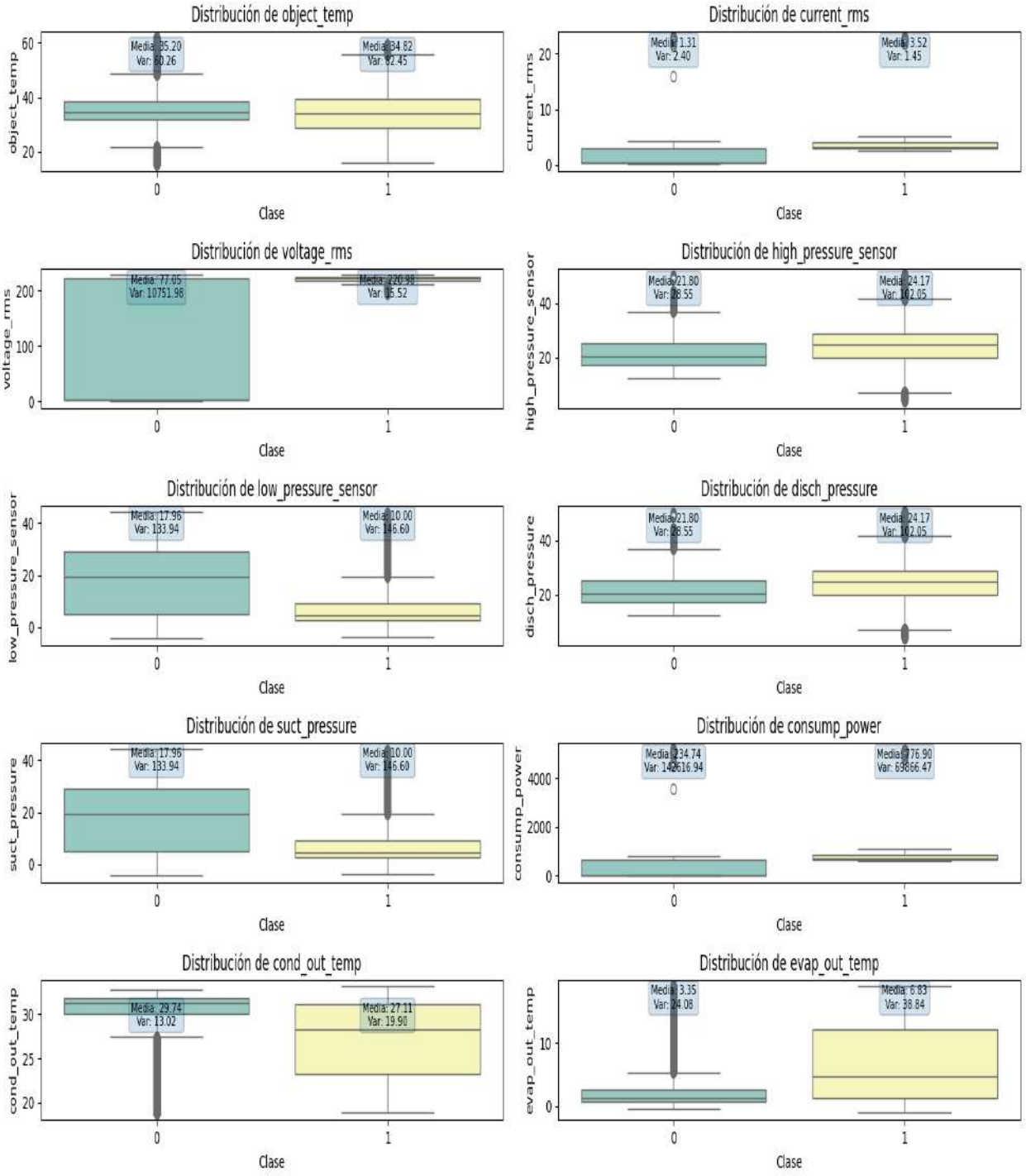


Figura 112

Análisis de la distribución de los datos de cada variable



Nota: *Se representa la distribución de los datos de las 10 variables, en función de la media y varianza.*

De manera intrínseca para entrenar el **algoritmo Random Forest**, mediante las funciones ampliamente desarrolladas y fundamentadas, por la comunidad que soporta la solución Scikit-Learn. Particularmente “fit” y “RandomForestClassifier”, esta función inicializa mediante sus parámetros las variables; “n_estimators” representa la cantidad de árboles que el modelo construye, “max_depth” controla cuántos niveles puede tener un árbol de decisión antes de detener el proceso de división, “min_samples_split” controla la cantidad mín de muestras que un nodo debe tener antes de que el árbol lo divida en nodos más pequeños y “min_samples_leaf” representa el último nodo después de que un árbol ya no puede dividirse más. Este modelo está fundamentado en la creación de múltiples árboles de decisión en el entrenamiento, la combinación de sus resultados para incrementar la precisión y reducir el riesgo de sobreajuste. Este enfoque se basa en la técnica de "bagging" (bootstrap aggregating), utiliza la aleatorización en la selección de muestras y datos de entrenamiento.

Este árbol, es una estructura que divide el repositorio de datos en subconjuntos fundamentados en ciertas características, mediante condiciones lógicas en cada nodo, se genera reglas de este tipo:

Si $X_1 < v_1$ Y $X_2 > v_2$, entonces $\hat{y}$= clase, donde X_1 y X_2 son características del conjunto de datos, y v_1 y v_2 son valores umbrales (Scikit learn, 2007-2024).

Cada decisión tree en el bosque se construyó en base de un subconjunto aleatorio del set de entrenamiento, seleccionados con reemplazo. Para realizar la segmentación del espacio necesitamos un criterio para decir que un corte es bueno, para eso utiliza la noción de impureza para cuantificar que tan mezcladas están las distintas clases, existen dos maneras, mediante la “Entropía” que se define.

$$Entropía = \sum_{i=1}^{C} - P_i . \log_2(P_i) \quad (27)$$

$$P_i = \frac{|\{x \in c, clase(x) = i\}|}{|c|} \quad (28)$$

Para esta evaluación, se eligió el criterio impureza de Gini, se fundamentó en maximizar la reducción de impureza, que se calcula como (Scikit learn, 2007-2024):

$$Gini = 1 - \sum_{i=1}^{C} {P_i}^2 \quad (29)$$

Donde P_i es la proporción de muestras de la clase i en el nodo actual y C es la cantidad total de clases (Scikit learn, 2007-2024).

Random Forest tiene varios parámetros que se debe ajustar para optimizar su rendimiento, la función de decisión para el modelo Random Forest se define como (Scikit learn, 2007-2024):

$$f(x) = mode\{ T_b(X)\}, b = 1,2, \dots, B \quad (30)$$

Donde $T_b(X)$ es la predicción del b-ésimo árbol de decisión y "n_estimators" = b, representa la cantidad de árboles de decisión que construye el modelo y está directamente relacionado con la función de decisión, como se aprecia en la ecuación (28). La predicción final del modelo es determinada al combinar las predicciones de múltiples árboles, donde la clase más votada es la elegida.

El modelo resultante seleccionó un total de "n_estimators" 450 estimadores (árboles de decisión) y un "max_depth" = 25, profundidad máxima del árbol, limita cuán profundamente puede dividirse un árbol. Controla cuántas veces el algoritmo evalúa la impureza de Gini para hacer divisiones adicionales en los nodos, lo que permitió una estabilidad entre el ajuste a los datos de entrenamiento y la generalización al conjunto de datos de prueba.

Adicionalmente, se ajustaron de manera óptima los parámetros de división mínima de muestras "min_samples_split" = 2, que determina cuántas muestras debe tener un nodo para dividirse. Controla la creación de nuevos nodos usando la función Gini y el mínimo de muestras en una hoja "min_samples_leaf" = 1, representa cuántas muestras tiene que haber en un nodo hoja, con el fin de controlar la profundidad y complejidad de los árboles.

Y para entrenar el **algoritmo SVM**, mediante las funciones ampliamente desarrolladas y fundamentadas, por la comunidad que soporta la solución Scikit-Learn. Particularmente “fit” y “SVC”, esta función inicializa mediante sus parámetros las variables; “C” se encarga de controlar el compromiso entre errores de detección y margen, “gamma” ajusta la influencia de la muestra de datos en el espacio transformado por el kernel e internamente el SVM busca un hiperplano óptimo que divida los datos de dos tipos diferentes, maximizando el margen entre ellas, este hiperplano se describe formalmente por.

$$W.X + b = 0 \tag{31}$$

Donde b el sesgo del modelo, que determina el desplazamiento del hiperplano, W representa el vector de pesos, que establece la pendiente del hiperplano y X el vector de características de la muestra.

Si el margen (M) es la separación entre el hiperplano y los puntos más proximos de cada clase, llamados vectores de soporte (Scikit learn, 2007-2024).

$$M = \frac{2}{\|w\|} \tag{32}$$

SVM busca maximizar este margen, lo que equivale a minimizar la norma de W, así.

$$Minimizar: \ \frac{1}{2}\|w\|^2 \tag{33}$$

Para manejar los casos en que los datos no son linealmente separables, se introduce un término de regularización en la función objetivo controlado por el parámetro “C” para permitir que algunos puntos de datos queden dentro del margen o sean clasificados incorrectamente (Scikit learn, 2007-2024).

$$Minimizar: \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \varepsilon_i \tag{34}$$

Donde $\|w\|^2$ es la magnitud del vector de pesos, que controla la amplitud del margen, ε_i es la variable de holgura que permiten que algunos puntos se encuentren dentro del margen o se clasifiquen incorrectamente y “C” es un parámetro de regularización que equilibra la amplitud del margen y la penalización por errores de clasificación. En este estudio, se fijó un valor de C = 500, lo que indica una

mayor penalización de los errores, favoreciendo modelos más precisos, aunque con márgenes de separación ligeramente menores.

Dado que los datos no presentaban una distribución linealmente separable en el estado original de características, se puede aplicar una transformación a un espacio de mayor dimensión utilizando un kernel, donde para el modelo SVM existen, el kernel polinomial (se utiliza cuando se espera que la relación entre las características sea de naturaleza polinómica).

$$K(X_i, X_j) = (\gamma(X_i . X_j) + r)^d \quad (35)$$

El kernel signoidal (es el menos usado).

$$K(X_i, X_j) = \tanh(\gamma(X_i . X_j) + r) \quad (36)$$

Donde r introduce un desplazamiento y d controla el grado de polinomio, parámetros ajustables.

Para este caso se utilizó el kernel Radial Basis Function (RBF), ya que suele ser una buena opción por defecto y es apropiado para una gran gama de problemas, descrito por la siguiente ecuación.

$$K(X_i, X_j) = \exp(-\gamma \| X_i - X_j \|^2) \quad (37)$$

Donde $K(X_i, X_j)$, es el valor del kernel entre las muestras X_i, X_j y "γ" = “gamma”, se define como el parámetro que controla la influencia de cada muestra de entrenamiento sobre el resto.

“Este kernel permitió transformar los datos a un espacio de mayor dimensión, donde sí fue posible encontrar una frontera de decisión lineal. Se seleccionó un valor de γ=0.04, lo que implicó que cada muestra tenía una influencia moderada en la determinación del margen, capturando adecuadamente la complejidad de los datos sin incurrir en sobreajuste” (Scikit learn, 2007-2024).

Finalmente, la función de decisión del modelo SVM, se basa en los vectores de soporte que son los puntos que definen el margen óptimo. Para una nueva muestra x, el SVM clasifica la muestra según el signo de la siguiente función, si el valor de la función es positivo, la muestra se clasifica como una clase; si es negativo, se clasifica a lo contrario (Scikit learn, 2007-2024):

$$f(x) = sign(\sum_{i=1}^{n} \propto_i y_i K(X_i, X) + b) \quad (38)$$

Donde $\propto_i$son los coeficientes de Lagrange asociados a los vectores de soporte, y_i son las etiquetas de clase (1 o 0), $K(X_i, X)$ es el valor del kernel entre la muestra X y un vector de soporte X_i, y b es el sesgo. La precisión del modelo se evaluó a través del criterio de exactitud (accuracy), alcanzando un valor alto de 93%.

En cambio, el modelo de clasificación **Decision Tree** fue entrenado utilizando las funciones ampliamente desarrolladas y fundamentadas por la comunidad de Scikit-Learn. Específicamente, la función fit y DecisionTreeClassifier son componentes fundamentales en la implementación. En este modelo, varios hiperparámetros sirven para el ajuste del rendimiento y la interpretación de los resultados:

1. Criterio de División: Para cada nodo, el modelo evalúa la calidad de la división a través del parámetro criterion, que puede utilizar tanto la *entropía* como la *impureza de Gini*. En este caso, se seleccionó la *impureza de Gini* para maximizar la reducción de impureza en cada nodo, permitiendo que el árbol construya divisiones significativas en el espacio de características. La fórmula de la impureza de Gini se expresa como (Scikit learn, 2007-2024):

$$Gini = 1 - \sum_{i=1}^{C} {P_i}^2 \quad (39)$$

Donde P_i es la relación de muestras pertenecientes a la clase i en el nodo actual, y C representa el número de clases en el repositorio de datos. Este cálculo permite reducir la heterogeneidad en los nodos, mejorando la precisión del modelo al minimizar la mezcla entre clases (Scikit learn, 2007-2024).

2. Profundidad Máxima (max_depth): Controla cuántas divisiones puede hacer un árbol antes de detenerse. Esto es crucial para evitar sobreajuste (overfitting), ya que árboles muy profundos pueden ajustarse en demás a los datos de entrenamiento. En este estudio, se estableció una profundidad máxima de max_depth =25, Esto permitió alcanzar un balance óptimo entre el ajuste a los datos de entrenamiento y la habilidad del modelo para generalizar.

3. División Mínima de Muestras (min_samples_split): Este parámetro regula cuántas muestras debe tener un nodo antes de poder dividirse, lo cual controla la creación de nuevos nodos mediante el criterio de impureza seleccionado. Para esta evaluación, se seleccionó un valor de min_samples_split = 2, lo que permitió que el modelo genere divisiones cuando haya al menos dos muestras en un nodo, asegurando la reducción de impureza sin una segmentación excesiva.
4. Muestras en Nodo Hoja (min_samples_leaf): Este parámetro especifica la cantidad mínima de muestras que un nodo hoja debe contener, previniendo que se creen nodos con muy pocas muestras. Se seleccionó un valor de min_samples_leaf = 1, asegurando que los nodos hoja contengan información relevante sin fragmentar excesivamente los datos.

Este modelo de Decision Tree se basa en separar el set de datos en subconjuntos más homogéneos, utilizando decisiones basadas en umbrales de características. En el caso específico, el modelo aplica reglas de decisión en cada nodo, tales como (Scikit learn, 2007-2024):

Si $X_1 < v_1$ Y $X_2 > v_2$, entonces $\hat{y}$= clase, donde X_1 y X_2 son características del conjunto de datos, v_1 y v_2 son valores umbrales específicos (Scikit learn, 2007-2024).

Al evaluar y seleccionar las divisiones en función del criterio de impureza de Gini, el modelo logra construir un árbol de decisiones que reduce al máximo la heterogeneidad en cada nodo, lo que permite obtener predicciones precisas en el conjunto de pruebas (Scikit learn, 2007-2024).

La predicción final del modelo de árbol de decisión se define por la ruta de decisiones en el árbol desde la raíz hasta el nodo hoja en función de la muestra de entrada. Esta metodología permitió alcanzar una exactitud alta del 93 % en el conjunto de prueba, lo cual valida la eficacia de la selección de hiperparámetros y estructura del modelo de árbol de decisión.

Para el caso del modelo **K-Nearest Neighbors** (KNN).

Implícitamente para entrenar el algoritmo KNN, mediante las funciones ampliamente desarrolladas y fundamentadas, por la comunidad que soporta la solución Scikit-Learn. Particularmente "fit" y "KneighborsClassifier", esta función inicializa mediante sus parámetros las variables; "n_neighbors" que representa el valor de los k datos de entrenamiento más cercanos, "weights" representa las dos principales estrategias para predecir para un peso uniforme o por la distancia del vecino y "metric" representa metodologías para determinar la separación entre el punto de prueba y los de entrenamiento, pero su elección depende del tipo de datos. Todo se fundamente en la idea, que una muestra debe clasificarse de manera similar a sus vecinas más cercanas en el espacio de características. Para ello, se determina la distancia entre la muestra de interés y las demás muestras del conjunto de datos. Se definen mediante las ecuaciones distancia Euclidiana, distancia Manhattan y distancia Minkowski respectivamente (Scikit learn, 2007-2024).

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{n} (X_{ik} - X_{jk})^2} \quad (40)$$

$$d(X_i, X_j) = \sum_{k=1}^{n} |X_{ik} - X_{jk}| \quad (41)$$

$$d(X_i, X_j) = (\sum_{k=1}^{n} |X_{ik} - X_{jk}|^p)^{1/p} \quad (42)$$

En este estudio, se seleccionó la distancia de Manhattan (39) como métrica de distancia óptima, donde X_i, X_j son los vectores de características de las muestras i y j, respectivamente, y n es el número de características (Scikit learn, 2007-2024).

El número óptimo de vecinos (k) se estableció en k=9, lo que significa que la clasificación de una nueva muestra se determinó considerando las 9 muestras más cercanas. Ya una vez que se determinó el vecindario para poder hacer la predicción existen dos estrategias, la instancia nueva la podemos predecir

cómo la etiqueta más frecuente en el vecindario donde todos votan con peso 1 – "uniform weights", pero en este caso se elige la estrategia "distance weights" para determinar la clase final, la cual se define como:

$$\mathrm{W_i} = \frac{1}{d(X_i, X)} \quad (43)$$

Donde W_i es el peso asignado a la muestra i, y $d(X_i, X)$ es la distancia de Manhattan entre la muestra i y la de interés X (Scikit learn, 2007-2024).

Para decidir a qué clase pertenece esta nueva muestra, el modelo "pregunta" a sus k vecinos cercanos y toma una decisión basada en una votación ponderada por la distancia.

La función de decisión del modelo KNN se caracteriza como:

$$f(x) = \arg max_c \sum_{i \in N_k}^{n} W_i . 1(\ y_i = \mathrm{c}) \quad (44)$$

Donde N_k es el conjunto de los k vecinos más próximos, W_i es el peso de cada vecino, y_i es la clase de cada vecino y $1(y_i{=}c)$ es la relación indicadora que toma el valor 1 si la clase y_i pertenece a la clase c, y 0 en caso contrario. Esta parte de la fórmula $\sum_{i \in N_k}^{n} W_i . 1(\ y_i = \mathrm{c})$ representa la sumatoria ponderada de los votos de los vecinos para la clase c, al final la clase que obtiene la mayor suma de votos ponderados será la que se asigna a la nueva muestra. Esta parte de la fórmula $arg\ max_c$ significa seleccionar la clase c que tiene la mayor suma de votos ponderados, en otras palabras, el modelo asigna la nueva muestra a la clase que recibió más votos de los vecinos cercanos, teniendo en cuenta la importancia de cada vecino según su distancia (Scikit learn, 2007-2024).

Para el caso **Gradient Boosting** (GB), el principio es elaborar el modelo de forma secuencial, con que cada nuevo árbol se ajusta a los errores residuales (diferencias entre las predicciones anteriores y las verdaderas etiquetas). El objetivo es corregir progresivamente los errores del modelo anterior, utilizando técnicas de descenso por gradiente para optimizar la función de pérdida. Entonces el modelo calcula la predicción final mediante la combinación lineal de las predicciones de todos los árboles construidos (Scikit learn, 2007-2024).

$$F_m(x) = F_{m-1}(x) + \eta . h_m(x) \quad (45)$$

donde:

- $F_m(x)$ es la predicción final del modelo tras el m-ésimo paso.
- $F_{m-1}(x)$ es la predicción acumulada hasta el paso anterior.
- η learning_rate, regula el impacto de cada árbol adicional.
- $h_m(x)$ representa la función de predicción del árbol m-ésimo.

Después, para ajustar el modelo, se minimiza la función de pérdida (error), utilizando la técnica de descenso por gradiente. En cada iteración, el modelo ajusta un árbol que predice la dirección de corrección necesaria para reducir el error (Scikit learn, 2007-2024):

$$h_m(x) = -\nabla \mathrm{L}\big(y,\ F_{m-1}(x)\big) \quad (46)$$

donde:

- $\mathrm{L}\big(y,\ F_{m-1}(x)\big)$ es la función de pérdida a minimizar, calculada en función de las etiquetas reales y y las predicciones acumuladas $F_{m-1}(x)$.
- ∇ representa el gradiente de la pérdida, que indica cómo ajustar el modelo para mejorar la precisión.

El modelo total es la suma ponderada de cada árbol de decisión, la predicción final se determina según la combinación lineal de las salidas de cada árbol individual, ajustado por la tasa de aprendizaje.

Por último, para el caso **Naive Bayes** (NB), el principio central es aplicar el Teorema de Bayes, bajo el supuesto de que todas las características son independientes entre sí, dado el resultado de la clase. El supuesto de independencia "ingenuo", simplifica significativamente los cálculos, permitiendo una clasificación rápida y efectiva (Scikit learn, 2007-2024).

Para calcular la probabilidad condicional de que una muestra x pertenezca a una clase $\mathrm{C_k}$:

$$P(\mathrm{C_k} \mid X) = \frac{P(\mathrm{x} \mid \mathrm{C_k}).\, P(\mathrm{C_k})}{P(x)} \quad (47)$$

$P(\mathrm{C_k} \mid X)$, probabilidad posterior de que X pertenezca a la clase $\mathrm{C_k}$.

$P(\mathrm{Ck})$, probabilidad a priori de la clase Ck .

$P(x)$, probabilidad de los datos observados x.

$P(\mathrm{x} \mid \mathrm{Ck})$, probabilidad de observar los datos X, dado que pertenecen a la clase Ck (probabilidad condicional).

Naive Bayes cuenta con que todas las características X= (x1, x2, ..., xn) son independientes entre sí, lo que simplifica la ecuación (Scikit learn, 2007-2024):

$$P(\mathrm{x} \mid \mathrm{Ck}) = \prod_{i=1}^{n} P(\mathrm{xi} \mid \mathrm{Ck}) \quad (48)$$

En la práctica, se omite P(X) porque es una constante para todas las clases y no afecta la clasificación final.

Específicamente, La probabilidad condicional para una característica xi dado que pertenece a la clase Ck , de Gaussian Naive Bayes, se calcula como (Scikit learn, 2007-2024):

$$P(\mathrm{xi} \mid \mathrm{Ck}) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(\mathrm{xi} - \mu\mathrm{k})^2}{2\sigma_k^2}\right) \quad (49)$$

Donde $\mu\mathrm{k}$ es la media y σ_k^2 es la varianza de la característica en la clase Ck .

Para identificar específicamente a qué clase pertenece una muestra, el modelo selecciona la clase con la mayor probabilidad posterior P(C|X). Esto se basa en la regla de decisión de máxima probabilidad, expresada matemáticamente mediante (Scikit learn, 2007-2024):

$$\hat{\mathrm{C}} = \arg max_{ck}\, P(\mathrm{Ck}) \prod_{i=1}^{n} P(\mathrm{xi} \mid \mathrm{Ck}) \quad (50)$$

donde:

- $\hat{\mathrm{C}}$ es la clase predicha.
- $P(\mathrm{Ck})$ es la probabilidad previa de la clase.
- $P(\mathrm{xi} \mid \mathrm{Ck})$ es la probabilidad de cada característica xi , dado que la clase es Ck.
- n es el número de características.

6.3 Elección del Modelo de Detección

En esta sección, se examinaron diferentes métricas de evaluación para verificar los resultados alcanzados a lo largo de las fases de entrenamiento, validación y prueba de los modelos elegidos. Con el propósito de seleccionar el mejor modelo clasificador, principalmente se usó la **accuracy**, porque da un entendimiento general del rendimiento total del modelo, representa el porcentaje de predicciones correctas hechas por el modelo en relación con el total de predicciones evaluadas, esta es una métrica muy utilizada en distintos antecedentes. Además, es importante buscar evitar los falsos positivos, por tanto, los niveles de precisión y especificidad también son importantes y deben ser altos.

Escoger el modelo más adecuado fue fundamental para garantizar la exactitud y confiabilidad. Las métricas indicaron que el modelo RF es el más efectivo, ya que puede gestionar datos con alta dimensionalidad. El modelo SVM al igual que el DT mostraron ser una opción viable, aunque con un rendimiento algo menor al de RF. Por último, se tiene a los modelos GB, KNN y NB con las métricas de evaluación más bajas en ese orden. Como se representó en las siguientes tablas 14, 15 y 16.

Tabla 14

Comparativa del nivel de Accuracy en la etapa de entrenamiento

Comparativo en % - Etapa Entrenamiento					
Accuracy SVM	**Accuracy KNN**	**Accuracy RF**	**Accuracy GB**	**Accuracy DT**	**Accuracy NB**
0.96	0.83	1	0.93	1	0.75

Tabla 15

Comparativa del nivel de Accuracy en la etapa de validación

Comparativo en % - Etapa Validación					
Accuracy SVM	**Accuracy KNN**	**Accuracy RF**	**Accuracy BG**	**Accuracy DT**	**Accuracy NB**
0.9505	0.8810	0.9528	0.9494	0.9312	0.7486

Tabla 16

Comparativa del nivel de Accuracy en la etapa de testeo

Comparativo en % - Etapa testeo					
Accuracy SVM	**Accuracy KNN**	**Accuracy RF**	**Accuracy BG**	**Accuracy DT**	**Accuracy NB**
0.93	0.83	0.96	0.91	0.93	0.77

En esta etapa se analizó el rendimiento con matrices de confusión. En la figura 113 se ve, que el modelo SVM presento un número elevado de FP, el modelo DT presenta aún más altos valores de FP, mientras que el modelo RF tiene indicadores más equilibrados y aceptables. En la figura 114 se ve que los modelos NB y KNN presentan la peor evaluación de rendimiento, con el número de FP y FN muy altos.

Figura 113

Selección de matrices de confusión de los tres mejores modelos de clasificación

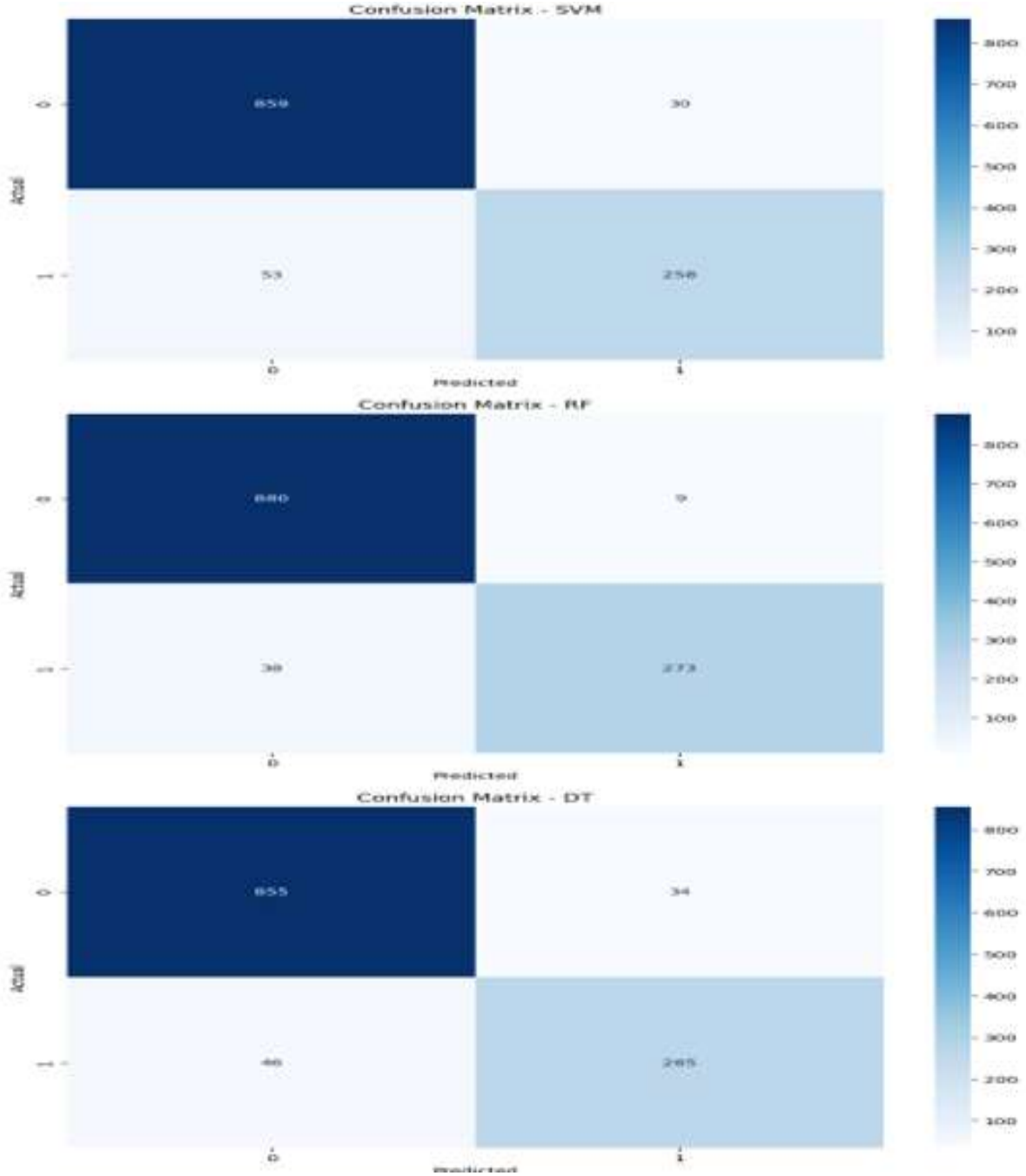
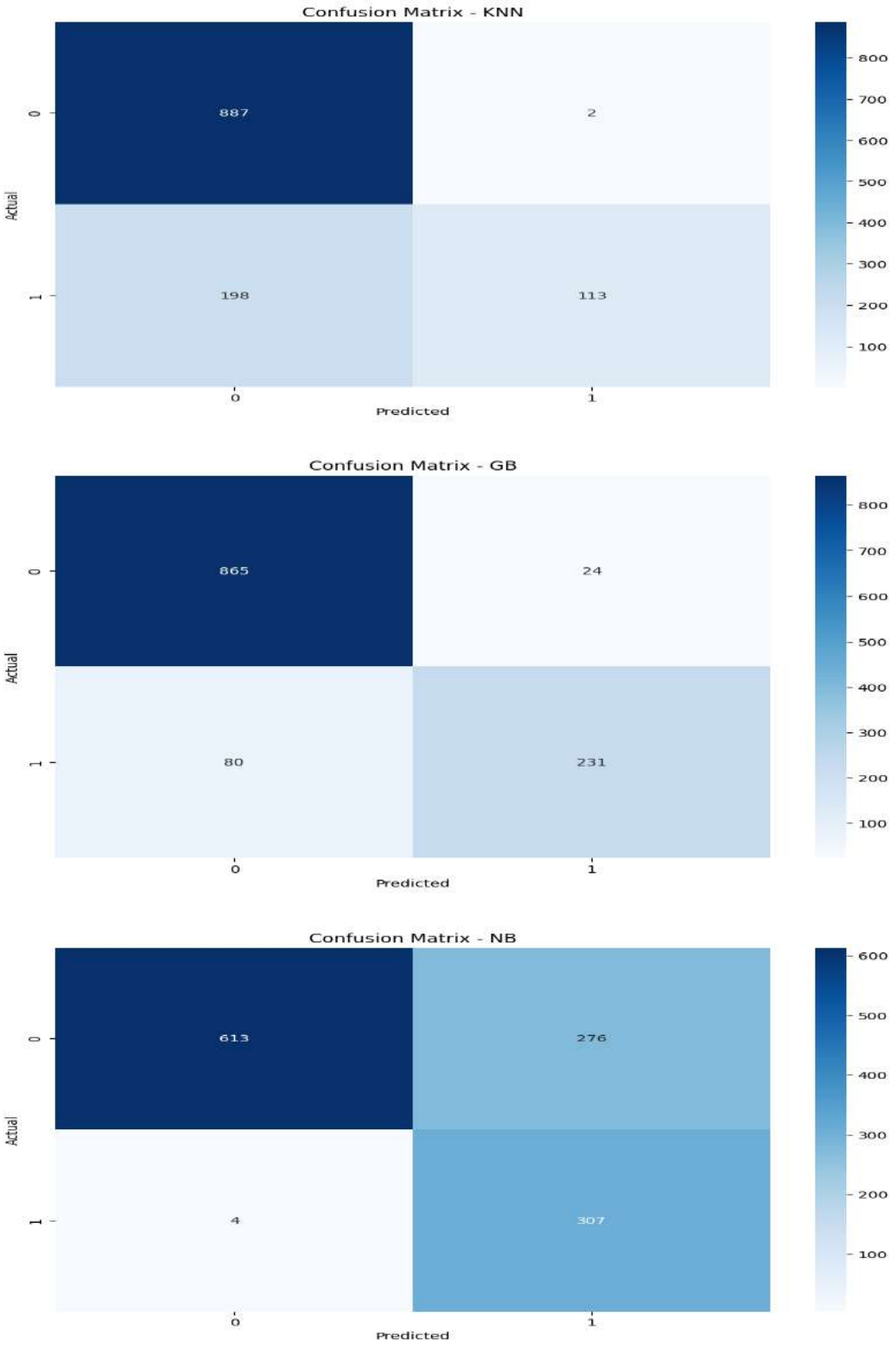


Figura 114

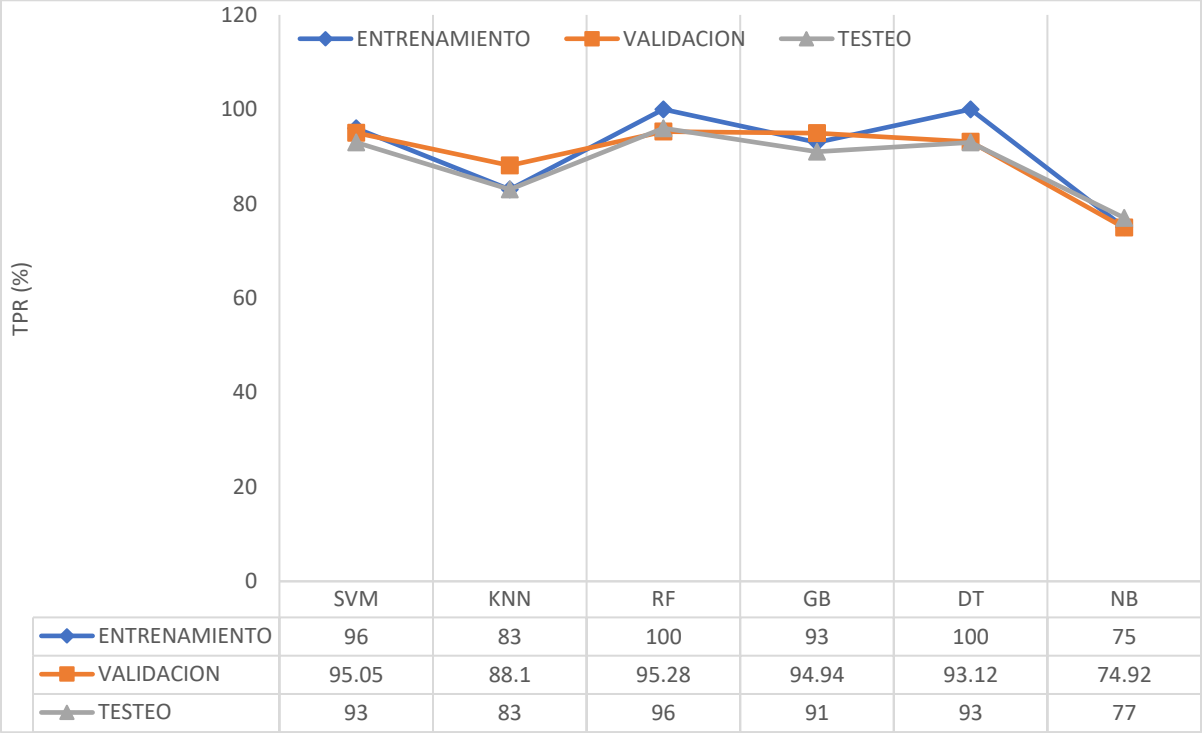
Selección de matrices de confusión de los tres peores modelos de clasificación



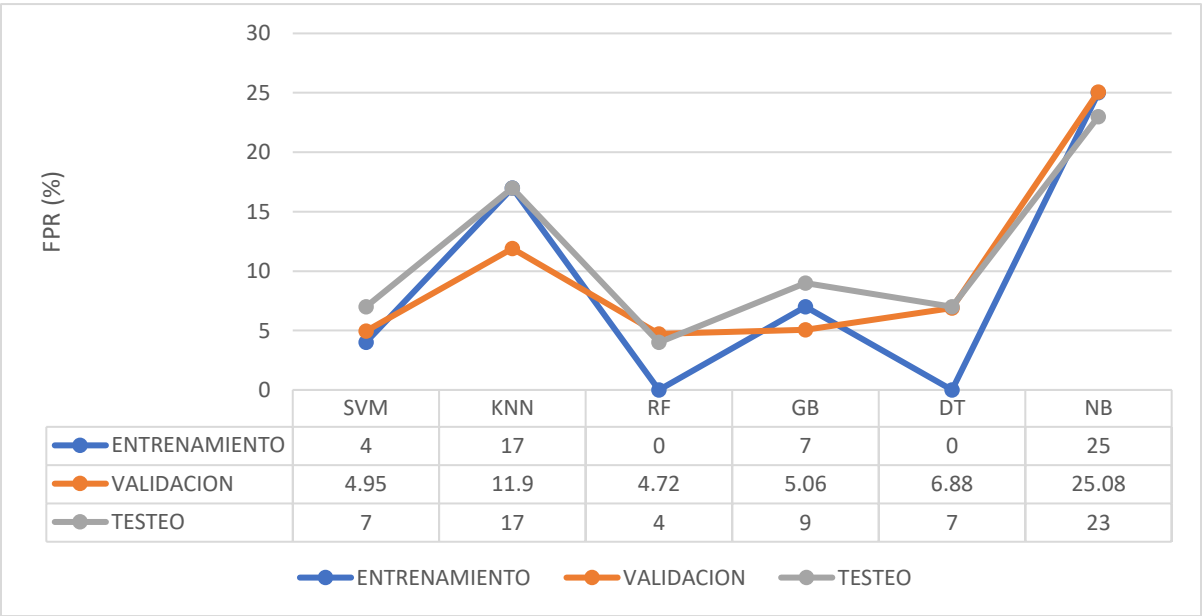
En las figuras 115 y 116, se representó el nivel de verdaderos positivos y falsos positivos, para cada modelo y en las tres etapas, se aprecia niveles altos en los modelos SVM, RF, DT y el modelo NB presenta los niveles más bajos, estas métricas señalan la capacidad de diferenciar cada falla.

Figura 115

Valores de TPR y FPR para cada método de clasificación



	SVM	KNN	RF	GB	DT	NB
ENTRENAMIENTO	96	83	100	93	100	75
VALIDACION	95.05	88.1	95.28	94.94	93.12	74.92
TESTEO	93	83	96	91	93	77

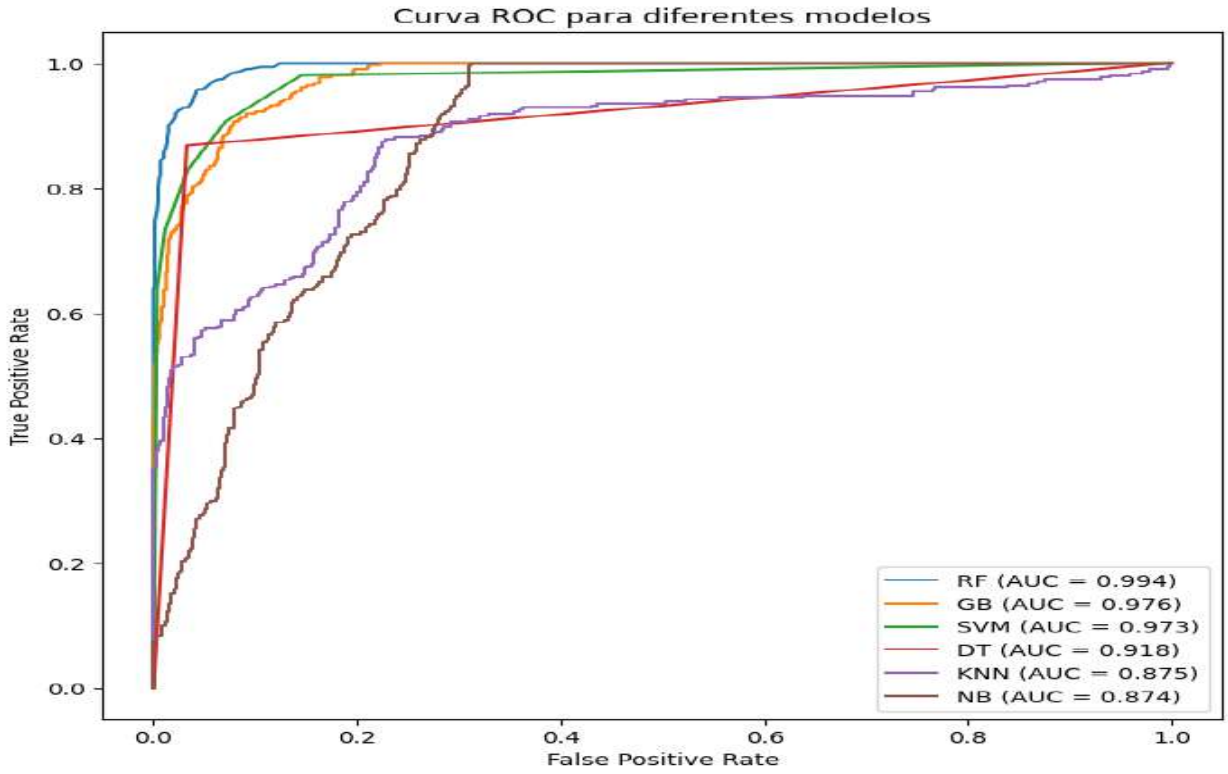


	SVM	KNN	RF	GB	DT	NB
ENTRENAMIENTO	4	17	0	7	0	25
VALIDACION	4.95	11.9	4.72	5.06	6.88	25.08
TESTEO	7	17	4	9	7	23

Además, con la herramienta grafica curva ROC se visualizó qué tan bien el modelo clasifica correctamente las fallas y qué cantidad de "falsos positivos" está generando, mediante la métrica AUC, donde el modelo RF se resalta como el mejor y el modelo NB presenta el valor más bajo.

Figura 116

Valores de TPR y FPR para cada método de clasificación



6.4 Pruebas y Resultados de los Modelos Clasificación en Tiempo Real

6.4.1 Pruebas del Sistema en Tiempo Real

Por último, se realizaron las pruebas del sistema FDD en el sistema piloto de aire acondicionado de precisión, en tiempo real, que son muy similares a las pruebas anteriores, para ello, se empleó el modelo que tiene el mayor nivel de acurracy, según la evaluación de las tablas comparativas desarrolladas en la sección anterior. Para fines simples, previo a comenzar con el estudio de las pruebas, es importante rememorar que el sistema de detección genera una etiqueta de clasificación cada segundo (1 para evento fallido y 0 para evento normal). No obstante, se analizó cada etiqueta individualmente.

6.4.1.1 Configuración del Sistema en el AAP para su Evaluación en Tiempo Real

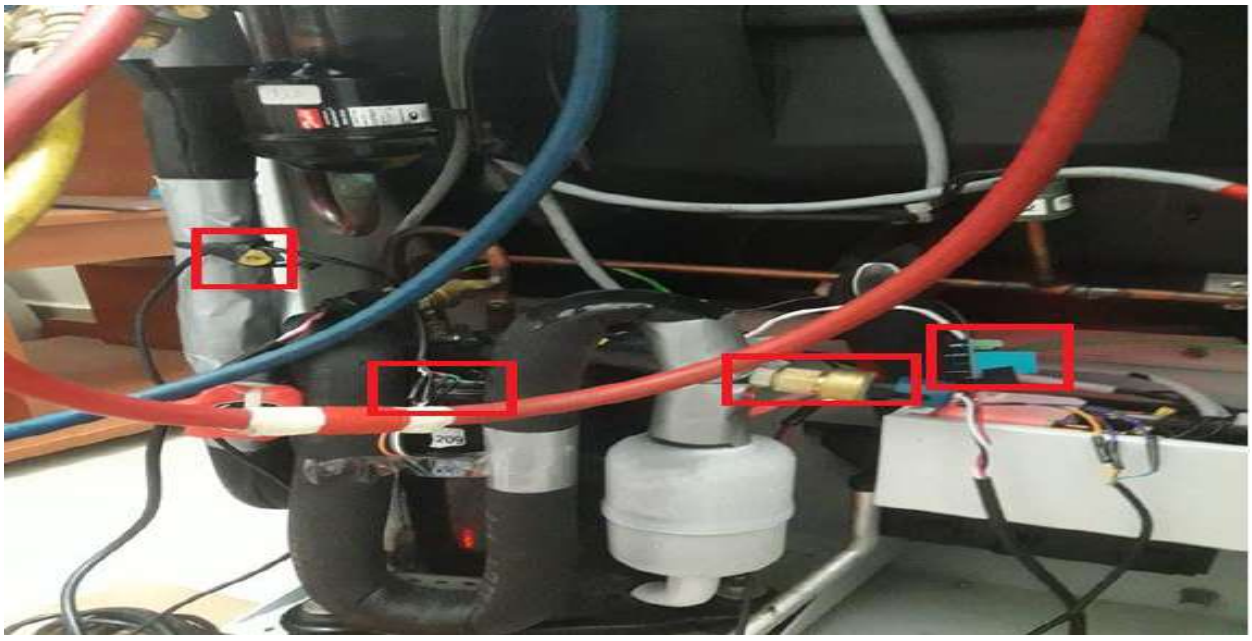
Como se explicó en el capítulo V, ya se implementó toda la parte de instrumentación y detección que detecta y diagnostica las fallas, sobre el aire acondicionado real en evaluación y debido a que luego del procedimiento de la obtención del repositorio de datos, el sistema AAP quedo con el nivel de refrigerante inferior al recomendado para su funcionamiento normal, es necesario volver a cargar el elemento refrigerante, siguiendo los pasos desarrollados en el capítulo V. Considerar que, en esta configuración, se evalúa su capacidad de detectar y diagnosticar fallas en un contexto de tiempo real y una prueba de validación simulando su funcionamiento para condiciones reales.

6.4.1.2 Pruebas de Funcionamiento del Sistema de Diagnóstico y Detección

En este sentido, se hicieron las pruebas cubriendo parte de cinco días, desde las 9:00 AM hasta las 22:00 PM, en este intervalo de prueba se obtuvo una base de datos para los problemas en evaluación. Donde en estas nuevas bases de datos, se insertaron las etiquetas predichas en tiempo real por el modelo seleccionado, de acuerdo con la metodología expresada en la sección, 5.4 Guía de Adquisición de la Base de Datos, similar a los procedimientos referenciados, en las figuras 59, 60, 61, 62, 63, para acondicionar los eventos de falla evaluado en este proyecto, como la representación visual de la figura 117.

Figura 117

Ejemplo de acondicionamiento para prueba de evento RO



También se hizo una prueba demostrativa de validación, de forma similar al funcionamiento real del aire acondicionado de precisión, donde se integró un sistema que genera calor de manera continua en el ambiente evaluado, con el modelo de clasificación que presenta mejores resultados de evaluación, permitiendo así una detección objetiva de los eventos de falla registrados en ese periodo evaluado (ciclo de funcionamiento), Como se evidencia en la representación fotográfica de la figura 118.

Figura 118

Prueba del sistema AAP con un sistema que genera calor de forma continua

6.4.1.3 Resultados de las Pruebas del Sistema en Tiempo Real

Son obtenidos de la construcción de un nuevo set de datos para cada evento de falla evaluado. Pero considerando el valor de la detección que determina el modelo seleccionado. Que en el Anexo 41 *Ejemplo de análisis para eventos clasificados por el sistema de detección (RF),* se representa un fragmento del proceso de análisis de los resultados de las pruebas para el modelo RF, donde se detalla el análisis de los eventos, constatar que en esta tabla esta la columna etiqueta_predicha, insertada de forma automática por el modelo de clasificación RF y su etiqueta real, además están las columnas

”FalsoPositivo”, ” FalsoNegativo”, “VerdaderoPositivo” y ”VerdaderoNegativo” que sirven para obtener el performance de los modelos de detección de fallas, mediante las métricas de evaluación, resultando en la matriz de confusión que se expresa en la tabla 17:

Tabla 17

Matriz de confusión para el modelo RF

	Matriz de Confusión				Accuracy del modelo	Precision del modelo	Sensitivity del modelo (TPR)	Specificity del modelo	FPR	Total eventos
	FP	FN	VP	VN						
Modelo RF	12	54	244	1090	0.9528	0.9531	0.8187	0.9891	0.0108	1400

Para la prueba demostrativa propuesta en la sección 6.4.1.2, (en un entorno en el que el sistema detecta las fallas del AAP, con un sistema que genera calor de forma continua), siguiendo la misma línea de análisis presentada en el Anexo 41 y la tabla 17 se procede a examinar los resultados obtenidos. Conforme se aprecia en el Anexo 42 Ejemplo de análisis eventos clasificados por el sistema de detección (P-V) y la tabla Matriz de Confusión Prueba Validación 18.

Tabla 18

Matriz de confusión para Prueba de Validación

	Matriz de Confusión				Accuracy del modelo	Precision del modelo	Sensitivity del modelo (TPR)	Specificity del modelo	FPR	Total eventos
	FP	FN	VP	VN						
Modelo P-V	40	71	1051	544	0.9349	0.9633	0.9367	0.9315	0.068	1706

6.5 Resultados Generales del Sistema

Para obtener el resultado global del sistema, se realizó un análisis y comparación a través de cuadros resumen que recopilan la matriz de confusión del modelo más preciso seleccionado y el análisis del mejor modelo seleccionado, en un entorno similar al trabajo real.

Tabla 19

Comparativa de matrices de confusión

	Matriz de Confusión				Accuracy del modelo	Precision del modelo	Sensitivity del modelo (TPR)	Specificity del modelo	FPR	Total eventos
	FP	FN	VP	VN						
Modelo RF	12	54	244	1090	0.9528	0.9531	0.8187	0.9891	0.0108	1400
Modelo P-V	40	71	1051	544	0.9349	0.9633	0.9367	0.9315	0.068	1706

Hasta este punto ya se tiene los resultados individuales de cada modelo evaluado y de las tablas obtenidas en la sección 6.3 Elección del Modelo de Detección, se dedujo que el modelo más óptimo de detección de fallas, es el modelo RF (Random Forest), con un nivel de accuracy, óptimo de 96% en condición diferida, gracias a la tabla 19 Comparativa de matrices de confusión, y se puede observar sus métricas de evaluación del mejor modelo, con un nivel de accuracy, de 95.28% en tiempo real.

Tabla 20

Métricas de evaluación del modelo más óptimo para el sistema AAP

Prueba de Validación (Random Forest)	Condición en tiempo real
Accuracy	95.28%
Precision	95.31%
Sensitivity (TPR)	81.87%
Specificity	98.91

La tabla 20 expresa los resultados alcanzados del modelo Random Forest, para la detección en tiempo real, logrando una exactitud global del 95.28% que clasifica las instancias. Además, su alta precisión del 95.31% señala que, cuando predice una clase positiva, tiene una probabilidad muy alta de ser correcta, minimizando los falsos positivos. La sensibilidad, con un valor del 81.87%, cuantifica la habilidad para detectar los verdaderos positivos (es decir, las fallas reales), reduciendo los falsos negativos. La especificidad, que alcanza el 98.91%, refleja la capacidad para señalar correctamente los verdaderos negativos (los eventos sin fallas), evitando falsos positivos, este equilibrio entre sensibilidad y especificidad evidencia la efectividad para la detección de fallas en tiempo real.

Capítulo VII

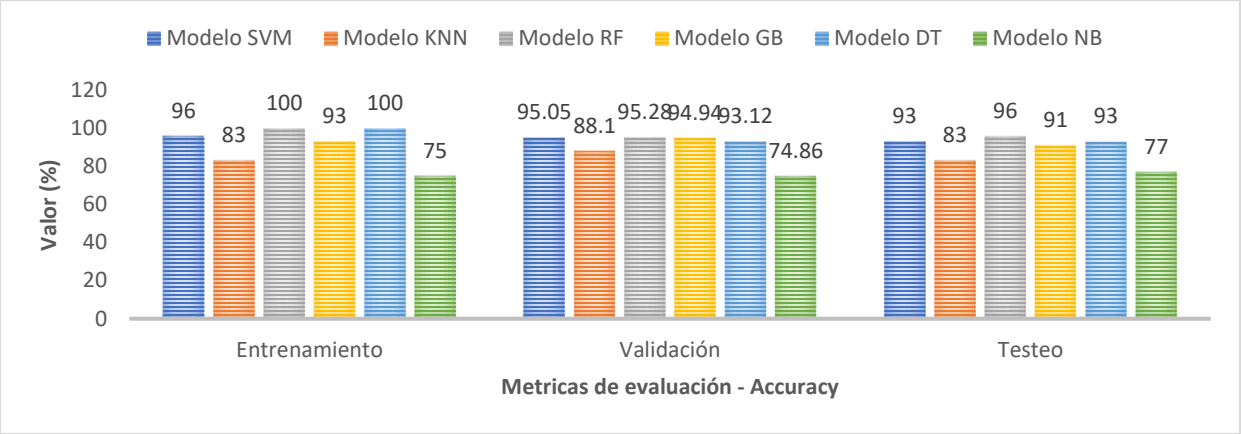
Discusión

Esta sección es crucial, ya que facilita la interpretación y contextualización de los resultados determinados, conectándolos con los objetivos establecidos y los antecedentes existentes. Dentro del proyecto, la discusión cobra una importancia particular al evaluar la efectividad de los modelos de aprendizaje implementados (KNN, SVM, RF, BG, DT y NB) y el rendimiento del subsistema de adquisición (instrumentación) seleccionado. Este capítulo no solo examina la exactitud y fiabilidad del sistema desarrollado, sino que también identifica sus fortalezas, limitaciones y posibles áreas de mejora, ofreciendo una visión completa del impacto y la relevancia del sistema en la clasificación de fallas.

7.1 Descripción de los hallazgos más significativos

En este proyecto, se llevó a cabo una implementación avanzada en un sistema modular de detección, compuesto por un procesador Raspberry Pi 4B y un dispositivo de adquisición Arduino Nano. Este sistema, junto con los elementos instrumentales como el sensor de corriente ACS712-30A, el de temperatura MLX90614, el de voltaje ZMPT, de temperatura DS18B20 y el de presión de la familia SPKT0043P, demuestra un enfoque técnico y tecnológico innovador para clasificar en tiempo real los eventos de fallas del sistema de aire acondicionado de precisión (AAP). A través de un análisis comparativo de modelos de aprendizaje basado en el criterio de accuracy, se determinó que el modelo Support Vector Machine (SVM) logró un 93%, el modelo Desicion Tree (DT) alcanzó un 93%, el modelo Gradient Boosting (GB) alcanzó un 91%, K-Nearest Neighbors (KNN) alcanzó un 83%, el modelo Naive Bayes (NB) alcanzó un 77% y el modelo Random Forest (RF) destacó con un 96% de exactitud en pruebas diferidas y un 95.28% en tiempo real. Este trabajo propone una estrategia de detección y diagnóstico de fallas (FDD) basada en datos, para cinco tipos de fallas típicas y la elección del hardware subraya la relevancia de contar con elementos de procesamiento adecuados para el sistema de detección.

Figura 119
Comparación de los modelos detección de fallas



7.2 Limitaciones del estudio

Durante la fase de diseño de este estudio, se seleccionaron los elementos más óptimos, teniendo en cuenta varios criterios limitantes, como factores económicos, técnicos, de tiempo y la disponibilidad en el mercado, que son cruciales en el diseño de ingeniería. Además, se identificó limitaciones como: la falta de un estándar de procedimientos para el preprocesamiento y la generación del set de datos, sobre el sistema Aire Acondicionado de Precisión, modelo SK3328.500 (enclosure top therm, cooling unit).

7.3 Comparación critica con la literatura existente

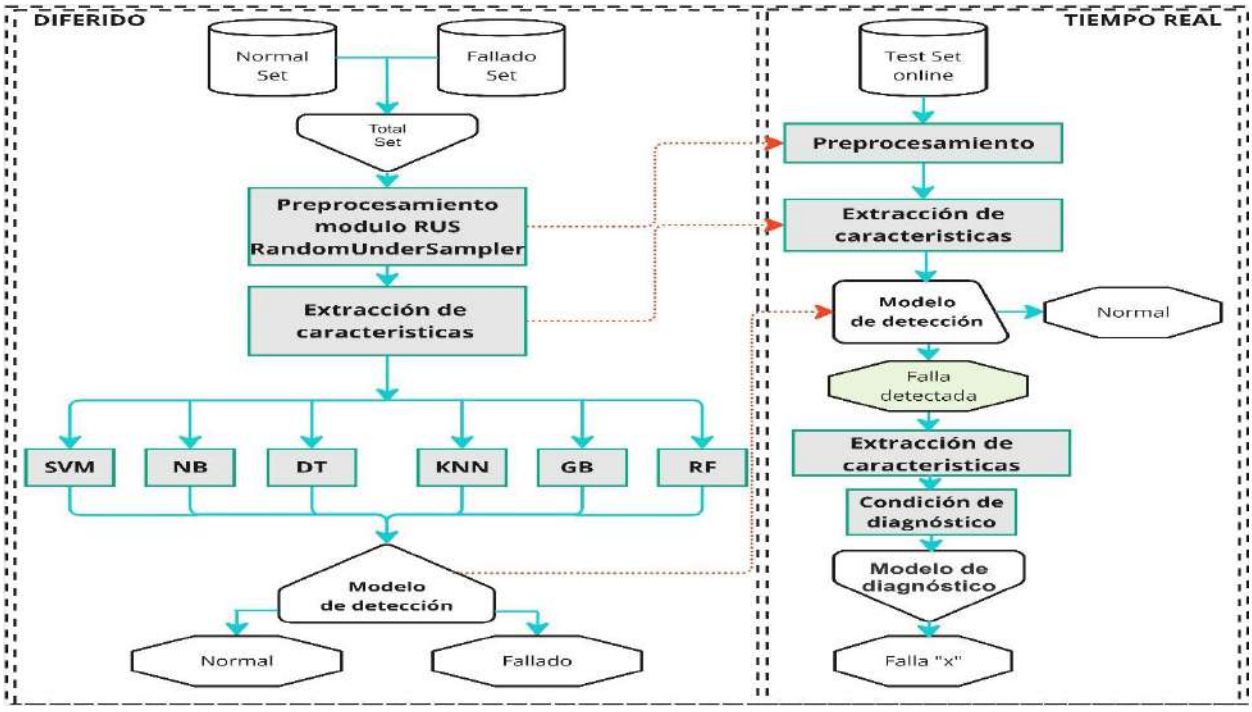
En el trabajo de investigación realizada por (Ebrahimifakhar, Kabirikopaei, & Yu, Data-driven fault detection and diagnosis for packaged rooftop units using statistical machine learning classification methods, 2020), en Nebraska-Lincoln University, Omaha, United States, basada en datos para RoofTop Units utilizando métodos de aprendizaje automático estadístico. La estrategia propuesta aborda la tarea FDD como un tema de clasificación de múlticlase. Para evaluar siete tipos de fallas definidos así: compressor valvem leakage(VL), refrigerant undercharge(UC), refrigerant overcharge(OC), restriction line(LL), condenser airflow reduction(CA), evaporador airflow reduction(EA) y non-condensable gas(NC). Donde se evaluó mediante los siguientes modelos quadratic discriminant analysis (QDA), bagging (BA), XGBoost (XGB), linear discriminant analysis (LDA), k-nearest neightbors (KNN), random forest (RF), adaBoost (AD), support vector machine (SVM) y regresión lógica (LR), basado variables como temperatura, presión y potencia del compresor. Finalmente, se empleó un método supervisado basado en el modelo

support vector machine, logrando un accuracy de 96.2%. Mediante un diagrama comparativo, se representó que "el rendimiento general del modelo SVM, ajustado con los parámetros óptimos, fue el mejor, mientras que el modelo LDA tiene el indice de precisión más bajo de 76.2%". Señalo que las fallas poco frecuentes son complejas de estudiar debido a la limitada cantidad de información accesible. En este estudio, se utiliza la técnica de sobremuestreo para producir muestras de clase minoritaria con el fin de equilibrar el set de datos, también se analiza la importancia relativa de las variables de entrada.

Para este estudio, se propuso un marco de estrategia FDD, como en la figura 120, la finalidad es detectar indicios de falla, en caso sí el sistema detecta falla, se procede a diagnosticar que tipo de falla es, mediante reglas de diagnóstico, basadas en la tabla 11. Inicialmente esta la etapa de base datos (con el conjunto fallado y normal), seguidamente hay una etapa de preprocesamiento de los datos (modulo RUS), luego una etapa de obtención de características para mediante la evaluación de seis algoritmos de clasificación, se seleccione el mejor modelo de detección esto para la prueba en diferido y en la prueba de tiempo real, se usa las mismas etapas con la diferencia que se considera la etapa de diagnóstico.

Figura 120

Framework de la estrategia FDD propuesta



Para este caso se introdujeron operaciones de fallas de acuerdo con los antecedentes, la subcarga de refrigerante (OC), sobrecarga de refrigerante (UC), restricción de la línea de líquido (RL), reducción de flujo de aire del condensador (EA) y reducción de flujo de aire del evaporador (CA). Se usó la metodología data-driven con seis modelos de aprendizaje supervisado, Support Vector Machine, Naive Bayes, Decision Tree, K-Nearest Neighbors, Gradient Boosting, y Random Forest, sobre las variables independientes presión, temperatura, corriente y voltaje. Finalmente, se empleó el modelo Random Forest, logrando un accuracy del 96%. Mediante un diagrama comparativo y se evidenció que el performance general del modelo RF, ajustado con los parámetros óptimos, fue el mejor, por otro lado, el modelo NB tiene el índice de precisión general más bajo de 77%. En este estudio, se utilizó la técnica RandomUnderSampler (RUS) para procesar (equilibrar) el repositorio de datos. El proceso de detección debe considerar las anomalías y las etapas transitorias, basadas sobre el comportamiento de los eventos de fallas evaluadas.

En el estudio desarrollado por (Jiangyu Wang, 2016), trata sobre la detección de fallas en el evento de retorno de refrigerante mediante el uso del algoritmo CART en un compresor scroll, se desarrollaron dos modelos de decision tree y se destacó que el preprocesamiento de datos no recibió la atención adecuada y se sugirió que la precisión del FDD podría incrementarse si se eliminaran los valores atípicos. También se subrayó la importancia de continuar investigando los pasos del preprocesamiento de datos.

En la investigación desarrollada por (Laughman , Armstrog, Leeb, & Armstrong, 2006), en Massachusetts Institute of Technology, Se enfoca en la detección de fallas utilizando únicamente mediciones eléctricas, implementando el monitoreo no intrusivo de carga (NILM) mediante el muestreo de voltaje, corriente y la reducción de transitorios de arranque o armónicos generados. Las variaciones de estas mediciones permiten, diagnosticar fallas en equipos y componentes de unidades de enfriamiento. El uso de NILM complementa otros esquemas FDD basados en sensores y análisis.

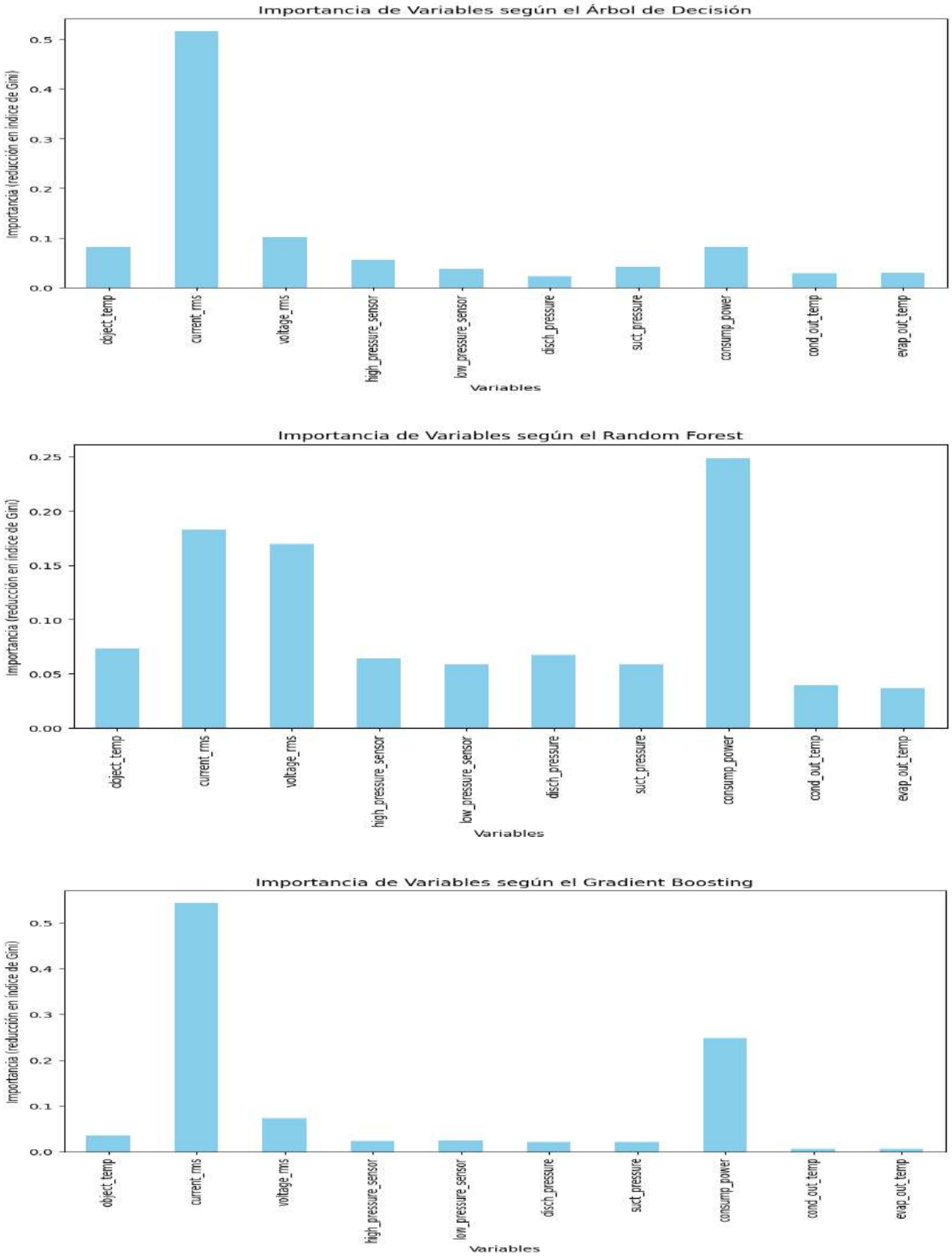
7.4 Implicancias del estudio

Asimismo, se hizo una prueba de validación simulando el funcionamiento real de un aire acondicionado de precisión. En esta prueba, se integró un sistema que genera calor de manera continua en el entorno evaluado, utilizando el modelo de detección que arrojó los mejores resultados, lo que permitió detectar de manera objetiva los eventos de falla registrados durante un ciclo de operación.

Y de manera interna se observaron patrones de variación de las señales, de forma única para cada falla en evaluación por ejemplo (niveles de voltaje elevado, con corriente baja, presión normal y temperatura elevada, para las fallas de subcarga de refrigerante, no comprime lo suficiente y no logra refrigerar, donde se observa sobre calentamiento de todo el circuito y el tiempo de su ciclo de refrigeración es muy largo), (niveles de voltaje normales, con corriente elevada, presión elevada y temperatura elevada, para las fallas de sobrecarga de refrigerante, donde se observa sobre calentamiento de todo el circuito, hay riesgo de congelamiento y el tiempo de su ciclo de refrigeración es largo) y (niveles de voltaje normales, con corriente normal, presión normal y temperatura elevada, para las fallas de restricción de la línea de líquido, donde se observa sobre calentamiento de todo el circuito, hay riesgo de congelamiento y el tiempo de su ciclo de refrigeración es largo), (niveles de voltaje elevados, con corriente baja, presión normal y temperatura elevada, para las fallas de reducción del flujo de aire del condensador, donde se observa sobre calentamiento de todo el circuito, hay riesgo de congelamiento y el tiempo de su ciclo de refrigeración es rápida) y (niveles de voltaje menores al promedio, con corriente elevada, presión normal y temperatura del compresor elevado, para las fallas de reducción del flujo de aire del evaporador, donde se observa sobre calentamiento de todo el circuito y el tiempo de su ciclo de refrigeración es largo), pero los tipos de fallas estudiadas aún son limitados. Para los algoritmos de clasificación DT, RF y GB podemos calcular la disminución total en el índice Gini, que un valor grande muestra una variable predictora importante. La figura 121 se gráfica de la relevancia de cada variable predictora en la tarea de clasificación . Basado en los valores conseguidos de estos 3 métodos de clasificación, I_{Comp}, V_{Comp} y T_{Comp} son las variables predictoras más importantes en el proceso de detección y diagnóstico de fallas.

Figura 121

Gráficos de la importancia de las variables para los métodos DT, RF y GB



Capítulo VIII

Costos de Implementación

Tabla 21

Sistema de adquisición

SISTEMA	MATERIALES	PRECIO UNITARIO - soles	CANTIDAD	PRECIO TOTAL (soles)
MODULO DE PROCESAMIENTO	RASPBERRY	400	1	400
ACONDICIONAMIENTO	Fuente de Alimentación	10	1	10
	Tarjeta SD 32G	30	1	30
	Sistema embebido Arduino Nano	45	1	45
	Placa protoboard	8	1	8
	Kit de resistencias, Capacitores, cables	15	1	15
SENSORES	S. de Corriente	15	1	15
	S. de Voltaje	15	1	15
	S. de Presión	250	4	1000
	S. de Temperatura	100	1	100
	S. de Temperatura	15	2	30
MONITOR	HP	300	1	300
REFRIGERACIÓN	Refrigerante R134a	40	4	160
	Válvulas de servicio	3	5	150
ALQUILER DE QUIT DE REFRIGERACIÓN	Juego de manómetros analógicos	300	1	300
ALQUILER DEL SISTEMA AAP	Sistema de aire acondicionado	2000	1	2000
	PRECIO TOTAL TENTATIVO ESTIMADO			4578 soles

Tabla 22

Coste de personal

Cantidad	Costo Horas-Hombre	Tiempo de Trabajo	Remuneración s/.
1	Bachiller (Tesista)	4 meses	4 100.00
1	Ing. Electrónico (Asesor)	4 meses	0.00
1	Técnico en Sistema de Refrigeración	1 día	150.00
	Costo Total Horas Hombre		s/. 4 250.00

Conclusiones

- Se diseñó e implementó con éxito el sistema de diagnóstico y detección de fallas de forma automática en un sistema de aire acondicionado de precisión (AAP) en tiempo real. Basado en las señales presión, voltaje, corriente y temperatura, mediante la implementación del mejor modelo de aprendizaje supervisado. En particular, se entrenaron y evaluaron seis algoritmos (K-Nearest Neighbors, Decision Tree, Support Vector Machine, Gradient Boosting, Random Forest y Naive Bayes) utilizando la base de datos generada, para este tipo de fallas (UC, OC, RL, CA, EA). Los resultados de las pruebas ejecutadas tanto en condición diferida (base de datos) como en tiempo real, demostraron la viabilidad de la detección temprana de fallas, resultando que el modelo Random Forest presentó el mejor rendimiento, con una exactitud del 96% para la condición diferida y para la prueba en tiempo real, una exactitud global del 95.28%, una precisión del 95.31% señala que, cuando el modelo predice una clase positiva, tiene una probabilidad muy alta de ser correcta, minimizando los falsos positivos. La sensibilidad, con un valor del 81.87%, cuantifica la habilidad del modelo para detectar los verdaderos positivos (es decir, las fallas reales), reduciendo los falsos negativos. La especificidad, que alcanza el 98.9%, expresa la habilidad para identificar correctamente los verdaderos negativos (las instancias sin fallas), evitando falsos positivos.
- Se diseñó e implementó satisfactoriamente el sistema de instrumentación y adquisición, sobre el sistema de aire acondicionado de precisión evaluado. En la fase de instrumentación, Los sensores seleccionados demostraron un rendimiento óptimo: el ACS712-30A mostró una excelente linealidad en la respuesta dentro del rango operativo del compresor (0-30A), con una sensibilidad de 66mV/A que permitió detectar incluso pequeñas variaciones en el consumo eléctrico durante los eventos de falla (corrientes de arranque del compresor). El sensor MLX90614 permitió mediciones de temperatura sin contacto con una precisión de ±0.5°C en el rango crítico de operación (5-80°C), minimizando perturbaciones en el sistema. El sensor de voltaje ZMPT101B,

con una precisión de ±1%, facilitó lecturas confiables de 110-250V AC tras la calibración, manteniendo bajos niveles de ruido (±0.3V). Los sensores DS18B20 implementados en múltiples puntos registraron temperaturas con precisión de ±0.5°C y demostraron excelente resistencia a la condensación, manteniendo su integridad incluso en condiciones de humedad variable. El sensor SPKT0043P midió presiones con precisión del ±1.2% en todo el rango operativo del sistema, proporcionando datos críticos para la detección temprana de anomalías en el circuito de refrigeración. Mediante una unidad de adquisición (microcontrolador Arduino Nano) que envía a la unidad de procesamiento (Raspberry Pi 4B), específicamente a un repositorio de datos, la selección de estas unidades se realiza tomando en cuenta la capacidad de procesamiento, almacenamiento y compatibilidad con diferentes frameworks.

- Se construyó una base de datos con 20,000 muestras de señales de condición de estado del sistema AAP. Balanceadas mediante la técnica RandomUnderSampler (RUS) apartir de los 31,057 registros iniciales. Sobre la unidad de procesamiento Raspberry Pi, para su posterior envió a un PC, donde se procesa de forma total. Esta base de datos contiene las siguientes fallas específicas; la subcarga de refrigerante (el nivel de la carga de refrigerante es de 65% - 618g), sobrecarga de refrigerante (el nivel de la carga de refrigerante es 130% - 1235g), restricción de la línea de líquido (se implementó mediante una válvula reguladora para imponer la perdida de presión deseada), reducción de flujo de aire del condensador (se implementó bloqueando una parte de la superficie (intercambiador) del condensador con mallas, para aumentar la resistencia de flujo de aire) y reducción de flujo de aire del evaporador (se implementó bloqueando una parte de la superficie del filtro del evaporador con mallas, para aumentar la resistencia de flujo de aire). Esta metodología de generación de datos permitió crear un conjunto de entrenamiento representativo de las condiciones de falla encontradas en sistemas AAP reales.

- Se logró seleccionar el mejor modelo de aprendizaje (IA), para la detección y clasificación de eventos de falla, entrenados con la base de datos generada. A través de un análisis comparativo de sus métricas de evaluación (usando la curva roc-auc y matriz de confusión), basado en el criterio de exactitud (accuracy), el modelo Random Forest sobresalió con un 96% frente a los otros algoritmos, si el modelo Support Vector Machine alcanzo un 93%, Decisión Tree alcanzo un 93%, Gradient Boosting alcanzo un 91%, K-Nearest Neighbors alcanzo un 83%, Naive Bayes alcanzo un 77% y para la prueba en tiempo real con el mejor modelo resulto un 95.28% de accuracy, en un sistema a escala piloto de desempeño, con etapa condensadora y evaporadora compacta e integrada, impulsados mediante un compresor de tipo scroll. Además, se hizo una prueba de validación con el mejor modelo seleccionado (RF), en tiempo real, donde se simula un entorno real para el sistema AAP y alcanzo una tasa de accuracy del 93.49%, si los hiperparámetros óptimos para este modelo fueron max_depth=25 y n_estimators=150. Aunque se presentaron algunos falsos positivos y negativos, los resultados destacan la eficiencia del sistema para detectar y clasificar indicios de falla en tiempo real, estableciendo un antecedente para estudios futuros.

Recomendaciones

- Considerando la continua transformación tecnológica, es recomendable la alternativa de usar más elementos embebidos con mayores capacidades que integren el sistema de detección y diagnóstico. De este modo, sería posible comparar su efectividad frente al sistema utilizado en este proyecto, con la opción de abaratar costos y optimizar la capacidad de detección. Esta investigación puede optimizar la implementación del sistema detector de fallas, de forma rentable y más confiable.
- Debido al continuo avance de la Inteligencia Artificial, se sugiere investigar y aplicar diversas metodologías (Deep Learning, general Machine) para esta línea de investigación Fault Detection and Diagnosis (FDD) con el fin de comparar su desempeño frente a las empleadas en este estudio. Facilitando un entendimiento más detallado de las habilidades y restricciones de diversas metodologías para la clasificación de fallas, lo que puede conducir a optimizaciones y progresos. significativos en la exactitud y precisión del sistema. Esto con la finalidad de impulsar la filosofía sobre usar la inteligencia artificial para resolver problemas de la industria.
- Si bien el sistema bajo la metodología de Data-Driven (gran cantidad de datos) ofrece varios beneficios como mejor precisión, eficiencia, innovación y transparencia. También tiene requerimientos importantes. Para enfoques futuros se sugiere adoptar una metodología hibrida, combinando lo mejor del método basado en modelos explícitos de entrada-salida con el método basados en datos esto permitirá abordar la complejidad de los sistemas y estructuras variables, mientras se optimiza el rendimiento del sistema detector de fallas. Sin embargo, no obstante, es esencial garantizar la fiabilidad de la información de entrenamiento y mantener un entendimiento profundo del funcionamiento del sistema para lograr resultados óptimos.
- Para disminuir la cantidad de errores de detección tanto los falsos positivos (evento clasificado como falla en un funcionamiento normal) como los falsos negativos (evento clasificado como

normal en un funcionamiento de evento con falla), se recomienda emplear un método de preprocesamiento más exhaustivo, para compensar los problemas de valores atípicos o faltantes en la detección de fallas en tiempo real. Lo que potencialmente incrementaría la precisión del sistema detector.

- Este sistema propuesto resulto con buen nivel de exactitud (accuracy) en la detección de fallas en tiempo real, no obstante, se recomienda usar más otras variables (señales de frecuencia, señal de vibración, señal de humedad, etc.), un buen análisis sobre los sensores más óptimos para cada variable y revisar periódicamente el estado físico de los sensores para descartar y prevenir un mal funcionamiento del sistema.
- Si bien se pudo generar una base de datos óptima, para futuras investigaciones, se sugiere incrementar la recopilación de datos con el mayor número de muestras posible y considerando otros eventos de falla que presentan estos sistemas AAP.

Referencias Bibliográficas

Adauto Arana, R. M. (2021). *Aplicación de la inteligencia artificial en la deteccion de fallas en los motores electricos de corriente continua de iman permanente.* Huancayo: Universidad Nacional del Centro Del Perú.

ANSI/ASHRAE. (1992). *Thermal Environmental Conditions for Human Occupancy.* ASHRAE 55.

Arduino.cc. (2024). *Arduino*. Obtenido de https://www.arduino.cc/

ASHRAE. (octubre de 2013). *TC 9.9 Thermal Guidelines for Data Processing Environments – Expanded Data Center Classes and Usage Guidance*. Obtenido de http://ecoinfo.cnrs.fr/IMG/pdf/ashrae_2011_thermal_guidelines_data_center.pdf

Becerra Robles, C. A. (2021). *Desarrollo de un Sistema de Monitoreo y Control en apoyo a las condiciones ambientales del Centro de Datos del Ministerio de Salud, distrito de Jesus Maria, provincia y departamentos de Lima- Perú.* Lima: Universidad Tecnologica del Perú.

Benites Condori, C. A. (2023). *SISTEMA PARA DETECCIÓN DE AVALANCHAS USANDO UN SOLO SENSOR INFRASÓNICO Y ALGORITMOS DE.* Cusco: Universida Nacional San ANTONIO Abad del CUSCO.

Briones Romero, A. J. (2023). IMPLEMENTACIÓN DE UN ANALIZADOR DE REDES TRIFÁSICO CON ARDUINO. *RESEARCHGATE*, 2-7.

Bustamante Milla, J. A. (2018). *Criterios y Recomendaciones Técnicas para el Ahorro de Energía en Sistemas HVAC.* Lima: Universidad Nacional de Ingenieria.

Calderon Solano, N. d. (2021). *Sistema experto para el monitoreo y alertade modos de falla para el equipo de aire acondicionado de presicion ubicado en el centro de datos de DATIC.* Cartago Costa Rica: TEC Tecnologica de Costa Rica.

Chen, J., Zhang, L., Li, Y., Shi, Y., Gao, X., & Hu, Y. (2022). A review of computing-based automated fault detection and diagnosis of heating, ventilation and air conditioning systems. *Renewable and Sustainable Energy Reviews*, 161.

Contreraz Alvarez, J. L. (2020). *Diseño de un modelo para mantenimiento predictivo en motores de inducción utilizando técnicas de la industria 4.0.* Lima: UTP.

De los Rios Tomala, G. A. (2019). *Mantenimiento predictivo para la supervisión de motores eléctricos aplicando técnicas de inteligencia artificial.* Ecuador Guayaquil: Universidad Catolica de Santiago de Guayaquil.

Dongo Arrayan, H. H., & Perez Olivera, J. C. (2020). *ANÁLISIS EXPERIMENTAL DEL DESEMPEÑO DE COMPRESORES DE REFRIGERACIÓN.* Arequipa: Universidad Católica Santa María.

Ebrahimifakhar, A., Kabirikopaei, A., & Yu, D. (2020). Data-driven fault detection and diagnosis for packaged rooftop units using statistical machine learning classification methods. *Energy and Buildings*, 225.

Espino Timon, C. (2017). *Análisis predictivo: técnicas y modelos utilizados y aplicaciones del mismo - herramientas Open Source que permiten su uso.* Catalunya: Universidad Oberta de Catalunya.

Fernandez DIez, P. (2000). *Compresores.* España: Universidad de Cantabria.

Gallegos Sánchez, C. A., & Huachín Herrera, C. E. (2018). *Desarrollo de un software de monitoreo y predicción en tiempo real de incidencias ambientales para data center de telecomunicaciones.* Lima: Universidad Peruana de Ciencias Aplicadas.

Gandhi, S. (2018). *Aprendizaje supervisado: Una introducción*. Obtenido de http://www.machinelearning.org/tutorial/supervised/index.html

Greene, C. (1992). *Compresores: diseño y operación.* Nueva York: McGraw-Hill.

Guamán Buestán, A. d. (2019). *Desarrollo de un modelo basado en datos a apartir de las señales de vibración para la detección de fallos de un compresor reciprocamente simple de doble efecto.* Lima: UNMSM.

Gutierrez, M., & Ituralde, S. (2017). *fundamentos de instrumentación y control.* Ecuador: Universidad Estatal Peninsula de Santa Elena UPSE.

ICREA. (2018). *International Computer Room Experts Association - International Computer Room Experts Association - ICREA.* ICREA.

Jiangyu Wang, G. L. (2016). *LIQUID FLOODBACK DETECTION FOR SCROLL COMPRESOR IN A VRF SYSTEM UNDER HEATING MODE.* Wuhan: Huazhong University of Science and Technology.

Laguens, M. (2018). *Industria 4.0: El futuro industrial en el mundo.* Madrid: Universidad Politécnica de Madrid.

Laughman , C. R., Armstrog, P. R., Leeb, S. B., & Armstrong, P. R. (2006). *DETECTION OF ROOFTOP COOLING UNIT FAULTS BASED ON ELECTRICAL MEASUREMENTS.* Cambridge: Massachusetts Institute of Technology.

Maisueche Cuadrado, A. (2019). *Utilización del Machine Learning en la Industria 4.0.* Valladolid: Universidad Valladolid.

Peralta Gutíerrez, M. A. (2014). *Diseño de un Sistema de Comunicación para un Sistema de Control de HVAC en una Central Telefónica.* Lima: Pontificie Universidad Católica del Perú.

Python. (2024). *python.org*. Obtenido de https://www.python.org

Raspberry Foundation. (1 de enero de 2020). *Raspberry Pi*. Obtenido de https://www.raspberrypi.org/about

Reyes Sosa, R. D. (2018). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE PARAMETROS DE UN MOTOR DE INDUCCIÓN.* Mexico-Tamaulipas: TECNOLÓGICO NACIONAL DE MÉXICO.

Rivas Quispe, R. A. (2013). *Monitoreo del sistema HVAC en una sala de control a través del Software Metasys.* Lima: Universidad Nacional de Ingeniería.

Rojko, M. (2017). *Industria 4.0: El futuro de la fabricación.* México: McGraw-Hill.

Román, V. (19 de Enero de 2019). *Supervised Learning: Basics of Classification and Main Algorithms.* Towards Dara Science. Obtenido de Retrieved from Obtenido de Towards Dara Science: https://towardsdatascience .com/supervised-learning-basics- of-classification-and-main-algorithms-c16b06806cd3

Salazar Garcia, H. E. (2023). *Sistema de monitorización y alerta temprana para mantenimiento preventivo de compresores en procesos industriales bajo industria 4.0.* Ecuador Ibarra: Universidad Tecnica del Norte.

Salvador, M. (1988). *Análisis, diseño y construcción de compresores.* Barcelona: Labor S.A.

Scikit learn. (2007-2024). *Scikit-learn.org*. Obtenido de https://scikit-learn.org

Uscamayta Quispetupa, R. (2019). *Diseño de un sistema de detección de fallas para el sensor de velocidad de un motor DC LUCAS NULLE configurado en derivación.* Cusco: UNSAAC.

Wang, Y. L. (2021). *RESEARCH ON DIAGNOSTIC STRATEGY FOR FAULTS IN VRF AIR CONDITIONING SYSTEM USING HYBRID DATA MINING METHODS.* Wuhan, China: Huazhong University of Science and Technology.

Zhu, X., Du, Z., Chen, Z., Jin, X., & Huang, X. (2019). *Hybrid model based refrigerant charge fault estimation for the data center air conditioning system.* China: Shanghai University.

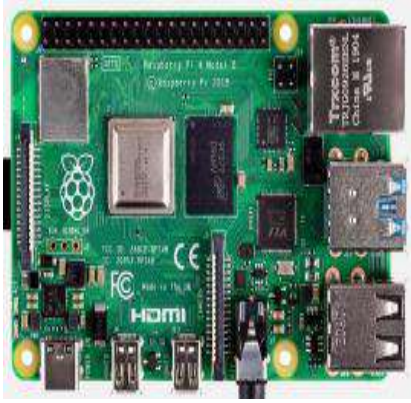
Anexos

Anexo 01: Datasheet – Sistema de Aire Acondicionado

Features

Start-up current max.	At 50 Hz: 20 A At 60 Hz: 22 A
Air throughput (unimpeded air flow)	External circuit: 980 m³/h Internal circuit: 980 m³/h
Energy efficiency ratio (EER) 50/60 Hz L35 L35	Refrigeration factor L35 L35 (EER) 50 Hz: 2.3 Refrigeration factor L35 L35 (EER) 60 Hz: 2.47
Design	wall-mounted
Dimensions	Width: 400 mm Height: 1,580 mm Depth: 295 mm
Protection category to IEC 60 529	External circuit IP 34 Internal circuit IP 54
Protection category NEMA	UL Type 12
Refrigerant/cooling medium	Refrigerant: R134a Quantity: 0.95 kg Global Warming Potential (GWP): 1,430 CO2 equivalent (CO2e): 1.36 t
Temperature control	e-Comfort controller (factory setting +35 °C)
Operating temperature range	10 °C...55 °C
Storage temperature range	-40 °C...70 °C
Setting range	20 °C...55 °C
Power consumption Pel	Power consumption L35 L35/50 Hz: 0.86 kW Power consumption L35 L35/60 Hz: 1.04 kW Power consumption L35 L50/50 Hz: 1.02 kW Power consumption L35 L50/60 Hz: 1.23 kW
Permissible operating pressure (p. max.)	28 bar
Pre-fuse	Miniature circuit-breaker/fuse: 16 A
Packs of	1 pc(s).
Weight/pack	66 kg
Copper weight (kg per piece)	0

Anexo 02: Datasheet - Raspberry Pi 4 Modelo B+



Raspberry Pi 4 Model B features a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro HDMI ports, hardware video decode at up to 4Kp60, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on). For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product retains backwards compatibility with the prior-generation Raspberry Pi 3 Model B+ and has similar power consumption, while offering substantial increases in processor speed, multimedia performance, memory, and connectivity.

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

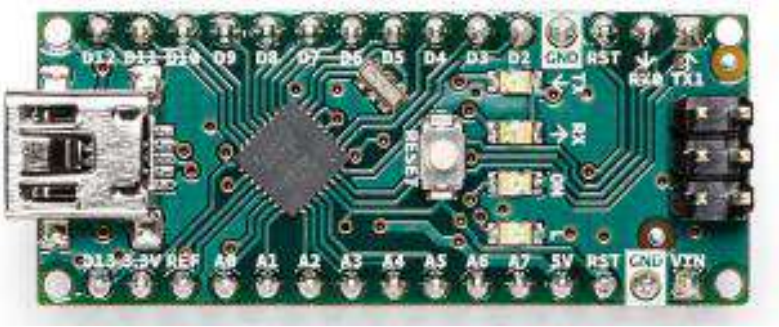
2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained

Anexo 03: Datasheet – Arduino Nano



Description

Arduino® Nano is an intelligent development board designed for building faster prototypes with the smallest dimension. Arduino Nano being the oldest member of the Nano family, provides enough interfaces for your breadboard-friendly applications. At the heart of the board is **ATmega328 microcontroller** clocked at a frequency of 16 MHz featuring more or less the same functionalities as the Arduino® Duemilanove. The board offers 20 digital input/output pins, 8 analog pins, and a mini-USB port.

Target Areas

Maker, Security, Environmental, Robotics and Control Systems

The ATmega48A/PA/88A/PA/168A/PA/328/P is a low power, CMOS 8-bit microcontrollers based on the AVR® enhanced RISC architecture. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix™ acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and VQFN package
    - Temperature Measurement
  - 6-channel 10-bit ADC in SPDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin SPDIP, 32-lead TQFP, 28-pad VQFN and 32-pad VQFN
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1µA
    - Power-save Mode: 0.75µA (Including 32kHz RTC)

Anexo 04: Especificaciones del sensor de temperatura

MLX90614 family

Datasheet Single and Dual Zone
Infra Red Thermometer in TO-39

Melexis
INSPIRED ENGINEERING

Features and Benefits

- Small size, low cost
- Easy to integrate
- Factory calibrated in wide temperature range:
  -40°C...+125°C for sensor temperature and
  -70°C...+380°C for object temperature.
- High accuracy of 0.5°C in a wide temperature range (0°C...+50°C for both Ta and To)
- High (medical) accuracy calibration
- Measurement resolution of 0.02°C
- Single and dual zone versions
- SMBus compatible digital interface
- Customizable PWM output for continuous reading
- Available in 3V and 5V versions
- Simple adaptation for 8V...16V applications
- Sleep mode for reduced power consumption
- Different package options for applications and measurements versatility
- Automotive grade

Application Examples

- High precision non-contact temperature measurements
- Thermal Comfort sensor for Mobile Air Conditioning control system
- Temperature sensing element for residential, commercial and industrial building air conditioning
- Windshield defogging
- Automotive blind angle detection
- Industrial temperature control of moving parts
- Temperature control in printers and copiers
- Home appliances with temperature control
- Healthcare
- Livestock monitoring
- Movement detection
- Multiple zone temperature control – up to 127 sensors can be read via common 2 wires
- Thermal relay / alert
- Body temperature measurement

Ordering Information

Part No.	Temperature Code	Package Code	- Option Code	Standard part	Packing form
MLX90614	E (-40°C...85°C) K (-40°C...125°C)	SF (TO-39)	- X X X (1) (2) (3)	-000	-TU

(1) Supply Voltage/ Accuracy
A - 5V
B - 3V
C - Reserved
D - 3V medical accuracy

(2) Number of thermopiles:
A – single zone
B – dual zone
C – gradient compensated*

(3) Package options:
A – Standard package
B – Reserved
C – 35° FOV
D/E – Reserved
F – 10° FOV
G – Reserved
H – 12° FOV (refractive lens)
I – 5° FOV
K – 13°FOV

10.2. Field Of View (FOV)

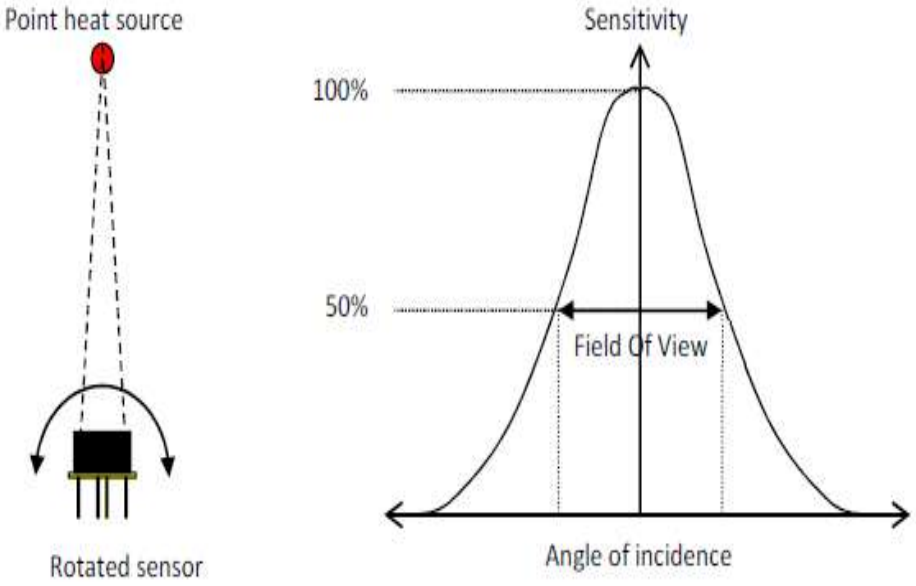


Figure 28: Field Of View measurement

Parameter	Type xAA	Type xBA	Type xCC	Type xCF	Type xCH	Type xCI	Type xCK
Peak zone 1	0°	+25°	0°	0°	0°	0°	0°
Width zone 1	90°	70°	35°	10°	12°	5°	13°
Peak zone 2	NA	-25°	NA	NA	NA	NA	NA
Width zone 2		70°					

Table 14: FOV summary table

Anexo 05: Especificaciones del sensor de corriente

ACS712

Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 µs output rise time in response to step input current
- 50 kHz bandwidth
- Total output error 1.5% at T_A = 25°C, and 4% at –40°C to 85°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kV_{RMS} minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

Package: 8 pin SOIC (suffix LC)

Approximate Scale 1:1 

Description

The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, automotive, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and overcurrent fault protection.

The device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sensing. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power

Continued on the next page...

Descripción

Este módulo basado en el circuito integrado ACS712 de Allegro MicroSystems permite medir la cantidad de corriente que fluye a través de un circuito de corriente alterna (AC) o corriente directa (DC). El método de sensado es a través de un sensor de efecto hall que provee un voltaje de salida proporcional a la corriente que fluye en el circuito. El trayecto para la medida de corriente es por el interior del circuito integrado y se encuentra aislado del circuito de procesamiento.

	5A Module	20A Module	30A Module
Supply Voltage (VCC)	5Vdc Nominal	5Vdc Nominal	5Vdc Nominal
Measurement Range	-5 to +5 Amps	-20 to +20 Amps	-30 to +30 Amps
Voltage at 0A	VCC/2 (nominally 2.5Vdc)	VCC/2 (nominally 2.5Vdc)	VCC/2 (nominally 2.5VDC)
Scale Factor	185 mV per Amp	100 mV per Amp	66 mV per Amp
Chip	ACS712ELC-05A	ACS712ELC-10A	ACS712ELC-30A

Anexo 06: Especificaciones del sensor de voltaje

ZMPT101B

MICRO PRECISION VOLTAGE TRANSFORMERS

InnovatorsGuru | Sensor | © InnovatorsGuru.com

Front view **Bottom view**

The main technical parameters:

Model	ZMPT101B
Rated input current	2mA
Rated output current	2mA
turns ratio	1000:1000
phase angle error	≤20′ （input 2mA，sampling resistor 100Ω）
operating range	**0~1000V** **0~10mA（sampling resistor 100Ω）**
linearity	≤0.2%(20%dot~120%dot)
Permissible error	-0.3%≤ f ≤+0.2%（input 2mA，sampling resistor 100Ω）
isolation voltage	4000V
application	voltage and power measurement
Encapsulation	Epoxy
installation	PCB mounting（Pin Length>3mm）
Operating temperature	-40℃~+60℃
Case Material	**ABS (Note: ABS CASE is NOT available for wave-soldering)**

Anexo 07: Especificaciones del sensor de presión

ENG

CAREL

7. P SERIES FEMALE

7.1 Technical specifications - P Series Female

Carel type P pressure sensors are cost-effective, highly accurate products that use piezoresistive technology, with a 0.5-4.5 ratiometric output and brass housing. Excellent EMC features make these sensors suitable for the harshest environments. These sensors can be directly installed on the refrigerant pipe (no capillary tube is needed) Compatible with the most common refrigerants. This series is excluded from the scope of the Pressure Equipment Directive 2014/68/EU (the sensor itself does not have safety function). The sensors are equipped with aesthetic o-rings to recognise the pressure range easily.

Electrical	
Power supply (protected against polarity reversal)	5 Vdc ±10%
Power supply overvoltage	18Vdc
Maximum reverse voltage	11Vdc
Current draw	5 mA typical
Output voltage	0.5-4.5 Vdc ratiometric
Short-circuit protection	yes
Output load	>47 kΩ
Response time	10 ms max
Insulation resistance	1 GΩ @ 50 Vdc
Electrical connector	Male, 3-pin Metri-Pack 150
Electrical connector insulation material	PBT 30GF
Electrical contact material and surface finish material	Cu Zn20, Ni 2-3µm Sn 5 ± 2.5 µm
Cable	See SPKC****** accessory

Performance	
Operating temperature	-40T135°C
Operating humidity	0-90%rH
Fluid temperature	-40T135°C
Storage temperature	-40T150°C
Ingress protection	IP55, IP69K depending on the connector plugged in. For more details, see sensor table and SPKC****** accessory table.
Accuracy (including linearity, hysteresis, repeatability, calibration error) static error @25°C at 5.0Vdc	±1.2% FS
Temperature error	±0.013% FS/°C
Total error band (including linearity, hysteresis, repeatability, calibration error) relative to all operating temperature and humidity values	±1.5% FS at 5 Vdc (0T50°C) ±2.1% FS at 5 Vdc (-40T90°C) ±2.6% FS at 5 Vdc (40T135°C)*
Life cycle	10 million cycles, 0-100% FS

Physical	
Vibrations IEC 60068-2-64	12 g (rms)
Shock IEC 60068-2-27	50 g 6 ms
Drop from any axis	1.5m (falling from 1.5 metre high)
Material in contact with refrigerant	Ceramic, brass and HNBR O-ring
Housing	Brass
Tightening torque	12 to 16 Nm
Mechanical connection	Female, 7/16"-20UNF - 45° flare. Complies with regulations SAE J513
Pressure range	From 4.2 barg to 45 barg
Over pressure	See table
Burst pressure	See table
Refrigerant compatibility	R12, R22, R134A, R404A, R407C, R410A, R448A, R449A, R452A, R454B, R454C, R502, R507, R513A, R600, R600A, R744, HFO 1234ze, R290, R32, water (temperature >3°C). Not compatible with R717 (ammonia), not suitable to be used with glycol-water mixtures.
Oil compatibility	PAG, POE PVE, PAO, mineral oil and alkylbenzene.
Vacuum pressure (referred to refrigerant circuit)	0 bar absolute
Weight	30 g (net weight)

EMC	
Electrostatic discharges: EN 61000-4-2	±4 kV contact, ±8 kV in air
Radiated immunity: EN 61000-4-3	10 V/m (80 MHz - 1 GHz) 3 V/m (1.4 GHz - 2 GHz) 1 V/m (2 GHz - 2.7 GHz)
Burst: EN 61000-4-4	±1 kV
Surge: EN 61000-4-5	±500 V
Immunity to conducted radio-frequency disturbance: EN 61000-4-6	10 V (150 kHz - 80 MHz)
Magnetic fields at power supply frequency: EN 61000-4-8	30 A/m continuous 300 A/m impulsive

Compliant with:	
Compliance	REACH - RoHS - CE IEC 60335-2-24 clause 22.110; IEC 60335-2-40 clause 22.117; IEC 60335-2-89 clause 22.114
UL certified	File E493623
ATEX - Directive 2014/34/EU	EN60079-0 & EN60079-15

CAREL

ENG

Part numbers

Carel P/N	Pressure (psi)		Pressure (bar)		Pressure (kPa)		Over pressure			Burst pressure			O-Ring
	0.5 V	4.5 V	0.5 V	4.5 V	0.5 V	4.5 V	psi	bar	kPa	psi	bar	kPa	
SPKT0053P* (1)	-15	60	-1	4.2	-100	420	360	25	2500	1595	110	11000	Blue
SPKT0013P* (1)	-15	135	-1	9.3	-100	930	430	30	3000	1595	110	11000	Red/Blue
SPKT00E3P* (1)	-15	185	-1	12.8	-100	1280	550	38	3800	1595	110	11000	Brown
SPKT0043P* (1)	0	250	0	17.3	0	1730	780	54	5400	1595	110	11000	Green
SPKT00F3P* (1)	0	300	0	20.7	0	2070	900	62	6200	1595	110	11000	White
SPKT0033P* (1)	0	500	0	34.5	0	3450	1010	70	7000	2494	172	17200	Black
SPKT00B6P* (1)	0	650	0	45	0	4500	1310	91	9100	2494	172	17200	Red

*Digit 10: 0=single packaging; 1=multiple packaging; 3=distribution package

All pressures are Sealed Gauge.

Notes

Measurement type: Sealed gauge

Full span definition: FS (full span) = MAX output - MIN output = 4 V

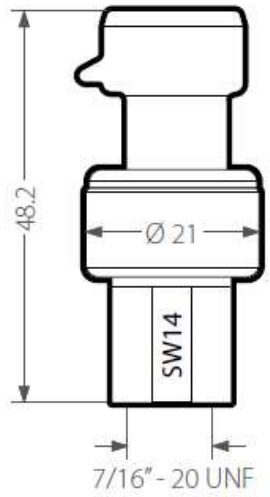
Requirements: Important, for the purpose of protecting the sensor against damage due to inducted overvoltage and incorrect use, it is recommended to proceed as follows.

- **Power supply:** pressure sensors must be powered by a PELV source. If not connected to a Carel controller, protect with a 50 mA fuse on the power supply positive.
- **Connection cable:** avoid winding the cable in spirals and adequately separate the cable from power cables.

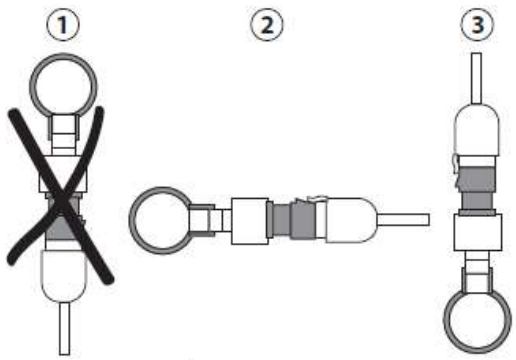
If the SPKT00**P* devices are used in ATEX applications, following Specific Conditions of Use shall be employed:

- Transient protection shall be provided that is set at a level not exceeding 140% of the peak rated voltage value at the supply terminals to the devices. (5Vdc).
- The devices shall be protected in end-use application by another suitable Ex certified enclosure or by an enclosure which has been submitted to Thermal endurance to heat and cold (Clauses 26.8 and 26.9 of IEC/EN 60079-0) and Test for resistance to impact (Clause 26.4.2 of IEC/EN 60079-0).

Dimensions



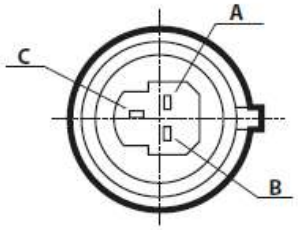
Installation



1	NOT RECOMMENDED
2	RECOMMENDED
3	ACCEPTED

Do not use sealing glue or copper gaskets for mechanical connection

Electrical connection diagram



A	GND
B	Power supply
C	V out

(*) For further details, contact to Carel

Anexo 08: Especificaciones del sensor de DS18B20

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

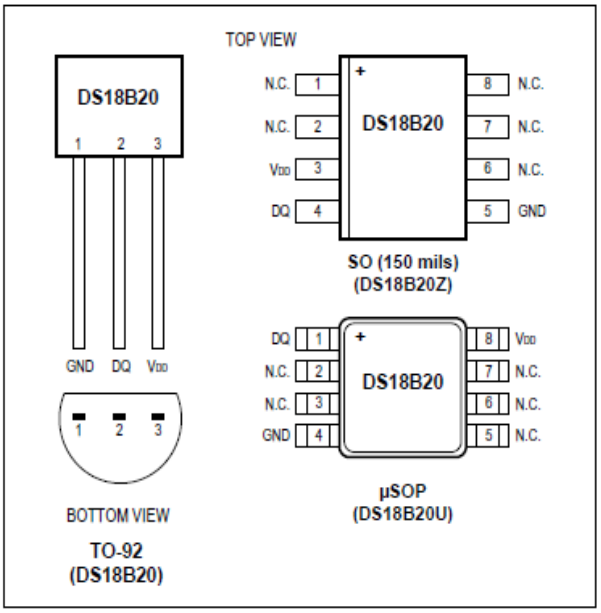
Ordering Information ***appears at end of data sheet.***

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



Anexo 09: Código 1 *Sensado y digitalización*

```
#include <ZMPT101B.h>              // Librería para ZMPT
#include <OneWire.h>               // Comunicación OneWire para DS18B20
#include <Wire.h>                  // I2C para MLX90614
#include <DallasTemperature.h>     // Librería para DS18B20
#include <Adafruit_MLX90614.h>     // Librería para MLX90614

// Pines de conexión
#define SENSOR_IN0 A0              // ACS712 (Corriente)
#define SENSOR_IN1 A1              // ZMPT (Voltaje)
#define PSENSOR2 A2                // Sensor de presión alta
#define PSENSOR3 A6                // Sensor de presión baja
#define ONE_WIRE_BUS A3             // Pin compartido para los dos DS18B20

// Parámetros de calibración
int mVperAmp = 66;                // Sensibilidad ACS712 (mV/A)
const float OFFSET = 0.454;       // Compensación de voltaje para los
sensores de presión
#define SENSITIVITY 325.25         // Sensibilidad del s. de voltaje

// Inicializar sensores
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
ZMPT101B voltageSensor(SENSOR_IN1, 60.0);

OneWire oneWire(ONE_WIRE_BUS);            // Bus OneWire compartido
DallasTemperature ds18b20(&oneWire);      // Instancia para sensores DS18B20

void setup() {
  Serial.begin(9600);
  mlx.begin();           // Iniciar sensor MLX90614
  ds18b20.begin();        // Iniciar los DS18B20
  voltageSensor.setSensitivity(SENSITIVITY);  // Configurar al sensor
}

void loop() {
  // Lectura de los sensores DS18B20
  ds18b20.requestTemperatures();  // Solicitar temperatura a todos los
sensores en el bus

  float tempC- 1 = ds18b20.getTempCByIndex(0);  // Primer DS18B20
  float tempC-2 = ds18b20.getTempCByIndex(1);  // Segundo DS18B20
  // Lectura del MLX90614
  float objectTemp = mlx.readObjectTempC();
  // Lectura del ACS712 (Corriente)
  float currentRMS = obtenerCorrienteRMS(SENSOR_IN0, 500);
  // Lectura del ZMPT (Voltaje)
  float voltageRMS = voltageSensor.getRmsVoltage();
    // Lectura de potencia de consumo
  float powerRMS = voltageRMS*currentRMS;
  // Lectura de los sensores de presión, si un 1 bar= 14.5038psi
  float V2 = analogRead(PSENSOR2) * 5.0 / 1024;
  float V3 = analogRead(PSENSOR3) * 5.0 / 1024;
  float P_kpa2 = ((V2 - OFFSET) * 250) * 0.145038;
  float P_kpa3 = ((V3 - OFFSET) * 250) * 0.145038;

  // Formatear los datos como JSON
  String jsonData = "{";
```

```
  jsonData += "\"object_temp\": " + String(objectTemp, 2) + ",";
  jsonData += "\"current_rms\": " + String(currentRMS, 2) + ",";
  jsonData += "\"voltage_rms\": " + String(voltageRMS, 2) + ",";
  jsonData += "\"high_pressure_sensor\": " + String(P_kpa2, 1) + ",";
  jsonData += "\"low_pressure_sensor\": " + String(P_kpa3, 1) + ",";
  jsonData += "\"disch_pressure\": " + String(P_kpa2, 1) + ",";
  jsonData += "\"suct_pressure\": " + String(P_kpa3, 1) + ",";
  jsonData += "\"consump_power\": " + String(powerRMS, 2) + ",";
  jsonData += "\"cond_out_temp\": " + String(tempC- 1, 2) + ",";
  jsonData += "\"evap_out_temp\": " + String(tempC-2, 2) + "}";

  // Enviar datos al puerto serie
  Serial.println(jsonData);

  delay(1);
}
// corriente RMS con ACS712
float obtenerCorrienteRMS(int sensorPin,int samplingTime) {
  float voltage = obtenervoltajePP(sensorPin, samplingTime);
  float VoltajeRMS = (voltage / 2.0) * 0.707;
  return ((VoltajeRMS - 2.52) * 1000) / mVperAmp;
}
// voltaje de pico a pico
float obtenervoltajePP(int sensorPin,int samplingTime) {
  int maxValor = 0, minValor = 1024;
  uint32_t startTime = millis();

  while ((millis() - startTime) < samplingTime) {
    int readValue = analogRead(sensorPin);
    maxValor = max(maxValor, readValue);
    minValor = min(minValor, readValue);
  }
  return ((maxValor - minValor) * 5.0) / 1024.0;
}
```

Anexo 10: Código 2 *Generación de la base de datos en MYSQL*

```
import serial
import mysql.connector
import json
#Se vincula al repositorio de datos
try:
    db_connection = mysql.connector.connect(
        user = "user_sistemadeadquisicion",
        passwd="123456",
        host = "localhost",
        database="sistemadeadquisicion"
    )
    db_cursor = db_connection.cursor()
    #configuracion del puerto serie
    ser=serial.Serial('/dev/ttyUSB',9600)
    ser.flushInput()
    #lee y almacena datos
    while True:
        if ser.in_waiting>0:
            # lee señales del puerto serial
```

```
            line=ser.readline().decode('utf-8').rstrip()
            print("DATOS RECIBIDOS:", line)
            #transforma a un diccionario JSON
            try:
                data=json.loads(line)
                object_temp = data.get("object_temp")
                current_rms = data.get("current_rms")
                voltage_rms = data.get("voltage_rms")
                high_pressure_sensor = data.get("high_pressure_sensor")
                low_pressure_sensor = data.get("low_pressure_sensor")
                disch_pressure = data.get(("disch_pressure”)
                suct_pressure = data.get("suct_pressure ")
                consump_power = data.get("consump_power ")
                cond_out_temp = data.get("cond_out_temp ")
                evap_out_temp = data.get("evap_out_temp ")
                # Se ingresa las señales a la BD
                insert_query = "INSERT INTO sistemadeadquisicion
(object_temp, current_rms, voltage_rms, high_pressure_sensor,
low_pressure_sensor, disch_pressure, suct_pressure, consump_power,
cond_out_temp, evap_out_temp)VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
               insert_valores = (object_temp, current_rms, voltage_rms,
high_pressure_sensor, low_pressure_sensor, disch_pressure, suct_pressure,
consump_power, cond_out_temp, evap_out_temp)
                db_cursor.execute(insert_query, insert_valores)
                db_connection.commit()
                print("Valores ingresados en el repositorio de datos")
            except ValueError:
                print("Error: Los datos recibidos están en formato JSON
válido.")
except mysql.connector.Error as error:
except serial.SerialException as error:
    print("Error al conectar a la base de datos:", error)
    print("Error al abrir el puerto serial:", error)
finally:
    # se desconecta del repositorio de datos y el puerto serial al finalizar
    if 'db_cursor' in locals():
        db_cursor.close()
    if 'db_connection' in locals():
        db_connection.close()
    if 'ser' in locals():
        ser.close()
```

Anexo 11: Código 3 *Exportar base de datos.csv*

```
import csv
import mysql.connector
db_connection = mysql.connector.connect(
      host="localhost",
      user="user_sistemadeadquisicion",
      passwd="123456",
      database="sistemadeadquisicion"
)
                # Se inserta los datos en el repositorio de datos
query = "SELECT*FROM sistemadeadquisicion;"
cursor = db_connection.cursor()
cursor.execute(query)
results=cursor.fetchall()
```

```
# Se proporciona la ruta
csv_file="/home/pi/Documents/databasegeneral.csv"
with open(csv_file, "w", newline="") as file:
      writer=csv.writer(file)
      writer.writerow([i[0] for i in cursor.description])
      writer.writerows(results)
cursor.close()
db_connection.close()
```

Anexo 12: Código 4 *Gestión de data frame*

```
import pandas as pd
# Se define la función para gestionar la BD, mediante pandas, a un dataframe

def leer_csv(ruta_del_csv):
    datos_dataframe= pd.read_csv(ruta_del_csv)
    return datos_dataframe

# Especificar la ruta del archivo CSV
ruta_del_csv = 'C:/Users/UNSAAC/tesispy/TESIS/training.csv'
dffinal= leer_csv(ruta_del_csv)
print(dffinal)
```

Anexo 13: Código 5 *Entrenamiento con super vector machine SVM*

```
from sklearn import svm

# Se define la función de entrenamiento, procesando los parámetros
específicos para cada modelo

def modelo_SVM_entrenamiento(gamma, C, train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
    # Aquí se entrena el modelo SVM, se encuentra definida y desarrollada
    # cada función(parámetro) en la documentación de la librería ScikitLearn
    #  la ecuación de decisión para clasificar está en función de parámetros
    #  "C" que se encarga de controlar el compromiso entre errores de
    #  clasificación y margen,"gamma" = γ ajusta la influencia de los puntos
    #  de datos en el espacio transformado por el kernel.
    #  Como son datos que no se pueden separar linealmente, se usa kernel
    #  existen tres tipos principalmente (POLINOMIAL,SIGMOIDAL, RBF), para
    #  este caso de entrenar un modelo SVM, se usa (RBF), por defecto.

    SVM = svm.SVC(gamma=0.04, C=500)
    SVM.fittrain_caracteristicas.values, train_clase.values)
    return SVM
```

Anexo 14: Código 6 *Entrenamiento con k nearest neighbors KNN*

```
from sklearn.neighbors import KNeighborsClassifier
# Se define la función de entrenamiento, procesando los parámetros
específicos para cada modelo
def modelo_KNN_entrenamiento(n_neighbors, weights, metric, train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
    # Aquí se entrena el modelo KNN, se encuentra definida y desarrollada
    # cada función(parámetro) en la documentación de la librería ScikitLearn
    #  la ecuación de decisión para clasificar está en función de parámetros
    #  "n_neighbors" que representa el valor de los k datos de entrenamiento
    #  más cercanos, "weights"=("uniform","distance")representa las dos
    #  principales estrategias para predecir para un peso uniforme o por la
    #  distancia del vecino, "metric"= ("manhattan","euclidean","minkowski")
    #  representa las metodologías para el cálculo de la distancia entre
    #  el punto de prueba y los de entrenamiento.
    Weights= "distance"
    Metric= "manhattan"
    KNN = KNeighborsClassifier(n_neighbors=3, weights=Weights,
metric=Metric)
    KNN.fittrain_caracteristicas.values, train_clase.values)
    return KNN
```

Anexo 15: Código 7 Entrenamiento con random forest RF

```
from sklearn.ensemble import RandomForestClassifier
# Se define la función de entrenamiento, procesando los parámetros
específicos para cada modelo
def modelo_RandomForest_entrenamiento(n_estimators, max_depth, train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
    # Aquí se entrena el modelo RF, se encuentra definida y desarrollada
    # cada función(parámetro) en la documentación de la librería ScikitLearn
    #  la ecuación de decisión para clasificar está en función de parámetros
    #  "n_estimators" que representa el número de árboles de decisión que el
    #  modelo construye, "max_depth" controla cuántos niveles puede tener un
    #  árbol de decisión antes de detener el proceso de división,
    #  "min_samples_split", controla el número mínimo de muestras que un
    #  nodo debe tener antes de que el árbol lo divida en nodos más pequeños
    #  "min_samples_leaf" representa el último nodo después de que un árbol
    #  ya no puede dividirse más.
    #  Existen dos tipos de criterios de impurezas ("entropía","gini")
    #  para este caso, de entrenar un modelo RF, se usa ("gini").
    Criterion= "gini"
    RF = RandomForestClassifier(n_estimators=450, max_depth=25,
min_samples_split=1, min_samples_leaf=2, criterion=Criterion )
    RF.fittrain_caracteristicas.values, train_clase.values)
    return RF
```

Anexo 16: Código 8 Entrenamiento con gradient boosting GB

```
from sklearn.tree import GradientBoostingClassifier
def modelo_GradientBoosting_entrenamiento(train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
#Se definen los hiperparametros específicos para el modelo gb
    gb = GradientBoostingClassifier(learning_rate=0.3, max_depth=6,
n_estimators=150, min_samples_split=5, min_samples_leaf=2)
```

```
    gb.fittrain_caracteristicas.values, train_clase.values)
    return gb
```

Anexo 17: Código 9 Entrenamiento con decisión tree DT

```
from sklearn.tree import DecisionTreeClassifier
def modelo_ArbolDecision_entrenamiento(max_depth, criterion, train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
#Se definen los hiperparametros específicos para el modelo dt
    Criterion= "gini"
    dt = DecisionTreeClassifier(max_depth=None, criterion=Criterion,
min_samples_split=2, min_samples_leaf=1)
    dt.fittrain_caracteristicas.values, train_clase.values)
    return dt
```

Anexo 18: Código 10 Entrenamiento con naive bayes NB

```
from sklearn.tree import GaussianNB
# Se define la función de entrenamiento, procesando los parámetros
específicos para cada modelo
def modelo_naivebayes_entrenamiento(var_smoothing, train):
    train_caracteristicas = train.drop(columns=['clase'])
    train_clase = train['clase']
#Se definen los hiperparametros específicos para el modelo nb
    nb = GaussianNB(var_smoothing=0.1)
    nb.fittrain_caracteristicas.values, train_clase.values)
    return nb
```

Anexo 19: Código 11 *Validación del modelo SVM, mediante lista de hiperparámetros*

```
from sklearn.model_selection import GridSearchCV
from sklearn import svm
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_SVM(validation, iteraciones):
    validation_caracteristicas  = validation.drop(columns=['clase'])
    validation_clase  = validation['clase']
    parameters = {'C':[400, 450, 500, 550, 600, 650, 700, 750, 800, 850,
900, 950, 1000, 1100, 1150],
                  'gamma':[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,
                           0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,0.09,
            0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009,]}
    svc = svm.SVC(kernel='rbf')
    clfsvm = GridSearchCV(svc, parameters, cv=iteraciones,
return_train_score=True)
    clfsvm.fitvalidation_caracteristicas , validation_clase )
    # Se representa los resultados del modelo
    evaluacion_resultados = pd.DataFrame(clfsvm.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 20 Código 12 *Validación del modelo KNN, mediante lista de hiperparámetros*

```
from sklearn.model_selection import GridSearchCV
from sklearn import neighbors
```

```
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_KNN(validation, iteraciones):
    validation_caracteristicas = validation.drop(columns=['clase'])
    validation_clase = validation['clase']
    parameters = {'n_neighbors':[1, 3, 5, 7, 9, 15, 25,35, 45, 55, 65, 75,
85, 95],
                  'weights':['uniform', 'distance'],
                  'metric': ['euclidean', 'manhattan', 'minkowski']}
    knn = neighbors.KNeighborsClassifier()
    clfknn = GridSearchCV(knn, parameters, cv=iteraciones,
return_train_score=True)
    clfknn.fitvalidation_caracteristicas, validation_clase)
    # Se representa los resultados del modelo
    evaluacion_resultados = pd.DataFrame(clfknn.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 21: Código 13 *Validación del modelo RF, mediante lista de hiperparámetros*

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_random_forest(validation, iteraciones):
    validation_caracteristicas = validation.drop(columns=['clase'])
    validation_clase = validation['clase']
    parameters = {'n_estimators': [100, 150, 200, 250, 300, 350, 400,
450,500],
                  'max_depth': [None, 10, 15, 20, 25, 30, 35, 40, 45, 50],
                  'min_samples_split': [0,5, 1, 2, 3, 4, 5],
                  'min_samples_leaf': [0,5, 1, 2, 3, 4, 5]}
    rf = RandomForestClassifier()
    clfrf = GridSearchCV(rf, parameters, cv=iteraciones,
return_train_score=False)
    clfrf.fitvalidation_caracteristicas, validation_clase)
    evaluacion_resultados = pd.DataFrame(clfrf.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 22: Código 14 *Validación del modelo GB, mediante lista de hiperparámetros*

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_gradientboosting(validation, iteraciones):
    validation_caracteristicas = validation.drop(columns=['clase'])
    validation_clase = validation['clase']
```

```
    parameters = {
        'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
        'n_estimators': [50, 100, 150, 200],
        'max_depth': [3, 4, 5, 6],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
    gb = GradientBoostingClassifier()
    clfgb = GridSearchCV(gb, parameters, cv=iteraciones,
return_train_score=True)
    clfgb.fitvalidation_caracteristicas, validation_clase)
    # Se representa los resultados del modelo
    evaluacion_resultados = pd.DataFrame(clfgb.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 23: Código 15 *Validación del modelo DT, mediante lista de hiperparámetros*

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_decision_tree(validation, iteraciones):
    validation_caracteristicas = validation.drop(columns=['clase'])
    validation_clase = validation['clase']
    parameters = {'criterion': ['gini', 'entropy'],
                  'max_depth': [None, 10, 20, 30],
                  'min_samples_split': [2, 5, 10],
                  'min_samples_leaf': [1, 2, 4]}
    dt = DecisionTreeClassifier()
    clfdt = GridSearchCV(dt, parameters, cv=iteraciones,
return_train_score=True)
    clfdt.fit(validation_caracteristicas, validation_clase)
    # Se representa los resultados del modelo
    evaluacion_resultados = pd.DataFrame(clfdt.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 24: Código 16 *Validación del modelo NB, mediante lista de hiperparámetros*

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
import pandas as pd
# Se define la función de validación, donde se prueba distintas
combinaciones de los hiperparametros de cada modelo
def validacion_naivebayes(validation, iteraciones):
    validation_caracteristicas = validation.drop(columns=['clase'])
    validation_clase = validation['clase']
    parameters = {'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-
3, 1e-2, 1e-1]}
    nb = GaussianNB()
    clfnb = GridSearchCV(nb, parameters, cv=iteraciones,
return_train_score=True)
    clfnb.fit(validation_caracteristicas, validation_clase)
```

```
    # Se representa los resultados del modelo
    evaluacion_resultados = pd.DataFrame(clfnb.cv_results_)
    evaluacion_resultados.sort_values(by='rank_test_score', inplace=True)
    return evaluacion_resultados
```

Anexo 25: Código 17 *Obtener el vector de característica en tiempo real*

```
import json
import serial
# Etapa de leer datos del puerto serial, transformándolo en json, para
introducirlo en el repositorio de datos
ser=serial.Serial('/dev/ttyUSB',9600)
ser.flushInput()
try:
    while True:
        if ser.in_waiting>0:
            line=ser.readline().decode('utf-8').rstrip()
            print("DATOS RECIBIDOS:", line)
            try:
                data=json.loads(line)
                object_temp = data.get("object_temp")
                current_rms = data.get("current_rms")
                voltage_rms = data.get("voltage_rms")
                high_pressure_sensor = data.get("high_pressure_sensor")
                low_pressure_sensor = data.get("low_pressure_sensor")
                disch_pressure = data.get("disch_pressure
                suct_pressure = data.get("suct_pressure ")
                consump_power = data.get("consump_power ")
                cond_out_temp = data.get("cond_out_temp ")
                evap_out_temp = data.get("evap_out_temp ")
except serial.SerialException as error:
    print("Problemas en la entrada serial:", error)
finally:
    if 'ser' in locals():
        ser.close()
```

Anexo 26: Código 18 Detec*ción de eventos*

```
import warnings
import pickle
from sklearn.exceptions import DataConversionWarning
import serial
import json
# Definir los rangos para cada tipo de falla
fallas = {
    "Falla OC": {
        "object_temp": (20, 60),
        "current_rms": (3.3, 4.2),
        "voltage_rms": (220, 222),
        "high_pressure_sensor": (10, 50),
        "low_pressure_sensor": (5, 40)},
    "Falla UC": {
        "object_temp": (20, 38),
        "current_rms": (3.1, 2.8),
        "voltage_rms": (220, 222),
```

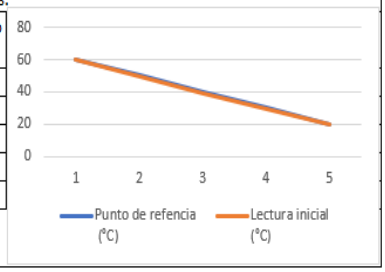
```
        "high_pressure_sensor": (10, 28),
        "low_pressure_sensor": (2, 27)},
    "Falla CA": {
        "object_temp": (20, 40),
        "current_rms": (3.1, 3.4),
        "voltage_rms": (221, 224)},
    "Falla EA": {
        "object_temp": (25, 48),
        "current_rms": (3, 4.8),
        "voltage_rms": (219, 214),
        "high_pressure_sensor": (10, 40),
        "low_pressure_sensor": (5, 40)}}
#se define las funciones para cargar y clasificar
def cargar_modelo(ruta_del_modelo):
    modelo = pickle.load(open(ruta_del_modelo, 'rb'))
    return modelo
def clasificacion_eventos(modelo, vector_caracteristicas):
    return modelo.predict([vector_caracteristicas])
def clasificar_tipo_falla(vector_caracteristicas):
    for falla, condiciones in fallas.items():
        cumple_condiciones = all(
            condiciones[sensor][0] <= valor <= condiciones[sensor][1]
            for sensor, valor in zip(condiciones.keys(),
vector_caracteristicas)
        )
        if cumple_condiciones:
            return falla
    return "Falla desconocida"
ruta_del_modelo = '/home/pi/Desktop/modelos_clasificacion/modelosvm.pkl'
modelo = cargar_modelo(ruta_del_modelo)

#Estable la comunicación serial
baud_rate = 9600
puerto_serial = '/dev/ttyUSB'
arduino = serial.Serial(puerto_serial, baud_rate)
# ignora algun error y continua
warnings.filterwarnings(action='ignore', category=UserWarning,
module='sklearn')
# Función para obtener el vector de características desde la lectura serial
y clasificar eventos
def obtener_y_clasificar_evento():
    datos_arduino = arduino.readline().decode().strip()
    if datos_arduino:
        try:
            datos_json = json.loads(datos_arduino)
            vector_caracteristicas = [
                datos_json["object_temp"],
                datos_json["current_rms"],
                datos_json["voltage_rms"],
                datos_json["high_pressure_sensor"],
                datos_json["low_pressure_sensor"],
                datos_json["disch_pressure"],
                datos_json["suct_pressure"],
                datos_json["consump_power"],
                datos_json["cond_out_temp"],
                datos_json["evap_out_temp"]]
```

```
                    evento_clasificado = clasificacion_eventos(modelo,
vector_caracteristicas)
            return evento_clasificado, vector_caracteristicas
        except (json.JSONDecodeError, KeyError) as e:
            print("Error al decodificar JSON o clave faltante:", e)
    else:
        print("Datos vacíos recibidos desde Arduino.")
    return None, None
while True:
    evento, vector_caracteristicas = obtener_y_clasificar_evento()
    if evento is not None:
        if evento[0] == 1:
            print("Evento clasificado: [1] fallado")
            tipo_falla = clasificar_tipo_falla(vector_caracteristicas)
            print(f"Tipo de falla detectado: {tipo_falla}")
        else:
            print("Evento clasificado: [0] normal")
```

Anexo 27: Código 19 *Exportación de modelos.pkl*

```
import pickle
#Se define las funciones para cargar y descargar el modelo ML
def descargar_modelo(modelo, nombre_del_modelo, ruta=''):
    with open(ruta + nombre_del_modelo, 'wb') as file:
        pickle.dump(modelo, file)
def cargar_modelo(nombre_del_modelo, ruta=''):
    with open(ruta + nombre_del_modelo, 'rb') as file:
        modelo = pickle.load(file)
    return modelo
```

Anexo 28: Hoja de procedimientos y datos de calibración para el sensor de presión (compresor)

<table>
<tr><td colspan="7">PROCEDIMIENTOS Y DATOS DE CALIBRACIÓN: SENSOR DE TEMPERATURA EN COMPRESOR</td></tr>
<tr><td>INFORMACIÓN DEL SENSOR</td><td colspan="6">. TIPO: MLX90614
. RANGO DE OPERACIÓN: -40°C a +125°C
. RESOLUCION: 0.14°C
. ACCURACY (EXACTITUD): Alta exactitud de ± 0,5 °C en un amplio rango de temperatura (0 °C...+50 °C para Ta y To)</td></tr>
<tr><td>INSTRUMENTO DE REFERENCIA</td><td colspan="6">. DISPOSITIVO: Termómetro IR y de contacto Fluke 568 / -40 a 800 °C
. RESOLUCIÓN: 0.1 °C / 0.1 °F
. ACCURACY (EXACTITUD): > 0 °C (32 °F): ±1 % o ± 0,5 °C para 0 a 50 °C
Infrared accuracy: < 0°C (32°F): ±(1.0°C (±2.0°F) + 0.1°/1°C or°F); > 0°C (32°F): ±1% or ±1.0°C (±2.0°F), whichever is greater
Display resolution: 0.1°C / 0.1°F</td></tr>
<tr><td rowspan="2">PROCEDIMIENTO DE CALIBRACIÓN</td><td>CALIBRACIÓN AISLADA (solo sensor)</td><td colspan="5">. Se acondicionaron los siguientes estados: temperatura de fusión del agua, temperatura ambiente, temperatura corporal y temperatura de ebullición del agua. Luego, se compararon las lecturas obtenidas con las del instrumento patrón, obteniendo resultados altamente similares.</td></tr>
<tr><td>CALIBRACIÓN EN EL SISTEMA</td><td colspan="5">. Se instaló el sensor MLX90614 junto con el instrumento de referencia en el mismo punto de medición (3/4 del compresor) del sistema HVAC, asegurando condiciones ambientales idénticas.
. Se definieron 5 puntos representativos dentro del rango operativo del sensor: 60°C, 50°C, 40°C, 30°C y 20°C.
. Se registraron 5 lecturas consecutivas cada segundo con el MLX90614 como con el termómetro de referencia.
. Para cada punto, se calcularon el promedio, el valor mínimo y el valor máximo utilizando funciones de Python, estos valores permitieron determinar el error en las medidas.</td></tr>
<tr><td rowspan="6">DATOS DE CALIBRACIÓN</td><td>Punto de refencia (°C)</td><td>Lectura inicial (°C)</td><td>Error de medida (°C)</td><td>Error absoluto (°C)</td><td>Error relativo (%)</td><td rowspan="6"> 
</td></tr>
<tr><td>60</td><td>59.6</td><td>-0.4</td><td>0.4</td><td>0.666</td></tr>
<tr><td>50</td><td>49.5</td><td>-0.5</td><td>0.5</td><td>1</td></tr>
<tr><td>40</td><td>39.3</td><td>-0.7</td><td>0.7</td><td>1.75</td></tr>
<tr><td>30</td><td>29.5</td><td>-0.5</td><td>0.5</td><td>1.66</td></tr>
<tr><td>20</td><td>19.4</td><td>-0.6</td><td>0.6</td><td>3</td></tr>
<tr><td>ANÁLISIS DE INCERTIDUMBRE</td><td colspan="6">. REPETIBILIDAD: ± 0.15°C (desviación estandar máxima)
. RESOLUCIÓN: ± 0.25°C(mitad de la resolución del sensor)</td></tr>
</table>

Anexo 29: Termómetro IR

Product specifications

	566	568
Infrared temperature range	-40 °C to 650 °C (-40 °F to 1202 °F)	-40 °C to 800 °C (-40 °F to 1472 °F)
Infrared accuracy	< 0 °C (32 °F): ± (1.0 °C (± 2.0 °F) + 0.1°/1 °C or °F); > 0 °C (32 °F): ± 1 % or ± 1.0 °C (± 2.0 °F), whichever is greater	
Display resolution	0.1 °C / 0.1 °F	
Infrared spectral response	8 µm to 14 µm	
Infrared response time	< 500 msec	
Thermocouple Type-K input temperature range	-270 °C to 1372 °C (-454 °F to 2501 °F)	
Thermocouple Type-K input accuracy	-270 °C to -40 °C: ± (1 °C + 0.2 °/1 °C) (-454 °F to -40 °F: ± (2 °F + 0.2 °/1 °F)) -40 °C to 1372 °C: ± 1 % or 1 °C (-40 °F to 2501 °F: ± 1 % or 2 °F), whichever is greater	
D:S (distance to measurement spot size)	30:1	50:1
Laser sighting	Single-point laser < 1 mw output Class 2 (II) operation, 630 nm to 670 nm	
Minimum spot size	19 mm (0.75 in)	
Emissivity adjustment	By built-in table of common materials or digitally adjustable from 0.10 to 1.00 by 0.01	
Data storage with Date/Time stamp	20 points	99 points
PC Interface and cable	None	USB 2.0 with FlukeView® Forms software
Hi/Low alarms	Audible and two-color visual	
Min/Max/Avg/Dif	Yes	
Display	Dot matrix 98 x 96 pixels with function menus	
Backlight	Two levels, normal and extra bright for darker environments	
Trigger lock	Yes	
Switchable Celsius and Fahrenheit	Yes	
Power	2 AA/LR6 Batteries	2 AA/LR6 Batteries and USB when used with a PC
Battery life	If used continuously: laser and backlight on,12 hours; laser and backlight off, 100 hours	
Operating temperature	0 °C to 50 °C (32 °F to 122 °F)	
Storage temperature	-20 °C to 60 °C (-40 °F to 140 °F)	
Bead thermocouple Type-K range	-40 °C to 260 °C (-40 °F to 500 °F)	
Bead thermocouple Type-K accuracy	± 1.1 °C (2.0 °F) from 0 °C to 260 °C (32 °F to 500 °F), typically within 1.1 °C (2.0 °F) from -40 °C to 0 °C (-40 °F to 32 °F)	

Anexo 30: Hoja de procedimientos y datos de calibración para el sensor de presión (evaporador)

PROCEDIMIENTOS Y DATOS DE CALIBRACIÓN: SENSOR DE TEMPERATURA EN EVAPORADOR Y CONDENSADOR

INFORMACIÓN DEL SENSOR

. TIPO: DS18B20
. RANGO DE OPERACIÓN: -55°C a 125°C
. RESOLUCION: 0.5°C
. ACCURACY (EXACTITUD): ± 0.5°C, ± 1°C, ± 2°C

INSTRUMENTO DE REFERENCIA

. DISPOSITIVO: OMEGA HH42A DIGITAL THERMISTOR THERMOMETER -20 a 130°C
. RESOLUCION: 0.01°C
. ACCURACY (EXACTITUD): ±0.02°C

Temp Range °C (°F)	Resolution °C (°F)
-20 to 102 (-4 to 215)	0.01 (0.01)
102 to 130 (215 to 266)	0.02 (0.05)

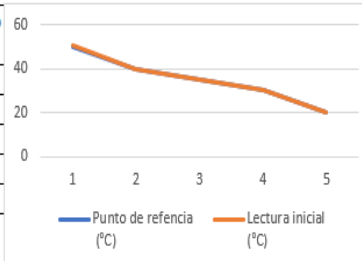
PROCEDIMIENTO DE CALIBRACIÓN

CALIBRACIÓN AISLADA (solo sensor): . Se acondicionaron los siguientes estados: temperatura de fusión del agua, temperatura ambiente, temperatura corporal y temperatura de ebullición del agua. Luego, se compararon las lecturas obtenidas con las del instrumento patrón, obteniendo resultados altamente similares.

CALIBRACIÓN EN EL SISTEMA:
. Se instaló el sensor DS18B20 junto con el instrumento HH42A de referencia en el mismo punto de medición (evaporador y condensador) sobre el sistema HVAC, asegurando condiciones ambientales idénticas.
. Se definieron 5 puntos representativos dentro del rango operativo del sensor: 50°C, 40°C, 35°C, 30°C y 20°C.
. Se registraron 5 lecturas consecutivas con el sensor DS18B20 como con el termómetro de referencia.
. Para cada punto, se calcularon el promedio, el valor mínimo y el valor máximo utilizando funciones de Python, estos valores permitieron determinar el error en las medidas.

DATOS DE CALIBRACIÓN

Punto de refencia (°C)	Lectura inicial (°C)	Error de medida (°C)	Error absoluto (°C)	Error relativo (%)
50	50.65	0.65	0.65	1.3
40	40.09	0.09	0.09	0.225
35	35.18	0.18	0.18	0.51
30	29.95	-0.05	0.05	0.16
20	20.24	0.24	0.24	1.2



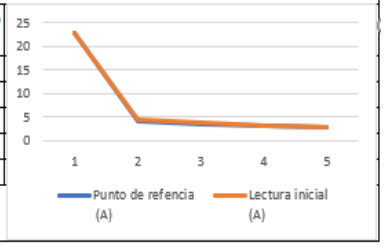
ANÁLISIS DE INCERTIDUMBRE

. REPETIBILIDAD: ± 0.15°C (desviación estandar máxima)
. RESOLUCIÓN: ± 0.25°C(mitad de la resolución del sensor)

Anexo 31: Termómetro digital (termistor) de muy alta exactitud y resolución

Anexo 32: Hoja de procedimientos y datos de calibración para el sensor de corriente

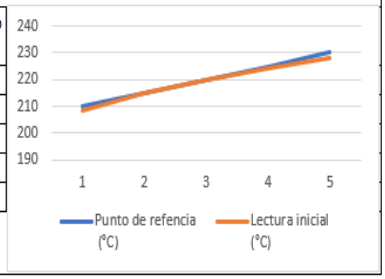
PROCEDIMIENTOS Y DATOS DE CALIBRACIÓN: SENSOR DE CORRIENTE						
INFORMACIÓN DEL SENSOR	. TIPO: ACS712-30A . RANGO DE OPERACIÓN: ±30 A . SENSIBILIDAD: 66 mV/A . ACCURACY (EXACTITUD): ±1.5% (@25 ºC) ajustado en fábrica					
INSTRUMENTO DE REFERENCIA	. DISPOSITIVO: FLUKE 325 TRUE RMS CLAMP METER . RANGO DE OPERACIÓN: 0 a 40A / 400 A . ACCURACY (EXACTITUD): ±2 % ± 5 dígitos (45 Hz a 65 Hz) (si la corriente promedio de funcionamiento es de 3A, entonces el error es de 0.06A y para el sensor 0.045A).	AC current	Range	40.00 A / 400.0 A		
			Accuracy	2% ± 5 digits (45 Hz to 65 Hz) 2.5% ± 5 digits (65 Hz to 400 Hz)		
PROCEDIMIENTO DE CALIBRACIÓN	CALIBRACIÓN AISLADA (solo sensor)	. Se acondicionaron los siguientes estados: niveles de corriente (3, 5, 7.5, 10 Amp), variando elementos resistivos (calefactor, plancha). Luego, se compararon las lecturas obtenidas con las del instrumento patrón, obteniendo resultados altamente similares.				
	CALIBRACIÓN EN EL SISTEMA	. Se instaló el sensor ACS712 junto con la pinza amperimetrica de referencia en el mismo punto de medición dentro del sistema HVAC, asegurando condiciones idénticas. . Se definieron 5 puntos de calibración en el rango de operación del sensor; 22.65, 4.2, 3.5, 3.1 y 2.8 A. . Se registraron 5 lecturas consecutivas cada segundo con el ACS712 como la pinza amperimetrica de referencia. . Para cada punto, se calcularon el promedio, el valor mínimo y el valor máximo utilizando funciones de Python, estos valores permitieron determinar el error en las medidas.				
DATOS DE CALIBRACIÓN	Punto de refencia (A)	Lectura inicial	Error de medida	Error absoluto	Error relativo (A)	
	22.65	22.7	0.05	0.05	2	
	4.2	4.35	0.15	0.15	6	
	3.5	3.7	0.2	0.2	3.33	
	3.1	3.18	0.08	0.08	2.5	
	2.8	2.83	0.03	0.03	1.2	
ANÁLISIS DE INCERTIDUMBRE	. REPETIBILIDAD: ±0.18 A (desviación estandar máxima) . RESOLUCIÓN: ±0.25 A					



Anexo 33: Pinza amperimétrica

Specifications		
AC current	Range	40.00 A / 400.0 A
	Accuracy	2% ± 5 digits (45 Hz to 65 Hz) 2.5% ± 5 digits (65 Hz to 400 Hz)
DC current	Range	40.00 A / 400.0 A
	Accuracy	2% ± 5 digits
AC voltage	Range	600.0 V
	Accuracy	1.5% ± 5 digits
DC voltage	Range	600.0 V
	Accuracy	1.0% ± 5 digits
Resistance	Range	400.0 Ω / 4000 Ω / 40.00 kΩ
	Accuracy	1.0% ± 5 digits
Continuity		30 Ω
Capacitance		0 to 100.0 µF / 100µF to 1000 µF
Frequency		5.0 Hz to 500.0 Hz
AC response		True-RMS
Backlight		Yes
Data hold		Yes
Contact temperature		-10.0°C to 400.0°C (14.0°F to 752.0°F)

Anexo 34: Hoja de procedimientos y datos de calibración para el sensor de voltaje

<table>
<tr><td colspan="7">PROCEDIMIENTOS Y DATOS DE CALIBRACIÓN: SENS0R DE VOLTAJE</td></tr>
<tr><td>INFORMACIÓN DEL SENSOR</td><td colspan="6">. TIPO: ZMPT101B
. RANGO DE OPERACIÓN: 0–250 V AC
. SENSIBILIDAD: se configuró en ~325.25 mV/V
. ACCURACY (EXACTITUD): ±1%</td></tr>
<tr><td>INSTRUMENTO DE REFERENCIA</td><td colspan="6">. DISPOSITIVO: FLUKE 289 TRUE RMS MULTIMETER
. RANGO DE OPERACIÓN: desde 50 mV hasta 1000V
. ACCURACY (EXACTITUD): ±0.4%
AC volts | Range / resolution | 50.000 mV, 500.00 mV, 5.0000 V, 50.000 V, 500.00 V, 1000.0 V
Basic accuracy | 0.4% (True-RMS)</td></tr>
<tr><td rowspan="2">PROCEDIMIENTO DE CALIBRACIÓN</td><td>CALIBRACIÓN AISLADA (solo sensor)</td><td colspan="5">. Se evaluaron distintos niveles de voltaje (218, 222, 225), midiendo el consumo de distintos elementos conectados a una subred (tomacorriente de prueba). Luego, se compararon las lecturas obtenidas con las del instrumento patrón, obteniendo resultados altamente similares. IMPORTANTE: Tiene un potenciometro para regular la ganancia (amplitud)</td></tr>
<tr><td>CALIBRACIÓN EN EL SISTEMA</td><td colspan="5">. Se instaló el sensor ZMPT101B junto con el multimetro de referencia en el mismo punto de medición dentro del sistema HVAC, asegurando condiciones ambientales idénticas.
. Se definieron 5 puntos representativos dentro del rango de operación: 210 V, 215 V, 220 V, 225 V y 230 V.
. Se registraron 5 lecturas consecutivas cada segundo con el ZMPT101B como con el multimetro de referencia.
. Para cada punto, se calcularon el promedio, el valor mínimo y el valor máximo utilizando funciones de Python, estos valores permitieron determinar el error en las medidas.</td></tr>
<tr><td rowspan="6">DATOS DE CALIBRACIÓN</td><td>Punto de refencia (°C)</td><td>Lectura inicial (°C)</td><td>Error de medida (°C)</td><td>Error absoluto (°C)</td><td>Error relativo (%)</td><td rowspan="6">  </td></tr>
<tr><td>210</td><td>208.5</td><td>-1.5</td><td>1.5</td><td>0.71</td></tr>
<tr><td>215</td><td>214.8</td><td>-0.2</td><td>0.2</td><td>0.093</td></tr>
<tr><td>220</td><td>220</td><td>0</td><td>2</td><td>0</td></tr>
<tr><td>225</td><td>224.5</td><td>-0.5</td><td>2</td><td>0.22</td></tr>
<tr><td>230</td><td>228</td><td>-2</td><td>2</td><td>0.87</td></tr>
<tr><td>ANÁLISIS DE INCERTIDUMBRE</td><td colspan="5">. REPETIBILIDAD: ±0.15 V (desviación estandar máxima)
. RESOLUCIÓN: ±0.25 V</td></tr>
</table>

Anexo 35: Multímetro digital

Specifications

Function	Range and Resolution	Basic Accuracy
DC volts AC volts	50.000 mV,500.00 mV, 5.0000 V, 50.000 V, 500.00 V, 1000.0 V	0.025 % 0.4 % (true-rms)
DC current AC current	500.00 µA, 5000.0 µA, 50.000 mA, 400.00 mA, 5.0000 A, 10.000 A	0.15 % 0.7 % (true-rms)
Temperature (excluding probe)	-200.0 °C to 1350.0 °C (-328.0 °F to 2462.0 °F)	1.0 %
Resistance	50.000 Ω, 500.00 Ω, 5.0000 kΩ, 50.000 kΩ, 500.00 kΩ, 5.0000 MΩ, 50.00 MΩ, 500.0 MΩ	0.05 %
Capacitance	1.000 nF,10.00 nF 100.0 nF, 1.000 µF, 10.00 µF, 100.0 µF, 1000 µF, 10.00 mF, 100 mF	1.0 %
Frequency	99.999 Hz, 999.99 Hz, 9.9999 kHz, 99.999 kHz, 999.99 kHz	.005 %

Additional Functions/Features	Fluke 289
Multiple on screen displays	Yes
True-rms ac bandwidth	100 kHz
dBV/dBm	Yes
DC mV resolution	1 µV
Megohm range	up to 500 MΩ
Conductance	50.00 nS
Continuity beeper	Yes
Battery/fuse access	Battery/fuse
Elapse time clock	Yes
Time of day clock	Yes
Min-max-avg	Yes
Duty cycle	Yes
Pulse width	Yes
Isolated optical interface	Yes
Auto/touch hold	Yes
Reading memory	Yes
Log to PC	Yes
Interval/event logging	Yes
Logging memory	up to 15,000 readings

Ordering information

289 True-rms Industrial Logging Multimeter with TrendCapture

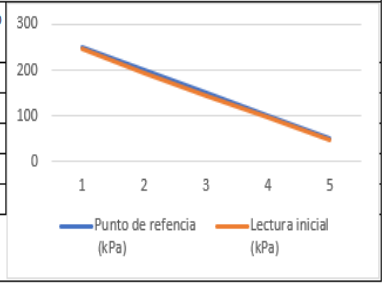
Optional accessories

PVF-SC2	FlukeView® Forms Software with Cable
TLK289	Industrial Test Lead Set
i400	AC Current Clamp
i410	AC/DC Current Clamp
80BK	Integrated DMM Temperature Probe
TPAK	Magnetic Hanging Kit
C280	Soft Case

Fluke. *Keeping your world up and running.®*

Anexo 36: Hoja de procedimientos y datos de calibración para el sensor de presión

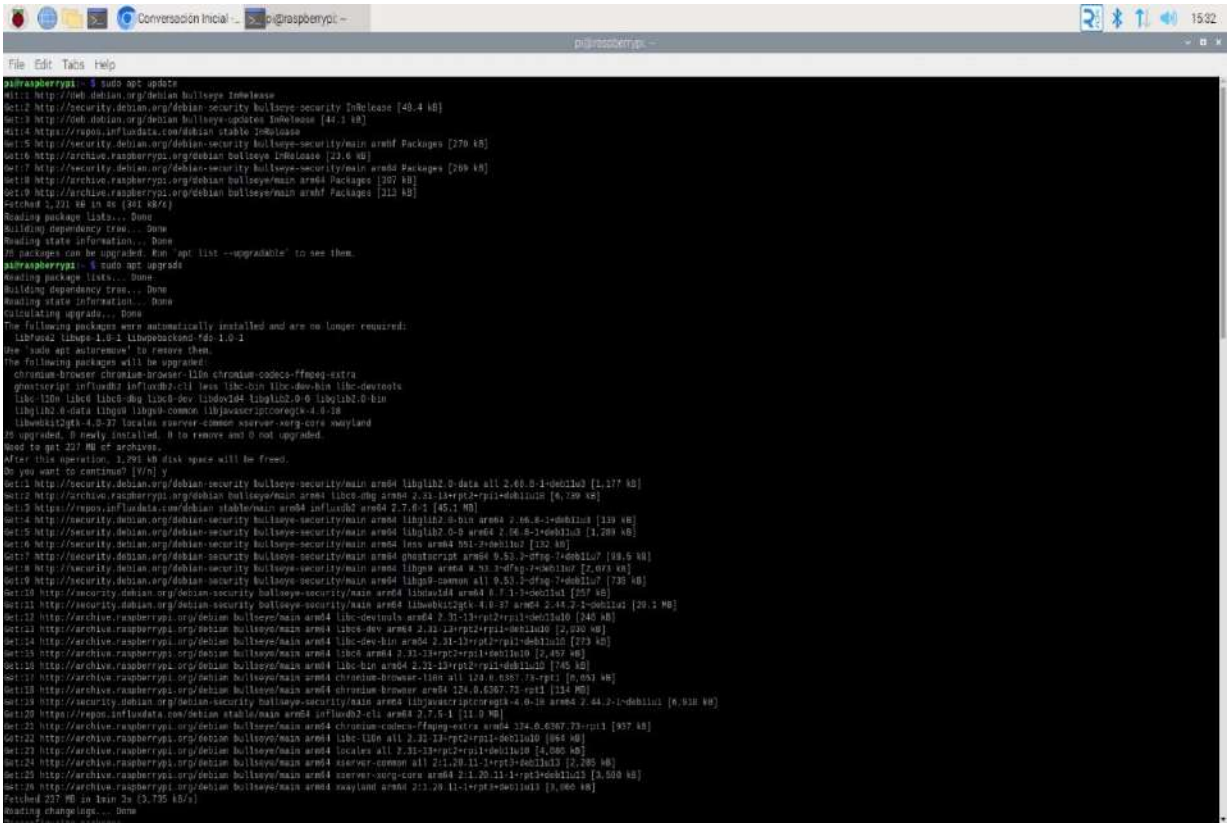
PROCEDIMIENTOS Y DATOS DE CALIBRACIÓN: SENS0R DE PRESIÓN						
INFORMACIÓN DEL SENSOR	. TIPO: SPKT00403P . RANGO DE OPERACIÓN: 0–400 kPa . RESOLUCION: 0.5 kPa . ACCURACY (EXACTITUD): ±2%					
INSTRUMENTO DE REFERENCIA	. DISPOSITIVO: A2LM BRASS MANIFOLD . ACCURACY (EXACTITUD): ±1%					
PROCEDIMIENTO DE CALIBRACIÓN	CALIBRACIÓN AISLADA (solo sensor)	. Sensor industrial calibrado por el fabricante, conforme a estándares internacionales				
	CALIBRACIÓN EN EL SISTEMA	. Se instaló el sensor SPKT00403P junto con el manómetro de referencia en el mismo punto de medición dentro del sistema HVAC, asegurando condiciones ambientales idénticas. . Se definieron 5 puntos representativos dentro del rango operativo del sensor: 250, 200, 150, 100 y 50 kPa. . Se registraron 5 lecturas consecutivas cada segundo con el SPKT00403P como con el manómetro de referencia. . Para cada punto, se calcularon el promedio, el valor mínimo y el valor máximo utilizando funciones de Python, estos valores permitieron determinar el error en las medidas.				
DATOS DE CALIBRACIÓN	Punto de refencia (kPa)	Lectura inicial (kPa)	Error de medida (kPa)	Error absoluto (kPa)	Error relativo (%)	
	250	245	-5	5	2	
	200	194	-6	6	3	
	150	145	-5	5	3.33	
	100	96	-4	4	4	
	50	47	-3	3	6	
ANÁLISIS DE INCERTIDUMBRE	. REPETIBILIDAD: ±0.2 kPa (desviación estandar máxima) . RESOLUCIÓN: ±0.5 kPa					



Anexo 37: Manómetros analógicos de presión

Anexo 38: *Instalación de la válvula de servicio*

Anexo 39: Actualización de instaladores de aplicativos



Anexo 40 Instalación de Mariadb server

Anexo 41: Mysql versión

Anexo 42: Ejemplo de análisis para eventos clasificados por el sistema de detección (RF)

	object_temp	current_rms	voltage_rms	high_pressure_sensor	low_pressure_sensor	disch_pressure	suct_pressure	consump_power	cond_out_temp	evap_out_temp	etiqueta_predicha	etiqueta_real	Verdadero Positivo	Verdadero Negativo	Falso Positivo	Falso Negativo
2	14.75	2.85	226.47	15.9	12.9	15.9	12.9	645.4395	18.81	18.5	1	1	1	0	0	0
3	14.79	2.88	226.64	16.3	13.6	16.3	13.6	652.7232	18.81	18.5	1	1	1	0	0	0
4	14.79	2.82	226.37	16.5	11.9	16.5	11.9	638.3634	18.81	18.5	1	1	1	0	0	0
5	14.79	2.85	226.53	16.8	10.6	16.8	10.6	645.6105	19.06	18.88	1	1	1	0	0	0
6	14.99	2.88	226.83	17	10.6	17	10.6	653.2704	18.81	18.5	1	1	1	0	0	0
7	15.29	2.85	226.84	17.2	9.6	17.2	9.6	646.494	18.81	18.5	1	1	1	0	0	0
8	15.29	2.93	226.57	17.7	10.6	17.7	10.6	663.8501	18.81	18.5	1	1	1	0	0	0
9	15.45	2.88	226.55	17.2	9.2	17.2	9.2	652.464	18.81	18.5	1	1	1	0	0	0
10	15.51	2.93	226.74	17.4	9.2	17.4	9.2	664.3482	18.81	18.5	1	1	1	0	0	0
11	15.75	2.85	226.44	17.9	8.7	17.9	8.7	645.354	18.88	18.5	1	1	1	0	0	0
12	15.77	2.88	226.26	18.1	8.9	18.1	8.9	651.6288	18.94	18.44	1	1	1	0	0	0
13	16.03	2.82	226.24	18.1	9	18.1	9	637.9968	18.94	18.44	1	1	1	0	0	0
14	16.27	2.85	226.51	18.1	8.3	18.1	8.3	645.5535	19	18.38	1	1	1	0	0	0
15	16.37	2.88	226.51	18.4	8.9	18.4	8.9	652.3488	19.06	18.31	1	1	1	0	0	0
16	16.75	2.85	226.5	18.4	8	18.4	8	645.525	19.06	18.25	1	1	1	0	0	0
17	16.99	2.85	226.72	18.6	7.8	18.6	7.8	648.152	19.13	18.19	1	1	1	0	0	0
18	17.41	2.85	226.95	18.4	7.6	18.4	7.6	646.8075	19.25	18.06	1	1	1	0	0	0
19	17.43	2.88	226.59	19.1	8	19.1	8	652.5792	19.31	18	1	1	1	0	0	0
20	17.51	2.82	227.18	18.4	7.3	18.4	7.3	640.6476	19.38	17.88	1	1	1	0	0	0
21	17.63	2.88	226.19	18.8	6.9	18.8	6.9	651.4272	19.44	17.75	1	1	1	0	0	0
22	17.85	2.82	226.84	18.9	6.6	18.9	6.6	639.6888	19.56	17.63	1	1	1	0	0	0
23	18.01	2.88	226.62	18.9	6.4	18.9	6.4	652.6656	19.63	17.56	1	1	1	0	0	0
24	18.31	2.85	226.1	18.8	6.2	18.8	6.2	644.385	19.69	17.44	1	1	1	0	0	0
25	18.55	2.85	226.83	19.1	6.4	19.1	6.4	646.4655	19.81	17.31	1	1	1	0	0	0
26	18.55	2.88	227.15	18.9	5.7	18.9	5.7	654.192	19.88	17.19	1	1	1	0	0	0
27	18.49	2.82	226.76	18.9	5.7	18.9	5.7	639.4632	19.94	17.06	1	1	1	0	0	0
28	18.55	2.88	227.13	19.1	6	19.1	6	654.1344	20.06	16.94	1	1	1	0	0	0
29	18.97	2.82	226.1	19.1	5	19.1	5	637.602	20.13	16.81	1	1	1	0	0	0
30	18.95	2.85	227.24	19.5	5.8	19.5	5.8	647.634	20.25	16.69	1	1	1	0	0	0
31	19.25	2.82	227.5	19.1	5.5	19.1	5.5	641.55	20.31	16.56	1	1	1	0	0	0
32	19.35	2.85	228.5	19.5	5.1	19.5	5.1	651.225	20.38	16.38	0	1	1	0	0	0
33	19.25	2.85	227.73	19.3	4.8	19.3	4.8	649.0305	20.44	16.25	0	1	1	0	0	0
34	19.71	2.85	226.99	19.5	5.1	19.5	5.1	646.9215	20.56	16.13	0	1	0	0	0	1
35	19.93	2.85	226.66	19.5	4.8	19.5	4.8	645.981	20.63	15.94	0	0	0	1	0	0
36	20.23	2.85	226.25	19.5	3.9	19.5	3.9	644.8125	20.75	15.75	0	0	0	1	0	0
37	20.01	2.85	226.74	19.5	5.1	19.5	5.1	646.209	20.81	15.63	0	0	0	1	0	0
38	20.31	2.88	226.25	19.7	5.5	19.7	5.5	651.6	20.94	15.5	0	0	0	1	0	0
39	20.55	2.85	225.6	19.8	4.3	19.8	4.3	641.96	21	15.31	0	0	0	1	0	0
40	20.61	2.85	226.5	19.7	4.6	19.7	4.6	645.525	21.13	15.19	0	0	0	1	0	0

Anexo 43: Ejemplo de análisis eventos clasificados por el sistema de detección (P-V)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
100	25.81	3.03	220.51	22.3	6.2	22.3	6.2	668.1453	26.88	7	1	1	1	0	0	0
101	25.85	3.03	220.2	22.5	5.3	22.5	5.3	667.206	26.94	6.88	0	0	0	1	0	0
102	25.91	3.06	219.91	22.3	5.3	22.3	5.3	672.9246	27	6.81	0	0	0	1	0	0
103	26.03	3.06	221.53	22.8	5.1	22.8	5.1	677.8818	27.06	6.69	0	0	0	1	0	0
104	26.21	3.03	220.52	22.8	5.1	22.8	5.1	668.1756	27.13	6.56	0	0	0	1	0	0
105	26.35	3.01	220.24	22.8	5.3	22.8	5.3	662.9224	27.19	6.44	0	0	0	1	0	0
106	26.43	3.03	220.75	23	5.5	23	5.5	668.8725	27.25	6.38	0	0	0	1	0	0
107	26.53	2.98	220.96	23	6.2	23	6.2	658.4608	27.31	6.25	1	1	1	0	0	0
108	26.71	3.01	220.61	22.8	3.4	22.8	3.4	664.0361	27.38	6.13	0	0	0	1	0	0
109	26.79	3.01	219.89	23.2	5.5	23.2	5.5	661.8689	27.44	6	1	1	1	0	0	0
110	27.03	3.03	221.01	23.6	5.5	23.6	5.5	669.6603	27.5	5.88	0	0	0	1	0	0
111	27.15	3.03	220.41	23.4	5.8	23.4	5.8	667.8423	27.63	5.81	1	0	0	0	1	0
112	26.91	3.03	219.86	23.6	5.1	23.6	5.1	666.1758	27.69	5.69	0	0	0	1	0	0
113	27.15	3.06	220.48	23.6	4.6	23.6	4.6	674.6688	27.75	5.56	0	0	0	1	0	0
114	27.23	3.03	220.99	23.7	3.5	23.7	3.5	669.5997	27.81	5.44	0	0	0	1	0	0
115	27.53	3.01	220	23.9	5	23.9	5	662.2	27.88	5.38	0	0	0	1	0	0
116	27.81	3.03	220.5	23.7	5.3	23.7	5.3	668.115	27.94	5.25	1	0	0	0	1	0
117	28.01	3.06	220.51	24.3	6.9	24.3	6.9	674.7606	28	5.19	1	1	1	0	0	0
118	28.11	3.03	220.18	23.7	5.5	23.7	5.5	667.1454	28.06	5.06	1	1	1	0	0	0
119	28.11	3.01	220.45	24.1	5.7	24.1	5.7	663.5545	28.13	5	1	1	1	0	0	0
120	28.33	3.06	222.33	23.6	5.5	23.6	5.5	680.3298	28.19	4.88	1	1	1	0	0	0
121	28.33	3.01	221.28	23.6	3.5	23.6	3.5	666.0528	28.31	4.81	0	0	0	1	0	0
122	28.31	3.03	220.04	24.4	5.3	24.4	5.3	666.7212	28.38	4.75	0	1	0	0	0	1
123	28.45	3.03	220.41	25	6.9	25	6.9	667.8423	28.44	4.63	1	1	1	0	0	0
124	28.57	3.01	220.01	24.4	4.6	24.4	4.6	662.2301	28.5	4.56	0	1	0	0	0	1
125	28.85	3.01	219.57	24.1	5	24.1	5	660.9057	28.56	4.5	1	1	1	0	0	0
126	28.95	3.06	220.21	24.4	5.5	24.4	5.5	673.8426	28.63	4.38	1	1	1	0	0	0
127	28.97	3.03	220.07	24.3	4.8	24.3	4.8	666.8121	28.69	4.31	1	1	1	0	0	0
128	29.25	3.03	220.15	24.4	4.3	24.4	4.3	667.0545	28.75	4.25	0	1	0	0	0	1
129	29.57	3.01	220.62	24.6	5	24.6	5	664.0662	28.81	4.13	1	1	1	0	0	0
130	29.31	3.03	220.81	23.9	4.3	23.9	4.3	669.0543	28.88	4.06	0	1	0	0	0	1
131	29.51	3.09	220.81	25.1	4.1	25.1	4.1	682.3029	28.94	3.94	0	1	0	0	0	1
132	29.69	3.06	220.85	24.8	7.8	24.8	7.8	675.801	29.06	3.88	1	1	1	0	0	0
133	29.65	3.06	220.75	24.4	4.6	24.4	4.6	675.495	29.06	3.75	1	1	1	0	0	0
134	29.71	3.06	219.84	25.5	7.4	25.5	7.4	672.7104	29.13	3.69	1	1	1	0	0	0
135	29.77	3.03	220.4	25	4.6	25	4.6	667.812	29.19	3.56	0	0	0	1	0	0
136	29.77	3.03	220.1	25.1	3.7	25.1	3.7	666.903	29.25	3.5	0	0	0	1	0	0
137	29.71	3.14	220.1	24.8	4.6	24.8	4.6	691.114	29.31	3.44	0	0	0	1	0	0