**UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO**
**FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA**
**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS**


TESIS

# OPTIMIZACIÓN DEL MANTENIMIENTO PREVENTIVO DE PALAS HIDRÁULICAS CAT 6060 FS APLICANDO TÉCNICAS DE MACHINE LEARNING AL ANÁLISIS TRIBOLÓGICO DE MOTORES

**PRESENTADO POR:**
BR. Ines Katia Huaman Lima
BR. Minerva Campana Cama

**PARA OPTAR AL TITULO PROFESIONAL DE INGENIERO INFORMÁTICO Y DE SISTEMAS**

**ASESOR:**
Dr. Rony Villafuerte Serna

CUSCO - PERÚ
2024

# INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: Optimización del mantenimiento preventivo de palas hidráulicas CAT6060 FS aplicando Técnicas de machine Learning al análisis tribológico de motores

presentado por: Minerva Campana Cama con DNI Nro.: 76676461 presentado por: Inés Katia Huamán Lima con DNI Nro.: 71457734 para optar el título profesional/grado académico de Ingeniero Informatico y de Sistema

Informo que el trabajo de investigación ha sido sometido a revisión por 3 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 3 **%.**

**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 se considera plagio. | 1 X |
| Del 11 al 30 % | Devolver al usuario para las correcciones. | |
| Mayor a 31% | El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica 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 Antiplagio.

Cusco, 12 de noviembre de 2024

Firma

Post firma Rony Villafuerte Serna

Nro. de DNI 23957778

ORCID del Asesor 0000-0003-4607-522X

**Se adjunta:**

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259:398302207

NOMBRE DEL TRABAJO
**palasHidraulicas**

AUTOR
**Ines & Minerva Huaman & Campana**

RECUENTO DE PALABRAS
**30907 Words**

RECUENTO DE CARACTERES
**166373 Characters**

RECUENTO DE PÁGINAS
**160 Pages**

TAMAÑO DEL ARCHIVO
**9.8MB**

FECHA DE ENTREGA
**Oct 25, 2024 8:28 AM GMT-5**

FECHA DEL INFORME
**Oct 25, 2024 8:30 AM GMT-5**

## 3% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base de datos.

- 2% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 1% Base de datos de trabajos entregados

## Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Coincidencia baja (menos de 20 palabras)
- Bloques de texto excluidos manualmente

# Agradecimientos

Gracias a Dios, a mis padres Severino y
Katty, a mi hermano Santi.
Inés.

Gracias a Dios, la familia y a todos aquellos
que comparten sus investigaciones
Minerva.

# Resumen

La minería es un sector que busca constantemente implementar nuevas tecnologías para enfrentar diversos desafíos como: el impacto ambiental, escasez de agua, aumento de los costos de operación, seguridad laboral, etc. Para abordar estos problemas se debe poner especial énfasis en la modernización del área de mantenimiento que a pesar del auge de la Inteligencia Artificial en diferentes industrias este campo es relativamente nuevo en esta área. La presente investigación se centra en aplicar *MachineLearning* a la técnica de Análisis Tribológico para optimizar el Mantenimiento Preventivo Basado en Condición de los motores de palas hidráulicas de una operación minera. El Análisis Tribológico es una de las técnicas más comunes y utilizadas en mantenimiento pues el aceite es un componente clave en el funcionamiento de muchos equipos. El aceite lubrica las partes móviles, reduce la fricción, el desgaste, y ayuda a disipar el calor generado durante el funcionamiento. Con el tiempo, el aceite se degrada y se contamina con partículas de desgaste, agua y otros contaminantes, lo que puede afectar el rendimiento y la vida útil del equipo.
El análisis de aceite permite detectar estos problemas antes de que se produzcan fallas costosas y tomar medidas preventivas para mantener el equipo en buen estado. Además, el análisis de aceite puede proporcionar información valiosa sobre la salud general del equipo, lo que puede ayudar al personal de mantenimiento a tomar decisiones informadas sobre la planificación y programación de tareas de mantenimiento preventivo. El problema se da cuando todos estos beneficios se ven disminuidos debido al proceso de Análisis Tribológico, pues al ser tedioso y ajustado a la interpretación de los ingenieros de mantenimiento tiende a tener grandes márgenes de error. Por ello se propone utilizar *MachineLearning* en esta tarea. El *MachineLearning* puede ayudar en el Análisis Tribológico al proporcionar una forma automatizada de identificar patrones, anomalías en los datos y realizar un análisis más preciso y eficiente de los datos, lo que permitirá una detección más temprana de problemas de mantenimiento y una mayor fiabilidad del equipo. Con este propósito se evaluará el desempeño de múltiples algoritmos de *MachineLearning* para obtener un modelo capaz de hallar anomalías, un modelo clasificador de la condición de muestras de aceite, un modelo predictor de parámetros de muestras de aceite.

**Palabras clave**: Mantenimiento Predictivo, Análisis tribológico, Machine learning.

# Abstract

Mining is a sector that is constantly seeking to implement new technologies to address various challenges such as: environmental impact, water scarcity, increased operating costs, labor safety, etc. To address these problems, special emphasis should be placed on the modernization of the maintenance area that despite the boom of Artificial Intelligence in different industries this field is relatively new in this area. The present research focuses on applying Machine Learning to the technique of Tribological Analysis to optimize Condition Based Preventive Maintenance of hydraulic shovel motors in a mining operation. Tribological Analysis is one of the most common and widely used techniques in maintenance as oil is a key component in the operation of many equipment. Oil lubricates moving parts, reduces friction and wear, and helps dissipate heat generated during operation. Over time, oil degrades and becomes contaminated with wear particles, water and other contaminants, which can affect the performance and life of the equipment. Oil analysis allows you to detect these problems before costly failures occur and take preventative measures to keep your equipment in good condition. In addition, oil analysis can provide valuable information about the overall health of the equipment, which can help maintenance personnel make informed decisions about planning and scheduling preventive maintenance tasks. The problem occurs when all these benefits are diminished due to the process of Tribological Analysis, as it is tedious and adjusted to the interpretation of maintenance engineers it tends to have large margins of error. Therefore, it is proposed to use machine learning in this task. Machine learning can help in Tribological Analysis by providing an automated way to identify patterns, anomalies in the data and perform a more accurate and efficient analysis of the data, which will allow an earlier detection of maintenance problems and greater reliability of the equipment. For this purpose, the performance of multiple machine learning algorithms will be evaluated to obtain a model capable of classifying the condition of an oil sample, a predictive model of oil sample parameters and finally a model that determines the lifetime of CAT 6060 FS hydraulic shovel engines. Predictive maintenance, Tribological analysis, Machine learning.

# Introducción

En el transcurso de los últimos años la inteligencia artificial ha ido ganando espacio en diversas áreas, y en la industria minera esa realidad no sería diferente. Este documento presenta un análisis comparativo de diferentes técnicas de *machinelearning* aplicado al analisis tribologico de motores de equipos mécanicos. Los enfoque basados en *machinelearning* se consideran prometedores para aumentar la seguridad y optimización en los tiempos de mantenimiento preventivo frente a fallas, de ese modo prolongar la vida útil de los equipos minero, para ello es vital el análisis de aceite. El análisis de aceite en la operación minera de la que se obtuvieron los datos, es una técnica de Monitoreo de Condición, cada 5 días se recolectan muestras de los motores se analizan en un laboratorio y luego se analizan los resultados. Analizar los datos de manera tradicional y solo con métodos estadísticos simples implica: Primero, invertir tiempo en analizar muestras catalogadas como 'Anómalas' pero que en el análisis realizado por los ingenieros de confiabilidad se etiquetan como 'Normales'(falsos positivos). Segundo, errores en la detección de fallas potenciales(hay 1100 horas de paradas no programadas desde 2014) Tercero, las interrupciones subjetivas de los responsables del análisis conllevan a errores significativos y paradas no programadas de los equipos. Para resolver estos problemas en el presente proyecto se analizan los resultados implementando machine learning, esta metodologia promete automatizar y refinar este análisis, permitiendo detecciones más precisas y tempranas de falencias potenciales, incrementando así la confiabilidad del equipo. Se evaluaran varios algoritmos de Machine Learning para desarrollar un modelos detector de anomalías de muestras de aceite, y un modelo clasificador de la condición del componente según su muestra de aceite.

Se planteó y desarrolló un flujo de modelos para el procesamiento de los datos recopilados de la base de datos de muestras de aceite de motores, tomadas de palas hidráulicas CAT 6060 FS:

**Modelo 1: Detección de anomalías** El primer modelo consta de las etapas de preprocesamiento para la limpieza de datos no relevantes, manejo de datos faltantes usando imputación por KNN sobre los datos, selección de características de acuerdo con la teoría tribológica se selecciona los parámetros más relevantes, manejo de datos atípicos comparando métodos estadísticos de Rango intercuartílico y el KNN, normalización de limites siguiendo la norma ASTM D7720 y el etiquetado mediante la covarianza robusta.

**Modelo 2: Clasificación de condición de la muestra** El segundo modelo consta de las etapas de etiquetado de condición donde basado en la detección de anomalías del primer modelo, se asigna una etiqueta entre crítico, seguimiento y normal basado en la revisión de datos y el historial de etiquetado manual previo. Luego, la selección de características y la selección de modelos.

Se hace una descripción de cada uno de los capítulos que contiene la tesis de modo que se describe lo que se encontró dentro. El presente proyecto se divide en cuatro capítulos.

**Capítulo 1: Aspectos Generales** En este capítulo se introduce el problema de estudio relacionado con el análisis de aceite en el mantenimiento de equipos, además se plantea el problema de investigación y se formulan las preguntas que se intentarán responder.Se presentan los objetivos generales y específicos de la investigación y se justifica la relevancia y pertinencia del estudio en el contexto actual.

**Capítulo 2: Marco Teórico** En esta capitulo se hace un breve resumen de la teoría de Monitoreo de Condición, análisis de aceite en motores, implicaciones de los diferentes parámetros en el análisis de aceite en los motores y tipos de fallas sintomáticas. Además se realiza una revisión de los antecedentes e investigaciones nacionales e internacionales relacionadas con el análisis de aceite y su aplicación en el mantenimiento incluyendo las bases teóricas y conceptuales necesarias para comprender el enfoque de la investigación además se detalla la revisión de la literatura existente sobre los modelos de *machinelearning* utilizados en el análisis de aceite.

**Capítulo 3: Desarrollo** En este capitulo se describe la metodología utilizada en el estudio, incluye: la recopilación de datos, el preprocesamiento , selección de variables, selección de modelos, selección de métricas de evaluación, selección de métodos explicativos. Se explica de manera detallada la aplicación de las técnicas utilizadas para el entrenamiento, validación, evaluación y explicación de los modelos.

En el **Capítulo 4: Análisis y Discusión de Resultados** Se presentan los resultados obtenidos a partir de la aplicación de los modelos de *machinelearning* analizando e interpretando los resultados y comparándolos con los objetivos planteados para discutir los hallazgos y su relevancia en el contexto. Finalmente se identifican posibles limitaciones y se plantean áreas para futuras investigaciones.

# Listado de Abreviaturas

**IA** Inteligencia Artificial

**ML** Aprendizaje Automático

**DL** Aprendizaje Profundo

**CAT** : Caterpillar

**MODDE** : Motor Derecho

**MBC** : Mantenimiento Basado en la Condición

**FTIR** : Espectrometría de Transmisión de Infrarrojo con Transformada de Fourier

**ASTM** : American Society for Testing and Materials

**TBN** : Número Básico Total

**LSTM** : Long Short-Term Memory

**IQR** : Métrica del Rango Intercuartil

**PCA** : Análisis de Componentes Principales

**UMAP** : Uniform Manifold Approximation and Projection

**AUC** : Área bajo la curva ROC

**MSE** : Error cuadrático Medio

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Aspectos Generales

## 1.1. Planteamiento del Problema

### 1.1.1. Descripción del problema

En la industria minera, la disponibilidad de los equipos de carguío y acarreo utilizados en minas en los procesos de extracción, transporte y procesamiento del mineral es esencial para asegurar la productividad, rentabilidad y la seguridad de la operación. La falta de disponibilidad de equipos puede causar retrasos en la producción y, en última instancia, afectar negativamente los resultados financieros de la empresa. Además, la falta de disponibilidad del equipo también puede llevar a un aumento de los costos de mantenimiento, ya que los equipos que están en uso constante pueden tener un mayor desgaste y requerir reparaciones y reemplazos más frecuentes. Por otro lado, la disponibilidad del equipo también es importante para garantizar la seguridad de la operación. Los equipos que no están en buenas condiciones pueden presentar riesgos de seguridad para los trabajadores y pueden causar accidentes en el lugar de trabajo. Para lograr alcanzar la mayor disponibilidad posible se necesita que el mantenimiento preventivo sea preciso, es decir determinar la condición de un equipo antes de llegar a una parada no programada(falla) para tomar las medidas correspondientes. El problema que este estudio busca abordar es la optimización de la eficiencia en la programación de mantenimiento preventivo y la predicción de fallas en los motores de las palas CAT6060FS. Actualmente el mantenimiento preventivo se da a partir de un Sistema Programado de Paradas en este caso para el motor se hace un mantenimiento preventivo cada 250H,500H,1500H,2500H de trabajo del motor, cada parada preventiva implica indisponibilidad, alto costo y horas-hombre del personal, aún con este tipo de mantenimiento tradicional las paradas no programadas con causa raíz de lubricación representan 107 días de parada desde el inicio de labores de las palas en 2014. De igual manera las fallas imprevistas (paradas no programadas)implican largos periodos de inactividad y reparaciones y repuestos costosos. Para hallar la condición de un componente una práctica común en la industria minera es utilizar el Monitoreo de Condición, proceso de

seguimiento continuo y sistemático de las condiciones de operación que permite monitorear y evaluar el estado de la maquinaria en tiempo real, a través de la medición y análisis de diferentes variables y parámetros y asignarles una condición correspondiente('Critico' parada inmediata,'Seguimiento'programar parada, 'Normal'sin acciones). Dentro del Monitoreo de Condición existen diversas técnicas para medir variables, entre las más destacadas están: el análisis de aceite(tribología), análisis de vibraciones, termografía, filtrografía, ferrografía, ultrasonido entre otros, el principal objetivo de estas técnicas es predecir el momento en el que se inicia la falla para recurrir a un mantenimiento preventivo del equipo. El presente trabajo se centra en el Análisis de Aceite una de las técnicas de monitoreo de condición más comunes debido a que los lubricantes se encuentran en la mayoría de los equipos mineros, pero también una de las más complicadas de analizar por la cantidad de parámetros que mide(64 parametros). Al analizar aceites lubricantes de las maquinarias se hace una analogía como si un médico estuviera analizando los resultados de una muestra de sangre de una persona, si se realiza un adecuado y exhaustivo análisis se logra determinar las condiciones anormales presentes en el aceite lubricante detectando modos de falla de la maquinaria. Actualmente la operación minera de nuestro estudio, cuenta con un laboratorio para medir parámetros tribologicos de las muestras de aceite. Los resultados de este examen son analizados e interpretados mediante métodos estadísticos simples por los ingenieros de mantenimiento, es decir se realiza un análisis univariado de los parámetros donde se establecen los limites de anomalías mediante Desviación Estándar, si solo uno de los parámetros de la muestra excede su limite toda la muestra se considera anómala y es necesario que un experto revise la muestra y determine la condición del equipo,este método que conlleva a una gran numero de falsos positivos 'anómalos' estas mismas muestras con este método tradicional posteriormente son clasificadas como 'Normal' por los expertos.Además, como se menciono antes también hay muestras clasificadas como 'Normal' a pesar de que preceden a una falla o una diálisis de aceite, esto debido que el método tradicional de evaluación presenta rangos de imprecisión y error considerables.Otro problema en este análisis tradicional es que el proceso de análisis es tedioso debido a que implica una serie de tareas manuales y repetitivas que requieren una gran cantidad de tiempo y esfuerzo por parte del personal encargado, a diario en promedio se toman 58 muestras y los resultados deben analizarse a diario pues solo así se tiene un reflejo real de la condición de los equipos. La operación minera de donde se extrajeron los datos ha recopilado datos de muestras de aceites desde 2014 hasta la actualidad, actualmente cuenta con 10 Palas Hidraulicas CAT 6060 FS cada uno con dos motores y el total de muestras de motor es 10004 cada uno con 78 parámetros. El presente trabajo tiene como objetivo diseñar e implementar modelos de *machinelearning*:

- Modelo de detección de Anomalías en muestras de aceite.
- Un modelo clasificador de condición de muestras de aceite en 3 niveles: "Normal","Seguimiento" y "Critico".

### 1.1.2. Identificación del problema

El problema que se aborda en esta investigación es la optimización de la programación de mantenimiento preventivo mediante la detección de anomalías y clasificación de la condición de muestras de aceite de Palas CAT60610fs. Los mantenimientos programados y no programados(fallas) ocasionan tiempos de inactividad inesperados y reparaciones costosas. Con el uso de técnicas de aprendizaje automático, se busca mejorar la capacidad de detectar anomalías que están relacionadas directamente con fallas de motor de manera más precisa y permitir un mantenimiento preventivo basado en la condición real del motor logrando prolongar su vida útil, mejorar su disponibilidad y confiabilidad y reducir los costos operativos.

## 1.2. Formulación del Problema

### 1.2.1. Problema General

¿Como optimizar el mantenimiento preventivo de Palas Hidráulicas aplicando técnicas de *machinelearning* al Análisis Tribologico de sus motores?

## 1.3. Objetivos

### 1.3.1. Objetivo General

Optimizar el mantenimiento preventivo de palas Hidráulicas mediante la aplicación de modelos de *Machine learning* al Análisis Tribologico de motores.

### 1.3.2. Objetivos Específicos

- Recolectar muestras de aceite de motores desde 2014.
- Recopilar las fechas de fallas asociadas a motores para establecer un histórico de eventos.
- Realizar el preprocesamiento de datos, limpieza exhaustiva de datos, detección y manejo de valores faltantes y outliers, y normalizar las mediciones del conjunto de datos.
- Etiquetar las muestras según la Norma ATSM (American Society for Testing and Materials) D7720 Standard Guide for Statistically Evaluating Measurand Alarm Limits when Using Oil Analysis to Monitor Equipment and Oil for Fitness and Contamination.

- Entrenar modelos para detectar anomalías sobre las características no aditivas.
- Evaluar el desempeño de los modelos para detectar anomalías.
- Aplicar técnicas de explicabilidad para comprender los resultados.
- Corregir etiquetado de muestras con datos con las fechas de fallas asociadas para clasificar las muestras según limites temporales:
    1. 'Normal':Muestra con niveles normales.
    2. 'Seguimiento': Muestras con una semana de anticipación a fechas de fallas.
    3. 'Critico': La muestra esta asociada a una falla inminente, una muestra antes de la fechas de falla.
- Comparar y evaluar los resultados obtenidos por los modelos de *Machine learning*.Aplicar técnicas de explicabilidad sobre el mejor modelo.
- Aplicar Voto Mayoritario sobre ambos modelos.

## 1.4. Justificación

### 1.4.1. Conveniencia

Durante muchos años el proceso de análisis de aceite para mejorar la confiabilidad y mantenimiento de las máquinas se ha mantenido relativamente sin cambios. Por lo general se toman las muestras de un activo con una frecuencia regular y se envían a un laboratorio para su análisis. A partir de los resultados emitidos por laboratorio los ingenieros de mantenimiento deben clasificar la condición de la muestra, producir recomendaciones y decidir si toman algunas medidas o se continúa con la operación normal. Los resultados analizados son datos extremadamente valiosos pero el proceso de análisis se vuelve engorroso y puede no proporcionar un espectro consistente de información profunda, incluso para grandes empresas industriales este método requiere un número importante de empleados capacitados para monitorear continuamente los resultados del laboratorio. Los analistas encargados de revisar diariamente muchas muestras de diversos tipos de activos pueden experimentar fatiga y en el peor de los casos, esto puede generar oportunidades perdidas para alertar sobre problemas que pueden resultar en fallas catastróficas de la máquina. Además, el análisis centrado en el ser humano puede ser inconsistente debido a las diferentes opiniones de analista a analista. Todos estos factores pueden conducir a la obtención de niveles variables de valor por el análisis de resultados en laboratorio, por lo que es necesario explorar diferentes formas de etiquetado en vez de fiarse solo de la clasificación existente dada por los ingenieros de mantenimiento.

### 1.4.2. Relevancia

La correcta lubricación es la medida más importante para garantizar la vida útil prolongada y sin problemas de los equipos. Se estima que el 75 % de todas las fallas se deben a problemas de lubricación, por lo que es crucial evaluar el estado de lubricación mediante la clasificación de la condición. Conocer el tiempo de vida útil restante de los componentes es muy útil para que el ingeniero de confiabilidad pueda tomar decisiones informadas en función del estado actual del equipo y el tiempo restante de utilidad. Hoy en día con el auge de la aplicación de la Inteligencia Artificial (IA) y la ciencia de datos se puede analizar grandes volúmenes de información para reconocer patrones, crear modelos de probabilidad y visualizarlos para descubrir tendencias, predecir el futuro y anticiparlo. Aplicar *MachineLearning* permitirá la integración de todos los datos del laboratorio en una sola plataforma, así como la capacidad de ver los activos de una manera más profunda y más granular. Esto se traduce en una mayor comprensión que puede ofrecer una mejor precisión, consistencia y plazos de entrega que los métodos tradicionales.

### 1.4.3. Implicancias Prácticas

Los 3 modelos que se obtienen de está investigación se utilizaran en el Análisis Tribológico de la operación minera de la que se obtuvieron los datos, y se explorara su comportamiento en todos los motores de la operación. Cabe resaltar que al momento del desarrollo de la tesis la empresa tiene planeado integrar un sistema de monitoreo en tiempo de real con sensores que proporcionen datos del aceite en tiempo real,se estima el tiempo de implementación de este sistema en tres años.

### 1.4.4. Valor Teórico

El valor teórico de estudiar la aplicación de *machinelearning* en el análisis de aceite radica en varias consideraciones:

- Mejora de la precisión y eficiencia: El uso de técnicas de *machinelearning* permite analizar grandes volúmenes de datos de manera automatizada, lo que puede llevar a una mayor precisión en la detección de anomalías, patrones y tendencias en el análisis de aceite.
- Optimización de recursos: Al aplicar *machinelearning* en el análisis de aceite, es posible optimizar la asignación de recursos al realizar un seguimiento más preciso del estado y la condición de los equipos, esto ayudará a priorizar el mantenimiento y la sustitución de componentes, evitando el reemplazo innecesario o el mantenimiento preventivo en momentos inapropiados.

- Toma de decisiones informadas: Al utilizar técnicas de *machinelearning*, se pueden obtener perspectivas y patrones ocultos en los datos de análisis de aceite que pueden ser difíciles de identificar manualmente. Esto proporcionará a los profesionales de mantenimiento y confiabilidad una base sólida para tomar decisiones informadas sobre el mantenimiento, la reparación y la planificación de reemplazos.
- Optimización de la vida útil del equipo: Al predecir el intervalo de tiempo de vida útil restante de los componentes basado en el análisis de aceite y el uso de algoritmos de *machinelearning*, es posible maximizar la vida útil del equipo al realizar el mantenimiento oportuno y planificado, evitando así fallas prematuras y prolongando la vida útil del equipo.

### 1.4.5. Utilidad Metodológica

La utilidad metodológica de estudiar los resultados de aplicar *machinelearning* sobre el Análisis Tribológico se refiere a los beneficios prácticos y técnicos que esta investigación puede aportar al campo. Algunas de las utilidades metodológicas incluyen:

- Automatización del proceso de análisis: Las técnicas de *machinelearning* pueden automatizar gran parte del proceso de análisis de aceite. Los algoritmos pueden procesar grandes volúmenes de datos de manera rápida y eficiente, ahorrando tiempo y recursos humanos,lo que a su vez permite realizar un análisis más frecuente y exhaustivo mejorando la detección y la predicción de problemas en los equipos.
- Avance en la investigación científica: El estudio del *machinelearning* aplicado al análisis de aceite contribuye al avance de la investigación científica en el campo de la Confiabilidad y el Mantenimiento preventivo de equipos. Esto puede abrir nuevas oportunidades de investigación, promover el desarrollo de nuevas técnicas y métodos, y fomentar la colaboración entre investigadores y profesionales del sector minero en la región del Cusco.

## 1.5. Delimitación de estudio

### 1.5.1. Delimitación Espacial

La operación minera de la que se extrajeron los datos está ubicada en la región andina de Perú a 5331 m.s.n.m., provincia de Apurimac, es una mina de tajo abierto y las materias primas que se extrae son: Cobre, Zinc, Plata, Oro.

### 1.5.2. Delimitación Temporal

Los datos se recolectaron desde el año 2014 hasta diciembre de 2022, y la investigación se realiza en el año 2023.

## 1.6. Método

### 1.6.1. Alcance

Este estudio incluye una etapa inicial de exploración de los parámetros de las muestras antes de aplicar los modelos de *machinelearning* por lo que incluye elementos de un análisis exploratorio, también se analiza el resultado de emplear los algoritmos de *machinelearning* sobre muestras de aceite de un motor de pala CAT6060fs, debido a que es la primera vez que se realiza en un estudio. Pero luego de pasar está etapa el estudio tiene una orientación descriptiva.
Según (Hernández Sampieri et al., 2014) un estudio de tipo descriptivo se define como un estudio que busca especificar propiedades y características importantes de cualquier fenómeno que se analice, además de describir las posibles tendencias de un grupo o muestra. En la presente investigación se busca especificar los parámetros que tienen más relevancia en un análisis tribologico de motores de Palas CAT6060fs, además de describir las tendencias de cada parámetro para poder predecir sus valores en el futuro. Además siguiendo la definición de estudio descriptivo (Hernández Sampieri et al., 2014). Se aplica procesos y modelos de datos ya establecidos y estructurados por los estudios previos de *machinelearning* los cuales contribuyen al análisis de las muestras de aceite,descripción de vida útil de los motores y predicción de posibles fallas en los motores de Palas CAT6060fs. Por lo tanto esta investigación es de tipo **descriptivo**.

### 1.6.2. Diseño

En el caso de este estudio sobre el análisis de aceite, se esta diseñando y realizando intervenciones específicas implementando técnicas de diagnóstico basado en *machinelearning*, además los modelos implementados serán consumidos por el sistema de reportearía del área de Confiabilidad de la empresa por lo tanto el presente estudio se considera como experimental.

### 1.6.3. Para el desarrollo de la parte informática

**Modulo para usuarios**

Se diseño un modulo dentro del Sistema de Monitoreo y Reporte del área de Confiabiliad, este modulo comsume los resultados de los modelos de machine learning a traves de una API Rest. El módulo muestra los resultados de ambos modelos, clasifica el nivel de anomalia de la muestra y si esta asociado a una falla o cambio de aceite.

Figura 1.1: Listado de muestras etiquetadas

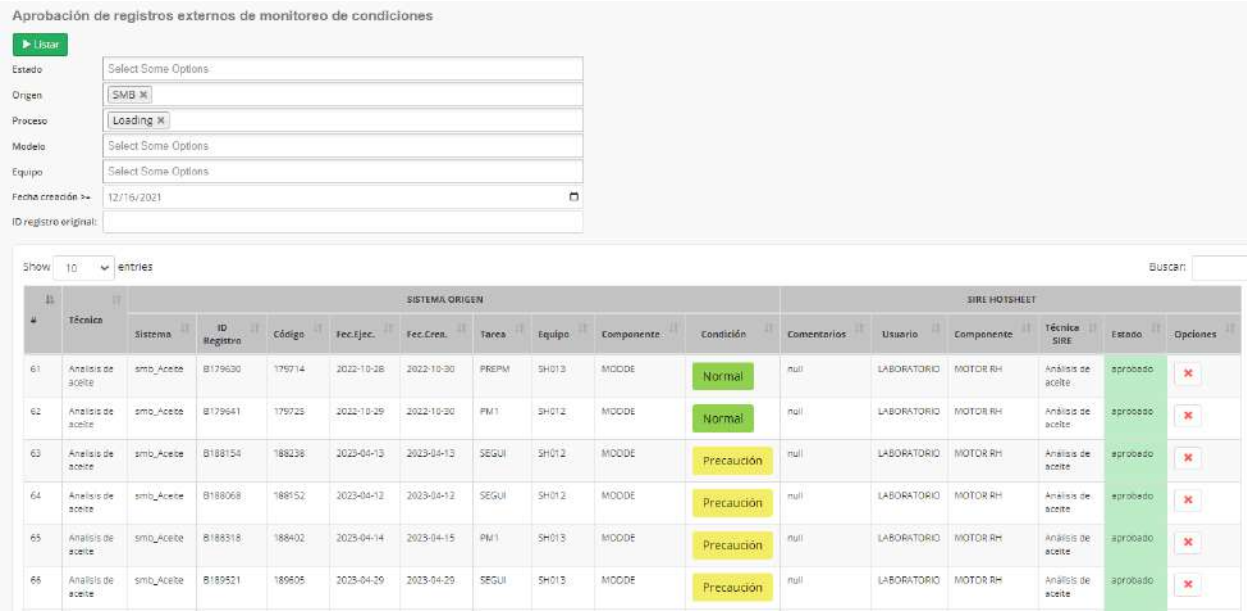
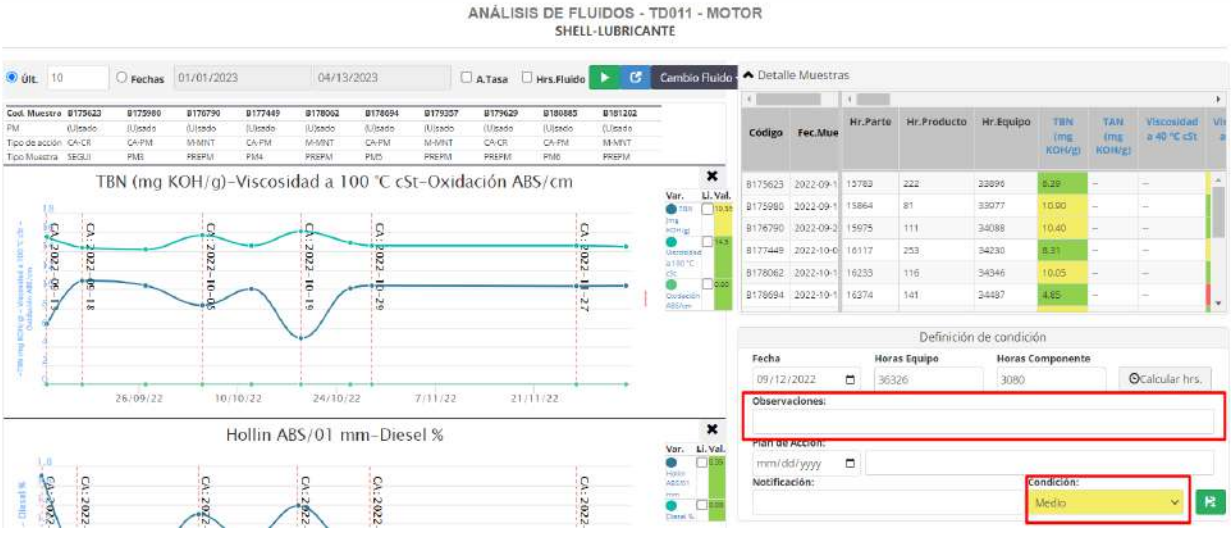



Figura 1.2: Dashboard de Análisis de Aceite



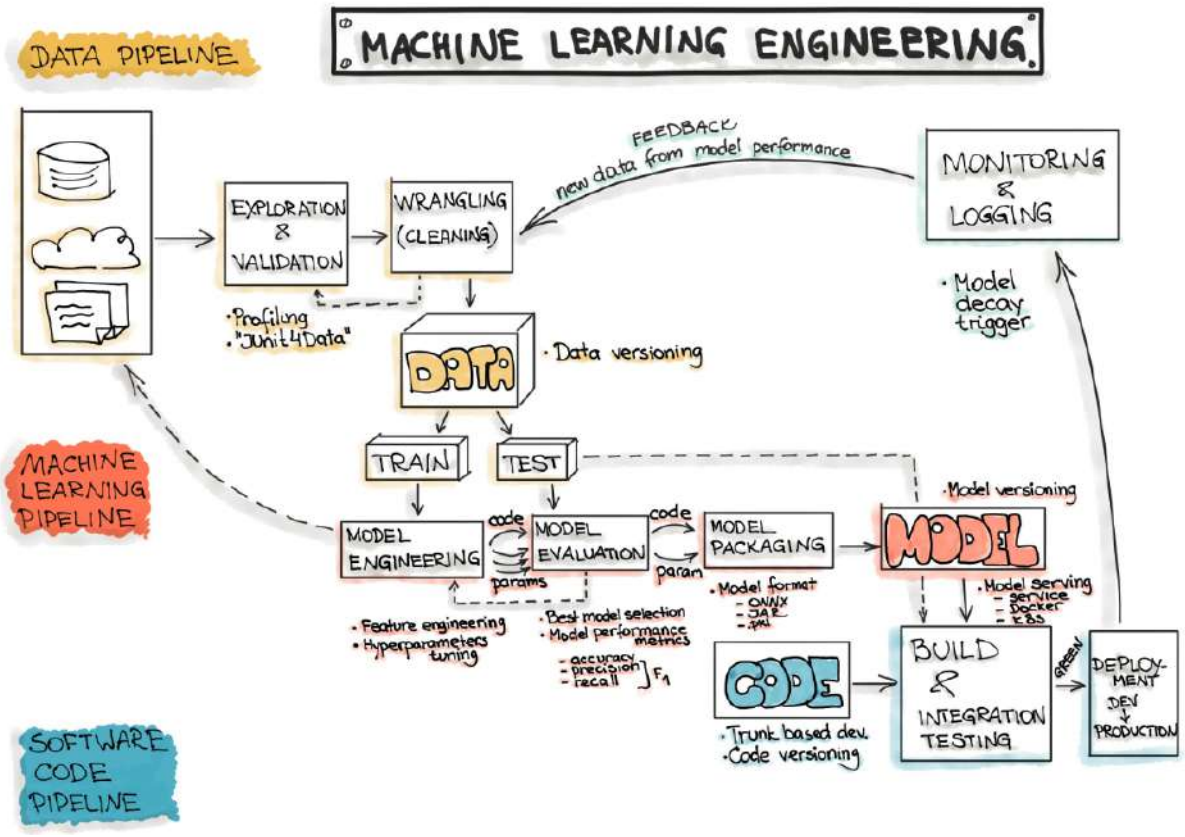

### Desplique de modelos de Machine Learning

En términos generales, el propósito fundamental de un proyecto de *machinelearning* es crear un modelo estadístico haciendo uso de información recopilada y aplicando algo-

ritmos de *machinelearning* . Por ende, todo software basado en ML se compone de tres elementos fundamentales: los datos, el modelo de machine learning y el código. Con relación a estos componentes, el proceso típico de trabajo en *machinelearning* consta de tres etapas principales:

1. Preparación de datos: Incluye la adquisición y la preparación de los datos necesarios.
2. Desarrollo de modelos de machine learning: Implica entrenar y utilizar el modelo de machine learning.
3. Integración del modelo en el código: Se refiere a la incorporación del modelo de machine learning en el producto final.

Figura 1.3: Flujo de despliegue de modelos de machine learning




Fuente: ML-OPS.org

### 1.6.4. Cronograma de Actividades

El proyecto establece el inicio de la realización de la tesis desde el 12 de julio de 2023, una vez levantado las observaciones del plan de tesis, el final proyectado es es para la fecha

de 18 de octubre de 2023, por lo que el plazo determinado teniendo en cuenta el limite de 20 horas semanales por estudiante es de 16 semanas.

# Cronograma de Actividades

| ACTIVIDAD | DURACIÓN DE LA ACTIVIDAD | FECHA DE INICIO | FECHA DE FIN |
|---|---|---|---|
| Selección del tema de la tesis y revisión bibliográfica. | 33 días | 3 de enero de 2024 | 4 de febrero de 2024 |
| Identificación de objetivos y alcances de la investigación. | 9 días | 5 de febrero de 2024 | 13 de febrero de 2024 |
| Presentación del plan de tesis. | 5 días | 14 de febrero de 2024 | 18 de febrero de 2024 |
| Recopilación de datos de muestras de aceite y datos relevantes sobre los equipos y componentes asociados. Preparación y limpieza de los datos. | 7 días | 19 de febrero de 2024 | 25 de febrero de 2024 |
| Análisis exploratorio de datos. | 7 días | 26 de febrero de 2024 | 3 de marzo de 2024 |
| Aplicación de preprocesamiento de datos | 3 días | 4 de marzo de 2024 | 6 de marzo de 2024 |
| Selección y aplicación de modelos de machine learning para la detección de anomalías (modelo 1). | 7 días | 7 de marzo de 2024 | 13 de marzo de 2024 |
| Experimentación y evaluación de resultados de modelos para la detección de anomalías. | 6 días | 14 de marzo de 2024 | 19 de marzo de 2024 |
| Selección y aplicación de modelos de machine learning para la clasificación de condición de la muestra (Modelo 2). | 7 días | 20 de marzo de 2024 | 26 de marzo de 2024 |
| Experimentación y evaluación de resultados de modelos para la clasificación de condición de la muestras | 11 días | 27 de marzo de 2024 | 6 de abril de 2024 |
| Analisis de los resultados obtenidos, interpretación de los hallazgos y comparación con la literatura existente. Identificación de patrones, tendencias y conclusiones. | 6 días | 7 de abril de 2024 | 12 de abril de 2024 |
| Revisión de informe final de la tesis incluyendo resultados, discusión y conclusiones. | 9 días | 13 de abril de 2024 | 21 de abril de 2024 |
| Revisión y corrección del informe final de la tesis en base a los comentarios del asesor o comité de tesis. | 5 días | 22 de abril de 2024 | 26 de abril de 2024 |
| Presentación del trabajo de tesis | 12 días | 27 de abril de 2024 | 8 de mayo de 2024 |

# Capítulo 2

# Marco Teórico

## 2.1. Antecedentes

### 2.1.1. Antecedentes Internacionales

Keartland et al.,2020, *" Automating predictive maintenance using oil analysis and machine learning"*, IEEE Xplore,USA.

El documento detalla la importancia del mantenimiento predictivo así como su objetivo de reducir reparaciones costosas que consumen mucho tiempo, así como evitar actividades innecesarias, proponiendo una estrategia de mantenimiento informada por la monitorización del estado de la maquina. El conjunto de datos se obtuvo de una empresa de análisis de aceite de cajas de cambio consta de 26 características y 887255 muestras, el conjunto fue etiquetado por expertos con 6 condiciones: Sobrecalentamiento(OVHT), Fugas de agua(LEAK),Acumulación de polvo(DUST),Desgaste de componentes(WEAR),Problema de aceite(POIL),Cualquier problema adicional(OTHR), cada clase posee un número de muestras positivas y para compensar el desequilibrio de clases se utilizó un submuestreo de la clase mayoritaria.

El preprocesamiento de datos se comparo el uso de varios tipo de imputación(media, mediana) y técnicas de relleno de k-Nearest Neighbours(kNN), pero no se obtuvieron resultados significativamente diferente,por lo que obtaron por relleno $KNN$ para los valores faltantes. En cuanto a la selección de características las variables importantes fueron Silicio, Aluminio, Cromo,Hierro, Estaño, Cobre,Agua, Sodio, Magnesio. Se estudia el uso de $RandomForest(RF)$, $FeedforwardNeural\ Network(FFNN)$ y $LogisticRegression$ para entrenar datos de análisis de aceite y clasificar las condiciones de la máquina. El uso de técnicas de aprendizaje automático reveló ciertas limitaciones con el uso de sistemas basados en Aprendizaje Profundo y Redes Neuronales Artificiales, pues estos métodos suelen clasificarse como métodos de "caja negra" y es difícil determinar por qué el algoritmo resuelve

cierto resultado, a diferencia de sistemas convencionales como regresión logística o máquinas de vectores de soporte que demuestren suficiente poder explicativo.Keartland y Zyl, 2020

**Conclusiones**: Como resultado de este estudio el modelo *RandomForest* supero a otros clasificadores para todas la condiciones de la máquina y para un mejor comprensión de los resultados se utilizó la interpretación de la importancia de las características para los modelos. Además para evaluar los resultados no solo se ha tenido en cuenta la precisión general sino también las métricas por cada clase. Para etiquetar LEAK, DUST y WEAR los modelo RF y FFNN mostraron capacidad de separación, pero RF supero a todos los modelos alcanzando una precisión de 0.98 % y un AUC de 0.98 %. Para OVHT, POIL, OTHR ninguno de los modelos mostró una capacidad de clasificación significativa RF consigue alta recuperación pero baja precisión. Los resultados que señalan los factores de influencia especifico que afecta determinada condiciones de la máquina, siendo los factores coherentes con los conocimientos del sector.

**Comentario**:Este estudio explica que el uso de Random Forest en la industria de mantenimiento predictivo es ampliamente valorado y que para sus datos este modelo obtuvo los mejores resultados, además explica las desventajas de utilizar técnicas de Aprendizaje Profundo en la clasificación de la condición es por ello que en nuestra investigación para el problema de clasificación de la condición se utilizan métodos de Aprendizaje supervisado. Para nuestro proyecto se tomaron en consideración:

- La aplicación de La ̈Interpretación de la Importancia de Características"debido a que los resultados de *machinelearning* pueden ser difíciles de interpretar, especialmente para las personas que no están familiarizadas con estas técnicas como en el caso del personal minero.
- La comparación de aplicar Imputación( media/mediana) y kNN para completar los datos faltantes.
- La aplicación de Random Forest ampliamente valorado en la industria de mantenimiento.
- Evaluación de resultados por cada clase.

Grebenian et al, 2021, *"Oil condition monitoring, an AI application study using the Classification Learner Technics"*, IOP Conf.

Este estudio se centra en predecir la vida útil del aceite lubricante en base a datos recopilados directamente de la caja de cambio y tren de transmisión. Los datos recopilados cuentan con 19 parámetros de estado del aceite que resultan de mediciones en línea en un banco de experimentos construido y operado en condiciones similares para determinar la salud de las cajas de cambio y los componentes del tren de transmisión, el conjunto de datos se recolectó durante seis meses validando continuamente los datos en varias instancias. Este estudio pretendía mostrar cómo predecir datos de series temporales utilizando ténicas de Inteligencia Artificial

sobre un conjunto de 19 parámetros, no todos independientes de los cuales se seleccionaron 3 parámetros de estado(que determinan el grado de degradación del aceite lubricante) y un conjunto de valores correspondientes al conjunto de valores objetivo. La metodología que emplearon comenzó primero con la selección de características.Segundo se completaron los datos faltantes, como los datos son recogidos por sensores muchos datos se pierden o son inconsistentes, por lo que se probaron 6 algoritmos de machine learning: SVM, KNN,Linear Discriminant, Quadratic Discriminant y Quadratic SMV para predecir los valores faltantes y siguientes de las secuencias de datos, tercero para evitar la divergencia de la predicción se normalizaron los datos, cuarto para predecir los valores de los siguientes pasos de una secuencia se abordo el uso de Support Vector Machine y finalmente el proceso de pronostico validando los valores de los múltiplos de tiempo utilizó la función predictedAndUpdateState de Matlab que permite anticipar pasos de tiempo consecutivos para actualizar el estado de la red en cada predicción.

**Conclusiones**:

- Conclusión 1. La aplicación desarrollada en este estudio permite abordar establecimiento de variables de tipo "predictores", lo que significa que el historial de esas variables serán tomado como series de tiempo que se procesan una por una con Support Vector Machine(SVM),K-Nearest Neighbors(KNN),Discriminante Lineal, Discriminante cuadratica, Support Vector Machine Cuadrática, obteniendo una KNN precisión de 99.7 %.

**Comentario**:De este documento tomamos como principal aporte en nuestro proyecto la metodología pues la forma de tratar con los datos se explica detalladamente por ejemplo: los datos de este estudio son recogidos por sensores por lo que se visibiliza el problema de datos perdidos e inconsistentes que necesitan cierto preprocesamiento para mantener la mayor cantidad de data disponible y así alcanzar mayor precisión, se utiliza la normalización de los datos para evitar la divergencia en la predicción, la selección de características se hace mediante Análisis de Componentes y Clasificación de Patrones. Finalmente en nuestro caso se utilizara algoritmos de aprendizaje profundo para la predicción de las series de tiempo en vez de hacer una clasificación por separa como en este antecedente.

- Propiedades físicas y químicas sujetas a degradación: Viscosidad a 40 y 100 C ,Humedad y contenido de Agua, Número total de ácidos/ Número total de bases(TAN/TBN), recuento de partículas en partes por millón(ppm).
- Técnicas de extracción de características Análisis de Componentes Principales.

Reveco Díaz María Ignacia, 2021, *"Análisis predictivo de activos mineros para obtención de intervalo de falla mediante algoritmos de machine learning"*, U. Chile. Con el objetivo de automatizar el análisis de aceite en la compañía minera Los Bronces de Anglo American, este estudio propone la creación de tres modelos de machine learning. El primer modelo se enfoca en la detección de anomalías en las muestras de aceite de los motores de camiones mineros. Se compararon varios algoritmos de detección de anomalías, como One-class SVM,

PCA Based Anomaly Detection, K-means y Robust Covariance, siendo este último el que logró la mayor precisión con un AUC cercano al 100

El segundo modelo fue entrenado para clasificar la causa de las fallas. Se evaluaron algoritmos de clasificación como Redes Neuronales, Árboles de Decisión y Regresión Logística Multivariada. Ninguno de estos modelos alcanzó un nivel de precisión satisfactorio. En conclusión, la autora recomienda no depender únicamente de datos de análisis de aceite para clasificar los modos de falla.

El tercer modelo se utiliza para calcular el Tiempo de Vida Remanente (RUL). Se emplearon diferentes algoritmos de regresión, como Boosted Decision Tree Regression, Decision Forest Regression, Linear Regression y Neural Network Regression. Este modelo de predicción de RUL se elaboró con los motores Cummins QSK60 HPI. Si bien este último algoritmo logró la mayor precisión con un 94.93 %, al ser probado con datos de motores Detroit, los resultados fueron negativos. Este resultado era previsible, ya que, en estricto rigor, un algoritmo debe ser entrenado y probado con datos de la misma naturaleza. **Conclusiones:**

- Conclusión 1.Los tres modelos propuestos en conjunto logran generar una predicción global del comportamiento de motores Diesel de camiones mineros. Los resultados de los modelos de detección de anomalías y predicción de RUL se consideran satisfactorios con un 99.8 %de precisión, a diferencia del modelo de clasificación de falla que al no considerar todas las causas de falla y confundirlas unas con otras al englobarlas en causas generales no agrega valor al proceso de Análisis de Aceite por lo que finalmente se descartó y finalmente la autora concluye que la clasificación hecha por un experto sigue siendo la mejor opción.
- Conclusión 2.En el contexto de este estudio, se observa que el registro de análisis de aceites proporciona una visión general del comportamiento de un motor diésel. Por lo tanto, recomienda no incluir otros parámetros adicionales, ya que el proceso de recopilación, extracción, procesamiento y etiquetado de datos es costoso. Aunque esto podría resultar en una ligera mejora en la precisión de las predicciones, no justificaría el esfuerzo adicional debido a la falta de un valor agregado significativo.

**Comentario:** Esta tesis sirvió de guía para definir que modelos eran necesarios para la detección de anomalías en nuestra operación minera, se aplicaron los algoritmos pero como se vera más adelante no se acoplaron lo suficiente a las exigencias de nuestra operación. Lo que se ha tomado de esta investigación para la nuestra son:

- Se exploro el uso de los algoritmos One-class SVM, PCA Based Anomaly Detection, K-means para la detección de anomalías.
- Esta tesis aporta la aplicación de varios tipos de etiquetado para las muestras, uno basado en la Norma ASTM D7720(Límites de normalidad) y según limites temporales.

- Abordar la predicción de RUL(Remaining Useful Life) mediante algoritmos supervisados de regresión utilizando las horas oeradas de cada motor, (dato obtenido de la misma base de datos de las muestras de aceite).

Wakiru et al.,2018,*"A review on lubricant condition monitoring information analysis for maintenance decision support"*, IEEE Xplore,USA.

El monitoreo de la condición de lubricación (LCM) desempeña un papel crucial no solo como sistema de alerta temprana en maquinaria, sino también en el diagnóstico y pronóstico de fallas dentro del mantenimiento basado en la condición (CBM). La LCM se considera una técnica esencial de monitoreo debido a la información detallada obtenida de las pruebas de lubricantes, ofreciendo una visión reflexiva sobre la condición de la maquinaria y su lubricante. La aplicación efectiva de LCM implica la evaluación y análisis de la información de análisis de lubricantes para extraer conocimientos, generando resultados interpretables y aplicables para respaldar las decisiones de mantenimiento. Este artículo tiene como objetivo llenar una brecha en la literatura al proporcionar una visión integral de los enfoques aplicativos para LCM, particularmente en el contexto del apoyo a las decisiones de mantenimiento. Se lleva a cabo una revisión sistemática de las tendencias de investigación recientes y el desarrollo de enfoques basados en LCM, centrándose en aplicaciones para el diagnóstico y pronóstico de equipos. Se contextualiza la revisión inicial de los aceites base, aditivos, muestreos y pruebas aplicados para LCM y apoyo a las decisiones de mantenimiento. Además, se clasifican las pruebas y parámetros de LCM en categorías como análisis fisicoquímicos, elementales, de contaminación y de aditivos. Los enfoques para analizar datos derivados de LCM se dividen en cuatro categorías: estadísticos, basados en modelos, inteligencia artificial y híbridos. Se exploran posibles mejoras para aumentar la confiabilidad en la toma de decisiones basada en los enfoques de LCM. El artículo concluye con una breve discusión sobre posibles tendencias futuras de LCM en el contexto de la toma de decisiones de mantenimiento. El estudio revisa enfoques aplicables para extraer conocimientos de datos de LCM con el fin de respaldar las decisiones de mantenimiento, abordando tanto los aspectos teóricos como prácticos de la lubricación.

**Conclusiones**: Este articulo recopila los 4 enfoques en los que se evalúa el análisis de aceite: Métodos estadísticos(Análisis de tendencias, Análisis de correlación, Análisis de regresión), Inteligencia Artificial, Híbrido, Basado en modelo. Destaca una tendencia creciente en la investigación sobre el tema. Se observa que el muestreo y monitoreo clásico fuera de línea prevalece en equipos de tecnología antigua debido a los altos costos de sistemas en línea. Sin embargo, equipos críticos requieren programas LCM automatizados, como sensores en línea y monitoreo en tiempo real. La toma de decisiones de mantenimiento se ve afectada por consideraciones técnicas y de costos, como la adquisición de máquinas preinstaladas, la instalación en máquinas existentes o el uso de muestreo y pruebas fuera de línea.

**Comentario**: Esta revisión contribuya a resolver lagunas en la teoría:

- El estudio destaca la importancia continua del muestreo y pruebas fuera de línea,

especialmente en análisis integrales y la clasificación de partículas. Se observa una alta dependencia en el uso de partículas elementales o de desgaste, vinculado directamente a las características de desgaste de la maquinaria. Aunque hay un crecimiento en el interés por categorías de parámetros múltiples, un 28 % de los artículos revisados, se señala que algunos parámetros raramente utilizados, como la nitración, oxidación y nitrooxidación, pueden tener un valor significativo para identificar efectos interactivos que afectan el desempeño del lubricante. El análisis de aditivos, a pesar de la limitada investigación aplicativa (3 % de los artículos revisados), se destaca por su potencial para revelar la composición del lubricante y facilitar el análisis de la causa raíz de fallas en la maquinaria.

- Se señala la importancia de considerar patrones en lugar de depender de un solo parámetro y se destaca el uso de técnicas analíticas por pares, como el análisis de correlación, para obtener mejores conocimientos. Sin embargo, se reconoce que el análisis de correlación solo puede no ayudar integralmente a los tomadores de decisiones, y las asociaciones derivadas pueden no ser funcionalmente viables.

Salem N.,2021, *"Oil condition monitoring and predicting actions using an Artificial Intelligence technique: Principal Components Analysis algorithm"*, IOP Conf. Ser.: Mater. Sci. Eng. 1169 012007.

El estudio se centra en el uso del algoritmo de Análisis de Componentes Principales (PCA) como un experimento favorable para prever datos de series temporales. Se destaca que, durante el uso, los aceites experimentan transformaciones químicas debido a factores como oxidación, degradación, contaminación y cambios en la viscosidad, afectando su vida útil. El objetivo principal es demostrar cómo anticipar datos de series temporales utilizando técnicas de Inteligencia Artificial, específicamente PCA, junto con modelos de máquinas de vectores de soporte (SVM), basándose en datos recopilados a través de sensores y determinaciones computacionales. Por lo general, un cambio de aceite se basa en criterios, a menudo fundamentados en determinaciones empíricas, cálculos que hacen referencia a las horas de operación, etc. Basándose en un análisis sólido, hay menos criterios basados en la composición química y los lubricantes existentes derivados de pruebas o registros en línea. Por lo general, los aceites se prueban en laboratorios especializados, siendo estas pruebas principalmente consumidoras de tiempo y energía. Debido a las pruebas realizadas por laboratorios y las operaciones tienen, por esta razón, desventajas relacionadas con la instantaneidad de la fecha de adquisición, cubriendo solo características básicas de degradación. Por ello se diseño un experimento, parte de un Programa de Investigación respaldado por empresas privadas de Rumania e Italia, en varias fases y abordó el comportamiento de un sistema mecánico. Se recopiló un conjunto de datos durante seis meses, validando 258,646 instancias para 19 parámetros operativos. Se evaluaron los elementos probados (parámetros fisicoquímicos) en una etapa inicial. A medida que se recopilaba información, se intercalaron "módulos de evento/error"basados en escenarios predefinidos para probar la velocidad de respuesta del sistema. La contribución de la Ïnteligencia Artificial"se volvió esencial para evaluar transformaciones infinitesimales de elementos multiparamétricos.

**Conclusiones**: Este articulo busca predecir los parámetros de las muestras para superar el problema de no contar con monitoreo en linea, esta estimación de series temporales están sujetas a errores debido a la naturaleza estadística de estas clases matemáticas. **Comentario**: Esta revisión contribuya a resolver el uso de PCA en datos de series temporales:

- El Análisis de Componentes Principales (PCA) es un procedimiento matemático cuantitativo riguroso para realizar tal simplificación. Es un método estable y eficiente para encontrar un algoritmo para estructurar un conjunto de datos multidimensional. Sin embargo, una desventaja principal del enfoque PCA es que el procedimiento y el resultado a menudo son difíciles de entender. La relación entre la entrada y la salida puede ser confusa, un ligero cambio en las entradas puede generar un resultado completamente diferente, y a menudo el usuario puede preguntarse si el PCA está haciendo lo correcto. El PCA genera un nuevo conjunto de variables llamadas componentes principales, y cada componente principal es una combinación lineal de las variables iniciales. La propiedad fundamental de estos componentes principales es que son ortogonales, por lo que no hay información redundante. El resultado cuantificable de este algoritmo es que todos los componentes principales en conjunto forman una base ortogonal para el espacio de datos.

Salem N.,2021, *"Classifying machinery condition using oil samples and binary logistic regression"*, IOP Conf. Ser.: Mater. Sci. Eng. 1169 012007.

En este estudio, se utiliza un conjunto de datos proporcionado por un laboratorio de análisis de aceite comercial centrado en el servicio para el mercado de equipos pesados móviles. El laboratorio procesa más de 400,000 muestras de aceite al año, provenientes de pruebas destinadas a determinar la condición del aceite y proporcionar información sobre la condición del subsistema. Las muestras de aceite provienen de camiones utilizados en la industria minera, específicamente del subsistema del motor diésel.

El análisis de las muestras de aceite genera datos para aproximadamente 30 variables de condición del aceite. Los resultados de todas las pruebas de muestra son revisados por analistas experimentados con la asistencia de un software que utiliza una herramienta de soporte para la toma de decisiones basada en métodos de control estadístico de procesos. Los analistas clasifican manualmente las muestras en una de cuatro clases, de 'A' a 'X'. Las muestras 'A' tienen propiedades de aceite dentro de límites aceptables y se consideran "buenas", indicativas de un subsistema "saludable". Las muestras 'X' muestran contaminación clara que requiere acción diagnóstica y correctiva inmediata para prevenir posibles fallas. Las muestras 'B' y 'C' representan etapas de deterioro entre 'A' y 'X'. En este estudio, las muestras 'B', 'C' y 'X' se describen colectivamente como muestras de aceite "no buenas.[ei]ndican subsistemas que se están deteriorando, o en el caso de 'X', gravemente deteriorados. Los subsistemas con muestras 'X' deben retirarse del servicio.

Se seleccionó un conjunto de variables explicativas importantes basadas en el conocimiento experto de los analistas de aceite, junto con un análisis exploratorio de datos. Las variables incluyen propiedades del aceite y dos variables relacionadas con la edad del aceite y

la clasificación de salud previa del subsistema. El conjunto de datos se extrae del subsistema del motor en una flota de 60 camiones todo terreno en un sitio minero, con 1332 muestras, siendo el 15.8

El objetivo del proyecto es determinar si el clasificador puede identificar correctamente las muestras 'A' y 'Not-A'. Solo las muestras 'Not-A' necesitan ser enviadas a los analistas para una evaluación más detallada, lo que optimiza la carga de trabajo de los analistas al dirigirla hacia las muestras más urgentes.

**Conclusiones**: Se evaluó el rendimiento de los modelos de Regresión Logística (LR), Red Neuronal Convolucional de Correlación Cruzada (CCNN) y Máquinas de Soporte Vectorial (SVM) para la clasificación de datos de muestras de aceite de camiones mineros. Se utilizaron conjuntos de datos e variables explicativas idénticos en los tres modelos. Al evaluar los resultados, se consideró la precisión total de la clasificación, la facilidad del proceso de construcción y actualización del modelo, y la transparencia en cómo los cambios en las variables afectan la salida del modelo para el analista.

El modelo de LR demostró la capacidad de clasificar aproximadamente el 89 % de las muestras de aceite correctamente. Utilizando un estudio de validación cruzada con el conjunto de datos analizado en el artículo, LR superó tanto a los modelos CCNN como a los SVM en términos de rendimiento predictivo. Además, el modelo de LR ofrece un marco paramétrico para estimar la probabilidad de la salud de una máquina, condicionada a variables explicativas cuyos efectos en la clasificación son fácilmente interpretados en términos de coeficientes de regresión o la razón de probabilidades (odds ratio). Por ejemplo, podemos observar en el modelo de LR la importancia de la historia pasada en la clasificación futura, coherente con las observaciones de los analistas. Además, cómo los cambios en los valores de variables individuales (por ejemplo, Fe) afectan la probabilidad de clasificación como 'A' o 'Not-A' también se puede explorar. Esta capacidad para determinar el efecto de variables explicativas individuales en la clasificación no es fácil de ver mediante los enfoques CCNN y SVM. La transparencia en cómo funciona la clasificación es importante para construir confianza en el enfoque de diagnóstico.

La capacidad de la industria para realizar la modelización y reticular las salidas a los usuarios a través de los sistemas empresariales existentes depende del personal calificado (interno o contratado) y la infraestructura informática. Es relativamente sencillo construir un modelo de LR y actualizarlo a medida que llegan nuevos datos, y no se requiere software comercial especializado. Los modelos de LR son probablemente un concepto accesible para los ingenieros que trabajan, a diferencia de las ANN y SVM que a menudo requieren capacitación especializada. Esta capacidad para gestionar la modelización internamente por parte del personal de la empresa de análisis de aceite es un factor importante al considerar la automatización del análisis.

**Comentario**: Esta revisión contribuya a resolver el uso de Regresión Logistica y SVM para clasificar muestras de aceite. Se menciona la importancia del historial pasado en la clasificación futura, lo cual es coherente con las observaciones de los analistas. Esta incorporación de

información histórica puede ser valiosa para la predicción de fallas en sistemas mecánicos. Además explica la relación entre Variables explicativas y las condiciones de aceite aporte que se utiliza en este estudio también.

Rosenkranz Andreas,2020, *"The Use of Artificial Intelligence in Tribology — A Perspective"*, Lubricants, 2020, vol. 9, no 1, p. 2.

Se han aplicado métodos de deep learning y machine learning en muchos campos de la tropología, desde el monitoreo de las condiciones, el diseño de composiciones de materiales y formulaciones de lubricantes o predicciones de espesor de lubricantes.

En el campo del monitorio de condiciones se ha probado: El entrenamiento de mapas de características multicapa y auto-organizado ANN para clasificar las partículas de desgaste por medio de características como ancho, largo, área de proyección, perímetro, diámetro representativo, elongación, reflectividad, etc. Usando datos microscópicos de experimentos de deslizamiento de ball-on-disk lubricados. Descubriendo que al entrenar con datos representativos se predijo correctamente la relación entre las características de las partículas de las muestras y las condiciones experimentales; también se determinó que la característica de auto-organización del mapa ANN clasifica data sin uso de data supervisada, por ello las características de las partículas pueden ser identificadas y pueden ser usadas para un monitoreo constante de condiciones.

También se experimentó con la aplicación de redes neuronales de múltiple capas entrenadas con back-propagation de errores supervisados (EBP) y una teoría de resonancia adaptativa no supervisada-2 (ART2) basada en red neuronal para la detección y diagnóstico de defectos localizados en rodamientos de bolas. Se uso un banco de prueba de señales de aceleración de vibraciones de un rodamiento con distintas condiciones de carga y velocidad para el entrenamiento de las redes. Los resultados determinaron que EBP y el modelo ART2 eran precisos para distinguir un rodamiento defectuoso con un 100 % fiabilidad, con la distinción de ART2 que es 100 veces más rápido y EBP puede clasificar correctamente los rodamientos de bolas en estados como bola o defecto en la pista, con una tasa de éxito de más del 95 %.

Posteriormente, también se presentó un enfoque con base en ANN para monitorear y clasificar el comportamiento de desgaste de cojinetes de deslizamiento lubricados. Utilizando un codificador automático para la detección de anomalías y señales de emisión acústica para entrenar una red neuronal convolucional para clasificar los modos de rodaje, lubricación insuficiente y contaminación por partículas del aceite. El resultado fue una detección con precisión de 97 % y sensibilidad de 100 % de lubricante contaminado.

**Conclusiones:**

- Conclusión 1. El uso de ANN es mas efectivo usando aprendizaje supervisado con el monitoreo de condiciones. En el trabajo a base del uso de Redes multicapa ANN indagan en usos especficos de monitoreo como la detección y diagnóstico de defectos localizados del rodamiento de esferas. Llegando a entrenar redes ANN con aprendizaje supervisa-

do (EBP) y una teoría de resonancia adaptativa no supervisada (ART2). Encontrando buenos resultados como la precisión para distinguir un rodamiento defectuoso con un 100 % de fiabilidad.

**Comentario:** Se ah estado experimentando con el uso de métodos de machine learning en campos de la tropología, en este caso resaltamos el uso de redes multicapa ANN con o sin aprendizaje supervisado para un monitoreo constante de componentes tropológicos y materiales lubricantes, con resultados satisfactorios.

### 2.1.2. Antecedentes Nacionales

Egoávil Méndez Diego,2019, *"Implementación de un programa de lubricación para aumentar la disponibilidad de los scoops Caterpillar R1600G en la Compañía Minera Casapalca"*, Universidad Tecnológica del Perú, Perú. La tesis tiene como objetivo implementar un programa de lubricación para incrementar la disponibilidad de los cargadores scoops Caterpillar modelo R1600G para la Compañía Minera Casapalca. Estos equipos son las máquinas que más inciden en la producción debido a que cargan, acarrean y descargan el mineral por lo tanto es imperativo que cuenten con una correcta lubricación ya que es un factor fundamental que afecta directamente la disponibilidad y confiabilidad de la maquinaria. En el año 2019 fecha de presentación de la tesis el área de mantenimiento de la Compañía Minera Casapalca no tenia implementadas estrategias y buenas prácticas en lubricación. Por lo que la tesis explica detalladamente todo lo referente a la tribología ciencia que estudia la interacción entre las superficies en movimiento relativo y los fenómenos asociados, como la fricción, el desgaste y la lubricación. En la cuestión del Análisis en si propone un Análisis Descriptivo y estadístico. Se sustenta en su marco teórico que el 70 % de las fallas se relacionan directamente con una deficiente lubricación o contaminación del fluido lubricante. La calidad de los datos fue una limitante para el proyecto debido a que la compañía no contaba con ningún software de mantenimiento. De acuerdo con E., 1995 las causas de pérdida de utilidad de los equipos en la industria son: 15 % por obsoletos, 15 % por descompostura y 70 % por deterioro de superficie, siendo esta última en la que la lubricación juega un papel crucial para incrementar la vida útil de la maquinaria. Muchas empresas, instituciones y fabricantes de equipo de clase mundial concuerdan en que la contaminación con partículas sólidas en el lubricante representa hasta el 80 % de las causas raíz de falla de la maquinaria.

**Conclusiones**:

- Conclusión 1.Luego de implementar el programa de lubricación se logró incrementar la disponibilidad de los scoops Caterpillar de 89 % a 94 % en los primeros tres meses de ejecutado el programa. Respecto a la MTTR se mantuvo con respecto al promedio durante 2018, la mejora del MTBF incidió directamente en el aumento de la disponibilidad.

**Comentario**: En este trabajo se detalla la importancia de los cargadores scoops Caterpillar

R1600G que en la operación minera Casapalca cumplen las mismas funciones que la Palas 6060fs de nuestro proyecto. Esta tesis explica con mucho detalle como es el Análisis de Aceite en equipos mineros refleja la poca intervención de la tecnología dentro del rubro minero sobre todo en mineras pequeñas dado que incluso algunas ni siquiera cuentan con programas de monitoreo además nuestros modelos demuestran que es posible automatizar el análisis de aceite a medida sin requerir mucho costo o recurrir a una empresa terciaria. Comparado con otros rubros como el sector financiero, de salud, etc. la minería recién empieza a despegar en innovaciones tecnológicas. De este estudio se toma las definiciones de tribología y su aplicación en mantenimiento minero.

Sánchez Gonzales Jesús Rodolfo,2022, *"Aplicación De Monitoreo De Condiciones Para El Diagnóstico De Fallas De Motor y Transmisión En Cargadores 966h Mediante Uso De Machine Learning"*, Universidad Católica de Santa María, Perú.

El mantenimiento basado en condición cobra mayor relevancia ante los elevados costos de reparación o mantenimiento correctivo que el mantenimiento preventivo no puede evitar, tiendo un riesgo residual de una posibilidad de falla, ya que las condiciones de operación de cada flota son diferentes a cada país, y en algunos casos a cada región. El estudio propone una forma de aplicación en el monitoreo de condición basado en elementos de desgaste (cobre, silicio, fierro, plomo) para revisar el desgaste en el aceite para mejorar el nivel de alerta en los componentes de motor y transmisión de Cargadores 966 Caterpillar, el algoritmo más viable para este estudio fue el árbol de clasificación utilizando la herramienta de análisis estadístico SPSS.
**Conclusiones**:

- Conclusión 1. La nueva propuesta se asocia a nuevos límites condenatorios (se ajustó los limites condenatorios del motor para: Fe a 43 unidades y Cu en el rango de 6 y 11 unidades), convirtiéndose en políticas de trabajo de monitoreo de condición en la empresa. Las condiciones del fabricante con la propuesta, difieren principalmente en Caterpillar que brinda un sistema unidimensional en sus especificaciones de desgaste de elementos, en comparación con la propuesta que trae reglas matriciales de los límites condenatorios.

**Comentario**: Este estudio propone una aplicación de monitoreo de condición desde el punto de vista de la ingeniería mecánica, por lo que para un experto de mantenimiento es importante saber que es lo que hace el algoritmo el árbol de decisión es un algoritmo sencillo y muy explicativo que muestra las reglas que está siguiendo para llegar a una conclusión este punto es lo que hace a este algoritmo de los utilizados en la bibliografía revisada.

## 2.2. Bases Teóricas

### 2.2.1. Proceso minero

El proceso productivo de la minera de la cual se obtuvieron los datos es el siguiente:

1. Extracción: Las máquinas perforadoras insertan explosivos que luego de su detonación dispersan escombros los cuales son cargados a los camiones mineros con destino a la chancadora.
2. Chancado: El material minado llega al chancador con diversos tamaños y el objetivo de esta parte del proceso es reducir el tamaño de la roca hasta un mínimo de 18cm.
3. Molienda: Luego del chancado, el mineral pasa a la planta concentradora, donde se produce la molienda. Aquí, se espera reducir la roca triturada hasta los 0,18mm.
4. Flotación: Se separan los sulfuros de cobre y otros elementos de la roca original. También separamos el molibdeno del cobre.
5. Filtración: Se filtra el concentrado de cobre y llevamos el producto resultante al almacén.
6. Transporte: Se utiliza un sistema bimodal (camiones y tren) para conducir el concentrado de cobre hasta el puerto de Matarani para su exportación.

### 2.2.2. Equipo de carguío

La cantidad de equipos en una operación minera, como camiones de transporte, perforadoras y palas, está determinada por la planificación y estrategia operativa de la mina, así como por las necesidades específicas de la operación y las características del yacimiento. Pero por diversas circunstancias en nuestra operación hay menos palas en comparación con otros equipos por ello las palas son consideradas prioritarias en la operación minera. Las palas son equipos utilizados para cargar material en los camiones de transporte en una mina a cielo abierto. Esta tarea es esencial para el flujo continuo del proceso de extracción y transporte del material. **Concepto de doble motor**: Las Palas CAT 6060FS tienen dos motores en el caso de la operación minera se denominan "MODIZ.ªl motor izquierdo y "MODDE.ªl motor derecho si un motor deja de funcionar el otro aegura a continuidad de la operación procurando mantener la productividad y garantizar la seguridad de los operadores.

### 2.2.3. Motor Diesel

El motor diesel es una máquina de combustión que aspira aire y lo comprime hasta alto nivel, sin la necesidad de chispa para el encendido del combustible. Los elementos del

Tabla 2.1: Especificaciones técnicas Pala hidráulica CAT 6060 FS

| **Especificaciones técnicas** | **CAT 6060 FS** |
|---|---|
| Modelo de motor | DIESEL 2 × Cat® 3512C |
| Potencia bruta | 2.240 kW 3.000 hp |
| Capacidad del cucharón: pala frontal | 34,0 m3 44,5 yd3 |
| Carga útil del cucharón | 61 toneladas métricas 67 tons EE.UU. |

Fuente: CAT 6060FS Pala Hidráulica

Tabla 2.2: Palas CAT 6060FS y componentes operando

| **Nombre de pala** | **Motor Derecho (MODE)** | **Motor Izquierdo (MOIZ)** |
|---|---|---|
| SH001 | ✓ | ✓ |
| SH002 | ✓ | ✓ |
| SH003 | ✓ | ✓ |
| : | ✓ | ✓ |
| SH010 | ✓ | ✓ |

Fuente: Elaboración propia

motor que forman su constitución pueden dividirse en los siguientes órganos más elementales: Elementos fijos, que son los que componen la estructura externa de motor, cuya misión es la de alojar, sujetar y tapar a otros elementos del conjunto, como: el bloque del motor, la culata (cabeza de cilindros) y su junta, el Carter y su junta, la tapa de balancines. (Reveco Díaz, 2019).

### Modos de falla en motores diesel

Para seleccionar las paradas que pueden ser predecidas se utiliza la norma ISO 14224 es una norma internacional que establece lineamientos para la especificación, recolección y aseguramiento de la calidad de los datos de confiabilidad y mantenimiento en la industria petrolera, petroquímica y de gas natural. La norma tiene como objetivo proporcionar un conjunto común de definiciones, sistemas de codificación y clasificación de fallas, y prácticas

recomendadas para la recopilación y análisis de datos de confiabilidad y mantenimiento, con el fin de mejorar la toma de decisiones en la gestión de activos y el mantenimiento de equipos. La norma también proporciona un marco para la comparación de la confiabilidad de los activos entre diferentes instalaciones y empresas. Según esta norma los tipos de falla de motores diesel son: Las fallas sintomáticas resaltadas de color amarillo son aquellas que

Tabla 2.3: Pruebas para análisis de desgaste

| Modo Falla | Cód. | Definición | Descripción |
|---|---|---|---|
| Operacional | FTS | No arranca al momento de encender | Incapacidad para arrancar el motor |
| Operacionales | STP | No se detiene al momento de apagar | Incapacidad para detener el motor o proceso incorrecto de interrupción |
| Operacional | SPS | Falsa parada | Interrupción inesperada del motor |
| Operacional | OWD | Opera sin accionar | Arranque no deseado |
| Operacional | BRD | Colapso | Daños graves ( agarrotamiento , roturas, explosión, etc.) |
| Sintomáticos | HIO | Alta energía de salida | Velocidad excesiva/energía de salida por encima de lo especificado |
| Sintomáticos | LOO | Baja energía de salida | Energía de salida por debajo de lo especificado |
| Sintomáticos | ERO | Energía de salida errática | Oscilante o fluctuante |
| Sintomáticos | ELF | Fuga externa-combustible | Fuga de gas combustible o diesel |
| Sintomáticos | ELU | Fuga externa-medio de servicio | Aceite lubricante, refrigerante, etc. |
| Sintomáticos | INL | Fuga interna | Por ejemplo, fuga de agua del refrigerador interno |
| Sintomáticos | VIB | Vibración | Vibración excesiva |
| Sintomáticos | NOI | Ruido | Ruido excesivo |
| Sintomáticos | OHE | Sobrecalentamiento | Temperatura excesiva |
| Sintomáticos | STD | Deficiencia estructural | Por ejemplo, roturas en la tapa o soporte de cilindro |
| Sintomáticos | STD | Deficiencia estructural | Por ejemplo, roturas en la tapa o soporte de cilindro |

Fuente: (Reveco Díaz, 2019)

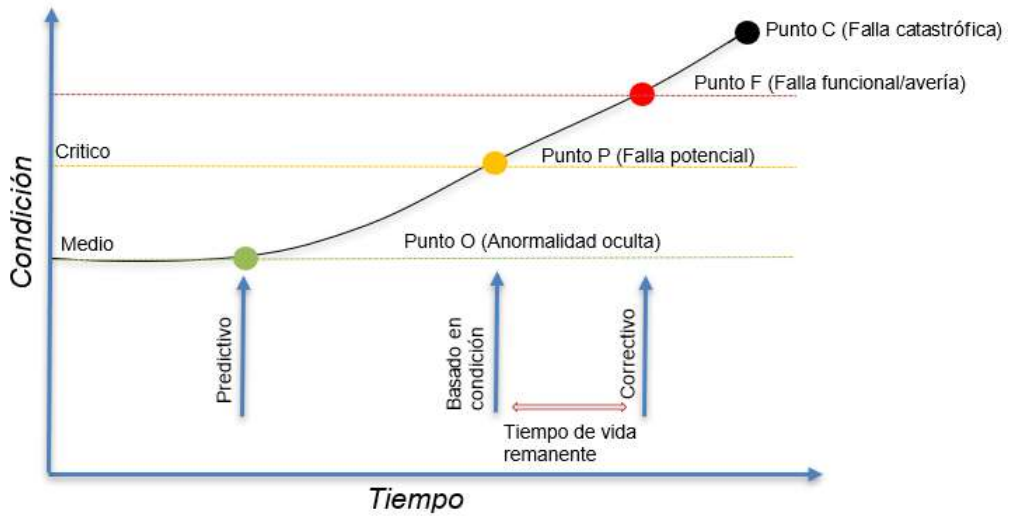
se tomaran en cuenta en este proyecto.

### 2.2.4. Mantenimiento Basado en Condición

El mantenimiento basado en Monitoreo de Condición (MBC) tiene como objetivo principal mejorar la confiabilidad y disponibilidad de los equipos, reducir los costos de mantenimiento y minimizar el tiempo de inactividad no planificado. Al implementar un programa de MBC, se busca monitorear de manera continua el estado de los equipos mediante la medi-

ción de parámetros físicos, químicos y operativos relevantes para detectar signos tempranos de falla o deterioro y tomar medidas preventivas o correctivas antes de que ocurra una falla catastrófica. De esta manera, se puede prolongar la vida útil de los equipos, mejorar la seguridad y reducir los costos de mantenimiento y reparación. Para implementar esta estrategia, es común utilizar técnicas de monitoreo continuo, como la medición de vibraciones, la termografía, el análisis de aceite, entre otros. A partir de estos datos, se puede construir una curva de evolución temporal de condición que representa la tendencia del estado del equipo a lo largo del tiempo. La curva de evolución temporal de la condición, también conocida como curva de degradación se utiliza para visualizar cómo cambia la condición de un componente o sistema a lo largo del tiempo. Esta curva representa la relación entre una medida de condición y el tiempo de operación del componente o sistema.(Quatrini et al., 2020)

Figura 2.1: Curva de evolución temporal de la condición




Fuente: Elaboración Propia.

La empresa de servicios de mantenimiento SKF encontró que el mantenimiento basado en condición puede reducir los costos de mantenimiento en un 30-40 % y aumentar la vida útil de los equipos en un 20-50 % en comparación con el mantenimiento preventivo basado en intervalos de tiempo predefinidos(Caroline, s.f.)

### 2.2.5. Tribología

Los diccionarios definen la tribología como la ciencia y la tecnología de las superficies que interactúan en movimiento relativo y de los temas y prácticas relacionados. La tribología es el arte de aplicar el análisis operativo a problemas de gran importancia económica, como la fiabilidad, el mantenimiento y el desgaste de los equipos técnicos, desde las naves espaciales hasta los electrodomésticos. (Bhushan, 2013). El análisis de aceite es una de las principales técnicas utilizadas en tribología para evaluar el desempeño y el estado de las máquinas y los componentes mecánicos.

#### 2.2.5.1. Análisis tribologico

El análisis de aceite es una técnica de monitoreo de la condición del lubricante y de la maquinaria en la que se aplica. Consiste en analizar la composición físico-química del aceite utilizado en la maquinaria y detectar las impurezas, partículas metálicas, productos de oxidación y otros contaminantes que puedan haberse acumulado en él. Esta técnica es ampliamente utilizada en la industria para identificar posibles fallas en los equipos de manera temprana, lo que permite programar su mantenimiento preventivo antes de que ocurra un fallo catastrófico. (Mortier et al., 2020). El análisis de aceites tiene dos objetivos principales:

- Control de degradación del lubricante: La degradación es el deterioro de las propiedades iniciales del aceite, y puede provocar fallas en los sistemas mecánicos. Se observa su degradación mediante sus propiedades mecánicas, como viscosidad, contenido de agua, oxidación, nivel de acidez, sulfatación, corrosión, agotamiento de aditivos, etc. La viscosidad de los aceites es medida por un viscosímetro, mientras que la lectura de componentes aditivos en ppm (partículas por millón) es realizada mediante FTIR (espectrometría de transmisión de infrarrojo con transformada de Fourier). En motores Diesel de camiones, es común encontrar los siguientes elementos como aditivos: potasio, molibdeno, fósforo, bario, magnesio, calcio, zinc y boro.
- Monitoreo del daño mecánico de componentes: A partir de este monitoreo es posible encontrar evidencia de diversas fallas. Mediante un espectrómetro es posible medir las concentraciones de elementos resultantes de elementos contaminantes, que pueden clasificarse en elementos de desgaste interno o elementos de contaminación externa. Los elementos de desgaste interno resultan de una lubricación defectuosa, lo que implica que exista fricción entre los materiales y finalmente desgaste. En motores Diesel de camiones, es común encontrar los siguientes elementos de desgaste interno: fierro, níquel, cobre, cromo, aluminio y plomo. Los elementos de contaminación externa resultan de un sellado defectuoso del sistema de lubricación. Es común encontrar como elementos de contaminación externa en motores Diesel: el sodio, que usualmente significa una filtración del fluido refrigerante, o silicio, que comúnmente significa una filtración de polvo al sistema de lubricación.
- Determinar el nivel de limpieza del lubricante
- Inspeccionar los contaminantes.
- Asegurarse que el lubricante en servicio es el correcto.
- Monitorear la degradación del lubricante.
- Optimizar los intervalos de cambio de aceite.

#### 2.2.5.2. Pruebas de Análisis de aceite

Las pruebas se dividen en tres categorías: Análisis de contaminación: Se realiza para determinar si el lubricante se adultera con materiales líquidos o sólidos, comprometiendo su rendimiento.

Tabla 2.4: Pruebas para análisis de contaminación

| Enfoque | Prueba |
|---|---|
| Predictivo | ▪ Determinación de concentración de combustible en el lubricante<br>▪ Determinación de concentración de hollín en el lubricante<br>▪ Conteo de partículas sólidas en el lubricante<br>▪ Determinación de contaminantes por método atómico: Si, Al, B, Na, K, V, Ba (otros metales del proceso o ambiente) |

Análisis de desgaste: Permite analizar elementos que constituyen la metalurgia de los componentes, por ejemplo: hierro, aluminio, plomo, cobre, cromo, plata y estaño.

Tabla 2.5: Pruebas para análisis de desgaste

| Enfoque | Prueba |
|---|---|
| Predictivo | ▪ Determinación de concentración de partículas ferromagnéticas por el método de Ferrografía de lectura directa<br>▪ Determinación de concentración de glicol en el lubricante<br>▪ Concentración de partículas ferromagnéticas por el método cuantificador de partículas ferrosas- índice PQ<br>▪ Determinación de la concentración de metales de desgaste por el método atómico: Fe, Cu, Pb, Sn, Al, Sb, Ni, Cd, Cr, Ag |

Fuente: (**empty citation**)

Análisis de aditivos: Incluye productos químicos que confieren nuevas propiedades específicas, mejoran las propiedades del aceite base existente.

Tabla 2.6: Pruebas para análisis de aditivos

| Enfoque | Prueba |
|---|---|
| Proactivo | ▪ Viscosidad 100°C-. La viscosidad reporta resultados a 100ºC o 40ºC y es la más importante caracteristica física de un lubricante, ya que deben tener y conservar la propiedad de fluir y proteger las partes de la maquinaria a diferentes temperaturas y condiciones. La prueba estándar de Viscosidad Cinemática es la ASTM 0455.<br>▪ Número Ácido Total (TAN): Indica la acidez relativa del aceite.Un incremento repentino del TAN es considerado como un indicador de condiciones anormales de operación (tal vez sobrecalentamiento).<br>▪ Número Básico Total (TBN): Inverso del TAN es una prueba de titulación utilizada para determinar la reserva alcalina del lubricante. El TBN es generalmente aceptado como indicador de la habilidad del lubricante para neutralizar ácidos peligrosos formados por la combustión de productos en motores de combustión interna.<br>▪ Oxidación, Nitración, Sulfatación, Determinación de aditivos por el método atómico: Zn, P,Mg,Ca,Mo, B, Ti |

### 2.2.6. Machine learning

Christopher Bishop define Machine Learning o Aprendizaje Automático como un subcampo de la informática que se centra en el diseño de algoritmos que pueden aprender de los datos y hacer predicciones sobre ellos. Una característica clave del aprendizaje automático es la capacidad de los algoritmos para mejorar su rendimiento en una tarea con experiencia o entrenamiento, sin estar explícitamente programado". (Bishop, 2006). Otra definición de Kevin Murphy indica lo siguiente: El aprendizaje automático es un conjunto de métodos que pueden detectar automáticamente patrones en los datos y luego usar los patrones descubiertos para predecir datos futuros o realizar otros tipos de toma de decisiones bajo incertidumbre". (Murphy, 2012) Esta definición enfatiza la capacidad del aprendizaje automático para aprender automáticamente patrones a partir de datos y utilizar.

#### 2.2.6.1. Tipos de Machine Learning

1. **Aprendizaje supervisado:** En el enfoque de Aprendizaje Supervisado el algoritmo aprende de los datos etiquetados, lo que significa que cada punto de datos está asociado con una etiqueta o variable objetivo conocida. El objetivo es aprender a asignar las salidas y desde las entradas x, en su configuración más simple tenemos: dado un conjunto etiquetado de pares de entrada-salida

$$D = \{(x_i, y_i)\}_{i=1}^{N} \tag{2.1}$$

   Donde $D$ se llama conjunto de entrenamiento y $N$ es el número de ejemplos de entrenamiento cada entrada de entrenamiento, $x_i$ es un vector de números de dimensión $D$, que representa, por ejemplo, la altura y el peso de una persona. Estos se denominan características, atributos o covariables. De manera similar, la forma de la variable de salida o respuesta puede, en principio, ser cualquier cosa, pero la mayoría de los métodos asumen que $y_i$ es una variable categórica o nominal de algún conjunto finito, $y_i \in \{1, \ldots, c\}$ (como hombre o mujer), o que $y_i$ es un escalar de valor real (como el nivel de ingresos). Cuando $y_i$ es categórico, el problema se conoce como clasificación o reconocimiento de patrones, y cuando $y_i$ tiene un valor real, el problema se conoce como regresión. (Murphy, 2012)

2. **Aprendizaje no supervisado:** En el enfoque de Aprendizaje no Supervisado el modelo se entrena con datos no etiquetados sin ninguna variable objetivo predefinida. Entonces los únicos datos de entrada son

$$D = \{(x_i)\}_{i=1}^{N} \tag{2.2}$$

   y el objetivo es encontrar "interesantes patrones" en los datos. Esto a veces se llama descubrimiento de conocimiento. Este es un problema mucho menos definido, ya que no se nos dice qué tipos de patrones buscar, y no hay una métrica de error obvia para usar (a diferencia del aprendizaje supervisado, donde podemos comparar nuestra predicción de $y$ para un $x$ dado con el valor observado).(Murphy, 2012)

3. **Aprendizaje semi supervisado:** El aprendizaje semi supervisado es un enfoque de aprendizaje automático que utiliza tanto datos etiquetados como no etiquetados para entrenar un modelo. En el aprendizaje semi supervisado, los datos no etiquetados se utilizan para aprender características y estructuras subyacentes, mientras que los datos etiquetados se utilizan para guiar el entrenamiento y mejorar la precisión del modelo. (Chapelle et al., 2006)

#### 2.2.6.2. Modelos para detección de anomalías

##### Covarianza Robusta

La covarianza robusta es una medida estadística utilizada para estimar la matriz de covarianza de un conjunto de datos que puede contener valores atípicos. A diferencia de los

métodos clásicos, como la matriz de covarianza empírica, que pueden verse significativamente afectados por la presencia de valores atípicos, los métodos de covarianza robusta están diseñados para proporcionar estimaciones más resistentes a estas desviaciones.Rousseeuw y Hubert, 2018

Formalmente, dada una muestra de datos $X = \{x_1, x_2, \ldots, x_n\}$ con $x_i \in \mathbb{R}^p$, la matriz de covarianza empírica $\mathbf{S}$ se calcula como:

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^\top, \tag{2.3}$$

donde $\bar{x}$ es el vector promedio:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2.4}$$

El problema con esta estimación es su sensibilidad a los valores atípicos, lo que puede llevar a estimaciones sesgadas y varianzas infladas.

Los métodos de covarianza robusta, como el *Minimum Covariance Determinant* (MCD) o el *Minimum Volume Ellipsoid* (MVE), buscan resolver este problema seleccionando un subconjunto de los datos que minimice el impacto de los valores atípicos.

**Estimador de Covarianza Mínima Determinante (MCD)**

El estimador MCD calcula un subconjunto de $h$ observaciones cuya matriz de covarianza tiene la determinante más pequeña. Este subconjunto, conocido como conjunto mínimo, se utiliza para estimar la matriz de covarianza y el vector de medias robustos:

$$\hat{\mu}_{\text{MCD}} = \frac{1}{h} \sum_{x_i \in H} x_i, \tag{2.5}$$

$$\hat{\Sigma}_{\text{MCD}} = \frac{1}{h-1} \sum_{x_i \in H} (x_i - \hat{\mu}_{\text{MCD}})(x_i - \hat{\mu}_{\text{MCD}})^\top, \tag{2.6}$$

donde $H$ es el conjunto mínimo que minimiza la determinante de la matriz de covarianza.

**Propiedades de la Covarianza Robusta**

Los métodos de covarianza robusta tienen las siguientes propiedades:

- **Resistencia a valores atípicos:** Proporcionan estimaciones fiables incluso en presencia de datos fuera de lo común.
- **Consistencia:** Las estimaciones son consistentes cuando el tamaño de la muestra crece.
- **Eficiencia:** Son menos eficientes que los métodos clásicos en datos sin valores atípicos, pero significativamente mejores en datos con valores atípicos.

**Aplicaciones**

La covarianza robusta se utiliza en diversas aplicaciones, como:

- **Detección de Anomalías:** Identificar valores atípicos en conjuntos de datos multivariados.
- **Análisis Discriminante:** Construir clasificadores robustos a valores atípicos.
- **Análisis Financiero:** Modelar el riesgo y las relaciones entre activos financieros.

**SVM One Class**

SVM (siglas del inglés Super Vector Machine) es un algoritmo supervisado de Machine Learning utilizado es clasificaciones lineales o no lineales, regresiones y detección de anomalías.Géron, 2017 SVM One Class es una extensión de las máquinas de vectores de soporte utilizadas principalmente para el problema de detección de anomalías, donde el objetivo es identificar los datos que se desvían de algún concepto de normalidad. A diferencia de las SVM tradicionales, que requieren datos de múltiples clases para el entrenamiento, SVM One-Class se entrena solo con datos que se suponen normales, buscando definir una frontera que englobe la mayor parte de estos datos. La SVM One-Class fue introducida por Schölkopf et al. en 2001. La idea central es transformar el problema de detección de anomalías en un problema de clasificación donde se busca separar la mayoría de los datos normales del origen (o de los puntos de outliers) con el máximo margen posible. La formulación matemática es la siguiente: Función Objetivo:

$$\min_{w,\xi_i,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \tag{2.7}$$

sujeto a:

$$w \cdot \phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{2.8}$$

donde:

- $w$: el vector normal al hiperplano.
- $\xi_i$: las variables de holgura para cada punto de datos $i$, que permiten cierto grado de violación del margen (soft margin).
- $\rho$: el umbral de decisión que se ajusta durante el entrenamiento.
- $\phi$: representa la función de transformación del kernel, que mapea los puntos de entrada a un espacio de características de alta dimensión, potencialmente infinita, donde es más fácil asegurar una separación.
- $\nu$: un parámetro (0 ¡$\nu \leq 1$) que controla tanto el número de vectores de soporte como el límite superior en la fracción de outliers.

**Isolation Forest**

Isolation Forest, creado por T. et al., 2008, es un algoritmo de detección de anomalías basado en el concepto de identificar anomalías en lugar de perfilar puntos de datos normales. El método se construye sobre la base de árboles de decisión, o más específicamente, árboles de aislamiento, que son árboles binarios propios donde cada nodo tiene exactamente cero o dos nodos hijos. Un árbol binario propio es un árbol en el que cada nodo, excepto las hojas, tiene dos hijos.

La estrategia general primero consiste en codificar un árbol, luego construir un bosque de árboles (ensamblaje) y, finalmente, medir cuán lejos llega una instancia determinada en cada árbol y determinar si es un valor atípico. Para detectar anomalías, el método ordena los puntos de datos según sus longitudes de camino o puntuaciones de anomalía, y las anomalías son puntos que están clasificados en la parte superior de la lista.T. et al., 2008 definen las longitudes de camino y las puntuaciones de anomalía de la siguiente manera. Definición de longitud de camino: La longitud de camino h(x) de un punto x se mide por el número de aristas que x atraviesa en un árbol de aislamiento desde el nodo raíz hasta que el recorrido termina en un nodo externo. Definición de puntuación de anomalía: Dado que los árboles de aislamiento tienen una estructura equivalente a los árboles de búsqueda binaria o BSTs, la estimación de la media de h(x) para terminaciones en nodos externos es la misma que la longitud media de camino de una búsqueda fallida en BST. Dado un conjunto de datos de n instancias, la longitud media de camino de una búsqueda no exitosa es:

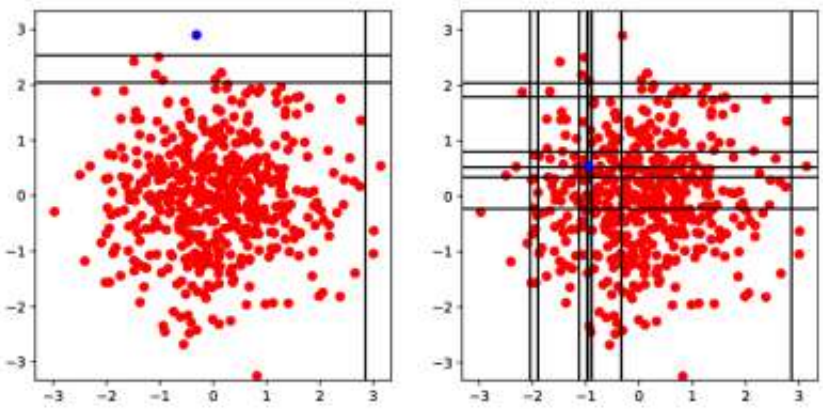
$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2.9}$$

donde H(i) es el número armónico. Este puede ser estimado por ln(i)+ 0.5772156649 ln(i)+ 0.5772156649 (la constante de Euler-Mascheroni). Dado que c(n) es el promedio sobre h(x) dado n, se puede usar para normalizar h(x). La puntuación de anomalía s de una instancia x se define como:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.10}$$

$E[h(x)]$ es el valor esperado de la longitud del camino para todo el bosque y el exponente $E[h(x)]/c(n)$ es una medida de cuánto se desvía la longitud promedio del camino para el punto x en comparación con la longitud promedio del camino para cualquier punto. Si la puntuación de anomalía s en la Ecuación 2.5 está cerca de 1, entonces es probable que x sea una anomalía. Si s es menor que 0.5, entonces es probable que x sea una observación normal. Si para una muestra dada todas las instancias se les asigna una puntuación de anomalía alrededor de 0.5, entonces no hay una anomalía distintiva entre todas las instancias.

Figura 2.2: Ilustración de dos consultas de bifurcación diferentes en un bosque de aislamiento.



Fuente:(OZAKI, s.f.) with Python.

El bosque de aislamiento no depende de ninguna medida basada en la distancia o la densidad para identificar anomalías, por lo que es rápido y poco costoso desde el punto de vista informático. para identificar anomalías, por lo que es rápido y poco costoso desde el punto de vista computacional. Los bosques de aislamiento también son escalables, ya que sus requisitos computacionales y de memoria son bajos en comparación con otras alternativas comunes [20]. Es fácil de implementar en Python y no requiere etiquetas. Python y no requiere etiquetas, por lo que es una opción adecuada para esta tesis.

**Principal Componente Analysis**

Dado un conjunto de variables correlacionadas presentadas en forma de matriz amplia,donde las filas son las observaciones y las columnas las variables, PCA aplica una transformación ortogonal para convertir este conjunto de datos en un grupo de valores linealmente no correlacionados, los Componentes Principales (PC) (Kimera & Nangolo, 2022). Es decir, el PCA transforma variables discretas en coeficientes no correlacionados que pueden interpretarse geométricamente como las proyecciones de las observaciones hacia los PC (de Carvalho Michalski & de Souza, 2022)

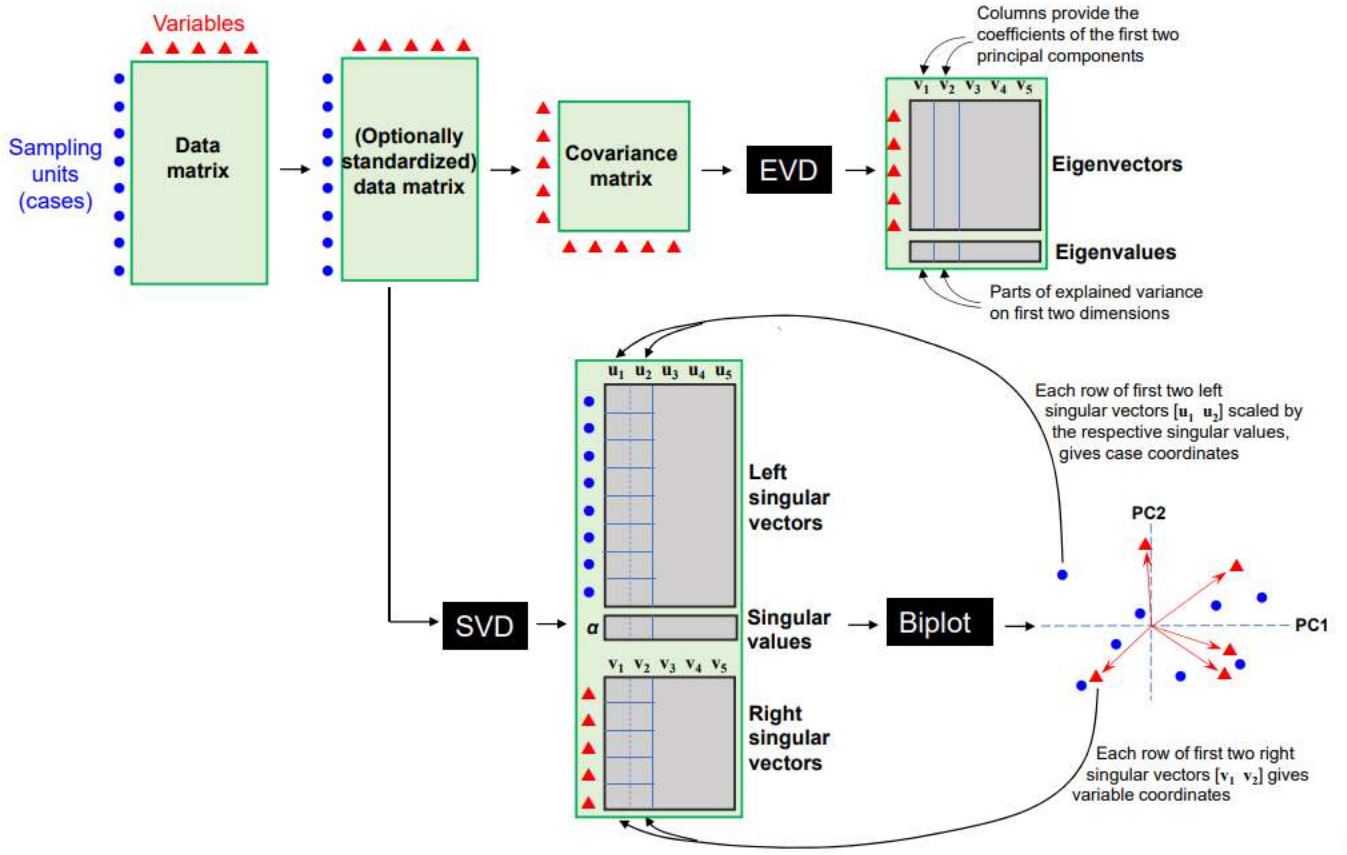
Los Componentes Principales (PC) se organizan en función de su varianza, comenzando con el componente que posee la mayor varianza y continuando en orden decreciente. El primer componente principal captura la mayor cantidad de información del conjunto de datos. En consecuencia, el segundo componente, que es ortogonal al primero, retiene la mayor parte de la información que el primer componente no captura. Este proceso se repite para los componentes subsiguientes, donde cada uno almacena la información restante no retenida por sus predecesores. El Análisis de Componentes Principales (PCA) pone énfasis en la varianza, manteniendo en cuenta las covarianzas y correlaciones. Generalmente, en análisis subsecuentes, se consideran solo los dos o tres primeros componentes, ya que son los que tienen la varianza más alta. Aspectos claves de PCA:

**Input:** Data $X$, number of trees $T$, subsample size $s$
**Output:** Anomaly scores for each instance in $X$
**procedure** IsolationForest($X, T, s$)
    Initialize forest $F$ as an empty list
    **for** $t = 1$ to $T$ **do**
        $S_t \leftarrow$ random sample of $s$ instances from $X$
        $Tree_t \leftarrow$ IsolationTree($S_t, 0$)
        Add $Tree_t$ to $F$
    **end for**
    **return** AnomalyScore($X, F$)
**end procedure**

**procedure** IsolationTree($S, e$)
    **if** $|S| \leq 1$ or depth limit reached **then**
        **return** leaf node with size $|S|$
    **else**
        Randomly select a feature $i$ and split value $p$ from range of $i$ in $S$
        $S_{\text{left}} \leftarrow \{x \in S : x_i < p\}$
        $S_{\text{right}} \leftarrow \{x \in S : x_i \geq p\}$
        Create a node with feature $i$, split $p$, left child IsolationTree($S_{\text{left}}, e+1$), right child IsolationTree($S_{\text{right}}, e+1$)
        **return** node
    **end if**
**end procedure**

**procedure** AnomalyScore($X, F$)
    Initialize scores as zero vector of length $|X|$
    **for** each instance $x$ in $X$ **do**
        $h \leftarrow 0$
        **for** each tree $T$ in $F$ **do**
            $h \leftarrow h+$ path length of $x$ in $T$
        **end for**
        scores[$x$] $\leftarrow$ exponential of average $h$
    **end for**
    **return** scores
**end procedure**

Figura 2.3: Vista esquemática del flujo de trabajo del PCA. La definición de los componentes principales (PCs) se puede obtener mediante la descomposición en valores propios (EVD) de la matriz de covarianza de las variables.




- Transformación Lineal: PCA transforma las variables originales en un nuevo conjunto de variables, las componentes principales, que son ortogonales (independientes) entre sí.
- Varianza Máxima: Las componentes principales se ordenan de manera que la primera captura la mayor varianza posible dentro del conjunto de datos, la segunda captura la mayor varianza restante bajo la restricción de ser ortogonal a la primera, y así sucesivamente.
- Reducción de Dimensionalidad: A menudo, las primeras pocas componentes principales pueden capturar una gran parte de la varianza total de los datos originales. Por lo tanto, se pueden usar para reducir el número de dimensiones del conjunto de datos con una pérdida mínima de información.
- Aplicaciones: PCA es útil en el análisis exploratorio de datos, para visualizar relaciones genéticas y patrones en grandes conjuntos de datos, en el preprocesamiento de datos para algoritmos de aprendizaje automático, y en la compresión de imágenes, entre otros.

- Método: Matemáticamente, PCA implica el cálculo de la matriz de covarianza de los datos, seguido de la aplicación del teorema espectral para encontrar sus vectores propios (componentes principales) y valores propios (que indican la cantidad de varianza capturada por cada componente).

Los métodos de clustering tienen como objetivo agrupar puntos de muestras similares. Se utilizan en aplicaciones como reconocimiento de patrones, análisis de imágenes, la bioinformática y el análisis de información. Las técnicas de agrupación pueden dividirse en dos categorías: basadas en particiones y jerárquicas . Un ejemplo de clustering en particiones es K-means, que crea K-clusters basado en una partición Voronoi del espacio de características. Del mismo modo, el clustering basado en modos crea una participación asignando cada observación a un modo de estimación de densidad.Por otra parte la agrupación jearquica produce una representación de los objetos basada en un arbol.(Cabezas Luben, 2023)

**K Means**

El algoritmo K-Means funciona dividiendo un conjunto de datos en K grupos, donde cada grupo se caracteriza por su centroide . El algoritmo comienza seleccionando K centroide inicial y luego asigna iterativamente cada punto de datos al centroide más cercano. Luego, los centroides se actualizan en función de los puntos que se les asignan. Este proceso se repite hasta que los centroides dejan de cambiar o se alcanza un cierto número de iteraciones.(Zulfauzi et al., 2023). En el contexto del algoritmo K-Means, un centroide es un punto representativo de un grupo (Ibrahim et al., 2016).

En el problema de agrupamiento, se nos proporciona un conjunto de entrenamiento $x^{(1)}, ..., x^{(m)}$ y queremos agrupar los datos en algunos "gruposçohesivos. Aquí, se nos proporcionan vectores de características para cada punto de datos $x^{(i)} \in \mathbb{R}^n$ como de costumbre; pero no hay etiquetas $y^{(i)}$. Nuestro objetivo es predecir $k$ centroides y una etiqueta $c^{(i)}$ para cada punto de datos.(Piech, 2013) El algoritmo de agrupamiento de k-medias se describe en Algoritmo 1:

**Algorithm 1** Algoritmo de Agrupamiento K-Means

```
Elegir el número de clústeres K
Inicializar los centroides μ_1, μ_2, ..., μ_K aleatoriamente
repeat
    Asignar los puntos al centroide más cercano para formar K clústeres
    for cada clúster k = 1 hasta K do
        Calcular el centroide del clúster k como la media de los puntos en k
    end for
until los centroides no cambien
```

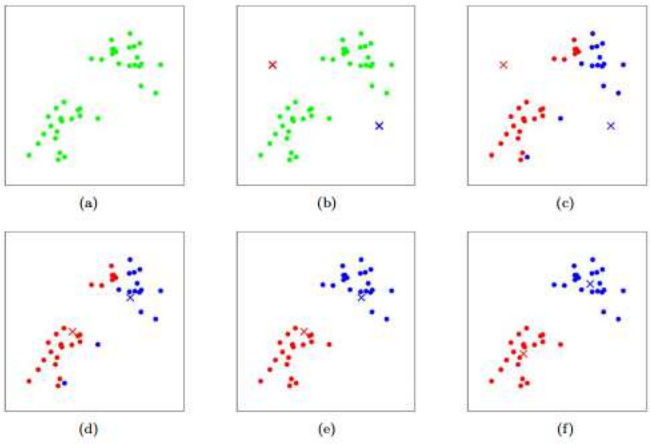
Es importante señalar que la ubicación inicial de los centroides puede tener un impacto significativo en los grupos finales. Para mitigar esto, el algoritmo se puede ejecutar varias veces con diferentes inicializaciones y los resultados se pueden promediar para obtener un conjunto de clústeres más estable y confiable. El algoritmo K-Means generalmente se evalúa

utilizando una métrica de rendimiento llamada suma de errores cuadrados dentro del clúster (WCSS). El WCSS es una medida de qué tan bien están agrupados los puntos de datos dentro de cada grupo.(Zulfauzi et al., 2023)

$$WCSS = \sum_{i} = K \sum_{x} \in C_i (x - \mu_i)^2 \tag{2.11}$$

donde K es el número de grupos, C es el conjunto de puntos de datos en el grupo i , $\mu$ es la media de los puntos de datos en el grupo i y x es un punto de datos.

Figura 2.4: Ejemplos de entrenamiento (a) Conjunto de datos originales. (b) Centroides de conglomerados iniciales aleatorios. (cf) Ilustración de la ejecución de dos iteraciones de k-medias.




Fuente:(Nicole, s.f.)

#### 2.2.6.3. Modelos a usar para clasificación de condición

**Random Forest**

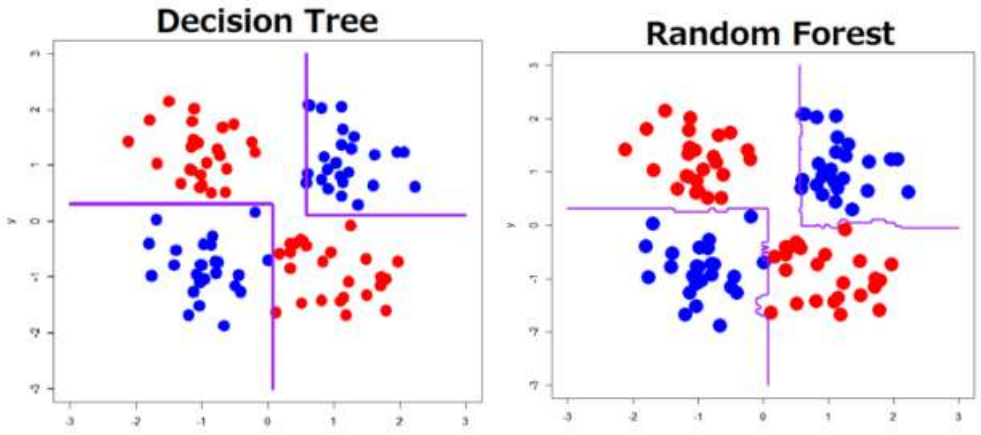
Random Forest es un algoritmo de aprendizaje automático que combina múltiples árboles de decisión para realizar una tarea de clasificación o regresión. Cada árbol de decisión en un Random Forest es construido utilizando un subconjunto aleatorio de las características del conjunto de datos y una muestra aleatoria de los datos de entrenamiento. La predicción final se hace tomando la mayoría de las predicciones de los árboles individuales. Esta técnica de combinación de modelos ayuda a reducir la varianza y el sobreajuste de los modelos individuales. (Breiman, 2001) Para crear nodos de decisión precisos, existen distintos algoritmos de decisión. Sin embargo todos operan de manera similar, primero se examinan las variables yy los criterios de decisión disponibles, y luego se selecciona el que tenga la mejor medida de evaluación. Las métricas de decisión mas usadas, corresponden a(Cutler et al., 2007):

- **Índice Gini** Sean $p$ y $q$ la probabilidad de acierto y falla en su respectivo orden.A mayor índice Gini mayor pureza en los nodos, se calcula $Gini = p^2 + q^2$

- **Ganancia de la información**,Sea $p$ y $q$ la probabilidad de acierto y falla en su respectivo orden y $Entropia = -p\log_2(p) - q\log_2(q)$ una Entropía menor significa que el modelo ha separado de manera más efectiva las clases.

El algoritmo Random Forest es altamente escalable, robusto y puede manejar datos con alta dimensionalidad. También es resistente a valores atípicos y no requiere mucha preparación de datos. Además, es fácil de entender e interpretar debido a la naturaleza del modelo de árbol de decisión subyacente.(Cutler et al., 2007)

Figura 2.5: árbol de desición vs Random Forest




Fuente:(OZAKI, s.f.) with Python.

**Algorithm 2** Algoritmo de Random Forest

Construir un modelo de Random Forest para clasificación o regresión **Entrada:** Datos de entrenamiento, número de árboles $N$, número de características a considerar en cada división $m$ Inicializar el conjunto de árboles $\{T_1, T_2, ..., T_N\}$

**for** $i \leftarrow 1$ $N$ **do** Generar una muestra bootstrap de los datos de entrenamiento Construir el árbol de decisión $T_i$ utilizando la muestra bootstrap:

**while** no se alcanza el criterio de parada **do** Seleccionar al azar $m$ características Realizar la mejor división del nodo usando las $m$ características Dividir el nodo en dos nodos hijos **Salida:** El modelo de Random Forest formado por los árboles $\{T_1, T_2, ..., T_N\}$

### Red Neuronal Artificial

La arquitectura del modelo Red Neuronal Artificial consiste en ordenar el número de capas ocultas, el número de neuronas en cada capa oculta y su conexión, y se utiliza en la Multilayer Layer Perceptron y se entrena mediante el algoritmo de retropropagación (Karaca, 2016).La función de activación utiliza para resolver el problema complejo y no lineal que permite alcanzar su mejor rendimiento al controlando la salida de cada neurona en la capa oculta y en la capa de salida.(Popov et al., 2022) Un algoritmo Un algoritmo de aprendizaje es un procedimiento sistemático paso a paso mediante el cual el peso de conexión entre neurona se

ajustan para minimizar la diferencia entre la salida prevista y la real del modelo. y controlado por el parámetro de aprendizaje, como la tasa de aprendizaje y el impulso . El conocimiento que la red adquiere durante el aprendizaje se codifica implícitamente en sus pesos numéricos y valores de sesgo. Cada capa de la red neuronal actúa como puente entre el parámetro de entrada y el de salida. Un sesgo es un parámetro constante adicional en la red neuronal que se utiliza para ajustar la salida junto con la suma ponderada de las entradas a la neurona. Así, un sesgo mueve la función de activación a izquierda o derecha, arriba o abajo en el gráfico. La salida de la red neuronal MLP viene dada por la Ec.(Atma Ram Sahu, 2020)

**Algorithm 3** Propagación hacia Adelante en una Red Neuronal

Inicializar pesos y sesgos de la red neuronal
Definir la función de activación $f$ (por ejemplo, sigmoide o ReLU)
Inicializar los valores de entrada $\mathbf{x}$

**function** FORWARDPROPAGATION($\mathbf{x}$)
  $\mathbf{a}^{(1)} \leftarrow \mathbf{x}$ ▷ Activación de la capa de entrada
  **for** $l$ desde 2 hasta $L$ **do** ▷ Para cada capa oculta y de salida
    $\mathbf{z}^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ ▷ Combinación lineal
    $\mathbf{a}^{(l)} \leftarrow f(\mathbf{z}^{(l)})$ ▷ Aplicar función de activación
  **end for**
  **return** $\mathbf{a}^{(L)}$ ▷ Salida de la capa de salida
**end function**

$\mathbf{x}_{\text{entrada}} \leftarrow$ datos de entrada
$\mathbf{y}_{\text{verdadera}} \leftarrow$ valores verdaderos (etiquetas)
$\hat{\mathbf{y}} \leftarrow$ FORWARDPROPAGATION($\mathbf{x}_{\text{entrada}}$) ▷ Obtener predicciones

Calcular la función de pérdida $J(\mathbf{y}_{\text{verdadera}}, \hat{\mathbf{y}})$
Aplicar el algoritmo de retropropagación para ajustar los pesos y sesgos =0

**Elementos básicos que componen una red neuronal artificial**

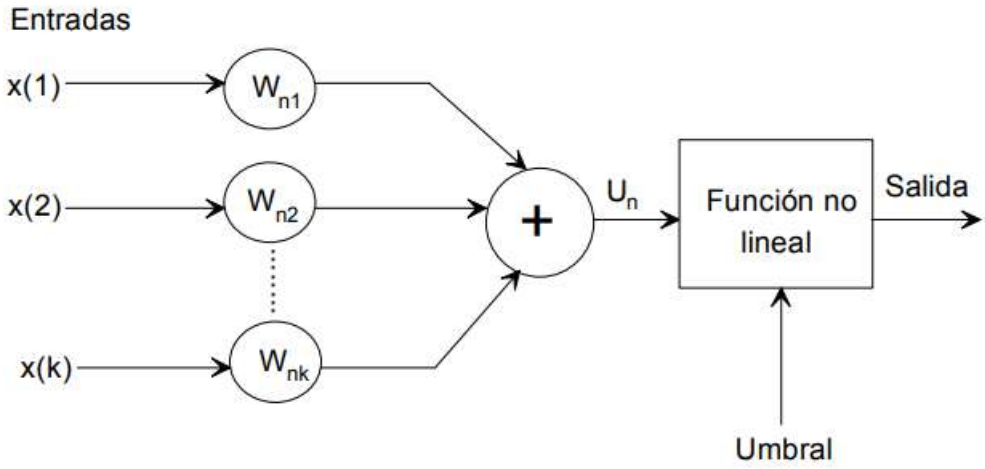
Está constituida por neuronas interconectadas los datos ingresan por medio de la "capa de entrada", pasan a través de la "capa oculta" y salen por la "capa de salida". Cabe mencionar que la capa oculta puede estar constituida por varias capas.

1. Función de entrada: La neurona trata a muchos valores de entrada como si fueran uno solo, las entradas $(in_{i1}, in_{i2}, ...)$se combinan a través de la función de entradaque se calcula a partir de un vector de entrada:

$$input_i = (in_{i1} * w_{i1}) * (in_{i2} * w_{i2}) * ... * (in_{in} * w_{in}) \tag{2.12}$$

   donde: * representa un operador(sumatoria, producto,etc) n es el número de entradas a la neurona $N_i$ y $w_i$ al peso.

Figura 2.6: Estructura de una neuronal artificial




2. Función de activación:La función de activación en una red neuronal es una transformación matemática aplicada a la salida de una neurona o capa de neuronas. Su propósito es introducir no linealidad en el modelo, lo que permite a la red aprender y modelar relaciones complejas entre los datos de entrada y salida. **Sigmoide**: $\sigma(x) = \frac{1}{1+e^{-x}}$, que introduce una curva en forma de S.

3. Función de salida: La función de salida en una red neuronal artificial es la última operación que se realiza sobre las señales procesadas dentro de la red. Su objetivo es convertir la representación interna de la red en una forma que sea adecuada para la tarea específica, como clasificación o regresión. Dado que nuestra clasificación es multiclase se utiliza la función softmax, que generaliza la función sigmoide a múltiples clases:

$$y_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.13}$$

   Aquí, $y_i$ es la probabilidad de que la entrada pertenezca a la clase $i$, y $x_i$ son las activaciones de la última capa para la clase $i$.

Aunque los modelos complejos de aprendizaje automático (p. ej., bosque aleatorio, redes neuronales) suelen superar a los modelos interpretables tradicionales y sencillos (p. ej., regresión lineal, árbol de decisión), en el ámbito industrial, a los ingenieros de confiabilidad les resulta difícil entender y confiar en estos modelos complejos debido a la falta de intuición y explicación de sus predicciones.ElShawi et al., 2018

La elección de la función de salida adecuada es esencial para el rendimiento de la red neuronal, y debe alinearse con los objetivos del modelo y la naturaleza de los datos de salida.

#### 2.2.6.4. Técnicas de explicabilidad de modelos

La explicabilidad de los modelos de machine learning es crucial en muchos sectores, incluido el monitoreo y análisis de muestras de aceite para clasificar si son normales o anómalas. Esta importancia se debe a varias razones clave que impactan tanto el aspecto operacional como el de confianza y responsabilidad en las decisiones automáticas:

- Comprensión y Confianza: Confianza de los Usuarios: En el contexto industrial, donde las decisiones pueden tener consecuencias significativas sobre la producción, la seguridad y los costos, es esencial que los operadores y técnicos confíen en las herramientas que utilizan. Si un sistema de machine learning proporciona explicaciones claras de por qué una muestra de aceite se clasifica como anómala, los usuarios estarán más inclinados a confiar y actuar de acuerdo con estas recomendaciones.
- Comprensión de la Decisión: Las explicaciones ayudan a los usuarios a entender por qué el modelo ha llegado a una conclusión específica. Esto es especialmente valioso en situaciones donde el resultado del modelo contradice las expectativas o la experiencia previa.
- Decisiones Informadas: Cuando el modelo identifica una muestra de aceite como anómala, una explicación detallada de esta clasificación puede guiar a los técnicos sobre qué acciones correctivas tomar, basándose en qué características influyeron más en la decisión.
- Validación y Ajuste de Respuestas: La explicabilidad permite a los expertos validar y ajustar las recomendaciones del modelo, asegurando que las acciones tomadas sean justificadas y adecuadas, minimizando así los errores costosos.
- Mejora Continua del Modelo: Diagnóstico de Errores: Las explicaciones proporcionan insights sobre el funcionamiento interno del modelo, lo que puede ayudar a identificar cuándo y por qué el modelo falla. Esto es crucial para iterar y mejorar el modelo continuamente, ajustando o reentrenando con nuevos datos para manejar mejor las situaciones que el modelo no había previsto inicialmente.
- Identificación de Sesgos: Explicar las decisiones del modelo también puede ayudar a detectar sesgos en los datos o en el aprendizaje del modelo, lo que es esencial para mantener la integridad y la justicia de las aplicaciones automáticas.
- Fomentar la Adopción de Tecnología: Aceptación de la IA: En sectores tradicionalmente conservadores o donde las decisiones tienen altos costos asociados (como en la industria de la energía y la manufactura), la explicabilidad es clave para fomentar la adopción de nuevas tecnologías. Al disipar dudas sobre la çaja negra"de los modelos de IA, las empresas pueden sentirse más seguras al integrar estas tecnologías en sus operaciones diarias.

Las técnicas de explicabilidad revisadas en este trabajo son:

**LIME** (Local Interpretable Model-Agnostic Explanations) es una técnica de interpretabilidad local que se basa en la suposición de que el límite de decisión de un modelo complejo de aprendizaje automático es aproximadamente lineal en las proximidades del punto de datos que se desea explicar. Para ello, LIME genera una muestra perturbada alrededor del punto de interés, obtiene las predicciones del modelo complejo para cada punto de la muestra, asigna pesos a los puntos en función de su proximidad al original, ajusta un modelo interpretable como un modelo lineal o un árbol de decisión utilizando la muestra perturbada, las predicciones y los pesos, y finalmente interpreta el modelo interpretable para obtener una explicación local del comportamiento del modelo complejo en ese punto específico. La técnica LIME7 es valiosa para comprender el funcionamiento de modelos complejos de aprendizaje automático, especialmente en aquellos casos donde la interpretabilidad es crucial, ya que permite identificar las características más importantes que influyen en las predicciones del modelo en puntos específicos de datos.ElShawi et al., 2018

**SHAP** SHAP (SHapley Additive exPlanations) es una técnica avanzada en la ciencia de datos para la interpretación de modelos de aprendizaje automático. Se basa en la teoría de juegos cooperativos, específicamente en los valores de Shapley, para proporcionar explicaciones precisas del efecto de cada característica en la predicción de un modelo.

SHAP es valorado por su capacidad para ofrecer interpretaciones tanto globales como locales de modelos complejos, siendo compatible con cualquier modelo de aprendizaje automático.ElShawi et al., 2018 Explicabilidad se refiere a la propiedad de un modelo de ser explicable en términos humanos. Es decir, qué tan bien puede un modelo explicar, en términos que los humanos puedan entender, las razones detrás de sus decisiones o predicciones. La explicabilidad es crucial en áreas donde es importante justificar, validar o verificar las acciones recomendadas o tomadas por un modelo de IA.

**GridSearchCV** GridSearchCV(Grid Search) algoritmo comúnmente empleado para la puntuación de modelos y el ajuste de hiperparámetros, para ello entrena modelos por cada posible combinaciones de los parámetros y puntuando los modelos para seleccionar el de mejor rendimiento. Es efectivo para bases de datos pequeñas y pocos parámetros. También ofrece control sobre las combinaciones deseadas de hiperparámetros.

Para empezar con la búsqueda se requiere de un estimador(modelo usado), param grid (un diccionario de los hiperparámetros y los valores a probar), scoring (criterio de evaluación del algoritmo) y cv (numero de bloques para la validación cruzada). Posteriormente, se entrena el modelo con el conjunto de entrenamiento y prueba. Finalmente, el resultado es el conjunto de salidas best estimator(mejor estimador), best score (mejor puntaje) y best params(mejores parámetros)

**Majority Voting** Majority Voting (Voto mayoritario), esta técnica de agregación permite que varios modelos colaboren y tomen decisiones en grupo, para generar predicciones acertadas. Imitando el proceso de voto, los modelos votan por cada predicción generada por modelo para decidir por mayoría de votos la mejor predicción.

El proceso consiste en tres fases:

1. Fase de entrenamiento: Recopilación de modelos de aprendizaje automático como SVM(máquinas de vectores de soporte), Random Forest (árboles de decisión) y regresión logística. Después se debe usar el mismo conjunto de datos para entrenar cada modelo.

2. Fase de predicción: Al entrar un nuevo conjunto de prueba, cada modelos recibirá el conjunto como entrada. Los modelos generan sus predicciones independientes. Y cada predicción es tratada como voto para cada etiqueta evaluada.

3. Fase de mayoritario: Finalmente se decide la predicción por la etiqueta con mayor cantidad de votos de los modelos. De haber un empate, se selecciona una estrategia de desempate.

## 2.3. Metodología

La metodología que se describe se basa en los pasos generales para la construcción de los modelos de Machine Learning, estos pasos son comunes a la mayoría de los proyectos de Machine Learning y se han convertido en una metodología estándar en la industria de la ciencia de datos.

1. Revisión bibliográfica: Se lleva a cabo una investigación de fuentes bibliográficas relevantes y un análisis preliminar del proceso minero, de los equipos utilizados en la mina y sobre el Análisis tribológico llevado a cabo en la operación minera.

2. Recopilación de muestras de aceite de motores.

3. Recopilación de fechas de paradas no programadas que tengan como modo de falla el motor diesel.

4. Análisis exploratorio de datos para identificar valores faltantes, valores extremos, distribución de los datos, entre otros aspectos importantes. En el preprocesamiento se limpian estos datos, es decir, se eliminan valores faltantes y extremos además de datos que no sean necesarios para el análisis.

5. El etiquetado de los datos es diferente para cada uno de los modelos predictivos, por lo que se especificará dicho proceso para cada modelo en el capitulo de desarrollo.

6. Se entrenarán los modelos de machine learning utilizando los datos preprocesados.

7. Se llevará a cabo un análisis de los resultados obtenidos por cada modelo y se propondrán recomendaciones para mejorar el sistema de monitoreo de condición con la implementación de los modelos.

Figura 2.7: Esquema de Metodología


## 2.4. Método de desarrollo

### 2.4.1. Tipo de Estudio

Este estudio incluye una etapa inicial de exploración de los parámetros de las muestras antes de aplicar los modelos de machine learning por lo que incluye elementos de un análisis exploratorio, también se analiza el resultado de emplear los algoritmos de machine learning sobre muestras de aceite de un motor de pala CAT6060fs, debido a que es la primera vez que se realiza en un estudio. Pero luego de pasar está etapa el estudio tiene una orientación descriptiva.
Según (Hernández Sampieri et al., 2014) un estudio de tipo descriptivo se define como un estudio que busca especificar propiedades y características importantes de cualquier fenómeno que se analice, además de describir las posibles tendencias de un grupo o muestra. En la presente investigación se busca especificar los parámetros que tienen más relevancia en un análisis tribologico de motores de Palas CAT6060fs, además de describir las tendencias de cada parámetro para poder predecir sus valores en el futuro. Además siguiendo la definición de estudio descriptivo (Hernández Sampieri et al., 2014). Se aplica procesos y modelos de datos ya establecidos y estructurados por los estudios previos de machine learning los cuales contribuyen al análisis de las muestras de aceite,descripción de vida útil de los motores y predicción de posibles fallas en los motores de Palas CAT6060fs. Por lo tanto esta investigación es de tipo **descriptivo**.

# Capítulo 3

# Desarrollo

### 3.0.1. Recopilación de datos

Se extrae las muestras de aceite de los motores, cada muestra tiene 64 parámetros, solo se extraen las muestras de los equipos mencionados anteriormente, además de solo tomar los datos de motores diesel:

Tabla 3.1: Parámetros en muestras de aceite

| **Información general** | | | | | | **64 Parámetros** | | |
|---|---|---|---|---|---|---|---|---|
| Cód. | Equipo | Fecha | Hr.Equipo | Hr.Componente | TBN | ... | Viscosidad 100° | |
| 1290 | SH001 | 9/08/2014 | 663 | 663 | 8.97 | ... | 15.33 | |
| 1290 | SH002 | 9/08/2014 | 663 | 663 | 8.97 | ... | 14.83 | |
| 1906 | SH002 | 24/09/2014 | 805 | 805 | 8.71 | ... | 13.85 | |
| 1248 | SH002 | 5/08/2014 | 635 | 635 | 9.26 | ... | 15.73 | |
| 801 | SH002 | 29/06/2014 | 273 | 635 | 9.61 | ... | 15.18 | |
| 211 | SH001 | 6/05/2014 | 293 | 293 | 9.18 | ... | 14.84 | |
| 245 | SH001 | 9/05/2014 | 3607 | 3607 | 9.24 | ... | 14.82 | |
| 10126 | SH001 | 7/08/2015 | 3607 | 3607 | 9.46 | ... | 14.59 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Los datos aceites se pueden encontrar en el siguiente link:

https://drive.google.com/drive/folders/

1buav9WdZJ4C3S77dSCZCzNrNs61je_Tv?usp=sharing

Se extraen las fechas de falla no planificados de tipo sintomático perteneciente a los motores: Se sabe que el cambio de aceite se cambia en cada PM del equipo que consta de 250h,se consideran las muestras de aceite cuyas las horas en el motor no cumplieron con su

Tabla 3.2: Fechas de Paradas no programadas

| Turno | Equipo | Comentarios | Tipo | |
|---|---|---|---|---|
| 07-Feb-2018D | SH001 | 0100 Cambio de AC | Unplanned Loss | Equipment |
| 05-Feb-2018D | SH001 | 0100 Cambio de AC | Unplanned Loss | Equipment |
| 20-Jun-2018N | SH002 | 0100 0100 Evaluación de motor baja presion de ac motor - cambio de motor | Unplanned Loss | Equipment |
| 04-Sep-2018N | SH002 | 0100 Cambio de aceite de motor por condición | Unplanned Loss | Equipment |
| ... | ... | ... | ... | |

ciclo de vida completo.

### 3.0.2. Modelo 1: Detección de anomalías

Se propone detectar anomalías con algoritmos especializados en la detección de anomalías aplicando un enfoque no supervisado(sin etiquetas previas) y sobre los parámetros de desgaste y contaminación que están relacionados con a condición del componente adicionando la recomendación de la Norma ASTM que recomienda considerar el 6 % de datos más alejados a la media como anómalos.

#### 3.0.2.1. Selección de características

Comparamos las características seleccionadas en la bibliografía revisada.

1. Modeling and classifying the in-operando effects of wear and metal contaminations of lubricating oil on diesel engine: A machine learning approach.Rahimi et al., 2022
2. Análisis predictivo de activos mineros para obtención de intervalo de falla mediante algoritmos de machine learningReveco Díaz, 2019
3. Automating predictive maintenance using oil analysis and machine learningKeartland y van Zyl, 2023.
4. A review on lubricant condition monitoring information analysis for maintenance decision support.Wakiru et al., 2019
5. Predicting motor oil condition using artificial neural networks and principal component analysis.Rodrigues et al., 2020

6. Oil condition monitoring, an AI application study using the Classification Learner Technics.Grebenişan1 et al., 2019

Tabla 3.3: Selección de características según la bibliografía revisada

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Hierro (Fe) (ppm) | X | X | | | | X |
| Vis40 | | X | | | | |
| Vis100 | X | X | X | | | |
| TBN | | X | | | | |
| Número de Ácido Total (TAN) | | | X | | | |
| Boro (B) (ppm) | X | | X | | | |
| Bario (Ba) (ppm) | | | | X | | |
| Calcio (Ca) (ppm) | X | | X | | | |
| Magnesio (Mg) (ppm) | X | | X | | | |
| Molibdeno (Mo) (ppm) | | | X | | | |
| Fósforo (P) (ppm) | X | | X | | | |
| Sodio (Na) (ppm) | X | X | X | X | | |
| Silicio (Si) (ppm) | | X | X | X | | |
| Zinc (Zn) (ppm) | X | | X | | | |
| Agua | | | | | X | |
| Glycol | | | | | X | |
| Vanadio (V) (ppm) | | X | X | | | |
| Potasio (K) (ppm) | | | | | X | |
| Silicio (ppm) | X | | | | | |
| Sodio (ppm) | | | | | X | |
| Boro (ppm) | X | | X | | | |
| Cromo (Cr) (ppm) | X | X | | | | |
| Aluminio (Al) (ppm) | X | X | | | | |
| Níquel (Ni) (ppm) | | X | | | | |
| Estaño (Sn) (ppm) | X | X | | | | |
| Plomo (Pb) (ppm) | X | X | | | X | |
| Cobre (Cu) (ppm) | X | X | | | X | |
| Partículas (PQ) (ppm) | X | | | | | |

Cada muestra de aceite cuenta con 64 parámetros diferentes pero no todos son relevantes al hallar anomalías de la muestra que indiquen problemas potenciales, por ello basados en la recopilación de la literatura y la experiencia de los ingenieros de confiabilidad se elige las siguientes características a evaluar para el modelo 1 por ser indicadores de anomalías y determinar una falla a largo y corto plazo:

- Sodio (Na), Silicio (Si), Vanadio (V), Titanio (Ti), Plata (Ag), Níquel (Ni), Aluminio (Al), Plomo (Pb), Estaño (Sn), Cobre (Cu), Cromo (Cr), Hierro (Fe).

- Viscosidad a 100C (en unidades de centiStokes cST).
- Hollín (en porcentaje)
- Sulfatación (en unidades de absorbancia abs/0.1 mm).
- Nitración (en unidades de absorbancia abs/0.1 mm)
- Oxidación (en unidades de absorbancia abs/0.1 mm)
- Contenido de agua (en porcentaje)
- Número total de Base TBN
- Partículas Ferrosas
- Contaminantes especiales: Glycol y Diesel
- Turbidez_NTU_

#### 3.0.2.2. Preprocesamiento

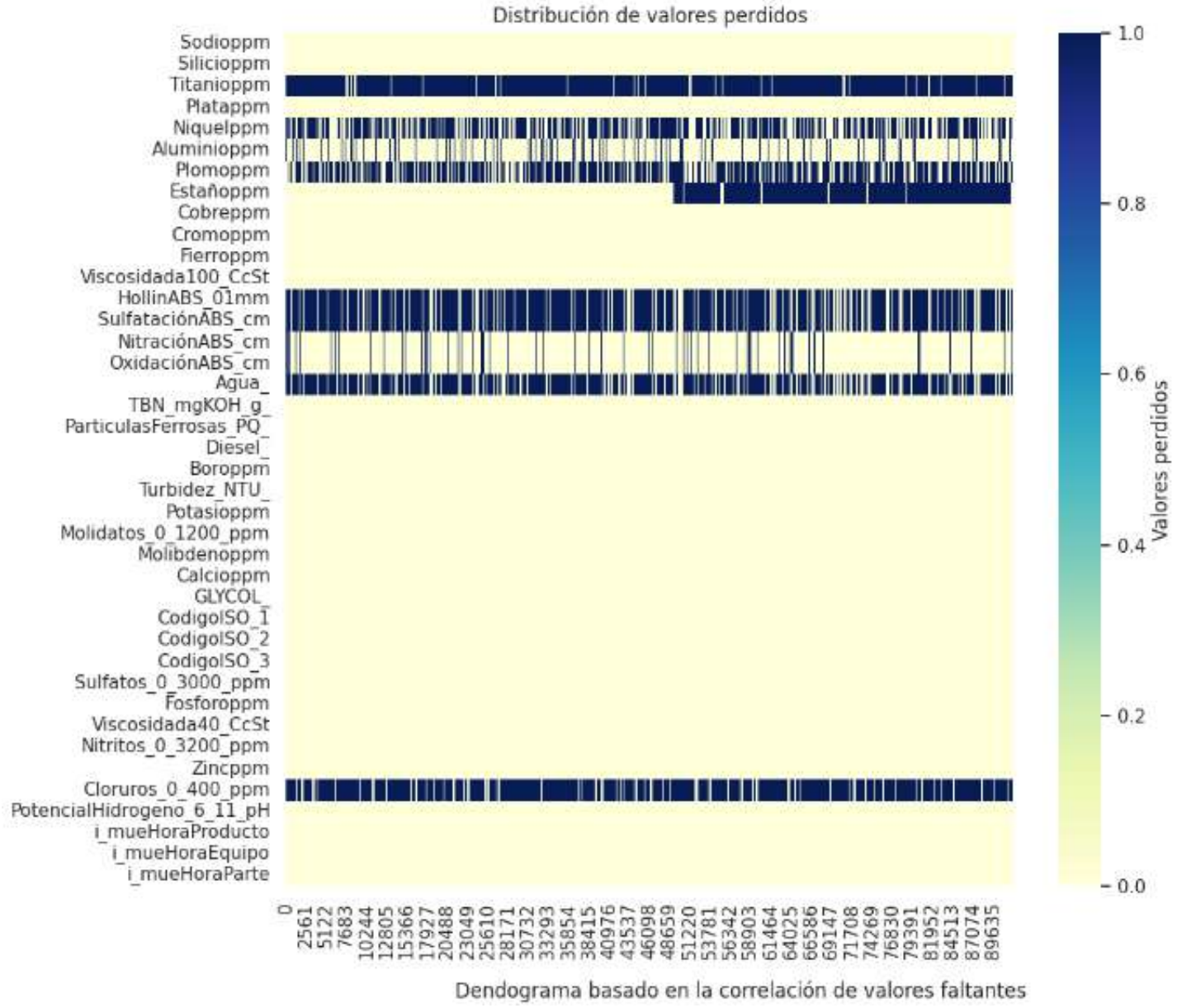
El preprocesamiento inicia con la limpieza de datos utilizando la libreria Numpy de Python y se convierte en NaN a los valores no numéricos.

Antes de realizar cualquier análisis estadístico, se debe manejar los valores faltantes y determinar de que tipo de valores perdidos tenemos:

- MCAR(falta completamente al azar): estos valores no dependen de ninguna otra característica.
- MAR (ausente al azar): estos valores pueden depender de algunas otras características.
- MNAR (Missing not at random): estos valores faltantes tienen alguna razón por la que faltan.

La Figura 3.1 muestra valores NaN en los parámetros Hollín, Sulfatos, Cloruros, Nitración, Oxidación, Agua, Titanio, Plomo, Estaño.

Figura 3.1: Distribución de valores perdidos

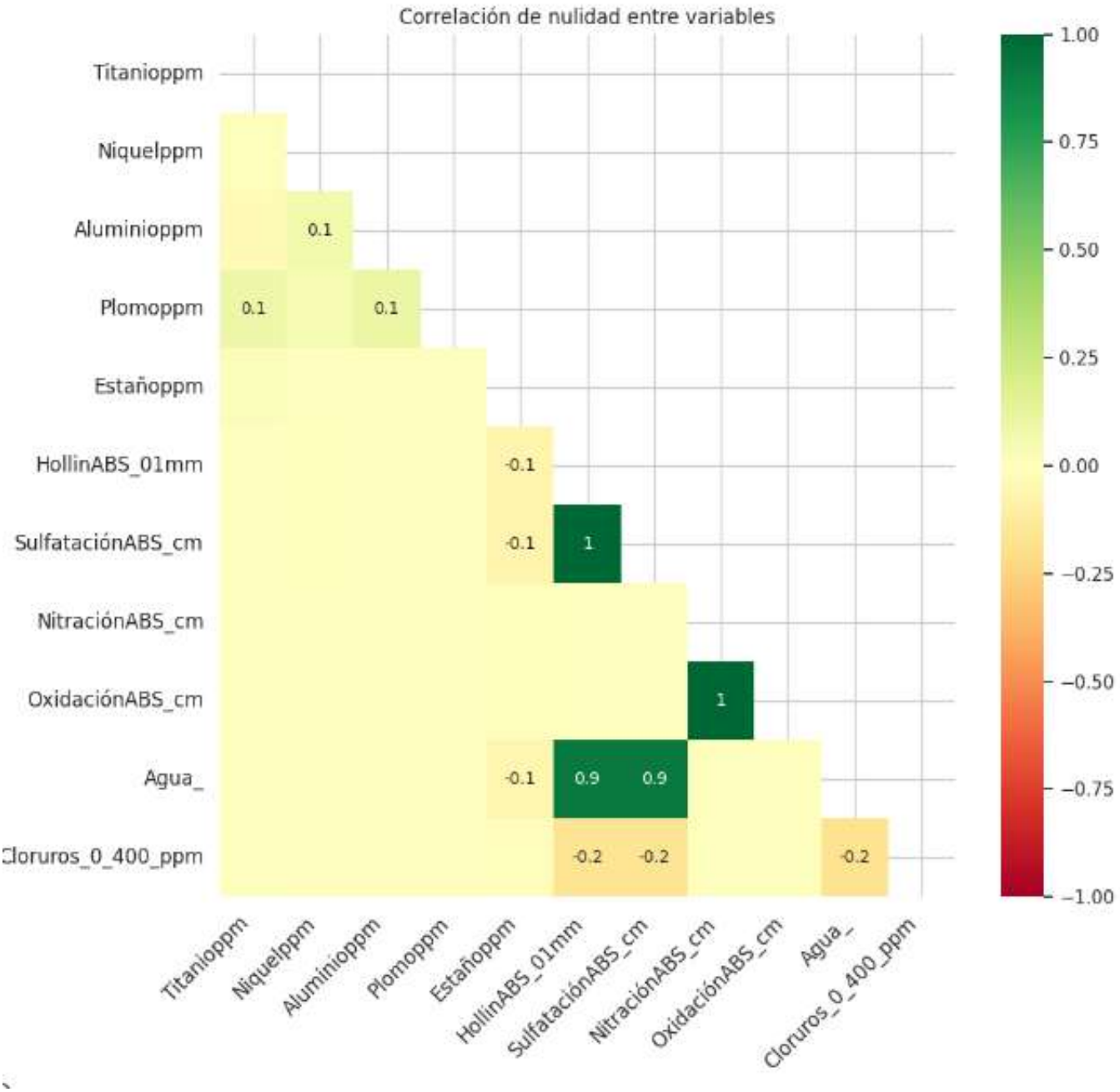



La matriz de correlación de nulidad en la Figura 3.2 mide la presencia simultánea de valores nulos (faltantes) en pares de variables.

Se presentan fuertes correlaciones de nulidad entre las variables 'SulfataciónABS_cm', 'Nitración', 'OxidaciónABS_cm' tienen una alta correlación de nulidad (0.9). Esto sugiere que cuando una de estas variables tiene un valor nulo, la otra también tiende a tener un valor nulo. Las fuertes correlaciones entre SulfataciónABS_cm, NitraciónABS_cm y OxidaciónABS_cm sugieren que los métodos de imputación de valores nulos deben considerar estas relaciones. Cloruros_0_400_ppm tiene correlaciones negativas débiles con SulfataciónABS_cm,

NitraciónABS_cm, OxidaciónABS_cm y Agua_, lo que sugiere una ligera tendencia de que cuando Cloruros_0_400_ppm tiene valores nulos, estas variables tienden a no tener valores nulos, aunque esta tendencia es muy débil. Las variables Níquelppm, Aluminioppm, Plomoppm, y Estañoppm tienen correlaciones muy bajas con otras variables, indicando que los valores nulos en estas variables no están fuertemente relacionados con los valores nulos en otras variables. En cuanto a las variables Titanio y Cloruros_0_400_ppm debido a su alto nivel de valores faltantes y u poca relevancia en el analisis de aceite se descartan para el analisis.

Figura 3.2: Correlación de nulidad entre variables



### Manejo de datos faltantes

Primero se experimento el método de **imputación por mediana** por que se utiliza en conjuntos de datos con características numéricas. Se selecciona la mediana porque es menos sensible a los valores atípicos en los datos que la media y es especialmente útil cuando la distribución de los datos no es simétrica. Este método asegura que los valores imputados son realistas y no están influenciados por valores extremos.Keartland y Zyl, 2020

Sin embargo, luego de revisar la correlación entre datos y se evalúa aquellos con mayor correlación por que el metodo KNN Imputer puede resolver mejor la imputación, de esta manera se tiene:

Figura 3.3: Matriz de correlación con las características seleccionadas para la detección de anomalías

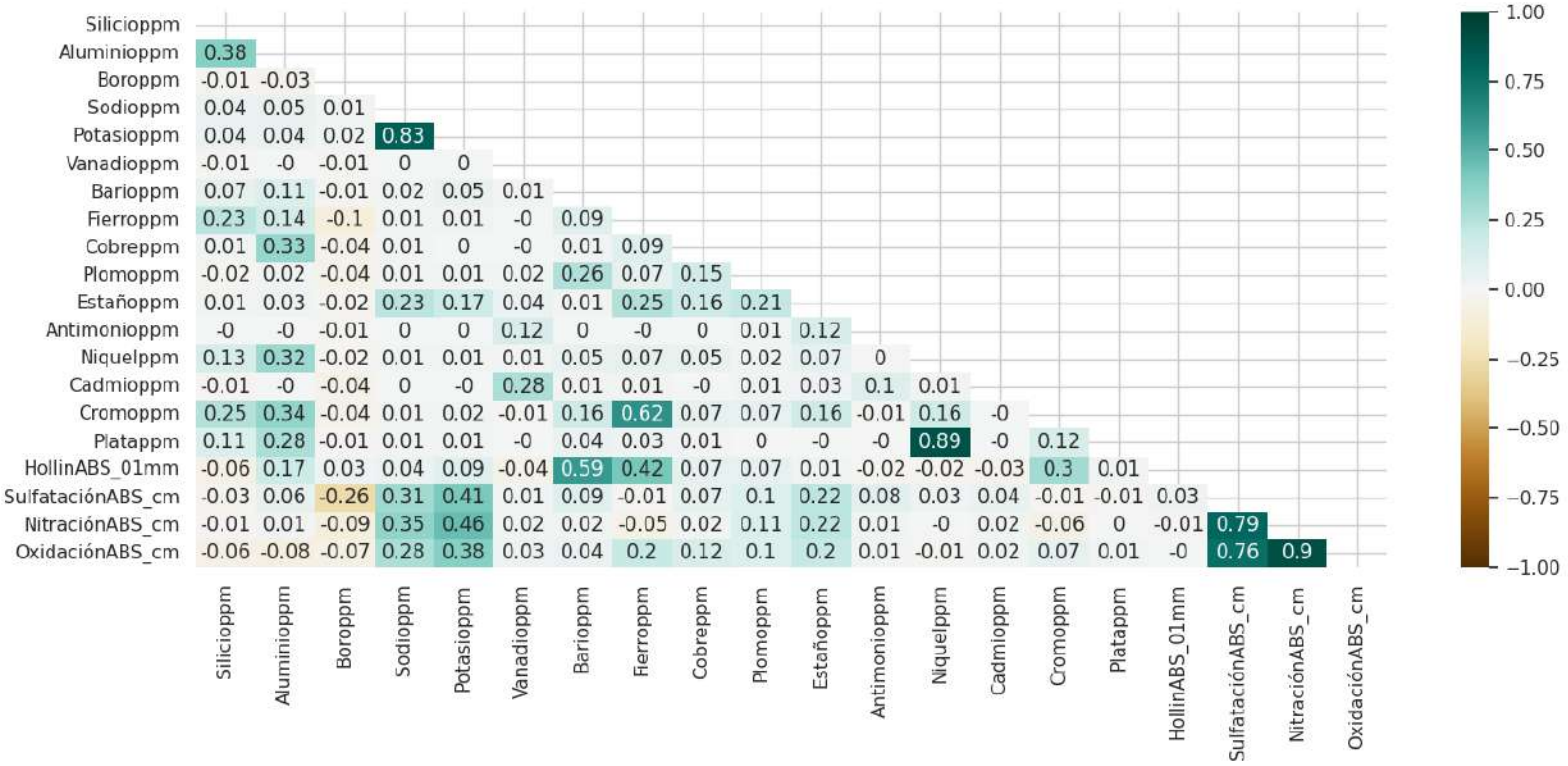
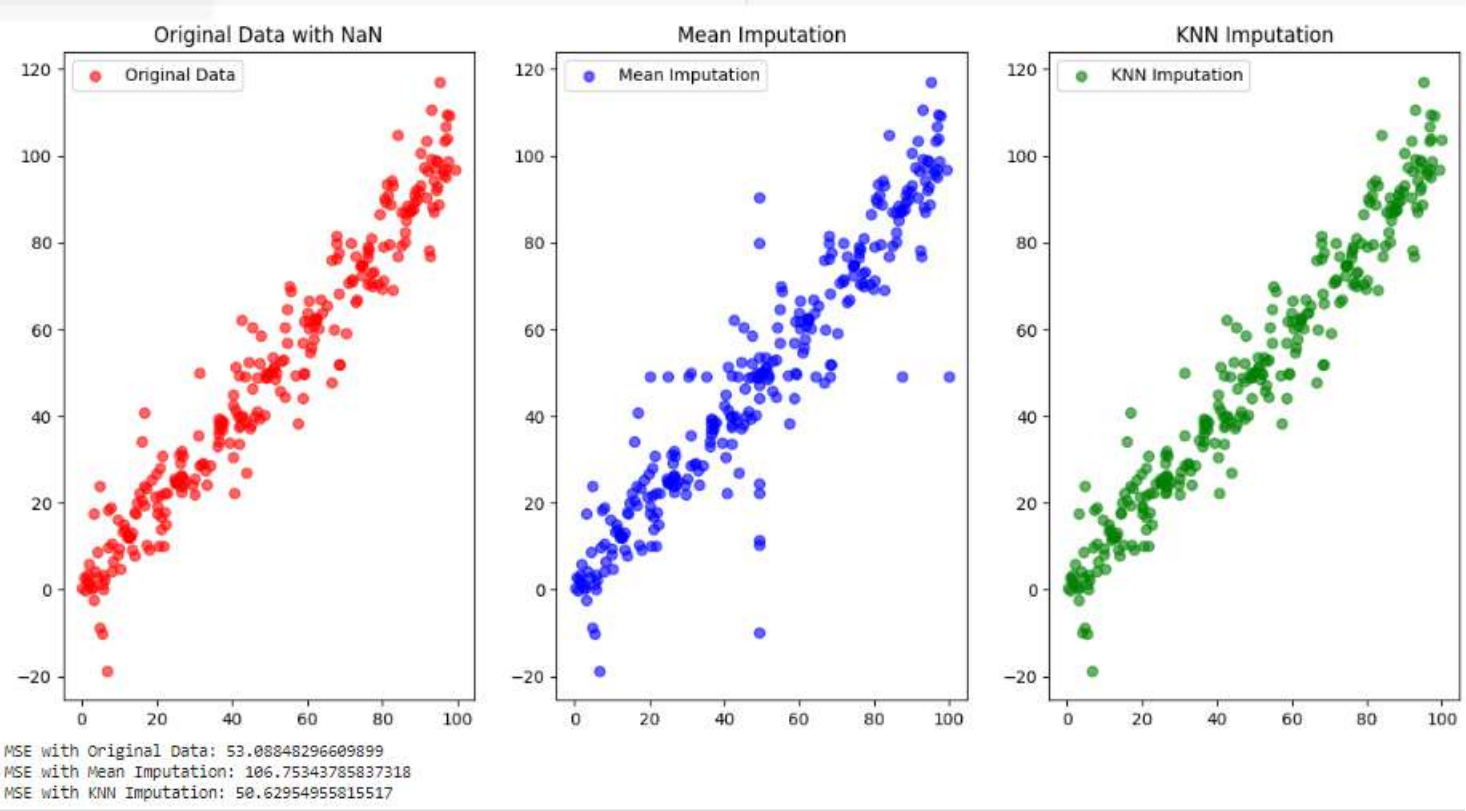


Figura 3.4: Imputación para Nitracion_cm y OxidaciónABS_cm




| Tipo de Datos | Datos Originales | Imputación por Media | Imputación KNN |
|---|---|---|---|
| Error Cuadrático Medio | 53.08848296609899 | 106.75343785837318 | 50.62954955815517 |

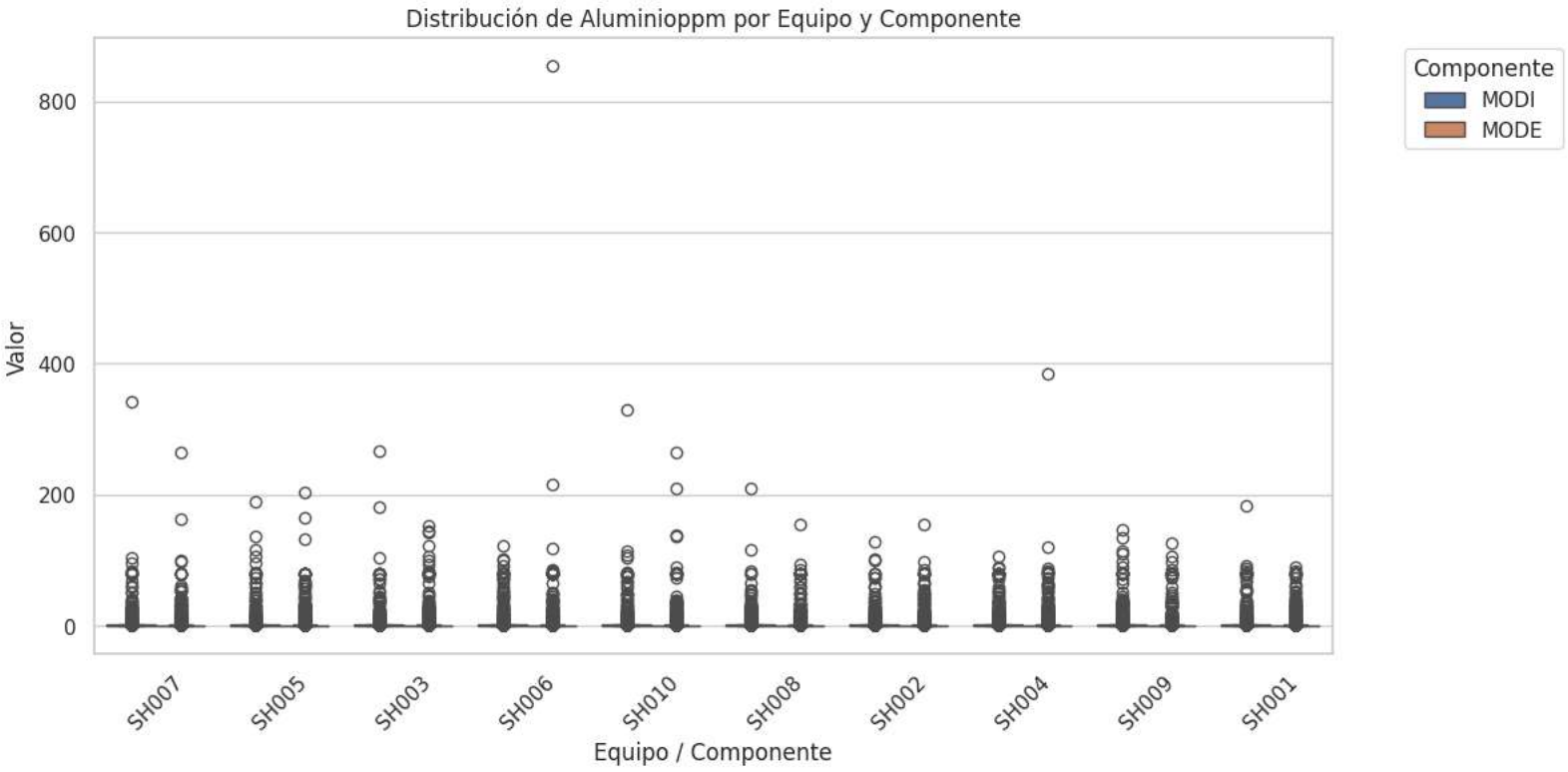
Tabla 3.4: Comparación del Error Cuadrático Medio para diferentes métodos de imputación para Sulfatación y Oxidación

Las diferencias entre los MSE son muy pequeñas, lo que sugiere que, para este conjunto de datos y modelo específicos, la elección del método de imputación no tendrá un impacto sustancial en el rendimiento de los modelo. Por ser menos costoso se elige la imputación por media.

**Manejo de datos atípicos**

Los valores atípicos (*outliers*) son valores que difieren significativamente del resto de los valores en un conjunto de datos y pueden tener un impacto en el modelo. Para identificar de forma inicial y rápida valores atípicos utilizamos el diagrama de cajas, como se puede ver en la figura 16. Algunos valores atípicos están por mucho fuera de un rango normal. La figura diagrama de caja (boxplot) Figura 3.5 muestra la distribución de las mediciones de aluminio en partes por millón (ppm) para diferentes equipos y componentes La mayoría de los equipos muestran una concentración de aluminio relativamente baja, con la mayoría de los

Figura 3.5: Outliers por componente y equipo para Aluminio




valores concentrados cerca de la parte inferior del rango. Hay valores atípicos significativos, especialmente notables en los equipos SH001, SH005, y SH009.

Para determinar si se eliminan los outliers se utiliza dos métodos:

Primero se aplico el método estadístico de **Rango intercuartílico (IQR) para la detección de valores atípicos**:

$$\text{Outliers} = \begin{cases} x < Q_1 - k \times \text{IQR}, \\ x > Q_3 + k \times \text{IQR}, \end{cases}$$

donde:

- $Q_1$ es el primer cuartil (el percentil 25).
- $Q_3$ es el tercer cuartil (el percentil 75).
- IQR es el rango intercuartílico, calculado como $Q_3 - Q_1$.
- $k$ es el factor de umbral (típicamente igual a 1.5).

- $x$ es un valor individual en el conjunto de datos.

Segundo se aplica **K-nearest neighbors(KNN)** con 0.1 de contaminación.Para evaluar el impacto de remover outliers del conjunto de datos se evaluara el estadístico de la prueba de Kolmogorov-Smirnov, el valor p (p-value) que indica la probabilidad de observar el efecto medido cuando la hipótesis nula es cierta. Un valor bajo (típicamente menor que 0.05) indica que hay evidencia estadística para rechazar la hipótesis nula de que las dos muestras provienen de la misma distribución continua :

Tabla 3.5: Resultados del test de Kolmogorov-Smirnov para K-Means(0.01) e IQR

| | KNN(0.01) | | IQR | |
|---|---|---|---|---|
| Elemento | Statistic | p-value | Statistic | p-value |
| Sodioppm | 0.0164 | 0.0000 | 0.2143 | 0.0000 |
| Silicioppm | 0.0153 | 0.0000 | 0.1273 | 0.0000 |
| Platappm | 0.0215 | 0.0000 | 0.2208 | 0.0000 |
| Niquelppm | 0.0144 | 0.0000 | 0.1915 | 0.0000 |
| Aluminioppm | 0.0142 | 0.0000 | 0.0896 | 0.0000 |
| Plomoppm | 0.0210 | 0.0000 | 0.1640 | 0.0000 |
| Estañoppm | 0.0248 | 0.0000 | 0.0481 | 0.0000 |
| Cobreppm | 0.0238 | 0.0000 | 0.1617 | 0.0000 |
| Cromoppm | 0.0156 | 0.0000 | 0.1178 | 0.0000 |
| Fierroppm | 0.0261 | 0.0000 | 0.1680 | 0.0000 |
| Viscosidada100_CcSt | 0.0299 | 0.0000 | 0.1392 | 0.0000 |
| HollinABS_01mm | 0.0039 | 0.3555 | 0.0933 | 0.0000 |
| SulfataciónABS_cm | 0.0012 | 1.0000 | 0.0682 | 0.0000 |
| NitraciónABS_cm | 0.0042 | 0.2566 | 0.0944 | 0.0000 |
| OxidaciónABS_cm | 0.0036 | 0.4281 | 0.1725 | 0.0000 |
| Agua_ | 0.0006 | 1.0000 | 0.0976 | 0.0000 |
| TBN_mgKOH_g_ | 0.0037 | 0.4036 | 0.0980 | 0.0000 |
| ParticulasFerrosas_PQ_ | 0.0269 | 0.0000 | 0.1283 | 0.0000 |
| Diesel_ | 0.0010 | 1.0000 | 0.0265 | 0.0014 |
| Boroppm | 0.0219 | 0.0000 | 0.1884 | 0.0000 |
| Turbidez_NTU_ | 0.0209 | 0.0000 | 0.1003 | 0.0000 |
| Potasioppm | 0.0224 | 0.0000 | 0.1481 | 0.0000 |
| Vanadioppm | 0.0207 | 0.0000 | 0.1366 | 0.0000 |
| Cadmioppm | 0.0200 | 0.0000 | 0.1432 | 0.0000 |
| Titanioppm | 0.0193 | 0.0428 | 0.0193 | 0.0428 |
| Barioppm | 0.0270 | 0.0000 | 0.1897 | 0.0000 |
| Antimonioppm | 0.0220 | 0.0000 | 0.0189 | 0.0506 |
| Estañoppm.1 | | | 0.4234 | 0.0000 |

Los valores de p-value en la tabla 3.5, muestran que KNN con una heurística de 0.01

tiene un impacto en la distribución de Sodio,Silicio,Aluminio,Cobre,Cromo,Fierro,Viscosidad100°, Hollín, Sulfatación,Nitración,Oxidación,TBN,etc. Por lo tanto no se eliminaran outliers. Primero por que afecta la distribución real de los datos y por que en este contexto esos valores atípicos pueden tener un significado importante estos valores atípicos pueden ser las anomalías que se buscan en este primer modelo y se espera contar con valores similares en el futuro.

**Normalización**

Los datos se normalizan para mejorar el rendimiento y la precisión de los modelos, consiste en ajustar los datos a una escala común para que todos los atributos contribuyan de manera equitativa al análisis y algunos de nuestros datos están en escalas decimales, y otros en decenas. La normalización más comúnmente usada es Min-Max que transforma los valores originales a un rango entre 0 y 1 conservando la distribución original de los datos.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

**Etiquetado de condición mediante ASTM D7720 Standard Guide for Statistically Evaluating Measurand Alarm Limits when Using Oil Analysis to Monitor Equipment and Oil for Fitness and Contamination**

La norma ASTM D7720 se utiliza para evaluar estadísticamente y ajustar límites para el monitoreo basado en la condición de medidas representativas de análisis de aceites de equipos en servicio. Para evaluar dichos límites, se utilizan técnicas de evaluación de alarmas establecidas en la norma, como el Control de Procesos Estadísticos (SPC, por sus siglas en inglés), que se basa en datos de medidas que puedan ajustarse a una distribución normal.

La norma proporciona aportes significativos para la presente investigación:

1. Estandarización de Datos: Utilizar una norma reconocida como la ASTM D7720 para el etiquetado de datos garantiza la consistencia y la estandarización en la manera en que se clasifican y se describen las muestras de aceite.

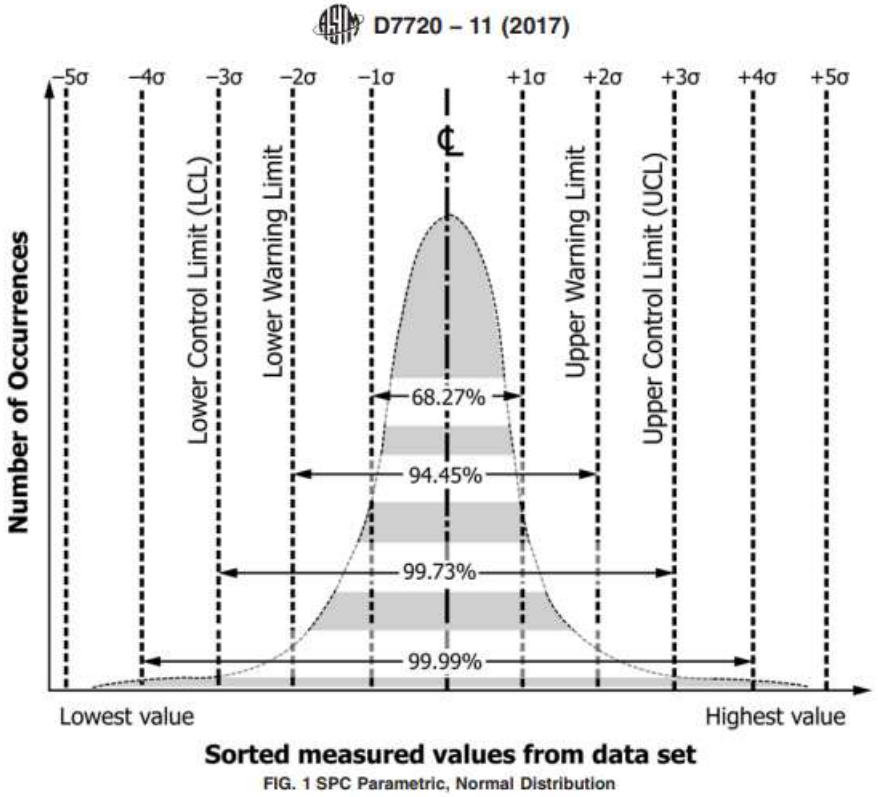
2. Comparación de Modelos: Con un conjunto de datos estandarizado y etiquetado de la forma en la que se hace actualmente en la operación, establece una línea base para comparar diferentes modelos de aprendizaje automático y evaluar objetivamente qué modelo tiene un mejor rendimiento en la tarea específica de clasificar o predecir resultados basados en las muestras de aceite.

3. Validación del Proceso de Mantenimiento: Los ingenieros de mantenimiento ya están utilizando etiquetas basadas en la norma ASTM D7720, el modelo propuesto basado en clustering y modelos de detección de anomalías puede servir para mejorar los procesos existentes.

### Metodología actual de detección de anomalías: Limites de Normalidad

La metodología de clasificación actual de la operación minera implica límites de normalidad , en el caso de análisis de aceites, se establecen límites condenatorios para cada una de las características que componen el análisis de aceites. Estos límites se determinan siguiendo los parámetros establecidos por la norma ASTM D7720 (D7720, 2011).

Figura 3.6: Determinación de límites según ASTM D7720




Conshohocken, 2011

Considerando está información se utiliza la distribución normal para modelar la relación entre las múltiples variables. Se establecen 5 percentiles $q1$, $q2$,$q3$,$q4$,$q5$ y *score* es la distancia de Mahalanobis del punto hasta el centro del conjunto de datos, considerando las condiciones de la siguiente manera:

- Si $(score < q1) or (score > q5)$ = "Anormal"
- Si $((scores >= q1) and (scores < q2)) or ((scores >= q4) and (scores < q5))$ = "Anormal"
- Si $((scores >= q2) and (scores < q3)) or ((scores >= q3) and (scores < q4))$ = "Normal"

En resumen se considera la muestra como anómala si cualquier valor del análisis de aceite excede los límites de condena.

Tabla 3.6: Número de muestras anómalas con método de percentiles

| Método | Normal | Anómalo |
|---|---|---|
| Etiquetado Actual(Percentiles) | 73880 | 17644 |
| Rango Intercuartilico | 84191 | 7333 |

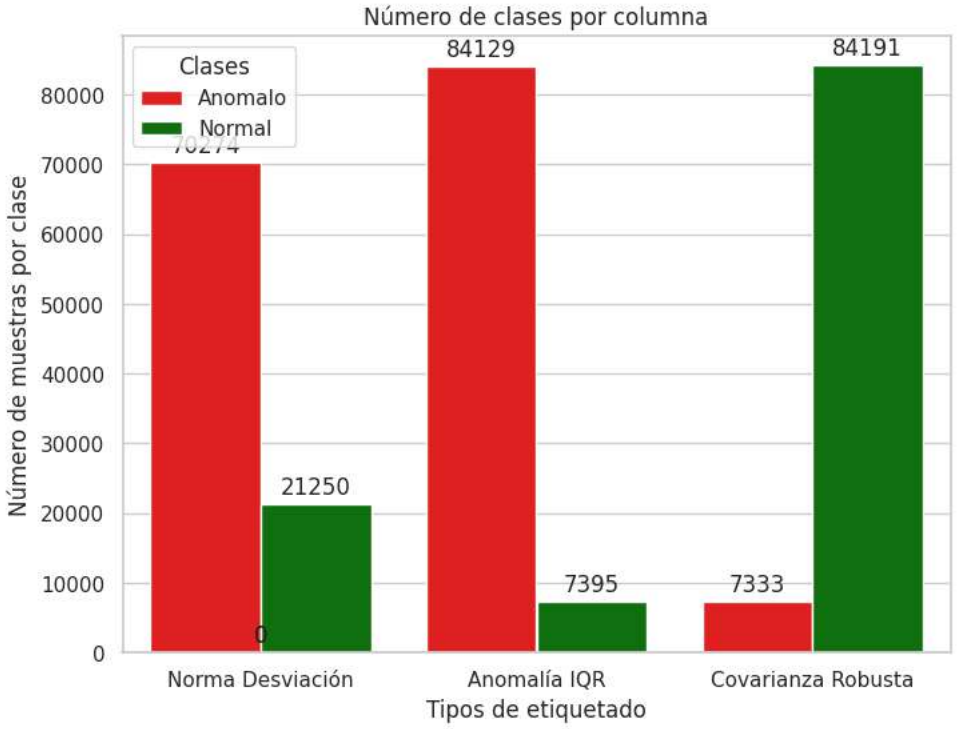
### Anomalías mediante Covarianza Robusta

El algoritmo de Covarianza Robusta se ha elegido debido a su capacidad para definir un intervalo preciso dentro del cual los datos se consideran anómalos.Siguiendo la norma ASTM D7720, se etiquetarán como anómalos aquellos datos que representen el 6 % del total y que se encuentren a más de dos desviaciones estándar de la media. Para ello, se empleará el algoritmo de Covarianza Robusta, el cual permite establecer un rango específico dentro del cual se clasifican las muestras como anómalas.

Tabla 3.7: Número de muestras etiquetadas

| Método | Normal | Anómalo |
|---|---|---|
| Covarianza Robusta | 85719 | 6470 |

Figura 3.7: Número de etiquetas por clase basado en Norma ASTM D7720



#### 3.0.2.3. Selección de modelos

Según la bibliografía revisada en el campo de mantenimiento los algoritmos más usados y que obtienen mejores resultados son los algoritmos de Aprendizaje Supervisado debido a su capacidad para modelar relaciones no lineales y su interpretabilidad, estos métodos se apoyaran en las normas sugeridas por la teoria. Los algoritmos de Aprendizaje supervisado son importantes para que los usuarios del sistema comprendan de mejor manera por que el algoritmo elige cierto resultado,además demuestran una alta capacidad de tolerancia a conjuntos de datos de alta dimensionalidad y ruido. Para la clasificación de las muestras se probaran los algoritmos de:

- Random Forest

Por otra parte dado que nuestras muestras no tienen etiquetado exploraremos la aplicación de algoritmos no supervisados basados en clustering:

- K-means
- Isolation Forest
- SVM-One Class
- PCA-Anomaly Detection

#### 3.0.2.4. Evaluación de modelos

Los modelos supervisados se evaluara mediante:

- Validación cruzada: Dado que nuestros datos de entrenamiento son limitados, el conjunto total de muestras se dividirá en subgrupos de entrenamiento y prueba, 80 % y 20 % respectivamente, y se realizaran iteraciones en las que se entrenara al modelo con diferentes combinaciones y se evaluará el rendimiento promedio.
- Matriz de Confusión: La matriz de confusión es una herramienta para evaluar el rendimiento de un modelo de clasificación. Permite visualizar y calcular el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.
- Curva ROC: La curva ROC es una representación gráfica del rendimiento de un modelo de clasificación. Muestra la tasa de verdaderos positivos frente a la tasa de falsos positivos a medida que se varía el umbral de clasificación.

A diferencia de los modelos supervisados, la validación de clustering puede ser más desafiante debido a la falta de etiquetas verdaderas. La validación del modelo permitirá evaluar el rendimiento y su capacidad predictiva, por ello se utilizara:

1. Métricas Internas:
    - Índice de Silueta mide qué tan similar es una muestra a su propio cluster en comparación con otros clusters. Los valores cercanos a +1 indican una buena agrupación, mientras que los valores cercanos a -1 indican agrupaciones incorrectas.
2. Validación Visual
    - Reducción de Dimensionalidad para Visualización: Utiliza técnicas como PCA para visualizar los clusters en dos o tres dimensiones.
3. Estabilidad de Clustering
    - Consistencia en Múltiples Ejecuciones: Verifica si el clustering es consistente en múltiples ejecuciones con diferentes inicializaciones.
    - Validación Cruzada de Clustering: Similar a la validación cruzada en aprendizaje supervisado, pero se aplica a clustering, como dividir los datos y comparar si los clusters son consistentes a través de las divisiones.

Además de la precisión, la sensibilidad y la especificidad que se derivan de la matriz de confusión, se medirá la exactitud (accuracy) que muestra la proporción de muestras correctamente clasificadas, el valor F1 que se calcula como la media armónica de precisión y exhaustividad, y proporciona una medida equilibrada del rendimiento del modelo en términos de falsos positivos y falsos negativos,y el área bajo la curva ROC (AUC) medida común para evaluar el rendimiento general del modelo.

#### 3.0.2.5. Explicación de modelos

Se explorar el uso de SHAP (SHapley Additive exPlanations) para la interpretación de modelos de aprendizaje automático. Se basa en la teoría de juegos cooperativos, específicamente en los valores de Shapley, para proporcionar explicaciones precisas del efecto de cada característica en la predicción de un modelo. SHAP es valorado por su capacidad para ofrecer interpretaciones tanto globales como locales de modelos complejos, siendo compatible con cualquier modelo de aprendizaje automático.

### 3.0.3. Modelo 2: Clasificación de condición de la muestra

El modelo 1 revisa los elementos de desgaste cuya degradación pueden indicar directamente problemas mecánicos o de desgaste, en cambio los elementos aditivos están diseñados para agotarse a un ritmo sincronizado con la vida útil esperada del aceite, por lo que su agotamiento normal no es típicamente una 'anomalía'. El segundo modelo analizara todos los parámetros de aceite seleccionados para determinar un cambio de aceite basado en la condición del mismo o una potencial falla por condición. Este enfoque se denomina Análisis

de Aceite Basado en la Condición o Condition-Based Oil Change (CBC) y utiliza datos de múltiples parámetros para hacer una evaluación holística de la salud del aceite y, por extensión, del motor o la máquina lubricada. Para lograr esta clasificación se entrenara un modelo con un enfoque orientado a eventos. El **Preprocesamiento** y la **Evaluación de modelos** sigue la misma estructura del primer modelo.

#### 3.0.3.1. Etiquetado de condición

Los ingenieros de confiabilidad de la operación revisan las muestras anómalas y asignan una condición a la muestra:

- Critico: Falla inminente, requiere parada inmediata para cambio de aceite o diálisis.
- Seguimiento: Requiere programar parada. Se debe indicar la fecha de parada.
- Normal: Muestra normal.

Además de las muestras etiquetadas por los expertos, se agregaran las etiquetas para muestras asociadas con fechas de paradas no programadas(falla o cambio de aceite). Se propone etiquetar los datos según los siguientes criterios:

1. Sabiendo la fecha de falla del motor y teniendo en cuenta una evolución temporal como la que aparece en Figura , se considerara las muestras con condición 'Seguimiento' a los datos con fecha de una semana antes de la falla, y las tres últimas fechas antes de falla se etiqueta como 'Critico'.
2. Se toma en cuenta las horas de motor, horas de producto(aceite) y horas de equipo.

Existe un desequilibrio en el número de muestras entre las clases, la clase 'Normal' tiene significativamente más muestras que las clases 'Precaución' y 'Alerta' pues se tienen 206 fechas de falla,

#### 3.0.3.2. Selección de características

Para el modelo 2 además de los parámetros anteriores se evaluaran los elementos aditivos, elementos que mantienen el aceite en buen estado su variabilidad puede ser indicativo para un cambio de aceite:

- Boroppm , Potasioppm, Molidatos_0_1200_ppm, Molibdenoppm, Calcioppm, CodigoISO_1, 'CodigoISO_2, CodigoISO_3, Sulfatos_0_3000_ppm,Fosforoppm, Viscosidada40_CcSt, Nitritos_0_3200_ppm, Zincppm, Cloruros_0_400_ppm, PotencialHidrogeno_6_11_pH

#### 3.0.3.3. Selección de modelos

En muchas aplicaciones prácticas, como la detección de fraude o el diagnóstico de enfermedades, las clases minoritarias ( 'Critico' y 'Seguimiento') son a menudo las más críticas para identificar. Aunque las clases minoritarias son menos frecuentes, su correcta identificación es crucial,en este caso random forest y redes neuronales proporcionan la flexibilidad y la capacidad para destacar estas clases a pesar de su menor representación en los datos. La investigación de métodos avanzados para tratar con clases desbalanceadas es un área activa en *MachineLearning* se pueden aplicar técnicas de manejo de desbalance de clases como sobremuestreo (oversampling) de las clases minoritarias, submuestreo (undersampling) de la clase mayoritaria, o utilizar métodos sintéticos como SMOTE(Libreria de python) para generar ejemplos de entrenamiento adicionales de las clases minoritarias.En nuestro caso se utilizara SMOTE para hacer un Oversampling de las muestras clasificadas como 'Critico' y 'Seguimiento'.

# Capítulo 4

# Análisis y discusión de resultados

## 4.1. Resultados

Todas las pruebas aplicadas a continuación fueron desarrolladas en lenguaje *Python* y utilizando la plataforma *GoogleColab*, que permite al usuario escribir y ejecutar código en el navegador de *Google*. Esta herramienta es especialmente adecuada para tareas de aprendizaje automático y análisis de datos.
A continuación se presenta los resultados para los modelos predictivos.

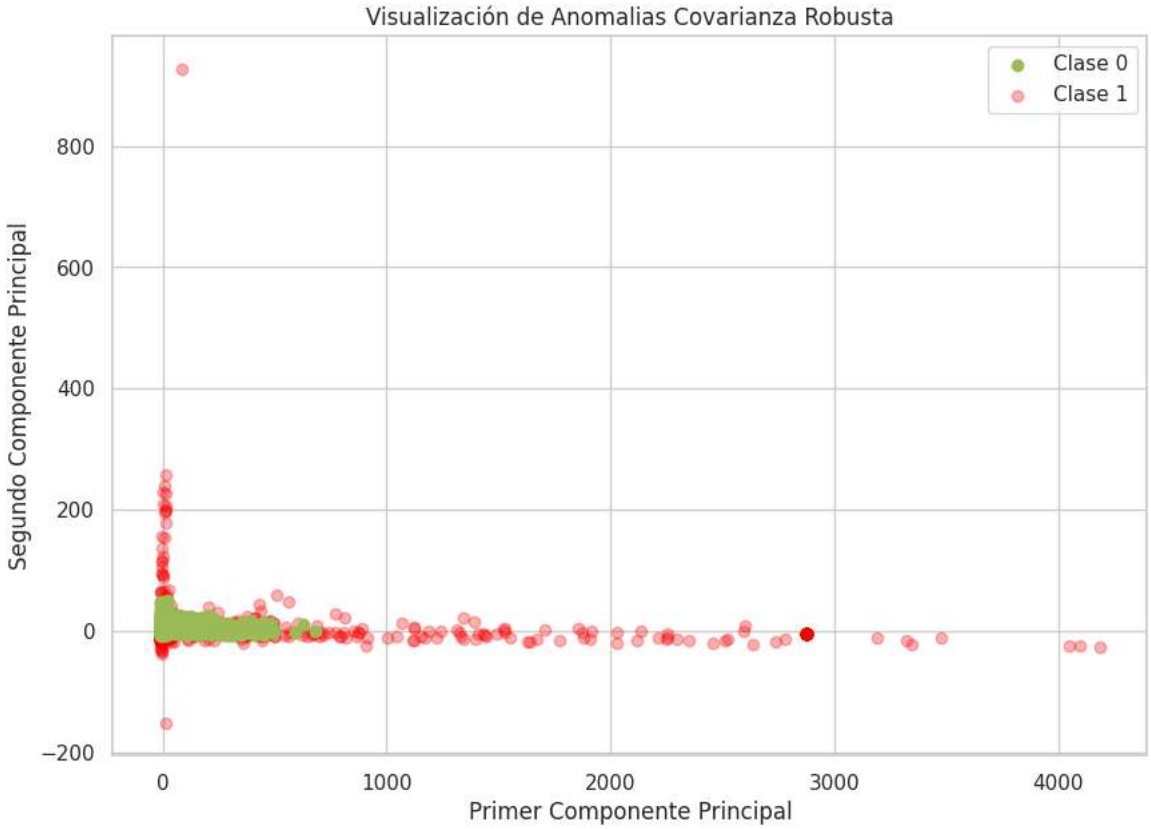
### 4.1.1. Resultado Modelo 1: Detección de anomalía

Las características seleccionadas para la tarea de detección de anomalías está directamente relacionado con posibles fallas y la salud del componente. Al monitorear estas características, se puede detectar señales tempranas de desgaste y contaminación. Por lo que para comparar su desempeño se utilizara el etiquetado de condición la clases 'Critico' y 'Seguimiento' serán la clase 'Anómala'.

**Covarianza Robusta**

La evaluación visual a través de la representación en el espacio de los componentes principales permite identificar patrones, agrupaciones y posibles anomalías en los datos.

Figura 4.1: PCA-Covarianza Robusta




En este caso es importante medir qué tan similar es un punto a su propia clase (cohesión) en comparación con la clase anómala (separación) por lo tanto se calcula el $Coeficiente\_de\_silueta = 0{,}83$, cuyo valor es muy alto e indica un buen nivel de separación entre los grupos. La matriz de confusión Figura 4.2 muestra que el modelo es muy efectivo en identificar correctamente las instancias de la clase 'Normal', con 82505 verdaderos negativos(El modelo predijo 'Normal' y la etiqueta real también es 'Normal') y 2265 falsos positivos (el modelo predijo 'Anómalo', pero la etiqueta real es 'Normal'.), resultando en una alta precisión y recall para esta clase. Sin embargo, para la clase 'Anómalo', el modelo presenta dificultades significativas, con 1686 falsos negativos(el modelo predijo 'Normal', pero la etiqueta real es 'Anómalo') y solo 5068 verdaderos positivos(el modelo predijo 'Anómalo' y la etiqueta real también es 'Anómalo'), lo que indica que muchas instancias de la clase Anómalo no son detectadas correctamente. Pero cabe resaltar que se busca hallar anomalías directamente relacionadas con desgaste y contaminación es decir la condición del motor, las etiquetas de condición también consideran la condición del componente y del aceite, por lo tanto no se espera una precisión altisima sino un buen nivel de separación entre las clases.

Figura 4.2: Matriz de confusión-Covarianza Robusta

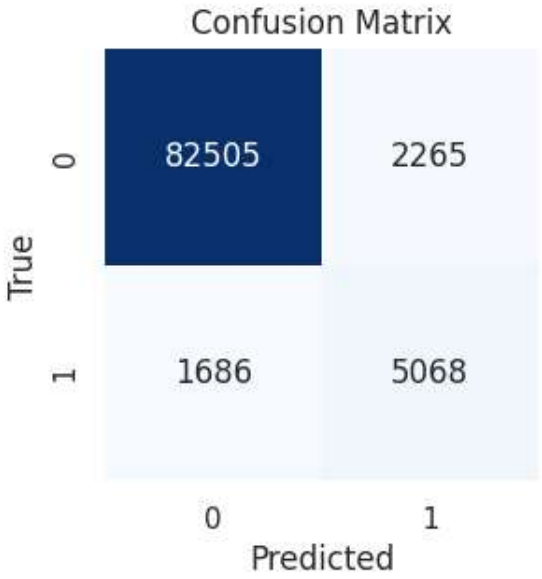


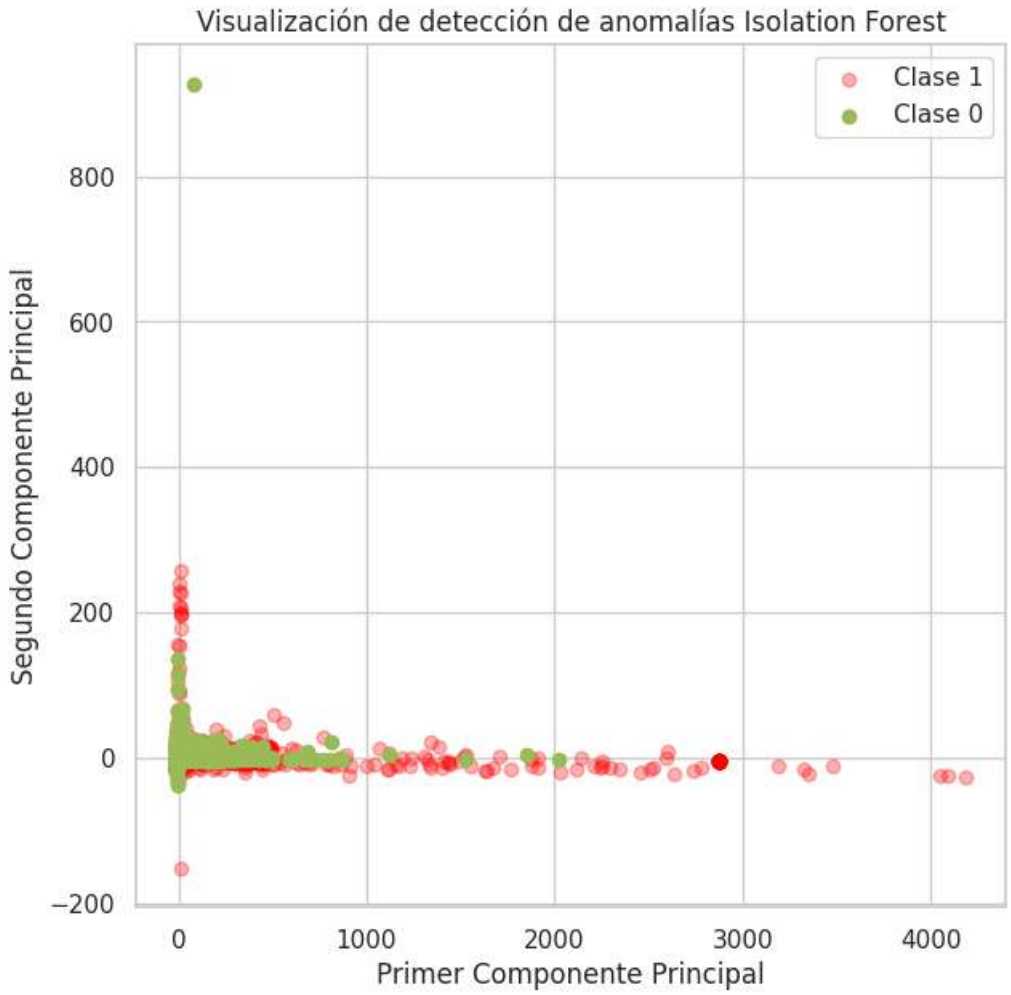

Tabla 4.1: Métricas de Covarianza Robusta frente a etiquetado de condición 'Critico' y 'Seguimiento'

| | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|
| **0** | 0.99 | 0.93 | 0.96 |
| **1** | 0.69 | 0.75 | 0.72 |

Clase Normal: El modelo tiene un rendimiento excelente en la identificación de la clase Normal, con una precisión de 0.98 y recall 0.97 altos, resultando en un F1-score alto 0.98. Esto indica que el modelo predice la clase 0 de manera muy confiable. Aunque el modelo tiene una precisión moderada al clasificar la clase Anómalo 69 %, el Recall de 0.75 indica que de todas las instancias que realmente pertenecen a la clase 1, el 75 % fueron correctamente identificadas por el modelo. Aunque no es tan alto como el recall para la clase 0, sigue siendo una tasa aceptable, mostrando que el modelo puede capturar la mayoría de las anomalías. El F1-score de 0.72 refleja un equilibrio moderado entre precisión y recall. Si bien es menos ideal que el F1-score para la clase 0, sugiere que el modelo tiene una capacidad razonable para manejar la clase de anomalías.

**Isolation Forest**, se configuró con una heurística de 0.06 Conshohocken, 2011, y el tiempo de entrenamiento fue de de apenas 3 min. La primera forma de evaluar su desempeño es mediante una evaluación visual. Aplicando el Análisis de Componentes Principales (PCA) se obtuvieron dos componente principales.

Figura 4.3: PCA-Isolation Forest



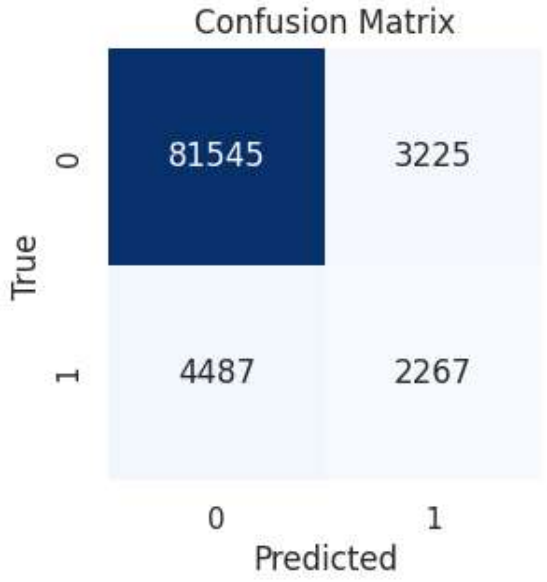

En la Figura 4.5 la mayoría de las observaciones normales (Clase 1) están agrupadas cerca del origen, lo que indica que estas observaciones comparten características similares y están bien representadas en el espacio de los componentes principales. Las anomalías (Clase -1) están dispersas y alejadas del grupo de observaciones normales, lo que sugiere que estas observaciones difieren significativamente de los datos normales y han sido correctamente identificadas como anomalías por el modelo.

| | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|
| **0** | 0.95 | 0.96 | 0.95 |
| **1** | 0.41 | 0.34 | 0.37 |

Tabla 4.2: Métricas de Isolation Forest frente a etiquetado de condición

En la Tablas 4.2 El modelo identifica correctamente el 95 % de las observaciones predichas como normales son realmente normales . El F1-Score alto indica un buen equilibrio entre precisión y recall para esta clase. La precisión del 41 % indica que de todas las instancias que el modelo predijo como pertenecientes a la clase 1, solo el 41 % realmente eran de la clase 1. Esto sugiere que hay un porcentaje significativo de falsos positivos, donde instancias normales son clasificadas incorrectamente como anómalas. El recall del 34 % indica que de todas las instancias que realmente pertenecen a la clase 1, solo el 34 % fueron correctamente identificadas por el modelo. Esto muestra que el modelo tiene dificultades para capturar las instancias anómalas. El F1-score de 0.37 refleja un equilibrio entre precisión y recall, aunque ambos son relativamente bajos. Este valor indica que el modelo tiene una capacidad limitada para manejar la clase de anomalías.

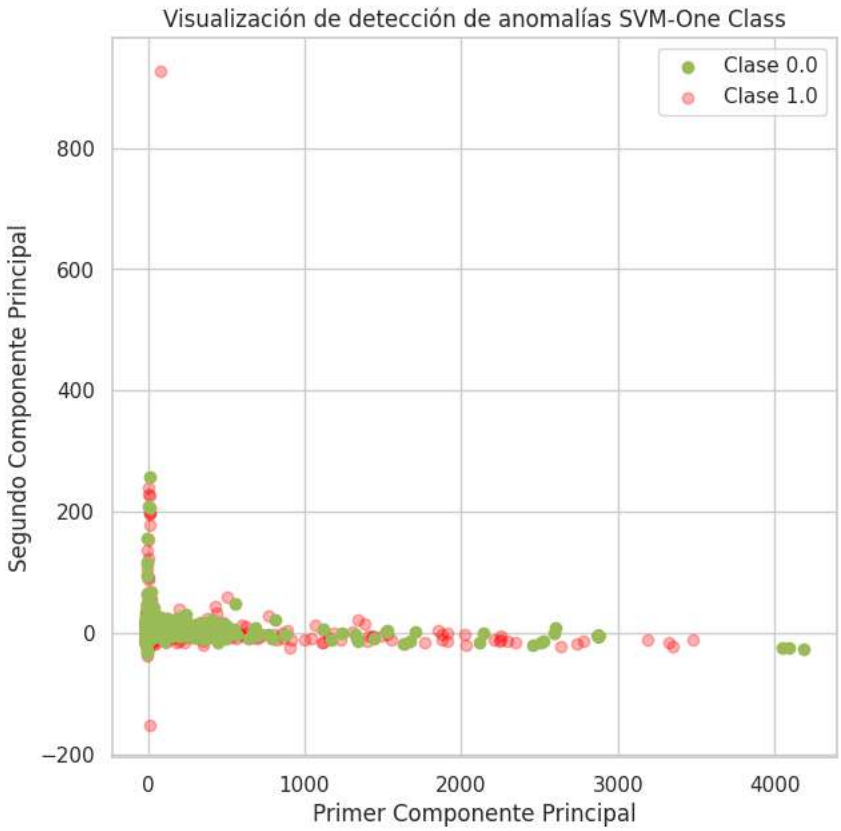
Figura 4.4: Matriz de confusión Isolation Forest frente a las clases 'Seguimiento' y 'Critico' considerados como Anómalos




La figura 4.4 se observa que el modelo tiene una alta exactitud, pero su desempeño en la detección de casos positivos (anomalías) es limitado, con baja precisión y recall. Esto indica que el modelo puede estar bien en predecir la clase mayoritaria (negativos), pero tiene dificultades en identificar correctamente las anomalías. Podrían ser necesarias técnicas adicionales de ajuste o un balance de clases para mejorar estas métricas.

**SVM-One Class** La aplicación de SVM-One Class es el mas costoso, el tiempo de entrenamiento fue de 48 minutos.

Figura 4.5: PCA-SVM-One Class

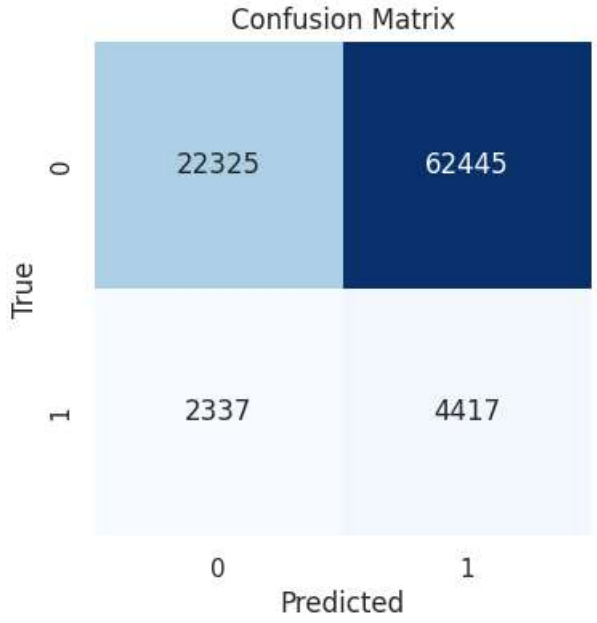



| | Precision | Recall | F1-Score |
|---|---|---|---|
| **0** | 0.91 | 0.26 | 0.41 |
| **1** | 0.07 | 0.65 | 0.12 |

Tabla 4.3: Métricas de SVM-One Class frente a etiquetado de condición

En la Tabla 4.3 para la clase 'Normal' La precisión del 91 % indica que, de todas las instancias que el modelo predijo como pertenecientes a la clase 'Normal', el 91 % realmente eran de la clase 'Normal'.El recall indica que , de todas las instancias que realmente pertenecen a la clase 0, solo el 26 % fueron correctamente identificadas por el modelo. El F1-score de 0.41 no es ideal, lo que indica una capacidad limitada del modelo para manejar la clase de normales. En cuanto a la clase 'Anómala', la precisión indica que,de todas las instancias que el modelo predijo como 'Anómala', solo el 7 % realmente eran de la clase. El recall indica que, de todas las instancias que realmente son 'Anómala', el 65 % fueron correctamente identificadas por el modelo. Aunque este valor es razonable, la baja precisión indica que muchos ejemplos normales están siendo clasificados incorrectamente como anómalos.

Figura 4.6: Matriz de confusión de SVM-One Class

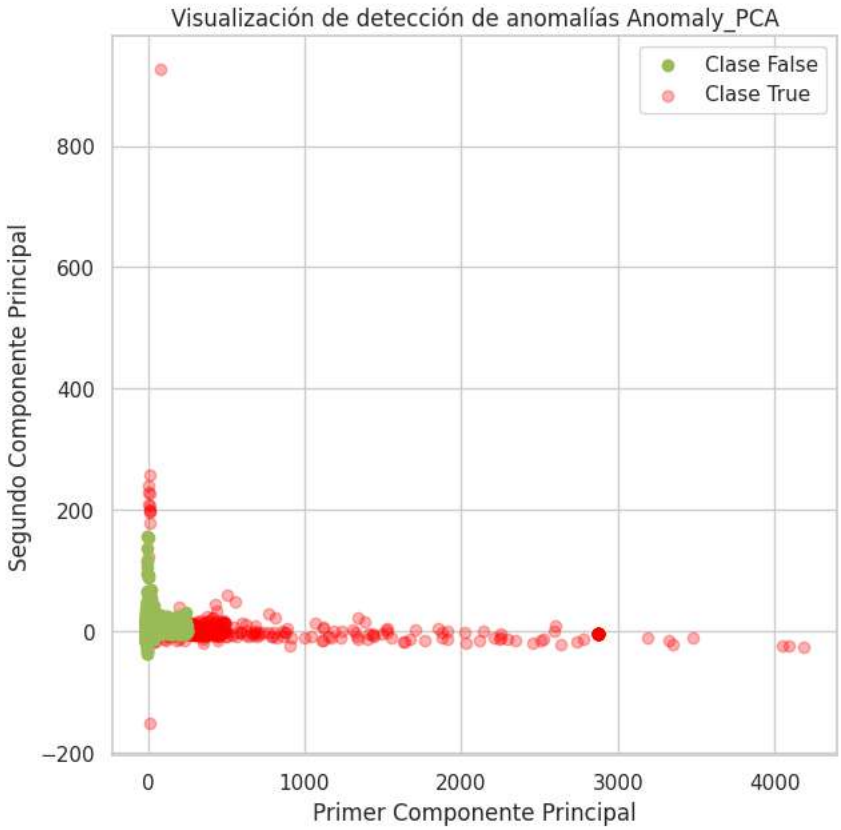



La Figura 4.6 es la matriz de confusión de SVM-One Class, se observa que hay 22,325 verdaderos negativos (casos negativos correctamente clasificados), 62,445 falsos positivos (casos negativos incorrectamente clasificados como positivos), 2,337 falsos negativos (casos positivos incorrectamente clasificados como negativos) y 4,417 verdaderos positivos (casos positivos correctamente clasificados).

**PCA-Anomaly Detection**

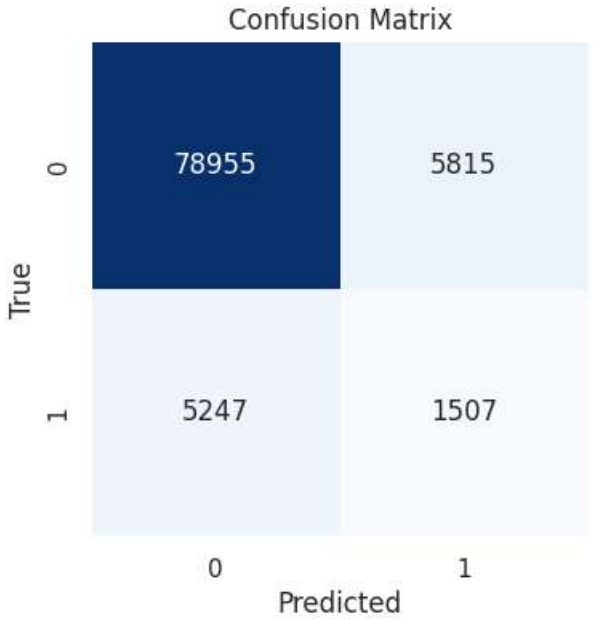
En la Figura 4.7, los puntos representan las observaciones proyectadas en el espacio de los componentes principales. Los puntos verdes (Clase Normal) corresponden a las observaciones normales, mientras que los puntos rojos (Clase Anomalo) corresponden a las anomalías detectadas por el modelo PCA-Anomaly Detection.

Figura 4.7: PCA Anomaly Detection




En la matriz de confusión Figura 4.8, hay 78,955 verdaderos negativos (casos negativos correctamente clasificados), 5,815 falsos positivos (casos negativos incorrectamente clasificados como positivos), 5,247 falsos negativos (casos positivos incorrectamente clasificados como negativos) y 1,507 verdaderos positivos (casos positivos correctamente clasificados).

Figura 4.8: Matriz de confusión - PCA Anomaly Detection




| | Precision | Recall | F1-Score |
|---|---|---|---|
| **0** | 0.94 | 0.93 | 0.93 |
| **1** | 0.21 | 0.22 | 0.21 |

Tabla 4.4: Métricas de PCA-Anomaly Detection frente a etiquetado de condición

De la tabla de métricas 4.4 se infiere que el modelo tiene un buen rendimiento para identificar instancias normales, con alta precisión y recall, lo que es crucial para la mayoría de las aplicaciones donde la mayoría de los datos son normales.

Aunque el modelo tiene dificultades para identificar correctamente las anomalías. La baja precisión y recall indican que el modelo produce muchos falsos positivos y no captura una gran parte de las verdaderas anomalías.

También se exploro la aplicación de **K-Means** en el conjunto de datos.

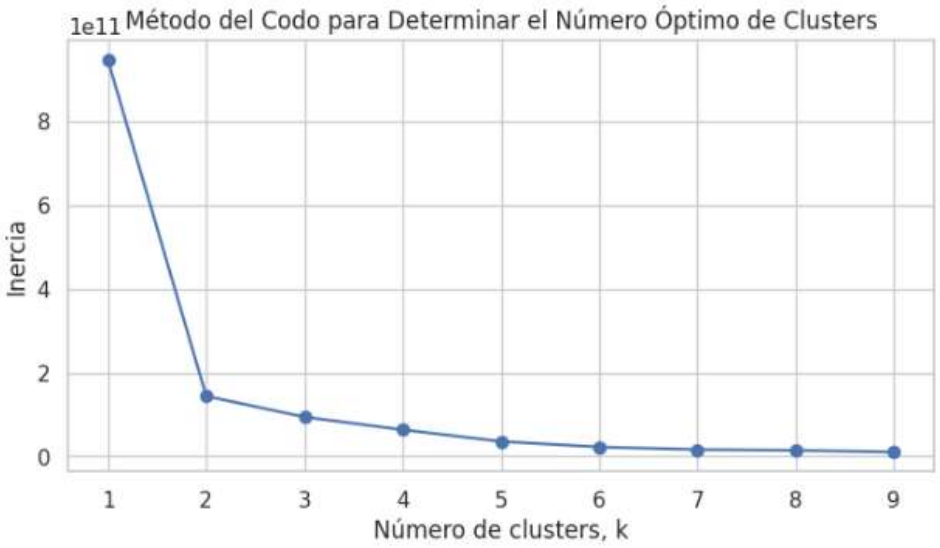



Figura 4.9: Número óptimo de clusters para K-Means.


Figura 4.10

Aunque el modelo alcanza una precisión general del 79.8 %, esta cifra debe interpretarse con cautela dado el contexto más amplio de su capacidad discriminativa y las proporciones de clase.

El valor AUC-ROC de 0.5337 destaca que la capacidad del modelo para diferenciar entre clases normales y anómalas es apenas mejor que la de un clasificador aleatorio. Esta métrica, siendo apenas superior a 0.5, indica un rendimiento marginal que apenas distingue entre las dos clases. Además, el valor extremadamente bajo del AUC-PR de 0.0653 refuerza la preocupación de que el modelo es particularmente ineficaz en la detección de la clase minoritaria, que en muchos contextos aplicados es la más crítica.

Al desglosar el rendimiento por clases, vemos que el modelo es muy eficaz en identificar instancias normales, con una precisión del 94 % y un recall del 83 %, reflejado en un F1-Score de 0.89. Estos indicadores sugieren que el modelo es competente en manejar la clase mayoritaria. Sin embargo, la situación es drásticamente diferente para la clase anómala, donde la precisión cae al 8 % y el recall al 23 %, resultando en un F1-Score de sólo 0.12. Este bajo rendimiento indica que el modelo genera un número elevado de falsos positivos y es incapaz de capturar una porción significativa de las verdaderas anomalías.

Esta diferencia en el rendimiento entre las clases puede ser atribuida en gran parte al desequilibrio de clases en el conjunto de datos. Las técnicas tradicionales de clasificación y los algoritmos como K-Means no están específicamente diseñados para manejar desequilibrios significativos, lo que puede llevar a que el modelo se ajuste predominantemente a la clase más frecuente, ignorando en gran medida la clase minoritaria que a menudo es la de mayor interés.

Figura 4.11: Matriz de confusión - KMeans

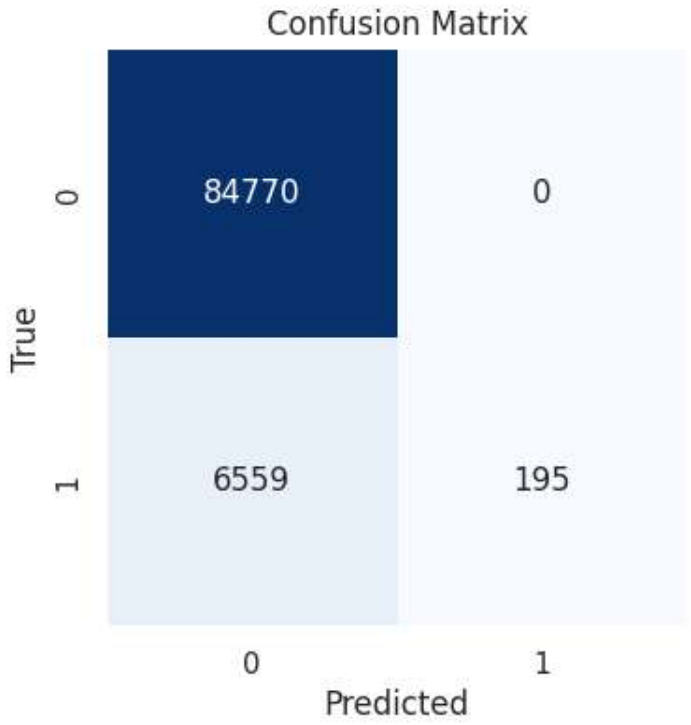



Tabla 4.5: Métricas para K-means comparado con Covarianza Robusta

| Modelo | Clase | Precisión | Recall | F1-score |
|---|---|---|---|---|
| K-Means | Normal | 0.93 | 1.00 | 0.96 |
| | Anómalo | 1.00 | 0.03 | 0.06 |

Tabla 4.6: Resultados en comparación con covarianza Robusta

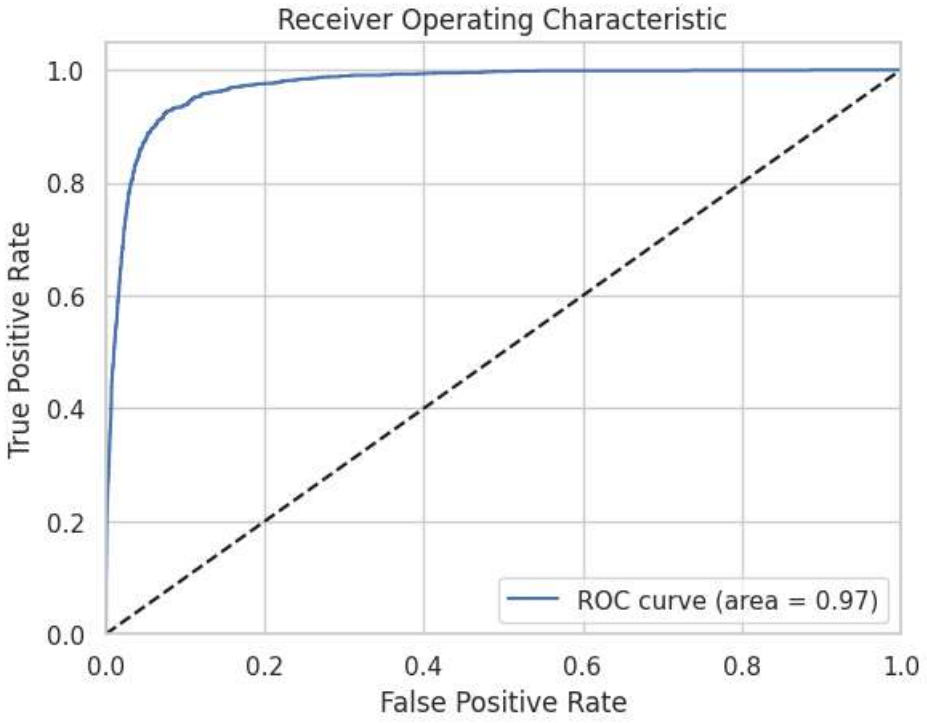
| Modelo | Clase | Precisión | Recall | F1-score |
|---|---|---|---|---|
| Covarianza Robusta | Normal | 0.98 | 0.97 | 0.98 |
| | Anómalo | 0.69 | 0.75 | 0.72 |
| SVM-One Class | Normal | 0.91 | 0.26 | 0.41 |
| | Anómalo | 0.07 | 0,65 | 0.12 |
| Isolation Forest | Normal | 0.94 | 0.93 | 0.93 |
| | Anómalo | 0.21 | 0.22 | 0.21 |
| PCA-Anomaly Detection | Normal | 0.83 | 0.95 | 0.89 |
| | Anómalo | 0.30 | 0.10 | 0.15 |
| K-Means | Normal | 0.93 | 1.00 | 0.96 |
| | Anómalo | 1.00 | 0.03 | 0.06 |

Los modelos SVM-OneClass, Isolation Forest, PCA-Anomaly Detection y K-means obtuvieron un rendimiento moderado, por ello se resuelve aplicar **Random Forest**, sobre la data etiquetada con Covarianza Robusta para generalizar la reglas en un modelo de machine learning y darle mayor explicabilidad al modelo entrenado. Se utilizo la librería *sklearn* y los módulos $RandomForest\ Classifier$ y para optimizar el algoritmo se utilizó el módulo $GridSearchCV$ que se enfoca en la experimentación iterativa para seleccionar los parámetros que mejor se ajustan a los datos:

- **n_estimator**: Número de árboles en el Bosque.$n\_estimator = \{15, 20, 25, 30\}$
- **max_depth**: La profundidad máxima del árbol.$max\_depth = \{10, 12, 15, 17\}$
- **min_samples_split**: Número mínimo de muestras requeridas para dividir un nodo interno del árbol.$max\_depth = \{2, 5, 10\}$
- **min_samples_leaf**: Número mínimo de muestras requeridas en un hoja del árbol. $min\_samples\_leaf = \{3, 5, 10\}$
- **classifier__criterion**: Medida de calidad de una división
  $classifier\_\_criterion = \{"giny", "entropy"\}$

Además se crea un pipeline que incluye el sobremuestreo SMOTE para mejorar los resultados. Para evitar el sobreajuste se utiliza una validación cruzada de 7 pliegues.

Figura 4.12: Curva ROC Random Forest para detección de anomalías con Covarianza Robusta

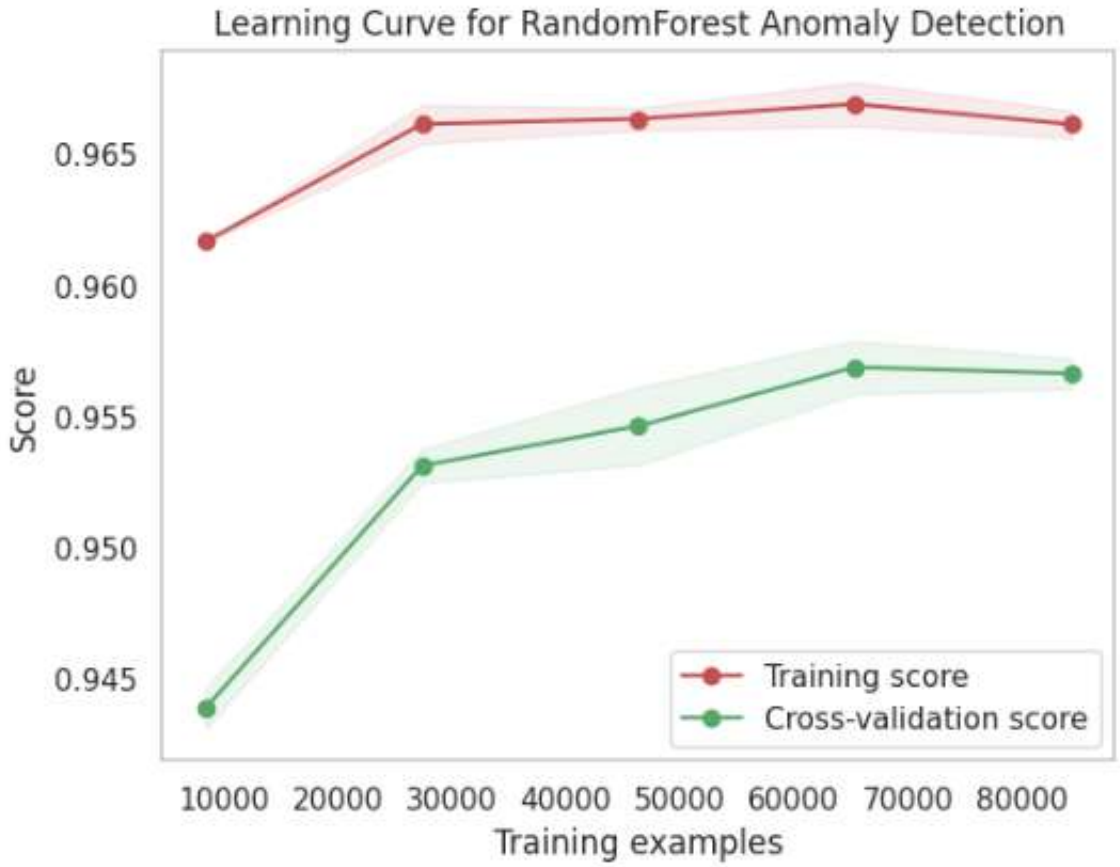



Los parámetros mejor puntuados son $max\_depth = 20$,$min\_samples\_split = 3$,$n\_estimators = 20$ con una precisión de 97.74 %, permitiendo estabilizar la curva de aprendizaje a medida que se aumenta el número de muestras:

Tabla 4.7: Metricas Random Forest para Anomalias

| Modelo | Clase | Precisión | Recall | F1-score |
|---|---|---|---|---|
| Random Forest | Normal | 0.99 | 0.99 | 0.99 |
| | Anómalo | 0.89 | 0.93 | 0.91 |

Figura 4.13: Curva de aprendizaje modelo Random Forest




El informe de clasificación Tabla 4.7 proporciona una visión más detallada del rendimiento del modelo para cada clase. La clase 0 (Normal) tiene una precisión, recall y f1-score de 0.99, lo que indica que el modelo es casi perfecto en la identificación de instancias normales. La clase 1 (Anómalo) tiene una precisión de 0.89, un recall de 0.93 y un f1-score de 0.91, lo que sugiere que el modelo también es muy efectivo en la detección de anomalías, aunque no tan preciso como para la clase normal.

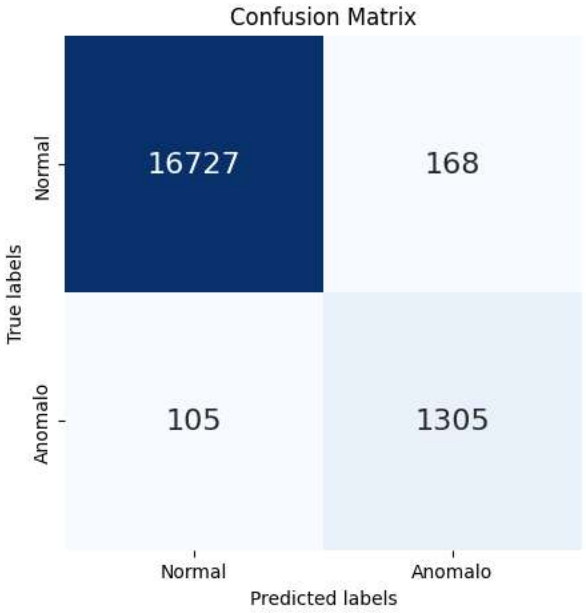
La curva de aprendizaje presentada en la Figura 4.13 muestra el desempeño de un modelo de Random Forest utilizado para la detección de anomalías en función del número de ejemplos de entrenamiento. A través del gráfico, se observan dos tendencias principales que proporcionan información valiosa sobre el comportamiento y la eficacia del modelo a medida que se incrementa el volumen de datos de entrenamiento.

**Puntaje de Entrenamiento (Línea Roja)** La curva de entrenamiento comienza con un alto rendimiento y se mantiene relativamente estable en todo el rango de ejemplos de entrenamiento. La leve disminución en el score al aumentar el número de ejemplos sugiere que a medida que el modelo se expone a más datos, se ajusta a una variabilidad más amplia, lo que puede llevar a una ligera degradación en el rendimiento en el conjunto de entrenamiento. Sin embargo, la estabilidad generalmente alta del puntaje de entrenamiento cerca del 96.5 %

indica que el modelo es consistentemente bueno en aprender de los datos de entrenamiento sin signos significativos de dificultades para adaptarse a nuevos datos.

**Puntaje de Validación Cruzada (Línea Verde)** Esta curva muestra una tendencia ascendente notable, comenzando en alrededor de 94.5 % y mejorando significativamente a medida que el modelo se entrena con más datos, alcanzando cerca del 95.5 %. El aumento constante en la precisión de la validación cruzada sugiere que el modelo está ganando la capacidad de generalizar mejor a medida que recibe más datos. Esto es un indicador de que los datos adicionales están ayudando al modelo a capturar mejor la esencia de los patrones subyacentes en los datos, mejorando su rendimiento en datos no vistos durante el entrenamiento. La convergencia gradual de las curvas de entrenamiento y validación es un signo saludable de que el modelo no está sobreajustado, ya que el desempeño en los datos no vistos se acerca al desempeño en los datos de entrenamiento.

Figura 4.14: Matriz de confusión Covarianza Robusta y condición de la muestras

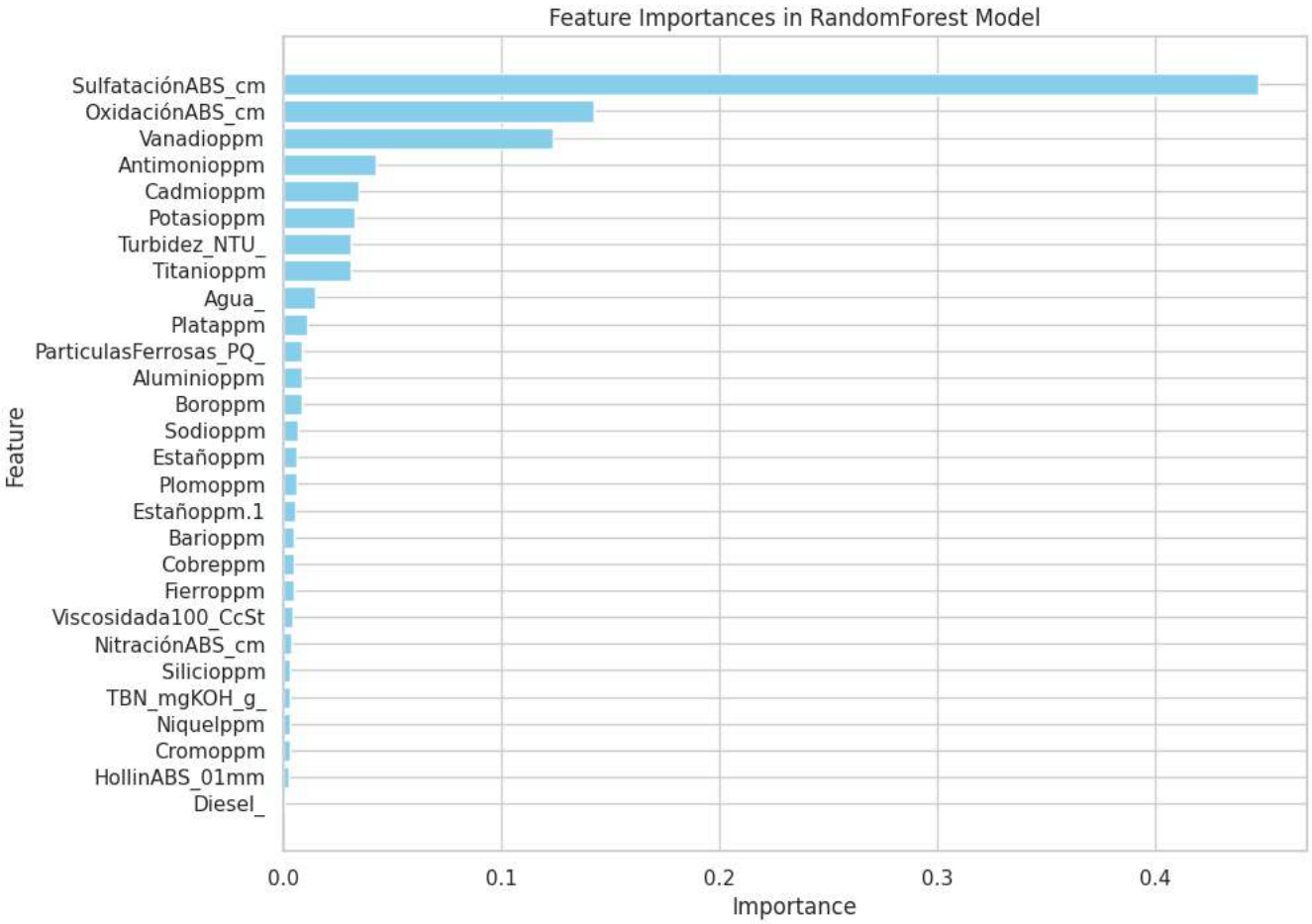



El modelo entrenado ha demostrado un rendimiento notablemente alto en la clasificación, alcanzando una precisión general del 98.51 %. La matriz de confusión Figura 4.14 muestra que de las 16,895 instancias etiquetadas como clase 0 (Normal), 16,727 fueron correctamente identificadas, mientras que solo 168 fueron clasificadas erróneamente. Por otro lado, de las 1,410 instancias etiquetadas como clase 1 (Anómalo), 1,305 fueron correctamente identificadas y 105 fueron clasificadas erróneamente.

**Interpretación del modelo** El gráfico de importancia de características en el modelo de Random Forest representa la relevancia de cada variable en la detección de anomalías en las muestras de aceite. Cada barra del gráfico indica cuánto contribuye una característica específica al desempeño del modelo. Las características más importantes aparecen en la

parte superior y tienen las barras más largas, mientras que las menos importantes están en la parte inferior con barras más cortas. El gráfico ayuda a identificar cuáles son las variables más influyentes en la detección de anomalías. En este caso, variables Turbidez, Titanio y Sodio son las más relevantes.

Figura 4.15: Importancia de características en el modelo de Random Forest




Cuando una nueva muestra sea clasificada como anómala se incluirá el siguiente gráfico:

Figura 4.16: Características más influyentes en la predicción de Random Forest para una instancia dadat

En la Figura 4.16 cada barra representa el efecto de una característica en la predicción de la muestra medido en términos de cambio en la predicción desde el valor base. El valor base es el valor esperado del modelo sobre el conjunto de datos. **Dirección y Color**, el color rojo indica que la característica está aumentando la predicción. Esto significa que un valor más alto de esta característica empuja la predicción hacia un valor más alto (o hacia la clase positiva en clasificación).El color azul indica que la característica está disminuyendo la predicción. Un valor más alto de esta característica empuja la predicción hacia un valor más bajo (o hacia la clase negativa en clasificación). El **Eje Horizontal** representa la magnitud del efecto de cada característica en la predicción del modelo. Las barras pueden extenderse hacia la izquierda o la derecha del valor base, mostrando su contribución positiva o negativa. En la Figura 4.16 'Turbidez_NTU_' parece tener un fuerte impacto positivo en la predicción, indicado por una barra larga azul extendiéndose hacia la derecha. En el contexto de SHAP, un efecto positivo significa que un valor más alto de 'Turbidez_NTU_' está asociado con mover la predicción hacia el valor más alto o la clase positiva.

### 4.1.2. Resultado Modelo 2: Clasificar condición de la muestra

El primer modelo probado es **Random Forest**. Se configura 'GridSearch' para realizar una búsqueda exhaustiva de los mejores hiperparámetros del modelo. Definición de hiperparametros:

- **n_estimator**: Número de árboles en el Bosque.$n\_estimator = \{10, 200, 300\}$
- **max_depth**: La profundidad máxima del árbol.$max\_depth = \{10, 15, 20\}$
- **min_samples_split**: Número mínimo de muestras requeridas para dividir un nodo interno del árbol.$max\_depth = \{3, 5, 10\}$
- **min_samples_leaf**: Número mínimo de muestras requeridas en un hoja del árbol. $min\_samples\_leaf = \{3, 5, 10\}$
- **classifier__criterion**: Medida de calidad de una división
  $classifier\_\_criterion = \{"giny", "entropy"\}$
- **class_weight**: Balanced

Se extrae el mejor modelo encontrado durante la búsqueda

$max\_depth = 12$,$min\_samples\_split = 5$,$n\_estimators = 15$ , $classifier\_\_criterion =$ *entropy*y se utiliza para hacer predicciones en un conjunto de prueba. Se crea un pipeline usando ImbPipeline de imblearn. Este pipeline incluye dos pasos:

- 'SMOTE': Aplicación de la técnica de sobremuestreo SMOTE para balancear las clases en el conjunto de entrenamiento.

- 'randomforestclassifier': Entrenamiento del clasificador 'RandomForestClassifier' con 'class_weight='balanced" para manejar el desbalanceo de clases.

Se utiliza StratifiedKFold para realizar una validación cruzada estratificada con 7 divisiones (n_splits=7), lo que asegura que cada fold tenga aproximadamente la misma proporción de clases que el conjunto de datos original. Se especifica una métrica de evaluación adecuada para datos desbalanceados, como el f1 score.

Tabla 4.8: Métricas para la clasificación de condición de muestras

| Modelo | Clase | Precisión | Recall | F1-score |
|---|---|---|---|---|
| Random Forest | Normal | 1 | 0.99 | 0.99 |
| | Seguimiento | 0.80 | 0.96 | 0.87 |
| | Critico | 0.95 | 0.90 | 0.92 |

En la Tabla 4.10 se observa que el desempeño del modelo varía significativamente entre las diferentes clases. Para la clase 'Crítico', el modelo muestra una precisión excepcionalmente alta de 0.95, indicando que el 95 % de muestras clasificadas como 'Critico' son 'Critico' en realidad. El recall de 89 % indica que de los verdaderos casos 'Critico' fueron correctamente identificados por el modelo.

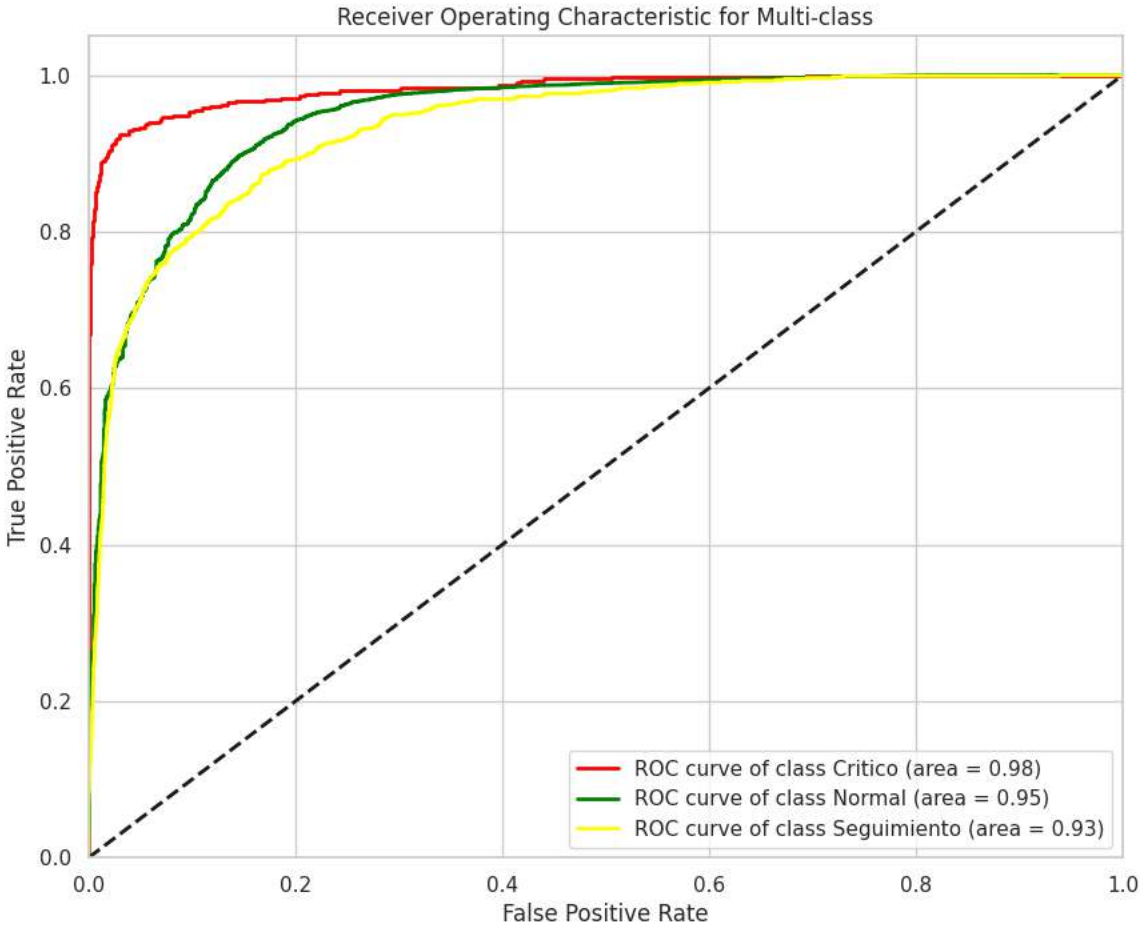
Para la clase 'Normal', la precisión fue del 99 % y el recall del 99 %, resultando en un F1-score de 0.99. Estas métricas indican que el modelo fue altamente efectivo en identificar correctamente las muestras normales, y logró capturar la mayoría de las instancias reales de esta clase.

Sin embargo, La clase 'Seguimiento' muestra un buen balance entre precisión y recall, aunque con una precisión más baja (0.80), sigue siendo altamente confiable en términos de detección (recall de 0.96).

Figura 4.17: Matriz de confusión Random Forest para muestras etiquetadas en condición

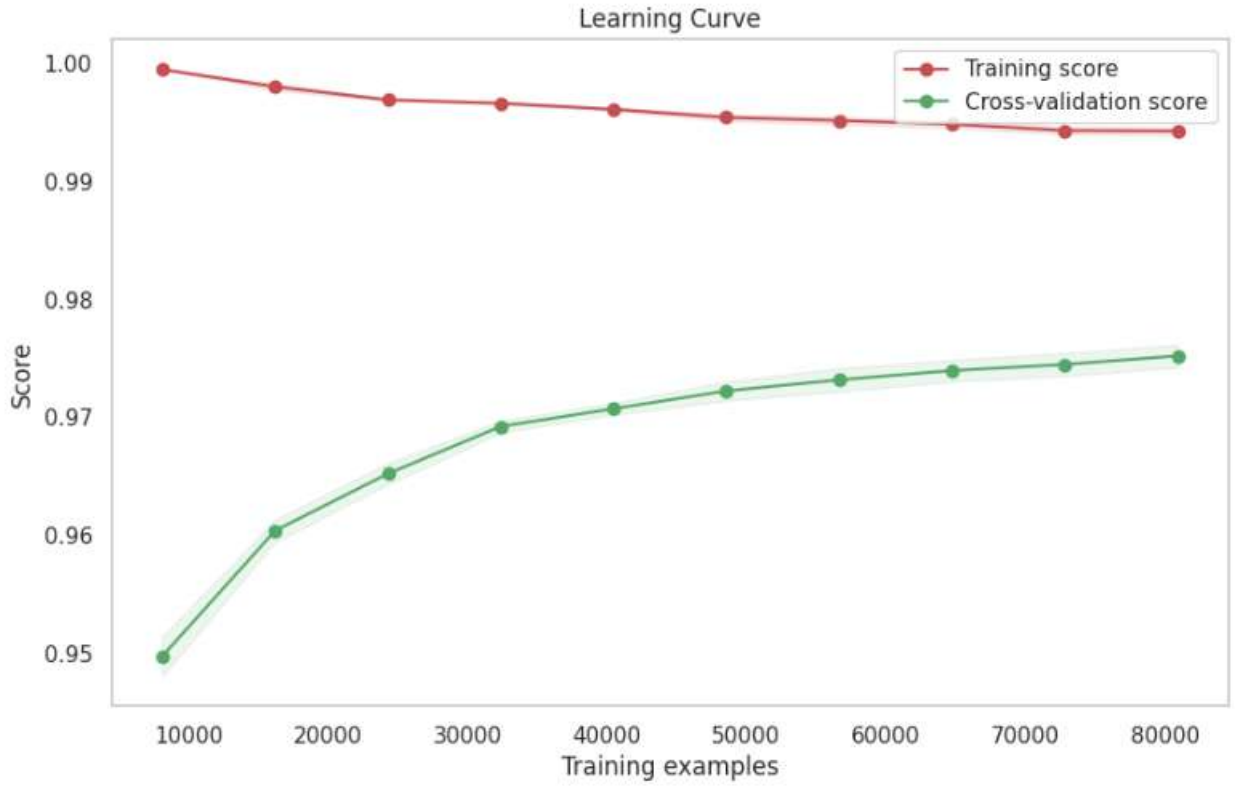

En la Figura 4.17 Para la clase 'Critico' el modelo clasificó correctamente 399 de los 449 casos críticos.No hay predicciones de "Normal"para los casos críticos. Y, clasificó incorrectamente 50 casos críticos como "Seguimiento". Para la clase 'Normal' el modelo clasificó correctamente 16837 de los 17006 casos normales. Seis casos críticos fueron clasificados incorrectamente como normales.Y clasificó incorrectamente 163 casos normales como "Seguimiento". Para la clase 'Seguimiento' el modelo clasificó correctamente 823 de los 850 casos de seguimiento.20 casos críticos fueron clasificados incorrectamente como seguimiento y se clasificó incorrectamente 7 casos de seguimiento como normales.

Figura 4.18: Curva ROC multiclase para condición




En general, **las curvas AUCs** Figura 4.18 para las tres clases son muy altas (0.99, 0.94 y 0.91), lo que sugiere que el modelo tiene un desempeño excelente en la clasificación de todas las clases. La comparación entre clases muestra que la clase Critico tiene el mejor AUC (0.99), seguida de la clase Normal (0.94), y luego la clase Seguimiento (0.96). Aunque todas las clases tienen un muy buen desempeño, el modelo es ligeramente mejor en identificar la clase Critico.

Figura 4.19: Validación cruzada




La curva de aprendizaje Figura 4.19 muestra cómo cambian las puntuaciones de entrenamiento y validación a medida que se incrementa el número de ejemplos de entrenamiento.

La curva de entrenamiento, representada por la línea roja, muestra que la puntuación de entrenamiento es consistentemente alta, cercana a 1.0, y presenta una ligera disminución a medida que se incrementa el tamaño del conjunto de entrenamiento. Esto indica que el modelo tiene un muy buen rendimiento en el conjunto de entrenamiento, aunque se observa una ligera caída conforme aumenta el tamaño del conjunto de entrenamiento.

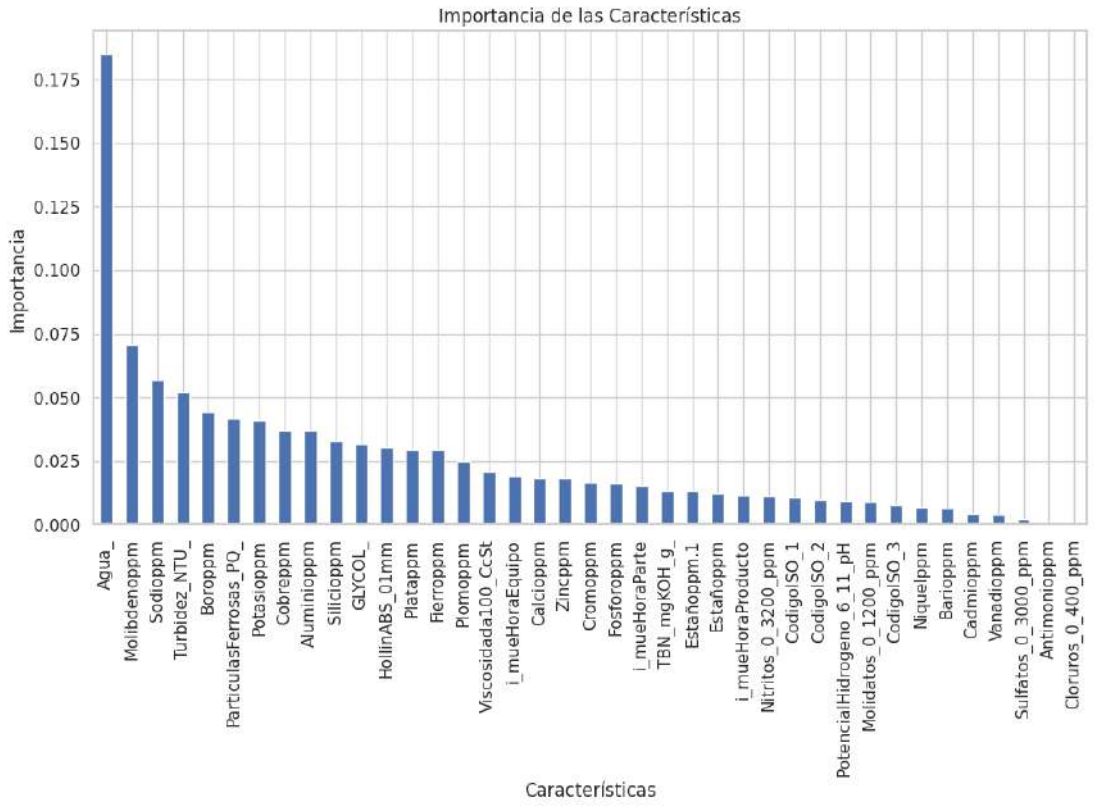
Por otro lado, la curva de validación cruzada, representada por la línea verde, empieza con una puntuación más baja que la de entrenamiento y aumenta a medida que se incrementa el tamaño del conjunto de entrenamiento. A medida que se incrementa el tamaño del conjunto de entrenamiento, la puntuación de validación también aumenta y se estabiliza, alcanzando un punto cercano a la puntuación de entrenamiento.

La alta puntuación de entrenamiento comparada con la menor puntuación de validación en los tamaños de entrenamiento más pequeños sugiere que el modelo estaba inicialmente sobreajustado (overfitting) a los datos de entrenamiento. Esto significa que el modelo se ajustaba muy bien a los datos de entrenamiento, pero no generalizaba bien a datos nuevos. Sin embargo, la mejora en la puntuación de validación con el aumento del tamaño del conjunto de entrenamiento sugiere que el modelo se beneficia de tener más datos de entrenamiento. Esto

es común, ya que más datos tienden a ayudar al modelo a aprender patrones más generales y a mejorar su capacidad de generalización.

La estabilización de la puntuación de validación cruzada en los tamaños de entrenamiento más grandes indica que el modelo está alcanzando un punto donde añadir más datos de entrenamiento no proporciona mejoras significativas en la precisión del modelo. En este punto, el modelo está bien generalizado y su rendimiento en datos nuevos es consistente. El hecho de que las curvas de entrenamiento y validación estén cerca una de la otra en los tamaños de conjunto de entrenamiento más grandes sugiere que el conjunto de datos disponible es suficiente para el modelo. El modelo no parece estar sufriendo de underfitting (subajuste), ya que la puntuación de validación es alta.

Figura 4.20: Importancia de la característica por permutación




La gráfica de importancia de las características Figura 4.20 muestra que 'Agua' es la variable más influyente en la predicción de la condición de las muestras, seguida por 'Molibdemo'. La alta importancia de'Agua' sugiere que el contenido de agua en las muestras es un factor crítico para determinar su estado.

Estas características clave deben ser monitoreadas de cerca en aplicaciones prácticas para asegurar un correcto mantenimiento y predicción del estado de las muestras. Las características con menor importancia, como 'PotencialHidrogeno_6_11_pH' y 'Nitritos_0_3200_ppm',

aunque tienen una contribución mínima, no deben ser completamente ignoradas ya que podrían tener relevancia en contextos específicos.

Estos hallazgos son consistentes con estudios previos que han destacado la importancia del contenido de Agua y Sodio en la determinación de la condición de materiales similares. En resumen, el modelo *RandomForest* ha identificado de manera efectiva las variables más críticas, proporcionando una base sólida para la toma de decisiones en el manejo y mantenimiento de las muestras.

#### 4.1.2.1. Explicabilidad del Modelo

Para interpretar el gráfico de SHAP (SHapley Additive exPlanations), se selecciona una muestra aleatoria y se clasifica su condición con el modelo Random Forest:

Figura 4.21: Valores SHAP

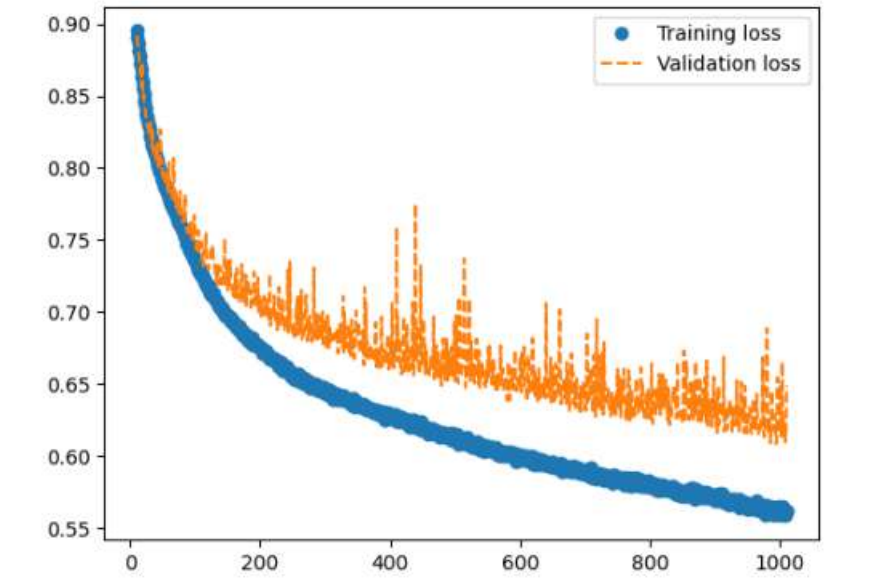

- En el gráfico, f(x) es 0.02. Esto significa que el modelo predice un valor de 0.02 para esta instancia en particular.
- El valor base es el valor promedio de la predicción en el modelo cuando ninguna característica se utiliza. En este gráfico, el valor base es 0.3335. Es el punto de partida desde donde se ajusta el valor predicho.
- HollinABS_01mm y Fierroppm son los principales impulsores que aumentan la predicción hacia 0.02.
- Las características como Molibdenoppm, Sodioppm, Potasio ppm, Boroppm y Aluminioppm son los principales impulsores que disminuyen la predicción desde el valor base.

El segundo algoritmo con el que se experimentó es la **Red Neuronal Artificial** . Se probaron las siguientes configuraciones para encontrar la arquitectura de red adecuada para modelas loas datos:

1. Número de capas y neuronas por capa.
2. Funciones de activación: Se probaron ReLU (Rectified Linear Unit), sigmoide o tangente hiperbólica.

3. Optimizador: SGD (descenso de gradiente estocástico), Adam, RMSprop.
4. Tasa de aprendizaje: Learning Rate=0.01 - 0.1.
5. Tamaño del lote (batch size)=10,50,80,100,500,1000
6. Número de épocas=50,100,200,300,400,500,600
7. Regularización: Se aplicaron técnicas de regularización como L1 o L2 para evitar el sobreajuste.
8. Función de pérdida: La entropía cruzada categórica o binaria, el error cuadrático medio, categorical crossentropy.
9. Métricas de evaluación: Además de la función de pérdida, las métricas que utilizadas para evaluar el rendimiento de la red: la precisión, el F1-score, la sensibilidad, la especificidad, el AUC.

Figura 4.22: Número de capas: 4, número de neuronas: 20,16,5,3




En la Figura 4.22 se muestra dos métricas de pérdida, la pérdida de entrenamiento (en azul) y la pérdida de validación (en naranja). Estas métricas indican qué tan bien el modelo está aprendiendo durante el entrenamiento y qué tan bien generaliza a datos no vistos, respectivamente. La línea azul sólida representa la pérdida de entrenamiento, que disminuye constantemente a medida que aumenta el número de épocas. Esto indica que el modelo está aprendiendo efectivamente de los datos de entrenamiento y mejorando su rendimiento en este conjunto a lo largo del tiempo.

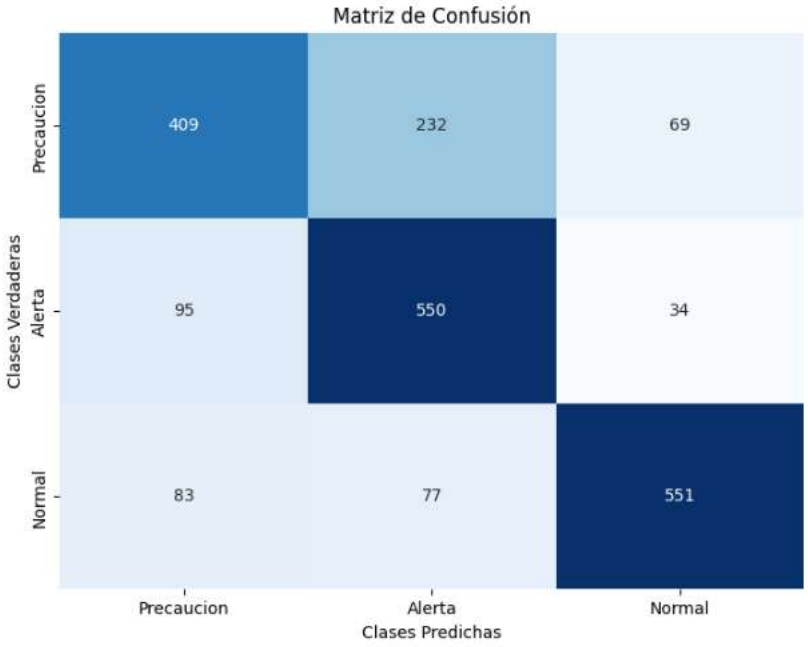
La línea naranja discontinua representa la pérdida de validación. Aunque esta también disminuye al principio, muestra mucha más variabilidad y picos, lo cual puede indicar sobreajuste o una capacidad de generalización menos estable.

Sobreajuste Potencial: El hecho de que la pérdida de validación sea más alta y más variable que la pérdida de entrenamiento, especialmente en las últimas épocas, puede ser señal de sobreajuste. Esto significa que el modelo podría estar aprendiendo patrones específicos de los datos de entrenamiento que no se aplican a los datos de validación.

Tabla 4.9: Métricas principales

| Métrica | Valor |
|---|---|
| Pérdida | 0.9306 |
| Accuracy | 0.729 |
| Recall | 0.729 |
| Especificidad | 0.63 |
| F1-Score | 072 |

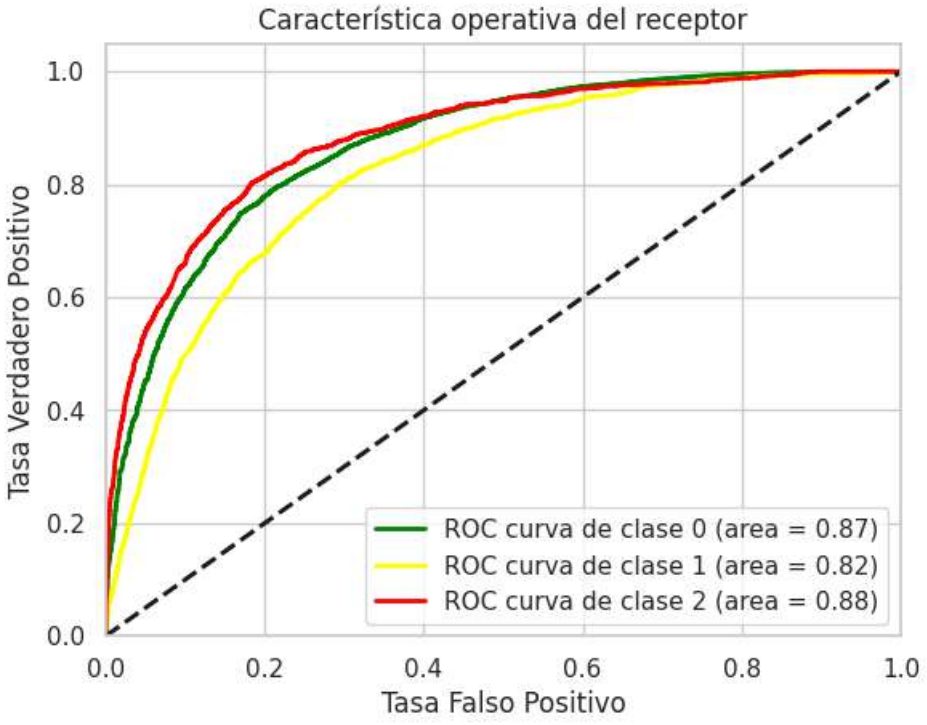
Figura 4.23: Matriz de confusión para la red neuronal




En la matriz de confusión Figura 4.23 se observa que la Diagonal Principal (409 para 'Precaucion', 550 para 'Alerta' y 551 para 'Normal') representan el número de predicciones correctas para cada clase. Los valores fuera de la diagonal principal son los errores de clasificación y se observa 95 casos que eran realmente 'Alerta' fueron clasificados incorrectamente como 'Precaucion', y 83 casos que eran realmente 'Normal' también fueron clasificados incorrectamente como 'Precaucion'. Rendimiento por Clase: La clase 'Normal' tiene la mayor

cantidad de predicciones correctas (551) y la menor cantidad de falsos positivos (83 para 'Precaucion' y 77 para 'Alerta'), lo que sugiere que el modelo es bastante bueno para identificar esta clase. La clase 'Alerta' también se clasifica razonablemente bien con 550 predicciones correctas, aunque hay una cantidad considerable de casos de 'Precaucion' que se confunden con 'Alerta' (232 casos). La clase 'Precaucion' tiene el menor número de predicciones correctas (409) y es la que más a menudo se confunde con 'Alerta'. Se puede consluir que el modelo tiene dificultades para distinguir entre 'Precaucion' y 'Alerta', ya que hay un número significativo de predicciones incorrectas entre estas dos clases. Todas las clases tienen un número significativo de predicciones correctas, lo que sugiere que el modelo está relativamente bien equilibrado en su capacidad para identificar cada clase. Sin embargo, puede haber espacio para mejorar la distinción entre 'Precaucion' y 'Alerta', quizás revisando las características utilizadas para el modelo o ajustando los parámetros del modelo.

Figura 4.24: Curva ROC por clase




La Figura 4.24 es la Curva Característica de Operación del Receptor (ROC) con tres líneas que representan la actuación de un clasificador para las tres clases. Cada curva ROC representa la habilidad del modelo para clasificar correctamente cada clase en comparación con todas las demás clases. El modelo es capaz de identificar las tres clases con un rendimiento superior al azar (indicado por la línea punteada que representa un clasificador aleatorio con AUC = 0.5). Las curvas ROC de las tres clases se encuentran significativamente por encima de la línea punteada diagonal, que representa un clasificador aleatorio. Esto indica que el modelo tiene un rendimiento significativamente mejor que un clasificador aleatorio para todas las clases.

Áreas Bajo la Curva (AUC):

- La clase 'Normal' (AUC = 0.87) tiene la AUC más alta, lo que sugiere que el modelo es más hábil para distinguir entre los casos 'Normal' y no 'Normal' que las otras clases.
- La clase 'Alerta' (AUC = 0.88) también muestra un buen rendimiento, aunque ligeramente inferior a la clase 'Normal'.
- La clase 'Precaución' (AUC = 0.82) tiene la AUC más baja de las tres, indicando que el modelo encuentra esta clase un poco más difícil de distinguir en comparación con las otras.

Comportamiento en Diferentes Umbrales: En la parte inicial de las curvas, especialmente para la clase 'Normal', hay un rápido aumento en la tasa de verdaderos positivos con un bajo incremento en la tasa de falsos positivos. Esto es deseable y muestra que se puede obtener una alta tasa de verdaderos positivos sin incurrir en muchos falsos positivos. Para las clases 'Alerta' y 'Precaución', las curvas son menos empinadas al principio, lo que indica que hay una compensación mayor entre aumentar los verdaderos positivos y controlar los falsos positivos.

### 4.1.3. Voto Mayoritario

Los modelos anteriores fueron entrenados con el conjunto de datos desde 2014 hasta 2023. Para comprobar el desempeño de ambos modelos en conjunto para hallar anomalías, se utilizan el conjunto de datos con muestras solo del 2024. Para combinar las predicciones de varios modelos individuales y seleccionar la predicción que recibe la mayoría de los votos se utilizó 'Voto Mayoritario' es una técnica utilizada en los ensambles de modelos de machine learning para mejorar la precisión y la robustez de las predicciones. Los resultados de aplicar esta técnica se obtuvo:

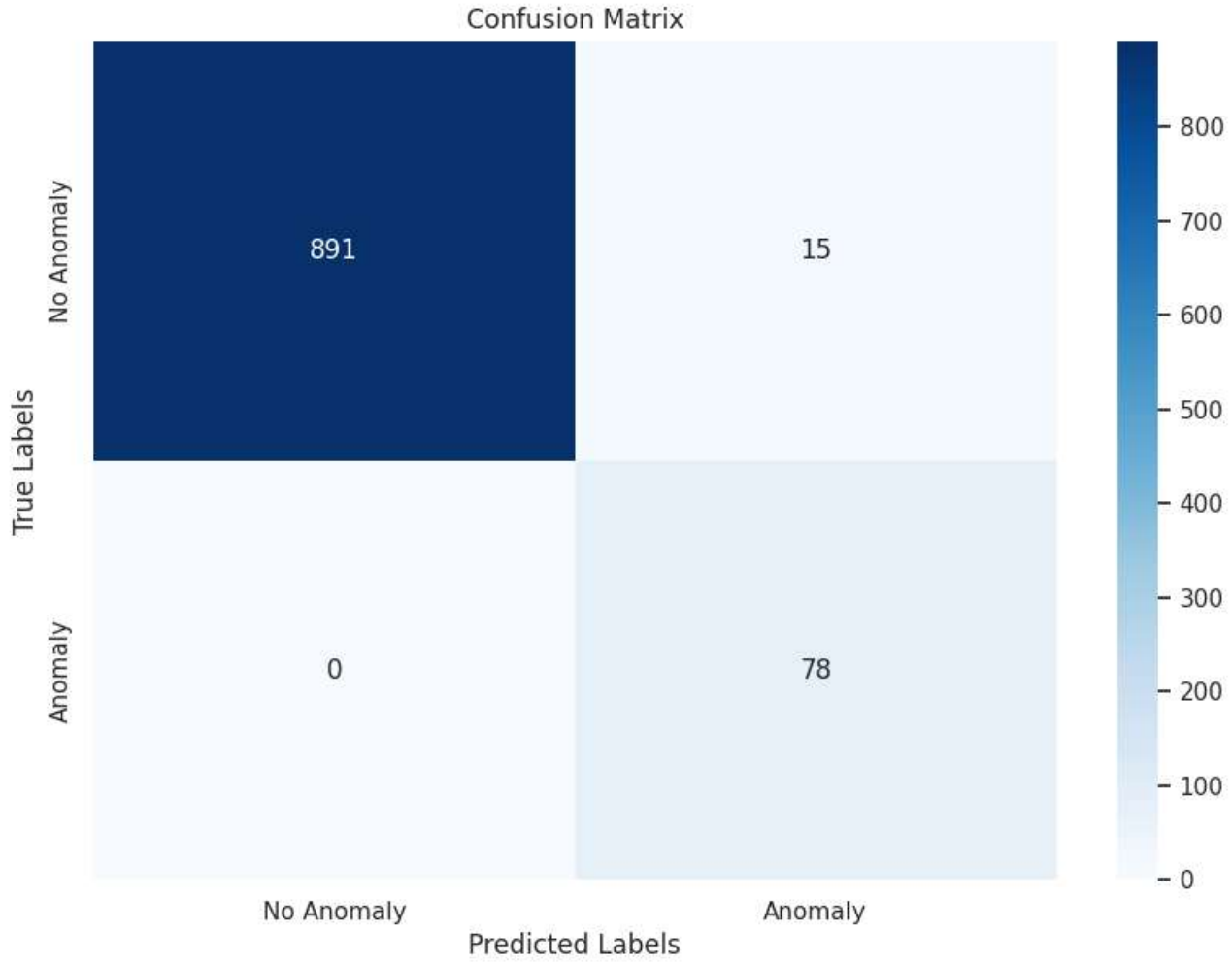
Tabla 4.10: Métricas para la clasificación de condición de muestras con el conjunto de datos de prueba 2024

| Modelo | Clase | Precisión | Recall | F1-score |
|---|---|---|---|---|
| | Normal | 1 | 0.98 | 0.99 |
| Voto Mayoritario | Anomalo | 0.84 | 1.00 | 0.91 |

La precisión de 1.00 en "Normal"muestra que el modelo casi no comete errores de falso positivo para esta clase. El recall de 1.00 en .ªnomaloïndica que el modelo no pasa por alto ningún caso de anomalía, capturando todos los eventos anómalos. El alto F1-Score en ambas clases sugiere que el modelo tiene un rendimiento equilibrado, siendo muy efectivo tanto en la identificación de casos normales como de anomalías. Estos resultados demuestran que el modelo combinado es altamente fiable y eficiente en la clasificación de las muestras,

con una excelente capacidad para identificar anomalías sin comprometer la precisión en la clasificación de casos normales.

Figura 4.25: Matriz de confusión aplicando Voto Mayoritario en ambos Modelos




La matriz Figura 4.25 muestra un rendimiento excepcional en la identificación de casos de "Normal", con un recall de 0.98 y una precisión de 1.00. Esto implica que el modelo casi no comete errores al clasificar muestras normales. En cuanto a la detección de anomalías, el modelo también muestra un buen rendimiento con una precisión de 0.84 y un recall de 1.00, asegurando que no se pierda ninguna muestra anómala. En general, el modelo demuestra una alta capacidad para diferenciar entre muestras normales y anómalas, con un balance adecuado entre precisión y recall.

### 4.1.4. Aporte Tecnológico

- El producto final de esta investigación serán los modelos de *machinelearning* entrenados para cada objetivo: uno para detectar anomalías en las muestras de aceite , el segundo para predecir los eventos de fallas. Toda la metodología realizada se encuentra en archivos de tipo *ipynb* que serán exportados en formato *joblib* en *python*. Luego el programa de reportes de la operación minera cargara el modelo desde el archivo y lo utilizara para hacer predicciones. El codigo del proceso de creación de los modelo se encuentra en los Anexos de esta Tesis.

# Conclusiones

Se recopilaron el 100 % de los datos a partir de 2014, provenientes del sistema proporcionado por el proveedor durante la implementación del laboratorio de análisis de aceite en dicho año. Aunque dicho sistema ya no cuenta con soporte, la base de datos asociada continúa en uso. (Ver Tabla 3.1).

Se recopilaron todas las fechas fallas de los equipos y se seleccionaron las fechas (Ver Tabla 3.2) siguiendo la teoría de modos de falla de motores diésel (Ver Tabla 2.3). En conjunto, se identificaron un total de 131 fallas en los tres equipos desde el año 2014.

Los datos de las muestras de aceite desde 2014 hasta 2023 fueron sometidos a un preprocesamiento , como se ilustra en la Sección 3.0.2.2. En un primer paso, se identificó que el 2 % de los datos contenían caracteres no numéricos, los cuales fueron reemplazados por valores nulos. El total de datos nulos representó el 2.5 % del conjunto, y se procedió a sustituirlos utilizando el método de imputación por KNN. Para la normalización, se aplicó el método de min-max, como se muestra en la Figura 3.1. En cuanto a la detección de valores atípicos se probaron dos tipos de detección de outliers detección por Rango Intercuartilico y K-Means (contamination=0.01), finalmente se determino no eliminar los outliers por que afectan a la distribución de los parámetros de aceite. Sin embargo, al observar que cada parámetro presentaba valores que podrían considerarse atípicos, y considerando la naturaleza de los datos con una gran cantidad de valores extremos y una distribución no normal, se optó por algoritmos alternativos robustos con alta tolerancia a conjuntos con variabilidad elevada. Sección 3.0.2.2.

Se utilizo la Normal ASTM D7720 para utilizar el 6 % sugerido por la norma como heuristica sobre los modelo de machine learning Covarianza Robusta, Isolation Forest, SVM-One Class, PCA-Anomaly Deteccion.

Para hallar el modelo que debe hallar anomalías en parámetros indicadores directos de fallas mecánicas(no aditivas) se entrenaron los modelos modelos Isolation Forest, SVM-One Class y PCA-Anomaly Detection.

Los modelos presentan un rendimiento variable Tabla4.6, influenciados por el desbalance inherente en las clases y la variabilidad en las distribuciones de los parámetros entre diferentes equipos y componentes. A pesar de estar diseñados para identificar desviaciones, el desafío se amplifica cuando se trata de conjuntos de datos con una mayoría de instancias

normales, lo que puede conducir a una alta tasa de falsos positivos, como se evidencia en los bajos valores de precisión para las clases anómalas en todos los modelos.

Dado que Covarianza Robusta ha obtenido las mejores métricas, por ello el modelo Random Forest con las etiquetas por Covarianza Robusta muestra un desempeño excelente en la clase normal. La precisión de 1 significa que todas las predicciones de la clase normal son correctas, y el alto recall de 0.99 indica que casi todas las instancias normales son identificadas correctamente. El F1-score de 0.99 refleja un equilibrio casi perfecto entre precisión y recall. Para la clase anómala, el modelo también muestra un rendimiento robusto. La precisión de 0.89 sugiere que la mayoría de las predicciones de anomalías son correctas, y el recall de 0.93 indica que el modelo es capaz de identificar la mayoría de las instancias anómalas. El F1-score de 0.91 demuestra un buen balance entre precisión y recall en la detección de anomalías. El modelo Random Forest posee cierto grado de explicabilidad al ofrecer la característica que mas influye en la clasificación de anomalías 4.15, otro aporte de la tesis es la aplicación de SHAP para medir la influencia de las características en cada muestra clasificada. Se corrigieron las muestras etiquetadas por los ingenieros de confiabilidad que no tenían un etiqueta de 'Critico' o 'Seguimiento' pese a que tienen una falla asociada con respecto a la fecha.

Se entrenaron dos algoritmos Random Forest y Redes neuronales Artificales para clasificar en las 3 clases correspondientes.

El modelo 2 con Random Forest es capaz de clasificar la condición de las muestras en tres clases: 'Critico','Seguimiento', 'Normal'. En la tabla 4.10 demuestra la alta precisión que alcanza el modelo para cada clase. La curva de aprendizaje Figura 4.19 muestra cómo cambian las puntuaciones de entrenamiento y validación a medida que se incrementa el número de ejemplos de entrenamiento. La puntuación de entrenamiento comparada con la menor puntuación de validación en los tamaños de entrenamiento más pequeños sugiere que el modelo estaba inicialmente sobreajustado a los datos de entrenamiento.La estabilización de la puntuación de validación cruzada en los tamaños de entrenamiento más grandes indica que el modelo está alcanzando un punto donde añadir más datos de entrenamiento no proporciona mejoras significativas en la precisión del modelo. El hecho de que las curvas de entrenamiento y validación estén cerca una de la otra en los tamaños de conjunto de entrenamiento más grandes sugiere que el conjunto de datos disponible es suficiente para el modelo.La arquitectura de la red neuronal empleada para clasificar muestras también obtuvieron métricas destacadas, consta de 4 capas con 20, 16, 5 y 3 neuronas en cada capa, logrando un Accuracy del 73 % Tabla 4.9. Es relevante tener en cuenta que se aplicó el método de sobremuestreo para equilibrar el número de muestras en cada clase, lo que puede llevar al sobreajuste del modelo. Se implementó la técnica de regularización para controlar este sobreajuste, pero el modelo sigue siendo sensible a detectar fallas que aún no han ocurrido. Finalmente también se utiliza SHAP para explicar la influencia de características.

Finalmente para robustecer la metodología de detección de anomalías se utiliza 'Voto Mayoritario' para integrar ambos modelos, lográndose una precisión altísima de 84 % para el conjunto de prueba con datos de 2024. Lo más importante es resaltar que el modelo identificó

todos los casos verdaderos de .ªnomaly", sin pasar por alto ninguno.

# Relevancia del Trabajo

Los modelos predictivos basados en técnicas de Machine Learning han sido implementados exitosamente en la unidad minera, y su impacto ha sido significativo en varios aspectos clave. Entre los principales aportes de estos modelos destacan los siguientes:

- Reducción en el tiempo de análisis de muestras: Uno de los efectos más notables ha sido la disminución del tiempo necesario para el análisis y la revisión de muestras. Antes de la implementación de los modelos, los expertos del área revisaban un promedio de 60 muestras diarias, dedicando aproximadamente 15 minutos por muestra. Sin embargo, con la integración de los modelos, el número de muestras revisadas se ha reducido a 24 por día, con un tiempo promedio de análisis de solo 5 minutos por muestra. Esto ha resultado en una reducción del 48 % en el tiempo total dedicado a esta tarea, permitiendo a los expertos concentrarse en actividades de mayor valor agregado y mejorando la eficiencia operativa del proceso de revisión.
- Impulso a nuevos proyectos de Machine Learning: El éxito de estos modelos no solo ha optimizado procesos actuales, sino que también ha generado un efecto multiplicador dentro de la unidad minera. Gracias a los resultados positivos obtenidos y al buen desempeño demostrado, se han anunciado y propuesto nuevos proyectos de Machine Learning. Estas iniciativas buscan expandir el uso de tecnologías avanzadas en otras áreas de la operación minera, fomentando una mayor innovación y eficiencia en la toma de decisiones basada en datos.

# Recomendaciones

El análisis tribológico de aceites en maquinaria pesada, como las palas hidráulicas, depende críticamente de la calidad y diversidad del conjunto de datos utilizado para entrenar modelos de Machine Learning. En la actualidad, los modelos pueden tener limitaciones en términos de robustez y capacidad de generalización debido a la limitada variedad en el conjunto de datos inicial. Se recomienda continuar el afinamiento de los modelos periódicamente

- Expansión Cuantitativa de Datos: A medida del paso del tiempo se recomienda incrementar el volumen de datos recolectados para incluir un mayor número de muestras de aceite. Esto permitirá a los modelos de Machine Learning operar sobre un conjunto de datos más representativo, mejorando la precisión y reduciendo el riesgo de sobreajuste. Se considera la implementación de técnicas de recolección de datos automatizada para aumentar la frecuencia y consistencia de las muestras.
- Diversificación de Fuentes de Datos: Incorporar muestras de una gama más amplia de equipos y de diferentes fabricantes. Esto ayudará a los modelos a capturar variaciones específicas del equipo y a generalizar mejor a través de diferentes tipos de maquinaria. Por ello, incluir datos de equipos en diversas etapas de su ciclo de vida, desde nuevos hasta cerca del final de su vida útil, para entender mejor el espectro completo de condiciones de aceite.
- Integración de Datos en Tiempo Real: Se sugiere la incorporación de datos en tiempo real para un monitoreo continuo, lo que puede mejorar la precisión en la predicción de la vida útil y permitir intervenciones más oportunas.

# Bibliografía


Atma Ram Sahu, S. K. P. (2020). Fault prediction of drag system using artificial neural network for prevention of dragline failure. *sciencedirect*. https://doi.org/https://doi.org/10.1016/j.engfailanal.2020.104542

Bhushan, B. (2013). Introduction to Tribology.

Bishop, C. (2006). Pattern Recognition and Machine Learning.

Breiman, L. (2001). Random forests.

Cabezas Luben, S. R., IZbicki Rafael. (2023). *Agrupación jerárquica: Vizualización, importancia de las carácteristicas, selección de modelos.* Universidad de Sao Paulo.

Caroline, E. (s.f.). *What Is Condition-Based Maintenance?* Consultado el 28 de marzo de 2023, desde https://www.getmaintainx.com/learning-center/condition-based-maintenance/#:~:text=Condition%5C%2Dbased%5C%20maintenance%20%5C%20eliminates%5C%20downtime,never%5C%20overspend%5C%20on%5C%20reactive%5C%20maintenance.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). Semi-supervised learning.

Conshohocken, W. (2011). Standard Guide for Statistically Evaluating Measurand Alarm Limits when Using Oil Analysis to Monitor Equipment and Oil for Fitness and Contamination. *ASTM International*. https://doi.org/DOI:10.1520/D7720-11

Cutler, A., Cutler, D. R., & Stevens, J. R. (2007). Random forests. Ensemble machine learning: methods and applications.

D7720, A. S. (2011). *Standard Guide for Statistically Evaluating Measurand Alarm Limits when Using Oil Analysis to Monitor Equipment and Oil for Fitness and Contamination*. ASTM International.

de Carvalho Michalski, M. A., & de Souza, G. F. M. (2022). Comparing PCA-based fault detection methods for dynamic processes with correlated and Non-Gaussian variables. *Construction and Building Materials*. https://doi.org/https://doi.org/10.1016/j.eswa.2022.117989

E., R. (1995). *Friction and Wear of Materials*. International Journal of Advanced Science; Technology.

ElShawi, R., Sherif, Y., Al-Mallah, M., & Sakr, S. (2018). Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. https://doi.org/https://doi.org/10.1111/coin.12410

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.

Grebenişan1, G., Salem, N., Bogdan, S., & Negrău1, D. C. (2019). Oil condition monitoring, an AI application study using the Classification Learner Technics. *IOPscience*. https://doi.org/https://doi.org/10.1088/1757-899X/1169/1/012012

Hernández Sampieri, R., Fernandez Collado, C., & Baptista Lucio, P. (2014). *Metodología de la Investigación*. Mc Graw Hill.

Ibrahim, M., Alsheikh, A., Awaysheh, F. M., & Mohamma, D. (2016). Machine Learning Schemes for Anomaly Detection in Solar Power Plants. *Department of Mechatronics Engineering, German Jordanian University*. https://doi.org/https://doi.org/10.3390/en15031082

Karaca, Y. (2016). Case Study on Artificial Neural Networks and Applications. *sciencedirect*. https://doi.org/doi:10.12988/ams.2016.65174

Keartland, S., & van Zyl, T. (2023). Automating predictive maintenance usiong oil analysis and machine learning. *Lubricants*. https://doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9041003

Keartland, S., & Zyl, T. L. V. (2020). Automating predictive maintenance using oil analysis and machine learning. *IEEE Xplore*. https://doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9041003

Kimera, D., & Nangolo, F. N. (2022). Improving ship yard ballast pumps' operations: A PCA approach to predictive maintenance. *Construction and Building Materials*. https://doi.org/https://doi.org/10.1016/j.martra.2020.100003

Mortier, R. M., Fox, M. F., & Orszulik, S. T. (2020). Condition-Based Maintenance—An Extensive Literature Review.

Murphy, K. (2012). Machine Learning: A Probabilistic Perspective.

Nicole, L. (s.f.). *Redes neuronales recurrentes*. Consultado el 28 de marzo de 2023, desde https://kwfoundation.org/blog/2021/07/13/redes-neuronales-recurrentes/

OZAKI, T. J. (s.f.). *What kind of decision boundaries does Deep Learning (Deep Belief Net) draw?* Consultado el 28 de marzo de 2023, desde https://tjo-en.hatenablog.com/entry/2015/02/15/194003

Piech, C. (2013). *K Means*. https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Popov, K., Bold, R. D., Chai, H.-K., Forde, M., Ho, C., Hyslip, J., Kashani, H., & P. Long e, S. H. f. (2022). Big-data driven assessment of railway track and maintenance efficiency using Artificial Neural Networks. *Construction and Building Materials*. https://doi.org/https://doi.org/10.3390/en15031082

Quatrini, E., Costantino, F., Di Gravio, G., & Patriarca, R. (2020). Condition-Based Maintenance—An Extensive Literature Review.

Rahimi, M., Pourramezan, M.-R., & Rohani, A. (2022). Modeling and classifying the in-operando effects of wear and metal contaminations of lubricating oil on diesel engine: A machine learning approach. *ScienceDirect*. https://doi.org/https://doi.org/10.1016/j.eswa.2022.117494

Reveco Díaz, M. I. (2019). Análisis predictivo de activos mineros para obtención de Intervalo de falla mediante algoritmos de machine learning.

Rodrigues, J., Cost, J., Inês Farinha, Mendes, M., & Margalho, L. (2020). Predicting motor oil condition using artificial neural net- works and principal component analysis. *Eksploatacja i Niezawodnosc - Maintenance and Reliability*. https://doi.org/10.17531/ein.2020.3.6

Rousseeuw, P. J., & Hubert, M. (2018). Anomaly detection by robust statistics Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

T., L. F., Ting, K. M., & Zhou, Z. (2008). Eighth IEEE International Conference on Data Mining.

Wakiru, J. M., Pintelon, L., Muchiri, P. N., & Chemweno, P. K. (2019). A review on lubricant condition monitoring information analysis for maintenance decision support. *Mechanical Systems and Signal Processing*. https://doi.org/https://doi.org/10.1016/j.ymssp.2018.08.039

Zulfauzi, I. A., Dahlan, N. Y., Sintuya, H., & Setthapun, W. (2023). Anomaly detection using K-Means and long-short term memory for predictive maintenance of large-scale solar (LSS) photovoltaic plant. *Energy Reports*, *9*, 154-158. https://doi.org/https://doi.org/10.1016/j.egyr.2023.09.159

# Anexos

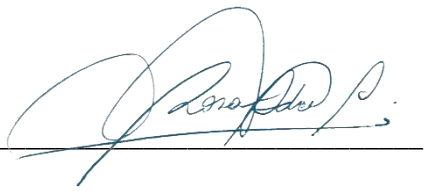
AUTORIZACIÓN DE USO DE DATOS

Yo, Rosa Adelaida Silva Sosa, ingeniera en Minera las Bambas, certifico que he otorgado a Inés Katia Huamán Lima identificada con DNI: 71457734, los datos de aceites de motores recopilados desde el año 2014.

Declaro que estos datos son verídicos, precisos y representan fielmente los análisis realizados en el contexto de la investigación para optar por el título de Ingeniero Informático y de Sistemas en la Universidad Nacional San Antonio Abad de Cusco. Estos datos fueron proporcionados a Inés Katia Huaman Lima con el fin de contribuir a su investigación de tesis titulada “OPTIMIZACIÓN DEL MANTENIMIENTO PREVENTIVO DE PALAS HIDRAULICAS CAT 6060 FS APLICANDO TECNICAS DE MACHINE LEARNING AL ANALISIS TRIBOLOGICO DE MOTORES”. Las condiciones de entrega de esta información son: No revelar la unidad minera dueña de los datos y ofuscación de  códigos de máquinas y modelos.

Dicho otorgamiento de datos se realizó en un marco de confianza mutua y con el propósito de apoyar su trabajo investigativo, asegurando que la información recopilada será utilizada estrictamente para los fines académicos mencionados.

Cusco, 21 de octubre de 2024.



Ing. Rosa A. Silva Sosa

Ing. Innovación y Desarrollo

Área de Ingeniería de Confiabilidad

Minera Las Bambas

```
colunmas_de_interes=[ 'Plomoppm','Aluminioppm','Estañoppm','Cobreppm']
```

```
!pip install pyod
```

```
!pip install pywaffle
```

```
pip install pandas openpyxl
```

```
pip install shap
```

```
!pip install pivottablejs
```

```
pip install joblib
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score,average_precision_score,f1_score,auc
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.metrics import roc_auc_score,roc_curve
from sklearn.model_selection import StratifiedKFold
import pandas as pd
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# Setting the aesthetics for plots
sns.set(style="whitegrid")
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from sklearn.utils import class_weight
from warnings import filterwarnings
import seaborn as sns
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import (confusion_matrix, classification_report,
                             roc_auc_score, precision_recall_curve,
                             average_precision_score, f1_score)
import numpy as np
from scipy import stats
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA
import pickle
from sklearn.cluster import KMeans
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, silhouette_score
from sklearn.preprocessing import LabelEncoder
from sklearn.covariance import EllipticEnvelope
from scipy.spatial.distance import mahalanobis
import plotly.express as px
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_samples, silhouette_score
import joblib

# Crear una instancia de LabelEncoder

filterwarnings('ignore')
```

```
file_path = '/content/DataSinImputar.xlsx'
df_1 = pd.read_excel(file_path)
```

```
df=df_1.copy()
```

```python
# Convertir todos los valores en la columna 'CodigoISO' a cadenas de texto
df['CodigoISO'] = df['CodigoISO'].astype(str)

# Crear una máscara para los valores que tienen el formato adecuado
mask = df['CodigoISO'].str.count('/') == 2

# Inicializar las columnas con NaN
df[['CodigoISO1', 'CodigoISO2', 'CodigoISO3']] = pd.NA

# Separar los valores que cumplen con el formato en tres columnas nuevas
# Para evitar el error, vamos a trabajar solo con los valores que cumplen el formato
df_valid = df.loc[mask, 'CodigoISO'].str.split('/', expand=True)

# Asignar los valores separados a las nuevas columnas en el DataFrame original
df.loc[mask, ['CodigoISO1', 'CodigoISO2', 'CodigoISO3']] = df_valid.values

# Verificar el resultado
print(df[['CodigoISO', 'CodigoISO1', 'CodigoISO2', 'CodigoISO3']].head(10))
```

```python
values_to_replace = [
    'H2O', 'AGUA', '  AGUA ', ' AGUA ', '  AGUA   ', '   AGUA ', '  AGUA',
    'H2O ', '    AGUA ', ' AGUA', '    AGUA', 'H20', 'TRAZAS', '  CORTA ',
    ' CORTA ', '  AGUA  ', 'CORTA', ' CORTA', '   6.800R ', '  11.200R ',
    '  65.92B'
]

# Reemplazar los valores especificados por 700
df['Viscosidada100_CcSt'] = df['Viscosidada100_CcSt'].replace(values_to_replace, 700)
```

```python
columnas_de_interes=['i_mueHoraProducto', 'i_mueHoraEquipo', 'i_mueHoraParte', 'Silicioppm',
       'Aluminioppm', 'Boroppm', 'Sodioppm', 'Potasioppm', 'Vanadioppm',
       'Barioppm', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
       'Antimonioppm', 'Niquelppm', 'Cadmioppm', 'Cromoppm', 'Platappm',
       'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'Agua_', 'TBN_mgKOH_g_',
       'ParticulasFerrosas_PQ_', 'GLYCOL_', 'Turbidez_NTU_', 'Diesel_',
       'Molidatos_0_1200_ppm', 'Molibdenoppm', 'Calcioppm', 'CodigoISO_1',
                'CodigoISO_2', 'CodigoISO_3',
       'Sulfatos_0_3000_ppm', 'Fosforoppm', 'Viscosidada40_CcSt',
       'Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH']
```

```python
df.replace('NeuN', np.nan, inplace=True)
df[columnas_de_interes] = df[columnas_de_interes].apply(pd.to_numeric, errors='coerce')
```

```python
df['Viscosidada40_CcSt'] = df['Viscosidada40_CcSt'].replace(values_to_replace, 700)
```

```python
#@title funciones de graficas
import matplotlib.pyplot as plt
import seaborn as sns
import missingno
def report_missings(data, opt = 0):
  """
  data = dataframe
  opt = opcion de graficar {0: todo, >0 solo faltantes}
        default= 0
  """
  data_rep = round(data.isna().sum().sort_values(ascending=False)/len(data)*100, 3)
  data_rep2 = data.isna().sum().sort_values(ascending=False)
  df = pd.concat([data_rep2, data_rep], axis=1).reset_index()
  df.columns = ["Variable", "Cant. Nulos", "% Nulos"]
  df["Cant. No Nulos"] = len(data) - df["Cant. Nulos"]
  df = df.reindex(columns=["Variable","Cant. No Nulos","Cant. Nulos","% Nulos"])

  print("*"*100)
  print("*"*20, "Reporte General", "*"*20)
  print("*"*100)
  print(df)

  if opt:
    data_rep = data_rep[data_rep > 0] # opcional
  miss = data_rep.to_frame()
  miss.columns = ['Cantidad (%)']
  miss.index.names = ['Variable']
  miss['Variable'] = miss.index

  fig = plt.figure(figsize=(20,29))
  #plot the missing value count
  #plt.figure(figsize=(10,6))
  ax1 = fig.add_subplot(3,2,1)
  print("*"*70, "Graficas de datos faltantes", "*"*70)
  print("*"*180)
  missingno.bar(data, figsize=(10,5), fontsize=12, ax=ax1, color="dodgerblue");
  plt.title("Conteo de muestras para cada variable (porcentaje y cantidad)")

  ax3 = fig.add_subplot(3,2,3)
  sns.heatmap(data.isna().transpose(),
              cmap="YlGnBu",
              cbar_kws={'label': 'Valores perdidos'})
  plt.title("Distribución de valores perdidos")
  plt.tight_layout()
```

```
ax2 = fig.add_subplot(3,2,2)
#sns.set(style="whitegrid", color_codes=True)
sns.barplot(x = 'Variable', y = 'Cantidad (%)', data=miss, ax=ax2)
plt.title("Porcentaje de datos faltantes por variable")
plt.xticks(rotation = 90)
#plt.savefig("missing1.png", dpi=100)



ax4 = fig.add_subplot(3,2,4)
missingno.heatmap(data, cmap="RdYlGn", figsize=(10,5), fontsize=12, ax=ax4)
plt.title("Correlación de nulidad entre variables")
ax5 = fig.add_subplot(3,2,5)
missingno.dendrogram(data, figsize=(10,5), fontsize=12,ax=ax5)
plt.title("Dendograma basado en la correlación de valores faltantes")
plt.tight_layout()
```

```
df.isna().sum()
```

```
def comparar_distribuciones_continuas(df_missing_val, df_missing_val2):
    df_missing_val.dropna(inplace = True)
    df_missing_val2.dropna(inplace = True)
    # Define the two dataframes
    df1 = df_missing_val.copy()
    df2 = df_missing_val2.copy()

    # Define the numerical columns
    numerical_columns = df1.select_dtypes(include=np.number).columns

    # Loop through the numerical columns and compare the distributions
    for col in numerical_columns:
        # Perform a Kolmogorov-Smirnov test
        statistic, pvalue = stats.ks_2samp(df1[col].dropna(), df2[col].dropna())

        # Print the results
        print(f"Variable: {col}")
        print(f"Statistic: {statistic:.4f}")
        print(f"p-value: {pvalue:.4f}")

        # Interpret the results
        if pvalue < 0.05:
            print(f"The distributions of {col} are significantly different.")
        else:
            print(f"The distributions of {col} are are not significantly different.")

        # Print a new line
```

```
df.info()
```

```
columnas_de_interes=['Sodioppm', 'Silicioppm', 'Titanioppm', 'Platappm',
       'Niquelppm', 'Aluminioppm', 'Plomoppm', 'Estañoppm', 'Cobreppm',
       'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'SulfataciónABS_cm', 'NitraciónABS_cm', 'OxidaciónABS_cm', 'Agua_',
       'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_', 'Boroppm',
       'Turbidez_NTU_', 'Potasioppm', 'Molidatos_0_1200_ppm', 'Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3',
       'Sulfatos_0_3000_ppm', 'Fosforoppm', 'Viscosidada40_CcSt',
       'Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
       'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte']
```

```
len(columnas_de_interes)
```

```
report_missings(df[columnas_de_interes])
```

```
varNumericas = df.select_dtypes(include=np.number)
varCategoricas = df.select_dtypes(exclude=np.number)
```

```
varNumericas=columnas_de_interes
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Función para análisis de variables numéricas
def analisisNumericas(df, variable):
    print(" " * 20, "Histograma", " " * 20)

    # Configuración del histograma
    plt.figure(figsize=(8, 4))
    plt.hist(df[variable], bins=25, edgecolor='black', alpha=0.7)
    plt.title(f'{variable} \n', fontdict={'fontsize': 16})
    plt.xlabel('Valor')
    plt.ylabel('Frecuencia')
    plt.yscale('log')  # Escala logarítmica en el eje y para mejorar la visualización
    plt.show()

    print("\n")
    print(" " * 20, "Boxplot", " " * 20)

    # Configuración del boxplot
    plt.figure(figsize=(8, 4))
    plt.boxplot(df[variable].dropna(), vert=False, patch_artist=True, notch=True)
    plt.title(f'{variable} \n', fontdict={'fontsize': 16})
    plt.xlabel('Valor')
    plt.show()
for numerica in varNumericas:
    print("#" * 20, numerica, "#" * 20)
    analisisNumericas(df, numerica)
    print("\n\n")
```

```python
df[df.duplicated()]
```

```python
varNumericas = df.select_dtypes(include=[np.number]).columns.tolist()
print(varNumericas)
df[varNumericas].describe()
```

```python
df.columns
```

```python
df['idcomponente']=df['equipo']+df['componente']
```

```python
sns.histplot(data = df, x="i_mueCodigo", hue="idcomponente", multiple="stack", stat="percent")
plt.show()
```

```python
plt.figure(figsize=(16,6))
mask = np.triu(np.ones_like(df[varNumericas].corr(), dtype = bool))
sns.heatmap(round(df[varNumericas].corr(), 2), mask = mask, cmap = "BrBG", annot = True, vmin = -1, vmax =1 )
plt.show()
```

```python
columnas_de_interes_corr=['Silicioppm', 'Aluminioppm', 'Boroppm', 'Sodioppm', 'Potasioppm',
 'Vanadioppm', 'Barioppm', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm', 'Antimonioppm', 'Niquelppm',
 'Cadmioppm', 'Cromoppm', 'Platappm','HollinABS_01mm', 'SulfataciónABS_cm', 'NitraciónABS_cm', 'OxidaciónABS_cm']
```

```python
plt.figure(figsize=(16,6))
mask = np.triu(np.ones_like(df[columnas_de_interes_corr].corr(), dtype = bool))
sns.heatmap(round(df[columnas_de_interes_corr].corr(), 2), mask = mask, cmap = "BrBG", annot = True, vmin = -1, vmax =1 )
plt.show()
```

```python
df_missing_val=df.copy()

# Select a random percentage of rows with missing values
df_missing_val2 = df_missing_val.copy()
df_missing_val_subset = df_missing_val[df_missing_val2.isna().any(axis=1)].sample(frac=0.00)

# Get the indices of the selected rows
indices = df_missing_val_subset.index

# Drop the selected rows from the original dataframe
df_missing_val2.drop(indices, inplace=True)
df_missing_val2
```

```python
num_filas_con_nulos = df.isnull().any(axis=1).sum()
num_filas_con_nulos
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```python
X=df['NitraciónABS_cm']
Y=df['OxidaciónABS_cm']
data_structured = np.column_stack((X, Y))

# Imputación por la media
imputer_mean = SimpleImputer(strategy='mean')
data_mean = imputer_mean.fit_transform(df)

# Imputación por KNN
imputer_knn = KNNImputer(n_neighbors=5)
data_knn = imputer_knn.fit_transform(df)

# Visualización
plt.figure(figsize=(15, 7))
plt.subplot(1, 3, 1)
plt.scatter(df['X'], df['Y'], alpha=0.6, color='red', label='Original Data')
plt.legend()
plt.title('Original Data with NaN')

plt.subplot(1, 3, 2)
plt.scatter(data_mean[:, 0], data_mean[:, 1], alpha=0.6, color='blue', label='Mean Imputation')
plt.legend()
plt.title('Mean Imputation')

plt.subplot(1, 3, 3)
plt.scatter(data_knn[:, 0], data_knn[:, 1], alpha=0.6, color='green', label='KNN Imputation')
plt.legend()
plt.title('KNN Imputation')
plt.show()

# Evaluación del impacto en un modelo de regresión lineal
model = LinearRegression()
X_orig, Y_orig =df.dropna().values.T
model.fit(X_orig.reshape(-1, 1), Y_orig)
mse_original = mean_squared_error(Y_orig, model.predict(X_orig.reshape(-1, 1)))

X_mean, Y_mean = data_mean.T
model.fit(X_mean.reshape(-1, 1), Y_mean)
mse_mean = mean_squared_error(Y_mean, model.predict(X_mean.reshape(-1, 1)))

X_knn, Y_knn = data_knn.T
model.fit(X_knn.reshape(-1, 1), Y_knn)
mse_knn = mean_squared_error(Y_knn, model.predict(X_knn.reshape(-1, 1)))
```

```python
print("MSE with Original Data:", mse_original)
print("MSE with Mean Imputation:", mse_mean)
print("MSE with KNN Imputation:", mse_knn)
```

```python
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Copy the data
df_ii_missing = df[varNumericas.columns].copy(deep=True)

# Init
ii_imp = IterativeImputer(
    estimator=ExtraTreesRegressor(), max_iter=10, random_state=123, verbose=3
)

# Tranform
df_ii_missing.loc[:, :] = ii_imp.fit_transform(df_ii_missing)
```

```python
from sklearn.impute import KNNImputer
def show_distr(df):
  print("missing values : \n{}".format(df.isna().sum()))
  boxplot = df.boxplot(column=columnas_de_interes)
  hist = df[columnas_de_interes].hist(bins=20)
  print("\n -------------------------")
imputer_knn = KNNImputer(n_neighbors=10, weights="uniform")
df_imp = imputer_knn.fit_transform(df[columnas_de_interes])
df_imp = pd.DataFrame(df_imp, columns = columnas_de_interes)
show_distr(df_imp)
```

```python
show_distr(df[columnas_de_interes])
```

```python
df.to_excel('/content/DatafinalCondicion_06_03_2024_2.xlsx')
```

```python
columnas_de_interes=['i_mueHoraProducto', 'i_mueHoraEquipo', 'i_mueHoraParte', 'Silicioppm',
       'Aluminioppm', 'Boroppm', 'Sodioppm', 'Potasioppm', 'Vanadioppm',
       'Barioppm', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
       'Antimonioppm', 'Niquelppm', 'Cadmioppm', 'Cromoppm', 'Platappm',
       'ParticulasFerrosas_PQ_', 'Titanioppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_', 'Molidatos_0_1200_ppm', 'Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3',
       'Sulfatos_0_3000_ppm', 'Fosforoppm', 'Viscosidada40_CcSt',
       'Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH']
```

```python
file_path = '/content/DatafinalCondicion_06_03_2024_2.xlsx'
df_1 = pd.read_excel(file_path)
```

```python
df=df_1.copy()
```

```python
# Crear una paleta de colores personalizada
palette = {'Critico': 'red', 'Normal': 'green', 'Seguimiento': 'yellow'}

# Verificar que la columna 'condicion_smb_1' tiene las categorías esperadas
print(df['condicion_smb.1'].unique())

# Crear los boxplots
f, axes = plt.subplots(nrows=4, ncols=3, figsize=(25, 15))

sns.boxplot(x="condicion_smb.1", y="Potasioppm", data=df, ax=axes[0, 0], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Molidatos_0_1200_ppm", data=df, ax=axes[0, 1], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Molibdenoppm", data=df, ax=axes[0, 2], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Calcioppm", data=df, ax=axes[1, 0], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Fosforoppm", data=df, ax=axes[2, 1], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Viscosidada40_CcSt", data=df, ax=axes[2, 2], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Turbidez_NTU_", data=df, ax=axes[3, 0], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Boroppm", data=df, ax=axes[3, 1], palette=palette)
sns.boxplot(x="condicion_smb.1", y="Zincppm", data=df, ax=axes[3, 2], palette=palette)


plt.tight_layout()
plt.show()
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Filtrar las columnas necesarias del DataFrame
df_frequency = df[['Potasioppm', 'Molidatos_0_1200_ppm', 'Molibdenoppm', 'Calcioppm',
                   'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3', 'Fosforoppm',
                   'Viscosidada40_CcSt', 'Turbidez_NTU_','Boroppm','Zincppm',
                   'HollinABS_01mm','SulfataciónABS_cm', 'NitraciónABS_cm', 'OxidaciónABS_cm', 'Agua_',
                   'condicion_smb.1']]

# Crear una paleta de colores personalizada
palette = {'Critico': 'red', 'Normal': 'green', 'Seguimiento': 'yellow'}

# Crear la figura con 3 columnas y 4 filas
fig, ax = plt.subplots(ncols=3, nrows=4, figsize=(20, 15))

# Crear los gráficos de dispersión
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Molibdenoppm", hue="condicion_smb.1", ax=ax[0, 0], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="CodigoISO_1", hue="condicion_smb.1", ax=ax[0, 1], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="CodigoISO_2", hue="condicion_smb.1", ax=ax[0, 2], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="CodigoISO_3", hue="condicion_smb.1", ax=ax[1, 0], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Fosforoppm", hue="condicion_smb.1", ax=ax[1, 1], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Viscosidada40_CcSt", hue="condicion_smb.1", ax=ax[1, 2], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Agua_", hue="condicion_smb.1", ax=ax[2, 0], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Turbidez_NTU_", hue="condicion_smb.1", ax=ax[2, 1], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Boroppm", hue="condicion_smb.1", ax=ax[2, 2], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="Zincppm", hue="condicion_smb.1", ax=ax[3, 0], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="HollinABS_01mm", hue="condicion_smb.1", ax=ax[3, 1], palette=palette)
sns.scatterplot(data=df_frequency, y="Potasioppm", x="SulfataciónABS_cm", hue="condicion_smb.1", ax=ax[3, 2], palette=palette)
# Ajustar el diseño
plt.tight_layout()
plt.show()
```

```
from pyod.models.knn import KNN
import joblib
```

```
file_path = '/content/DatafinalCondicion_06_03_2024_2.xlsx'
df_1 = pd.read_excel(file_path)
```

```
df=df_1.copy()
```

```
# Imputar las columnas seleccionadas con la media
df[columnas_de_interes] = df[columnas_de_interes].apply(lambda col: col.fillna(col.mean()), axis=0)
```

```
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
```

```
clf = KNN(contamination=0.01)
clf.fit(df[columnas_de_interes])
y_pred = clf.predict(df[columnas_de_interes])
#0: inliers
#1: outliers
df[y_pred == 0]
```

```
joblib.dump(clf, '/content/modelo_knn_outliers.pkl')

# Cargar el modelo desde el archivo
clf_cargado = joblib.load('/content/modelo_knn_outliers.pkl')
```

```
comparar_distribuciones_continuas(df[columnas_de_interes], df[df['y_out_knn']==0][columnas_de_interes])
```

```
df.shape
```

```
df[df['y_out_knn']==0][columnas_de_interes]
```

```
df[df['y_out_knn']==0].to_excel('/content/DatafinalCondicion_sinoutliers.xlsx')
```

```
# Contar los valores de cada categoría
conteo_anomalias = df['y_out_knn'].value_counts()

# Calcular el porcentaje de cada categoría
porcentaje_anomalias = conteo_anomalias / len(df_Anomalia_IQR) * 100

# Mostrar los resultados
print("Conteo de anómalos y no anómalos:")
print(conteo_anomalias)
print("\nPorcentaje de anómalos y no anómalos:")
print(porcentaje_anomalias)
#El 8.77% de los datos son clasificados como anomalías
```

```
def etiquetar_anomalias_grupo(group, columnas_de_interes):
    # Inicializar la columna de Anomalia_IQR como 0 (normal)
    group['Anomalia_IQR'] = 0

    for columna in columnas_de_interes:
        Q1 = group[columna].quantile(0.25)
        Q3 = group[columna].quantile(0.75)
        IQR = Q3 - Q1
        limite_inferior = Q1 - 1.5 * IQR
        limite_superior = Q3 + 1.5 * IQR

        # Detectar anomalías
        mask = (group[columna] < limite_inferior) | (group[columna] > limite_superior)

        # Si cualquier valor en la columna es anómalo, marcar toda la fila como anómala
        group.loc[mask, 'Anomalia_IQR'] = 1

    return group
```

```
df_Anomalia_IQR = df.groupby(['equipo', 'componente']).apply(etiquetar_anomalias_grupo, columnas_de_interes=columnas_de_interes)
```

```
df_Anomalia_IQR.reset_index(drop=True, inplace=True)
```

```
df.reset_index(drop=True, inplace=True)
```

```
df['Anomalia_IQR']=df_Anomalia_IQR['Anomalia_IQR']
```

```
# Contar los valores de cada categoría
conteo_anomalias = df_Anomalia_IQR['Anomalia_IQR'].value_counts()

# Calcular el porcentaje de cada categoría
porcentaje_anomalias = conteo_anomalias / len(df_Anomalia_IQR) * 100

# Mostrar los resultados
print("Conteo de anómalos y no anómalos:")
print(conteo_anomalias)
print("\nPorcentaje de anómalos y no anómalos:")
print(porcentaje_anomalias)
#El 8.77% de los datos son clasificados como anomalias
```

```
df['Anomalia_IQR'].unique()
```

```
comparar_distribuciones_continuas(df[columnas_de_interes], df_Anomalia_IQR[df_Anomalia_IQR['Anomalia_IQR']==0][columnas_de_interes])
```

```
comparar_distribuciones_continuas(df[columnas_de_interes], df[df['y_out_knn']==0][columnas_de_interes])
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Función para análisis de variables numéricas
def analisisNumericas(df, variable):
    print(" " * 20, "Histograma", " " * 20)

    # Configuración del histograma
    plt.figure(figsize=(8, 4))
    plt.hist(df[variable], bins=25, edgecolor='black', alpha=0.7)
    plt.title(f'{variable} \n', fontdict={'fontsize': 16})
    plt.xlabel('Valor')
    plt.ylabel('Frecuencia')
    plt.yscale('log')  # Escala logarítmica en el eje y para mejorar la visualización
    plt.show()

    print("\n")
    print(" " * 20, "Boxplot", " " * 20)

    # Configuración del boxplot
    plt.figure(figsize=(8, 4))
    plt.boxplot(df[variable].dropna(), vert=False, patch_artist=True, notch=True)
    plt.title(f'{variable} \n', fontdict={'fontsize': 16})
    plt.xlabel('Valor')
    plt.show()
for numerica in varNumericas:
    print("#" * 20, numerica, "#" * 20)
    analisisNumericas(df, numerica)
    print("\n\n")
```

```
file_path = '/content/DatafinalCondicion_06_03_2024.xlsx'
df_1 = pd.read_excel(file_path)
```

```
df=df_1.copy()
```

```
df['idcomponente']=df['equipo']+df['componente']
```

```
columnas_de_interes=['Silicioppm', 'Aluminioppm', 'Boroppm', 'Sodioppm',
'Potasioppm', 'Vanadioppm', 'Barioppm', 'Fierroppm', 'Cobreppm', 'Plomoppm',
'Estañoppm', 'Antimonioppm', 'Niquelppm', 'Cadmioppm', 'Cromoppm', 'Platappm',
'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm', 'NitraciónABS_cm',
'OxidaciónABS_cm', 'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_',
'GLYCOL_', 'Turbidez_NTU_', 'Diesel_']
```

```
def etiquetar_muestra(row, parametros, limites):
    for param in parametros:
        if row[param] > limites[param]['severo_superior'] or row[param] < limites[param]['severo_inferior']:
            return 1
        elif row[param] > limites[param]['leve_superior'] or row[param] < limites[param]['leve_inferior']:
            return 1
    return 0
```

```
def calcular_limites(df, parametros):
    limites = {}
    for param in parametros:
        media = df[param].mean()
        desv_std = df[param].std()
        limites[param] = {
            'severo_superior': media + 3 * desv_std,
            'severo_inferior': media - 3 * desv_std,
            'leve_superior': media + desv_std,
            'leve_inferior': media - desv_std
        }
    return limites
```

```
# Aplicar cálculo de límites y etiquetado por cada grupo
def etiquetar_grupo(df):
    limites = calcular_limites(df, columnas_de_interes)
    df['etiquetaNormaDesv'] = df.apply(lambda row: etiquetar_muestra(row, columnas_de_interes, limites), axis=1)
    return df

# Aplicando la función a cada grupo definido por 'equipo' y 'componente'
df_etiquetaNormaDesv = df.groupby(['equipo', 'componente'], as_index=False).apply(etiquetar_grupo)
```

```
df_etiquetaNormaDesv.reset_index(drop=True, inplace=True)
df.reset_index(drop=True, inplace=True)
df['etiquetaNormaDesv']=df_etiquetaNormaDesv['etiquetaNormaDesv']
```

```
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
df_comprobar_scaled['idcomponente']=df['idcomponente']

df_encoded = pd.get_dummies(df_comprobar_scaled, columns=['idcomponente'], prefix='idcomponente')
boolean_columns = df_encoded.select_dtypes(include=['bool']).columns
df_encoded[boolean_columns] = df_encoded[boolean_columns].astype(int)
# df_encoded
df_encoded.reset_index(drop=True, inplace=True)

# Aplicar PCA a tus datos con 3 componentes
pca = PCA(n_components=3)
df_pca = pca.fit_transform(df_encoded[columnas_de_interes])

# Convertir a DataFrame y añadir etiquetas
df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2', 'PC3'])
df_pca['etiquetaNormaDesv'] = df['etiquetaNormaDesv'].values

# Definir colores para las clases
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde claro
    1: "#fd0101"  # Rojo
}

# Crear el gráfico interactivo con Plotly
fig = px.scatter_3d(df_pca, x='PC1', y='PC2', z='PC3', color='etiquetaNormaDesv',
                    color_discrete_map=colores_pasteles_claros,
                    title='Visualización de Anomalías en 3D')

# Ajustar la escala del eje
fig.update_layout(scene=dict(
    xaxis=dict(type='log', title='Primer Componente Principal'),
    yaxis=dict(type='log', title='Segundo Componente Principal'),
    zaxis=dict(type='log', title='Tercer Componente Principal')
))

fig.show()
```

```
columnas_de_interes=['Plomoppm','Aluminioppm','Estañoppm','Cobreppm']
```

```
# Aplicar PCA a tus datos
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_comprobar_scaled[columnas_de_interes])

df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

# Añadir la columna de etiquetas al DataFrame PCA
df_pca['etiquetaNormaDesv'] = df['etiquetaNormaDesv'].values

# Definir colores para las clases
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde claro
    1: "#fd0101"  # Rojo
}

# Crear el gráfico
plt.figure(figsize=(10, 7))

# Asegúrate de que las clases estén bien definidas y ordenadas
clases_ordenadas = sorted(df_pca['etiquetaNormaDesv'].unique(), key=lambda x: x == 1)

for clase in clases_ordenadas:
    # Verifica si la clase es 1 o no
    if clase == 1:
        alpha_value = 0.3  # Más transparente para clase 1
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1.0  # Opaco para otras clases
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['etiquetaNormaDesv'] == clase]['PC1'],
        df_pca[df_pca['etiquetaNormaDesv'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),
        alpha=alpha_value,
        zorder=zorder_value
    )

# Añadir etiquetas y leyenda
plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de Anomalías Desviacion Estandar')
plt.legend()
plt.show()
```

```python
# Contar los valores de cada categoría
conteo_anomalias = df_etiquetaNormaDesv['etiquetaNormaDesv'].value_counts()

# Calcular el porcentaje de cada categoría
porcentaje_anomalias = conteo_anomalias / len(df_etiquetaNormaDesv) * 100

# Mostrar los resultados
print("Conteo de anómalos y no anómalos:")
print(conteo_anomalias)
print("\nPorcentaje de anómalos y no anómalos:")
print(porcentaje_anomalias)
#El 8.77% de los datos son clasificados como anomalías
```

```python
df[columnas_de_interes] = df[columnas_de_interes].fillna(df[columnas_de_interes].mean())
```

```python
file_path = '/content/DatafinalCondicion.xlsx'
df_1 = pd.read_excel(file_path)
```

```python
df=df_1.copy()
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
```

```python
# Definir una función para aplicar EllipticEnvelope y calcular la distancia de Mahalanobis
def apply_elliptic_envelope(group, columnas_de_interes):
    if len(group) > 2:
        # Inicializar EllipticEnvelope
        env = EllipticEnvelope(support_fraction=1., contamination=0.08)
        # Ajustar el modelo
        env.fit(group[columnas_de_interes])
        # Predecir los outliers (-1 para outliers, 1 para inliers)
        group['etiquetado_covarianza_robusta'] = env.predict(group[columnas_de_interes])
        # Cambiar -1 a 1 (outliers) y 1 a 0 (inliers)
        group['etiquetado_covarianza_robusta'] = group['etiquetado_covarianza_robusta'].map({-1: 1, 1: 0})

        # Calcular la distancia de Mahalanobis para cada punto con regularización
        mean = env.location_
        cov = env.covariance_ + np.eye(env.covariance_.shape[0]) * 1e-6  # Regularización
        inv_cov = np.linalg.inv(cov)
        group['mahalanobis_distance'] = group[columnas_de_interes].apply(lambda x: mahalanobis(x, mean, inv_cov), axis=1)
    else:
        # Asignar 0 si no hay suficientes puntos para calcular la covarianza (asumiendo normalidad)
        group['etiquetado_covarianza_robusta'] = 0
        group['mahalanobis_distance'] = 0
    return group

# Aplicar la función a cada grupo
df = df.groupby(['equipo', 'componente'], as_index=False).apply(apply_elliptic_envelope, columnas_de_interes)
```

```python
# Calcular el coeficiente de silueta modificado
def modified_silhouette_score(df, distance_col, label_col):
    distances = df[distance_col]
    labels = df[label_col]

    silhouette_vals = silhouette_samples(distances.values.reshape(-1, 1), labels)
    silhouette_score = silhouette_vals.mean()

    return silhouette_score

silhouette_score = modified_silhouette_score(df, 'mahalanobis_distance', 'etiquetado_covarianza_robusta')
print(f"Coeficiente de Silueta Modificado: {silhouette_score}")
```

```python
df.reset_index(drop=True, inplace=True)
```

```python
# Contar los valores de cada categoría
conteo_anomalias = df['etiquetado_covarianza_robusta'].value_counts()

# Calcular el porcentaje de cada categoría
porcentaje_anomalias = conteo_anomalias / len(df) * 100

# Mostrar los resultados
print("Conteo de anómalos y no anómalos:")
print(conteo_anomalias)
print("\nPorcentaje de anómalos y no anómalos:")
print(porcentaje_anomalias)
#El 8.77% de los datos son clasificados como anomalías
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```python
# Aplicar PCA a tus datos
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df[columnas_de_interes])

df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

# Añadir la columna de etiquetas al DataFrame PCA
df_pca['etiquetado_covarianza_robusta'] = df['etiquetado_covarianza_robusta'].values

# Definir colores para las clases
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde claro
    1: "#fd0101"  # Rojo
}

# Crear el gráfico
plt.figure(figsize=(10, 7))

# Asegúrate de que las clases estén bien definidas y ordenadas
clases_ordenadas = sorted(df_pca['etiquetado_covarianza_robusta'].unique(), key=lambda x: x == 1)

for clase in clases_ordenadas:
    # Verifica si la clase es 1 o no
    if clase == 1:
        alpha_value = 0.3  # Más transparente para clase 1
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1.0  # Opaco para otras clases
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['etiquetado_covarianza_robusta'] == clase]['PC1'],
        df_pca[df_pca['etiquetado_covarianza_robusta'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),
        alpha=alpha_value,
        zorder=zorder_value
    )

# Añadir etiquetas y leyenda
plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de Anomalías Covarianza Robusta')
plt.legend()
plt.show()
```

```python
columns_to_plot = ['etiquetaNormaDesv', 'Anomalia_IQR', 'etiquetado_covarianza_robusta']

count_df = pd.DataFrame()

# Contar el número de clases en cada columna y agregar al DataFrame
for column in columns_to_plot:
    count_series = df[column].value_counts().reset_index()
    count_series.columns = ['Class', 'Count']
    count_series['Column'] = column
    count_df = pd.concat([count_df, count_series])

# Asignar nombres a las clases
class_mapping = {0: 'Normal', 1: 'Anomalo'}
count_df['Class'] = count_df['Class'].map(class_mapping)

# Asignar colores específicos a las clases
palette = {'Normal': 'green', 'Anomalo': 'red'}

# Graficar los conteos usando seaborn
plt.figure(figsize=(8, 6))
ax = sns.barplot(data=count_df, x='Column', y='Count', hue='Class', palette=palette)

# Añadir etiquetas de texto a las barras
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

plt.title('Número de clases por columna')
plt.xlabel('Tipos de etiquetado')
plt.ylabel('Número de muestras por clase')
plt.legend(title='Clases')

# Cambiar las etiquetas de las columnas
new_labels = ['Norma Desviación', 'Anomalía IQR', 'Covarianza Robusta']
ax.set_xticklabels(new_labels)

plt.show()
```

```python
file_path = '/content/DatafinalCondicion.xlsx'
df_1 = pd.read_excel(file_path)

df=df_1.copy()
```

```python
df['condicion_smb.1'].unique()
```

```python
df['condicion_smb.1'].value_counts()
```

```python
df['condicion_smb.1'] = df['condicion_smb.1'].fillna('Normal')
```

```
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df['etiquetado_covarianza_robusta']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(3, 3))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
df['idcomponente']=df['equipo']+df['componente']
```

```
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
```

```
scaler = MinMaxScaler()
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']
df_comprobar_scaled['idcomponente']=df['idcomponente']
df_encoded = pd.get_dummies(df_comprobar_scaled[columnas_de_interes], columns=['idcomponente'], prefix='idcomponente')
boolean_columns = df_encoded.select_dtypes(include=['bool']).columns
df_encoded[boolean_columns] = df_encoded[boolean_columns].astype(int)
# df_encoded
df_encoded.reset_index(drop=True, inplace=True)
df_encoded
```

```
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm','Cobreppm', 'Plomoppm', 'Estañoppm', 'Vanadioppm', 'Cadmioppm',
       'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm', 'Barioppm','Antimonioppm', 'Niquelppm', 'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm','NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_', 'idcomponente_SH001MODDE','idcomponente_SH001MODI', 'idcomponente_SH002MODDE',
       'idcomponente_SH002MODI', 'idcomponente_SH003MODDE','idcomponente_SH003MODI', 'idcomponente_SH004MODDE',
       'idcomponente_SH004MODI', 'idcomponente_SH005MODDE','idcomponente_SH005MODI', 'idcomponente_SH006MODDE',
       'idcomponente_SH006MODI', 'idcomponente_SH007MODDE','idcomponente_SH007MODI', 'idcomponente_SH008MODDE',
       'idcomponente_SH008MODI', 'idcomponente_SH009MODDE','idcomponente_SH009MODI', 'idcomponente_SH010MODDE',
       'idcomponente_SH010MODI', 'idcomponente_SH011MODDE','idcomponente_SH011MODI', 'idcomponente_SH012MODDE',
       'idcomponente_SH012MODI', 'idcomponente_SH013MODI']
```

```
# Cargar tus datos
# Supongamos que X representa tus características
X = df_encoded[columnas_de_interes]  # ajusta esto según tus columnas

start_time = time.time()
# Configurar Isolation Forest
iso_forest = IsolationForest(n_estimators=100, contamination=0.06, random_state=42)

# Entrenar el modelo
iso_forest.fit(X)

# Predecir las etiquetas de anomalía (-1 para anomalías, 1 para normales)
labels = iso_forest.predict(X)
end_time=time.time()
training_duration = end_time - start_time
# print(f"El entrenamiento tomó {training_duration} segundos.")
# Añadir etiquetas al dataframe para revisión
df['anomaly_label_isolation'] = labels
```

```
print(training_duration)
```

```
df['anomaly_label_isolation'].unique()
```

```
# Mapeo manual de los valores
df['anomaly_label_isolation'] = df['anomaly_label_isolation'].map({1: 0, -1: 1})
```

```python
# Aplicar PCA a tus datos con 3 componentes
pca = PCA(n_components=3)
df_pca = pca.fit_transform(df_encoded[columnas_de_interes])

# Convertir a DataFrame y añadir etiquetas
df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2', 'PC3'])
df_pca['anomaly_label_isolation'] = df['anomaly_label_isolation'].values

# Definir colores para las clases
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde claro
    1: "#fd0101"  # Rojo
}

# Crear el gráfico interactivo con Plotly
fig = px.scatter_3d(df_pca, x='PC1', y='PC2', z='PC3', color='anomaly_label_isolation',
                    color_discrete_map=colores_pasteles_claros,
                    title='Visualización de Anomalías en 3D')

# Ajustar la escala del eje
fig.update_layout(scene=dict(
    xaxis=dict(type='log', title='Primer Componente Principal'),
    yaxis=dict(type='log', title='Segundo Componente Principal'),
    zaxis=dict(type='log', title='Tercer Componente Principal')
))

fig.show()
```

```python
columnas_de_interes=['Plomoppm','Aluminioppm','Estañoppm','Cobreppm']
```

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA

# Asumiendo que 'df_comprobar_scaled' es tu DataFrame escalado y 'preprocess_todo_df' tiene las etiquetas de clasificación
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_comprobar_scaled[columnas_de_interes])

# Convierte el resultado a un DataFrame para facilitar la manipulación
df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

# Añade la columna 'anomaly_label_isolation' al DataFrame de PCA para poder colorear según la clase
df_pca['anomaly_label_isolation'] = df['anomaly_label_isolation'].values

# Define una paleta de colores pasteles claros
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde pastel claro
    1: "#fd0101"   # Rojo
}

# Ahora grafica los resultados utilizando la paleta de colores pasteles claros
plt.figure(figsize=(7, 7))

# Ordenar las clases para que 'Anomalía' (1) sea graficada primero y por lo tanto en el fondo
clases_ordenadas = [1, 0]

for clase in clases_ordenadas:
    if clase == 1:
        alpha_value = 0.3  # Más transparente para 'Anomalía'
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1    # Opaco para 'Normal'
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['anomaly_label_isolation'] == clase]['PC1'],
        df_pca[df_pca['anomaly_label_isolation'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),  # Usamos 'gray' si la clase no está en la paleta
        alpha=alpha_value,  # Ajuste de la transparencia
        zorder=zorder_value  # Ajuste de zorder para controlar el orden de visualización
    )
plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de detección de anomalías Isolation Forest')
plt.legend()
plt.show()
```

```python
df['etiquetado_covarianza_robusta'].unique()
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df['anomaly_label_isolation']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(3, 3))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df['anomaly_label_isolation']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación Isolation Forest vs Condición:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
print("AUC-ROC:", roc_auc_score(y_true, y_pred))
print("AUC-PR:", average_precision_score(y_true, y_pred))
print("F1-Score:", f1_score(y_true, y_pred))
```

```python
y_true = df['etiquetado_covarianza_robusta']  # Asegúrate de tener esta columna en tu DataFrame

# Obtener scores de anomalía (cuanto más bajo, más anómalo)
scores = iso_forest.decision_function(X)

# Convertir scores a positivos (opcional, depende de cómo desees calcular ROC-AUC)
scores = scores.max() - scores

# Calcular ROC-AUC
roc_auc = roc_auc_score(y_true, df['anomaly_label_isolation'])
print("ROC-AUC Score:", roc_auc)
```

```python
# Guardar el modelo en un archivo
with open('iso_forest.pkl', 'wb') as file:
    pickle.dump(iso_forest, file)
```

```python
# Cargar el modelo
with open('iso_forest.pkl', 'rb') as file:
    loaded_iso_forest = pickle.load(file)
```

```python
file_path = '/content/DatafinalCondicion_06_03_2024_2.xlsx'
df_1 = pd.read_excel(file_path)
df=df_1.copy()
```

```python
import numpy as np
import pandas as pd
from sklearn.svm import OneClassSVM
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import time
```

```python
df['idcomponente']=df['equipo']+df['componente']
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_Cc5t', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
```

```python
scaler = MinMaxScaler()
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_Cc5t', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']
df_comprobar_scaled['idcomponente']=df['idcomponente']
df_encoded = pd.get_dummies(df_comprobar_scaled[columnas_de_interes], columns=['idcomponente'], prefix='idcomponente')
boolean_columns = df_encoded.select_dtypes(include=['bool']).columns
df_encoded[boolean_columns] = df_encoded[boolean_columns].astype(int)
# df_encoded
df_encoded.reset_index(drop=True, inplace=True)
```

```python
start_time = time.time()
# Configuración de One-Class SVM
oc_svm = OneClassSVM(kernel='rbf', gamma='auto', nu=0.06)  # nu es aproximadamente el % de anomalías
oc_svm.fit(df_encoded)
# Predicción
predicciones = oc_svm.predict(df_encoded)
end_time = time.time()
training_duration = end_time - start_time
print(f"El entrenamiento tomó {training_duration} segundos.")
# Asignar etiquetas a los resultados (-1 para anomalías, 1 para normales)
df['anomaly_SVM'] = predicciones
df['anomaly_SVM'] = df['anomaly_SVM'].map({1: 'anomaly_SVM', 0: 'normal'})
```

```python
import pickle
```

```python
# Guardar el modelo en un archivo
with open('oc_svm_model.pkl', 'wb') as file:
    pickle.dump(oc_svm, file)
```

```python
# Cargar el modelo
with open('/content/oc_svm_model (2).pkl', 'rb') as file:
    oc_svm = pickle.load(file)
```

```python
predicciones = oc_svm.predict(df_encoded)
```

```python
df['anomaly_SVM'] = predicciones
df['anomaly_SVM'] = df['anomaly_SVM'].map({1: 'anomaly_SVM', 0: 'normal'})
```

```python
df['anomaly_SVM'].unique()
```

```python
df['anomaly_SVM'] = df['anomaly_SVM'].replace('anomaly_SVM', 1)

# Cambiar NaN por 0
df['anomaly_SVM'] = df['anomaly_SVM'].fillna(0)
```

```python
# Contar los valores de cada categoría
conteo_anomalias = df['anomaly_SVM'].value_counts()

# Calcular el porcentaje de cada categoría
porcentaje_anomalias = conteo_anomalias / len(df) * 100

# Mostrar los resultados
print("Conteo de anómalos y no anómalos:")
print(conteo_anomalias)
print("\nPorcentaje de anómalos y no anómalos:")
print(porcentaje_anomalias)
#El 8.77% de los datos son clasificados como anomalías
```

```python
y_true=df['etiquetado_covarianza_robusta']
y_pred=df['anomaly_SVM']
```

```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
conf_matrix=confusion_matrix(y_true, y_pred)
print("Matriz de Confusión:\n", conf_matrix)
print("\nReporte de Clasificación:\n", classification_report(y_true, y_pred))
print("Precisión:", accuracy_score(y_true, y_pred))
print("AUC-ROC:", roc_auc_score(y_true, y_pred))
print("AUC-PR:", average_precision_score(y_true, y_pred))
print("F1-Score:", f1_score(y_true, y_pred))
```

```python
df['anomaly_SVM'].value_counts()
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df['anomaly_SVM']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación anomaly_SVM:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
df['anomaly_SVM'].unique()
```

```python
# Asumiendo que 'df_comprobar_scaled' es tu DataFrame escalado y 'preprocess_todo_df' tiene las etiquetas de clasificación
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_comprobar_scaled[columnas_de_interes])

# Convierte el resultado a un DataFrame para facilitar la manipulación
df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

# Añade la columna 'Condicion_smb' al DataFrame de PCA para poder colorear según la clase
df_pca['anomaly_SVM'] = df['anomaly_SVM'].values

# Define una paleta de colores pasteles claros
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde pastel claro
    1: "#fd0101"  # Rojo
}

# Ahora grafica los resultados utilizando la paleta de colores pasteles claros
plt.figure(figsize=(7, 7))

# Ordenar las clases para que 'Normal' sea graficada primero y por lo tanto en el fondo
clases_ordenadas = sorted(df_pca['anomaly_SVM'].unique(), key=lambda x: x == 1)

for clase in clases_ordenadas:
    if clase == 1:
        alpha_value = 0.3  # Más transparente para 'Anomalía'
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1    # Opaco para 'Normal'
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['anomaly_SVM'] == clase]['PC1'],
        df_pca[df_pca['anomaly_SVM'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),  # Usamos 'gray' si la clase no está en la paleta
        alpha=alpha_value,  # Ajuste de la transparencia
        zorder=zorder_value  # Ajuste de zorder para controlar el orden de visualización
    )

plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de detección de anomalías SVM-One Class')
plt.legend()
plt.show()
```

```
import joblib
import numpy as np

model_path = '/content/oc_svm_model (2).pkl'

# Paso 3: Usar el modelo para predecir anomalías
predicciones = oc_svm_model.predict(new_data)

# Paso 4: Interpretar las predicciones
# En el caso de One-Class SVM, generalmente:
# -1 indica una anomalía
#  1 indica un punto normal
print(predicciones)
```

```
df=df_1.copy()
```

```
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'EstaÑoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'EstaÑoppm.1',
       'SulfataciónABS_cm', 'ViscosidadaA100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
scaler = MinMaxScaler()
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'EstaÑoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'EstaÑoppm.1',
       'SulfataciónABS_cm', 'ViscosidadaA100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']
df_comprobar_scaled['idcomponente']=df['idcomponente']
df_encoded = pd.get_dummies(df_comprobar_scaled[columnas_de_interes], columns=['idcomponente'], prefix='idcomponente')
boolean_columns = df_encoded.select_dtypes(include=['bool']).columns
df_encoded[boolean_columns] = df_encoded[boolean_columns].astype(int)
# df_encoded
df_encoded.reset_index(drop=True, inplace=True)
```

```
train_data, test_data = train_test_split(df_encoded, test_size=0.20, random_state=42)

# Configurar y entrenar PCA sobre el conjunto de entrenamiento
pca = PCA(n_components=2)
pca.fit(train_data)

# Guardar el modelo PCA entrenado
joblib.dump(pca, 'modelo_pca.pkl')

# Transformar los datos de entrenamiento con PCA
X_pca_train = pca.transform(train_data)

# Reconstruir los datos transformados de entrenamiento
X_projected_train = pca.inverse_transform(X_pca_train)
reconstruction_error_train = np.square(np.subtract(train_data, X_projected_train)).mean(axis=1)

# Detectar anomalías usando un umbral basado en el percentil del error de reconstrucción en los datos de entrenamiento
threshold = np.percentile(reconstruction_error_train, 92)
anomalies_train = reconstruction_error_train > threshold

# Añadir las etiquetas de anomalías al DataFrame de entrenamiento
train_data['Anomaly_PCA'] = anomalies_train

# Mostrar el DataFrame de entrenamiento con las anomalías detectadas
print("Anomalías en el conjunto de entrenamiento:")
print(train_data)
```

```
print(reconstruction_error_train)
```

```
# Graficar el histograma del error de reconstrucción
plt.figure(figsize=(10, 6))
plt.hist(reconstruction_error_train, bins=50, alpha=0.75, color='blue')
plt.axvline(x=threshold, color='r', linestyle='--', label='Umbral de Anomalía')
plt.title('Distribución del Error de Reconstrucción en el Conjunto de Entrenamiento')
plt.xlabel('Error de Reconstrucción')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```

```
# Cargar el modelo PCA desde el archivo
pca_cargado = joblib.load('modelo_pca.pkl')

# Transformar los datos completos con PCA
X_pca_complete = pca_cargado.transform(df_encoded)

# Reconstruir los datos transformados completos
X_projected_complete = pca_cargado.inverse_transform(X_pca_complete)

# Calcular el error de reconstrucción en el conjunto completo
reconstruction_error_complete = np.mean((df_encoded - X_projected_complete) ** 2, axis=1)

# Detectar anomalías usando el umbral basado en el percentil del error de reconstrucción en los datos de entrenamiento
threshold = np.percentile(reconstruction_error_complete, 92)
anomalies_complete = reconstruction_error_complete > threshold

# Añadir las etiquetas de anomalías al DataFrame completo
df_encoded['Anomaly_PCA'] = anomalies_complete
```

```
train_data.shape
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df_encoded['Anomaly_PCA']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación Anomaly_PCA:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
# Asumiendo que 'df_comprobar_scaled' es tu DataFrame escalado y 'preprocess_todo_df' tiene las etiquetas de clasificación
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_comprobar_scaled[columnas_de_interes])

# Convierte el resultado a un DataFrame para facilitar la manipulación
df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

# Añade la columna 'Condicion_smb' al DataFrame de PCA para poder colorear según la clase
df_pca['Anomaly_PCA'] = df_encoded['Anomaly_PCA'].values

# Define una paleta de colores pasteles claros
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde pastel claro
    1: "#fd0101"  # Rojo
}

# Ahora grafica los resultados utilizando la paleta de colores pasteles claros
plt.figure(figsize=(7, 7))

# Ordenar las clases para que 'Normal' sea graficada primero y por lo tanto en el fondo
clases_ordenadas = sorted(df_pca['Anomaly_PCA'].unique(), key=lambda x: x == 1)

for clase in clases_ordenadas:
    if clase == 1:
        alpha_value = 0.3  # Más transparente para 'Anomalía'
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1    # Opaco para 'Normal'
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['Anomaly_PCA'] == clase]['PC1'],
        df_pca[df_pca['Anomaly_PCA'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),  # Usamos 'gray' si la clase no está en la paleta
        alpha=alpha_value,  # Ajuste de la transparencia
        zorder=zorder_value  # Ajuste de zorder para controlar el orden de visualización
    )

plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de detección de anomalías Anomaly_PCA')
plt.legend()
plt.show()
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
```

```python
# Datos normalizados (suponiendo df_scaled como tu DataFrame normalizado)
inertias = []
ks = range(1, 10)  # Ejemplo: evaluar desde 1 hasta 10 clusters

for k in ks:
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(df_comprobar_scaled[columnas_de_interes])
    inertias.append(model.inertia_)

# Graficar el método del codo
plt.figure(figsize=(8, 4))
plt.plot(ks, inertias, '-o')
plt.xlabel('Número de clusters, k')
plt.ylabel('Inercia')
plt.xticks(ks)
plt.title('Método del Codo para Determinar el Número Óptimo de Clusters')
plt.show()
```

```python
from sklearn.metrics import silhouette_score

silhouette_scores = []

for k in ks:
    model = KMeans(n_clusters=k, random_state=42)
    labels = model.fit_predict(df_comprobar_scaled[columnas_de_interes])
    if k > 1:  # Silhouette score sólo tiene sentido si k es mayor que 1
        score = silhouette_score(df_comprobar_scaled[columnas_de_interes], labels)
        silhouette_scores.append(score)

# Graficar los scores de silueta
plt.figure(figsize=(8, 4))
plt.plot(ks[1:], silhouette_scores, '-o')
plt.xlabel('Número de clusters, k')
plt.ylabel('Silhouette Score')
plt.xticks(ks[1:])
plt.title('Score de Silueta para Diferentes Números de Clusters')
plt.show()
```

```python
# Identificar el mejor k (el valor de k que maximiza el Silhouette Score)
best_k = ks[1:][np.argmax(silhouette_scores)]
print(f"El mejor número de clusters es {best_k}")
```

```python
best_k=3
# Entrenar el modelo K-Means con el mejor k
best_model = KMeans(n_clusters=best_k, random_state=42)
best_labels = best_model.fit_predict(df_comprobar_scaled[columnas_de_interes])

# Añadir las etiquetas al DataFrame original
df['label_Kmeans'] = best_labels
df_comprobar_scaled['label_Kmeans']=best_labels
```

```python
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_comprobar_scaled[columnas_de_interes])

df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])

df_pca['label_Kmeans'] = df['label_Kmeans'].values

# paleta de colores pasteles claros
colores_pasteles_claros = {
    0: "#9bbb59",  # Verde pastel claro
    1: "#fd0101"  # Rojo
}

plt.figure(figsize=(10, 7))

# Ordenar las clases para que 'Normal' sea graficada primero y por lo tanto en el fondo
clases_ordenadas = sorted(df_pca['label_Kmeans'].unique(), key=lambda x: x == 1)

for clase in clases_ordenadas:
    if clase == '1':
        alpha_value = 0.3  # Más transparente para 'Normal'
        zorder_value = 1   # Menor zorder para estar en el fondo
    else:
        alpha_value = 1.0  # Opaco para otras clases
        zorder_value = 2   # Mayor zorder para estar en frente

    plt.scatter(
        df_pca[df_pca['label_Kmeans'] == clase]['PC1'],
        df_pca[df_pca['label_Kmeans'] == clase]['PC2'],
        label=f'Clase {clase}',
        color=colores_pasteles_claros.get(clase, 'gray'),  # Usamos 'gray' si la clase no está en la paleta
        alpha=alpha_value,  # Ajuste de la transparencia
        zorder=zorder_value  # Ajuste de zorder para controlar el orden de visualización
    )

plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de Clusters por KMeans')
plt.legend()
plt.show()
```

```python
# Guardar el modelo en un archivo
with open('/content/kmeans.pkl', 'wb') as file:
    pickle.dump(best_model, file)
```

```python
# Cargar el modelo
with open('/content/kmeans.pkl', 'rb') as file:
    kmeans_model = pickle.load(file)

print(type(kmeans_model))  #KMeans
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_']
scaler = MinMaxScaler()
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
```

```python
# Entrenar el modelo K-Means con el mejor k
best_model = kmeans_model
# Realizar las predicciones
# Realizar las predicciones
best_labels = kmeans_model.predict(df_comprobar_scaled[columnas_de_interes])

# Añadir las etiquetas al DataFrame original
df['label_Kmeans'] = best_labels
df_comprobar_scaled['label_Kmeans'] = best_labels
```

```python
df['label_Kmeans'] = df['label_Kmeans'].replace({2: 1, 1: 1})
```

```python
df['label_Kmeans'].unique()
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
y_test=df['estado']
y_pred=df['label_Kmeans']
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación K Means:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
print("Matriz de Confusión:\n", conf_matrix)
print("\nReporte de Clasificación:\n", classification_report(y_true,y_pred))
print("Precisión:", accuracy_score(y_true, y_pred))
print("AUC-ROC:", roc_auc_score(y_true, y_pred))
print("AUC-PR:", average_precision_score(y_true, y_pred))
print("F1-Score:", f1_score(y_true, y_pred))
```

```python
# Graficar la matriz de confusión
plt.figure(figsize=(3, 3))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
from sklearn.metrics import silhouette_score

# Supongamos que k es el número óptimo de clusters que encontramos del método del codo
k = 2
model = KMeans(n_clusters=k)
labels = model.fit_predict(df_comprobar_scaled[columnas_de_interes])

# Calcular el Silhouette Score
silhouette = silhouette_score(df_comprobar_scaled[columnas_de_interes], labels)
print('Silhouette Score:', silhouette)
```

```python
file_path = '/content/DatafinalCondicion.xlsx'
df_1 = pd.read_excel(file_path)
```

```python
df=df_1.copy()
```

```python
scaler = MinMaxScaler()
df_comprobar_scaled = scaler.fit_transform(df[columnas_de_interes])
df_comprobar_scaled = pd.DataFrame(df, columns=df[columnas_de_interes].columns)
```

```python
df['idcomponente']=df['equipo']+df['componente']
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']
```

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import KNNImputer
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline


columnas_de_interes = ['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                       'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
                       'Barioppm', 'Antimonioppm', 'Niquelppm', 'Aluminioppm', 'Boroppm', 'Estañoppm.1',
                       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
                       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_', 'Diesel_', 'Turbidez_NTU_', 'Idcomponente']

X = df[columnas_de_interes]
y = df['etiquetado_covarianza_robusta']

# onjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# columnas numéricas y categóricas
numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = ['Idcomponente']

# Transformando de columnas
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', KNNImputer(n_neighbors=5)),
            ('scaler', StandardScaler())#NORMALIZADO
        ]), numeric_features),
        ('cat', Pipeline([
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_features)
    ])
```

```python
#pipeline completo con el preprocesador y el clasificador
pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', RandomForestClassifier(random_state=42))
])

# Parámetros que quieres buscar, ajustados para el pipeline
param_grid = {
    'classifier__n_estimators': [50,100,200,300],  # Probar con más árboles
    'classifier__max_features': ['auto'],  # Diferentes estrategias para la selección de características
    'classifier__max_depth': [10,20,30],  # Profundidades variadas para ver el efecto
    'classifier__min_samples_split': [3,5,8],  # Número mínimo de muestras requeridas para dividir un nodo
    'classifier__min_samples_leaf': [3,5,8],  # Número mínimo de muestras requeridas en un nodo hoja
    'classifier__criterion': ['gini', 'entropy']  # Medida de calidad de una división
}

# Configurar GridSearchCV para utilizar el pipeline
grid_search = GridSearchCV(estimator=pipeline,
                           param_grid=param_grid,
                           scoring='balanced_accuracy',  # 'balanced_accuracy' para las clases están desbalanceadas
                           cv=5,
                           verbose=1,
                           n_jobs=-1)

grid_search.fit(X_train, y_train)

print("Mejor puntuación:", grid_search.best_score_)
print("Mejores parámetros:", grid_search.best_params_)
```

```python
best_model = grid_search.best_estimator_

# Predicciones y probabilidades sobre el conjunto de prueba
y_pred = best_model.predict(X_test)
y_probs = best_model.predict_proba(X_test)[:, 1]  # Probabilidades para la clase positiva

# Mostrar la precisión, la matriz de confusión y otros informes
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Calcular y mostrar el AUC de la ROC
print("ROC AUC:", roc_auc_score(y_test, y_probs))

# Dibujar la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc_score(y_test, y_probs))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```python
# Suponiendo que 'df' es tu DataFrame y 'columnas_de_interes' son las características que vas a usar
columnas_de_interes = ['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                       'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
                       'Barioppm', 'Antimonioppm', 'Niquelppm', 'Aluminioppm', 'Boroppm', 'Estañoppm.1',
                       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
                       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_', 'Diesel_', 'Turbidez_NTU_', 'idcomponente']

X = df[columnas_de_interes]
y = df['etiquetado_covarianza_robusta']

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

# Predicciones y probabilidades sobre el conjunto de prueba
y_pred = best_model.predict(X_test)
y_probs = best_model.predict_proba(X_test)[:, 1]  # Probabilidades para la clase positiva

# Mostrar la precisión, la matriz de confusión y otros informes
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Calcular y mostrar el AUC de la ROC
print("ROC AUC:", roc_auc_score(y_test, y_probs))

# Dibujar la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc_score(y_test, y_probs))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Decodificar y_test si está en formato OneHot
if y_test.ndim > 1 and y_test.shape[1] > 1:
    y_test_decoded = np.argmax(y_test, axis=1)
else:
    y_test_decoded = y_test

# Predicción usando el mejor modelo
y_pred = best_model.predict(X_test)

# Calcula la matriz de confusión
conf_matrix = confusion_matrix(y_test_decoded, y_pred)
class_names = ['Normal', 'Anomalo']
# Visualiza la matriz de confusión utilizando seaborn
plt.figure(figsize=(5, 5))
heatmap=sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues", cbar=False,annot_kws={"size": 16},xticklabels=class_names, yticklabels=class_names)
# Ajustar el tamaño de las etiquetas
heatmap.set_title('Confusion Matrix')

plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

```python
# Definir una función para graficar la curva de aprendizaje
def plot_learning_curve(estimator, X, y, title='Learning Curve', ylim=None, cv=None,
                        n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")

    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes, scoring='accuracy')

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")

    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt

# Graficar la curva de aprendizaje con 7 pliegues
plot_learning_curve(grid_search.best_estimator_, X_train, y_train, cv=7, n_jobs=-1,
                    title='Learning Curve for RandomForest Anomaly Detection')
plt.show()
```

```python
joblib.dump(best_model, joblib_file)
print(f"Modelo guardado en {joblib_file}")
```

```python
# Asumiendo que tienes un DataFrame de pandas para tus características
feature_names = X.columns

# Crear un DataFrame para almacenar las importancias y los nombres de las características
importances_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
})

# Ordenar el DataFrame por importancia
importances_df = importances_df.sort_values(by='Importance', ascending=False)
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))  # Configura el tamaño de la figura
plt.barh(importances_df['Feature'], importances_df['Importance'], color='skyblue')  # Crea un gráfico de barras horizontal
plt.xlabel('Importance')  # Etiqueta del eje X
plt.ylabel('Feature')  # Etiqueta del eje Y
plt.title('Feature Importances in RandomForest Model')  # Título del gráfico
plt.gca().invert_yaxis()  # Invierte el eje Y para que las barras más importantes aparezcan arriba
plt.show()
```

```python
import joblib
# Cargar el modelo
modelo_anomalia_file = "/content/random_forest_model_anomalia.pkl"
modelo_anomalia = joblib.load(modelo_anomalia_file)
print("Modelo cargado correctamente")
```

```python
file_path = '/content/test_data.xlsx'
df_1_test = pd.read_excel(file_path)
df_test=df_1_test.copy()
```

```python
df_test['idcomponente']=df_test['equipo']+df_test['componente']
```

```python
columnas_de_interes=['Sodioppm', 'Potasioppm', 'PartículasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                     'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']
```

```python
y_predict=modelo_anomalia.predict(df_test[columnas_de_interes])
```

```python
df_hu = pd.DataFrame(y_predict)
```

```python
df_test['estado'] = df_test['condicion_smb.1'].replace({'Crítico': 1, 'Normal': 0, 'Seguimiento': 1})
df_test['estado'].unique()
```

```
y_test=df_test['estado']
y_pred=y_predict
cm = confusion_matrix(y_test, y_pred)
print("Reporte de Clasificación ANOMALIAS -TEST RANDOM FOREST:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
# Graficar la matriz de confusión
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
print("Precisión del modelo:", accuracy_score(y_test, y_predict))
```

```
#. Error de Clasificación
accuracy = accuracy_score(y_test, y_pred)
classification_error = 1 - accuracy
#Error Cuadrático Medio (MSE) para Regresión
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
importances = grid_search.best_estimator_.named_steps['classifier'].feature_importances_
```

```
pip install shap
```

```
# Preprocesar los datos de prueba usando el pipeline
X_test_preprocessed = grid_search.best_estimator_.named_steps['preprocessor'].transform(X_test)

# Usar el explainer SHAP con el modelo entrenado
explainer = shap.TreeExplainer(grid_search.best_estimator_.named_steps['classifier'])
shap_values = explainer.shap_values(X_test_preprocessed)
```

```
X_test.shape
```

```
X_test_preprocessed.shape
```

```
explainer.expected_value[class_idx]
```

```
# Inicializar JavaScript para las visualizaciones de SHAP
shap.initjs()
print("shap.initjs() ejecutado correctamente")

# Verificar las dimensiones y tipo de shap_values
print(f"Tipo de shap_values: {type(shap_values)}")
print(f"Dimensiones exactas de shap_values: {shap_values.shape}")

# Seleccionar una clase específica para visualización, por ejemplo, la clase 0
class_idx = 0

# Seleccionar una fila aleatoria
random_idx = np.random.randint(0, X_test_preprocessed.shape[0])

print(f"Fila seleccionada aleatoriamente: {random_idx}")

# Convertir X_test_preprocessed de nuevo a DataFrame con nombres de columnas adecuados
preprocessor.fit(X_train)  # Asegurarse de que el preprocessor esté ajustado
X_test_preprocessed_df = pd.DataFrame(X_test_preprocessed, columns=preprocessor.get_feature_names_out())

# Filtrar las columnas que deseas mostrar (excluir la columna categórica 'idcomponente')
selected_columns = [col for col in X_test_preprocessed_df.columns if 'idcomponente' not in col]
selected_indices = [X_test_preprocessed_df.columns.get_loc(col) for col in selected_columns]

# Filtrar los valores SHAP y el DataFrame para las columnas seleccionadas
shap_values_filtered = shap_values[:, selected_indices, class_idx]
sample_for_visualization_filtered = X_test_preprocessed_df.iloc[random_idx, selected_indices]

# Visualizar explicaciones para la fila aleatoria
shap.force_plot(
    explainer.expected_value[class_idx],
    shap_values[random_idx, selected_indices, class_idx],
    sample_for_visualization_filtered
)
```

```
# Guardar el objeto explainer y los shap_values en un archivo
joblib.dump(explainer, 'explainerRandomForestAnomalia.joblib')
joblib.dump(shap_values, 'shap_valuesRandomForestAnomalia.joblib')
```

```
# Para cargar los objetos nuevamente
loaded_explainer = joblib.load('explainerRandomForestAnomalia.joblib')
loaded_shap_values = joblib.load('shap_valuesRandomForestAnomalia.joblib')
```

```
import shap
```

```python
# Inicializar JavaScript para las visualizaciones de SHAP
shap.initjs()
print("shap.initjs() ejecutado correctamente")

# Verificar las dimensiones y tipo de shap_values
print(f"Tipo de shap_values: {type(shap_values)}")
print(f"Dimensiones exactas de shap_values: {shap_values.shape}")

# Seleccionar una clase específica para visualización, por ejemplo, la clase 0
class_idx = 0

# Seleccionar una fila aleatoria
random_idx = np.random.randint(0, X_test_preprocessed.shape[0])

print(f"Fila seleccionada aleatoriamente: {random_idx}")

# Convertir X_test_preprocessed de nuevo a DataFrame con nombres de columnas adecuados
preprocessor.fit(X_train)  # Asegurarse de que el preprocessor esté ajustado
X_test_preprocessed_df = pd.DataFrame(X_test_preprocessed, columns=preprocessor.get_feature_names_out())

# Filtrar las columnas que deseas mostrar (excluir la columna categórica 'idcomponente')
selected_columns = [col for col in X_test_preprocessed_df.columns if 'idcomponente' not in col]
selected_indices = [X_test_preprocessed_df.columns.get_loc(col) for col in selected_columns]

# Filtrar los valores SHAP y el DataFrame para las columnas seleccionadas
shap_values_filtered = shap_values[:, selected_indices, class_idx]
sample_for_visualization_filtered = X_test_preprocessed_df.iloc[random_idx, selected_indices]

# Visualizar explicaciones para la fila aleatoria
shap.force_plot(
    explainer.expected_value[class_idx],
    shap_values[random_idx, selected_indices, class_idx],
    sample_for_visualization_filtered
)
```

```python
from pivottablejs import pivot_ui
import IPython
pivot_ui(df[columnas_de_interes])
IPython.display.HTML(filename='/content/pivottablejs.html')
```

```
pip install pivottablejs
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
    Dense(100, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(50, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='softmax')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```python
history_redNeuronal=model.fit(X_train, y_train, class_weight=class_weight_dict, epochs=100, batch_size=32,validation_split=0.2)
```

```python
# Evaluar el modelo en el conjunto de prueba
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_accuracy)
print("Test Loss:", test_loss)
```

```python
import matplotlib.pyplot as plt

# Graficar la precisión de entrenamiento y validación
plt.figure(figsize=(8, 4))
plt.plot(history_redNeuronal.history['accuracy'], label='Accuracy (training data)')
plt.plot(history_redNeuronal.history['val_accuracy'], label='Accuracy (validation data)')
plt.title('Accuracy Over Epochs')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()

# Graficar la pérdida de entrenamiento y validación
plt.figure(figsize=(8, 4))
plt.plot(history_redNeuronal.history['loss'], label='Loss (training data)')
plt.plot(history_redNeuronal.history['val_loss'], label='Loss (validation data)')
plt.title('Loss Over Epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```

```python
from sklearn.metrics import classification_report, confusion_matrix

# Hacer predicciones
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int)  # Convertir probabilidades a etiquetas binarias

# Matriz de confusión
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

# Informe de clasificación
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV, StratifiedKFold, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.preprocessing import label_binarize
from itertools import cycle
from imblearn.over_sampling import SMOTE, RandomOverSampler
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.metrics import roc_curve, auc, roc_auc_score
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve, precision_score, recall_score, f1_score
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.preprocessing import label_binarize
from itertools import cycle
from imblearn.over_sampling import SMOTE  # Importar SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline  # Importar Pipeline de imblearn
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier  # O RandomForestRegressor si es un problema de regresión
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score  # Cambia por la métrica adecuada para tu problema
```

```python
file_path = '/content/DatafinalCondicion.xlsx'
df_1 = pd.read_excel(file_path)
```

```python
df=df_1.copy()
```

```python
columnas_de_interes=['Sodioppm', 'Silicioppm', 'Platappm', 'Niquelppm', 'Aluminioppm', 'Plomoppm',
 'Estañoppm', 'Cobreppm', 'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
  'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_',
 'Boroppm', 'Turbidez_NTU_','Molidatos_0_1200_ppm','Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3','NitraciónABS_cm',
       'Sulfatos_0_3000_ppm', 'Fosforoppm','Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
        'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte','idcomponente']
```

```python
# Suponiendo que 'df' es tu DataFrame y 'columnas_de_interes' son las características que vas a usar
columnas_de_interes = ['Sodioppm', 'Silicioppm', 'Platappm', 'Niquelppm', 'Aluminioppm', 'Plomoppm',
 'Estañoppm', 'Cobreppm', 'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
  'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_',
 'Boroppm', 'Turbidez_NTU_','Molidatos_0_1200_ppm','Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3','NitraciónABS_cm',
       'Sulfatos_0_3000_ppm', 'Fosforoppm','Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
        'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte','idcomponente']

X = df[columnas_de_interes]
y = df['condicion_smb.1']

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

# Lista de columnas numéricas y categóricas
numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = ['idcomponente']

# transformador de columnas
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', KNNImputer(n_neighbors=5)),
            ('scaler', StandardScaler())
        ]), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

# pipeline completo con el preprocesador y el clasificador
pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', RandomForestClassifier(class_weight='balanced', random_state=42))
])
```

```python
# Definir los parámetros que quieres buscar, ajustados para el pipeline
param_grid = {
    'classifier__n_estimators': [100,200,300],  # Probar con más árboles
    'classifier__max_features': ['auto'],  # Diferentes estrategias para la selección de características
    'classifier__max_depth': [10,15,20],  # Profundidades variadas para ver el efecto
    'classifier__min_samples_split': [3,5],  # Número mínimo de muestras requeridas para dividir un nodo
    'classifier__min_samples_leaf': [3,5],  # Número mínimo de muestras requeridas en un nodo hoja
    'classifier__criterion': ['gini','entropy']  # Medida de calidad de una división
}

# Configurar GridSearchCV para utilizar el pipeline
cv = StratifiedKFold(n_splits=7)
grid_search = GridSearchCV(estimator=pipeline,
                           param_grid=param_grid,
                           scoring='balanced_accuracy',  # Métrica de evaluación
                           cv=cv,  # Validación cruzada estratificada
                           verbose=1,  # Muestra más información durante el entrenamiento
                           n_jobs=-1)  # Usa todos los cores de tu CPU

# Medir el tiempo de entrenamiento
start_time = time.time()
grid_search.fit(X_train, y_train)
end_time = time.time()

# Calcular el tiempo de entrenamiento
training_time = end_time - start_time
print(f"Tiempo de entrenamiento: {training_time:.2f} segundos")
```

```python
# Predicciones y probabilidades sobre el conjunto de prueba
y_pred = grid_search.predict(X_test)
y_score = grid_search.predict_proba(X_test)

# Imprimir el orden de las clases
class_order = grid_search.best_estimator_.named_steps['classifier'].classes_
print(f"Order of classes: {class_order}")

# Generar reporte de clasificación y matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_report(y_test, y_pred, target_names=class_order))

# Graficar la matriz de confusión
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=class_order, yticklabels=class_order)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

# Mapear las etiquetas a índices
class_map = {label: idx for idx, label in enumerate(class_order)}
y_test_mapped = np.array([class_map[label] for label in y_test])

# Binarizar las etiquetas para calcular las curvas ROC
y_test_binarized = label_binarize(y_test_mapped, classes=list(class_map.values()))

# Calcular FPR, TPR, y AUC para cada clase
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(len(class_map)):
    fpr[i], tpr[i], _ = roc_curve(y_test_binarized[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Graficar la curva ROC
plt.figure(figsize=(10, 8))
colors = cycle(['red', 'green', 'yellow'])
for i, color in zip(range(len(class_map)), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label=f'ROC curve of class {list(class_map.keys())[i]} (area = {roc_auc[i]:0.2f})')
```

```python
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic for Multi-class')
plt.legend(loc="lower right")
plt.show()
```

```python
# Mejores parámetros
print("Mejores parámetros:", grid_search.best_params_)

# Evaluar el modelo
best_model = grid_search.best_estimator_

# Predicciones y probabilidades sobre el conjunto de prueba
y_pred = best_model.predict(X_test)
y_probs = best_model.predict_proba(X_test)  # Probabilidades para todas las clases
```

```python
import joblib
```

```python
from joblib import dump, load

joblib_file = "/content/random_forest_model_condicion.pkl"
joblib.dump(best_model, joblib_file)
print(f"Modelo guardado en {joblib_file}")
```

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier  # O RandomForestRegressor si es un problema de regresión
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score  # Cambia por la métrica adecuada para tu problema

# Supongamos que 'df' es tu DataFrame y que 'target' es la columna objetivo
X = df[columnas_de_interes]
y = df['condicion_smb.1']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
# Calcular la curva de aprendizaje
train_sizes, train_scores, test_scores = learning_curve(
    best_model, X_train, y_train, cv=cv, scoring='f1_weighted', n_jobs=-1,
    train_sizes=np.linspace(0.1, 1.0, 10), verbose=1)

# Calcular medias y desviaciones estándar de las puntuaciones de entrenamiento y validación
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

# Graficar la curva de aprendizaje
plt.figure(figsize=(10, 6))
plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1,
                 color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
         label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
         label="Cross-validation score")

plt.title('Learning Curve')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend(loc="best")
plt.grid()
plt.show()
```

```python
rf_classifier = best_model.named_steps['randomforestclassifier']
feature_importances = pd.Series(rf_classifier.feature_importances_, index=X_train.columns)
feature_importances = feature_importances.sort_values(ascending=False)
plt.figure(figsize=(12, 6))
feature_importances.plot(kind='bar')
plt.title('Importancia de las Características')
plt.xlabel('Características')
plt.ylabel('Importancia')
plt.show()
```

```python
# Preprocesar los datos de prueba usando el pipeline
X_test_preprocessed = grid_search.best_estimator_.named_steps['preprocessor'].transform(X_test)

# Usar el explainer SHAP con el modelo entrenado
explainer_2 = shap.TreeExplainer(grid_search.best_estimator_.named_steps['classifier'])
shap_values_2 = explainer.shap_values(X_test_preprocessed)
# Inicializar JavaScript para las visualizaciones de SHAP
```

```python
shap.initjs()
print("shap.initjs() ejecutado correctamente")

print(f"Tipo de shap_values: {type(shap_values_2)}")
print(f"Dimensiones exactas de shap_values: {shap_values_2.shape}")

# Seleccionar una clase específica para visualización, por ejemplo, la clase 0
class_idx = 0

# Seleccionar una fila aleatoria
random_idx = np.random.randint(0, X_test_preprocessed.shape[0])

print(f"Fila seleccionada aleatoriamente: {random_idx}")

# Convertir X_test_preprocessed de nuevo a DataFrame con nombres de columnas adecuados
preprocessor.fit(X_train)  # Asegurarse de que el preprocessor esté ajustado
X_test_preprocessed_df = pd.DataFrame(X_test_preprocessed, columns=preprocessor.get_feature_names_out())

# Filtrar las columnas que deseas mostrar (excluir la columna categórica 'idcomponente')
selected_columns = [col for col in X_test_preprocessed_df.columns if 'idcomponente' not in col]
selected_indices = [X_test_preprocessed_df.columns.get_loc(col) for col in selected_columns]

# Filtrar los valores SHAP y el DataFrame para las columnas seleccionadas
shap_values_filtered = shap_values_2[:, selected_indices, class_idx]
sample_for_visualization_filtered = X_test_preprocessed_df.iloc[random_idx, selected_indices]

# Visualizar explicaciones para la fila aleatoria
shap.force_plot(
    explainer_2.expected_value[class_idx],
    shap_values_2[random_idx, selected_indices, class_idx],
    sample_for_visualization_filtered
)
```

```python
joblib.dump(explainer, 'shap_explainer_condicion.pkl')
joblib.dump(shap_values, 'shap_values_condicion.pkl')
print("Explainer y valores SHAP guardados en 'shap_explainer.pkl' y 'shap_values.pkl'")
```

```python
import joblib
```

```python
# Para cargar los objetos nuevamente
loaded_explainer = joblib.load('/content/shap_explainer_condicion.pkl')
loaded_shap_values = joblib.load('/content/shap_values_condicion.pkl')
```

```python
# Inicializar JavaScript para las visualizaciones de SHAP
shap.initjs()
print("shap.initjs() ejecutado correctamente")

# Verificar las dimensiones y tipo de shap_values
print(f"Tipo de shap_values: {type(shap_values)}")
print(f"Dimensiones exactas de shap_values: {shap_values.shape}")

# Seleccionar una clase específica para visualización, por ejemplo, la clase 0
class_idx = 0

# Seleccionar una fila aleatoria
random_idx = np.random.randint(0, X_test_preprocessed.shape[0])

print(f"Fila seleccionada aleatoriamente: {random_idx}")

# Convertir X_test_preprocessed de nuevo a DataFrame con nombres de columnas adecuados
preprocessor.fit(X_train)  # Asegurarse de que el preprocessor esté ajustado
X_test_preprocessed_df = pd.DataFrame(X_test_preprocessed, columns=preprocessor.get_feature_names_out())

# Filtrar las columnas que deseas mostrar (excluir la columna categórica 'idcomponente')
selected_columns = [col for col in X_test_preprocessed_df.columns if 'idcomponente' not in col]
selected_indices = [X_test_preprocessed_df.columns.get_loc(col) for col in selected_columns]

# Filtrar los valores SHAP y el DataFrame para las columnas seleccionadas
shap_values_filtered = shap_values[:, selected_indices, class_idx]
sample_for_visualization_filtered = X_test_preprocessed_df.iloc[random_idx, selected_indices]

# Visualizar explicaciones para la fila aleatoria
shap.force_plot(
    explainer.expected_value[class_idx],
    shap_values[random_idx, selected_indices, class_idx],
    sample_for_visualization_filtered
)
```

```python
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt

# Suponiendo que ya tienes un modelo entrenado llamado `best_model`
# y un conjunto de prueba `X_test` y `y_test`

# Realizar la importancia de características por permutación
result = permutation_importance(best_model, X_test, y_test, n_repeats=30, random_state=42, n_jobs=-1)

# Obtener las importancias medias y los índices de las características ordenados por importancia
importances_mean = result.importances_mean
sorted_idx = result.importances_mean.argsort()

# Graficar la importancia de las características
plt.figure(figsize=(12, 8))
plt.barh(range(len(sorted_idx)), importances_mean[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), X_test.columns[sorted_idx])
plt.xlabel("Importancia de la característica (media)")
plt.title("Importancia de la característica por permutación")
plt.show()
```

```python
df['RandomForest']=best_model.predict(df[columnas_de_interes])
```

```python
columnas_de_interes = ['Sodioppm', 'Silicioppm', 'Platappm', 'Niquelppm', 'Aluminioppm', 'Plomoppm',
 'Estañoppm', 'Cobreppm', 'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
  'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_',
 'Boroppm', 'Turbidez_NTU_','Molidatos_0_1200_ppm','Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3','NitraciónABS_cm',
       'Sulfatos_0_3000_ppm', 'Fosforoppm','Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
        'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte','idcomponente']
```

```python
import joblib
model_condicion = joblib.load('/content/random_forest_model_condicion.pkl')
```

```python
print("Modelo cargado correctamente")
file_path = '/content/test_data.xlsx'
df_1_test = pd.read_excel(file_path)
df_test=df_1_test.copy()
df_test['idcomponente']=df_test['equipo']+df_test['componente']
y_predict=model_condicion.predict(df_test[columnas_de_interes])
y_pred=y_predict
```

```python
y_test=df_test['condicion_smb.1']
```

```python
from sklearn.preprocessing import LabelEncoder
```

```python
# Paso 1: Convertir las etiquetas de clase a números
label_encoder = LabelEncoder()
y_test_encoded = label_encoder.fit_transform(y_test)
y_pred_encoded = label_encoder.transform(y_pred)

# Paso 2: Generar la matriz de confusión
conf_matrix = confusion_matrix(y_test_encoded, y_pred_encoded)

# Nombres de las clases
class_names = label_encoder.classes_
print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(y_test_encoded, y_pred_encoded))

# Paso 3: Visualizar la matriz de confusión con etiquetas de clase
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.ylabel('Etiqueta Verdadera')
plt.xlabel('Etiqueta Predicha')
plt.title('Matriz de Confusión con Etiquetas de Clase')
plt.show()
```

```python
df_hu['condicion_condicion']=y_pred
```

```python
df_hu.to_excel('/content/test_data_eti.xlsx')
```

```python
import joblib
# Cargar el modelo
modelo_condicion_file = "/content/random_forest_model_condicion.pkl"
modelo_condicion = joblib.load(modelo_condicion_file)
print("Modelo cargado correctamente")
```

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

df_test['idcomponente']=df_test['equipo']+df_test['componente']

columnas_de_interes=['Sodioppm', 'Silicioppm', 'Titanioppm', 'Platappm', 'Niquelppm', 'Aluminioppm', 'Plomoppm',
 'Estañoppm', 'Cobreppm', 'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
  'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_',
 'Boroppm', 'Turbidez_NTU_','Molidatos_0_1200_ppm','Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3','NitraciónABS_cm',
       'Sulfatos_0_3000_ppm', 'Fosforoppm','Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
        'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte','idcomponente']
columnas_anomaly=['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                  'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
       'Barioppm','Antimonioppm', 'Niquelppm',
       'Aluminioppm', 'Boroppm', 'Estañoppm.1',
       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_',
       'Diesel_', 'Turbidez_NTU_','idcomponente']

X_test_condicion= df_test[columnas_de_interes]
X_test_anomaly= df_test[columnas_anomaly]
y_multiclass = df_test['condicion_smb.1']

# Predicciones
rf_anomaly_pred = modelo_anomalia.predict(X_test_anomaly)
rf_multiclass_pred = modelo_condicion.predict(X_test_condicion)
```

```python
# Mapear las clases del modelo multicategórico
anomaly_map = {'Normal': 0, 'Seguimiento': 1, 'Critico': 1}
rf_multiclass_mapped_pred = np.array([anomaly_map[label] for label in rf_multiclass_pred])

# Votación mayoritaria
final_pred = np.array([1 if (rf_anomaly_pred[i] == 1 or rf_multiclass_mapped_pred[i] == 1) else 0 for i in range(len(rf_anomaly_pred))])

# Evaluación
print("Confusion Matrix:")
print(confusion_matrix(rf_multiclass_mapped_pred, final_pred))
print("\nClassification Report:")
print(classification_report(rf_multiclass_mapped_pred, final_pred, target_names=['No Anomaly', 'Anomaly']))

# Graficar la matriz de confusión
conf_matrix = confusion_matrix(rf_multiclass_mapped_pred, final_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['No Anomaly', 'Anomaly'], yticklabels=['No Anomaly', 'Anomaly'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

```python
df_hu.to_excel('/content/test_data_eti.xlsx')
```

```python
import pandas as pd
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display

# Suponiendo que tu DataFrame se llama df
# df = pd.read_csv('tu_archivo.csv')  # Si estás cargando desde un archivo CSV

# Función para filtrar y graficar los datos
def plot_time_series(equipo, componente, start_date, parametro):
    if start_date is None:
        print("Por favor, seleccione una fecha de inicio.")
        return

    # Filtrar datos
    filtered_df = df[(df['equipo'] == equipo) &
                     (df['componente'] == componente) &
                     (df['i_mueFecha'] >= pd.to_datetime(start_date))]

    # Configurar el gráfico
    plt.figure(figsize=(15, 5))

    # Graficar el parámetro seleccionado
    plt.plot(filtered_df['i_mueFecha'], filtered_df[parametro], label=parametro)
    plt.title(f'{parametro} para {equipo} - {componente} desde {start_date}')
    plt.xlabel('Fecha')
    plt.ylabel('Valores')
    plt.legend(loc='upper left')
    plt.grid(True)

    plt.show()

# Crear widgets de selección
equipo_widget = widgets.Dropdown(options=df['equipo'].unique(), description='Equipo:')
componente_widget = widgets.Dropdown(options=df['componente'].unique(), description='Componente:')
start_date_widget = widgets.DatePicker(description='Fecha Inicio:', disabled=False)
parametro_widget = widgets.Dropdown(options=df.columns[6:], description='Parámetro:')

# Función para actualizar el gráfico basado en la selección del widget
def on_button_clicked(b):
    plot_time_series(equipo_widget.value, componente_widget.value, start_date_widget.value, parametro_widget.value)

# Crear el botón de graficar
button = widgets.Button(description="Graficar Parámetro")
button.on_click(on_button_clicked)
```

```python
# Mostrar widgets y botón
display(equipo_widget, componente_widget, start_date_widget, parametro_widget, button)
```

```
pip install tensorflow
```

```python
columnas_de_interes=['Sodioppm', 'Silicioppm', 'Titanioppm', 'Platappm', 'Niquelppm', 'Aluminioppm', 'Plomoppm',
 'Estañoppm', 'Cobreppm', 'Cromoppm', 'Fierroppm', 'Viscosidada100_CcSt', 'HollinABS_01mm', 'SulfataciónABS_cm',
  'Agua_', 'TBN_mgKOH_g_', 'ParticulasFerrosas_PQ_', 'Diesel_',
 'Boroppm', 'Turbidez_NTU_','Molidatos_0_1200_ppm','Molibdenoppm',
       'Calcioppm', 'GLYCOL_', 'CodigoISO_1', 'CodigoISO_2', 'CodigoISO_3','NitraciónABS_cm',
       'Sulfatos_0_3000_ppm', 'Fosforoppm','Nitritos_0_3200_ppm', 'Zincppm', 'Cloruros_0_400_ppm',
       'PotencialHidrogeno_6_11_pH',
        'i_mueHoraProducto', 'i_mueHoraEquipo',
       'i_mueHoraParte' ]
```

```python
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
import numpy as np
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras import backend as K
```

```python
file_path = '/content/DatafinalCondicion.xlsx'
df_1 = pd.read_excel(file_path)
```

```python
df=df_1.copy()
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
import numpy as np
from imblearn.over_sampling import ADASYN
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
from tensorflow.keras import backend as K
from sklearn.ensemble import VotingClassifier

# Definir la función de pérdida focal con parámetros ajustados
def focal_loss(gamma=3., alpha=0.5):
    def focal_loss_fixed(y_true, y_pred):
        y_true = K.cast(y_true, tf.float32)
        y_pred = K.cast(y_pred, tf.float32)
        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1. - epsilon)
        cross_entropy = -y_true * K.log(y_pred)
        loss = alpha * K.pow(1 - y_pred, gamma) * cross_entropy
        return K.sum(loss, axis=1)
    return focal_loss_fixed

# Supongamos que 'df_comprobar_scaled' es tu DataFrame de características escaladas
# y 'columnas_de_interes' son las columnas de características que vas a usar
X = df[columnas_de_interes]
y = df['condicion_smb.1']

# Convertir las características a tipo float si es necesario
X = X.astype(float)

# Verificar las clases originales
print("Clases originales en y:")
print(y.value_counts())

# Imputación de valores faltantes por la media
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Normalizar las características
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```python
# Codificar las etiquetas usando OneHotEncoder
encoder = OneHotEncoder(sparse=False)
y_encoded = encoder.fit_transform(y.values.reshape(-1, 1))

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

# Decodificar las etiquetas para aplicar submuestreo y ADASYN
y_train_decoded = np.argmax(y_train, axis=1)

# Ajustar el submuestreo de la clase mayoritaria asegurando que las muestras no superen las disponibles
class_counts = np.bincount(y_train_decoded)
sampling_strategy = {0: min(class_counts[0] // 2, class_counts[0]),
                     1: class_counts[1],
                     2: min(class_counts[2] * 2, class_counts[2])}
print("Estrategia de submuestreo:", sampling_strategy)

rus = RandomUnderSampler(sampling_strategy=sampling_strategy, random_state=42)
X_train_res, y_train_res = rus.fit_resample(X_train, y_train_decoded)

# Aplicar ADASYN para balancear las clases en el conjunto de entrenamiento
adasyn = ADASYN(random_state=42)
X_train_balanced, y_train_balanced = adasyn.fit_resample(X_train_res, y_train_res)

# Codificar las etiquetas después de aplicar ADASYN
y_train_balanced_encoded = encoder.fit_transform(y_train_balanced.reshape(-1, 1))

# Verificar la distribución de clases en los conjuntos de entrenamiento y prueba
train_class_distribution = pd.Series(y_train_balanced).value_counts(normalize=True)
test_class_distribution = pd.Series(np.argmax(y_test, axis=1)).value_counts(normalize=True)

print("Distribución de clases en el conjunto de entrenamiento después de ADASYN:\n", train_class_distribution)
print("Distribución de clases en el conjunto de prueba:\n", test_class_distribution)

# Ajuste manual de parámetros para Random Forest
rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=20,
    min_samples_split=5,
    min_samples_leaf=5,
    random_state=42
)

# Entrenar el modelo de Random Forest
rf_model.fit(X_train_balanced, y_train_balanced)
```

```python
# Definir el modelo de Red Neuronal
nn_model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],), kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(60, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(y_encoded.shape[1], activation='softmax')
])

# Compilar el modelo con RMSprop como optimizador y la función de pérdida focal ajustada
optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.0001)
nn_model.compile(optimizer=optimizer,
                 loss=focal_loss(alpha=0.5, gamma=3),
                 metrics=['accuracy'])

# Entrenar el modelo de Red Neuronal
history = nn_model.fit(X_train_balanced, y_train_balanced_encoded,
                       epochs=200,
                       batch_size=64,
                       validation_split=0.2,
                       callbacks=[
                           ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=0.00001),
                           EarlyStopping(monitor='val_accuracy', patience=20, restore_best_weights=True)
                       ])

# Evaluar el modelo de Random Forest
rf_pred = rf_model.predict(X_test)

# Evaluar el modelo de Red Neuronal
nn_pred = np.argmax(nn_model.predict(X_test), axis=1)

# Ensemble de modelos mediante voto mayoritario
final_pred = np.array([np.bincount([rf_pred[i], nn_pred[i]]).argmax() for i in range(len(rf_pred))])

# Matriz de Confusión y Reporte de Clasificación
print(confusion_matrix(np.argmax(y_test, axis=1), final_pred))
print(classification_report(np.argmax(y_test, axis=1), final_pred))

# Graficar los resultados de la Red Neuronal
plt.figure(figsize=(12, 4))
```

```python
# Graficar la pérdida
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Time')

# Graficar la precisión
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Over Time')

plt.show()
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
import numpy as np
from imblearn.over_sampling import ADASYN
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
from tensorflow.keras import backend as K
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Definir la función de pérdida focal con parámetros ajustados
def focal_loss(gamma=3., alpha=0.5):
    def focal_loss_fixed(y_true, y_pred):
        y_true = K.cast(y_true, tf.float32)
        y_pred = K.cast(y_pred, tf.float32)
        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1. - epsilon)
        cross_entropy = -y_true * K.log(y_pred)
        loss = alpha * K.pow(1 - y_pred, gamma) * cross_entropy
        return K.sum(loss, axis=1)
    return focal_loss_fixed

# Supongamos que 'df_comprobar_scaled' es tu DataFrame de características escaladas
# y 'columnas_de_interes' son las columnas de características que vas a usar
columnas_de_interes = ['Sodioppm', 'Potasioppm', 'ParticulasFerrosas_PQ_', 'Fierroppm', 'Cobreppm', 'Plomoppm', 'Estañoppm',
                       'Vanadioppm', 'Cadmioppm', 'Cromoppm', 'Platappm', 'Titanioppm', 'Silicioppm',
                       'Barioppm', 'Antimonioppm', 'Niquelppm', 'Aluminioppm', 'Boroppm', 'Estañoppm.1',
                       'SulfataciónABS_cm', 'Viscosidada100_CcSt', 'HollinABS_01mm',
                       'NitraciónABS_cm', 'OxidaciónABS_cm', 'TBN_mgKOH_g_', 'Agua_', 'Diesel_', 'Turbidez_NTU_', 'idcomponente']

X = df[columnas_de_interes]
y = df['condicion_smb.1']

# Lista de columnas numéricas y categóricas
numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = ['idcomponente']
```

```python
# Crear el transformador de columnas
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='mean')),
            ('scaler', StandardScaler())
        ]), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

# Crear el pipeline para el preprocesamiento y el balanceo de datos
pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ('under', RandomUnderSampler(sampling_strategy='auto', random_state=42)),
    ('over', ADASYN(random_state=42))
])

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

# Ajustar y transformar los datos de entrenamiento
X_train_balanced, y_train_balanced = pipeline.fit_resample(X_train, y_train)

# Preprocesar los datos de prueba
X_test_preprocessed = preprocessor.transform(X_test)

# Convertir y_train_balanced a un array de NumPy antes de aplicar reshape
y_train_balanced_np = y_train_balanced.to_numpy()

# Codificar las etiquetas después de aplicar ADASYN
encoder = OneHotEncoder(sparse=False)
y_train_balanced_encoded = encoder.fit_transform(y_train_balanced_np.reshape(-1, 1))
y_test_encoded = encoder.transform(y_test.to_numpy().reshape(-1, 1))

# Verificar la distribución de clases en los conjuntos de entrenamiento y prueba
train_class_distribution = pd.Series(y_train_balanced).value_counts(normalize=True)
test_class_distribution = pd.Series(y_test).value_counts(normalize=True)

print("Distribución de clases en el conjunto de entrenamiento después de ADASYN:\n", train_class_distribution)
print("Distribución de clases en el conjunto de prueba:\n", test_class_distribution)
```

```python
# Ajuste manual de parámetros para Random Forest
rf_model = RandomForestClassifier(
    n_estimators=30,
    max_depth=30,
    min_samples_split=5,
    min_samples_leaf=5,
    random_state=42
)

# Entrenar el modelo de Random Forest
rf_model.fit(X_train_balanced, y_train_balanced)

# Definir el modelo de Red Neuronal
nn_model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(90, activation='relu', input_shape=(X_train_balanced.shape[1],), kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(30, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(y_train_balanced_encoded.shape[1], activation='softmax')
])

# Compilar el modelo con RMSprop como optimizador y la función de pérdida focal ajustada
optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.0001)
nn_model.compile(optimizer=optimizer,
                 loss=focal_loss(alpha=0.5, gamma=3),
                 metrics=['accuracy'])

# Entrenar el modelo de Red Neuronal
history = nn_model.fit(X_train_balanced, y_train_balanced_encoded,
                       epochs=200,
                       batch_size=64,
                       validation_split=0.2,
                       callbacks=[
                           ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=0.00001),
                           EarlyStopping(monitor='val_accuracy', patience=20, restore_best_weights=True)
                       ])

# Evaluar el modelo de Random Forest
rf_pred = rf_model.predict(X_test_preprocessed)

# Evaluar el modelo de Red Neuronal
nn_pred = np.argmax(nn_model.predict(X_test_preprocessed), axis=1)

# Mapear las etiquetas de clase a enteros solo para rf_pred
class_map = {label: idx for idx, label in enumerate(encoder.categories_[0])}
rf_pred_mapped = np.array([class_map[label] for label in rf_pred])
```

```python
# Ensemble de modelos mediante voto mayoritario
final_pred = np.array([np.bincount([rf_pred_mapped[i], nn_pred[i]]).argmax() for i in range(len(rf_pred_mapped))])

# Matriz de Confusión y Reporte de Clasificación
print(confusion_matrix(np.argmax(y_test_encoded, axis=1), final_pred))
print(classification_report(np.argmax(y_test_encoded, axis=1), final_pred))

# Graficar los resultados de la Red Neuronal
plt.figure(figsize=(12, 4))

# Graficar la pérdida
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Time')

# Graficar la precisión
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Over Time')

plt.show()
```

```python
test_loss, test_accuracy = nn_model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```python
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Suponiendo que ya has entrenado tu modelo y tienes X_test y y_test

# Predecir las clases del conjunto de prueba
y_pred = nn_model.predict(X_test)

# Convertir las predicciones a etiquetas de clase]
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)

# Generar la matriz de confusión
conf_matrix = confusion_matrix(y_test_classes, y_pred_classes)
print("Confusion Matrix:")
print(conf_matrix)

# Generar el informe de clasificación
class_report = classification_report(y_test_classes, y_pred_classes)
print("Classification Report:")
print(class_report)

# Visualizar la matriz de confusión usando seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=encoder.categories_[0], yticklabels=encoder.categories_[0])
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title('Confusion Matrix')
plt.show()
```

```python
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
import numpy as np
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras import backend as K
from itertools import cycle
from sklearn.preprocessing import label_binarize

# Definir la función de pérdida focal
def focal_loss(gamma=2., alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        y_true = K.cast(y_true, tf.float32)
        y_pred = K.cast(y_pred, tf.float32)
        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1. - epsilon)
        cross_entropy = -y_true * K.log(y_pred)
        loss = alpha * K.pow(1 - y_pred, gamma) * cross_entropy
        return K.sum(loss, axis=1)
    return focal_loss_fixed

# Supongamos que 'df_comprobar_scaled' es tu DataFrame de características escaladas
# y 'columnas_de_interes' son las columnas de características que vas a usar
X = df[columnas_de_interes]
y = df['condicion_smb.1']

# Imputación de valores faltantes por la media
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Normalizar las características
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Codificar las etiquetas usando OneHotEncoder
encoder = OneHotEncoder(sparse=False)
y_encoded = encoder.fit_transform(y.values.reshape(-1, 1))

# Nombres de las clases
class_names = encoder.categories_[0]
```

```python
# Configurar KFold para validación cruzada
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Función para crear el modelo de red neuronal
def create_nn_model(input_shape):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(128, activation='relu', input_shape=(input_shape,), kernel_regularizer=tf.keras.regularizers.l2(0.01)),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(64, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(y_encoded.shape[1], activation='softmax')
    ])
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
    model.compile(optimizer=optimizer,
                  loss=focal_loss(alpha=0.25, gamma=2),
                  metrics=['accuracy'])
    return model

# Almacenar resultados
accuracies = []
confusion_matrices = []
all_y_true = []
all_y_pred = []
all_y_score = []
history_list = []

for train_index, val_index in kf.split(X):
    X_train, X_val = X[train_index], X[val_index]
    y_train, y_val = y_encoded[train_index], y_encoded[val_index]

    # Decodificar las etiquetas para aplicar submuestreo y SMOTE
    y_train_decoded = np.argmax(y_train, axis=1)

    # Ajustar el submuestreo de la clase mayoritaria
    rus = RandomUnderSampler(sampling_strategy={0: int(np.sum(y_train_decoded == 0) * 0.5),
                                                1: int(np.sum(y_train_decoded == 1) * 1.0),
                                                2: int(np.sum(y_train_decoded == 2) * 1.0)},
                             random_state=42)
    X_train_res, y_train_res = rus.fit_resample(X_train, y_train_decoded)

    # Aplicar SMOTE para balancear las clases en el conjunto de entrenamiento
    smote = SMOTE(random_state=42, k_neighbors=5)
    X_train_balanced, y_train_balanced = smote.fit_resample(X_train_res, y_train_res)

    # Codificar las etiquetas después de aplicar SMOTE
    y_train_balanced_encoded = encoder.fit_transform(y_train_balanced.reshape(-1, 1))
```

```python
    # Crear y entrenar el modelo de Red Neuronal
    nn_model = create_nn_model(X_train.shape[1])
    history = nn_model.fit(X_train_balanced, y_train_balanced_encoded,
                           epochs=100,
                           batch_size=32,
                           validation_data=(X_val, y_val),
                           callbacks=[
                               ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=0.00001),
                               EarlyStopping(monitor='val_accuracy', patience=20, restore_best_weights=True)
                           ],
                           verbose=1)

    # Almacenar el historial de entrenamiento
    history_list.append(history.history)

    # Evaluar el modelo de Red Neuronal
    y_val_pred = np.argmax(nn_model.predict(X_val), axis=1)
    y_val_true = np.argmax(y_val, axis=1)
    y_val_score = nn_model.predict(X_val)

    accuracy = accuracy_score(y_val_true, y_val_pred)
    cm = confusion_matrix(y_val_true, y_val_pred)

    accuracies.append(accuracy)
    confusion_matrices.append(cm)
    all_y_true.append(y_val_true)
    all_y_pred.append(y_val_pred)
    all_y_score.append(y_val_score)

# Encontrar la longitud mínima de las épocas en todos los historiales
min_length = min([len(h['loss']) for h in history_list])

# Truncar los historiales a la longitud mínima
history_list = [{key: val[:min_length] for key, val in h.items()} for h in history_list]

# Promedio de las métricas
mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

print(f"Mean Accuracy: {mean_accuracy:.4f}")
print(f"Standard Deviation of Accuracy: {std_accuracy:.4f}")

# Sumar todas las matrices de confusión
total_cm = np.sum(confusion_matrices, axis=0)
print("Total Confusion Matrix:")
print(total_cm)
```

```python
# Concatenar todas las predicciones y valores verdaderos
all_y_true = np.concatenate(all_y_true)
all_y_pred = np.concatenate(all_y_pred)
all_y_score = np.concatenate(all_y_score)

# Reporte de clasificación general
target_names = [str(i) for i in class_names]
print("Classification Report:")
print(classification_report(all_y_true, all_y_pred, target_names=target_names))

# Binarizar las etiquetas para ROC
y_true_binarized = label_binarize(all_y_true, classes=[0, 1, 2])

# Calcular FPR, TPR, y AUC para cada clase
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(len(target_names)):
    fpr[i], tpr[i], _ = roc_curve(y_true_binarized[:, i], all_y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
```

```python
import matplotlib.pyplot as plt

# Graficar la pérdida
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Time')

# Graficar la precisión
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Over Time')

plt.show()
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# Calcular curvas ROC y AUC para cada clase
n_classes = len(np.unique(y_test))
y_test_bin = tf.keras.utils.to_categorical(y_test, num_classes=n_classes)

fpr = {}
tpr = {}
roc_auc = {}

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Graficar todas las curvas ROC
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'ROC curve of class {i} (area = {roc_auc[i]:0.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```python
accuracy = np.mean(y_pred_classes == y_test)
print(f'Overall accuracy: {accuracy:.2f}')
```

```python
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt

# Binarizar las etiquetas si aún no lo has hecho
y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

# Calcular ROC curve y ROC area para cada clase
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Graficar todas las curvas ROC
plt.figure(figsize=(8, 6))
colors = ['blue', 'red', 'green']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa Falso Positivo')
plt.ylabel('Tasa Verdadero positivo')
plt.title('Característica operativa del receptor')
plt.legend(loc="lower right")
plt.show()
```

```python
import numpy as np
import tensorflow as tf
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Suponiendo que tienes tus datos X y y
# X = ...
# y = ...

# Configuración de KFold
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Normalizar los datos
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Definir una función para crear el modelo
def create_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(128, activation='relu', input_shape=(X.shape[1],)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

# Almacenar las puntuaciones de precisión para cada pliegue
accuracies = []

# Validación cruzada
for train_index, val_index in kf.split(X):
    X_train, X_val = X[train_index], X[val_index]
    y_train, y_val = y[train_index], y[val_index]

    model = create_model()

    # Entrenamiento del modelo
    model.fit(X_train, y_train,
              epochs=50,
              batch_size=32,
              validation_data=(X_val, y_val),
              class_weight=class_weight_dict,
              verbose=0)  # verbose=0 para no imprimir el progreso de cada pliegue

    # Evaluar el modelo
    y_val_pred = np.argmax(model.predict(X_val), axis=1)
    accuracy = accuracy_score(y_val, y_val_pred)
    accuracies.append(accuracy)

# Resultados
mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

print(f"Mean accuracy: {mean_accuracy}")
print(f"Standard deviation: {std_accuracy}")
```

```python
from tensorflow.keras.utils import to_categorical

# Convertir las etiquetas a formato one-hot
y_train = to_categorical(y_train, num_classes=3)
y_test = to_categorical(y_test, num_classes=3)
```

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

```python
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```python
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.001)

history = model.fit(X_train, y_train, epochs=100, validation_split=0.2,
                    batch_size=32, callbacks=[early_stopping, reduce_lr])
```

```python
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test accuracy:", test_accuracy)
```

```python
# Convertir y_test a un vector unidimensional
y_test_single = np.argmax(y_test, axis=1)

# Hacer predicciones sobre el conjunto de prueba
y_pred = np.argmax(model.predict(X_test), axis=1)

# Verificar el formato de y_test_single y y_pred
print(f"y_test_single shape: {y_test_single.shape}")
print(f"y_test_single: {y_test_single[:10]}")  # Mostrar las primeras 10 etiquetas para ver el formato
print(f"y_pred shape: {y_pred.shape}")
print(f"y_pred: {y_pred[:10]}")  # Mostrar las primeras 10 predicciones para ver el formato

# Generar la matriz de confusión
cm = confusion_matrix(y_test_single, y_pred)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=unique_classes)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

```python
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score, f1_score
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

# Asegúrate de que tienes tus datos X_train, y_train, X_test, y_test listos
# X_train, y_train = ...
# X_test, y_test = ...

# Verifica la forma original de y_train y y_test
print(f"Original y_train shape: {y_train.shape}")
print(f"Original y_test shape: {y_test.shape}")

# Asegúrate de que y_train y y_test son vectores unidimensionales
if y_train.ndim > 1:
    y_train = y_train.ravel()
if y_test.ndim > 1:
    y_test = y_test.ravel()

# Convertir las etiquetas a formato one-hot
y_train_one_hot = to_categorical(y_train, num_classes=3)
y_test_one_hot = to_categorical(y_test, num_classes=3)

# Convertir los datos a matrices NumPy si son DataFrames de pandas
X_train_np = X_train.values if isinstance(X_train, pd.DataFrame) else X_train
X_test_np = X_test.values if isinstance(X_test, pd.DataFrame) else X_test

# Calcular los pesos de las clases
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))
print(f"Class weights: {class_weight_dict}")

# Verificar las formas de los datos después de la conversión
print(f"X_train_np shape: {X_train_np.shape}")
print(f"y_train_one_hot shape: {y_train_one_hot.shape}")
print(f"X_test_np shape: {X_test_np.shape}")
print(f"y_test_one_hot shape: {y_test_one_hot.shape}")

# Definir el modelo como una función para poder usarlo en cada pliegue
def create_model():
```

```
    model = Sequential([
        Dense(120, activation='relu', input_shape=(X_train_np.shape[1],)),
        Dropout(0.2),
        Dense(80, activation='relu'),
        Dropout(0.2),
         Dense(150, activation='relu'),
        Dropout(0.2),
        Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

# Configuración de KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Listas para almacenar los resultados de cada pliegue
accuracies = []
precisions = []
recalls = []
f1s = []

for train_index, val_index in kf.split(X_train_np):
    X_train_fold, X_val_fold = X_train_np[train_index], X_train_np[val_index]
    y_train_fold, y_val_fold = y_train_one_hot[train_index], y_train_one_hot[val_index]

    # Verificar las formas de los pliegues de entrenamiento y validación
    print(f"X_train_fold shape: {X_train_fold.shape}")
    print(f"y_train_fold shape: {y_train_fold.shape}")
    print(f"X_val_fold shape: {X_val_fold.shape}")
    print(f"y_val_fold shape: {y_val_fold.shape}")

    model = create_model()

    early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.001)

    history = model.fit(X_train_fold, y_train_fold, epochs=100, validation_data=(X_val_fold, y_val_fold),
                        batch_size=32, callbacks=[early_stopping, reduce_lr], verbose=0, class_weight=class_weight_dict)

    y_val_pred = np.argmax(model.predict(X_val_fold), axis=1)
    y_val_true = np.argmax(y_val_fold, axis=1)

    accuracies.append(accuracy_score(y_val_true, y_val_pred))
    precisions.append(precision_score(y_val_true, y_val_pred, average='weighted'))
    recalls.append(recall_score(y_val_true, y_val_pred, average='weighted'))
    f1s.append(f1_score(y_val_true, y_val_pred, average='weighted'))

print(f'Mean Accuracy: {np.mean(accuracies)}, Std Accuracy: {np.std(accuracies)}')
print(f'Mean Precision: {np.mean(precisions)}, Std Precision: {np.std(precisions)}')
print(f'Mean Recall: {np.mean(recalls)}, Std Recall: {np.std(recalls)}')
print(f'Mean F1 Score: {np.mean(f1s)}, Std F1 Score: {np.std(f1s)}')

# Entrenar el modelo en todo el conjunto de entrenamiento
model = create_model()
history = model.fit(X_train_np, y_train_one_hot, epochs=100, validation_split=0.2,
                    batch_size=32, callbacks=[early_stopping, reduce_lr], class_weight=class_weight_dict)

# Hacer predicciones sobre el conjunto de prueba
y_test_pred = np.argmax(model.predict(X_test_np), axis=1)
y_test_true = np.argmax(y_test_one_hot, axis=1)
```

```python
# Generar la matriz de confusión
cm = confusion_matrix(y_test_true, y_test_pred)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_train))
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()

# Calcular y mostrar otras métricas
print(classification_report(y_test_true, y_test_pred, target_names=['Class 0', 'Class 1', 'Class 2']))
```

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Calcular la curva ROC y AUC para cada clase
n_classes = y_test_bin.shape[1]
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Graficar la curva ROC para cada clase
colors = ['green', 'yellow', 'red']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curva de clase {0} (area = {1:0.2f})'.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa Falso Positivo')
plt.ylabel('Tasa Verdadero Positivo')
plt.title('Caracteristica operativa del receptor')
plt.legend(loc="lower right")
plt.show()
```

```python
import numpy as np
import tensorflow as tf
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.utils.class_weight import compute_class_weight
import pandas as pd

# Suponiendo que tienes tus datos en arrays de NumPy X_train y y_train
# X_train = ...
# y_train = ...

# Convertir arrays de NumPy a DataFrames de pandas si es necesario
if isinstance(X_train, np.ndarray):
    X_train = pd.DataFrame(X_train)
if isinstance(y_train, np.ndarray):
    y_train = pd.Series(y_train)

# Asegúrate de que los índices de los DataFrames son secuenciales y comienzan desde 0
X_train = X_train.reset_index(drop=True)
y_train = y_train.reset_index(drop=True)

# Calcular los pesos de las clases
class_weights = compute_class_weight(class_weight='balanced',
                                     classes=np.unique(y_train),
                                     y=y_train)

# Crear un diccionario para pasar a Keras
class_weight_dict = dict(enumerate(class_weights))
print("Pesos de clase:", class_weight_dict)

# Definir una función para crear el modelo
def create_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```python
# Configuración de KFold
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Almacenar las puntuaciones de precisión para cada pliegue
accuracies = []
precisions = []
recalls = []
f1_scores = []

# Validación cruzada
for train_index, val_index in kf.split(X_train):
    X_train_fold, X_val_fold = X_train.iloc[train_index], X_train.iloc[val_index]
    y_train_fold, y_val_fold = y_train.iloc[train_index], y_train.iloc[val_index]

    model = create_model()

    # Entrenamiento del modelo
    history = model.fit(X_train_fold, y_train_fold,
                        epochs=50,
                        batch_size=32,
                        validation_data=(X_val_fold, y_val_fold),
                        class_weight=class_weight_dict,
                        verbose=0)  # verbose=0 para no imprimir el progreso de cada pliegue

    # Evaluar el modelo
    y_val_pred = np.argmax(model.predict(X_val_fold), axis=1)
    accuracies.append(accuracy_score(y_val_fold, y_val_pred))
    precisions.append(precision_score(y_val_fold, y_val_pred, average='weighted'))
    recalls.append(recall_score(y_val_fold, y_val_pred, average='weighted'))
    f1_scores.append(f1_score(y_val_fold, y_val_pred, average='weighted'))

# Resultados
print(f"Mean Accuracy: {np.mean(accuracies)}, Std Accuracy: {np.std(accuracies)}")
print(f"Mean Precision: {np.mean(precisions)}, Std Precision: {np.std(precisions)}")
print(f"Mean Recall: {np.mean(recalls)}, Std Recall: {np.std(recalls)}")
print(f"Mean F1 Score: {np.mean(f1_scores)}, Std F1 Score: {np.std(f1_scores)}")
```

```python
import matplotlib.pyplot as plt

def plot_learning_curve(history):
    # Curvas de precisión
    plt.figure(figsize=(12, 6))
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Learning Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid()
    plt.show()

    # Curvas de pérdida
    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Learning Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid()
    plt.show()

# Llamar a la función para el último historial de entrenamiento
plot_learning_curve(history)
```

```python
df.columns
```

```python
df['condicion_smb.1'].unique()
```

```python
df['estado'] = df['condicion_smb.1'].replace({'Critico': 1, 'Normal': 0, 'Seguimiento': 1})
```

```python
df['etiquetado_covarianza_robusta'].unique()
```

```python
y_test=df['estado']
y_pred=df['etiquetado_covarianza_robusta']
```

```python
cm = confusion_matrix(y_test, y_pred)
```

```python
print("Reporte de Clasificación:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n", cm)
```

```python
# Graficar la matriz de confusión
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```python
y_test=df['estado']
y_pred=df['anomaly_label_isolation']
```

```python
cm = confusion_matrix(y_test, y_pred)
```

```python
print("Reporte de Clasificación:\n", classification_report(y_test, y_pred))
print("Matriz de Confusión:\n",cm)
```

```python
# Graficar la matriz de confusión
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```