ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

"DETECCIÓN DE FALLAS EN CAJAS DE ENGRANAJES UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINAS SUPPORT VECTOR MACHINE (SVM)"

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR:

JOSÉ LUIS ESCOBAR CHÁVEZ

Riobamba – Ecuador

2022

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

"DETECCIÓN DE FALLAS EN CAJAS DE ENGRANAJES UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINAS SUPPORT VECTOR MACHINE (SVM)"

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR: JOSÉ LUIS ESCOBAR CHÁVEZ

DIRECTOR: Ing. FÉLIX ANTONIO GARCÍA MORA

Riobamba – Ecuador

2022

Yo, JOSÉ LUIS ESCOBAR CHÁVEZ, declaro que el presente trabajo de integración curricular es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de integración curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 25 de mayo de 2022

José Luis Escobar Chávez
060450699-8

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; tipo: Proyecto de Investigación, **DETECCIÓN DE FALLAS EN CAJAS DE ENGRANAJES UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINAS SUPPORT VECTOR MACHINE (SVM)**, realizado por el señor: **JOSÉ LUIS ESCOBAR CHÁVEZ**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal autoriza su presentación.

	FIRMA	FECHA
Ing. Marco Antonio Ordoñez Viñan **PRESIDENTE DEL TRIBUNAL**		2022-05-25
Ing. Félix Antonio García Mora **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2022-05-25
Ing. Eduardo Segundo Hernández Dávila **MIEMBRO DEL TRIBUNAL**		2022-05-25

DEDICATORIA

El presente trabajo de integración curricular se lo dedico a mis padres Joaquín Escobar y Zoila Leonor por su trabajo y sacrificio, quienes a lo largo de mi formación académica me han acompañado y apoyado incondicionalmente no solo económicamente sino también con consejos y palabras de aliento inspirándome a salir adelante, a no rendirme y a cumplir todos mis objetivos. A mis hermanos y hermanas por acompañarme y brindarme su apoyo moral y confianza en todo momento. En general a todos mis familiares y amigos por creer en mí y extender sus manos en los momentos más difíciles de mi formación académica.

José

AGRADECIMIENTO

Agradezco a la escuela Superior Politécnica de Chimborazo en especial a la carrera de Mantenimiento Industrial y a sus docentes que han sido parte fundamental de mi formación profesional, en especial al Ing. Félix García director del trabajo de integración curricular y al Ing. Eduardo Hernández miembro del mismo quienes me han acompañado, guiado y brindado su ayuda a lo largo del desarrollo de este trabajo de integración curricular, haciendo que se desenvuelva de la mejor manera posible.

José

TABLA DE CONTENIDO

CAPÍTULO II

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ABREVIATURAS

SVM	Máquina de vectores de soporte
IA	Inteligencia artificial
ML	Aprendizaje de máquinas
IFD	Detección inteligente de fallas
RFECV	Eliminación recursiva de características con validación cruzada
ROC	Curva de característica operativa del receptor
AUC	Área bajo la curva

ÍNDICE DE ANEXOS

RESUMEN

El objetivo de esta investigación fue crear un modelo predictivo bajo el enfoque de aprendizaje de máquinas y verificar su efectividad para clasificar y detectar fallas en cajas de engranajes de manera automática, para lo cual se utilizó un conjunto de datos de señales de vibración obtenido del repositorio de Iniciativa de Datos de Energía Abierta (OEDI) del departamento de energía de EE. UU. La creación del modelo se llevó a cabo utilizando el método de aprendizaje de máquinas supervisado Support Vector Machine (SVM) y con la ayuda del software de programación Python, donde se realizó el preprocesamiento y análisis del conjunto de datos. Al conjunto de datos se le extrajo características en el dominio del tiempo y dominio de la frecuencia. Para seleccionar las mejores características se aplicó el método de Eliminación Recursiva de Características con Validación Cruzada (RFECV). Para ingresar al clasificador SVM los datos se dividieron en 70% para entrenamiento y 30% para prueba. Como resultado se obtuvo tres modelos de detección de fallas, un primer modelo donde se utilizó un conjunto de datos recopilados por cuatro acelerómetros bajo una carga de 50%, un segundo modelo donde se combinó los datos recopilados por cuatro acelerómetros y cargas en un rango de 0 a 90% y un tercer modelo utilizando los datos de un solo acelerómetro del modelo dos. Cada modelo se entrenó y probo obteniéndose excelentes resultados, logrando una exactitud de 99,84% y una precisión de 99,82% para el mejor modelo. Los resultados demuestran que el método empleado clasifica y predice fallas con alta exactitud y precisión, siendo un método prometedor y de gran aporte para el mantenimiento industrial. Se recomienda reducir y estandarizar el conjunto de características, de esa forma se consigue reducir la carga computacional y a su vez mejorar el rendimiento del modelo.

Palabras clave: <MODELO PREDICTIVO> <CAJAS DE ENGRANAJES> <DETECCIÓN DE FALLAS> <APRENDIZAJE DE MÁQUINAS> <MÁQUINA DE VECTORES DE SOPORTE>.

SUMMARY

The objective of this research was to create a predictive model under the machine learning approach and verify its effectiveness to classify and detect faults in gearboxes automatically, for which a data set of vibration signals obtained from the repository was used from the Open Energy Data Initiative (OEDI) of the US Department of Energy. The creation of the model was carried out using the Support Vector Machine (SVM) supervised machine learning method and with the aid of Python programming software, where the preprocessing and analysis of the data set was performed. Features in the time domain and frequency domain were extracted from the data set. To select the best features, the Recursive Features Elimination with Cross Validation (RFECV) method was applied. To enter the SVM classifier, the data was divided into 70% for training and 30% for testing. As a result, three fault detection models were obtained, a first model where a set of data collected by four accelerometers under a load of 50% was produced, a second model where the data collected by four accelerometers and loads in a range of 0 to 90% and a third model using the data from a single accelerometer of model two. Each model was trained and tested obtaining excellent results, achieving an accuracy of 99,84% and a precision of 99,82% for the best model. The results show that the method used classifies and predicts faults with high accuracy and precision, being a promising method and of great contribution to industrial maintenance. It is recommended to reduce and standardize the set of features, in this way it is possible to reduce the computational load and in turn improve the performance of the model.

Keywords: <PREDICTIVE MODEL> <GEARBOXES> <FAULT DETECTION> <MACHINE LEARNING> <MACHINE SUPPORT VECTORS>.

Sandra Paulina Porras Pumalema
C.I.0603357062

INTRODUCCIÓN

En la industria moderna se espera que las máquinas funcionen correctamente y sin interrupciones para lograr los objetivos de producción. Las cajas de engranajes se utilizan extensamente en diversas aplicaciones industriales para transmitir potencia mecánica. Los engranajes son parte importante de diferentes tipos de máquinas rotativas en la industria y su correcto funcionamiento asegura la disponibilidad de todo un sistema. Si una falla no se detecta a tiempo las consecuencias pueden ser graves y costosas. El diagnóstico mediante análisis de señales de vibración es una técnica de monitoreo de la condición que se ha aplicado ampliamente en la industria para detectar fallas en cajas de engranajes y otros sistemas mecánicos. El proceso de análisis de señales de vibración requiere que se realice un reconocimiento de patrones que muestre una relación entre los datos de monitoreo y la condición de la máquina, tradicionalmente este paso se realiza de forma manual lo que incrementa el trabajo, disminuye la precisión del diagnóstico y requiere del conocimiento especializado de expertos. Por estos motivos, varias industrias prefieren contar con métodos que realicen dicho proceso de manera automática.

En el área de la inteligencia artificial, el Aprendizaje de Máquinas (ML) se ha convertido en una herramienta muy importante en el desarrollo de algoritmos predictivos inteligentes y su aplicación en diversas áreas como, mantenimiento, control y optimización están dando excelentes resultados. Los enfoques de ML tienen la capacidad de identificar patrones a partir de datos históricos disponibles y crear modelos predictivos capaces de realizar un diagnóstico mucho más ágil y preciso acerca del estado de salud de una máquina. La clasificación ha ganado mucha popularidad en los últimos años y la Máquina de Vectores de Soporte (SVM) es una de las técnicas de ML más representativas que se utilizan para este propósito, debido su alta precisión y buena capacidad de generalización.

En el presente trabajo de integración curricular se propone un método para clasificar y predecir fallas de manera automática utilizando el enfoque de ML. Específicamente se utilizará el algoritmo de aprendizaje de máquinas supervisado SVM para construir un modelo predictivo que clasifique y detecte fallas a partir de datos de señales de vibración de manera automática. Para lograr dicho fin se utilizará una base de datos de señales de vibración de caja de engranajes obtenida de un repositorio y el software de programación Python. A partir de los datos se creará y entrenará un modelo predictivo lo más exacto y preciso posible, el cual deberá ser capaz de generalizar correctamente datos nuevos y predecir de manera automática si estos presentan o no un fallo.

Justificación.

Con la llegada de la industria 4.0, las máquinas industriales cada vez son más inteligentes, complejas y a su vez de costosa reparación cuando no se lleva un adecuado control de mantenimiento, por lo tanto, surge la necesidad de investigar y crear nuevas metodologías basadas en datos y técnicas de monitoreo que permitan lograr un diagnóstico confiable, preciso y ágil sobre el estado de las máquinas de manera automática. La caja de engranajes es una parte importante de las máquinas rotativas en cualquier industria, cuando se producen fallas inesperadas en componentes importantes de esta se generan pérdidas indeseables, por este motivo es importante realizar una detección temprana y oportuna para asegurar su correcto funcionamiento y evitar el tiempo de inactividad provocado por el paro no planificado de las máquinas (Chen et al., 2019, p. 1).

Mediante la necesidad de un monitoreo continuo del estado de las máquinas y la disposición de grandes cantidades de datos industriales, los enfoques de aprendizaje de máquinas permiten realizar un diagnóstico automático de manera ágil e inteligente acerca del estado de salud de las máquinas. Además, disponer una de estrategia de mantenimiento acompañada del aprendizaje de máquinas permitirá obtener beneficios en términos de seguridad, eficiencia, optimización y toma de decisiones. Los métodos comunes de diagnóstico de fallas requieren tiempo y la necesidad de contar con personal experto y calificado, en comparación los modelos predictivos creados a partir de métodos de aprendizaje de máquinas tienen la capacidad de realizar un diagnóstico ágil y automático acerca de la condición de una máquina permitiendo que el personal con poca experiencia pueda ser capaz de tomar decisiones en el área de mantenimiento sin la necesidad de un especialista.

En el Ecuador se han realizado varias investigaciones empleando métodos de aprendizaje de máquinas en diversas aplicaciones, sin embargo, son pocas las investigaciones que se enfocan específicamente al mantenimiento industrial. En la Carrera de Mantenimiento Industrial de la Facultad de Mecánica de la Escuela Superior Politécnica de Chimborazo, no existen trabajos de investigación relacionados con la aplicación de métodos de aprendizaje de máquinas enfocados al mantenimiento industrial, por lo que en el presente trabajo de integración curricular se propone una idea innovadora y moderna encaminada a mejorar la calidad del mantenimiento industrial, mediante el desarrollo de un modelo inteligente que será capaz de detectar fallas de manera ágil, confiable y automática aprovechando las nuevas tecnologías y enormes cantidades de datos disponibles, así mismo se busca extender el conocimiento y dar a conocer la importancia de estas nuevas técnicas de mantenimiento que cada vez van tomando más fuerza y serán el futuro del mantenimiento.

Problema.

Las cajas de engranajes desempeñan un papel fundamental en sistemas de transmisión mecánica, siendo utilizas ampliamente en máquinas rotativas y otras aplicaciones a nivel industrial. Funcionan durante largos periodos de tiempo y bajo diferentes condiciones de operación. Un diagnóstico incorrecto o deficiente de su estado puede traer como consecuencias perdidas de rendimiento, indisponibilidad y costos elevados de mantenimiento. El diagnóstico mediante señales de vibración es uno de los métodos más utilizados para la detección de fallas en cajas de engranajes, sin embargo, uno de los problemas que se presentan durante el diagnóstico y reconocimiento de patrones es que se realiza manualmente, lo que aumenta el trabajo, reduce la precisión del diagnóstico y requiere del conocimiento especializado de expertos, por lo tanto, surge la necesidad de disponer de técnicas de diagnóstico automáticas, confiables, de alta calidad y que brinden una mejor precisión de diagnóstico y detección de fallas.

Hipótesis.

Utilizando el método de aprendizaje de máquinas support vector machine se detectan fallas en cajas de engranajes.

Variable dependiente.

Detección de fallas.

Variable independiente.

- Support Vector Machine (SVM).
- Matriz de confusión.
- Exactitud.
- Precisión.

Objetivos.

Objetivo general.

Detectar fallas en cajas de engranajes utilizando el método de aprendizaje de máquinas support vector machine (SVM).

Objetivos específicos.

Procesar la base de datos del repositorio de Iniciativa de Datos de Energía Abierta (OEDI) del departamento de energía de EE. UU.

Dividir la base de datos de señales de vibración de una caja de engranajes en datos de entrenamiento y datos de prueba.

Encontrar las características de extracción que mejor correlacionan con los datos de fallas.

Aplicar el algoritmo de clasificación support vector machine a los datos de fallas en cajas de engranajes.

Verificar la efectividad del método empleado para detectar fallos en cajas de engranajes.

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL

1.1. Cajas de engranajes

Las cajas de engranajes son una de las partes más importantes y críticas de cualquier máquina rotativa y otros tipos de máquinas a nivel industrial, se utilizan extensamente en diferentes sistemas mecánicos con el fin de transmitir potencia mecánica entre ejes y hacer funcionar otros sistemas acoplados. En la mayoría de industrias se espera que estos sistemas operen correctamente durante todas las horas del día para lograr los objetivos de producción, de lo contrario cualquier tipo de falla puede generar, indisponibilidad, costos elevados de reparación y accidentes en el peor de los casos, por lo que su alto rendimiento y fiabilidad es de suma importancia para muchas aplicaciones industriales (Bhadane y Ramachandran, 2017: p. 1).

Figura 1-1: Caja de engranajes

Fuente: (Tercesa, 2017)

1.2. Fallas en cajas de engranajes

Las fallas que pueden aparecer en una caja de engranajes generalmente son debido a la aplicación de cargas elevadas, mala lubricación, desalineación, altas velocidades, fatiga, mala instalación o incluso errores de fabricación. Comúnmente aparecen en los dientes y muy pocas veces en el cuerpo del engranaje. Las fallas más comunes que se presentan en los dientes de los engranajes son, agrietamiento, rayadura, picadura y rotura de diente, de las cuales la falla por rotura de diente es la más riesgosa. Un engranaje con falla de diente roto es el resultado final de una grieta que ha comenzado en la base y se ha propagado por todo el cuerpo causando la rotura total o parcial del diente. Cuando esta falla aparece el funcionamiento de los dientes continuos también es afectado, debido a la carga adicional resultante (Mohammed y Rantatalo, 2020: p. 4).

1.3. Detección de fallas en cajas de engranajes

El estado de salud de una caja de engranajes puede ser determinado por medio de técnicas de monitoreo tales como, ensayos tribológicos, vibraciones, corriente eléctrica, entre otras, de las cuales la técnica de monitoreo mediante análisis de vibraciones es la más conocida y empleada, debido a su capacidad para identificar con alta exactitud alrededor del 90% de fallas, lo que permite tomar decisiones de mantenimiento con suficiente antelación (Chingal et al., 2019).

1.3.1. Detección de fallas mediante análisis de vibraciones

El análisis de vibraciones es una técnica que permite verificar el estado de un sistema mecánico, mediante la identificación de cualquier variación en la señal de vibración causada por la degradación. El análisis de vibraciones consta de tres etapas que son, adquisición de datos, procesamiento e identificación de patrones de falla. La adquisición de datos puede ser mediante experimentos, modelos dinámicos o simulación. El preprocesamiento consiste en extraer características de las señales, las cuales son analizadas mediante el reconocimiento de patrones para evaluar la condición de las máquinas (Mohammed y Rantatalo, 2020: p. 3).

1.3.2. Detección de fallas tradicional

Tradicionalmente, la detección de fallas se realiza inspeccionando manualmente la condición de una máquina, lo que aumenta el trabajo y reduce la precisión del diagnóstico. Los métodos avanzados de preprocesamiento de señales permiten identificar fallas y el lugar de la máquina donde estas ocurren, sin embargo, estos métodos requieren de la experiencia y conocimiento experto de un especialista de lo cual carecen los usuarios de máquinas. Por lo tanto, varias industrias prefieren métodos que realicen una detección automática de fallas. Actualmente es posible lograr dicho propósito, mediante el uso de algoritmos de aprendizaje de máquinas que permiten identificar el estado de salud de una máquina de manera automática (Lei et al., 2020: p. 4).

1.3.3. Detección Inteligente de Fallas (IFD)

La detección inteligente de fallas se refiere a la aplicación de métodos de aprendizaje de máquinas, los cuales permiten crear modelos predictivos, mediante el uso de algoritmos inteligentes que son capaces de identificar el estado de salud de una máquina de forma automática en lugar de utilizar la experiencia y alto conocimiento requerido por los expertos. Estos algoritmos aprenden de forma adaptativa el conocimiento de diagnóstico de las máquinas a partir de los datos (Lei et al., 2020: p. 2).

1.4. Inteligencia artificial (IA)

La inteligencia artificial es un subconjunto de las ciencias de la computación que estudia el diseño y creación de máquinas capaces de tomar decisiones y solucionar problemas de manera autónoma, basándose en el comportamiento y conducta del pensamiento humano (Morales, 2017, p. 1).

La inteligencia artificial se utiliza en una amplia variedad de funciones como, filtrado de correo electrónico, marketing, búsqueda, servicios al cliente, ingeniería, distribución digital, producción de video, desarrollo de juegos, creación de noticias, informes financieros, reconocimiento de voz, reconocimiento de rostros, reconocimiento de iris, lenguaje natural, clasificación de contenido, entre otras (Waymond, 2020, p. 1).

1.5. Machine Learning (ML)

Machine learning o aprendizaje de máquinas es una rama de la inteligencia artificial, a partir del cual se crean modelos matemáticos basados en datos de muestra o datos de entrenamientos para tomar decisiones o realizar predicciones sin estar programados explícitamente para realizar dicha acción (Zhang, 2020, p. 223).

El aprendizaje de máquinas responde a la pregunta de cómo crear un programa de computadora a partir de datos históricos, para solucionar un problema dado y mejorar automáticamente el rendimiento del programa en base a la experiencia. Se utiliza en muchas aplicaciones industriales y de uso cotidiano, por ejemplo, los vehículos autónomos utilizan algoritmos que les permiten navegar e interactuar con su entorno, así como tomar decisiones sin la necesidad de la intervención humana, en la detección de transacciones bancarias fraudulentas, estudios neurobiológicos, sistemas de filtrado de información que aprenden de la preferencia de lectura del usuario, entre otras (Shobha y Rangaswamy, 2018: p. 197).

Figura 2-1: Aplicaciones de ML

Fuente: (Ulma, 2020)

1.6. Algoritmos de machine learning

Visto desde un enfoque computacional, los algoritmos básicamente son instrucciones que han sido programadas de forma explícita para ser utilizadas por las computadoras con el fin de resolver o calcular un problema. En comparación los algoritmos de aprendizaje de máquinas permiten que las computadoras construyan modelos a partir de datos históricos disponibles y automaticen el proceso de toma de decisiones basándose en la entrada de datos y la experiencia (Shobha y Rangaswamy, 2018: p. 197).

1.7. Técnicas de machine learning

El aprendizaje de máquinas consta de tres técnicas principales que son, aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Cada técnica tiene un método distinto de entrenamiento y aprendizaje y a su vez cada una se divide en algoritmos de clasificación, regresión y recompensa dependiendo de la función que deba realizar al momento de su aplicación (Rodríguez, 2020).

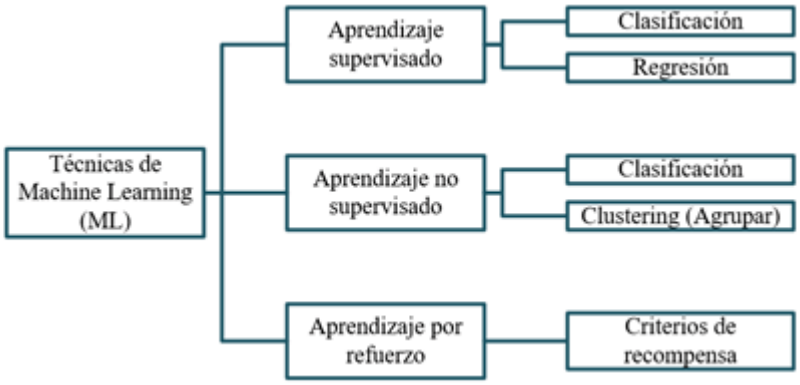


Gráfico 1-1: Técnicas de ML

Fuente: (Rodríguez, 2020)

Realizado por: Escobar, José, 2021

1.7.1. Aprendizaje supervisado

Son algoritmos de ML que toman como entrada un conjunto de datos conocidos, denominados también como datos de entrenamiento los cuales incluyen entradas y salidas. Las salidas se denominan etiquetas o valores de respuesta de las entradas. El objetivo es aprender un mapeo o modelar una asociación entre los datos de entrada y sus etiquetas para crear un modelo predictivo que pueda ser utilizado en un futuro para predecir la respuesta de salida para una nueva muestra de datos de entrada nunca antes vistos, gracias al conocimiento adquirido previamente en el proceso de relación entre las entradas y salidas (Sarkar et al., 2018: p. 35).

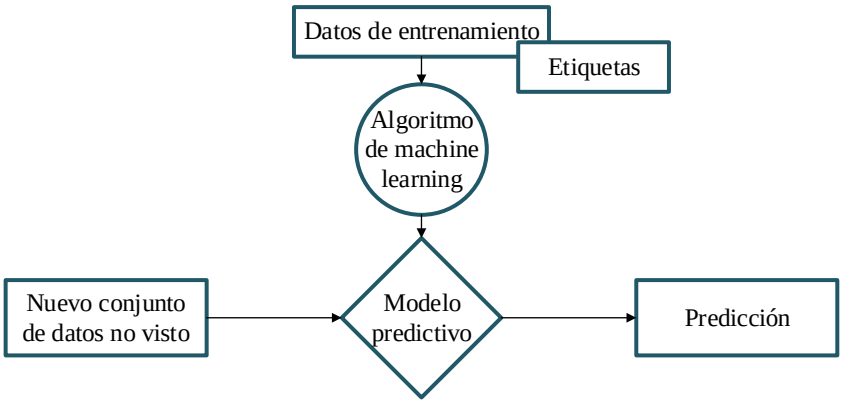


Gráfico 2-1: Diagrama de flujo del aprendizaje supervisado

Fuente: (Velásquez, 2020)

Realizado por: Escobar, José, 2021

En el aprendizaje supervisado se crean modelos a partir de algoritmos de clasificación y regresión, estos algoritmos incluyen la regresión lineal para problemas de regresión, SVM para clasificación y bosques aleatorios para problemas de regresión y clasificación (Waymond, 2020, p. 19).

1.7.2. Aprendizaje no supervisado

El aprendizaje no supervisado se utiliza cuando los datos de entrenamiento no están etiquetados, es decir, no se conoce la salida deseada, pero aun así se desea encontrar patrones o extraer información del conjunto de datos. Este tipo de aprendizaje se enfoca más en obtener información de los datos en lugar de tratar de predecir algún resultado, identificando patrones que se puedan agrupar en eventos o clases específicas. Se utiliza en tareas de agrupación, detección de anomalías y reducción de dimensionalidad (Sarkar et al., 2018: p. 38).

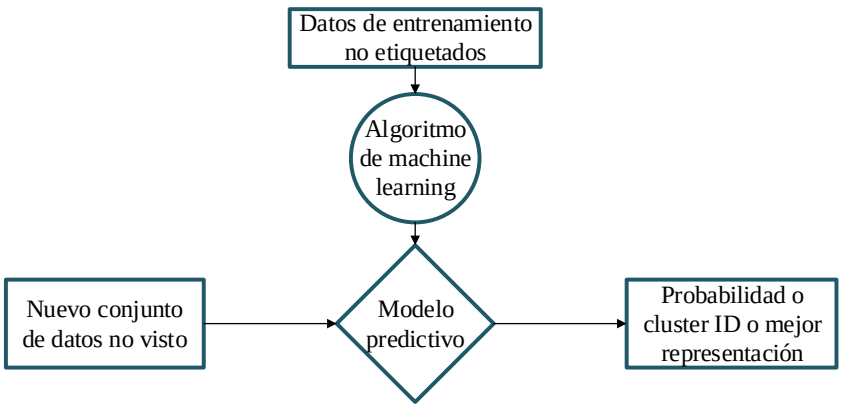


Gráfico 3-1: Diagrama de flujo del aprendizaje no supervisado

Fuente: (Velásquez, 2020)

Realizado por: Escobar, José, 2021

1.7.3. Aprendizaje por refuerzo

Este tipo de aprendizaje analiza el comportamiento y acciones de un agente autónomo en su entorno. El agente es recompensado en función de las acciones que realiza en su entorno, considera las consecuencias que pueden traer sus acciones y toma decisiones óptimas para evitarlas. Algunos ejemplos de este tipo de aprendizaje son, un programa de computadora jugando ajedrez con una persona y un programa para conducir un automóvil que tendrá que interactuar constantemente con su entorno para cumplir un determinado objetivo (Shobha y Rangaswamy, 2018: p. 200).

Figura 3-1: Aprendizaje por refuerzo

Fuente: (Gonzalez, 2018)

1.8. Support Vector Machine (SVM)

SVM es un algoritmo que forma parte de la técnica de aprendizaje de máquinas supervisado, el cual está diseñado para realizar tareas de clasificación y regresión a través del análisis de datos y búsqueda de patrones ocultos o visibles. SVM ejecuta clasificaciones lineales, sin embargo, cuando las clases no son linealmente separables el algoritmo opera proyectando los datos de entrenamiento a espacios de mayor dimensión donde estos pueden ser linealmente separables en clases (Al-Zoubi et al., 2018: p. 3).

1.8.1. Principio de funcionamiento de SVM

Si al algoritmo SVM se le proporciona un conjunto de muestras de entrenamiento cuyas etiquetas son conocidas y pertenecen a dos clases, el algoritmo entrenará un modelo que será capaz de clasificar cualquier muestra nueva a la clase correspondiente a la que pertenezca. En primer lugar, las muestras etiquetadas serán proyectadas a un espacio vectorial. Luego, el algoritmo tratará de separar de la mejor forma las muestras proyectadas de tal manera que exista una distancia máxima entre estas. Cuando se ingrese una nueva muestra al modelo entrenado, esta será proyecta en el espacio vectorial y la clase a la que pertenece será predicha dependiendo del lado del hiperplano en el que se encuentre. Durante el entrenamiento el algoritmo determina distintos hiperplanos

posibles hasta encontrar un hiperplano óptimo, el cual debe separar las clases y maximizar el margen entre estas. Los puntos proyectados en el espacio vectorial que están cerca al hiperplano se denominan vectores de soporte. Los vectores de soporte son coordenadas en el espacio vectorial, a partir de los cuales el algoritmo encuentra el hiperplano óptimo de separación y separa las clases (Shobha y Rangaswamy, 2018: p. 217).

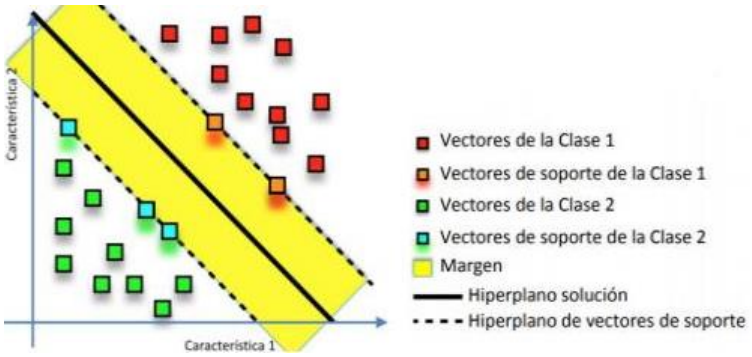


Gráfico 4-1: Clasificador Support Vector Machine (SVM)

Fuente: (Lopez, 2021)

1.8.2. Explicación matemática

Dado un conjunto de datos de entrenamiento $D=\{ x_i,y_i \}^{M}_{i=1}$, cada instancia x_i representa un vector de características de entrada n-dimensional y posee una etiqueta de clase binaria $y_i \in \{-1, +1\}$. El objetivo de SVM es encontrar un hiperplano de decisión en un espacio n-dimensional que separe las muestras de entrenamiento en dos clases y maximice la distancia del margen a medida que reduce el error de clasificación (García et al., 2021: p. 74).

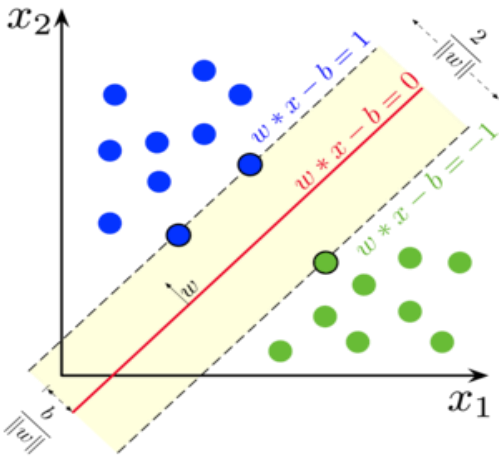


Gráfico 5-1: Hiperplano óptimo de SVM

Fuente: (Atoledo, 2020)

Si los datos se pueden separar de manera lineal, entonces el algoritmo encontrará fácilmente un

hiperplano de separación, el cual clasificará sin errores las muestras de ambas clases. La ecuación de dicho hiperplano se representa de la siguiente manera.

$$w^T x+b=0 \quad \textbf{(1.1)}$$

Donde **w** representa un vector de peso, **b** representa el sesgo y **x** representa la muestra de entrenamiento. El hiperplano divide el espacio en dos dimensiones, un semiespacio positivo donde se ubican las muestras de la clase positiva y un semiespacio negativo donde se ubican las muestras de la clase negativa. La finalidad del algoritmo es encontrar los valores de w y b de tal manera que el hiperplano se oriente a una distancia lo más distante posible de las muestras de ambas clases y para lograr dicho fin debe encontrar dos hiperplanos paralelos de la siguiente manera (Tharwat, 2019).

$$w^T x_i+b \geq +1 \text{ para } y_i=+1 \quad \textbf{(2.1)}$$

$$w^T x_i+b \leq -1 \text{ para } y_i=-1 \quad \textbf{(3.1)}$$

Donde $w^T x_i+b \geq +1$ representa el plano de la clase positiva y $w^T x_i+b \leq -1$ representa el plano de la clase negativa. Ambas ecuaciones se pueden reescribir de la siguiente manera:

$$y_i(w^T x_i+b) -1 \geq 0, \ \forall i = 1, 2,..., M \quad \textbf{(4.1)}$$

El margen que separa la clase positiva y la clase negativa se representa como $2/\|w\|$. Entonces, para encontrar el hiperplano óptimo de separación el algoritmo SVM deberá maximizar este margen el cual está sujeto restricciones resolviendo el siguiente problema de optimización cuadrática (Tharwat, 2019).

$$\min \frac{1}{2}\|w\|^2 \quad \textbf{(5.1)}$$

$$\text{s.t. } y_i(w^T x_i+b) -1 \geq 0, \ \forall i = 1, 2,..., M \quad \textbf{(6.1)}$$

1.8.3. SVM de margen suave

Cuando SVM no encuentra un hiperplano óptimo que pueda separar las muestras en dos clases debido a la presencia de ruido en estas, se utiliza el método de margen suave el cual relaja las restricciones de las ecuaciones **(2.1**) y **(3.1)**, mediante la introducción de variables de holguras

positivas ξ_i. El nuevo problema a optimizar se expresa de la siguiente manera (Lei, 2017, p. 123).

$$\min \frac{1}{2}\|w\|^2+C\sum_{i=i}^{M}\xi_i \quad \textbf{(7.1)}$$

$$\text{s.t.} \begin{cases} y_i(w^T x_i+b) \geq 1-\xi_i, & i=1,2,\ldots,M \\ \xi_i \geq 0, & i=1,2,\ldots,M \end{cases} \quad \textbf{(8.1)}$$

Donde ξ_i mide la distancia entre el margen y las muestras x_i que se encuentran en el lado incorrecto del margen.

1.8.4. SVM para casos linealmente no separables

Cuando las muestras no se pueden separar linealmente como se indica en el grafico 6-1. SVM amplía sus capacidades de clasificación, mediante la aplicación de funciones kernel que mapean las muestras a un espacio de alta dimensión en el cual es posible la clasificación lineal. Una función kernel $k(x,y)$ es el producto interno entre las muestras donde $k(x,y) = \langle \varphi(x), \varphi(y) \rangle$ (Lei, 2017, p. 123).

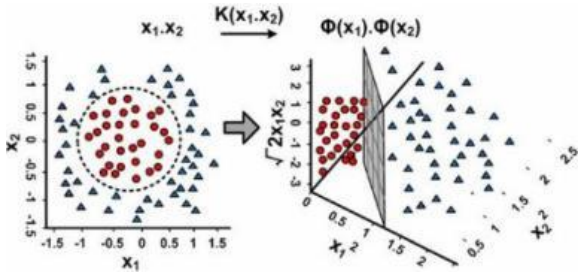


Gráfico 6-1: Uso de la función kernel

Fuente: (Torres, 2019)

Mediante el uso de la función kernel la expresión del problema de optimización en forma dual se expresa de la siguiente manera:

$$\text{Max } L(\alpha)=\sum_{i=1}^{M}\alpha_i-\frac{1}{2}\sum_{i,j=0}^{M}\alpha_i\alpha_j y_i y_j \mathbf{k}(x_j x_j) \quad \textbf{(9.1)}$$

$$\text{s.t.} \begin{cases} 0 \leq \alpha_i \leq C, & i=1,2,\ldots,M \\ \sum_{i=1}^{M}\alpha_i y_i=0 \end{cases} \quad \textbf{(10.1)}$$

Para una clasificación no lineal la función de decisión se describe de la siguiente manera:

$$f(x)=sing(\sum_{i,j=1}^{M} \boldsymbol{\alpha}_i y_i \mathbf{k}(x_i x_j)+b) \quad \textbf{(11.1)}$$

Las fusiones kernel definen el espacio de alta dimensión donde se clasificarán las muestras y su uso es importante para mejorar el rendimiento de SVM. En la tabla 1-1 se muestran las funciones kernel comúnmente utilizadas en SVM, de las cuales la función kernel RBF es la más utilizada para realizar el diagnóstico de fallas.

Tabla 1-1: Tipos de funciones kernel de SVM

Kernel	$\mathbf{k(x, x_j)}$
Lineal	$x^T x_j$
Polynomial	$(\gamma x^T x_j + r)^d, \gamma > 0$
Radial (RBF)	$\exp(-\frac{\|x - x_j\|^2}{2\gamma^2})$

Fuente: (Lei, 2017, p. 124)

Realizado por: Escobar, José, 2021

1.8.5. Parámetros de ajuste de SVM

1.8.5.1. Parámetro de penalización C

En el entrenamiento SVM tiene como finalidad encontrar un hiperplano óptimo que maximice el margen entre los puntos cercanos de diferentes clases, sin embargo, este proceso es susceptible de generar un modelo que se sobreajuste a los datos de entrenamiento. Por esta razón, este proceso está controlado por un parámetro C, el cual permite un margen suave de error de clasificación reduciendo la posibilidad de sobreajuste, mientras se mantiene una predicción precisa. El valor de C debe permitir un equilibrio de tal manera que se consiga mejorar el rendimiento de un modelo y evitar el sobreajuste (Al-Zoubi et al., 2018: p. 3).

1.8.5.2. Parámetro gamma (γ)

Gamma es un parámetro que forma parte del kernel radial. Un valor de gamma alto permite identificar únicamente puntos cercanos al hiperplano y tiende a generar modelos con sobreajuste, mientras que un valor de gamma bajo identifica puntos a mayor distancia haciendo que se agrupen más puntos o vectores de soporte. Al igual que C el valor de gamma debe permitir un equilibrio de tal manera que se consiga mejorar el rendimiento de un modelo (Gualoto, 2021, p. 11).

1.9. Pasos para construir un modelo de machine learning

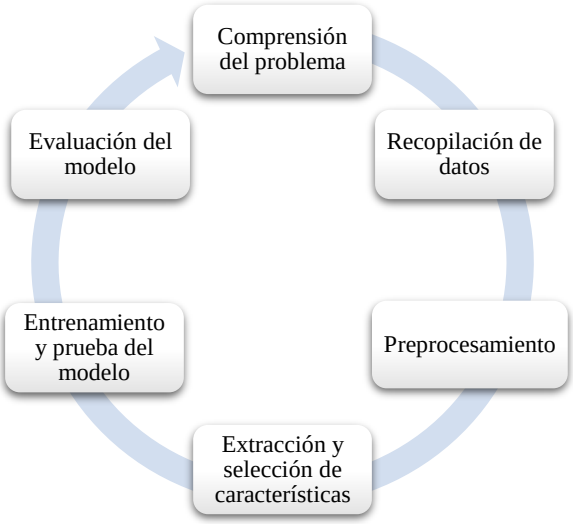


Gráfico 7-1: Pasos para construir un modelo de ML

Fuente: (Manrique, 2020)

Realizado por: Escobar, José, 2021

1.9.1. Comprensión del problema

Este paso se menciona muy pocas veces en publicaciones y libros, sin embrago, es uno de los pasos más importantes, debido a que para iniciar cualquier proyecto primero se debe conocer que se va a solucionar especialmente en ML donde pueden existir distintos enfoques y maneras de analizar un mismo conjunto de datos. En esta investigación, el problema de ML es realizar una clasificación que identifique la condición saludable que representa una condición sin falla y la condición de diente roto de engranaje que representa una condición de falla, utilizando datos de señales de vibración de cuatro acelerómetros. A partir de la formulación anterior se derivan elementos principales del problema de ML que son los siguientes:

- **Características:** Datos de señales de vibración de 4 acelerómetros
- **Tarea:** Clasificación binaria
- **Objetivo:** Condición saludable y condición de diente roto de engranaje

1.9.2. Recopilación de datos

En este paso se utilizan sensores los cuales se montan sobre las máquinas para hacer mediciones y recopilar datos. Generalmente se emplean diferentes tipos de sensores como, vibración, temperatura y emisión acústica, de los cuales los datos obtenidos mediante vibración son los que se emplean con mayor frecuencia para la detección de fallas en cajas de engranajes y otras

máquinas rotativas. Además, se puede hacer uso de datos recopilados por sensores de diferentes fuentes dichos datos pueden aportar información complementaria que podría combinarse para obtener una mejor precisión de diagnóstico. También se puede obtener datos de repositorios y datos obtenidos a partir de experimentos en bancos de prueba (Lei et al., 2020: p. 5)

1.9.3. Preprocesamiento de datos

Este paso es importante debido a que el éxito del modelo dependerá en gran medida de la calidad y forma que tengan los datos. Los datos recopilados suelen están incompletos, es decir, existirán errores tales como, valores atípicos, valores de atributos faltantes e inconsistencia lo que genera discrepancia entre los datos. Los datos no tratados tienden a generar salidas no fiables e invalidas, además entrenar un modelo utilizando datos no tratados puede llevar mucho tiempo y una alta carga computacional. El preprocesamiento tiene como objetivo transformar un conjunto de datos de tal manera que pueda ser procesado de forma eficiente por un algoritmo de ML (Swamynathan, 2019, p. 70).

Los pasos para mejorar la calidad de los datos son las siguientes:

- Eliminación de ruido o valores atípicos.
- Eliminación de inconsistencias y datos duplicados.
- Elección de estrategias para manejar y completar datos faltantes.
- Reducción de la dimensión y numerosidad de los datos.
- Transformación de variables categóricas y normalización.

1.9.4. Extracción de características

La extracción de características es un paso importante en el proceso de creación de modelos de ML, por medio de este proceso las señales sin procesar se transforman en representaciones relevantes las cuales estarán listas para ser ingresadas a un clasificador para entrenar y optimizar una función de decisión. Las técnicas de clasificación comúnmente utilizadas para la detección de fallas en las máquinas son, las SVM y las Redes Neuronales Artificiales (ANN) (Chen et al., 2019: p. 2).

La extracción de características se realiza mediante dos pasos. Primero, de los datos recopilados se extraen características en el dominio del tiempo y características en el dominio de la frecuencia. Estas características muestran información que refleja la condición de las máquinas. En segundo lugar, se emplean métodos de selección de características como son, filtro, envoltorios y métodos

integrados, para seleccionar aquellas características con mayor sensibilidad a los estados de salud de las máquinas con el objetivo de eliminar la información redundante, reducir la carga computacional y mejorar el rendimiento del modelo (Lei et al., 2020: p. 5).

1.9.4.1. Extracción de características en el dominio del tiempo

Las características en el dominio del tiempo se dividen en dimensionales y adimensionales. Las características dimensionales son, la desviación estándar, la media, la raíz cuadrática media, la raíz de la amplitud, valor pico, entre otras, que se caracterizan por verse afectadas por la velocidad y la carga de la máquina. Por otra parte, las características adimensionales incluyen la curtosis, indicador de forma, indicador de holgura, asimetría, indicador de impulso que se caracterizan por ser robustas ante las condiciones de funcionamiento de las máquinas (Lei et al., 2020: p. 5).

1.9.4.2. Extracción de características en el dominio de la frecuencia

Estas características se obtienen del espectro de frecuencia y generalmente son la frecuencia media cuadrática, frecuencia de desviación estándar, frecuencia media, centro de frecuencia, que poseen información que no es posible encontrar en las características del dominio del tiempo. También se extraen características en el dominio tiempo-frecuencia como la entropía de la energía que refleja la condición de la máquina en condición de funcionamiento no estacionario, entre otras (Lei et al., 2020: p. 5).

1.9.5. Métodos de selección de características

1.9.5.1. Métodos de filtro

Los métodos de filtro tienen como objetivo reducir el número de características extraídas, eliminando aquellas de menor importancia o que son redundantes, permitiendo obtener únicamente características con la mejor información predictiva y que puedan ser interpretadas de manera eficiente por un clasificador. Además, eliminar las características menos importantes permite reducir la carga computacional y acelerar el entrenamiento de un modelo predictivo. Los métodos de filtro comúnmente utilizados son la eliminación de características altamente correlacionadas y características cuya varianza es cercana a cero (Pisner y Schnyer, 2020: p. 105).

1.9.5.2. Métodos de envoltura

Con los métodos de envoltura el clasificador se entrena repetidamente intentando construir varios

modelos, mediante subconjuntos de características con el fin de obtener un modelo con el mejor desempeño y las mejores características. Es un método que demanda una alta carga computacional pero efectivo porque elimina las características que no aportan valor predictivo al modelo (Sarkar et al., 2018: p. 242).

1.9.5.3. Eliminación Recursiva de Características con Validación Cruzada (RFECV)

El método RFECV es uno de los enfoques de envoltura cuyo fin es determinar un conjunto de características durante cada proceso de recursividad mediante un estimador. El estimador es utilizado para obtener los coeficientes de peso de las características y los puntajes de validación cruzada en cada recursividad. Las características son seleccionadas de acuerdo con el valor de sus coeficientes de peso los cuales determinan que tan importantes son las características para el modelo y se eliminan aquellas características cuyos coeficientes de peso son pequeños. Después de la recursividad, las características más importantes son seleccionadas de acuerdo con los resultados de validación cruzada (Lu et al., 2021).

1.10. Entrenamiento y prueba del modelo

El entrenamiento consiste en proporcionarle a un algoritmo de ML un conjunto de datos a partir de los cuales aprenderá patrones y optimizará una función de decisión, como resultado se obtiene un modelo capacitado y listo para ser implementado. El modelo entrenado es implementado en la práctica para hacer predicciones, mediante la introducción de datos nuevos que actúan como datos de entrada para posteriormente obtener una predicción como salida. La fase de entrenamiento suele ser muy intensiva computacionalmente y requiere de grandes conjuntos de datos (Verbraeken et al., 2020: p. 5).

1.11. Evaluación del modelo

Un modelo de ML tiene como objetivo aprender de patrones de los datos de entrenamiento y estar en la capacidad de generalizar de manera correcta datos nuevos. Una vez creado el modelo el siguiente paso es comprobar el rendimiento de este, mediante su evaluación a partir de datos nuevos de esa manera se verifica que tan preciso es el algoritmo para realizar las predicciones deseadas y de no ser el caso realizar cambios en los parámetros hasta alcanzar un rendimiento aceptable. Existen diferentes métodos para evaluar el rendimiento de un modelo predictivo, en esta investigación se utilizó la matriz de confusión como principal método de evaluación y como métodos secundarios, la curva ROC con su valor de área bajo la curva y análisis de curvas de aprendizaje.

1.11.1. Matriz de confusión

La matriz de confusión es una estructura tabular que permite evaluar el rendimiento de un modelo predictivo mediante el seguimiento de las clasificaciones correctas, así como de las clasificaciones incorrectas, al comparar las etiquetas de los datos verdaderos con las etiquetas de los datos predichos. Cada columna representa recuentos de instancias clasificadas de acuerdo a las predicciones realizadas por el modelo, mientras que cada fila representa recuentos de instancias clasificadas de acuerdo a las etiquetas reales (Sarkar et al., 2018: p. 272).

Tabla 2-1: Matriz de confusión

		Etiquetas predichas	
		Negativo	**Positivo**
Etiquetas reales	**Negativo**	Verdaderos Negativos **(VN)**	Falsos positivos **(FP)**
	Positivo	Falsos negativos **(FN)**	Verdaderos positivos **(VP)**

Fuente: (Sarkar et al., 2018)

Realizado por: Escobar, José, 2021

- **VP:** Total de muestras de la clase positiva predichas correctamente.
- **FP**: El verdadero valor es negativo, pero el modelo predijo como positivo.
- **VN:** Total de muestras de la clase negativa predichas correctamente.
- **FN:** El verdadero valor es positivo, pero el modelo predijo como negativo.

La diagonal principal representa las instancias clasificadas correctamente, mientras que la diagonal secundaria representa las instancias clasificadas incorrectamente.

1.11.2. Métricas de evaluación de la matriz de confusión

1.11.2.1. Exactitud

Expresa el número de instancias clasificadas de forma correcta del total de instancias positivas y negativas. Se obtiene de dividir el número de instancias predichas correctamente sobre la totalidad de instancias predichas correcta e incorrectamente (Shobha y Rangaswamy, 2018: p. 203).

$$\textbf{Exactitud} = \frac{VP+VN}{VP+FP+VN+FN} \qquad \textbf{(12.1)}$$

1.11.2.2. Precisión

La precisión o valor predictivo positivo, permite cuantificar cuantas instancias predichas como positivas son realmente positivas. Esta métrica se utiliza cuando el objetivo es limitar el número de falsos positivos (Müller y Guido, 2017: p. 283).

$$\textbf{Precisión} = \frac{VP}{VP+FP} \qquad \textbf{(13.1)}$$

Un modelo con alta precisión reconocerá más muestras de la clase positiva en comparación con un modelo de baja precisión. La precisión es importante cuando se desea identificar el número máximo de instancias positivas, incluso si se reduce la exactitud (Sarkar et al., 2018: p. 274).

1.11.2.3. Sensibilidad

Se define como el número de muestras de la clase positiva predichas de forma correcta, también se le conoce como la tasa de aciertos. La sensibilidad se convierte en una métrica importante del rendimiento del clasificador cuando se desea identificar la mayor cantidad de instancias de una clase en particular, incluso si los falsos positivos aumentan (Sarkar et al., 2018: p. 275).

$$\textbf{Sensibilidad} = \frac{VP}{VP+FN} \qquad \textbf{(14.1)}$$

1.11.2.4. Especificidad

Se define como el número de instancias de la clase negativa predichas de forma correcta.

$$\textbf{Especificidad} = \frac{VN}{VN+FP} \qquad \textbf{(15.1)}$$

1.11.2.5. Puntuación F1

Es la media armónica de la precisión y sensibilidad que permite optimizar un clasificador para lograr un balance en el rendimiento de precisión y sensibilidad. Esta métrica tiene mayor uso cuando los datos están desequilibrados (Sarkar et al., 2018: p. 275).

$$\textbf{Puntuación F1} = \frac{2*\text{Precisión}*\text{Sensibilidad}}{\text{Precisión}+\text{Sensibilidad}} \qquad \textbf{(16.1)}$$

1.11.3. Curva Característica Operativa del Receptor (ROC)

La curva ROC es una herramienta que permite visualizar gráficamente la relación y equilibrio entre la Tasa de Verdaderos Positivos (TPR) o sensibilidad frente a la Tasa de Falsos Positivos (FPR) de un modelo de clasificación binaria para distintos puntos de corte (Ali et al., 2019).

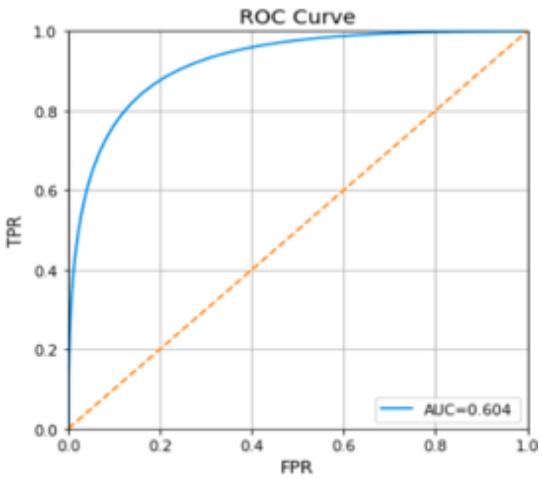


Gráfico 8-1: Curva ROC

Fuente: (Nazrul, 2018)

1.11.3.1. Área Bajo la Curva (AUC)

El AUC es un indicador de la calidad de un modelo de ML, mientras más hacia la izquierda está la curva habrá más área bajo esta, por lo tanto, mejor será el clasificador para identificar correctamente las clases. Un clasificador que identifica correctamente diferentes clases sin equivocarse tendrá un AUC de 1, sin embargo, esto es un caso ideal, debido a que siempre existieran errores de clasificación. Un clasificador que realiza predicciones al azar obtendrá un AUC de 0,5 y a medida que mejora esta puntuación, es decir cuando el AUC más se aproxime a 1 mejor será la capacidad discriminativa del clasificador (Ali et al., 2019).

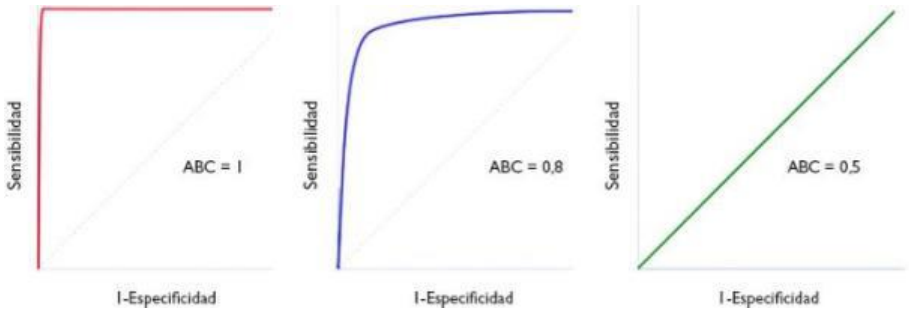


Gráfico 9-1: Curvas ROC con distintos grados de convexidad

Fuente: (Ochoa, 2017)

1.12. Optimización de hiperparámetros

Cada modelo de aprendizaje de máquinas dispone de diferentes hiperparámetros, seleccionar los hiperparámetros correctos permite mejorar el rendimiento de un modelo, lo cual puede marcar una diferencia significativa en los resultados de predicción. Por lo tanto, es importante realizar un ajuste de hiperparámetros en lugar de entrenar un modelo con parámetros predeterminados. La búsqueda de estos hiperparámetros se lo puede realizar manualmente, pero eso conlleva mucho tiempo y esfuerzo. Para optimizar este proceso se utilizan algoritmos como la búsqueda de cuadricula que busca automáticamente dichos hiperparámetros (Wicaksono y Supianto, 2018: p. 264).

1.12.1. Búsqueda de cuadrícula

La búsqueda de cuadrícula es un método de optimización de hiperparámetros cuyo objetivo es encontrar la mejor combinación de hiperparámetros que permitan mejorar el rendimiento de un modelo predictivo. Para su aplicación se debe establecer una lista de valores de hiperparámetros los cuales son evaluados mediante un estimador y haciendo uso de la validación cruzada. Como resultado el método devuelve la combinación de hiperparámetros con los cuales el modelo logra desempeñares de la mejor manera (Sarkar et al., 2018: p. 291).

1.13. Overfitting y underfitting

1.13.1. Overfitting

EL overfitting o sobreajuste es un problema que se presenta cuando un modelo de ML memoriza las especificaciones y el ruido de los datos de entrenamiento lo cual afecta el rendimiento y capacidad discriminativa del modelo, provocando que este sea incapaz de generalizar de manera correcta datos nuevos nunca vistos. Si un modelo tiene un error de entrenamiento bajo pero un alto error de prueba entonces está sujeto a sobreajuste (Wagner et al., 2021).

1.13.2. Underfitting

El underfitting o desajuste es un problema que se presenta generalmente cuando un modelo de ML muestra un bajo rendimiento tanto con los datos de entrenamiento como con los datos de prueba. El modelo no logra aprender las características más importantes del conjunto de datos de entrenamiento y, en consecuencia, no puede clasificar correctamente datos nuevos. El underfitting se detecta cuando el clasificador tiene un alto error de entrenamiento y un alto error de prueba (Rojas, 2020, p. 9).

1.13.3. Soluciones al overfitting y underfitting

Cuando un modelo creado con un algoritmo especifico presenta desajuste lo más recomendable es intentar probar con otros algoritmos, mientras que para solucionar los problemas de sobreajuste existen diferentes alternativas tales como, utilizar más datos para entrenar el modelo con el fin de que existan ejemplos más representativos de posibles muestras futuras, reducir el tamaño de las características de entrada eliminando aquellas que no aportan valor a la predicción y por último regularizar. Mediante la regularización se penalizan ciertos parámetros propios de los modelos permitiendo que estos sean más flexibles y menos propensos de ajustarse al ruido de esta manera se consigue que un modelo se adapte bien a futuras muestras reduciendo así el sobreajuste (Díaz, 2021, p. 34).

1.13.4. Validación cruzada

Para entrenar un modelo generalmente se utiliza train_test_split que genera aleatoriamente un conjunto de entrenamiento y un conjunto de prueba, sin embargo, cuando se mide el rendimiento del modelo los resultados pueden darse por casualidad, debió a la aleatoriedad de la división realizada, sobre todo en conjuntos de datos cuyas clases están desequilibradas. Para comprobar que los resultados obtenidos son confiables se utiliza el método de validación cruzada que evalúa la capacidad de generalización de un modelo creado para diferentes conjuntos de prueba y es más confiable que una división aleatoria del conjunto de entrenamiento y conjunto de prueba (Varela, 2020, p. 51).

1.13.5. Validación cruzada de n iteraciones

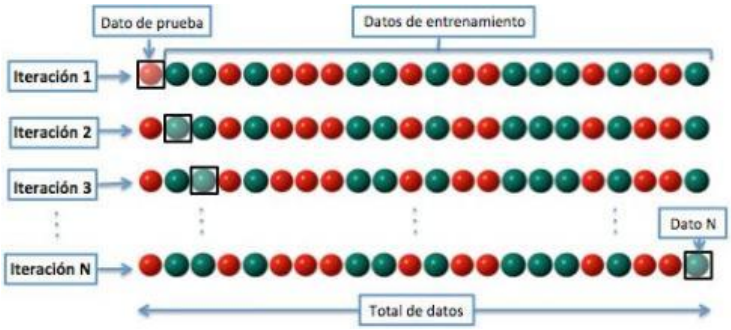


Figura 4-1: Método de validación cruzada

Fuente: (Ashfaque, 2019)

Es un método que divide los datos de entrenamiento en n partes de igual proporción, de las cuales una parte se utiliza como conjunto de prueba y el resto (n-1) como conjunto de entrenamiento. El entrenamiento y la prueba se repiten n veces y en cada iteración se toma un conjunto diferente

para probar el modelo y el resto (n-1) para entrenar el mismo, cuando finaliza las n repeticiones se obtiene diferentes puntajes de precisión por cada iteración, a partir de estos puntajes se calcula la precisión promedio y la desviación estándar. Los resultados de la precisión promedio y de la desviación estándar permiten tener una estimación más fiable de la precisión que tendrá el modelo cuando sea implementado y probado con datos no vistos (Ochoa, 2019, p. 2).

1.14. Curvas de aprendizaje

Estas curvas representan las puntuaciones de entrenamiento y de prueba de un modelo predictivo para diferentes tamaños de muestras de entrenamientos. El objetivo de estas curvas es indicar si es conveniente agregar más muestras de entrenamiento y en qué medida y si eso conlleva a generar errores de sesgo y varianza (González, 2017, p. 95).

Gráfico 10-1: Curvas de aprendizaje

Fuente: (Blanco, 2019)

- Si al agregar más muestras de entrenamiento las puntuaciones de entrenamiento y validación convergen en un punto bajo significa que el modelo está sujeto a sesgo alto y no conviene agregar más datos ya que el modelo tiene un bajo rendimiento y este no mejora con el incremento de muestras.
- Si para todo el conjunto de muestras de entrenamiento, la puntuación de entrenamiento es mayor que la puntuación de validación, pero la curva de validación mantiene una pendiente positiva, agregar más muestras de entrenamiento probablemente ayudará a mejorar el rendimiento del modelo. (González, 2017, p. 95).

1.14.1. Sesgo

El error de sesgo es la diferencia entre el valor de predicción del estimador de un modelo y el valor real que se quiere predecir. También se lo define como el error de aproximación promedio de un modelo con respecto a todo el conjunto de datos de entrenamiento. Un sesgo alto genera un

bajo rendimiento tanto en el entrenamiento como en la prueba (Sarkar et al., 2018: p. 285).

1.14.2. Varianza

Representa la sensibilidad de los resultados de predicción de un modelo para diferentes conjuntos de datos de entrenamiento. En la curva de aprendizaje la varianza cuantifica y muestra el cambio de los resultados de precisión del modelo con el cambio del conjunto de datos de entrenamiento. Si los resultados se mantienen estables se dice que el modelo tiene una baja varianza, pero si estos varían considerablemente se dice que el modelo tiene una alta varianza. El objetivo de ML es crear un modelo que tenga un bajo sesgo y una baja varianza (Sarkar et al., 2018: p. 285).

1.15. Prueba de permutación

En ML se utiliza la prueba de permutación para evaluar la significancia estadística del rendimiento de un algoritmo. La prueba de permutación mide la probabilidad de que el rendimiento obtenido por un algoritmo sea por casualidad. En esta prueba las etiquetas son permutadas k veces de tal manera que se pierde cualquier relación estadística entre las características y las etiquetas, seguidamente se entrena y prueba un modelo en cada permutación haciendo uso de la validación cruzada. Los resultados obtenidos que son una distribución de medidas de permutación, reflejan la hipótesis nula de que la precisión obtenida por el modelo es por casualidad o que no existe dependencia entre las etiquetas y características (Good, 1994; citado en Vieira et al., 2020).

Sea $\widehat{D}=\{D'_1, D'_2,.....,D'_k\}$, un conjunto de k versiones aleatorias de datos con etiquetas permutadas de D, donde D es el conjunto original sin permutación. El valor empírico de p para un clasificador f se calcula de la siguiente manera:

$$p = \frac{\left| D' \in D: e(f, D') \leq e(f, D) \right| + 1}{k+1} \qquad \textbf{(17.1)}$$

El valor p representa la fracción de muestras aleatoria en las cuales el clasificador obtuvo mejores resultados con los datos aleatorios que con los datos originales. Si el valor de p obtenido después de la prueba de permutación es menor a un umbral de $\alpha=0,05$ se rechaza la hipótesis nula de que las características y las etiquetas son independientes y se acepta la hipótesis alternativa de que existe una dependencia entre las características y las etiquetas y el modelo utilizo esa dependencia para lograr su buen rendimiento, es decir, el resultado obtenido por la métrica de evaluación no fue por casualidad (Arias et al., 2017: p. 4).

1.16. Lenguajes de programación de machine learning

En el mundo de la tecnología actual el aprendizaje de máquinas es un tema que está ganando mucha popularidad y relevancia. Esta popularidad ha permitido que se puedan crear diferentes soportes en términos de lenguajes de programación, así como la creación de bibliotecas para una gran variedad de lenguajes de programación tales como, C++, Julia, R, Scala, Python, entre otros (Sarkar et al., 2018: p. 67).

Figura 5-1: Lenguajes de programación más populares

Fuente: (Morales, 2021)

1.16.1. Python

Python es un lenguaje de programación de alto nivel, versátil, multiparadigma, multiplataforma, multipropósito e interpretado cuyo objetivo es proporcionar una sintaxis que favorezca un código limpio, legible y fácil de interpretar (Quintero, 2021, p. 13).

1.16.2. Ventajas de Python

- Posee una sintaxis legible y fácil de comprender para un inexperto.
- Permite realizar programación estructurada, programación orientada a objetos y programación funcional, lo que lo convierte en un lenguaje de programación de usos múltiples.
- Python tiene a su disposición un gran número módulos de fácil instalación y uso que cubren todos los aspectos de la programación, garantizando que un analista de datos sea más productivo.
- Es de código abierto y apoyado por una gran comunidad de desarrolladores activos.
- Puede ser ejecutado en distintas plataformas como: Windows, Linux, MAC y Android.
- No requiere licencia para ser utilizado (Sarkar et al., 2018: p. 68).

Al ser un lenguaje de programación interpretado Python tiene la desventaja de que al ejecutar un

código la velocidad de ejecución es lenta en comparación con otros lenguajes compilados.

1.16.3. ¿Por qué Python?

De acuerdo con el índice Tiobe una empresa especializada en evaluar la calidad de código de software y medir la popularidad de lenguajes de programación, en su última actualización realizada en enero de 2022, indica que Python lidera con un 13,58% convirtiéndose en el lenguaje de programación más popular y de mayor preferencia por la mayoría de los programadores de todo el mundo.

ene 2022	ene 2021	Cambio	Lenguaje de programación	Calificaciones	Cambio
1	3	▲	Pitón	13,58%	+1.86%
2	1	▼	C	12,44%	-4,94%
3	2	▼	Java	10,66%	-1,30%

Figura 6-1: Índice de la comunidad de programación Tiobe

Fuente: (Tiobe, 2021)

Python puede ser utilizado en aplicaciones de todos los campos científicos o tecnológicos ya sea para análisis de datos, desarrollo de aplicaciones, aplicaciones de escritorio, creación de software, juegos, desarrollo de sitios web, entre otras muchas aplicaciones, a diferencia de otros lenguajes de programación, por ejemplo, R que únicamente está destinado al análisis de datos estadísticos. Por estas razones Python es el lenguaje de programación de mayor preferencia por desarrolladores de software y científicos de datos (Quintero, 2021, p. 15).

1.16.4. Jupyter Notebook

Jupyter notebook es un entorno interactivo que sirve para realizar el análisis de datos basado en Python. Específicamente son cuaderno en los cuales se pueden escribir códigos, agregar textos, insertar imágenes, resultados y organizar paso a paso todo el proceso de análisis de datos que se va elaborando, lo que lo convierte en una herramienta maravillosa para realizar análisis e investigación al mismo tiempo. A medida que evoluciona el proceso de análisis, este puede ser documentado y esta combinación de código, documentación y resultados hacen que esta sea una herramienta muy necesaria y útil para todos los analistas de datos (Sarkar et al., 2018: p. 73).

1.16.5. Librerías de Python

Python posee una gran cantidad de librerías de alto nivel que amplían sus capacidades de

programación, así como una gran munidad de desarrolladores que constantemente están creando y proporcionando nuevas librerías.

1.16.5.1. NumPy

Permite realizar cálculos numéricos, creación de matrices normales y de gran dimensión, cálculos con vectores, entre otros. Una de las ventajas de esta librería es su capacidad para acelerar los cálculos matemáticos y lograr ejecuciones más rápidas (Sarkar et al., 2018: p. 75).

1.16.5.2. Scikit-learn

Esta librería incorpora una gran cantidad de algoritmos para cubrir todas las necesidades de ML como, clasificación, regresión y agrupación. Distintos algoritmos como, SVM, bosques aleatorios, agrupación jerárquica, regresión logística y otros forman parte de esta biblioteca, por lo tanto, su uso para llevar a cabo cualquier proyecto de ML es indispensable (Sarkar et al., 2018: p. 96).

1.16.5.3. SciPy

Proporciona una variedad de funciones para realizar cálculos matemáticos en Python. Permite optimizar funciones matemáticas, realizar procedimientos avanzados de algebra lineal, procesar señales, análisis estadístico, funciones matemáticas especiales, entre otras (Müller y Guido, 2017: p. 8).

1.16.5.4. Matplotlib

Es una librería que permite realizar gráficas de alta calidad como, histogramas, diagramas de dispersión, gráficos de barras, mapas de calor, funciones matemáticas, visualización de datos, entre otras (Müller y Guido, 2017: p. 9).

1.16.5.5. Pandas

Es una librería utilizada especialmente para el análisis de datos. Esta librería es muy utilizada en ML para crear, modificar y acceder a estructuras de datos de gran dimensión específicamente para importar y leer estructuras de datos de distintos formatos como, CSV, SQL, formatos Excel, entre otros.

CAPÍTULO II

2. MARCO METODOLÓGICO

2.1. Propuesta general de los modelos de detección de fallas

Los modelos de detección de fallas de máquinas rotativa y otros tipos de máquinas basados en técnicas de ML, por lo general tienen cinco fases como se muestra en el gráfico 1-2. Para el presente trabajo de integración curricular se han diseñado tres modelos de detección de fallas denominados métodos experimentales a partir de una misma base de datos. Los tres modelos constan de las cinco fases descritas en el gráfico 1-2 y se diferencian únicamente en que cada uno se crea con un conjunto distinto de datos. En el método experimental uno (ME1) se crea un modelo predictivo utilizando datos registrados por cuatro acelerómetros bajo una carga de 50%. En el método experimental dos (ME2) se crea un modelo predictivo utilizando datos registrados por cuatro acelerómetros bajo un rango de cargas de 0 a 90% y por último en el método experimental tres (ME3) se crea un modelo predictivo utilizando únicamente los datos registrados por un solo acelerómetro del conjunto total de datos. El objetivo de crear tres modelos predictivos es comprobar la efectividad del método propuesto y demostrar que se pueden obtener los mismos resultados o similares utilizando diferentes conjuntos de datos, ya sea si se dispone únicamente de datos de un solo sensor o de varios sensores o a su vez datos registrados bajo una cierta carga o un amplio rango de cargas.

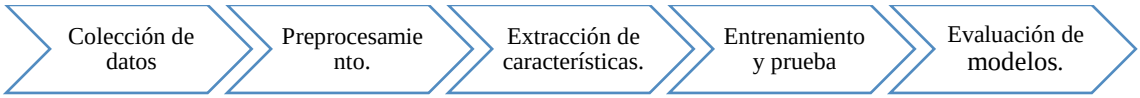


Gráfico 1-2: Fases para crear un modelo de ML

Fuente: (Manrique, 2020)

Realizado por: Escobar, José, 2021

Fase I: Se realiza una descripción detallada del conjunto de datos de señales de vibración de una caja de engranajes obtenido de un repositorio.

Fase II: Se utiliza el software de programación Python para analizar y comprender la información contenida en el conjunto de datos. Se realiza un análisis exploratorio y limpieza del conjunto de datos para solucionar problemas de datos incompletos, valores atípicos, datos duplicados, variables categóricas, así como elección de estrategias para manejar y completar datos faltantes

para que puedan ser procesados de forma eficiente por el algoritmo de ML.

Fase III: Se extraen características o indicadores de condición en los dominios de tiempo y frecuencia. El conjunto de características extraídas se divide en 70% para entrenamiento y 30% para prueba. Se realiza una selección de características utilizando el método de filtro basado en correlación y el método de Eliminación Recursiva de Características con Validación Cruzada (RFECV) para seleccionar las mejores características.

Fase IV: Se estandariza el conjunto de características seleccionadas y se ingresan al algoritmo SVM para entrenar los modelos predictivos. Los modelos entrenados se prueban con el conjunto de datos de prueba y posteriormente se optimizan hiperparámetros mediante la técnica de búsqueda de cuadrícula para mejorar el rendimiento de los modelos.

Fase V: Se evalúa el rendimiento de cada uno de los tres modelos mediante el análisis de validación cruzada, curvas de aprendizaje, curva ROC y métricas de evaluación de la matriz de confusión como principal método de evaluación. Finalmente se comprueba la hipótesis planteada mediante la prueba de permutación.

2.2. Fase I-Colección de datos

El conjunto de datos se señales de vibración de caja de engranajes se obtuvo del repositorio de Iniciativa de Datos de Energía Abierta (OEDI) del departamento de energía de EE. UU., el conjunto de datos incorpora datos de vibración registrados a partir del simulador de fallas de cajas de engranajes de SpectraQuest. Los datos han sido registrados en dos escenarios, para condición saludable y para condición de diente roto con la ayuda de 4 acelerómetros de vibración ubicados en cuatro posiciones diferentes del cuerpo y bajo la aplicación de distintas cargas con una variación de 0 a 90% (Pandya, 2018).

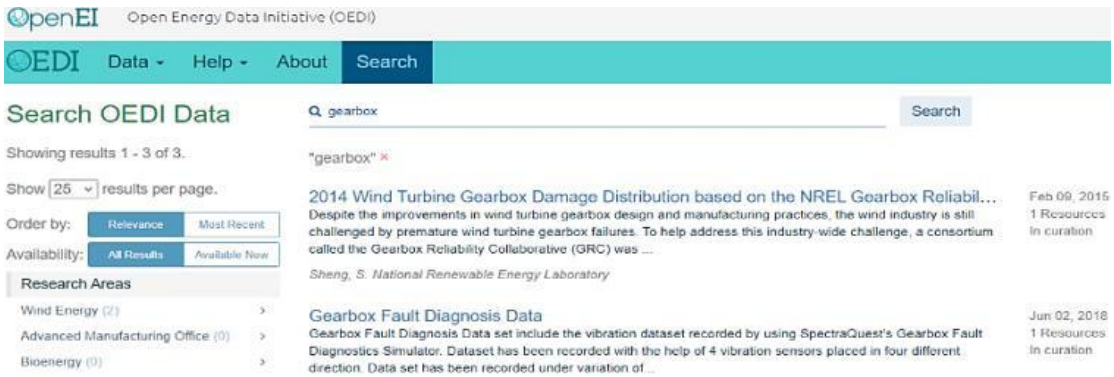
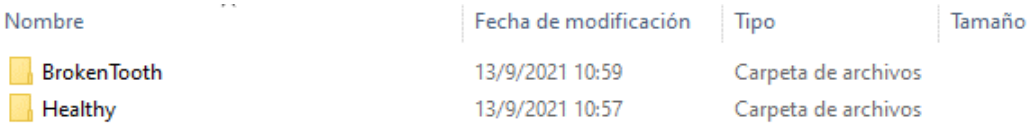


Figura 1-2: Repositorio de la base de datos OEDI

Realizado por: Escobar, José, 2021

2.2.1. Descripción del conjunto de datos

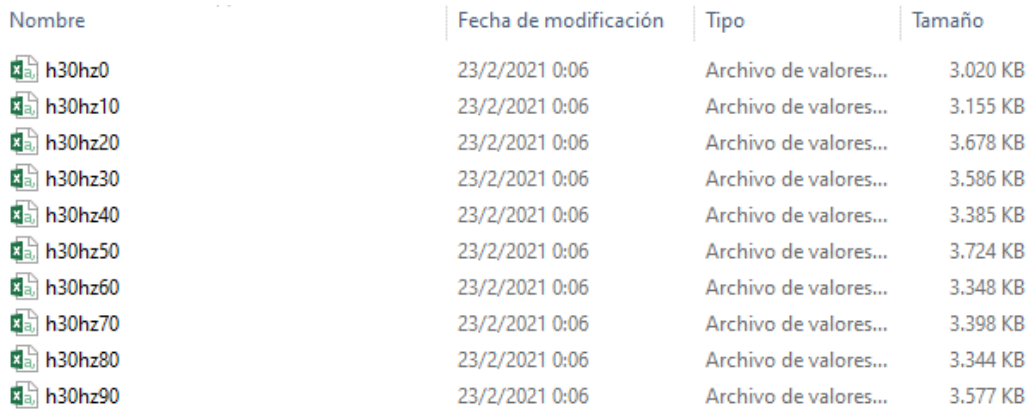
El conjunto de datos está conformado por dos subcarpetas; una carpeta llamada Healthy que contiene diez archivos en formato CSV correspondientes a señales de vibración medidas a partir de una caja de engranajes en condición saludable, es decir, sin falla y otra carpeta llamada BrokenTooth que contiene diez archivos en formato CSV correspondientes a señales de vibración medidas a partir de una caja de engranajes con falla de diente roto. A continuación, se muestran los archivos que contienen a los datos, así como una tabla general donde se describe detalladamente cada uno de estos.



Nombre	Fecha de modificación	Tipo	Tamaño
BrokenTooth	13/9/2021 10:59	Carpeta de archivos	
Healthy	13/9/2021 10:57	Carpeta de archivos	

Figura 2-2: Conjunto de datos de señales de vibración

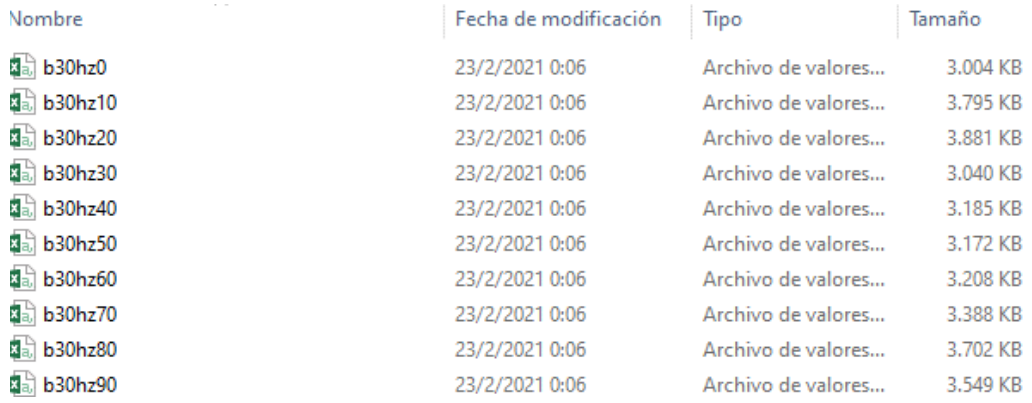
Realizado por: Escobar, José, 2021



Nombre	Fecha de modificación	Tipo	Tamaño
h30hz0	23/2/2021 0:06	Archivo de valores...	3.020 KB
h30hz10	23/2/2021 0:06	Archivo de valores...	3.155 KB
h30hz20	23/2/2021 0:06	Archivo de valores...	3.678 KB
h30hz30	23/2/2021 0:06	Archivo de valores...	3.586 KB
h30hz40	23/2/2021 0:06	Archivo de valores...	3.385 KB
h30hz50	23/2/2021 0:06	Archivo de valores...	3.724 KB
h30hz60	23/2/2021 0:06	Archivo de valores...	3.348 KB
h30hz70	23/2/2021 0:06	Archivo de valores...	3.398 KB
h30hz80	23/2/2021 0:06	Archivo de valores...	3.344 KB
h30hz90	23/2/2021 0:06	Archivo de valores...	3.577 KB

Figura 3-2: Conjunto de datos de señales de vibración en condición sana

Realizado por: Escobar, José, 2021



Nombre	Fecha de modificación	Tipo	Tamaño
b30hz0	23/2/2021 0:06	Archivo de valores...	3.004 KB
b30hz10	23/2/2021 0:06	Archivo de valores...	3.795 KB
b30hz20	23/2/2021 0:06	Archivo de valores...	3.881 KB
b30hz30	23/2/2021 0:06	Archivo de valores...	3.040 KB
b30hz40	23/2/2021 0:06	Archivo de valores...	3.185 KB
b30hz50	23/2/2021 0:06	Archivo de valores...	3.172 KB
b30hz60	23/2/2021 0:06	Archivo de valores...	3.208 KB
b30hz70	23/2/2021 0:06	Archivo de valores...	3.388 KB
b30hz80	23/2/2021 0:06	Archivo de valores...	3.702 KB
b30hz90	23/2/2021 0:06	Archivo de valores...	3.549 KB

Figura 4-2: Conjunto de datos de señales de vibración en condición de diente roto

Realizado por: Escobar, José, 2021

Tabla 1-2: Descripción general del conjunto de datos

Nombre	Descripción			
Healthy	**Frecuencia de muestreo (Hz)**	**Carga aplicada**	**Número de datos**	**Clase 0 (sin falla)**
h30hz0	30	0	88832	0
h30hz10	30	10	92928	0
h30hz20	30	20	108544	0
h30hz30	30	30	106240	0
h30hz40	30	40	100608	0
h30hz50	30	50	110848	0
h30hz60	30	60	99840	0
h30hz70	30	70	101376	0
h30hz80	30	80	99840	0
h30hz90	30	90	106752	0
Broken	**Frecuencia de muestreo**	**Carga aplicada**	**Número de datos**	**Clase 1 (con falla)**
b30hz0	30	0	88320	1
b30hz10	30	10	111616	1
b30hz20	30	20	114432	1
b30hz30	30	30	89856	1
b30hz40	30	40	94464	1
b30hz50	30	50	94208	1
b30hz60	30	60	95488	1
b30hz70	30	70	100864	1
b30hz80	30	80	110355	1
b30hz90	30	90	105728	1

Realizado por: Escobar, José, 2021

2.3. Fase II-preprocesamiento

Para llevar a cabo el preprocesamiento del conjunto de datos se utilizó Jupyter Notebook que incorpora por defecto el lenguaje de programación Python. En primera instancia se importó algunas librerías como son, pandas, numpy, matplotlib, scipy, entre otras necesarias para la importación, visualización, limpieza, manejo y cálculo estadístico del conjunto de datos de señales de vibración. En la figura 5-2 se muestran las librearías utilizadas para el preprocesamiento del conjunto de datos.

```
# LIBRERÍAS PARA EL PREPROCESAMIENTO
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
import matplotlib.gridspec as gridspec
from pylab import *

from scipy.stats import normaltest # Prueba de normalidad de D'Agostino's K-squared
from scipy.stats import kstest   # Prueba de normalidad de Kolmogorov-Smirnov
from sklearn.preprocessing import PowerTransformer # Normalizacion de datos método Yeo-Johnson
from scipy import fftpack # Transformada rápida de furier.
```

Figura 5-2: Librerías utilizadas para el preprocesamiento

Realizado por: Escobar, José, 2021

2.3.1. Lectura del conjunto de datos

Se realizó la importación y lectura de 20 archivos en formato CSV, de los cuales 10 archivos corresponden a datos de señales de vibración sin falla registrados bajo un rango de cargas de 0 a 90%, mientras que los otros 10 corresponden a datos de señales de vibración con falla de diente roto igualmente registrados bajo un rango de cargas de 0 a 90 %.

```
# Realiza la lectura de datos sin falla ubicados en el directorio especificado.
 dt_h0 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz0.csv",header=0)
dt_h10 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz10.csv",header=0)
dt_h20 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz20.csv",header=0)
dt_h30 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz30.csv",header=0)
dt_h40 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz40.csv",header=0)
dt_h50 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz50.csv",header=0)
dt_h60 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz60.csv",header=0)
dt_h70 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz70.csv",header=0)
dt_h80 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz80.csv",header=0)
dt_h90 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz90.csv",header=0)

# Realiza la lectura de datos con falla ubicados en el directorio especificado.
dt_b0 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz0.csv",header=0)
dt_b10 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz10.csv",header=0)
dt_b20 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz20.csv",header=0)
dt_b30 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz30.csv",header=0)
dt_b40 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz40.csv",header=0)
dt_b50 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz50.csv",header=0)
dt_b60 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz60.csv",header=0)
dt_b70 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz70.csv",header=0)
dt_b80 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz80.csv",header=0)
dt_b90 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz90.csv",header=0)
```

Figura 6-2: Lectura del conjunto de datos de señales vibración

Realizado por: Escobar, José, 2021

A cada archivo se le agregó una columna para identificar la carga aplicada y posteriormente se concatenó todo el conjunto de datos en únicamente dos archivos, un archivo denominado Healthy para representar al conjunto de datos de señales de vibración sin falla y otro denominado Broken para representar al conjunto de datos de señales de vibración con falla de diente roto.

2.3.2. Conjunto de datos de señales de vibración sin falla

	a1	a2	a3	a4	carga
0	4.636710	0.516978	-3.205940	1.82241	0.0
1	1.992800	4.184660	-2.740610	2.80436	0.0
2	-3.764110	0.997335	-1.303090	1.83668	0.0
3	-4.558710	6.104330	-1.720690	1.72311	0.0
4	0.575382	0.170980	-0.497967	-1.32895	0.0
...	...	...	...	...	...
106747	0.677448	-3.234410	-1.725990	-3.14302	90.0
106748	-10.575400	7.725400	-2.184010	2.56965	90.0
106749	-4.033290	2.576920	1.468430	2.72891	90.0
106750	1.868670	-5.089400	5.342290	-1.36563	90.0
106751	7.581480	6.205960	-8.121330	11.54830	90.0

1015808 rows × 5 columns

Figura 7-2: Conjunto de datos sin falla

Realizado por: Escobar, José, 2021

2.3.3. Conjunto de datos de señales de vibración con falla de diente roto

	a1	a2	a3	a4	carga
0	2.350390	1.454870	-1.667080	-2.055610	0.0
1	2.452970	1.400100	-2.825100	0.984487	0.0
2	-0.241284	-0.267390	0.793540	0.605862	0.0
3	1.130270	-0.890918	0.696969	0.613068	0.0
4	-1.296140	0.980479	-1.130560	-0.346971	0.0
...	...	...	...	...	...
105723	4.434170	-2.037930	-0.546417	1.181320	90.0
105724	2.903320	-2.820350	-2.486930	0.115640	90.0
105725	-1.617990	1.278610	-6.159510	-1.376230	90.0
105726	-2.011220	-4.029450	-0.082050	0.932847	90.0
105727	-4.437360	-1.970410	3.074570	-3.557610	90.0

1005311 rows × 5 columns

Figura 8-2: Conjunto de datos con falla

Realizado por: Escobar, José, 2021

2.3.4. Análisis exploratorio y limpieza de datos

Se realizó un análisis exploratorio para examinar los datos disponibles y verificar la naturaleza de estos, para lo cual se utilizó el método info () que muestra la información de un conjunto de datos indicando el número de columnas, número de filas y el tipo de datos identificando si estos son, numéricos, no numéricos, enteros o decimales. Es importante que los datos sean de tipo numérico, debido a que los modelos de ML se basan únicamente en ecuaciones y cálculos matemáticos. Como se muestra en la figura 9-2 todos los datos corresponden únicamente a valores numéricos de tipo flotante, por lo tanto, no se realizó ninguna transformación.

```
Healthy.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1015808 entries, 0 to 106751
Data columns (total 5 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   a1      1015808 non-null  float64
 1   a2      1015808 non-null  float64
 2   a3      1015808 non-null  float64
 3   a4      1015808 non-null  float64
 4   carga   1015808 non-null  float64
dtypes: float64(5)
memory usage: 46.5 MB
```

```
Broken.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1005311 entries, 0 to 105727
Data columns (total 5 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   a1      1005311 non-null  float64
 1   a2      1005311 non-null  float64
 2   a3      1005311 non-null  float64
 3   a4      1005311 non-null  float64
 4   carga   1005311 non-null  float64
dtypes: float64(5)
memory usage: 46.0 MB
```

Figura 9-2: Información del conjunto de datos Healthy y Broken

Realizado por: Escobar, José, 2021

Después de verificar el tamaño y la forma de los datos se realizó la búsqueda de valores faltantes, para lo cual se utilizó el método isna (). sum () que realiza la búsqueda de valores faltantes en

cada una de las columnas del conjunto de datos y devuelve la suma de dichos valores en caso de su existencia.

```
Healthy.isna().sum()
```

```
a1        0
a2        0
a3        0
a4        0
load      0
failure   0
```

Figura 10-2: Búsqueda de valores faltantes o nulos

Realizado por: Escobar, José, 2021

Como se muestra en la figura 10-2 no se encontraron valores faltantes en ninguna de las columnas del conjunto de datos. EL proceso de búsqueda de valores faltantes utilizando el método descrito se aplicó al conjunto de datos sin falla y al conjunto de datos con falla de diente roto y en ninguno de los dos casos se encontraron dichos valores. También se verifico la existencia de valores duplicados en filas haciendo uso del método df. drop_duplicates (), que encuentra y elimina automáticamente valores duplicados. El método devolvió como resultado el mismo número de filas por lo tanto se comprobó que no existen valores duplicados, adicionalmente se aplicó el método df. duplicated ().sum (), que devuelve la suma de valores duplicados en caso de su existencia. Este proceso es importante debido a que de otra manera los datos duplicados podrían ser considerados más importantes que el resto por parte de un modelo de ML provocando que este se entrene de manera sesgada.

```
Healthy.duplicated().sum()
```

```
0
```

```
Broken.duplicated().sum()
```

```
0
```

Figura 11-2: Búsqueda de valores duplicados

Realizado por: Escobar, José, 2021

2.3.5. Análisis gráfico exploratorio

En el gráfico 2-2 se representa el número de datos de acuerdo con la carga aplicada para los conjuntos de datos sin falla y con falla de diente roto. Este gráfico es importante debido a que permite identificar si los datos están equilibrados o desequilibrados. Cuando los datos están desequilibrados los algoritmos de clasificación se ven afectados debido a que no son capaces de generalizar de manera correcta, siendo la clase minoritaria afectada por la clase mayoritaria. En el gráfico mostrado se puede ver qué los datos mantienen un balance entre ambas clases.

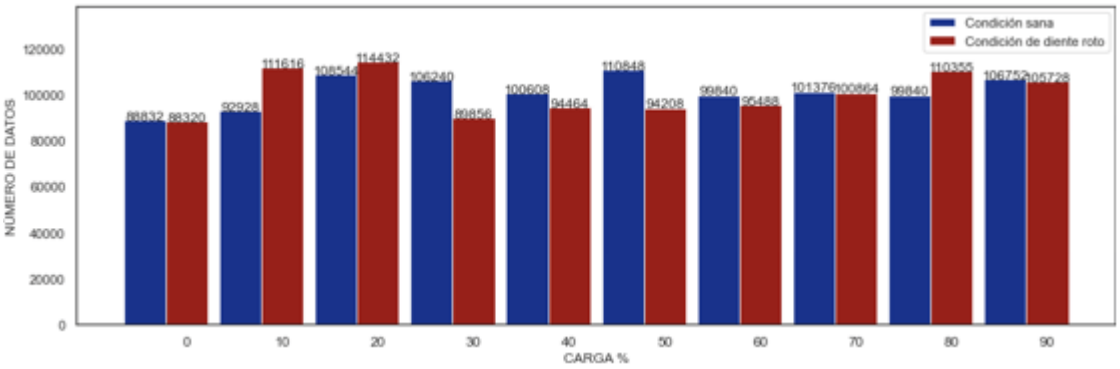


Gráfico 2-2: Número de datos según la carga aplicada

Realizado por: Escobar, José, 2021

En el gráfico 3-2 se puede observar que cada una de las variables de entrada de los conjuntos de datos sin falla y con falla de diente roto aparentemente siguen una distribución normal, sin embargo, al aplicar las pruebas estadísticas de normalidad de Kolmogórov-Smirnov y D'Agostino, ambas rechazaron la hipótesis nula H_0 de que los datos proceden de una distribución normal. En ML los datos que siguen una distribución normal son beneficiosos para la creación de modelos, debido a que facilitan los cálculos matemáticos que realizan los algoritmos.

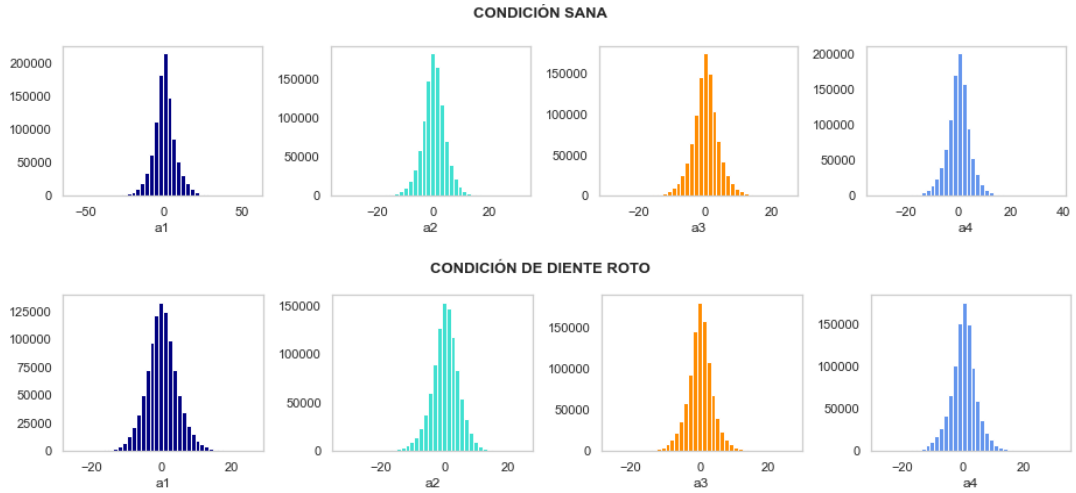


Gráfico 3-2: Distribución de las columnas del conjunto de datos

Realizado por: Escobar, José, 2021

Se consideró aplicar la transformación Box-Cox y la transformación Jeo-Johnjon que son una familia de transformaciones paramétricas utilizadas para hacer que los datos se aproximen a una distribución normal. Para el análisis se descartó la transformación Box-Cox debido a que esta solo admite como entrada únicamente valores positivos y en su lugar se aplicó la transformación Jeo-Johnjon, sin embargo, en la fase de entrenamiento los modelos obtuvieron un rendimiento similar independientemente de la aplicación de este método de normalización, por lo que se consideró trabajar con el conjunto de datos original ya que fue con el cual se obtuvo un mejor rendimiento.

2.3.6. Control de valores atípicos

Para controlar la existencia de valores atípicos se aplicó la teoría del rango intercuartílico en la cual se considera un valor atípico aquel que es 1,5 veces el rango intercuartílico mayor que el cuartil Q3 o a su ves 1,5 veces el rango intercuartílico menor que el cuartil Q1. En el gráfico 4-2 se representan los diagramas de cajas de cada una de las columnas de los conjuntos de datos de señales de vibración descritos anteriormente. Cada columna presenta un gran número de valores atípicos por lo tanto se procedió a eliminar los mismos.

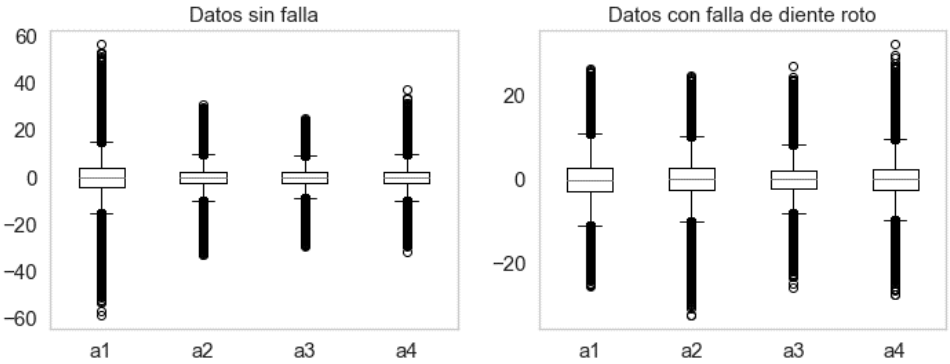


Gráfico 4-2: Conjunto de datos con valores atípicos

Realizado por: Escobar, José, 2021

En el gráfico 5-2 se representa el conjunto de datos después de haber eliminado los valores atípicos, sin embargo, en la fase de entrenamiento los modelos mostraron un mejor rendimiento al utilizar todo el conjunto de datos incluidos aquellos valores identificados como atípicos, por lo tanto, se utilizó todo el conjunto de datos. El mismo análisis se realizó para cada uno de los tres métodos experimentales y en cada uno se obtuvo el mismo resultado. Además, se debe considerar que el algoritmo SVM maneja muy bien los valores atípicos ya que posee un parámetro de ajuste que le permite evitar el sobreajuste causado por la presencia de dichos valores. Por otra parte, dentro de la creación de modelos de ML también se dispone de métodos de escalado como RobustScaler que se utiliza cuando un conjunto de datos presenta valores atípicos.

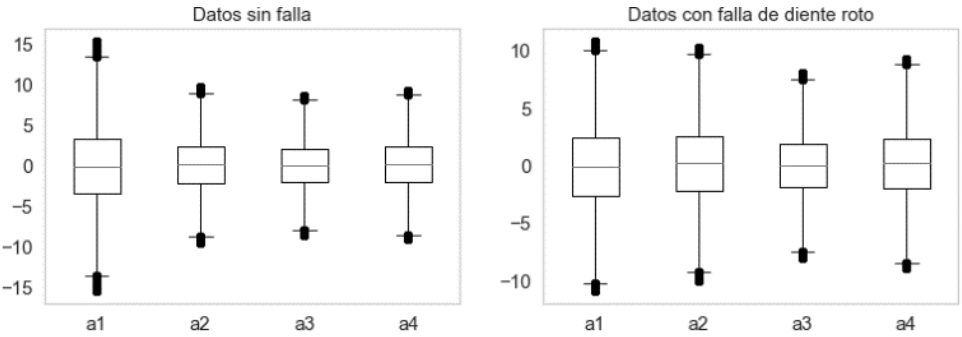


Gráfico 5-2: Conjunto de datos sin valores atípicos

Realizado por: Escobar, José, 2021

2.3.7. Representación en el dominio del tiempo y dominio de la frecuencia

2.3.7.1. Representación gráfica en el dominio del tiempo

Se representó las señales de vibración que han sido recopiladas bajo distintas cargas en el dominio del tiempo y dominio de la frecuencia. En el gráfico 6-2 se representa el conjunto de datos en condición sana y en condición de diente roto en el dominio del tiempo, donde cada dato es un vector que contiene a la base discretizada que forma la onda de aceleración de 0 a 4000 ms, la base de datos tiene uno de estos datos.

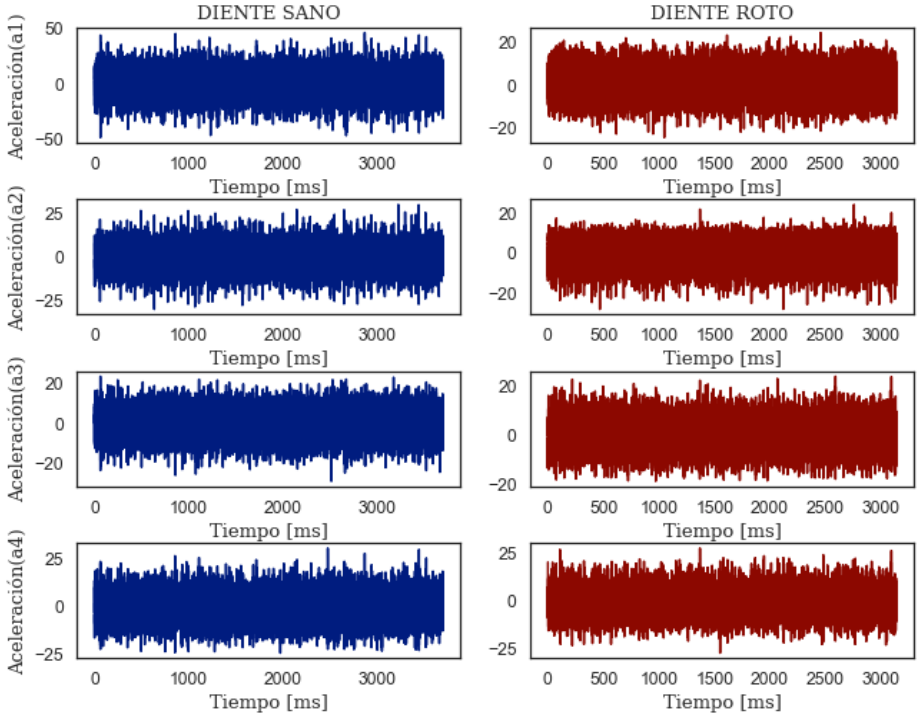


Gráfico 6-2: Gráficas de la aceleración vs tiempo de las señales de vibración

Realizado por: Escobar, José, 2021

2.3.7.2. Representación gráfica en el dominio de la frecuencia

En el gráfico 7-2 se representan los espectros de frecuencias del conjunto de datos sin falla con color azul y los espectros de frecuencias del conjunto de datos con falla de diente roto con color rojo. De acuerdo con la carta de Charlotte un diente roto excitara la frecuencia natural del engranaje (f_n) con bandas laterales a su velocidad de giro. En el gráfico mostrado se puede ver como el espectro de color rojo genera picos importantes y aunque no se conoce con certeza cuales son los armónicos y la frecuencia natural del engranaje, este gráfico permite únicamente apreciar y conocer visualmente la forma de los espectros de frecuencia para las señales de vibración sin falla y con falla de diente roto del conjunto de datos utilizado.

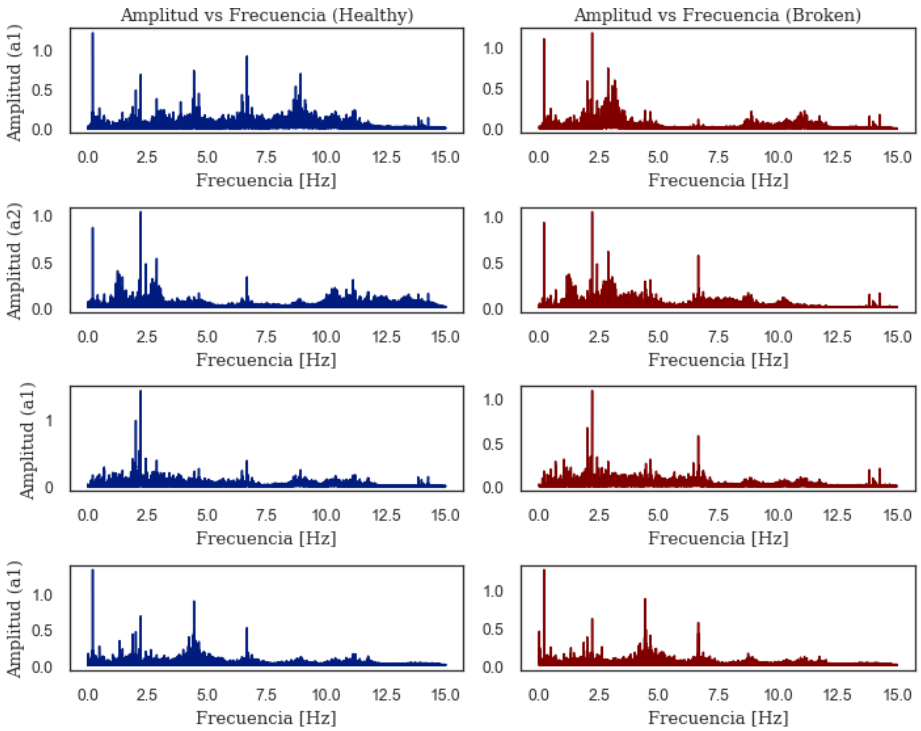


Gráfico 7-2: Gráficos de la amplitud vs frecuencia de las señales de vibración

Realizado por: Escobar, José, 2021

2.4. Fase III-Extracción de características

La extracción de características para los tres métodos experimentales propuestos se llevó a cabo utilizando la librería TSFEL, una librería muy completa lanzada en el año 2020 que permite calcular y extraer fácilmente características de los dominios de tiempo y frecuencia de manera automática. Se eligió esta librería, debido a que permite personalizar y combinar características de varios dominios mediante la edición de un diccionario de configuración (Barandas et al., 2020).

```
import tsfel

cfg_file = tsfel.get_features_by_domain()               # Se extraerán todas las características.
cgf_file = tsfel.get_features_by_domain("statistical")  # Se extraerán todas las características del dominio estadísticas.
cgf_file = tsfel.get_features_by_domain("temporal")     # Se extraerán todas las características del dominio temporal.
cgf_file = tsfel.get_features_by_domain("spectral")     # Se extraerán todas las características del dominio espectral.
```

Figura 12-2: Librería de extracción de características TSFEL

Realizado por: Escobar, José, 2021

La librería descrita admite como parámetros entrada el conjunto de datos, el tamaño de ventana que se aplicará para calcular y extraer las características y la frecuencia de muestreo. Para el ME1 se aplicó un tamaño de ventana de 20, mientras que para el ME2 y ME3 un tamaño de ventana de 30. La frecuencia de muestreo para los tres modelos es de 30Hz. En la tabla 2-2 se indica el conjunto de características que serán extraídas del conjunto de datos tanto en el dominio del tiempo como en el dominio de la frecuencia.

Tabla 2-2: Características del dominio del tiempo y dominio de la frecuencia

Características en el dominio del tiempo		Características en el dominio de la frecuencia	
Media	$T1=\frac{1}{N}\sum_{i=1}^{N} x_i$	Frecuencia media	$F1=\frac{\sum_{k=1}^{K} X(k)}{K}$
Varianza	$T2=\frac{1}{N}\sum_{i=1}^{N} (x_i - T1)^2$	Varianza de frecuencia	$F2=\frac{\sum_{k=1}^{K}(X(k)-F1)^2}{K-1}$
Deviación estándar	$T3=\sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - T1)^2}$	Asimetría de frecuencia	$F3=\frac{\sum_{k=1}^{K}(X(k)-F1)^3}{K(\sqrt{(F2)^3}}$
Raíz media cuadrada	$T4=\sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i)^2}$	Curtosis de frecuencia	$F4=\frac{\sum_{k=1}^{K}(X(k)-F1)^4}{K(F2)^4}$
Curtosis	$T5=\frac{N\sum_{i=1}^{N}(x_i - T1)^4}{\left[\sum_{i=1}^{N}(x_i - T1)^2\right]^2}$	Frecuencia máxima	$F5=\max(P(k))$, k=1…M.
Asimetría	$T6=\frac{N\sum_{i=1}^{N}(x_i - T1)^3}{T3^3}$	Potencia total	$F6=\sum_{K=1}^{M} P(k)$
Entropía de Shannon	$T7=-\sum_{i=1}^{N} x_i^2 * log\,(x_i)^2$	Entropía espectral	$F7=-\sum_{k=1}^{K-1} P_n(k)\log_2[P_n(k)]$ Donde P_n es la energía total normalizada: $P_n=\frac{X(k)}{\sum_{k=1}^{K} X(k)}$
Desviación absoluta media	$T8=\frac{\sum_{i=1}^{N}\lvert(x_i)^2 - T1\rvert}{N}$		
Centroide	$T9=\frac{\sum_{i=0}^{N} t_i * x_i^2}{\sum_{i}^{N}(x_i)^2}$		
Rango intercuartílico	$T10=Q_3-Q_1$		
Valor máximo	$T11=\max(x_i)$		
Valor mínimo	$T12=\min(x_i)$		
Mediana	$T13=\frac{x_{\frac{N}{2}} + x_{\frac{N}{2}+1}}{2}$		
Suma de diferencias absolutas	$T14=\sum_{i=0}^{N-1}\lvert\Delta x_i\rvert$		
Desviación absoluta mediana	$T15=\text{Median}(\lvert x - \text{median}(x_i)\rvert)$		
Valor pico-pico	$T16=\max(x_i) - \min(x_i)$		
Diferencia absoluta media	$T17=\frac{1}{N-1}\sum_{i=1}^{N-1}\lvert x_{i+1} - x_i\rvert$		
Diferencia media	$T18=\frac{1}{N-1}\sum_{i=1}^{N-1}(x_{i+1} - x_i)$		
Área bajo la curva	$T19=\sum_{i=0}^{N} t_i - t_{i-1} * \frac{x_i + x_{i-1}}{2}$		

Fuente: (Sánchez, 2018)

Realizado por: Escobar, José, 2021

2.5. Métodos experimentales

Debido a la dimensión del conjunto de datos obtenido la cual es de 1015808 para el conjunto de datos en condición saludable y 1005311 para el conjunto de datos en condición de diente roto, fue posible llevar a cabo la creación de tres modelos predictivos con el fin comprobar la efectividad del método propuesto y demostrar que se pueden obtener los mismos resultados o similares utilizando diferentes conjuntos de datos, ya sea si se dispone únicamente de datos de un solo sensor o de varios sensores o a su vez datos registrados bajo una cierta carga o un amplio rango de cargas. En cada método experimental se explica el procedimiento de las fases III, IV y V descritas en el gráfico 1-2 y los resultados de estas. El rendimiento de cada modelo será evaluado principalmente en función de las métricas de evaluación de la matriz de confusión como son, la exactitud, precisión, sensibilidad y especificidad. También se evaluará el rendimiento utilizando el análisis de validación cruzada, curvas de aprendizaje y curva ROC.

2.5.1. Método experimental uno (ME1)

Para el desarrollo del ME1 se utilizó el conjunto de datos de señales de vibración registrados por cuatro acelerómetros bajo una carga de 50%. El conjunto de datos descrito tiene una dimensión de 110848 muestras para un estado de condición saludable y 94208 muestras para un estado de condición de diente roto. En la figura 13-2 se muestra el conjunto de datos de señales de vibración para un estado de condición saludable. Al conjunto de datos se le agregó una columna denominada carga para especificar la carga aplicada de 50%.

	a1	a2	a3	a4	carga
0	2.144160	-1.958210	-0.190533	-4.584750	50.0
1	-9.920150	-7.475190	1.794680	-7.472510	50.0
2	-1.330590	0.751472	-3.557400	0.328149	50.0
3	7.761710	-1.498460	-1.764630	10.991900	50.0
4	-0.714011	-0.164771	9.650560	8.970950	50.0
...	...	...	...	...	...
110843	3.245040	-2.692110	2.714350	-0.639589	50.0
110844	1.347960	-2.303540	2.583290	-1.174450	50.0
110845	0.003272	-3.241480	-1.423700	-0.610475	50.0
110846	-0.031329	-3.361870	-1.493090	-2.147170	50.0
110847	1.031390	-0.380219	-2.235860	-0.474274	50.0

110848 rows × 5 columns

Figura 13-2: Datos sin falla y carga de 50%

Realizado por: Escobar, José, 2021

En la figura 14-2 se muestra el conjunto de datos de señales de vibración registrados por cuatros

acelerómetros bajo una carga de 50% para un estado de condición de diente roto de engranaje. Al conjunto de datos se le agregó una columna denominada carga para especificar la carga aplicada de 50%.

	a1	a2	a3	a4	carga
0	-3.93468	6.55216	-1.237980	20.310300	50.0
1	2.40285	9.99438	-3.242650	8.313200	50.0
2	6.24273	-3.17577	-0.686974	-4.193820	50.0
3	-3.99411	-14.14480	3.845360	-5.658600	50.0
4	1.76551	-6.46492	5.096230	-9.001360	50.0
...	...	...	...	...	...
94203	-1.46153	2.78470	-1.151670	-4.537800	50.0
94204	-3.87952	0.85365	6.423880	-5.474920	50.0
94205	1.73315	4.06958	5.364500	0.216419	50.0
94206	8.97857	10.06520	1.552720	3.656620	50.0
94207	6.15826	5.58468	2.412330	2.038780	50.0

94208 rows × 5 columns

Figura 14-2: Datos con falla y carga de 50%

Realizado por: Escobar, José, 2021

2.5.2. Fase III-Extracción de características (ME1)

La extracción de características para los conjuntos de datos descritos se realizó por separado. En total se extrajo 26 características por cada columna de la variable acelerómetro obteniéndose en total 104 características. Además, se aplicó un tamaño de ventana de 20 a partir del cual se obtuvo 10252 submuestras de un total de 205056 muestras. Por último, al conjunto de datos sin falla se le asigno la clase negativa 0 y al conjunto de datos con falla de diente roto la clase positiva 1 en la columna falla. Las características extraídas de ambos conjuntos de datos se concatenaron en un solo archivo como se muestra en la figura 15-2.

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute deviation	...	3_Root mean square	3_Skewness	3_Spectral entropy
0	3.089500	0.384194	1.0	10.347798	-0.839346	16.13430	0.188574	13.333333	0.678641	6.972570	...	7.524775	0.410972	0.28321
1	3.822627	0.254412	1.0	17.501360	-1.091989	11.85680	0.110701	13.333333	0.075423	8.799153	...	8.057351	-0.294834	0.42605
2	2.977592	0.355117	1.0	9.742185	-0.816881	14.28810	0.221902	13.333333	1.457016	6.170889	...	6.521623	0.189503	0.35930
3	2.427898	0.249948	1.0	7.849475	-0.626452	11.26140	0.270927	15.000000	0.697798	4.602284	...	4.811258	-0.706948	0.72468
4	2.038543	0.338758	1.0	5.822616	-0.232435	8.82280	0.236482	15.000000	-1.672413	4.038083	...	3.826065	0.907036	0.83504
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10247	0.872752	0.264036	1.0	2.763622	-0.734197	3.37051	0.470541	15.000000	-0.152010	1.529198	...	2.267141	0.087656	0.81500
10248	1.992187	0.408660	1.0	5.362023	1.281796	6.87850	0.409349	13.333333	-0.302956	3.566407	...	5.510908	0.314092	0.64582
10249	3.006687	0.295486	1.0	9.659995	-1.234963	9.82473	0.311170	13.333333	0.532390	4.967042	...	3.355112	0.634064	0.47908
10250	1.945077	0.275813	1.0	5.182773	-0.500194	10.63980	0.161888	15.000000	1.793892	3.553627	...	5.069023	0.873172	0.71981
10251	2.449425	0.222164	1.0	8.118180	-1.208222	8.04652	0.270213	13.333333	0.518729	4.415785	...	3.939686	-0.492471	0.78197

10252 rows × 105 columns

Figura 15-2: Conjunto de características extraídas (ME1)

Realizado por: Escobar, José, 2021

2.5.3. División del conjunto de datos (ME1)

El conjunto de características extraídas se dividió en dos subconjuntos, una parte para entrenar el modelo correspondiente al 70% y otra parte para probar el modelo entrenado correspondiente al 30%. En la tabla 3-2 se muestran los resultados de la división realizada para entrenamiento y prueba. En total se obtuvo 7176 muestras para entrenar el modelo con sus correspondientes etiquetas conocidas y 3076 muestras para probar el modelo entrenado.

Tabla 3-2: División de datos para entrenamiento y prueba (ME1)

Características	**X_Train**	7176
	X_Test	3076
	y_train	7176
	y_test	3076

Realizado por: Escobar, José, 2021

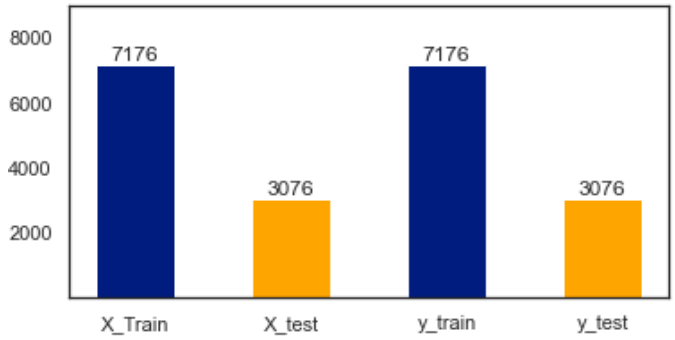


Gráfico 8-2: Datos para entrenamiento y prueba (ME1)

Realizado por: Escobar, José, 2021

2.5.4. Selección de características (ME1)

Dentro de la fase III, como primer paso para la selección de características se creó un mapa de calor para verificar la correlación entre las variables con el propósito de eliminar aquellas altamente correlacionadas ya que brindan la misma información y reducen la velocidad de entrenamiento del modelo. Se eliminaron aquellas características con un umbral de correlación superior a 0,95 evitando eliminar aquellas que pudieran brindar información relevante y por lo tanto ser importantes para el entrenamiento del modelo. Se utilizó un módulo de correlación de la librería TSFEL que calcula la correlación entre pares de características utilizando el método de Pearson y elimina automáticamente características correlacionadas de acuerdo con un umbral de correlación establecido. Posterior a la aplicación del módulo descrito se obtuvo como resultado 82 características de un total de 104. El gráfico 9-2 representa el mapa de calor del conjunto de características originales en el cual se puede identificar que existen áreas claras y oscuras siendo las áreas más claras aquellas características con mayor correlación.

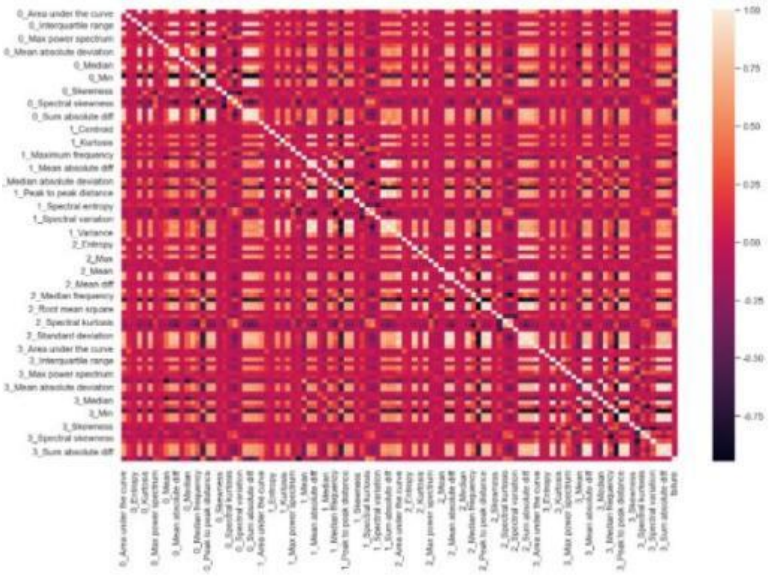


Gráfico 9-2: Mapa de calor original (ME1)

Realizado por: Escobar, José, 2021.

El gráfico 10-2 representa el mapa de calor resultante después de haber eliminado aquellas características con un grado de correlación superior a 0,95. Este proceso es importante debido a que también se aplicará el método de Eliminación Recursiva de Características con Validación Cruzada (RFECV) que da mejores resultados cuando no existen variables correlacionadas.

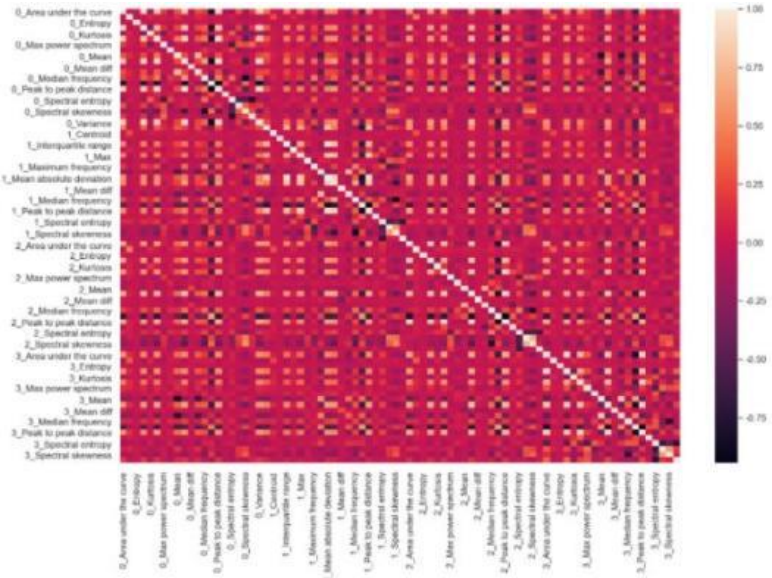


Gráfico 10-2: Mapa de calor resultante (ME1)

Realizado por: Escobar, José, 2021

2.5.5. Selección de características mediante RFECV (ME1)

Para seleccionar las mejores características se utilizó el método de Eliminación Recursiva de

Características con Validación Cruzada (RFECV), el cual es un algoritmo que selecciona las mejores características descartando aquellas con baja importancia hasta obtener un número óptimo de características con el cual se obtenga el mejor rendimiento haciendo uso de la validación cruzada.

```
Estimador=SVC(kernel="linear")
rfecv = RFECV(estimator=Estimador, step=1, cv=StratifiedKFold(4),scoring='accuracy')
rfetrain=rfecv.fit(X1, y)
print('Número óptimo de características :', rfecv.n_features_)
```

Figura 16-2: Aplicación del método RFECV (ME1)

Realizado por: Escobar, José, 2021

Mediante la aplicación del método RFECV se obtuvo como resultado 52 características de un total de 82. En el gráfico 11-2 se muestra el número óptimo de características seleccionadas por el método descrito y su puntuación de validación cruzada.

Gráfico 11-2: Selección de características (ME1)

Realizado por: Escobar, José, 2021

2.5.6. Estandarización de características (ME1)

Como último paso, antes de pasar a la fase IV de entrenamiento se estandarizo el conjunto de características, debido a que los modelos de ML tienen un mejor rendimiento cuando los datos están en una misma escala. En la figura 15-2 las características frecuencia máxima y centroide presentaban una escala diferente esta diferencia podría causar que el centroide deje de ser representativo o importante para el análisis, por lo tanto, se estandarizo el conjunto de características. Para dicho propósito se utilizó el método StandardScaler, mediante el cual, a cada valor de las columnas se le resta la media y se lo divide por la desviación estándar. Después de la estandarización los conjuntos de características para entrenamiento y prueba pasaron a estar en una misma escala como se muestran en la figura 17-2 y figura 18-2.

X_train_1

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean absolute diff	0_Median frequency	0_Skewness	...	3_Centroid	3_Max power spectrum	3_Me
0	-0.837114	-0.363985	-0.726475	-0.391199	-1.003033	-0.500192	1.542922	-0.786604	-1.334566	0.542261	...	-0.974830	-1.630120	0.7870
1	1.370021	0.000131	2.181916	-1.319803	0.398234	0.386359	-0.137201	0.923187	0.159132	-0.784316	...	-0.249386	0.520605	-0.5918
2	-1.346829	-1.142366	-1.421794	1.492426	-1.140536	1.123511	1.542922	-1.078446	-1.334566	1.811196	...	-0.575900	-0.490431	0.4703
3	0.381935	0.089165	0.399005	-0.344558	0.536990	-0.916422	-0.137201	0.204429	0.159132	0.354827	...	0.866605	1.859503	-0.7819
4	1.037368	-0.460013	0.291134	-0.114434	0.359969	1.383517	-0.137201	-0.030343	-1.334566	-0.982468	...	1.495440	-1.480705	-0.7946
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7171	1.616809	-1.078392	1.214731	-0.767307	0.749052	0.219439	-0.137201	0.123482	-1.334566	0.166532	...	-0.230778	0.370054	-0.8946
7172	-0.002170	1.097934	0.747680	0.304147	0.766157	0.377383	-0.137201	1.247078	1.652830	-0.824541	...	1.326539	-0.070946	0.1475
7173	-0.394103	-0.956604	-0.012883	-0.652965	-0.456153	0.647829	1.542922	0.048806	0.905981	0.348589	...	-0.616475	-0.349229	1.2759
7174	-0.634664	-0.803959	-0.583391	0.308335	-0.365591	-0.670343	-0.137201	-0.189245	0.905981	1.419395	...	-1.284535	-1.103111	0.8500
7175	-1.650146	0.714619	-1.237216	-0.155495	-1.336098	-0.685862	1.542922	-1.032809	0.905981	-0.354130	...	1.901859	0.886808	0.1052

7176 rows × 52 columns

Figura 17-2: Conjunto de características para entrenamiento (ME1)

Realizado por: Escobar, José, 2021

X_test_1

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean absolute diff	0_Median frequency	0_Skewness	...	3_Centroid	3_Max power spectrum	3_Me
0	-0.205313	-0.821414	-0.620577	0.635912	0.139574	-1.092236	-0.137201	-0.132963	0.905981	2.141443	...	1.563254	0.455171	0.6188
1	-1.031385	2.529602	-1.199337	6.523125	0.359154	-1.178689	1.542922	-0.613672	0.159132	4.376280	...	0.867638	2.106753	-0.6127
2	0.076278	0.449525	0.567575	-1.198916	-0.240105	-0.752857	-0.137201	0.173244	0.905981	-0.361632	...	0.714638	0.608405	-0.1993
3	-0.743608	1.202710	-0.755837	-0.103634	-0.872367	-0.890895	-0.137201	-1.074916	-1.334566	1.612157	...	0.480257	-1.081250	-0.0304
4	-0.274028	0.701280	-0.468992	-0.108013	0.052421	0.436976	-0.137201	-0.719686	-1.334566	1.174709	...	-0.491229	0.229740	-0.4302
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3071	-0.411494	-0.906819	-0.886903	2.381223	0.343390	-0.141683	-1.817323	-0.284262	0.159132	2.400613	...	-0.483527	0.145107	0.4042
3072	0.209750	0.650684	-0.447703	-0.927033	-0.369781	2.397301	-1.817323	-0.761641	-1.334566	-0.463535	...	0.231527	1.631222	-4.7813
3073	1.440193	-0.106878	1.023101	-0.222937	1.402718	0.741215	-1.817323	1.280697	0.905981	-0.167991	...	0.234764	-0.048149	-0.3971
3074	0.545734	0.806396	0.442619	-0.638583	0.143885	0.284548	-0.137201	0.236951	0.905981	-0.137192	...	-0.182850	2.042965	-0.4763
3075	1.946263	0.900588	2.488804	-0.237547	2.034425	0.350999	-0.137201	2.740350	1.652830	-0.353851	...	0.748427	0.153686	-0.1677

3076 rows × 52 columns

Figura 18-2: Conjunto de características para prueba (ME1)

Realizado por: Escobar, José, 2021

2.5.7. Fase IV-Entrenamiento del modelo (ME1)

Las características estandarizadas se ingresaron al algoritmo clasificador SVM donde se entrenó un modelo utilizando tres funciones kernel que son, el kernel lineal, kernel radial y kernel polinomial. La teoría afirma que cuando se trata crear modelos de ML utilizando datos de maquinaria rotativa el kernel que mejores resultados proporciona es el kernel radial y el objetivo de entrenar el modelo con las tres funciones kernel descritas es justamente para comprobar dicha afirmación. Por otra parte, el entrenamiento se realizó tanto con el conjunto de características originales como con el conjunto de características seleccionadas con el fin de verificar el impacto en el rendimiento del modelo al utilizar características seleccionadas. En la figura 19-2 se muestran los resultados del modelo entrenado, estos resultados reflejan la exactitud con la cual el modelo clasificó las muestras de la clase positiva y muestras de la clase negativa.

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.74	99.76
SVM - Kernel lineal	98.93	98.84
SVM - Kernel radial	99.74	99.76
SVM - kernel polynomial	99.54	99.61

Figura 19-2: Resultados de entrenamiento (ME1)

Realizado por: Escobar, José, 2021

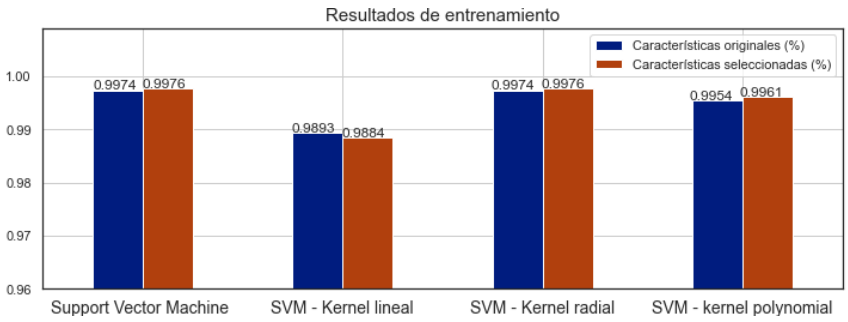


Gráfico 12-2: Resultados de entrenamiento (ME1)

Realizado por: Escobar, José, 2021

El modelo entrenado que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 99,76%. El resultado obtenido permite inferir que el modelo se ajusta bien a los datos de entrenamiento y logra clasificar correctamente las muestras de ambas clases, sin embargo, el verdadero rendimiento del modelo entrenado se obtendrá cuando este sea probado con el conjunto de datos asignado para la prueba. Por otra parte, el modelo entrenado logró un mejor rendimiento utilizando el conjunto de características seleccionadas logrando un margen de mejora de 0,02% con respecto a las características originales. En el gráfico 13-2 se representa la matriz de confusión del modelo entrenado en la cual se puede ver que el modelo clasificó bastante bien muestras de la clase positiva y muestras de la clase negativa y cometió pocos errores de clasificación en su diagonal secundaria.

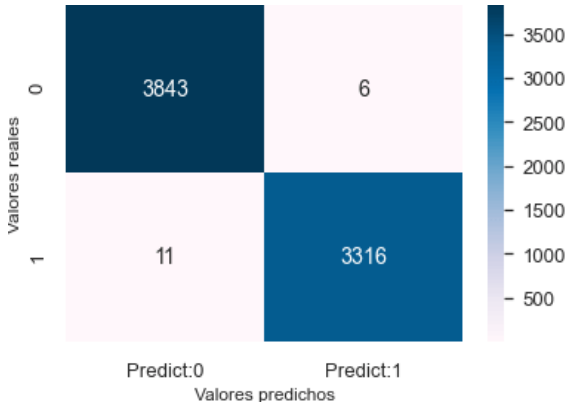


Gráfico 13-2: Matriz de confusión de modelo entrenado (ME1)

Realizado por: Escobar, José, 2021

2.5.8. Resultados de prueba (ME1)

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.12	99.32
SVM - Kernel lineal	98.37	98.24
SVM - Kernel radial	99.12	99.32
SVM - kernel polynomial	98.28	98.54

Figura 20-2: Resultados de prueba (ME1)

Realizado por: Escobar, José, 2021

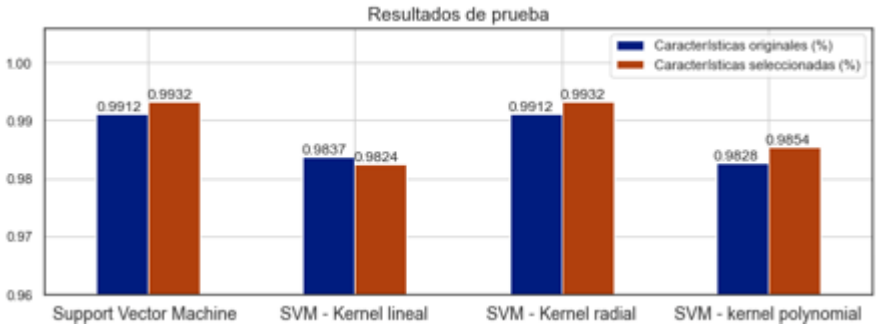


Gráfico 14-2: Resultados de prueba (ME1)

Realizado por: Escobar, José, 2021

Después de la prueba el modelo que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 99,32%. El resultado obtenido es excelente y demuestra que el modelo entrenado fue capaz de generalizar y clasificar correctamente las muestras de prueba asignándolas a la clase correspondiente a la que pertenecen. Por otra parte, el modelo obtuvo un mejor rendimiento utilizando las características seleccionadas logrando un margen de mejora de 0,20% con respecto a las características originales. En el gráfico 15-2 se representa la matriz de confusión del modelo probado.

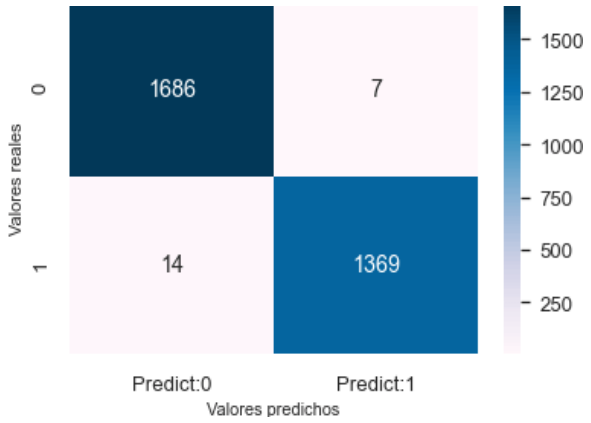


Gráfico 15-2: Matriz de confusión del modelo probado (ME1)

Realizado por: Escobar, José, 2021

En el gráfico 16-2 se muestran las características seleccionadas y su grado de importancia. Estas características son las que más aportaron en la predicción realizada por el modelo. El modelo entrenado con características seleccionadas logró predecir con una exactitud de 99,32%, mientras que el modelo entrenado con características originales predijo con una exactitud de 99,12% existiendo un margen de mejora de 0,20%.

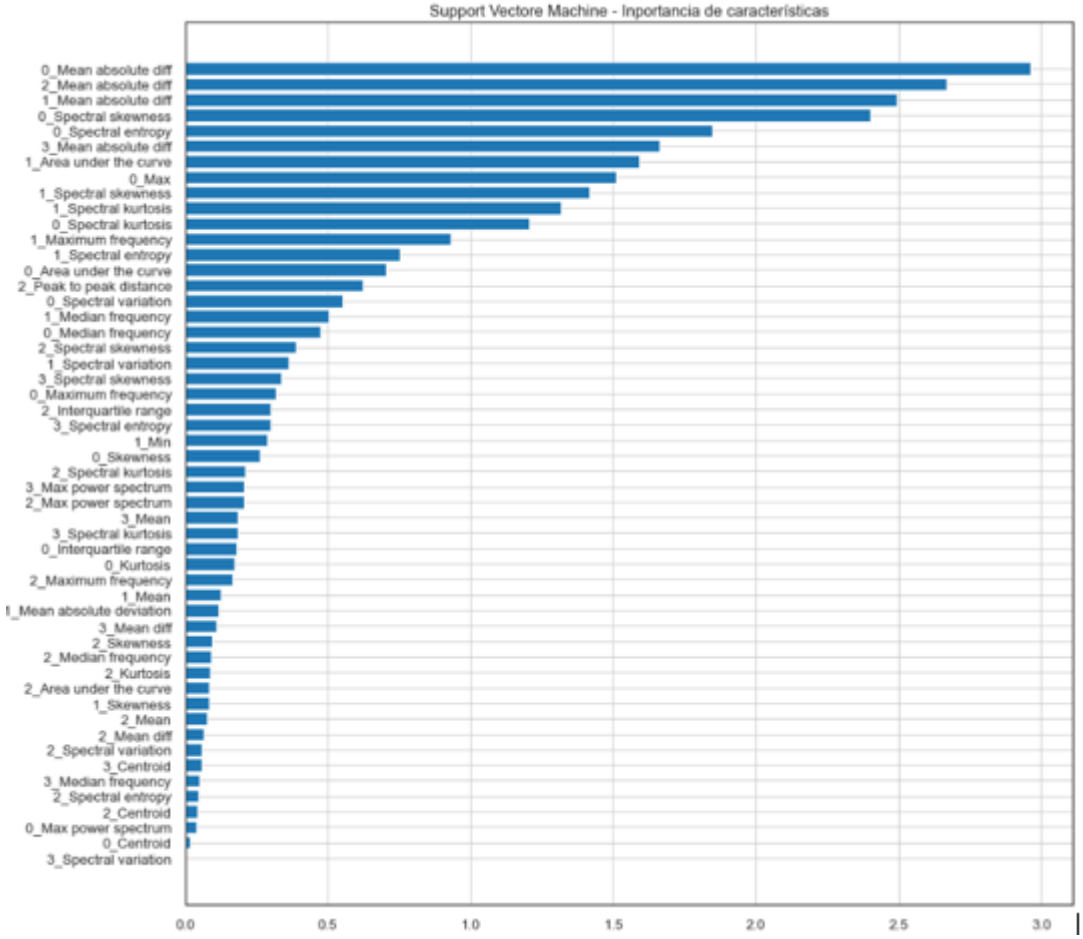


Gráfico 16-2: Características seleccionadas por el método RFECV (ME1)

Realizado por: Escobar, José, 2021

2.6. Método experimental dos (ME2)

Para el desarrollo del ME2 se utilizó todo el conjunto de datos, es decir, los datos de señales de vibración registrados por cuatro acelerómetros y bajo la aplicación de distintas cargas con una variación 0 a 90%. El conjunto de datos descrito posee una dimensión de 1015808 muestras para un estado de condición saludable y 1005311 muestras para un estado de condición de diente roto, disponiéndose de un total de 2021119 muestras para la creación del modelo. En la figura 21-2 se representa el conjunto total de datos de señales de vibración sin falla y con falla de diente roto concatenados en un solo marco de datos.

	a1	a2	a3	a4	carga
0	4.636710	0.516978	-3.205940	1.822410	0.0
1	1.992800	4.184660	-2.740610	2.804360	0.0
2	-3.764110	0.997335	-1.303090	1.836680	0.0
3	-4.558710	6.104330	-1.720690	1.723110	0.0
4	0.575382	0.170980	-0.497967	-1.328950	0.0
...	...	...	...	...	...
2021114	4.434170	-2.037930	-0.546417	1.181320	90.0
2021115	2.903320	-2.820350	-2.486930	0.115640	90.0
2021116	-1.617990	1.278610	-6.159510	-1.376230	90.0
2021117	-2.011220	-4.029450	-0.082050	0.932847	90.0
2021118	-4.437360	-1.970410	3.074570	-3.557610	90.0

2021119 rows × 5 columns

Figura 21-2: Conjunto total de datos (ME2)

Realizado por: Escobar, José, 2021

Como se indica en la figura 21-2 el conjunto de datos para el ME2 tiene un total de 2021119 muestras por cada columna de la variable acelerómetro. Dada la longitud del conjunto de datos para la extracción de características se seleccionó un tamaño de ventana de 30, a partir de lo cual se obtuvo como resultado 67370 submuestras. Cada submuestra corresponde a un cálculo que representa una medida de la señal en el dominio del tiempo y en el dominio de la frecuencia. En la figura 22-2 se representa el conjunto de características extraídas para el ME2.

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute deviation	...	3_Root mean square	3_Skewness	3_Spectra entropy
0	2.425088	0.497873	1.0	5.560467	-0.336024	8.55000	0.341149	13.928571	-0.199320	3.298424	...	2.905705	-0.222020	0.70301
1	2.139392	0.769280	1.0	2.947143	3.630327	13.45210	0.044406	13.928571	0.311557	2.745577	...	2.697244	1.063826	0.77050
2	3.227193	0.523374	1.0	6.370395	-0.183804	7.41089	0.230542	13.928571	-0.782972	4.045817	...	3.365191	-0.232303	0.75326
3	2.419700	0.338981	1.0	5.222493	-0.630353	8.29858	0.179231	12.857143	0.772855	2.992536	...	2.842399	0.608876	0.62639
4	2.485566	0.681217	1.0	4.960278	0.069765	7.76152	0.369654	13.928571	0.346473	3.191558	...	2.495408	0.530618	0.79527
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
67365	2.696374	0.485568	1.0	4.933307	0.108549	10.17000	0.515926	13.928571	1.448651	2.827813	...	5.398334	0.119078	0.72647
67366	5.139555	0.673915	1.0	10.239218	-0.426140	15.40210	0.421572	12.857143	0.475481	6.475218	...	8.224410	0.211426	0.51890
67367	4.022611	0.417820	1.0	7.656767	-0.048745	10.98300	0.248095	12.857143	-1.126454	4.849107	...	6.069938	0.673590	0.70834
67368	3.348222	0.556503	1.0	7.384438	-0.869171	8.97503	0.399281	13.928571	-0.654825	4.035066	...	4.163523	1.036517	0.77349
67369	2.206902	0.520176	1.0	5.241318	-0.954487	5.45863	0.290548	12.857143	0.463703	2.668335	...	2.828307	-0.338186	0.76375

67370 rows × 105 columns

Figura 22-2: Conjunto de características extraídas (ME2)

Realizado por: Escobar, José, 2021

2.6.1. División del conjunto de datos (ME2)

En la tabla 4-2 se muestran los resultados de la división realizada para entrenamiento y prueba. En total se obtuvo 47159 muestras para entrenar el modelo con sus correspondientes etiquetas conocidas y 20211 muestras para probar el modelo entrenado.

Tabla 4-2: Ejemplos usados para entrenamiento y prueba (ME2)

Características	X_Train	47159
	X_Test	20211
	y_train	47159
	y_test	20211

Realizado por: Escobar, José, 2021

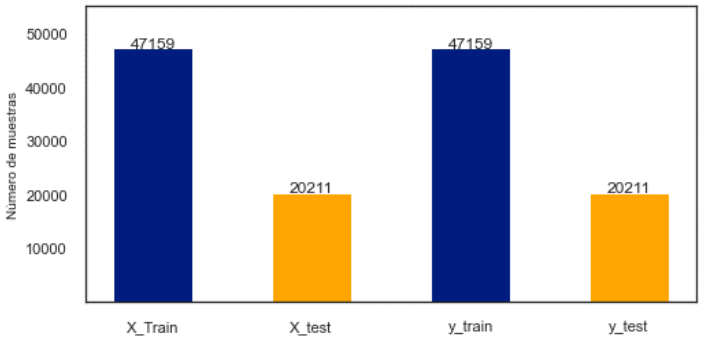


Gráfico 17-2: Datos para entrenamiento y prueba (ME2)

Realizado por: Escobar, José, 2021

2.6.2. Selección de características (ME2)

Del conjunto de características extraídas se eliminó aquellas características con un grado de correlación superior a un umbral de 0,95 obteniéndose como resultado 81 características de un total de 104 como se muestra en la figura 23-2.

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute diff	...	3_Mean diff	3_Median	3_Median frequency
0	-0.945356	-1.141501	0.05515	-0.769628	0.905458	-0.397915	-1.152184	0.983716	0.238863	-0.693184	...	0.493100	0.465810	2.600984
1	-0.472767	0.553569	0.05515	-0.477975	-0.580850	-1.028479	1.712470	-0.315244	0.842388	-0.981211	...	-0.560867	-0.017321	-1.052351
2	-1.248536	-0.986319	0.05515	-1.125196	0.221369	-1.135703	-0.945181	-0.315244	-0.723215	-1.130952	...	-0.074591	-0.596858	-0.139017
3	0.060304	-0.080276	0.05515	0.111959	-0.977623	-0.283066	2.012570	0.983716	0.538603	-0.405862	...	-0.370323	-1.312580	1.687651
4	-0.951203	-1.673555	0.05515	-0.784107	-0.657841	-1.284670	-0.212186	0.983716	-0.441674	-1.143111	...	-0.016885	0.391558	-1.052351
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
47154	-1.054750	0.024676	0.05515	-0.997367	-0.337725	-0.913383	-0.396001	-0.315244	0.185532	-0.788571	...	0.690758	0.212847	1.687651
47155	-0.442984	-1.177167	0.05515	-0.201948	-0.056494	-0.334382	0.474422	-0.315244	-1.759961	-0.085871	...	-0.503687	1.645217	-1.052351
47156	1.158488	0.109006	0.05515	1.091298	-1.125964	-0.086431	0.927315	0.983716	-0.692042	-0.111927	...	-0.821574	0.586734	-0.139017
47157	-0.426443	-0.649194	0.05515	-0.251717	-0.167111	-0.252127	-0.184048	-1.614205	-0.511971	-0.387188	...	-0.287333	1.112116	0.774317
47158	0.545762	-1.566226	0.05515	0.368201	1.522824	1.451746	-0.361016	-1.614205	-0.995393	1.114454	...	2.581535	1.423979	0.774317

47159 rows × 81 columns

Figura 23-2: Reducción de características mediante el método de filtro (ME2)

Realizado por: Escobar, José, 2021

2.6.3. Selección de características mediante RFECV (ME2)

Mediante la aplicación del método RFECV se obtuvo como resultado 63 características de un total de 81. En el gráfico 18-2 se muestra el número óptimo de características seleccionadas por el método descrito y su puntuación de validación cruzada.

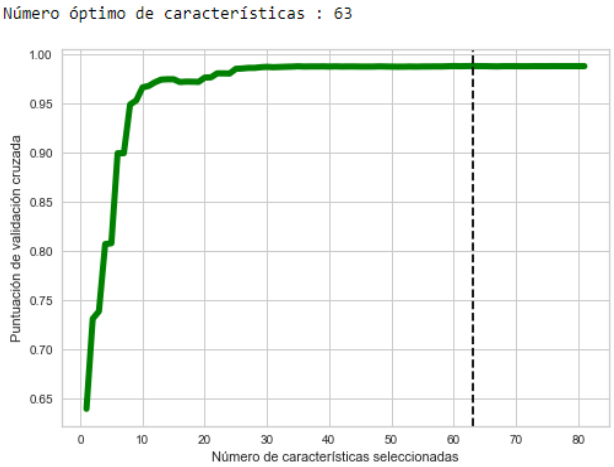


Gráfico 18-2: Selección de características (ME2)

Realizado por: Escobar, José, 2021

Las características seleccionadas se estandarizaron, debido a que se encontraban en escalas diferente. En la figura 24-2 se representa el conjunto de características estandarizadas para entrenamiento, mientras que en la figura 25-2 el conjunto de características estandarizadas para prueba.

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute diff	0_Mean diff	...	3_Kurtosis	3_Max power spectrum	3_Maximu frequency
0	-0.945356	-1.141501	-0.769628	0.905458	-0.397915	-1.152184	0.983716	0.238863	-0.693184	-0.567017	...	-0.992260	-0.683941	1.6806
1	-0.472767	0.553569	-0.477975	-0.580850	-1.028479	1.712470	-0.315244	0.842388	-0.981211	0.284999	...	-0.454656	-1.270048	-1.5880
2	-1.248536	-0.986319	-1.125196	0.221369	-1.135703	-0.945181	-0.315244	-0.723215	-1.130952	-0.296744	...	-0.532389	0.093970	0.5910
3	0.060304	-0.080276	0.111959	-0.977623	-0.283066	2.012570	0.983716	0.538603	-0.405862	0.305097	...	0.550455	-1.241440	1.6806
4	-0.951203	-1.673555	-0.784107	-0.657841	-1.284670	-0.212186	0.983716	-0.441674	-1.143111	0.352262	...	-0.595528	-0.495409	-0.4984
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
47154	-1.054750	0.024676	-0.997367	-0.337725	-0.913383	-0.396001	-0.315244	0.185532	-0.788571	0.245856	...	-0.412125	-0.470185	0.5910
47155	-0.442984	-1.177167	-0.201948	-0.056494	-0.334382	0.474422	-0.315244	-1.759961	-0.085871	0.013804	...	-0.551656	-0.601085	-0.4984
47156	1.158488	0.109006	1.091298	-1.125964	-0.086431	0.927315	0.983716	-0.692042	-0.111927	1.047702	...	-0.696782	0.333600	0.5910
47157	-0.426443	-0.649194	-0.251717	-0.167111	-0.252127	-0.184048	-1.614205	-0.511971	-0.387188	-0.350315	...	-1.013900	0.351726	-0.4984
47158	0.545762	-1.566226	0.368201	1.522824	1.451746	-0.361016	-1.614205	-0.995393	1.114454	-0.450510	...	1.732522	-1.487181	1.6806

47159 rows × 63 columns

Figura 24-2: Conjunto de características para entrenamiento (ME2**)**

Realizado por: Escobar, José, 2021

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute diff	0_Mean diff	...	3_Kurtosis	3_Max power spectrum	3_Maximu frequency
0	-0.842567	1.377602	-0.484074	0.279751	-0.874935	-0.992836	-0.315244	-0.750062	-0.412622	-0.433486	...	-0.739895	-0.193410	-1.5880
1	0.630222	0.509888	-0.099424	-0.489797	-0.180213	1.931984	-1.614205	0.598788	-0.552730	-0.248962	...	0.093199	0.934294	-1.5880
2	-0.342418	1.446695	-0.437105	-0.125066	-0.163155	-1.016238	-0.315244	1.376474	-0.362710	-1.201653	...	0.007847	-0.820470	0.5910
3	0.038208	1.517093	-0.414272	0.025315	0.152983	0.108370	-0.315244	-0.747377	-0.158009	-0.567380	...	-0.783534	1.541637	-0.4984
4	0.890690	0.364202	0.752027	-0.430142	1.276313	-0.104455	-0.315244	1.054207	1.083470	-0.086179	...	-0.949600	0.229753	0.5910
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20206	0.098116	-0.295187	-0.138737	-0.586923	-0.331835	-0.799376	-0.315244	-0.286648	-0.681537	-0.095768	...	-0.872392	0.947261	-0.4984
20207	-1.076577	0.735785	-0.940535	-0.968143	-1.129320	-0.726876	0.983716	0.243398	-1.049041	-0.137564	...	0.446005	-0.546526	0.5910
20208	-0.859853	-1.685764	-1.017730	0.654349	-0.584284	-1.506616	0.983716	-1.288290	-0.522935	-0.034219	...	-0.877058	-0.541444	1.6806
20209	1.184728	0.254845	0.380797	-0.257007	0.696113	-0.873991	-0.315244	1.194287	1.096785	0.226269	...	0.196139	0.015784	0.5910
20210	0.751866	-1.416131	0.724538	-0.725306	-0.605460	0.566998	-0.315244	-0.844101	-0.646786	-0.793732	...	-1.174741	-0.354787	-0.4984

20211 rows × 63 columns

Figura 25-2: Conjunto de características para prueba (ME2)

Realizado por: Escobar, José, 2021

2.7. Fase IV- entrenamiento del modelo (ME2)

En la figura 26-2 se muestran los resultados de entrenamiento del modelo utilizando tres funciones de kernel tanto para las características originales como para las características seleccionadas.

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.95	99.96
SVM - Kernel lineal	99.50	99.50
SVM - Kernel radial	99.95	99.96
SVM - kernel polynomial	99.91	99.93

Figura 26-2: Resultados de entrenamiento (ME2)

Realizado por: Escobar, José, 2021

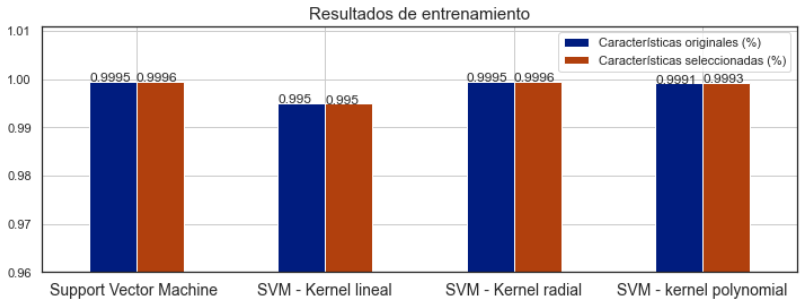


Gráfico 19-2: Resultados de entrenamiento (ME2)

Realizado por: Escobar, José, 2021

El modelo entrenado que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 99,96%. El resultado obtenido permite inferir que el modelo se ajusta bien a los datos de entrenamiento y logra clasificar correctamente las muestras de ambas clases, sin embargo, el verdadero rendimiento del modelo entrenado se obtendrá cuando este sea probado con el conjunto de datos asignado para la prueba.

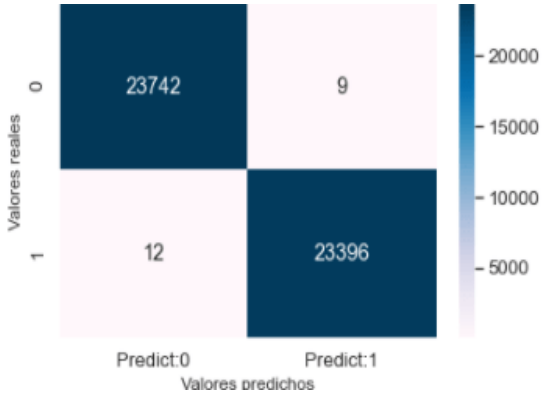


Gráfico 20-2: Matriz de confusión del modelo entrenado (ME2)

Realizado por: Escobar, José, 2021

2.7.1. Resultados de prueba (ME2)

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.80	99.84
SVM - Kernel lineal	99.40	99.42
SVM - Kernel radial	99.80	99.84
SVM - kernel polynomial	99.66	99.66

Figura 27-2: Resultados de prueba (ME2)

Realizado por: Escobar, José, 2021

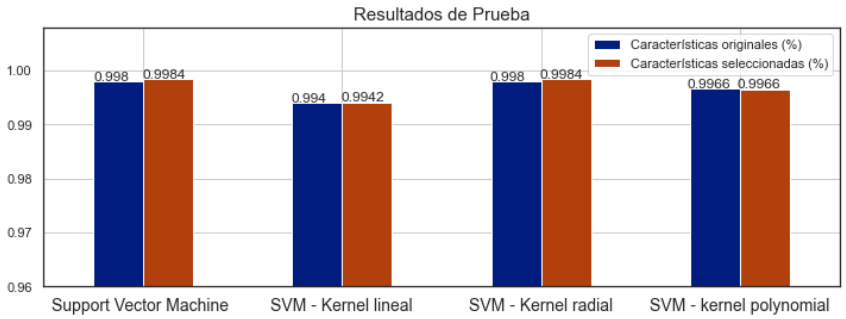


Gráfico 21-2: Resultados de prueba ME2

Realizado por: Escobar, José, 2021

Después de la prueba el modelo que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 99,84%. El resultado obtenido es excelente y demuestra que el modelo entrenado fue capaz de generalizar y clasificar correctamente las muestras de prueba asignándolas a la clase correspondiente a la que pertenecen. Por otra parte, el modelo obtuvo un mejor rendimiento utilizando el conjunto de características seleccionadas logrando un margen de mejora de 0,04% con respecto a las características originales. En el gráfico 22-2 se representa la matriz de confusión del modelo probado y sus resultados.

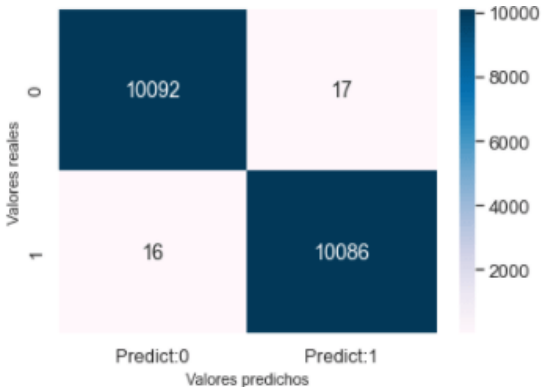


Gráfico 22-2: Matriz de confusión del modelo probado (ME2)

Realizado por: Escobar, José, 2021

\

El modelo entrenado predice bastante bien las muestras de ambas clases y comete pocos errores de clasificación en su diagonal secundaria. En el gráfico 23-2 se indican las características seleccionadas por el método RFECV y su grado de importancia. El modelo entrenado con características seleccionadas logró predecir con una exactitud de 99,84%, mientras que el modelo entrenado con características originales predijo con una exactitud de 99,80% existiendo un margen de mejora de 0,04%.

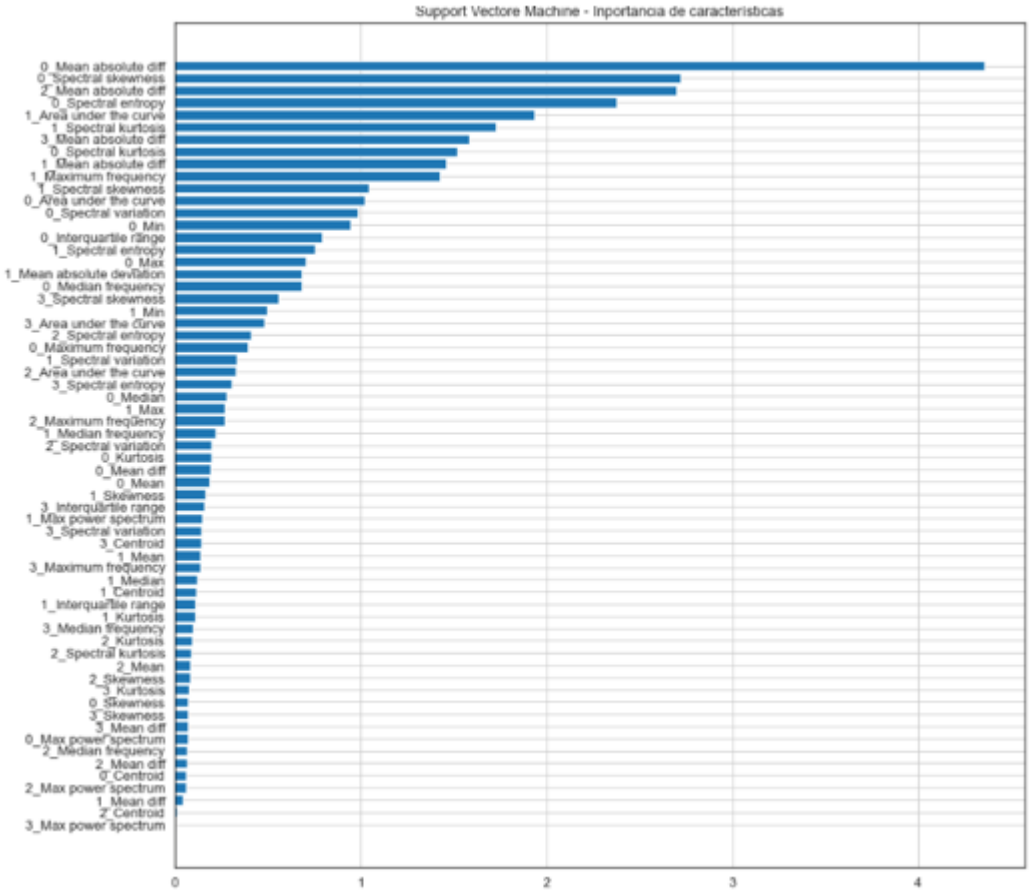


Gráfico 23-2: Características seleccionadas por el método RFECV (ME2)

Realizado por: Escobar, José, 2021

2.7.2. Método experimental tres (ME3)

Para el desarrollo del ME3 se utilizó únicamente los datos de señales de vibración recopilados por un solo acelerómetro del conjunto total de datos. Se mantiene la fase I y la fase II, mientras que para la fase III correspondiente a la extracción de características, mediante la aplicación del método del filtro basado en correlación se obtuvo 20 características de un total de 26 y mediante la aplicación del método RFECV para seleccionar las mejores características se obtuvo que el número óptimo de características es de 12. Las características obtenidas se estandarizaron y posteriormente se ingresaron al algoritmo SVM donde se entrenó el modelo.

2.8. Fase de entrenamiento (ME3)

En la figura 28-2 se muestran los resultados de entrenamiento del modelo utilizando tres funciones de kernel tanto para el conjunto de características originales como para el conjunto de características seleccionadas. La métrica de evaluación utilizada es la exactitud.

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	98.85	98.81
SVM - Kernel lineal	94.71	94.66
SVM - Kernel radial	98.85	98.81
SVM - kernel polynomial	97.89	97.65

Figura 28-2: Resultados de entrenamiento (ME3)

Realizado por: Escobar, José, 2021

El modelo entrenado que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 98,81%. El resultado obtenido permite inferir que el modelo se ajusta bien a los datos de entrenamiento y logra clasificar correctamente las muestras de ambas clases, sin embargo, el verdadero rendimiento del modelo entrenado se obtendrá cuando este sea probado con el conjunto de datos asignado para la prueba. Por otra parte, el modelo tuvo un mejor rendimiento al entrenarse con el conjunto de características originales logrando un margen de mejora de 0,04% con respecto al conjunto de características seleccionadas. En el gráfico 24-2 se representa la matriz de confusión del modelo entrenado, en la cual se puede ver que el modelo clasificó bastante bien muestras de la clase positiva y muestras de la clase negativa y cometió pocos errores de clasificación en su diagonal secundaria.

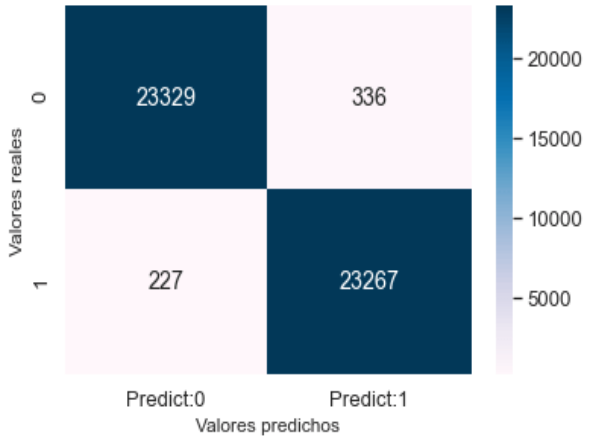


Gráfico 24-2: Matriz de confusión del modelo entrenado (ME3)

Realizado por: Escobar, José, 2021

2.8.1. Resultados de prueba (ME3)

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	98.78	98.79
SVM - Kernel lineal	94.91	94.82
SVM - Kernel radial	98.78	98.79
SVM - kernel polynomial	97.71	97.56

Figura 29-2: Resultados de prueba (ME3)

Realizado por: Escobar, José, 2021

Después de la prueba el modelo que obtuvo el mejor rendimiento fue SVM con la función kernel de base radial, el cual obtuvo una exactitud de 98,79%. El resultado obtenido es excelente y demuestra que el modelo entrenado fue capaz de generalizar y clasificar correctamente las muestras de prueba asignándolas a la clase correspondiente a la que pertenecen. Por otra parte, el modelo entrenado presenta un mejor rendimiento cuando es probado con el conjunto de características seleccionadas logrando un margen de mejora de 0,01% con respecto al conjunto de características originales. En el gráfico 25-2 se muestra la matriz de confusión del modelo probado.

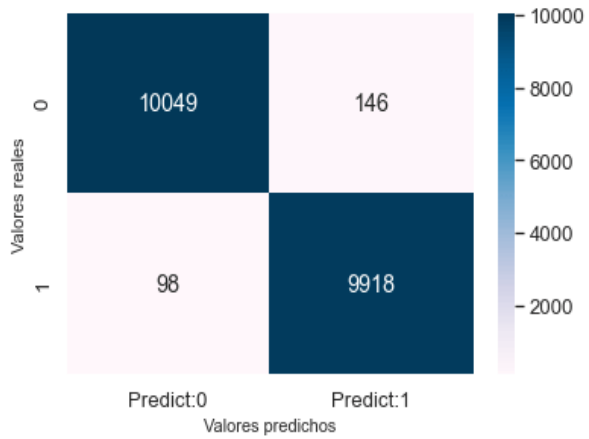


Gráfico 25-2: Matriz de confusión del modelo probado (ME3)

Realizado por: Escobar, José, 2021

El modelo entrenado predijo bastante bien las muestras de ambas clases y cometió pocos errores de clasificación en su diagonal secundaria. En el gráfico 26-2 se indican las características seleccionadas por el método RFECV y su grado de importancia. El modelo entrenado con el conjunto de características seleccionadas logró predecir con una exactitud de 98,79%, mientras que el modelo entrenado con el conjunto de características originales predijo con una exactitud de 98,78% existiendo un margen de mejora de 0,01%.

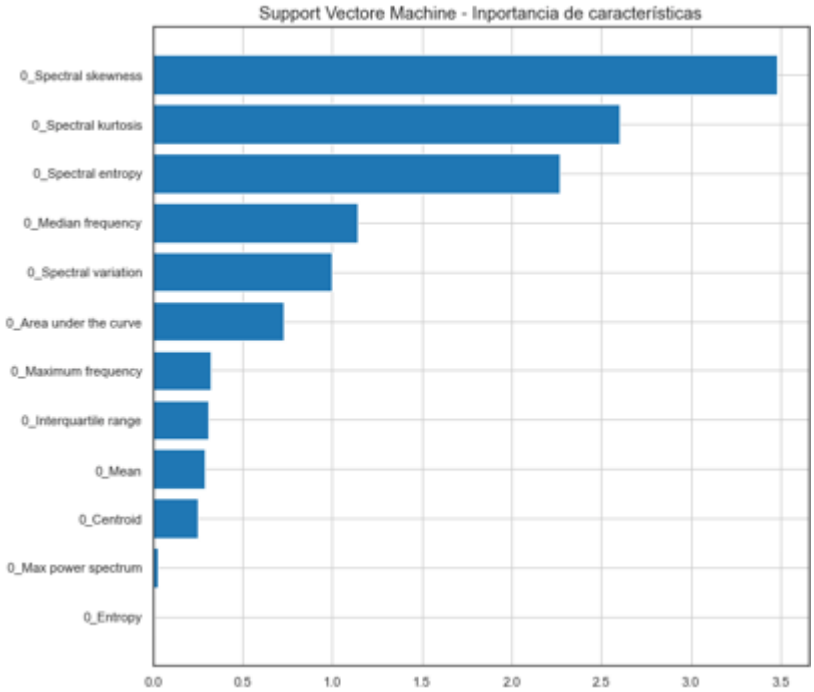


Gráfico 26-2: Características seleccionadas por el método RFECV (ME3)

Realizado por: Escobar, José, 2021

2.8.2. Optimización de hiperparámetros

Para optimizar los hiperparámetros de cada modelo se utilizó el método de búsqueda de cuadrícula mediante validación cruzada. En la figura 30-2 se muestra la lista de hiperparámetros utilizados. La lista de hiperparámetros se estableció únicamente para la función kernel radial ya que fue con esta función que se logró el mejor resultado de entrenamiento y prueba de cada uno de los tres modelos.

```
folds = KFold(n_splits = 5, shuffle = True, random_state = 42)
hyper_params = [ {'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 0.5 ,5,10], 'C': [0.1,0.5,1,3,5,10,15, 25, 50, 100, 1000]}]

model= SVC(kernel="rbf")
model_cv = GridSearchCV(estimator = model, param_grid = hyper_params,scoring = 'recall',cv = folds,verbose = 1,
                        return_train_score=True,)
model_cv.fit(X_train_1, y_train_1)
y_pred_op = model_cv.predict(X_test_1)
gscores_recall = model_cv.cv_results_
best_score = model_cv.best_score_
best_hyperparams = model_cv.best_params_
print(" El mejor puntaje es de {0} :hiperparámetros seleccionados  {1}".format(best_score, best_hyperparams))
```

Figura 30-2: Optimización de hiperparámetros mediante búsqueda de cuadrícula

Realizado por: Escobar, José, 2021

En la tabla 5-2 se muestran las combinaciones de hiperparámetros encontradas por el método aplicado. Para el ME1 los hiperparámetros encontrados fueron C= 5 y gamma = 0,01. Para el ME2 los hiperparámetros encontrados fueron C=3 y gamma= 0,01 y para el ME3 los hiperparámetros encontrados fueron C=1 y gamma= 0,1.

Tabla 5-2: Resultados de optimización de hiperparámetros C y gamma

ME1	ME2	ME3
C=5 gamma=0,01	C=3 gamma=0,01	C=1 gamma=0,1

Realizado por: Escobar, José, 2021

En la tabla 6-2 se muestran los resultados obtenidos por cada modelo entrenado utilizando hiperparámetros optimizados. Los modelos que mejoraron su rendimiento son los obtenidos en el ME1 y ME3, mientras que para el ME2 no se logró una mejora en su rendimiento, sin embargo, aun sin la optimización de hiperparámetros cada modelo presentó un excelente rendimiento lo cual se evidencia en los resultados obtenidos por cada una de las métricas de evaluación de la matriz de confusión descritas en la tabla.

Tabla 6-2: Comparación de resultados de optimización de hiperparámetros C y gamma

	ME1		ME2		ME3	
	Antes	**Después**	**Antes**	**Después**	**Antes**	**Después**
Exactitud	99,32%	99,41%	99,84%	99,84%	98,79%	98,80%
Precisión	99,49%	99,64%	99,83%	99,82%	98,54%	98,62%
Sensibilidad	98,99%	99,06%	99,84%	99,85%	99,02%	98,96%
Especificidad	99,59%	99,70%	99,83%	99,82%	98,57%	98,64%
Puntuación F	99,24%	99,35%	99,84%	99,84%	98,78%	98,78%

Realizado por: Escobar, José, 2021

CAPÍTULO III

3. MARCO DE RESULTADOS Y DISCUSIÓN DE LOS RESULTADOS

3.1. Análisis de resultados del método experimental uno (ME1)

Al conjunto de datos utilizado para entrenar el modelo se le aplicó una validación cruzada de diez iteraciones para estimar el rendimiento real del modelo entrenado y comprobar que los resultados obtenidos en la prueba no son debido de la aleatoriedad con la cual los datos fueron divididos para entrenamiento y prueba y a su vez para conocer con que precisión el modelo será capaz de generalizar datos nuevos.

```
clf= SVC(kernel="rbf",C=5,gamma=0.01,random_state=42)
cv_scores = cross_validate(estimator= clf, X= X_train_1, y= y_train_1, scoring= ('precision'), cv= 10,
                                                                    return_train_score = True)
cv_scores = pd.DataFrame(cv_scores)
scores=cv_scores*100
Puntajes = scores.drop( ['fit_time','score_time'], axis=1)
print(f"Precisión Promedio: {(cv_scores.mean())*100}")
print(f"Desviación estandar: {cv_scores.std()}")
```

Figura 1-3: Aplicación de validación cruzada al conjunto de datos

Realizado por: Escobar, José, 2021

Tabla 1-3: Resultados de validación cruzada (ME1)

Número de iteraciones	Puntuación de validación cruzada
1	99,10%
2	100%
3	99,10%
4	99,10%
5	99,70%
6	98,22%
7	99,10%
8	98,10%
9	99,10%
10	99,40%
Precisión promedio	**99,19%**
Desviación estándar	**0,0046**

Realizado por: Escobar, José, 2021

Mediante validación cruzada se obtuvo una precisión promedio de 99,19% la cual es una estimación más fiable de la calidad del modelo y una desviación estándar de 0,0046 que indica baja varianza en los resultados con respecto a la media. Debido a que el valor de la precisión promedio es similar al valor de la precisión obtenida por el modelo probado la cual fue de 99,64% se comprueba que el modelo no produce resultados al azar, por el contrario, es fiable y tendrá un rendimiento similar para cualquier conjunto de datos de prueba.

3.1.1. Curvas de aprendizaje del ME1

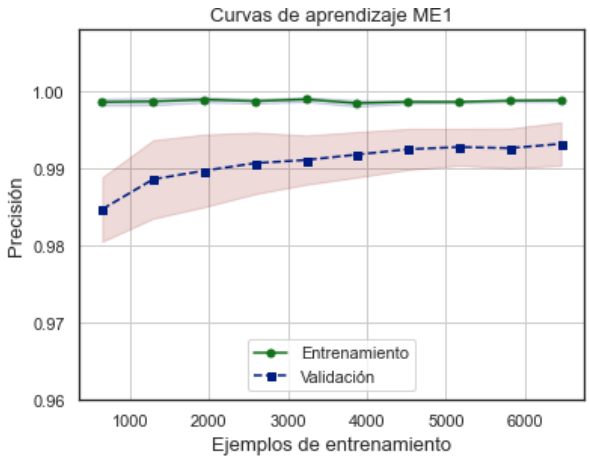


Gráfico 1-3: Curvas de aprendizaje del ME1

Realizado por: Escobar, José, 2021

En el gráfico 1-3 la curva de entrenamiento se mantiene casi constante para todo el conjunto de datos mostrando alta precisión y bajo sesgo, mientras que la curva de validación mejora su precisión a medida que se incrementa el número de muestras de entrenamiento y posiblemente seguirá mejorando si se aumentan más muestras debido a que la curva mantiene una pendiente positiva. Además, a partir de las 4000 muestras el modelo ya no gana mucha precisión, por lo tanto, agregar más muestras para mejorar su rendimiento no es recomendable. El análisis realizado a partir de las curvas de aprendizaje indica que para el conjunto de datos utilizado el modelo presenta bajo sesgo y baja varianza el cual es el objetivo que se desea alcanzar en un modelo predictivo, por lo tanto, el rendimiento del modelo creado se puede considerar como excelente.

3.1.2. Resultados obtenidos mediante optimización de hiperparámetros (ME1)

En la tabla 2-3 se muestran los resultados obtenidos por las métricas de evaluación de la matriz de confusión para el modelo entrenado y probado con parámetros normales e hiperparámetros optimizados, así como el margen de mejora obtenido.

Tabla 2-3: Resultados de la optimización de hiperparámetros (ME1)

Mejor modelo	Mejores Hiperparámetros	Métricas de evaluación	Parámetros normales	Hiperparámetros optimizados	Margen de mejora
SVM- kernel radial optimizado	C=5 Gamma=0,01	Exactitud	99,32%	99,41%	0,09%
		Precisión	99,49%	99,64%	0,15%
		Sensibilidad	98,99%	99,06%	0,07%
		Especificidad	99,59%	99,70%	0,11%
		Puntuación F	99,24%	99,35%	0,11%

Realizado por: Escobar, José, 2021

Mediante la optimización de hiperparámetros el modelo mejoró su exactitud en un 0,09% y su precisión en un 0,15%. El modelo entrenado clasifica correctamente los datos de prueba a la clase correspondiente a la que pertenecen con mejor exactitud y precisión. El modelo también mejoró su sensibilidad y especificidad lo que le permitió cometer menos errores de clasificación y de esa manera mejorar su rendimiento y capacidad predictiva. En el gráfico 2-3 se representa el margen de mejora obtenido por cada métrica de evaluación de la matriz de confusión.

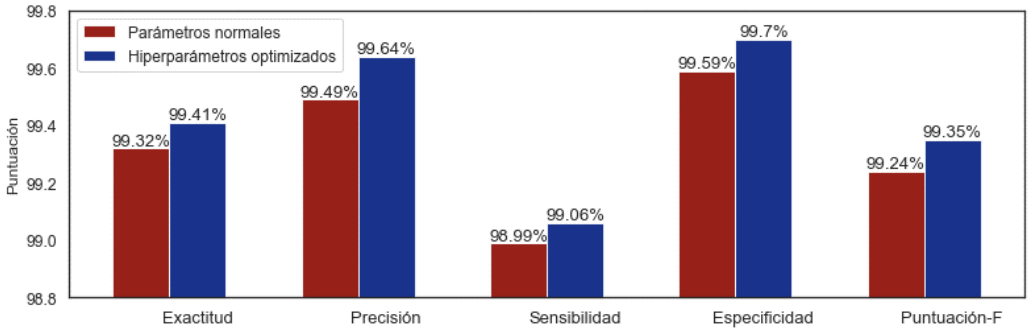


Gráfico 2-3: Margen de mejora con hiperparámetros optimizados (ME1)

Realizado por: Escobar, José, 2021

3.1.3. Análisis de resultados de la matriz de confusión para el ME1

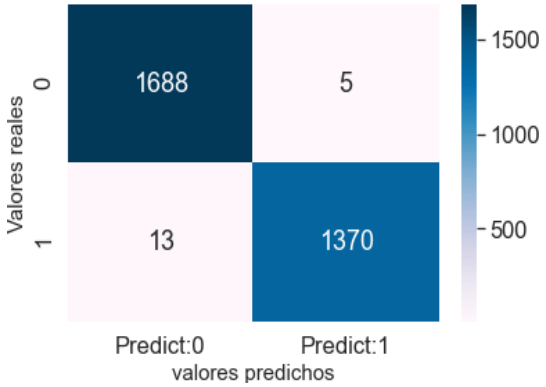


Gráfico 3-3: Matriz de confusión ME1

Realizado por: Escobar, José, 2021

Tabla 3-3: Resultados de la matriz de confusión para ME1

Clase negativa (0)	**VN**=1688	**FP**=5	**Total**=1693
Clase positiva (1)	**FN** =13	**VP**=1370	**Total**=1383
	Predicciones negativas (0)	**Predicciones positivas (1)**	**Total** = 3076

Realizado por: Escobar, José, 2021

- **VN**: el modelo clasificó y predijo correctamente que 1688 muestras corresponden a señales de vibración sin falla.
- **VP**: el modelo clasificó y predijo correctamente que 1370 muestras corresponden a señales de vibración con falla de diente roto.

- **FP:** 5 muestras corresponden a señales de vibración sin falla, pero el modelo las clasificó incorrectamente como señales de vibración con falla de diente roto.
- **FN:** 13 muestras corresponden a señales de vibración con falla de diente roto, pero el modelo las clasificó incorrectamente como señales de vibración sin falla.

El modelo entrenado se probó con un total de 3076 muestras de las cuales 1693 pertenecen clase 0 es decir sin falla y 1383 pertenecen a la clase 1 es decir con falla de diente roto. Para la clase 0 el modelo predijo correctamente que 1688 muestras corresponden a señales de vibración sin falla, mientras que 5 muestras fueron clasificadas erróneamente como falsos positivos. Para esta clase el modelo acertó en un 99,70%, por otra parte, para la clase 1 el modelo predijo correctamente que 1370 muestras corresponden a señales de vibración con falla de diente roto, mientras que 13 muestras fueron clasificadas erróneamente como falsos negativos, para esta clase el modelo acertó en un 99,06%. El modelo predijo mejor la clase negativa que la clase positiva con un margen de diferencia de 0,64%. El modelo clasificó los datos de prueba a la clase correspondiente a la que pertenecen con una exactitud de 99,41% y una precisión de 99,64%. Los resultados obtenidos demuestran que el modelo creado es capaz de clasificar y predecir de manera automática si un conjunto de datos presenta o no un fallo con alta exactitud y precisión.

3.1.4. Curva ROC y área bajo la curva del ME1

En el gráfico 4-3 se representa la curva ROC del modelo y el valor del área bajo la curva. El AUC obtenido es de 0,9998 un valor muy cercano a 1, lo cual demuestra que el modelo no genera resultados aleatorios por el contrario es capaz de distinguir correctamente entre las instancias de la clase positiva y las instancias de la clase negativa con una probabilidad de 99,98 %.

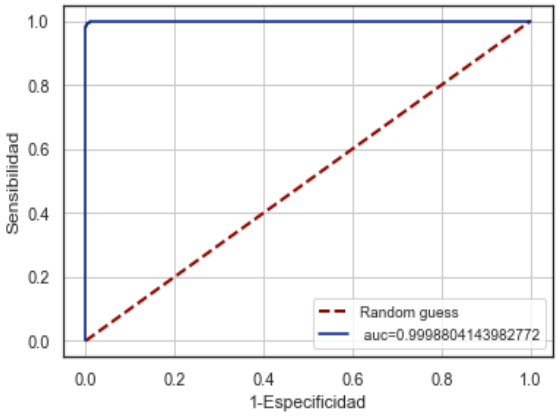


Gráfico 4-3: Curva ROC y AUC (ME1)

Realizado por: Escobar, José, 2021

3.2. Análisis de resultados del método experimental dos (ME2)

Tabla 4-3: Resultados de validación cruzada (ME2)

Número de iteraciones	Puntuación de validación cruzada
1	99,83%
2	99,96%
3	99,62%
4	99,96%
5	99,79%
6	99,87%
7	99,91%
8	99,66%
9	99,62%
10	99,87%
Precisión promedio	**99,81%**
Desviación estándar	**0,0013**

Realizado por: Escobar, José, 2021

Mediante validación cruzada se obtuvo una precisión promedio de 99,81% la cual es una estimación más fiable de la calidad del modelo y una desviación estándar de 0,0013 que indica baja varianza en los resultados con respecto a la media. Debido a que el valor de la precisión promedio es similar al valor de la precisión obtenida por el modelo probado la cual fue de 99,82% se comprueba que el modelo no produce resultados al azar, por el contrario, es fiable y tendrá el mismo rendimiento o similar para cualquier conjunto de datos de prueba.

3.2.1. Curva de aprendizaje del ME2

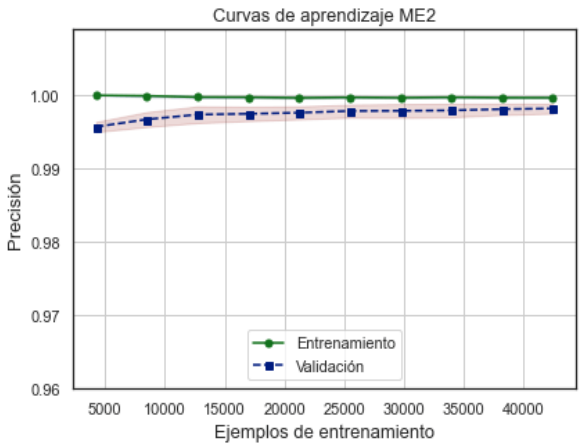


Gráfico 5-3: Curvas de aprendizaje del ME2

Realizado por: Escobar, José, 2021

En el gráfico 5-3 se representan las curvas de aprendizaje del ME2 en una escala ampliada del eje y, mediante está representación se puede observar que el rendimiento del modelo es excelente. La curva de entrenamiento presenta alta precisión y bajo sesgo para todo el conjunto de datos, mientras que la curva de validación mejora su precisión a medida que se incrementa el número

de muestras de entrenamiento, mostrando una ligera varianza en el rango de 0 a 15000 muestras aproximadamente y a partir de este punto el modelo ya no gana mucha precisión. El modelo tiene un buen rendimiento incluso para 5000 muestras este mantiene una precisión por encima del 99% mostrando un bajo error de sesgo y baja varianza, el cual es el objetivo que se desea alcanzar en un modelo predictivo, por lo tanto, el rendimiento del modelo creado se puede considerar como excelente.

3.2.2. Resultados obtenidos mediante optimización de hiperparámetros (ME2)

Tabla 5-3: Resultados de la optimización de hiperparámetros (ME2)

Mejor modelo	Mejores Hiperparámetros	Métricas de evaluación	Parámetros normales	Hiperparámetros optimizados	Margen de mejora
SVM-kernel radial optimizado	C=3 Gamma=0,01	Exactitud	99,84%	99,84%	0%
		Precisión	99,83%	99,82%	-0,01%
		Sensibilidad	99,84%	99,85%	0,01%
		Especificidad	99,83%	99,82%	-0,01%
		Puntuación F	99,84%	99,84%	0%

Realizado por: Escobar, José, 2021

Mediante la optimización de hiperparámetros el modelo entrenado obtuvo el mismo rendimiento al ser probado con el conjunto de datos de prueba. Por una parte, la sensibilidad mejoró en un 0,01%, mientras que las puntuaciones de las demás métricas bajaron en el mismo porcentaje. El modelo mantiene su exactitud original de 99,84%, por lo tanto, los resultados obtenidos no afectan significativamente el rendimiento de este.

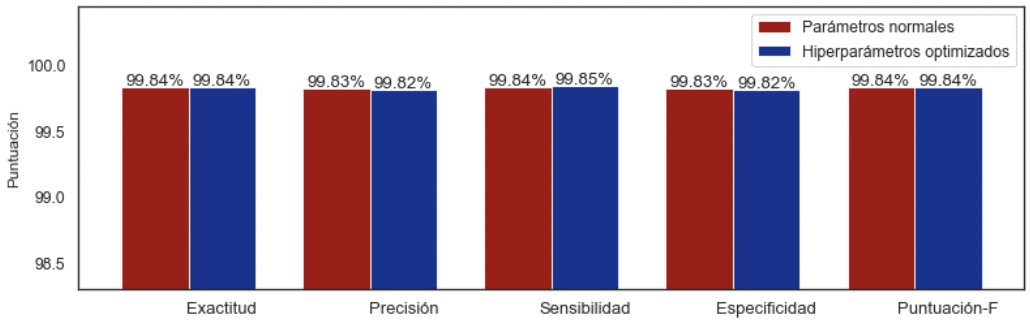


Gráfico 6-3: Margen de mejora con hiperparámetros optimizados (ME2)

Realizado por: Escobar, José, 2021

3.2.3. Análisis de resultados de la matriz de confusión para el ME2

En el gráfico 7-3 se representa la matriz de confusión del modelo creado en el ME2, el mismo que corresponde al modelo SVM con la función kernel de base radial e hiperparámetros optimizados de C=3 y gamma=0,01.

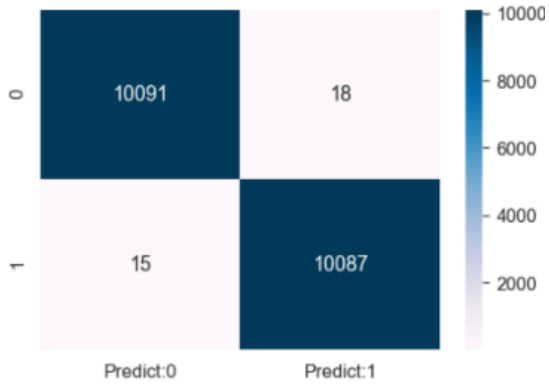


Gráfico 7-3: Matriz de confusión ME2

Realizado por: Escobar, José, 2021

Tabla 6-3: Resultados de la matriz de confusión del ME2

Clase negativa (0)	**VN**=10091	**FP**=18	**Total**=10109
Clase positiva (1)	**FN** =15	**VP**=10087	**Total**=10102
	Predicciones negativas (0)	**Predicciones positivas (1)**	**Total**=20211

Realizado por: Escobar, J, 2021

- **VN**: el modelo clasificó y predijo correctamente que 10091 muestras corresponden a señales de vibración sin falla.
- **VP**: el modelo clasificó y predijo correctamente que 10087 muestras corresponden a señales de vibración con falla de diente roto
- **FP:** 18 muestras corresponden a señales de vibración sin falla, pero el modelo las clasificó incorrectamente como señales de vibración con falla de diente roto.
- **FN:** 15 muestras corresponden a señales de vibración con falla de diente roto, pero el modelo las clasificó incorrectamente como señales de vibración sin falla.

El modelo entrenado se probó con un total de 20211 muestras de las cuales 10109 corresponden a la clase 0 es decir sin falla y 10102 corresponden a la clase 1, es decir con falla de diente roto. Para la clase 0 el modelo predijo correctamente que 10091 muestras corresponden a señales de vibración sin falla, mientras que 18 muestras fueron clasificadas erróneamente como falsos positivos. Para esta clase el modelo acertó en un 99,82%, por otra parte, para la clase 1 el modelo predijo correctamente que 10087 muestras corresponden a señales de vibración con falla de diente roto, mientras que 15 muestras fueron clasificadas erróneamente como falsos negativos, para esta clase el modelo acertó en un 99,85%. El modelo predijo mejor la clase positiva que la clase negativa con un margen de diferencia de 0,03%. El modelo clasificó los datos de prueba a la clase correspondiente a la que pertenecen con una exactitud de 99,84% y una precisión de 99,82%. Los resultados obtenidos demuestran que el modelo creado es capaz de clasificar y predecir de manera automática si un conjunto de datos presenta o no un fallo con alta exactitud y precisión.

3.2.4. Curva ROC y área bajo la curva para el ME2

En el gráfico 8-3 se representa la curva ROC del modelo y el valor del área bajo la curva. El AUC obtenido es de 0,9999 un valor muy cercano a 1, lo cual demuestra que el modelo no genera resultados aleatorios por el contrario es capaz de distinguir correctamente entre las instancias de la clase positiva y las instancias de la clase negativa con una probabilidad de 99,99%.

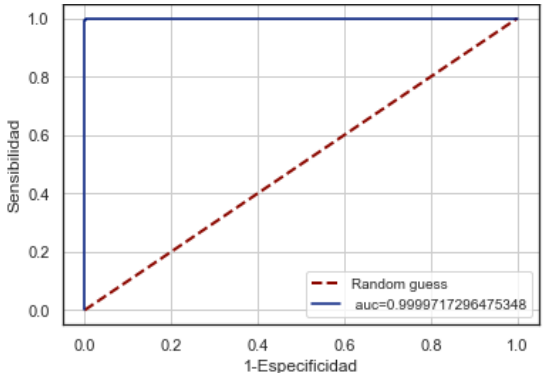


Gráfico 8-3: Curva ROC y AUC (ME2)

Realizado por: Escobar, José, 2021

3.3. Análisis de resultados del método experimental tres (ME3)

Tabla 7-3: Resultados de validación cruzada (ME3)

Número de iteraciones	Puntuación de validación cruzada
1	97,97%
2	98,60%
3	98,56%
4	98,52%
5	98,39%
6	98,35%
7	98,43%
8	98,47%
9	98,81%
10	98,55%
Precisión promedio	**98,47%**
Desviación estándar	**0,0022**

Realizado por: Escobar, José, 2021

Mediante validación cruzada se obtuvo una precisión promedio de 98,47% la cual es una estimación más fiable de la calidad del modelo y una desviación estándar de 0,0022 que indica baja varianza en los resultados con respecto a la media. Debido a que el valor de la precisión promedio es similar al valor de la precisión obtenida por el modelo probado la cual fue de 98,62% se comprueba que el modelo entrenado no produce resultados al azar, por el contrario, es fiable y tendrá un rendimiento similar para cualquier conjunto de datos de prueba.

3.3.1. Curvas de aprendizaje del ME3

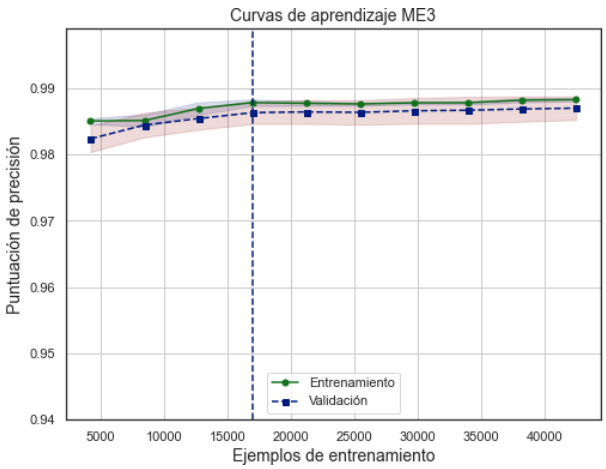


Gráfico 9-3: Curvas de aprendizaje del ME3

Realizado por: Escobar, José, 2021

En el gráfico 9-3 la curva de entrenamiento presenta varianza en su puntuación de precisión en el rango de 0 a 16000 muestras aproximadamente y a partir de este punto la curva se mantiene casi constante mostrando un puntaje alto de precisión para todo el conjunto de datos, mientras que la curva de validación al igual que la curva de entrenamiento presenta varianza en el rango de 0 a 16000 muestras aproximadamente y a partir de este punto el rendimiento del modelo es el mismo. Teniendo en cuanta que ambas curvas mantienen una pendiente positiva, es posible agregar más datos para mejorar el rendimiento del modelo, pero no es recomendable, debido a que el modelo ya no gana mucha precisión.

El análisis realizado a partir de las curvas de aprendizaje para los tres modelos indica que el algoritmo utilizado para crear los modelos se ajusta bastante bien a cualquier tamaño de conjunto de datos presentando bajo sesgo y baja varianza en los resultados. Los modelos del ME1 y ME2 presentan un excelente rendimiento incluso con 3000 muestras estos logran obtener una precisión por encima del 99%, mientras que el modelo del ME3 con 3000 muestras logra una precisión por encima del 98% lo cual permite constatar la teoría la cual menciona que cuando se combinan datos de varios sensores se pueden lograr mejores resultados en el rendimiento de un modelo siendo este el caso del ME1 y ME2.

3.3.2. Resultados obtenidos mediante optimización de hiperparámetros (ME3)

Con la optimización de hiperparámetros el modelo entrenado mejoró su exactitud en 0,01% y su precisión en 0,07%. El modelo no presenta una mejora significativa en su rendimiento debido a que la exactitud casi es la misma y esta mejora únicamente en 0,01%, sin embargo, el modelo

mejoró su precisión lo que quiere decir que se enfoca más en identificar instancias de la clase positiva que realmente son positivas limitando el número de falsos positivos. En la tabla 8-3 se muestra el margen de mejora obtenido por cada métrica de evaluación de la matriz de confusión.

Tabla 8-3: Resultados de la optimización de hiperparámetros (ME3)

Mejor modelo	Mejores Hiperparámetros	Métricas de evaluación	Parámetros normales	Hiperparámetros optimizados	Margen de mejora
SVM con kernel radial optimizado	C=1 Gamma=0,1	Exactitud	98,79%	98,80%	0,01%
		Precisión	98,55%	98,62%	0,07%
		Sensibilidad	99,02%	98,96%	-0,06%
		Especificidad	98,57%	98,64%	0,07%
		Puntuación F	98,78%	98,79%	0,01%

Realizado por: Escobar, José, 2021

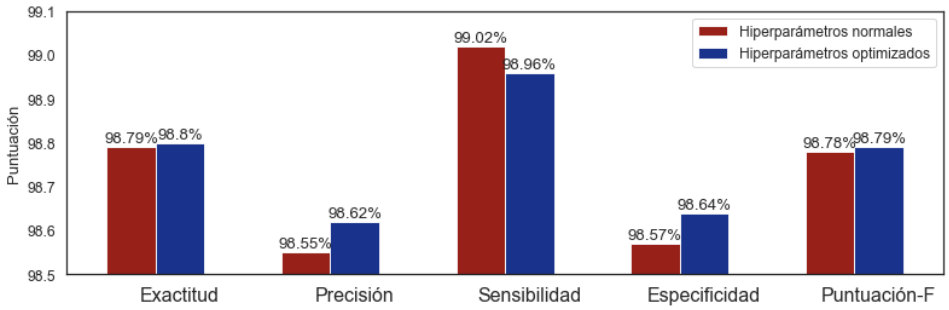


Gráfico 10-3: Margen de mejora con hiperparámetros optimizados (ME3)

Realizado por: Escobar, José, 2021

3.3.3. Análisis de resultados de la matriz de confusión para el ME3

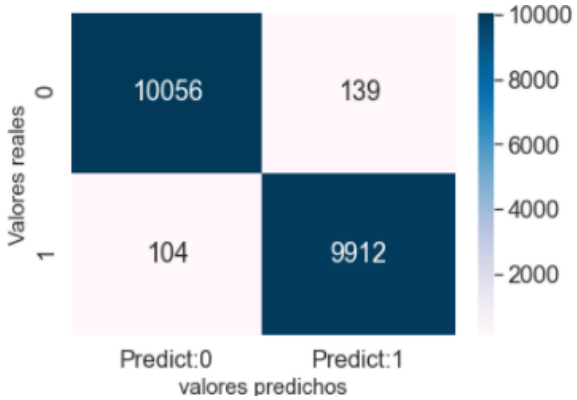


Gráfico 11-3: Matriz de confusión ME3

Realizado por: Escobar, José, 2021

Tabla 9-3: Resultados de la matriz de confusión del ME3

Clase negativa (0)	**VN**=10056	**FP**=139	**Total**=10195
Clase positiva (1)	**FN** =104	**VP**=9912	**Total**=10016
	Predicciones negativas (0)	**Predicciones positivas (1)**	**Total**=20211

Realizado por: Escobar, José, 2021

- **VN**: el modelo predijo que 10056 muestras corresponden a señales de vibración sin falla y realmente si corresponden.
- **VP**: el modelo predijo que 9912 muestras corresponden a señales de vibración con falla de diente roto y realmente si corresponden.
- **FP:** 139 muestras corresponden a señales de vibración sin falla, pero el modelo las clasificó incorrectamente como señales de vibración con falla de diente roto.
- **FN:** 104 muestras corresponden a señales de vibración con falla de diente roto, pero el modelo las clasificó incorrectamente como señales de vibración sin falla.

El modelo entrenado se probó con un total de 20211 muestras de las cuales 10195 corresponden a la clase 0 y 10016 corresponden a la clase 1. Para la clase 0 el modelo predijo correctamente que 10056 muestras corresponden a señales de vibración sin falla, mientras que 139 muestras fueron clasificadas erróneamente como falsos positivos. Para esta clase el modelo acertó en un 98,64%, por otra parte, para la clase 1 el modelo predijo correctamente que 9912 muestras corresponden a señales de vibración con falla de diente roto de engranaje, mientras que 104 fueron clasificadas erróneamente como falsos negativos, para esta clase el modelo acertó en un 98,96%. El modelo predijo mejor la clase positiva que la clase negativa con un margen de diferencia de 0,32%, es decir, el modelo identifica un mayor porcentaje de muestras que corresponden a señales de vibración con de falla de diente roto. El modelo obtuvo una exactitud de 98,80% y una precisión de 98,62%. Estos resultados demuestran que el modelo creado es capaz de clasificar cualquier muestra nueva a la clase correspondiente a la que pertenece con alta exactitud y precisión.

3.3.4. Curva ROC y área bajo la curva para el ME3

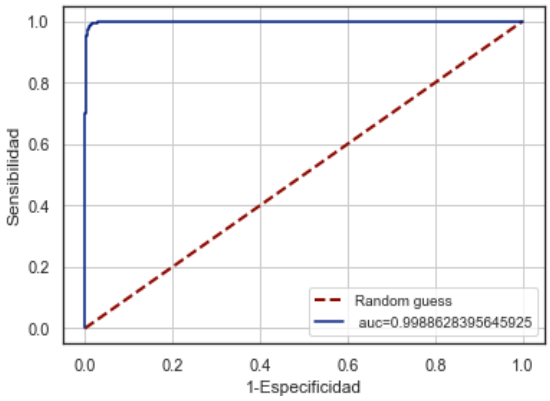


Gráfico 12-3: Curva ROC y AUC (ME3)

Realizado por: Escobar, José, 2021

En el gráfico 12-3 se representa la curva ROC del modelo y el valor del área bajo la curva. El

AUC obtenido es de 0,9988 un valor próximo a 1, lo cual demuestra que el modelo no genera resultados aleatorios por el contrario es capaz de distinguir correctamente entre las instancias de la clase positiva y las instancias de la clase negativa con una probabilidad de 99,88%.

3.4. Prueba de hipótesis

La hipótesis planteada en esta investigación es la siguiente:

Utilizando el método de aprendizaje de máquinas support vector machine se detectan fallas en cajas de engranajes.

Para verificar la hipótesis planteada se aplicó la prueba de permutación a los conjuntos de datos utilizados para crear los tres modelos predictivos. Además, se planteó las siguientes hipótesis estadísticas.

- **H0:** El rendimiento de los modelos es por casualidad.
- **H1:** El rendimiento de los modelos no es por casualidad.

La prueba de permutación se aplicó a los tres modelos creados, debido a que cada uno fue creado con distintos tamaños de datos. Para los tres modelos la permutación aplicada sobre las etiquetas fue de n= 100, mientras que la validación cruzada estratificada fue de k=3. Los valores de permutación y de validación cruzada se establecieron teniendo en cuenta la carga computacional que representa la aplicación de la prueba utilizada, la cual para el modelo del ME2 tuvo una duración aproximada de 85 horas. En la tabla 10-3 se muestran los resultados de la prueba de permutación aplicada a cada modelo.

Tabla 10-3: Resultados de la prueba de permutación

	ME1	ME2	ME3
Tamaño del conjunto de datos.	10252	67380	67380
Puntuación con datos originales.	99,15%	99,84%	99,02%
p-value	0,01	0,01	0,01

Realizado por: Escobar, José, 2022

Los resultados del p-valor de las pruebas de permutación de los tres modelos son menores al umbral de significancia de α=0,05, lo que permite rechazar la hipótesis nula de que el rendimiento de los modelos es por casualidad. A continuación, se representan mediante gráficos de histogramas las puntuaciones de permutación para los tres modelos. En los gráficos la línea vertical representa la puntuación obtenida por los modelos utilizando sus conjuntos de datos originales. La métrica de evaluación utilizada para los tres modelos fue la sensibilidad.

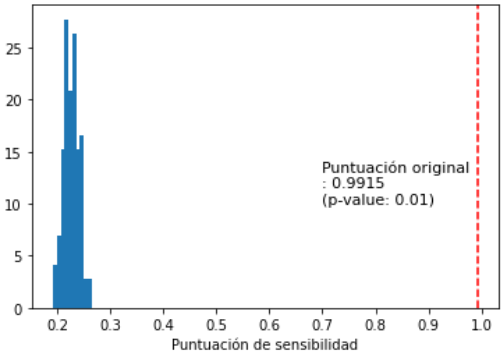


Gráfico 13-3: Prueba de permutacion ME1

Realizado por: Escobar, José, 2022

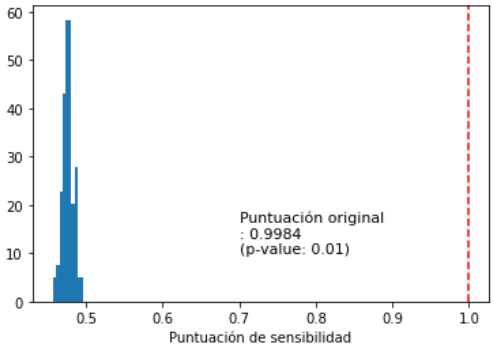


Gráfico 14-3: Prueba de permutación ME2

Realizado por: Escobar, José, 2022

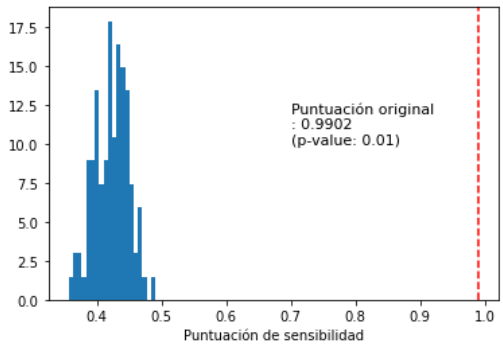


Gráfico 15-3: Prueba de permutación ME3

Realizado por: Escobar, José, 2022

Para los tres modelos la puntuación obtenida con los datos originales es mejor que la puntuación obtenida con los datos permutados. Las puntuaciones obtenidas con datos permutados indican que hay una baja posibilidad de que los resultados obtenidos por los tres modelos sean por casualidad y proporcionan evidencia suficiente de que el conjunto de datos de falla de cajas de engranajes tiene una dependencia entre las características y las etiquetas y los modelo pudieron haber hecho uso de esa dependencia para lograr un buen rendimiento de clasificación, por lo tanto, se acepta la hipótesis alternativa de que el rendimiento de los modelos no es por casualidad y de esta manera se comprueba que los modelos realmente están prediciendo y generando resultados confiables y

son aptos para detectar fallas en cajas de engranajes.

3.5. Comparación con otros clasificadores

Algoritmos	Train Exactitud	Test Exactitud	Precisión	Sensibilidad	F1 score	AUC
SVC	99.76	99.32	99.49	98.99	99.24	99.29
XGBClassifier	100.00	99.28	99.20	99.20	99.20	99.28
RandomForestClassifier	100.00	98.63	98.20	98.77	98.49	98.65
LogisticRegression	98.80	98.28	98.19	97.98	98.08	98.25
KNeighborsClassifier	98.51	97.07	97.50	95.95	96.72	96.97
DecisionTreeClassifier	100.00	96.55	95.38	97.04	96.20	96.60

Figura 2-3: Comparación con otros clasificadores de ML

Realizado por: Escobar, José, 2021

Adicionalmente, se comparó el clasificador SVM con otros clasificadores de aprendizaje de máquinas supervisado los cuales se muestran en la figura 2-3 para ver el rendimiento de estos utilizando la misma base de datos. Como resultado SVM fue el clasificador que obtuvo el mejor rendimiento logrando una exactitud de 99,32% como se muestra en el gráfico 16-3. Por lo tanto, su uso para crear modelos predictivos enfocados a la detección de fallas en cajas de engranajes es recomendable.

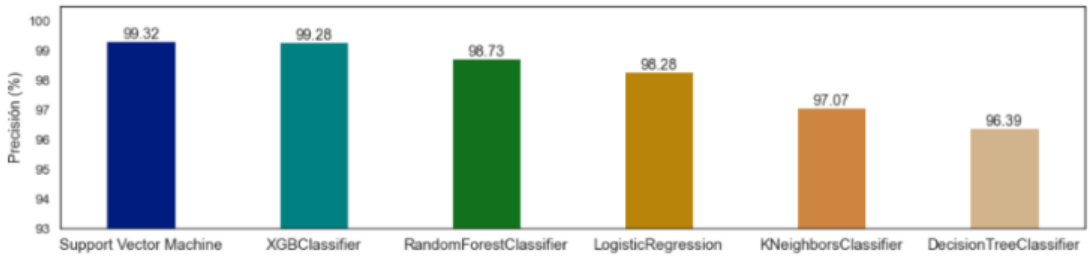


Gráfico 16-3: Comparación de clasificadores de ML

Realizado por: Escobar, José, 2021

Finalmente, los resultados obtenidos por los modelos desarrollados en cada método experimental demuestran que la metodología utilizada en esta investigación la cual se basa en el uso de algoritmos de aprendizaje de máquinas para crear modelos predictivos que realicen una detección automática de fallas es altamente eficiente, por lo tanto, puede ser implementada con toda seguridad en aplicaciones del mundo real como un método de diagnóstico mucho más rápido y preciso.

CONCLUSIONES

Con el desarrollo de la presente investigación se demostró que es posible crear un modelo predictivo basado en datos de señales de vibración para detectar fallas en cajas de engranajes de forma automática. En este sentido, el algoritmo de aprendizaje de máquinas supervisado SVM con una configuración correcta de kernel e hiperparámetros optimizados demostró ser altamente efectivo a la hora de clasificar y predecir correctamente a qué tipo de falla pertenece un nuevo grupo de datos, logrando una exactitud de 99,84% y una precisión de 99,82% para el mejor modelo.

Para los conjuntos de datos de entrenamiento y prueba, el modelo SVM con la función kernel de base radial fue el que proporciono los mejores resultados de clasificación en términos de exactitud, precisión y sensibilidad. Por otra parte, el uso del kernel depende en gran medida de la naturaleza de los datos y en esta investigación se demostró que para el conjunto de datos utilizado la función kernel de base radial es la más apropiada, por lo tanto, puede ser utilizada con toda certeza en la creación de modelos predictivos para predecir fallas de esta naturaleza.

Los modelos no mostraron una mejora significativa después del proceso de selección de características mediante la técnica de RFECV y la selección de los mejores hiperparámetros, sin embargo, sí lo hicieron después de la normalización y reducción de características altamente correlacionadas, por lo tanto, se determinó que el algoritmo SVM se desempeña mucho mejor cuando se entrena con datos que están normalizados y que no son redundantes. Además, se evidencio que este proceso ahorra tiempo y reduce la carga computacional lo que facilita el trabajo cuando se dispone de conjuntos de datos de gran dimensión.

La investigación finalmente mostró el potencial del algoritmo SVM como clasificador para predecir fallas en cajas de engranaje, lo cual se ve reflejado en los resultados obtenidos por las métricas de evaluación de la matriz de confusión y la prueba de hipótesis realizada sobre cada modelo para validar sus resultados. Los modelos predicen correctamente el tipo de fallo con un mínimo margen de error, lo que demuestra que se ajustan bien a los datos y son capaces de generalizar de forma correcta datos nuevos, por lo tanto, son altamente confiables y pueden ser implementados con toda seguridad en el campo del mantenimiento industrial como métodos de detección de fallas automáticos.

RECOMENDACIONES

Se recomienda normalizar el conjunto de datos probando con diferentes técnicas de normalización tales como, StandardScaler, MinMaxScaler y RobustScaler, debido a que el rendimiento de un modelo puede variar dependiendo de la técnica que se utilice.

La prueba de permutación permite evaluar la significancia estadística de los resultados de un algoritmo, sin embrago, esta prueba representa una alta carga computacional, por lo tanto, para conjuntos de datos de gran dimensión se recomienda tener en cuenta el número de permutaciones a ejecutar, así como el número de pliegues de validación cruzada a aplicar.

Para la extracción de características de conjuntos de datos de gran dimensión, si se utiliza la librería TSFEL para calcular y extraer las características se recomienda utilizar un tamaño de ventana mínimo de 15, debido a que esto ayuda a reducir la carga computacional y además se obtienen mejores resultados en términos de rendimiento.

Realizar el mismo proceso de investigación, utilizando una base de datos con distintas clases de fallas y comprobar el rendimiento de SVM frente a una clasificación multiclase.

BIBLIOGRAFÍA

AL-ZOUBI, A; et al. "Evolving Support Vector Machines using Whale Optimization Algorithm for spam profiles detection on online social networks in different lingual contexts". *Knowledge-Based Systems* [en línea], 2018, (Jordania) 153(1), pp. 91-104, [Consulta: 1 noviembre 2021]. ISSN 0950-7051. Disponible en: https://doi.org/10.1016/j.knosys.2018.04.025.

ALI, L; et al. "An Optimized Stacked Support Vector Machines Based Expert System for the Effective Prediction of Heart Failure". *IEEE Access* [en línea], 2019, (China) 7(1), pp. 54007-54014, [Consulta: 20 octubre 2021]. ISSN 2169-3536. Disponible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8684835.

ARIAS, M; et al. "Classifier selection with permutation tests". *Frontiers in Artificial Intelligence and Applications* [en línea], 2017, (España) 300(1), pp. 96-105, [Consulta: 5 enero 2022]. Disponible en: https://arxiv.org/abs/1711.09708.

BARANDAS, M; et al. "TSFEL: Time Series Feature Extraction Library". *SoftwareX* [en línea], 2020, (Portugal) 11(1), pp. 2-7, [Consulta: 20 octubre 2021]. ISSN 2352-7110. Disponible en: https://www.sciencedirect.com/science/article/pii/S2352711020300017.

BHADANE, M; & RAMACHANDRAN, K. "Bearing fault identification and classification with convolutional neural network". *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2017* [en línea], 2017, (India), 2015(1), pp. 1-5, [Consulta: 25 octubre 2021]. Disponible en: https://ieeexplore.ieee.org/document/8074401.

CHEN, Z; et al. "Mechanical fault diagnosis using Convolutional Neural Networks and Extreme Learning Machine". *Mechanical Systems and Signal Processing* [en línea], 2019, (Bélgica) 133(1), pp. 1, [Consulta: 15 noviembre 2021]. ISSN 0888-3270. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S088832701930487X.

CHINGAL, D; et al. "Comparación de señales de vibración y corriente para la detección de la severidad de fallos en engranajes". *BISTUA Revista de la Facultad de Ciencias Básicas* [en línea], 2019, (Colombia) 17(1), pp. 187-195, [Consulta: 20 noviembre 2021]. ISSN 0120-4211. Disponible en: http://revistas.unipamplona.edu.co/ojs_viceinves/index.php/BISTUA/article/view/3147.

DÍAZ, Ana. Análisis y detección de vulnerabilidades en torno al aprendizaje automático (ML) [en línea] (Trabajo de titulación). Universidad Politécnica de Cataluña, España. 2021. pp. 22-23. [Consulta: 6 noviembre 2021]. Disponible en: https://upcommons.upc.edu/handle/2117/352740.

GARCÍA, F; et al. "Classification of failures bearings using machine learning". *Dominio de las ciencias* [en línea], 2021, (Ecuador) 7(4), pp. 70-89, [Consulta: 30 octubre 2021]. ISSN 2477-8818. Disponible en: https://dominiodelasciencias.com/ojs/index.php/es/article/view/2082/4364.

GONZÁLEZ, Lucía. Herramientas Avanzadas De Análisis De Datos De Aplicación En Ingeniería Civil [en línea] (Trabajo de titulación). (Maestria) Universidad de Cantabria, Madrid, España. 2017. pp. 136. [Consulta: 20 noviembre 2021]. Disponible en: https://repositorio.unican.es/xmlui/bitstream/handle/10902/12684/Lucia Moreno González-Paramo.pdf?sequence=1&isAllowed=y.

GOOD, P. *Permutation Test* [en línea]. Huntington Beac-Estados Unidos: Springer, 1994. [Consulta: 5 enero 2022]. Disponible en: https://link.springer.com/book/10.1007/978-1-4757-2346-5#about.

GUALOTO, Deysi. Estimación de número de personas en imágenes de multitudes usando aprendizaje de máquina [en línea] (Trabajo de titulación). Universidad Tecnica del Norte, Ibarra, Ecuador. 2021. pp. 9-11. [Consulta: 28 octubre 2021]. Disponible en: http://repositorio.utn.edu.ec/bitstream/123456789/11444/2/04 MEC 371 TRABAJO GRADO.pdf.

LEI, Y; et al. "Applications of machine learning to machine fault diagnosis: A review and roadmap". *Mechanical Systems and Signal Processing* [en línea], 2020, (China) 138(1), [Consulta: 23 octubre 2021]. ISSN 0888-3270. Disponible en: https://doi.org/10.1016/j.ymssp.2019.106587.

LEI, Y. *Individual intelligent method-based fault diagnosis* [en línea]. China: Elsevier, 2017. [Consulta: 20 noviembre 2021]. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/B9780128115343000032.

LU, P; et al. "A hybrid feature selection combining wavelet transform for quantitative analysis of heat value of coal using laser-induced breakdown spectroscopy". *Applied Physics B: Lasers and Optics* [en línea], 2021, (China) 127(2), pp. 1-11, [Consulta: 10 noviembre 2021]. ISSN 0946-2171. Disponible en: https://link.springer.com/article/10.1007/s00340-020-07556-8.

MANRIQUE, E. "Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo". *Revista Ibérica de Sistemas e Tecnologias de Informação* [en línea], 2020, (España) pp. 586-599, [Consulta: 5 enero 2022]. ISSN 16469895. Disponible en: https://search.proquest.com/openview/c7e24c997199215aa26a39107dd2fe98/1?pq-origsite=gscholar&cbl=1006393.

MOHAMMED, O; & RANTATALO, M. "Gear fault models and dynamics-based modelling for gear fault detection – A review". *Engineering Failure Analysis* [en línea], 2020, (Suecia) 117(1), pp. 4-12, [Consulta: 25 octubre 2021]. ISSN 1350-6307. Disponible en: https://doi.org/10.1016/j.engfailanal.2020.104798.

MORALES, D. "Técnicas de inteligencia artificial aplicadas a problemas de ingeniería civil.". *Revista de Arquitectura e Ingeniería* [en línea], 2017, (Cuba) 11(3), pp. 1-7, [Consulta: 23 octubre 2021]. ISSN 1990-8830. Disponible en: https://www.redalyc.org/pdf/1939/193955164005.pdf.

MÜLLER, A; & GUIDO, S. *Introduction to with Python Learning Machine* [en línea]. Estados Unidos: O'Reilly, 2017. [Consulta: 5 noviembre 2021]. Disponible en: https://www.nrigroupindia.com/e-book/Introduction to Machine Learning with Python (PDFDrive.com)-min.pdf.

OCHOA, L. "Evaluation of classification algorithms using cross validation". *Proceedings of the LACCEI international Multi-conference for Engineering, Education and Technology* [en línea], 2019, (Perú), [Consulta: 15 noviembre 2021]. ISSN 2414-6390. Disponible en: https://www.researchgate.net/publication/335730977_Evaluacion_de_Algoritmos_de_Clasificacion_utilizando_Validacion_Cruzada.

PANDYA, Yogesh. *Datos de diagnóstico de fallas de la caja de cambios* [blog].2018. [Consulta: 20 octubre 2021]. Disponible en: https://data.openei.org/submissions/623.

PISNER, D; & SCHNYER, D. *Support vector machine* [en línea]. Texas-United States: Elsevier, 2020. [Consulta: 28 octubre 2021]. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/B9780128157398000067.

QUINTERO, S. *Aprende Python* [en línea]. 2021. [Consulta: 10 noviembre 2021]. Disponible en: https://aprendepython.es/_downloads/907b5202c1466977a8d6bd3a2641453f/aprendepython.pdf.

RODRÍGUEZ, David. Industrial IoT. Machine learning en la industria 4.0 [en línea] (Trabajo de titulación).Universidad Politécnica de catalunya, España. 2020. pp. 8-24. [Consulta: 15 noviembre 2021]. Disponible en: https://upcommons.upc.edu/handle/2117/336086.

ROJAS, Brian. Una introducción a los modelos de Machine Learning [en línea] (Trabajo de titulación). (Licenciatura) Universidad Autónoma de Puebla, Zaragoza, México. 2020. pp. 77. [Consulta: 5 noviembre 2021]. Disponible en: https://repositorioinstitucional.buap.mx/handle/20.500.12371/10527.

SÁNCHEZ, René. Diagnóstico de fallos en cajas de engranajes con base en la fusión de datos de señales de vibración, corriente y emisión acústica [en línea] (Tarabajo de titulación). (Doctoral) Universidad Pontificia Bolivariana, Medellín, Colombia. 2018. pp. 23-34. [Consulta: 20 noviembre 2021]. Disponible en: https://repository.upb.edu.co/handle/20.500.11912/4020.

SARKAR, D; et al. *Practical Machine Learning with Python* [en línea]. Allahabad-India: Apress Media LLC, 2018. [Consulta: 25 octubre 2021]. Disponible en: https://link.springer.com/book/10.1007/978-1-4842-3207-1.

SHOBHA, G; & RANGASWAMY, S. *Aprendizaje automático* [en línea]. Bengaluru-India: Elsevier B.V., 2018. [Consulta: 25 octubre 2021]. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S0169716118300191?via%3Dihub.

SWAMYNATHAN, M. *Mastering Machine Learning with Python in Six Steps* [en línea]. 2ª ed. Karnataka-India: Apress, 2019. [Consulta: 1 noviembre 2021]. Disponible en: https://link.springer.com/book/10.1007%2F978-1-4842-4947-5.

THARWAT, A. "Parameter investigation of support vector machine classifier with kernel functions". *Knowledge and Information Systems* [en línea], 2019, (Germany) 61(1), pp. 1269-1302, [Consulta: 15 octubre 2021]. ISSN 02193116. Disponible en: https://doi.org/10.1007/s10115-019-01335-4.

VARELA, Christian. Uso de algoritmos de aprendizaje máquina para la clasificación de tráfico de red [en línea] (Trabajo de titulación). (Ingeniería informática) Universidad de Curuña, Facultad de informática,Curuña, España. 2020. pp. 51-69. [Consulta: 15 noviembre 2021]. Disponible en: https://ruc.udc.es/dspace/handle/2183/27116.

VERBRAEKEN, J; et al. "A Survey on Distributed Machine Learning". *ACM Computing*

Surveys [en línea], 2020, (Bélgica) 53(2), pp. 1-33, [Consulta: 5 noviembre 2021]. ISSN 0360-0300. Disponible en: https://dl.acm.org/doi/abs/10.1145/3377454.

VIEIRA, S; et al. *Main concepts in machine learning* [en línea]. Londres-Reino Unido: Elsevier Inc, 2020. [Consulta: 5 enero 2022]. Disponible en: https://www.sciencedirect.com/science/article/pii/B978012815739800002X?via%3Dihub.

WAGNER, M; et al. "Radiomics, machine learning, and artificial intelligence-what the neuroradiologist needs to know". *Neuroradiology* [en línea], 2021, (Canada) 63(1), [Consulta: 9 noviembre 2021]. ISSN 1432-1920. Disponible en: https://doi.org/10.1007/s00234-021-02813-9.

WAYMOND, R. *Artificial Intelligence in a Throughput Model* [en línea]. Boca Ratón- Florida: CRC Press, 2020. [Consulta: 23 octubre 2021]. Disponible en: https://www.taylorfrancis.com/books/mono/10.1201/9780429266065/artificial-intelligence-throughput-model-rodgers-waymond.

WICAKSONO, Ananto; & SUPIANTO, Ahmad. "Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction". *International Journal of Advanced Computer Science and Applications* [en línea], 2018, (Indonesia) 9(12), pp. 263-267, [Consulta: 20 noviembre 2021]. ISSN 21565570. Disponible en: https://pdfs.semanticscholar.org/c573/489f37e7f5b66d9155e8c8b66c1dc4d66d2a.pdf.

ZHANG, X. *Machine Learning* [en línea]. Tsinghua-China: Springer, 2020. [Consulta: 24 octubre 2021]. Disponible en: https://link.springer.com/chapter/10.1007/978-981-15-2770-8_6#citeas.

ANEXOS

ANEXO A: CÓDIGO DE PROGRAMACIÓN DEL PREPROCESAMIENTO DE DATOS.

In [1]:
```
# LIBRERÍAS PARA EL PREPROCESAMIENTO
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
import matplotlib.gridspec as gridspec
from pylab import *
from scipy.stats import normaltest # Prueba de normalidad de D'Agostino's K-squared
from scipy.stats import kstest   # Prueba de normalidad de Kolmogorov-Smirnov
from sklearn.preprocessing import PowerTransformer # Normalizacion de datos método Yeo-Johnson
from scipy import fftpack # Transformada rápida de furier.
```

LECTURA DE DATOS

Información sobre el conjunto de datos

El conjunto de datos se ha registrado utilizando 4 sensores de vibración colocados en cuatro direcciones diferentes y con una variación de carga de '0' a '90' por ciento. Se incluyen dos escenarios diferentes: Condición saludable y Condición de diente roto*. Hay 20 ficheros en total, 10 ficheros de señales en condicion saludable y 10 en condicion de diente roto. Cada fichero corresponde a una carga determinada del 0% al 90% en pasos del 10%.

- **Healthy:** Representa el conjunto de datos de señales de vibración de caja de engranajes en condicion saludable.
- **Broken:** Representa el conjunto de datos de señales de vibración de caja de engranajes con falla de diente roto.
- **a1:** Acelerómetro 1
- **a2:** Acelerómetro 2
- **a3:** Acelerómetro 3
- **a4:** Acelerómetro 4

Datos sin falla.

In [2]:
```
# Realiza la lectura de datos sin falla ubicados en el directorio especificado.
# header=0 significa que tiene los nombres de las columnas en la primera fila del archivo.
dt_h0 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz0.csv",header=0)
dt_h10 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz10.csv",header=0)
dt_h20 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz20.csv",header=0)
dt_h30 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz30.csv",header=0)
dt_h40 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz40.csv",header=0)
dt_h50 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz50.csv",header=0)
dt_h60 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz60.csv",header=0)
dt_h70 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz70.csv",header=0)
dt_h80 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz80.csv",header=0)
dt_h90 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/Healthy/h30hz90.csv",header=0)
```

Datos con falla de diente roto.

In [3]:
```
# Realiza la lectura de datos con falla ubicados en el directorio especificado.
# header=0 significa que tiene los nombres de las columnas en la primera fila del archivo.
dt_b0 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz0.csv",header=0)
dt_b10 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz10.csv",header=0)
dt_b20 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz20.csv",header=0)
dt_b30 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz30.csv",header=0)
dt_b40 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz40.csv",header=0)
dt_b50 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz50.csv",header=0)
dt_b60 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz60.csv",header=0)
dt_b70 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz70.csv",header=0)
dt_b80 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz80.csv",header=0)
dt_b90 = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/dataset/BrokenTooth/b30hz90.csv",header=0)
```

Se agrega una columna para identificar la carga aplicada.

In [4]:
```
# conjuntos de datos en condición saludable
carga = 0
dt_b0['carga'] = carga*np.ones((len(dt_b0.index),1))
carga = 10
dt_b10['carga'] = carga*np.ones((len(dt_b10.index),1))
carga = 20
dt_b20['carga'] = carga*np.ones((len(dt_b20.index),1))
carga = 30
dt_b30['carga'] = carga*np.ones((len(dt_b30.index),1))
carga = 40
dt_b40['carga'] = carga*np.ones((len(dt_b40.index),1))
carga = 50
dt_b50['carga'] = carga*np.ones((len(dt_b50.index),1))
carga = 60
dt_b60['carga'] = carga*np.ones((len(dt_b60.index),1))
carga = 70
dt_b70['carga'] = carga*np.ones((len(dt_b70.index),1))
carga = 80
dt_b80['carga'] = carga*np.ones((len(dt_b80.index),1))
carga = 90
dt_b90['carga'] = carga*np.ones((len(dt_b90.index),1))
```

```
# Conjunto de datos en condición de diente roto
carga = 0
dt_h0['carga'] = carga*np.ones((len(dt_h0.index),1))
carga = 10
dt_h10['carga'] = carga*np.ones((len(dt_h10.index),1))
carga = 20
dt_h20['carga'] = carga*np.ones((len(dt_h20.index),1))
carga = 30
dt_h30['carga'] = carga*np.ones((len(dt_h30.index),1))
carga = 40
dt_h40['carga'] = carga*np.ones((len(dt_h40.index),1))
carga = 50
dt_h50['carga'] = carga*np.ones((len(dt_h50.index),1))
carga = 60
dt_h60['carga'] = carga*np.ones((len(dt_h60.index),1))
carga = 70
dt_h70['carga'] = carga*np.ones((len(dt_h70.index),1))
carga = 80
dt_h80['carga'] = carga*np.ones((len(dt_h80.index),1))
carga = 90
dt_h90['carga'] = carga*np.ones((len(dt_h90.index),1))
```

Concatenación de datos sin falla.

In [5]:
```
# Concatena todos los ficheros en uno solo marco de datos.
# axis = 0: indica que el método se ejecuta hacia abajo a lo largo de cada columna o etiqueta de fila.
Healthy = pd.concat([dt_h0, dt_h10, dt_h20, dt_h30, dt_h40, dt_h50, dt_h60, dt_h70, dt_h80, dt_h90 ],axis=0)
# Guarda el conjunto de datos para el ME1.
dt_h50.to_csv('C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Healthy_carga_50.csv',index=False)
# Guarda el conjunto de datos concatenado para ME2 Y ME3.
Healthy.to_csv('C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Healthy.csv',index=False)
Healthy # muestra el conjunto de datos en condicion saludable.
```

Out[5]:

	a1	a2	a3	a4	carga
0	4.636710	0.516978	-3.205940	1.82241	0.0
1	1.992800	4.184660	-2.740610	2.80436	0.0
2	-3.764110	0.997335	-1.303090	1.83668	0.0
3	-4.558710	6.104330	-1.720690	1.72311	0.0
4	0.575382	0.170980	-0.497967	-1.32895	0.0
...	...	...	...	...	...
106747	0.677448	-3.234410	-1.725990	-3.14302	90.0
106748	-10.575400	7.725400	-2.184010	2.56965	90.0
106749	-4.033290	2.576920	1.468430	2.72891	90.0
106750	1.868670	-5.089400	5.342290	-1.36563	90.0
106751	7.581480	6.205960	-6.121330	11.54830	90.0

1015808 rows × 5 columns

Concatenación de datos con falla.

In [6]:
```
Broken = pd.concat([dt_b0, dt_b10, dt_b20, dt_b30, dt_b40, dt_b50, dt_b60, dt_b70, dt_b80, dt_b90],axis=0)
# Guarda el conjunto de datos para el ME1.
dt_b50.to_csv('C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Broken_carga_50.csv',index=False)
# Guarda el conjunto de datos concatenado para ME2 Y ME3
Broken.to_csv('C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Broken.csv',index=False)
Broken # muestra el conjunto de datos en condicion de diente roto de engranaje.
```

Out[6]:

	a1	a2	a3	a4	carga
0	2.350390	1.454870	-1.867080	-2.055610	0.0
1	2.452970	1.400100	-2.825100	0.984487	0.0
2	-0.241284	-0.267390	0.793540	0.605882	0.0
3	1.130270	-0.890918	0.696969	0.613068	0.0
4	-1.298140	0.980479	-1.130560	-0.346971	0.0
...	...	...	...	...	...
105723	4.434170	-2.037930	-0.546417	1.181320	90.0
105724	2.903320	-2.820350	-2.486930	0.115640	90.0
105725	-1.617990	1.278610	-6.159510	-1.376230	90.0
105726	-2.011220	-4.029450	-0.082050	0.932847	90.0
105727	-4.437380	-1.970410	3.074570	-3.557610	90.0

1005311 rows × 5 columns

Conteo de datos de acuerdo a la carga aplicada.

In [7]:
```
# Datos en condición saludable.
freq1 = Healthy.groupby(['carga']).count() # Cuenta el número de datos en cada columna.
print(freq1)
```

```
          a1      a2      a3      a4
carga
0.0     88832   88832   88832   88832
10.0    92928   92928   92928   92928
20.0   108544  108544  108544  108544
30.0   106240  106240  106240  106240
40.0   100608  100608  100608  100608
50.0   110848  110848  110848  110848
60.0    99840   99840   99840   99840
70.0   101376  101376  101376  101376
80.0    99840   99840   99840   99840
90.0   106752  106752  106752  106752
```

In [8]:
```
# Datos en condición de diente roto (falla).
freq1 = Broken.groupby(['carga']).count()  # Cuenta el número de datos en cada columna.
print(freq1)
```

```
          a1      a2      a3      a4
carga
0.0     88320   88320   88320   88320
10.0   111616  111616  111616  111616
20.0   114432  114432  114432  114432
30.0    89856   89856   89856   89856
40.0    94464   94464   94464   94464
50.0    94208   94208   94208   94208
60.0    95488   95488   95488   95488
70.0   100864  100864  100864  100864
80.0   110335  110335  110335  110335
90.0   105728  105728  105728  105728
```

ANÁLISIS ESTADÍSTICO DESCRIPTIVO.

El análisis estadístico descriptivo permite conocer las propiedases de los datos. La función "describe" automáticamente calcula estadísticas básicas para todas las variables.

In [9]:
```
Healthy[["a1","a2","a3","a4"]].describe() # Datos sin falla.
```

Out[9]:

	a1	a2	a3	a4
count	1.015808e+06	1.015808e+06	1.015808e+06	1.015808e+06
mean	3.696123e-03	-1.198666e-03	8.137643e-03	-1.118025e-04
std	7.381206e+00	4.427153e+00	4.110277e+00	4.523553e+00
min	-5.872160e+01	-3.294430e+01	-2.916740e+01	-3.133450e+01
25%	-3.921110e+00	-2.396700e+00	-2.204780e+00	-2.382110e+00
50%	-1.153235e-01	6.161250e-02	5.242570e-02	1.312545e-01
75%	3.786158e+00	2.486720e+00	2.290940e+00	2.521833e+00
max	5.670660e+01	3.092830e+01	2.513000e+01	3.734990e+01

In [10]:
```
Broken[["a1","a2","a3","a4"]].describe() # Datos con falla de diente roto de engranaje.
```

Out[10]:

	a1	a2	a3	a4
count	1.005311e+06	1.005311e+06	1.005311e+06	1.005311e+06
mean	-1.037558e-03	1.731702e-03	7.305638e-04	1.339568e-03
std	4.602696e+00	4.389516e+00	3.808909e+00	4.408367e+00
min	-2.534610e+01	-3.250230e+01	-2.585610e+01	-2.739530e+01
25%	-2.771530e+00	-2.469080e+00	-2.030490e+00	-2.365505e+00
50%	-5.425910e-02	1.257260e-01	4.504200e-02	1.012820e-01
75%	2.668080e+00	2.679470e+00	2.104100e+00	2.444225e+00
max	2.637010e+01	2.467780e+01	2.693890e+01	3.236610e+01

ANÁLISIS EXPLORATORIO Y LIMPIEZA DE DATOS

In [11]:
```
# Como se muestra los datos corresponden únicamente a valores numéricos de tipo flotante.
Healthy.info() # Muestra informacion de cada columna, tipos de datos, y conteo de los mismos.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1015808 entries, 0 to 106751
Data columns (total 5 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   a1      1015808 non-null  float64
 1   a2      1015808 non-null  float64
 2   a3      1015808 non-null  float64
 3   a4      1015808 non-null  float64
 4   carga   1015808 non-null  float64
dtypes: float64(5)
memory usage: 46.5 MB
```

In [12]:
```
Broken.info()  # Muestra informacion de cada columna, tipos de datos, y conteo de los mismos.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1005311 entries, 0 to 105727
Data columns (total 5 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   a1      1005311 non-null  float64
 1   a2      1005311 non-null  float64
 2   a3      1005311 non-null  float64
 3   a4      1005311 non-null  float64
 4   carga   1005311 non-null  float64
dtypes: float64(5)
```

Control de datos faltantes.

In [13]:
```
# Datos sin falla.
# control de datos faltantes o nulos; esto lo podemos realizar utilizando el método isnull() del siguiente modo:
# El método devolvera true en caso de la existencia de valores faltantes o false si no existe dichos valores.
Healthy.isnull().any().any()
```

Out[13]:
```
False
```

In [14]:
```
# Realiza la búsqueda de valores faltantes en cada una de las columnas del conjunto de datos y devuelve la suma de dichos valore
Healthy.isna().sum()
```

Out[14]:
```
a1       0
a2       0
a3       0
a4       0
carga    0
dtype: int64
```

In [15]:
```
# Datos con falla de diente roto de engranaje.
# control de datos faltantes o nulos; esto lo podemos realizar utilizando el método isnull() del siguiente modo:
# El método devolvera true en caso de la existencia de valores faltantes o false si no existe dichos valores.
Broken.isnull().any().any()
```

Out[15]:
```
False
```

In [16]:
```
Broken.isna().sum() # Cuenta valores faltantes para cada columna, luego suma para contar los valores de NaN.
```

Out[16]:
```
a1       0
a2       0
a3       0
a4       0
carga    0
dtype: int64
```

Control de datos duplicados.

Este proceso es importante debido a que de otra manera los datos duplicados podrían ser considerados más importantes que el resto por parte de un modelo de ML provocando que este se entrene de manera sesgada.

In [17]:
```
# Datos sin falla
Healthy.drop_duplicates()# Elimina los datos duplicados en caso de su existencia
```

Out[17]:

	a1	a2	a3	a4	carga
0	4.636710	0.516978	-3.205940	1.82241	0.0
1	1.992800	4.184660	-2.740610	2.80436	0.0
2	-3.764110	0.997335	-1.303090	1.83668	0.0
3	-4.558710	6.104330	-1.720690	1.72311	0.0
4	0.575382	0.170980	-0.497967	-1.32895	0.0
...	...	...	...	...	...
106747	0.677448	-3.234410	-1.725990	-3.14302	90.0
106748	-10.575400	7.725400	-2.184010	2.56965	90.0
106749	-4.033290	2.576920	1.468430	2.72891	90.0
106750	1.868670	-5.089400	5.342290	-1.36563	90.0
106751	7.581480	6.205960	-6.121330	11.54830	90.0

1015808 rows × 5 columns

In [18]:
```
# Datos sin falla.
Healthy.duplicated().sum() # devuelve la suma de los datos dupllicados enccontrados.
```

Out[18]:
```
0
```

In [19]:
```
# Datos con falla de diente roto de engranaje.
Broken.drop_duplicates()# Elimina datos duplicados en caso de su existencia.
```

Out[19]:

	a1	a2	a3	a4	carga
0	2.350390	1.454870	-1.667080	-2.055610	0.0
1	2.452970	1.400100	-2.825100	0.984487	0.0
2	-0.241284	-0.267390	0.793540	0.605862	0.0
3	1.130270	-0.890918	0.696969	0.613068	0.0
4	-1.296140	0.980479	-1.130560	-0.346971	0.0
...	...	...	...	...	...
105723	4.434170	-2.037930	-0.546417	1.181320	90.0
105724	2.903320	-2.820350	-2.486930	0.115640	90.0
105725	-1.617990	1.278610	-5.159510	-1.376230	90.0
105726	-2.011220	-4.029450	-0.082050	0.932847	90.0
105727	-4.437360	-1.970410	3.074570	-3.557610	90.0

1005311 rows × 5 columns

In [20]:
```
# Datos con falla de diente roto de engranaje.
Broken.duplicated().sum()# devuelve una suma de los datos dupllicados enccontrados.
```

Out[20]:
```
0
```

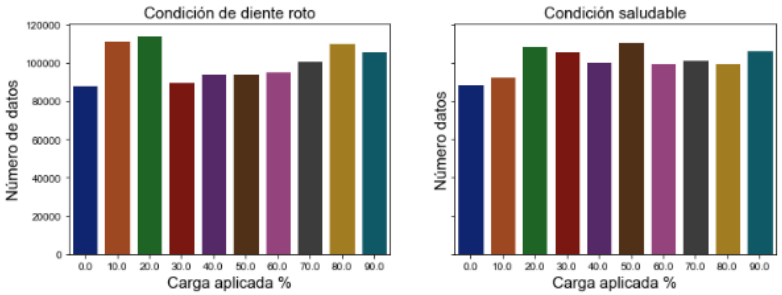
RESULTADOS:

- **No se encontro datos duplicados en ninguna columna.**
- **Los datos son de tipo numérico.**
- **No se encontro datos faltantes o nulos.**
- **Debido a los resultados obtenidos no fue necesario realizar ninguna transformación al conjunto de datos.**

Representación gráfica del conjunto de datos.

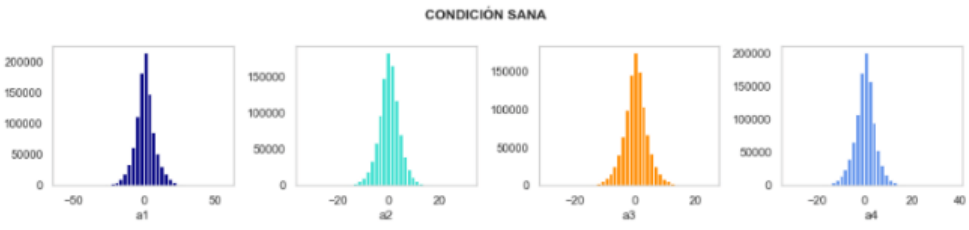
In [21]:
```
# Crea gráficos de barras para representar el conjunto de datos.
fig, axes = plt.subplots(1, 2, figsize=(12, 4), sharey=True)

sns.set(style='whitegrid', palette='dark', font_scale=1)# Estilo y Escala de las gráficas.

# Datos con falla de diente roto.
# Muestre los conteos de observaciones en cada contenedor usando barras.
broken = sns.countplot(ax=axes[0], x='carga',data=Broken)
axes[0].set_title('Condición de diente roto', fontsize=15) # Genera el título.
broken.set_xlabel('Carga aplicada %',fontsize=15)          # Genera el título del eje x
b=broken.set_ylabel('Número de datos',fontsize=15)         # Genera el título del eje y.

# Datos sin falla.
# Muestre los conteos de observaciones en cada contenedor usando barras.
healthy = sns.countplot(ax=axes[1], x='carga',data=Healthy)
axes[1].set_title('Condición saludable' , fontsize=15)      # Genera el título.
healthy.set_xlabel('Carga aplicada %',fontsize=15)          # Genera el título del eje x.
healthy.set_ylabel('Número datos',fontsize=15)              # Genera el título del eje y.
```

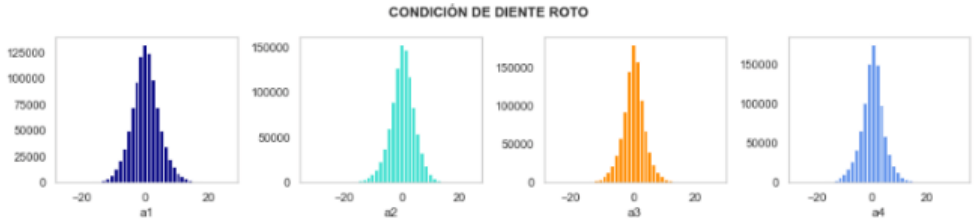
Out[21]:
```
Text(0, 0.5, 'Número datos')
```



Distribución de los datos.

In [22]:
```
# Crea graficas para ver la forma de distribución de las variables (gaussiana o no).
Columnas=['a1','a2','a3','a4'] # Genera una lista que contiene el nombre de las columnas.
colors = ['navy', 'turquoise', 'darkorange', 'cornflowerblue','gold','fuchsia','slateblue','teal'] # Colores de cada histograma.

fig,ax= plt.subplots(1,4,figsize=(13,3))
fig1,ax1= plt.subplots(1,4,figsize=(13,3))

# Genera los histogramas para el conjunto de datos en condición saludable.
for i in range(len(Columnas)):
    ax[i].hist(Healthy[Columnas[i]], color=colors[i],alpha=1,bins=40) # Genera los histogramas.

    ax[i].set_xlabel((Columnas[i])) # asigna al eje x los nombres almacenados en la lista columnas.
    ax[i].grid()

# Genera los histogramas para el conjunto de datos en condicion de diente roto.
for j in range(len(Columnas)):
    ax1[j].hist(Broken[Columnas[j]], color=colors[j],alpha=1,bins=40) # Genera los histogramas.

    ax1[j].set_xlabel((Columnas[j])) # asigna al eje x los nombres almacenados en la lista columnas.
    ax1[j].grid()

fig.suptitle('CONDICIÓN SANA', fontsize = 13, fontweight = "bold");
fig.tight_layout()
fig1.suptitle('CONDICIÓN DE DIENTE ROTO', fontsize = 13, fontweight = "bold");
fig1.tight_layout()
plt.subplots_adjust(top = 0.85)# Ajusta la posición de los histogramas
```





PRUEBAS DE NORMALIDAD

En ML los datos que siguen una distribución normal son beneficiosos para la creación de modelos debido a que facilitan los cálculos matemáticos que realizan los algoritmos. Se aplicó las pruebas estadísticas de normalidad de D'Agostino's y Kolmogorov-Smirnov, se descartó la prueba de shapiro wilk dado que el tamaño de los datos es mayor a 5000.

En la implementación SciPy de estas pruebas, puede interpretar el valor p de la siguiente manera.

- p <= alfa : rechazar H0, no normal.
- p > alfa : falla al rechazar H0, normal.

D'Agostino's K-squared test

In [23]:
```
# normality test
stat, p = normaltest(Healthy["a3"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Los datos tienen una distribución normal (No se rechaza H0)')
else:
    print('Los datos no tienen una distribución normal (Se rechaza H0)')
```

```
Statistics=42472.700, p=0.000
Los datos no tienen una distribución normal (Se rechaza H0)
```

La prueba se aplicó para las cuatro columnas y en todas se se obtuvo que los datos no siguen una distribución normal.

kstest de Kolmogorov-Smirnov

In [24]:
```
# realiza la prueba kstest de Kolmogorov-Smirnov en cada columna de los conjuntos de datos con falla y sin falla.
test_stat0 = kstest(Healthy["a1"], 'norm')
test_stat1 = kstest(Healthy["a2"], 'norm')
test_stat2 = kstest(Healthy["a3"], 'norm')
test_stat3 = kstest(Healthy["a4"], 'norm')

test_stat4 = kstest(Broken["a1"], 'norm')
test_stat5 = kstest(Broken["a2"], 'norm')
test_stat6 = kstest(Broken["a3"], 'norm')
test_stat7 = kstest(Broken["a4"], 'norm')

print("**Condición saludable**")
print(test_stat0)
print(test_stat1)
print(test_stat2)
print(test_stat3)

print("**Condición de diente roto**")
print(test_stat4)
print(test_stat5)
print(test_stat6)
print(test_stat7)
```

```
**Condición saludable**
KstestResult(statistic=0.33996050026893865, pvalue=0.0)
KstestResult(statistic=0.27674914651204563, pvalue=0.0)
KstestResult(statistic=0.2606535914583298, pvalue=0.0)
KstestResult(statistic=0.2805405064110501, pvalue=0.0)
**Condición de diente roto**
KstestResult(statistic=0.2935869804041567, pvalue=0.0)
KstestResult(statistic=0.2913414879265709, pvalue=0.0)
KstestResult(statistic=0.246249168507637, pvalue=0.0)
KstestResult(statistic=0.2751051305716644, pvalue=0.0)
```

Normalización de datos método yeo-johnson.

Las transformaciones de potencia son una familia de transformaciones paramétricas y monótonas que se aplican para hacer los datos más parecidos a los de Gauss.Esto es útil para modelar cuestiones relacionadas con la heteroscedasticidad (variación no constante),u otras situaciones en las que se desea la normalidad.

- Actualmente,power_transform admite la transformación Box-Cox y la transformación Yeo-Johnson.El parámetro óptimo para estabilizar la varianza y minimizar la asimetría se estima mediante máxima verosimilitud.
- Box-Cox exige que los datos de entrada sean estrictamente positivos, mientras que Yeo-Johnson apoya tanto los datos positivos como los negativos.

Normalización de datos con carga de 50 para el ME1.

In [25]:
```
dt_h50 # datos sin falla y carga de 50 (ME1)
pt = PowerTransformer()
data0 = dt_h50.drop(["carga"], axis=1) # Elimina la columna carga para hacer la prueba de normalidad.
print(pt.fit(data0))
PowerTransformer(copy=True, method='yeo-johnson', standardize=False) # Ejecuta la transformación.
print(pt.lambdas_)
Healthy_nor= pd.DataFrame(pt.transform(data0)) # Guarda los resultados en un dataframe.
Healthy_nor.columns = ['a1', 'a2', 'a3', 'a4'] # Genera el nombre de las columnas.
Healthy_nor.to_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Healthy_nor_yeo-johnson.csv",index=False)
Healthy_nor
```

Out[25]:

	a1	a2	a3	a4
0	0.296770	-0.455054	-0.068339	-0.965248
1	-1.337245	-1.843241	0.404350	-1.534807
2	-0.164618	0.148745	-0.842537	0.036986
3	1.027537	-0.354249	-0.433090	2.440945
4	-0.081703	-0.058648	2.331625	1.974877
...	...	...	...	...
110843	0.440839	-0.615159	0.626561	-0.166807
110844	0.192154	-0.530503	0.594809	-0.277363
110845	0.014220	-0.734462	-0.354628	-0.160755
110846	0.009611	-0.760561	-0.370617	-0.476042
110847	0.150432	-0.106821	-0.541181	-0.132389

110848 rows × 4 columns

In [26]:
```
dt_b50 # datos con falla de diente roto y carga de 50 (ME1)
pt = PowerTransformer()
data1 = dt_b50.drop(["carga"], axis=1)  # Elimina la columna carga para hacer la prueba de normalidad.
print(pt.fit(data1))
PowerTransformer(copy=True, method='yeo-johnson', standardize=False) # Ejecuta la transformación.
print(pt.lambdas_)
Broken_nor = pd.DataFrame(pt.transform(data1)) # Guarda los resultados en un dataframe.
Broken_nor.columns = ['a1', 'a2', 'a3', 'a4']  # Genera el nombre de las columnas.
Broken_nor.to_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Broken_nor_yeo-johnson.csv",index=False)
Broken_nor
```

```
PowerTransformer()
[0.96754169 1.06162631 1.01215577 1.03097083]
```

Out[26]:

	a1	a2	a3	a4
0	-0.823161	1.587283	-0.321027	4.708347
1	0.526089	2.498069	-0.826624	1.867836
2	1.292050	-0.748901	-0.181230	-0.913135
3	-0.836335	-2.958392	0.989384	-1.215021
4	0.396774	-1.430372	1.316012	-1.896818
...	...	...	...	...
94203	-0.280470	0.620964	-0.299159	-0.984241
94204	-0.810938	0.146568	1.663584	-1.177288
94205	0.390184	0.945833	1.386175	0.018496
94206	1.828997	2.517015	0.393673	0.791543
94207	1.275377	1.335469	0.616482	0.424319

94208 rows × 4 columns

Se realiza nuevamente el test de normalidad:

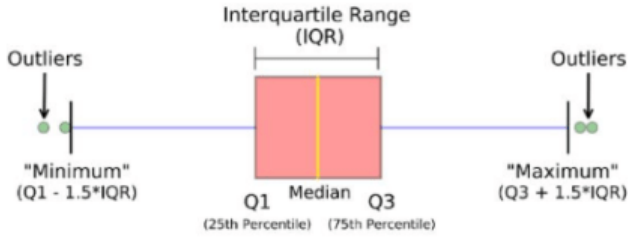
In [27]:
```
# normality test
stat, p = normaltest(Healthy_nor["a3"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Los datos tienen una distribución normal (No se rechaza H0)')
else:
    print('Los datos no tienen una distribución normal (Se rechaza H0)')
```

```
Statistics=3630.614, p=0.000
Los datos no tienen una distribución normal (Se rechaza H0)
```

Resultados.

Se aplicó nuevamente el test de normalidad y el resultado indica nuevamente que los datos no siguen una distribución normal, sin embargo, en la fase de entrenamiento los modelos obtuvieron un rendimiento similar independientemente de la aplicación de este método de normalización, por lo que se consideró trabajar con el conjunto de datos original ya que fue con el cual se obtuvo un mejor rendimiento.

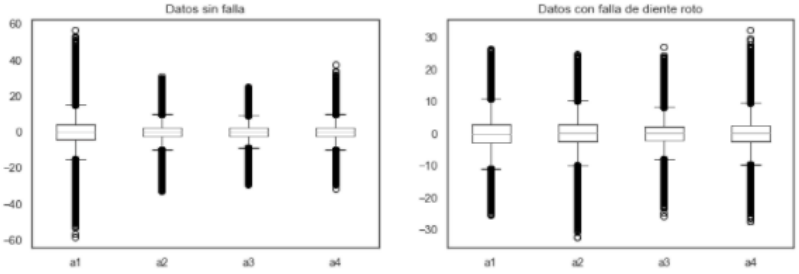
BÚSQUEDA DE VALORES ATÍPICOS.



Las siguientes funciones observan una columna de valores dentro de un marco de datos y calculan el primer y tercer cuartiles, el rango intercuartil y el mínimo y el máximo. Cualquier valor fuera del mínimo y máximo es un valor atípico.

In [28]:
```
def get_iqr_values(df_in, col_name):
    median = df_in[col_name].median()
    q1 = df_in[col_name].quantile(0.25) #  Priemer cuartil.
    q3 = df_in[col_name].quantile(0.75) #  Tercer cuartil.
    iqr = q3-q1 # Calcula el rango intercuartílico.
    minimum  = q1-1.5*iqr  # Mínimo.
    maximum = q3+1.5*iqr   # Máximo.
    return median, q1, q3, iqr, minimum, maximum # devuelve el cuartil Q1, Q3 , Rango intercuartilíco y la mediana.

def remove_outliers(df_in, col_name): # Define una funsión para remover los datos atípicos.
    _, _, _, _, minimum, maximum = get_iqr_values(df_in, col_name)
    df_out = df_in.loc[(df_in[col_name] > minimum) & (df_in[col_name] < maximum)]
    return df_out
def count_outliers(df_in, col_name): # Define una funsión para contar los valores atípicos.
    _, _, _, _, minimum, maximum = get_iqr_values(df_in, col_name)
    df_outliers = df_in.loc[(df_in[col_name] <= minimum) | (df_in[col_name] >= maximum)]
    return df_outliers.shape[0]
```

In [29]:
```
fig=plt.figure(figsize=(20,4))
df=Healthy #conjunto de datos sin falla.
df1=Broken # conjunto de datos con falla de diente roto.
subplot(131)
df.boxplot( column =['a1',"a2","a3","a4"], grid = False) # Genera los diagramas de cajas de datos sin falla.
title('Datos sin falla')
subplot(132)
df1.boxplot( column =['a1',"a2","a3","a4"], grid = False) # Genera los diagramas de cajas de datos sin falla.
title('Datos con falla de diente roto')
```



In [30]:
```
# Cuenta los valores atípicos en cada columna especificada y los imprime.
print(f"Outliers encontrados en el conjunto de datos sin falla:")
print(f"La columna_a1 Tiene {count_outliers(df, 'a1')} outliers")
print(f"La columna_a2 Tiene {count_outliers(df, 'a2')} outliers")
print(f"La columna_a3 Tiene {count_outliers(df, 'a3')} outliers")
print(f"La columna_a4 Tiene {count_outliers(df, 'a4')} outliers")

# Cuenta los valores atípicos en cada columna especificada y los imprime.
print(f"Outliers encontrados en el conjunto de datos con falla de diente roto:")
print(f"La columna_a1 Tiene {count_outliers(df1, 'a1')} outliers")
print(f"La columna_a2 Tiene {count_outliers(df1, 'a2')} outliers")
print(f"La columna_a3 Tiene {count_outliers(df1, 'a3')} outliers")
print(f"La columna_a4 Tiene {count_outliers(df1, 'a4')} outliers")
```

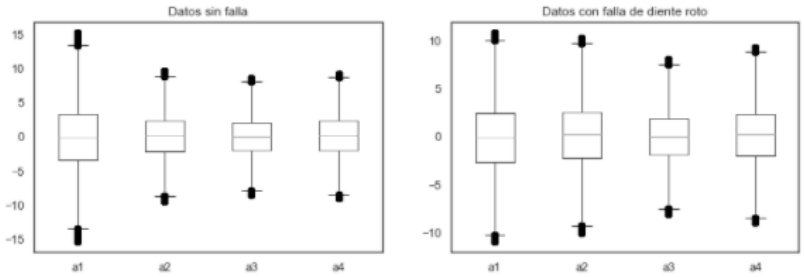
```
Outliers encontrados en el conjunto de datos sin falla:
La columna_a1 Tiene 52539 outliers
La columna_a2 Tiene 40801 outliers
La columna_a3 Tiene 43602 outliers
La columna_a4 Tiene 45798 outliers
Outliers encontrados en el conjunto de datos con falla de diente roto:
La columna_a1 Tiene 28601 outliers
La columna_a2 Tiene 29924 outliers
La columna_a3 Tiene 45110 outliers
La columna_a4 Tiene 44265 outliers
```

Eliminación de valores atípicos.

In [31]:
```
# Elimina los valores atípicos encontrados en el conjunto de datos sin falla.
df = remove_outliers(df, 'a1') # Remueve los valores atípicos de la columna a1.
df = remove_outliers(df, 'a2') # Remueve los valores atípicos de la columna a2.
df = remove_outliers(df, 'a3') # Remueve los valores atípicos de la columna a3.
df = remove_outliers(df, 'a4') # Remueve los valores atípicos de la columna a4.
# Elimina los valores atípicos encontrados en el conjunto de datos con falla de diente roto.
df1 = remove_outliers(df1, 'a1')  # Remueve los valores atípicos de la columna a1.
df1 = remove_outliers(df1, 'a2')  # Remueve los valores atípicos de la columna a2.
df1 = remove_outliers(df1, 'a3')  # Remueve los valores atípicos de la columna a3.
df1 = remove_outliers(df1, 'a4')  # Remueve los valores atípicos de la columna a4.
```

In [32]:
```
# Genera de manera rápida los diagramas de cajas de cada una de las columnas.
fig=plt.figure(figsize=(20,4))
subplot(131)
df.boxplot( column =['a1',"a2","a3","a4"], grid = False) # Genera los diagramas de cajas del conjunto de datos sin falla.
title('Datos sin falla')
subplot(132)
df1.boxplot( column =['a1',"a2","a3","a4"], grid = False) # Genera los diagramas de cajas del conjunto de datos con falla.
title('Datos con falla de diente roto')
```

Out[32]:
```
Text(0.5, 1.0, 'Datos con falla de diente roto')
```



In [33]:
```
df.to_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Healthy_removed_va",index=False) # Guarda.
df # Datos sin falla con valores atípicos removidos.
```

In [34]:
```
df1.to_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Broken_removed_va",index=False) # Guarda.
df1 # Datos con falla de diente roto con valores atípicos removidos.
```

RESULTADOS:

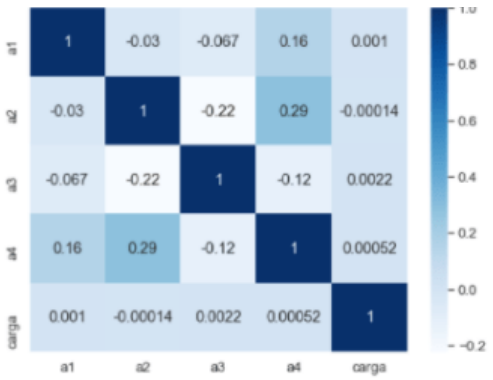
Durante la fase de entrenamiento los modelos obtuvieron mejores rendimientos al ser entrenados con todo el conjunto de datos incluidos aquellos valores identificados como atípicos, por lo tanto, se consideró trabajar con el conjunto de datos original. El mismo análisis se realizó para cada uno de los tres métodos experimentales y en cada uno se obtuvo el mismo resultado. Además, se debe considerar que el algoritmo SVM maneja muy bien los valores atípicos ya que posee parámetros de ajuste que le permite evitar el sobreajuste causado por la presencia de dichos valores. Por otra parte, dentro de la creación de modelos de ML también se dispone de métodos de escalado como RobustScaler que se utiliza cuando un conjunto de datos presenta valores atípicos.

MATRIZ DE CORRELACIÓN.

La matriz de correlación es una importante métrica de análisis de datos que se calcula para resumir los datos a fin de comprender la relación entre las diversas variables y tomar decisiones en consecuencia.

Datos sin falla

In [35]:
```
rcParams['figure.figsize'] = 8,6
columns = ['a1', 'a2', 'a3', 'a4', 'carga'] # almaceno en la variable columnna cada una de las columnas del conjunto de datos.
sns.set(font_scale=1.2) # Estilo y escala (optativo)
sns.heatmap(Healthy[columns].corr(),annot=True,cmap='Blues') # Genera la matriz de confusión.
plt.show()
```



Datos con falla de diente roto

In [36]:
```
rcParams['figure.figsize'] = 8, 6
sns.set(font_scale=1.2)
columns = ['a1', 'a2', 'a3', 'a4', 'carga'] # almacena en la variable columnna cada una de las columnas del dataset 1
sns.heatmap(Broken[columns].corr(),annot=True,cmap='Blues') # Genera la matriz de confusión.
plt.show()
```

Representación gráfica de las señales de vibración.

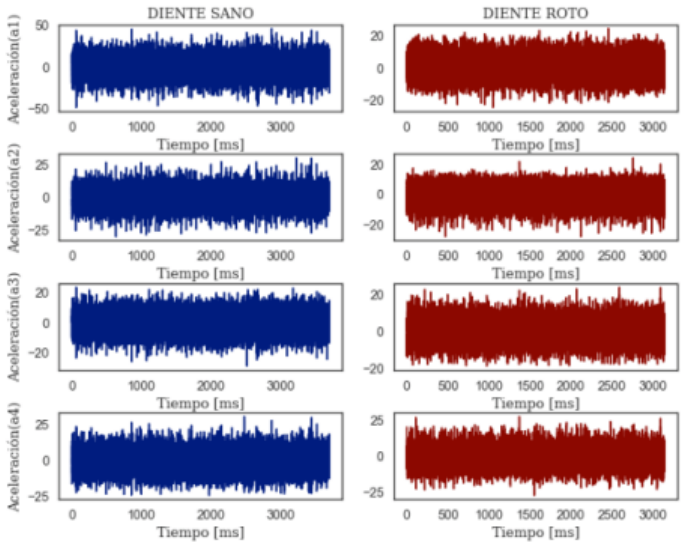
GRÁFICAS DE ACELERACIÓN VS TIEMPO.

In [37]:
```
sns.set(style='white', palette='dark',font_scale=1.1)

acel_1_H=dt_h50["a1"]
acel_2_H=dt_h50["a2"]
acel_3_H=dt_h50["a3"]
acel_4_H=dt_h50["a4"] # datos del aceleroimetro a1 columna 1

acel_1_B=dt_b50["a1"]
acel_2_B=dt_b50["a2"]
acel_3_B=dt_b50["a3"]
acel_4_B=dt_b50["a4"]# datos del aceleroimetro a1 columna 1

Nd=acel_1_H.size   # tamaño de datos
#Nd=16384

Nd1=acel_1_B.size
#Nd1=16384

x=acel_1_H[0:Nd]
x1=acel_2_H[0:Nd]
x2=acel_3_H[0:Nd]
x3=acel_4_H[0:Nd]

x4=acel_1_B[0:Nd1]
x5=acel_2_B[0:Nd1]
x6=acel_3_B[0:Nd1]
x7=acel_4_B[0:Nd1]
```

```
fs=30  #frecuencia de muestreo?
dT = 1/fs
# Arreglo columna tiempo
time = np.linspace(0.0, (Nd*dT), Nd)
time1 = np.linspace(0.0, (Nd1*dT), Nd1)

fig=plt.figure(figsize=(9,7))
fig.tight_layout()
ax1=fig.add_subplot(4,2,1)
ax2=fig.add_subplot(4,2,2)
ax3=fig.add_subplot(4,2,3)
ax4=fig.add_subplot(4,2,4)
ax5=fig.add_subplot(4,2,5)
ax6=fig.add_subplot(4,2,6)
ax7=fig.add_subplot(4,2,7)
ax8=fig.add_subplot(4,2,8)
fig.tight_layout()
ax1.plot(time,x) # Crea la gráfica
ax1.set_ylabel( "Aceleración(a1)",fontfamily="serif") # Nombra los ejes
ax1.set_xlabel("Tiempo [ms]",fontfamily="serif") # Nombra los ejes
ax1.set_title("DIENTE SANO",fontfamily="serif") # Crea un título

ax2.plot(time1,x4,'r')
ax2.set_title("DIENTE ROTO",fontfamily="serif") # Crea un título
ax2.set_xlabel("Tiempo [ms]",fontfamily="serif")

ax3.plot(time,x1)
ax3.set_ylabel("Aceleración(a2)",fontfamily="serif") # Nombra los ejes
ax3.set_xlabel("Tiempo [ms]",fontfamily="serif") # Nombra los ejes
ax4.plot(time1,x5,'r')
ax4.set_xlabel("Tiempo [ms]",fontfamily="serif")
```

```
ax5.plot(time,x2)
ax5.set_ylabel("Aceleración(a3)",fontfamily="serif") # Nombra los ejes
ax5.set_xlabel("Tiempo [ms]",fontfamily="serif") # Nombra los ejes
ax6.plot(time1,x6,'r')
ax6.set_xlabel("Tiempo [ms]",fontfamily="serif")

ax7.plot(time,x3)
ax7.set_ylabel("Aceleración(a4)",fontfamily="serif") # Nombra los ejes
ax7.set_xlabel("Tiempo [ms]",fontfamily="serif") # Nombra los ejes
ax8.plot(time1,x7,'r')
ax8.set_xlabel("Tiempo [ms]",fontfamily="serif")
```

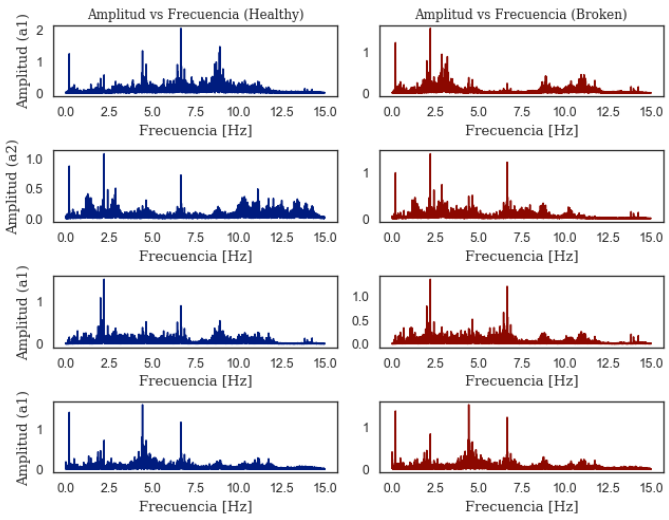
Out[37]: `Text(0.5, 37.49999999999994, 'Tiempo [ms]')`



GRÁFICAS DE AMPLITUD VS FRECUENCIA.

In [38]:
```
from scipy import fftpack
#Esta función calcula la Transformada de Fourier discreta (DFT) unidimensional de n puntos con el eficiente algoritmo de Transfo
xf = np.fft.fft(x)
xf1 = np.fft.fft(x4)
xf2 = np.fft.fft(x1)
xf3 = np.fft.fft(x5)
xf4 = np.fft.fft(x2)
xf5 = np.fft.fft(x6)
xf6 = np.fft.fft(x3)
xf7 = np.fft.fft(x7)
#xwf = np.fft.fft(xw)
freq = np.linspace(0.0, 1.0/(2.0*dT), Nd//2)
freq1 = np.linspace(0.0, 1.0/(2.0*dT), Nd1//2)

fig=plt.figure(figsize=(9,7))
fig.tight_layout()
ax1=fig.add_subplot(4,2,1)
ax2=fig.add_subplot(4,2,2)
ax3=fig.add_subplot(4,2,3)
ax4=fig.add_subplot(4,2,4)
ax5=fig.add_subplot(4,2,5)
ax6=fig.add_subplot(4,2,6)
ax7=fig.add_subplot(4,2,7)
ax8=fig.add_subplot(4,2,8)
fig.tight_layout()
sns.set(style='white',palette='dark',font='sans-serif')

# Arreglo columna frecuencia
freq = np.linspace(0.0, 1.0/(2.0*dT), Nd//2)

ax1.plot(freq[1:Nd//2], 2.0/Nd * np.abs(xf[1:Nd//2])) # Crea la gráfica.
ax1.set_ylabel("Amplitud (a1)",fontfamily="serif") # Nombra los ejes.
ax1.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.
ax1.set_title("Amplitud vs Frecuencia (Healthy)",fontfamily="serif") # Crea un título.
ax2.plot(freq1[1:Nd1//2], 2.0/Nd1 * np.abs(xf1[1:Nd1//2]), 'r')
ax2.set_title("Amplitud vs Frecuencia (Broken)",fontfamily="serif") # Crea un título.
ax2.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.
```

```
ax3.plot(freq[1:Nd//2], 2.0/Nd * np.abs(xf2[1:Nd//2])) # Crea la gráfica.
ax3.set_ylabel("Amplitud (a2)",fontfamily="serif") # Nombra los ejes.
ax3.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes .
ax4.plot(freq1[1:Nd1//2], 2.0/Nd1 * np.abs(xf3[1:Nd1//2]), 'r')
ax4.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.

ax5.plot(freq[1:Nd//2], 2.0/Nd * np.abs(xf4[1:Nd//2])) # Crea la gráfica.
ax5.set_ylabel("Amplitud (a1)",fontfamily="serif") # Nombra los ejes.
ax5.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.
ax6.plot(freq1[1:Nd1//2], 2.0/Nd1 * np.abs(xf5[1:Nd1//2]), 'r')
ax6.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.

ax7.plot(freq[1:Nd//2], 2.0/Nd * np.abs(xf6[1:Nd//2])) # Crea la gráfica.
ax7.set_ylabel("Amplitud (a1)",fontfamily="serif") # Nombra los ejes.
ax7.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes .
ax8.plot(freq1[1:Nd1//2], 2.0/Nd1 * np.abs(xf7[1:Nd1//2]), 'r')
ax8.set_xlabel("Frecuencia [Hz]",fontfamily="serif") # Nombra los ejes.
fig.tight_layout()
```


Amplitud vs Frecuencia (Healthy)
Amplitud vs Frecuencia (Broken)
Amplitud (a1)
Amplitud (a2)
Amplitud (a1)
Amplitud (a1)
Frecuencia [Hz]
0.0 2.5 5.0 7.5 10.0 12.5 15.0

ANEXO B: CÓDIGO DE PROGRAMACIÓN GENERAL DEL MODELO PREDICTIVO.

LIBRERÍAS.

In [1]:
```
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams

from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import tsfel # Librería de extracción de características
from sklearn.model_selection import GridSearchCV, StratifiedKFold, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFECV
from sklearn.feature_selection import SelectFromModel

# Métricas de evaluación de la matriz de confusión
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score,recall_score,accuracy_score,f1_score,roc_auc_score,roc_curve,plot_confusion_matrix

# Validación cruzada
from sklearn.model_selection import cross_validate
from sklearn.model_selection import RepeatedKFold

# Curva de aprendizaje
from sklearn.model_selection import learning_curve

# Librerías que contienen los algoritmos.
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
import warnings
warnings.filterwarnings('ignore')
```

LECTURA DE DATOS.

- **Healthy:** Representa el conjunto de datos de señales de vibración de caja de engranajes en condicion saludable.
- **Broken:** Representa el conjunto de datos de señales de vibración de caja de engranajes con falla de diente roto.

In [2]:
```
# Realiza la lectura de datos ubicados en el directorio especificado.
# header=0 significa que tiene los nombres de las columnas en la primera fila del archivo
Healthy = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Healthy_carga_50.csv",header=0)
Broken = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/PROGRAMACIÓN/PREPROCESAMIENTO/Broken_carga_50.csv",header=0)
```

Datos sin falla

In [3]:
```
Healthy # Muestra el conjunto de datos de señales de vibración sin falla y carga de 50
```

Out[3]:

	a1	a2	a3	a4	carga
0	2.144160	-1.958210	-0.190533	-4.584750	50.0
1	-9.920150	-7.475190	1.794680	-7.472510	50.0
2	-1.330590	0.751472	-3.557400	0.328149	50.0
3	7.761710	-1.498460	-1.764830	10.991900	50.0
4	-0.714011	-0.164771	9.650560	8.970950	50.0
...	...	...	...	...	...
110843	3.245040	-2.692110	2.714350	-0.639589	50.0
110844	1.347960	-2.303540	2.583290	-1.174450	50.0
110845	0.003272	-3.241480	-1.423700	-0.610475	50.0
110846	-0.031329	-3.361870	-1.493090	-2.147170	50.0
110847	1.031390	-0.380219	-2.235860	-0.474274	50.0

110848 rows × 5 columns

Datos con falla

In [4]:
```
Broken # Muestra el conjunto de datos de señales de vibración con falla de diente roto y carga de 50
```

Out[4]:

	a1	a2	a3	a4	carga
0	-3.93468	6.55216	-1.237980	20.310300	50.0
1	2.40285	9.99438	-3.242650	8.313200	50.0
2	6.24273	-3.17577	-0.688974	-4.193820	50.0
3	-3.99411	-14.14480	3.845360	-5.658600	50.0
4	1.76551	-6.46492	5.096230	-9.001360	50.0
...	...	...	...	...	...
94203	-1.46153	2.78470	-1.151670	-4.537800	50.0
94204	-3.87952	0.85365	6.423880	-5.474920	50.0
94205	1.73315	4.06958	5.364500	0.216419	50.0
94206	8.97857	10.06520	1.552720	3.656620	50.0
94207	6.15826	5.58468	2.412330	2.038780	50.0

94208 rows × 5 columns

EXTRACCIÓN DE CARACTERÍSTICAS

Se extraen características en el dominio del **tiempo** y en el dominio de la **frecuencia** de la siguinete manera:

- A cada conjunto de datos se le agrega una columna denominada falla indicando la clase o categoria.
- **0** para la clase negativa o sin falla
- **1** para la clase positiva o con falla de diente roto
- Por cada columna de la variable acelerómetro se extraen 26 características.
- Se aplica un tamaño de ventana de 20 y una frecuancia de muestreo de 30Hz.
- Se concatenan las características extraidas por cada clase y se guarda en un Dataframe denominado Características para su posterior análisis.

Extracción de características de datos sin falla.

In [5]:
```
directorio = "C:/Users/josel/Desktop/Código_TIC/FEATURES.json" # Especifica el directorio donde se ubica las características.
cfg_file = tsfel.load_json(directorio) # DICCIONARIO DE CONFIGURACIÓN JSON.
Características_H50 = tsfel.time_series_features_extractor(cfg_file, Healthy[["a1","a2","a3","a4"]], fs=30, window_size=20)
failureArray = np.zeros((len(Características_H50.index),1)) # Agrega una columna para identificar la clase.
Características_H50['falla'] = failureArray
Características_H50.to_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características_H50.csv",index=False) # Guarda en el directori
Características_H50 # muestra las características extraidas.
```

```
*** Feature extraction started ***
```

Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[5]:

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute deviation	...	3_Root mean square	3_Skewness	3_Spectral entropy
0	3.089500	0.384194	1.0	10.347797	-0.839346	16.1343	0.188574	13.333333	0.678641	6.972570	...	7.524775	0.410972	0.283213
1	3.822627	0.254412	1.0	17.501360	-1.091989	11.8568	0.110701	13.333333	0.075423	8.799153	...	8.057351	-0.294834	0.426058
2	2.977592	0.355117	1.0	9.742185	-0.816881	14.2881	0.221902	13.333333	1.457016	6.170889	...	6.521623	0.189503	0.359309
3	2.427698	0.249948	1.0	7.649475	-0.626452	11.2614	0.270927	15.000000	0.697798	4.602284	...	4.611258	-0.706948	0.724686
4	2.038543	0.338758	1.0	5.822616	-0.232435	8.8228	0.236482	15.000000	-1.672413	4.038083	...	3.626065	0.907036	0.835044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5537	3.367895	0.429428	1.0	14.852425	-0.632422	17.1948	0.082612	13.333333	-1.310147	8.364401	...	6.049958	-0.020065	0.753211
5538	5.160211	0.330848	1.0	26.271900	-0.804863	20.1282	0.298829	13.333333	-0.961794	12.305121	...	7.796252	0.099694	0.589246
5539	4.680645	0.263314	1.0	25.411025	-1.386487	21.2250	0.383420	13.333333	-0.219746	11.778391	...	4.492351	0.000979	0.801533
5540	1.934996	0.197938	1.0	7.499162	0.482230	17.8986	0.031253	15.000000	-0.193213	5.169473	...	4.075712	1.007132	0.778904
5541	2.830783	0.150840	1.0	5.496685	0.976384	21.1281	0.210137	13.333333	-0.119812	5.992101	...	3.302612	0.281622	0.868822

5542 rows × 105 columns

Extracción de características de datos con falla.

In [6]:
```
Características_B50= tsfel.time_series_features_extractor(cfg_file, Broken[["a1","a2","a3","a4"]], fs=30, window_size=20)
failureArray = np.ones((len(Características_B50.index),1)) # agrega una columna de unos.
Características_B50['falla'] = failureArray # agrega el nombre a la columna.
Características_B50.to_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características_B50.csv",index=False) # Guarda en el directori
Características_B50  # muestra las características extraidas.
```

```
*** Feature extraction started ***
```

Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[6]:

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute deviation	...	3_Root mean square	3_Skewness	3_Spectral entropy
0	2.214591	0.329130	1.0	8.709588	-0.936527	10.17610	0.251273	13.333333	0.170013	4.685339	...	6.897306	1.016097	0.628735
1	1.584097	0.277667	1.0	7.042877	-0.885800	7.03109	0.208940	13.333333	-1.129920	3.691302	...	4.453180	-0.711508	0.806893
2	2.099011	0.212135	1.0	5.928009	-0.821597	6.85497	0.346868	13.333333	-1.592181	3.469967	...	3.017482	0.421303	0.668164
3	0.808880	0.252472	1.0	2.752978	-1.060264	2.92295	0.327957	13.333333	-0.255019	1.418561	...	2.047618	-0.894622	0.766680
4	1.247340	0.267123	1.0	3.376162	-0.302159	6.86921	0.143070	15.000000	1.258848	2.206213	...	4.831092	-0.055754	0.673553
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4705	0.872752	0.264036	1.0	2.763622	-0.734197	3.37051	0.470541	15.000000	-0.152010	1.529198	...	2.267141	0.087656	0.815003
4706	1.992187	0.408660	1.0	5.362023	1.281796	6.87850	0.409349	13.333333	-0.302956	3.566407	...	5.510908	0.314092	0.645828
4707	3.006687	0.295486	1.0	9.659995	-1.234963	9.82473	0.311170	13.333333	0.532390	4.967042	...	3.355112	0.634064	0.479086
4708	1.945077	0.275813	1.0	5.182773	-0.500194	10.63980	0.161888	15.000000	1.793892	3.553627	...	5.069023	0.873172	0.719619
4709	2.449425	0.222164	1.0	8.118180	-1.208222	8.04652	0.270213	13.333333	0.518729	4.415785	...	3.939686	-0.492471	0.781977

4710 rows × 105 columns

Concatenación de características extraidas.

In [7]:
```
# Concatena todos los ficheros en uno solo marco de datos.
# axis = 0: indica que el método se ejecuta hacia abajo a lo largo de cada columna o etiqueta de fila.
# index=False; evita que se cree otro indice.
Características=pd.concat([Características_H50,Características_B50],axis=0,ignore_index=True)
Características.to_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características.csv",index=False) # Guarda en el directorio.
```

Lectura de características guardadas

In [8]:
```
# Realiza la lectura de características guardadas en el directorio.
# header=0: significa que tiene los nombres de las columnas en la primera fila del archivo
Características = pd.read_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características.csv",header=0) # Guarda en el directorio.
Características # muestra las características extraidas y guardadas.
```

Out[8]:

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute deviation	...	3_Root mean square	3_Skewness	3_Spectra entropy
0	3.089500	0.384194	1.0	10.347798	-0.839346	16.13430	0.188574	13.333333	0.678641	6.972570	...	7.524775	0.410972	0.28321
1	3.822627	0.254412	1.0	17.501360	-1.091989	11.85680	0.110701	13.333333	0.075423	8.799153	...	8.057351	-0.294834	0.42605
2	2.977592	0.355117	1.0	9.742185	-0.816881	14.28810	0.221902	13.333333	1.457016	6.170889	...	6.521623	0.189503	0.35930
3	2.427698	0.249948	1.0	7.649475	-0.626452	11.26140	0.270927	15.000000	0.697798	4.602284	...	4.611258	-0.706948	0.72468
4	2.038543	0.338758	1.0	5.822616	-0.232435	8.82280	0.236482	15.000000	-1.672413	4.038083	...	3.626065	0.907036	0.83504
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10247	0.872752	0.264036	1.0	2.763622	-0.734197	3.37051	0.470541	15.000000	-0.152010	1.529198	...	2.267141	0.087656	0.81500
10248	1.992187	0.408660	1.0	5.362023	1.281796	6.87850	0.409349	13.333333	-0.302956	3.566407	...	5.510908	0.314092	0.64582
10249	3.006687	0.295486	1.0	9.659995	-1.234963	9.82473	0.311170	13.333333	0.532390	4.967042	...	3.355112	0.634064	0.47908
10250	1.945077	0.275813	1.0	5.182773	-0.500194	10.63980	0.161888	15.000000	1.793892	3.553627	...	5.069023	0.873172	0.71961
10251	2.449425	0.222164	1.0	8.118180	-1.208222	8.04652	0.270213	13.333333	0.518729	4.415785	...	3.939686	-0.492471	0.78197

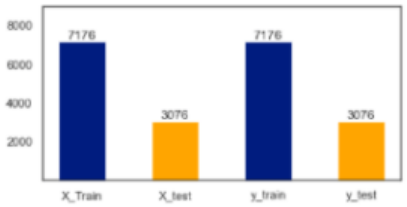
10252 rows × 105 columns

División de características para entrenamiento y prueba 70-30.

In [9]:
```
X = Características.drop('falla', axis=1) # separa las características de la variable objetivo.
y = Características.falla # variables objetivo.
# train_test_split: Divide los datos en 70% para entrenamiento y 30% para prueba.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=42)
print("Datos para entrenar: (X_train): ", len(X_train))
print("Datos para probar: (X_test): ", len(X_test))
```

```
Datos para entrenar: (X_train):  7176
Datos para probar: (X_test):  3076
```

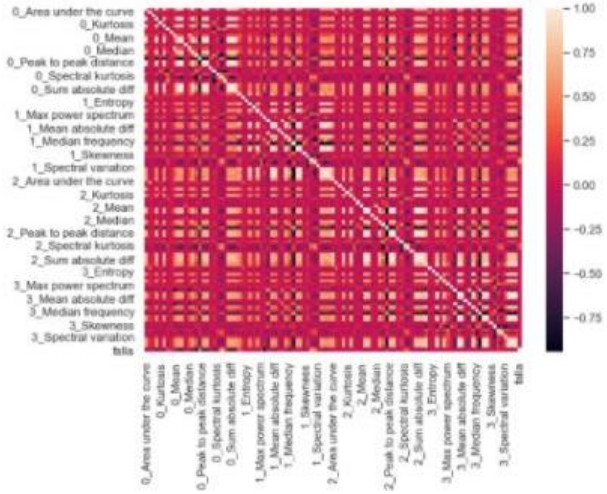
In [10]:
```
fig=plt.figure(figsize=(6,3))
sns.set(style='white',palette='dark',font_scale=1)# Estilo y escala
# Construir datos
Datos = [7176, 3076,7176, 3076]
plt.bar(range(4), Datos, width=0.50,align = 'center', color = ['b',"orange"], alpha = 1)
# Agregar etiqueta de eje.
plt.xticks(range(4), ['X_Train', 'X_test',"y_train","y_test"])
# Establecer el rango de escala del eje Y.
plt.ylim([1, 9000])
# Agregue etiquetas numéricas a cada gráfico de barras.
for x,y in enumerate(Datos):
    plt.text(x, y+100, '%s' %round(y,1), ha='center')
plt.show()
```



Mapa de calor de caracteristicas extraidas.

Se crea un mapa de calor para verificar la correlación entre las variables con el propósito de eliminar aquellas altamente correlacionadas ya que brindan la misma información y reducen la velocidad de entrenamiento del modelo.

In [11]:
```
# Mapa de calor
f,ax = plt.subplots(figsize=(8, 6))
sns.heatmap(Características.corr())
plt.show()
```



Eliminación de características correlacionadas.

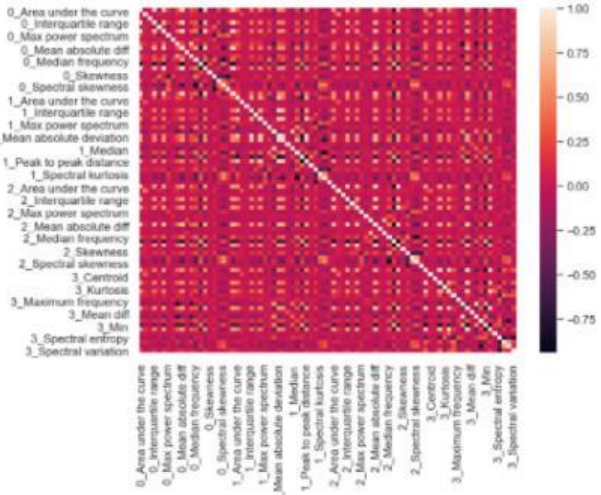
In [12]:
```
Características_correlacionadas = tsfel.correlated_features(X_train)
C=X_train
X_train.drop(Características_correlacionadas, axis=1, inplace=True) # elimina las características correlacionadas
X_test.drop(Características_correlacionadas,  axis=1, inplace=True)
print(f"'X_train' longitud: {X_train.shape}")  # Imprime el tamaño del conjunto de entrenamiento.
print(f"'X_test' longitud:  {X_test.shape}" )  # Imprime el tamaño del conjunto de prueba.
```

```
'X_train' longitud: (7176, 82)
'X_test' longitud:  (3076, 82)
```

Mapa de calor del nuevo conjunto de características.

In [13]:
```
f,ax = plt.subplots(figsize=(8, 6)) # Define el tamaño de la figura
sns.heatmap(X_train.corr())         # Mapa de calor del conjunto de datos entrenamiento.
```

Out[13]: <AxesSubplot:>



Estandarización de características.

In [14]:
```
columnas = C.columns # nombre de columnas
Estandarización = preprocessing.StandardScaler()
X_train = Estandarización.fit_transform(X_train)    # Estandariza el conjunto de entrenamiento
X_test  = Estandarización.transform(X_test)         # Estandariza el conjunto de prueba
X_train = pd.DataFrame(X_train,columns=[columnas])  # Crea un dataframe con las características estandarizadas
X_train # muestra las características estandarizadas
```

Out[14]:

	0_Area under the curve	0_Centroid	0_Entropy	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean	0_Mean absolute diff	...	3_Mean diff	3_Median	3_Median frequency
0	-0.837114	-0.363985	0.026406	-0.726475	-0.391199	-1.003033	-0.500192	1.542922	-1.175788	-0.786604	...	-0.550822	1.204067	-0.405693
1	1.370021	0.000131	0.026406	2.181916	-1.319803	0.398234	0.386359	-0.137201	1.253952	0.923187	...	-1.098029	-0.651903	-0.405693
2	-1.346829	-1.142366	0.026406	-1.421794	1.492426	-1.140536	1.123511	1.542922	-0.653365	-1.078446	...	-0.169221	0.270719	-1.658922
3	0.381935	0.089165	0.026406	0.399005	-0.344558	0.536990	-0.916422	-0.137201	0.898628	0.204429	...	-0.819160	-0.144313	-0.405693
4	1.037368	-0.460013	0.026406	0.291134	-0.114434	0.359969	1.383517	-0.137201	1.262585	-0.030343	...	-2.240348	-0.398119	0.847536
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7171	1.616809	-1.078392	0.026406	1.214731	-0.767307	0.749052	0.219439	-0.137201	0.867013	0.123482	...	1.333115	-1.101139	-0.405693
7172	-0.002170	1.097934	0.026406	0.747680	0.304147	0.766157	0.377383	-0.137201	0.499780	1.247078	...	0.417022	-0.025044	-0.405693
7173	-0.394103	-0.956604	0.026406	-0.012883	-0.652965	-0.456153	0.647829	1.542922	-1.502681	0.048806	...	-1.076546	1.358226	-1.658922
7174	-0.634664	-0.803959	0.026406	-0.583391	0.308335	-0.365591	-0.670343	-0.137201	-1.231132	-0.189245	...	0.597553	0.728885	3.353995
7175	-1.650146	0.714619	0.026406	-1.237216	-0.155495	-1.336098	-0.685862	1.542922	0.210756	-1.032809	...	-0.883419	-0.132640	0.847536

7176 rows × 82 columns

Selección de características.

In [15]:
```
# Se elimina las Características correlacionadas de todo el conjunto de datos.
X1 = X # Conjunto de Características.
y  = Características.falla  # Etiquetas.
Características_correlacionadas_1 = tsfel.correlated_features(X1)  # Verifica correlación.
X1.drop(Características_correlacionadas_1, axis=1, inplace=True)   # Elimina las características correlacionadas.
print(f"'X_1' longitud: {X1.shape}") # Muestra el tamaño del conjunto total de características.
```

```
'X_1' longitud: (10252, 82)
```

Método RFECV.

In [16]:
```
Estimador=SVC(kernel="linear") # Estimador para obtener los coeficientes de peso y puntajes de validación cruzada.
rfecv = RFECV(estimator=Estimador, step=1, cv=StratifiedKFold(4),scoring='accuracy') # Aplica el método RFECV.
rfetrain=rfecv.fit(X1, y)  # Se ejecuta el método en el conjunto de datos especificado.
print('Número óptimo de características :', rfecv.n_features_) # Indica el número óptimo de características seleccionadas.
```

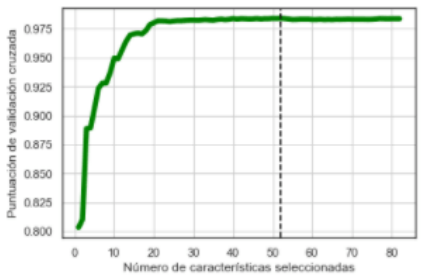
```
Número óptimo de características : 52
```

In [17]:
```
# Guardamos las características seleccionadas
Características_seleccionadas = X1.iloc[:, rfecv.support_]
Características_seleccionadas.to_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características_seleccionadas.csv",index=False)
```

Gráfico de características seleccionadas.

In [18]:
```
plt.figure()
sns.set(style='white',palette='dark', font_scale=1) # Estilo y escala de la gráfica
plt.xlabel("Número de características seleccionadas") # Título del eje x
plt.ylabel("Puntuación de validación cruzada") # Título del eje y
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_ ,color='green',linewidth=5)# Grafica los resultados
print('Número óptimo de características :', rfecv.n_features_) # Imprime el número optimo de características
plt.axvline(x=52,color="black", linestyle="--")
plt.grid() # Genera cuadrícula
plt.show() # Grafica
```

```
Número óptimo de características : 52
```



Lectura de características seleecionadas.

In [19]:
```
# Lee el conjunto de características seleccionadas y guardadas
X2= pd.read_csv("C:/Users/josel/Desktop/Código_TIC/ME1/Características_seleccionadas.csv",header=0)
X2
```

Out[19]:

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean absolute diff	0_Median frequency	0_Skewness	...	3_Centroid	3_Max power spectrum	3_Mear
0	3.089500	0.384194	10.347798	-0.839346	16.13430	0.188574	13.333333	10.480130	8.333333	0.058340	...	0.362944	0.338706	-0.0667
1	3.822627	0.254412	17.501360	-1.091989	11.85680	0.110701	13.333333	9.522101	6.666667	-0.527985	...	0.261787	0.448700	-0.5780
2	2.977592	0.355117	9.742185	-0.816881	14.28810	0.221902	13.333333	8.224173	8.333333	-0.240691	...	0.332708	0.325764	-2.0491
3	2.427698	0.249948	7.649475	-0.626452	11.26140	0.270927	15.000000	5.151102	6.666667	0.072969	...	0.346465	0.287816	-0.1404
4	2.038543	0.338758	5.822616	-0.232435	8.82280	0.236482	15.000000	6.138875	6.666667	-0.143593	...	0.446760	0.145798	1.4211
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10247	0.872752	0.264036	2.763622	-0.734197	3.37051	0.470541	15.000000	1.331960	5.000000	0.124085	...	0.385590	0.311102	0.0036
10248	1.992187	0.408660	5.362023	1.281796	6.87850	0.409349	13.333333	3.563217	5.000000	-0.786019	...	0.306881	0.158349	-1.5421
10249	3.006687	0.295486	9.659995	-1.234963	9.82473	0.311170	13.333333	4.091517	3.333333	-0.074642	...	0.299837	0.348965	-1.4182
10250	1.945077	0.275813	5.182773	-0.500194	10.63980	0.161888	15.000000	4.464654	5.000000	0.293210	...	0.280207	0.275629	-1.1365
10251	2.449425	0.222164	8.118180	-1.208222	8.04652	0.270213	13.333333	2.932927	3.333333	-0.259535	...	0.259193	0.224303	0.0396

10252 rows × 52 columns

División de características seleccionadas 70-30.

In [20]:
```
X2
C1=X2
y = Características.falla
# Divide el conjunto de características en 70% para entrenamiento y 30% para prueba
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X2, y, test_size=0.3, random_state=42)
print("Ejemplos usados para entrenar: (X_train): ", len(X_train_1))
print("Ejemplos usados para probar: (X_test): ", len(X_test_1))
```

```
Ejemplos usados para entrenar: (X_train):  7176
Ejemplos usados para probar: (X_test):  3076
```

Estandarización de características selecionadas.

In [21]:
```
# Estandariza el conjunto de características seleccionadas.
columnas_1=C1.columns
Estandarización_1 = preprocessing.StandardScaler()
X_train_1 = Estandarización_1.fit_transform(X_train_1)
X_test_1 = Estandarización_1.transform(X_test_1)
X_train_1=pd.DataFrame(X_train_1,columns=[columnas_1])
X_test_1=pd.DataFrame(X_test_1,columns=[columnas_1])
```

Datos para entrenamiento.

In [22]:
```
X_train_1 # Datos de entrenamiento
```

Out[22]:

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean absolute diff	0_Median frequency	0_Skewness	...	3_Centroid	3_Max power spectrum	3_Mear
0	-0.837114	-0.363985	-0.726475	-0.391199	-1.003033	-0.500192	1.542922	-0.786604	-1.334566	0.542261	...	-0.974830	-1.630120	0.7870
1	1.370021	0.000131	2.181916	-1.319803	0.398234	0.386359	-0.137201	0.923187	0.159132	-0.784316	...	-0.249386	0.520605	-0.5918
2	-1.346829	-1.142366	-1.421794	1.492426	-1.140536	1.123511	1.542922	-1.078446	-1.334566	1.811196	...	-0.575900	-0.490431	0.4703
3	0.381935	0.089165	0.399005	-0.344558	0.536990	-0.916422	-0.137201	0.204429	0.159132	0.354827	...	0.866605	1.859503	-0.7819
4	1.037368	-0.460013	0.291134	-0.114434	0.359969	1.383517	-0.137201	-0.030343	-1.334566	-0.982468	...	1.495440	-1.480705	-0.7946
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7171	1.616809	-1.078392	1.214731	-0.767307	0.749052	0.219439	-0.137201	0.123482	-1.334566	0.166532	...	-0.230778	0.370054	-0.8946
7172	-0.002170	1.097934	0.747680	0.304147	0.766157	0.377383	-0.137201	1.247078	1.652830	-0.824541	...	1.326539	-0.070946	0.1475
7173	-0.394103	-0.956604	-0.012883	-0.652965	-0.456153	0.647829	1.542922	0.048806	0.905981	0.348589	...	-0.616475	-0.349229	1.2759
7174	-0.634864	-0.803959	-0.583391	0.308335	-0.365591	-0.670343	-0.137201	-0.189245	0.905981	1.419395	...	-1.284535	-1.103111	0.8500
7175	-1.650146	0.714619	-1.237216	-0.155495	-1.336098	-0.685862	1.542922	-1.032809	0.905981	-0.354130	...	1.901859	0.886808	0.1052

7176 rows × 52 columns

Datos para prueba.

In [23]:
```
X_test_1 # Datos de Prueba
```

Out[23]:

	0_Area under the curve	0_Centroid	0_Interquartile range	0_Kurtosis	0_Max	0_Max power spectrum	0_Maximum frequency	0_Mean absolute diff	0_Median frequency	0_Skewness	...	3_Centroid	3_Max power spectrum	3_Mear
0	-0.205313	-0.821414	-0.620577	0.635912	0.139574	-1.092236	-0.137201	-0.132963	0.905981	2.141443	...	1.563254	0.455171	0.6188
1	-1.031385	2.529602	-1.199337	6.523125	0.359154	-1.178689	1.542922	-0.613672	0.159132	4.376280	...	0.867638	2.106753	-0.6127
2	0.076278	0.449525	0.567575	-1.198916	-0.240105	-0.752857	-0.137201	0.173244	0.905981	-0.361632	...	0.714638	0.608405	-0.1993
3	-0.743608	1.202710	-0.755837	-0.103634	-0.872367	-0.890895	-0.137201	-1.074916	-1.334566	1.612157	...	0.480257	-1.081250	-0.0304
4	-0.274028	0.701280	-0.468992	-0.108013	0.052421	0.436976	-0.137201	-0.719686	-1.334566	1.174709	...	-0.491229	0.229740	-0.4302

3071	-0.411494	-0.906819	-0.886903	2.381223	0.343390	-0.141683	-1.817323	-0.284262	0.159132	2.400613	...	-0.483527	0.145107	0.4042
3072	0.209750	0.650684	-0.447703	-0.927033	-0.389781	2.397301	-1.817323	-0.761641	-1.334566	-0.463535	...	0.231527	1.631222	-4.7813
3073	1.440193	-0.106878	1.023101	-0.222937	1.402718	0.741215	-1.817323	1.280697	0.905981	-0.167991	...	0.234764	-0.048149	-0.3971
3074	0.545734	0.806396	0.442619	-0.638583	0.143885	0.284548	-0.137201	0.236951	0.905981	-0.137192	...	-0.182850	2.042965	-0.4763
3075	1.946263	0.900588	2.488804	-0.237547	2.034425	0.350999	-0.137201	2.740350	1.652830	-0.353851	...	0.748427	0.153686	-0.1677

3076 rows × 52 columns

Entrenamiento de modelos con características originales y seleccionadas.

In [24]:
```
# Construcción del modelo con tres funsiones kernel
random_state = 42
# Características originales
# Asignamos una lista donde se guardan los nombres del algoritmo con las tres funsiones kernel
Nombre_de_clf = [ "Support Vector Machine ",
                  "SVM - Kernel lineal",
                  "SVM - Kernel radial",
                  "SVM - kernel polynomial"]
# Asignamos una lista donde se guardan los algoritmos con las tres funsiones kernel
Tipo_de_kernel = [SVC(),SVC(kernel="linear", random_state=random_state),
                  SVC(kernel="rbf", random_state=random_state),
                  SVC(kernel="poly", random_state=random_state)]

#Características seleccionadas
# Asignamos una lista donde se guardan los nombres del algoritmo con las tres funsiones kernel
Nonbre_de_clf_1 = ["Support Vector Machine ",
                   "SVM - Kernel lineal",
                   "SVM - Kernel radial",
                   "SVM - kernel polynomial"]
# Asignamos una lista donde se guardan los algoritmos con las tres funsiones kernel
Tipo_de_kernel_1 = [SVC(),SVC(kernel="linear", random_state=random_state),
                    SVC(kernel="rbf", random_state=random_state),
                    SVC(kernel="poly", random_state=random_state)]

clfs_caract_originales = dict(zip(Nombre_de_clf, Tipo_de_kernel))  # Combina las dos listas
clfs_caract_seleccionadas = dict(zip(Nonbre_de_clf_1, Tipo_de_kernel_1 ))  # Combina las dos listas
```

In [25]:
```
# Características originales
print("Entrenamiento con características originales:")
for nombre_de_clasificador, clf in clfs_caract_originales.items(): # Entrena el modelo con el algoritmo almacenado en la lista
    clf.fit(X_train, y_train)
    clfs_caract_originales[nombre_de_clasificador] = clf
    print(nombre_de_clasificador, "Entrenamiento: Terminado!")


# Características seleccionadas
print("Entrenamiento con características seleccionadas: ")
for nombre_de_clasificador_1, clf_1 in clfs_caract_seleccionadas.items(): # Entrena el modelo con el algoritmo almacenado en la
    clf_1.fit(X_train_1, y_train_1)
    clfs_caract_seleccionadas [nombre_de_clasificador_1] = clf_1
    print(nombre_de_clasificador_1, "Entrenamiento: Terminado!")
```

```
Entrenamiento con características originales:
Support Vector Machine  Entrenamiento: Terminado!
SVM - Kernel lineal Entrenamiento: Terminado!
SVM - Kernel radial Entrenamiento: Terminado!
SVM - kernel polynomial Entrenamiento: Terminado!
Entrenamiento con características seleccionadas:
Support Vector Machine  Entrenamiento: Terminado!
SVM - Kernel lineal Entrenamiento: Terminado!
SVM - Kernel radial Entrenamiento: Terminado!
SVM - kernel polynomial Entrenamiento: Terminado!
```

Resultados de entrenamiento.

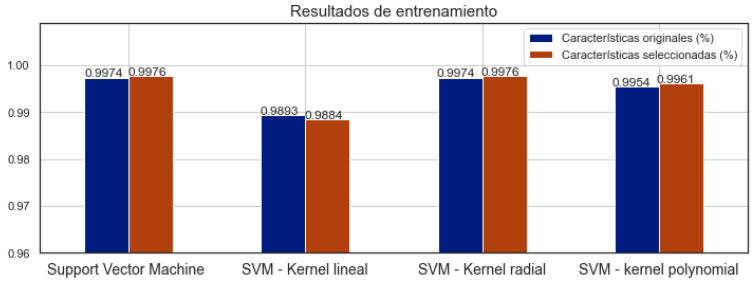
In [26]:
```
# Características originales
Resultados = [] # Estamos asignando la lista vacía a la variable resultados
for nombre_de_clasificador, clf in clfs_caract_originales.items(): # entrena el algoritmo con las tres funsiones kernel
    y_pred = clf.predict(X_train)
    Resultados.append(accuracy_score(y_train, y_pred)) # Agrega el resultado a la lista vacia creada "resultado=[]"

# Características seleccionadas
Resultados_1 = []
for nombre_de_clasificador_1, clf_1 in clfs_caract_seleccionadas.items(): # entrena el algoritmo con las tres funsiones kernel
    y_pred_1 = clf_1.predict(X_train_1)
    Resultados_1.append(accuracy_score(y_train_1, y_pred_1)) # Agrega el resultado a la lista vacia creada "resultado=[]"

Resultados = pd.DataFrame({"Características originales (%)": Resultados, "Características seleccionadas (%)": Resultados_1},
                          index=Nombre_de_clf)
Resultados_de_entrenamiento=Resultados*100
Resultados_de_entrenamiento= np.round(Resultados_de_entrenamiento, decimals = 2)
Resultados_de_entrenamiento
```

Out[26]:

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.74	99.76
SVM - Kernel lineal	98.93	98.84
SVM - Kernel radial	99.74	99.76
SVM - kernel polynomial	99.54	99.61

Representación gráfica de los resultados de entrenamiento.

In [27]:
```
sns.set(style='white',palette='dark') # Estilo y escala
ax = Resultados.plot.bar(figsize=(12, 4)) # Grafica los resultados almacenados en resultados_1
for p in ax.patches: # Ubica los valores de ax en el grafico de barras, centra y redondea.
    ax.annotate(str(p.get_height().round(4)), (p.get_x()*0.995, p.get_height()*1.0001))

plt.ylim((0.96,1.009)) # Escala del eje y.
plt.xticks(rotation=0,fontsize=14) # Orientación de las etiquetas del eje x.
plt.title("Resultados de entrenamiento",fontsize=15)  # Título.
plt.grid()  # Mostrar cuadrícula.
plt.show()  # Graficar.
```



Resultados de Prueba.

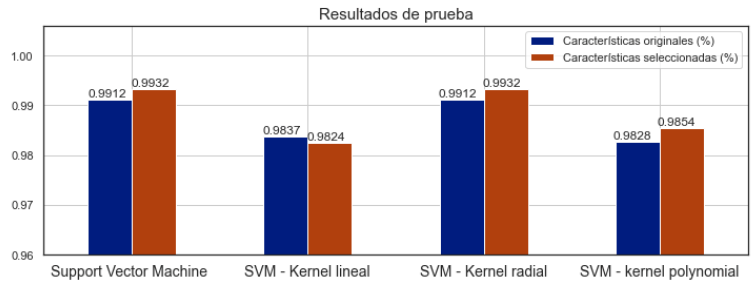
In [28]:
```
# Características originales
Resultados = [] # Estamos asignando la lista vacía a la variable resultados
for nombre_de_clasificador, clf in clfs_caract_originales.items():
    y_pred = clf.predict(X_test) # Realiza las predicciones con el conjunto de datos de prueba  para las tres funsiones kernel
    Resultados.append(accuracy_score(y_test, y_pred)) # Agrega el resultado a la lista vacia creada "resultado=[]"

# Características seleccionadas
Resultados_1 = []  # Estamos asignando la lista vacía a la variable resultados
for nombre_de_clasificador_1, clf_1 in clfs_caract_seleccionadas.items():
    y_pred_1 = clf_1.predict(X_test_1) # Realiza las predicciones.
    Resultados_1.append(accuracy_score(y_test_1, y_pred_1)) # Agrega el resultado a la lista vacia creada "resultado=[]"
# Guarda los resultados en un DataFrame
Resultados_1 = pd.DataFrame({"Características originales (%)": Resultados, "Características seleccionadas (%)": Resultados_1},
                            index=Nombre_de_clf)
Resultados_de_prueba=Resultados_1*100
Resultados_de_prueba= np.round(Resultados_de_prueba, decimals = 2)
Resultados_de_prueba
```

Out[28]:

	Características originales (%)	Características seleccionadas (%)
Support Vector Machine	99.12	99.32
SVM - Kernel lineal	98.37	98.24
SVM - Kernel radial	99.12	99.32
SVM - kernel polynomial	98.28	98.54

Representación gráfica de los resultados de prueba.

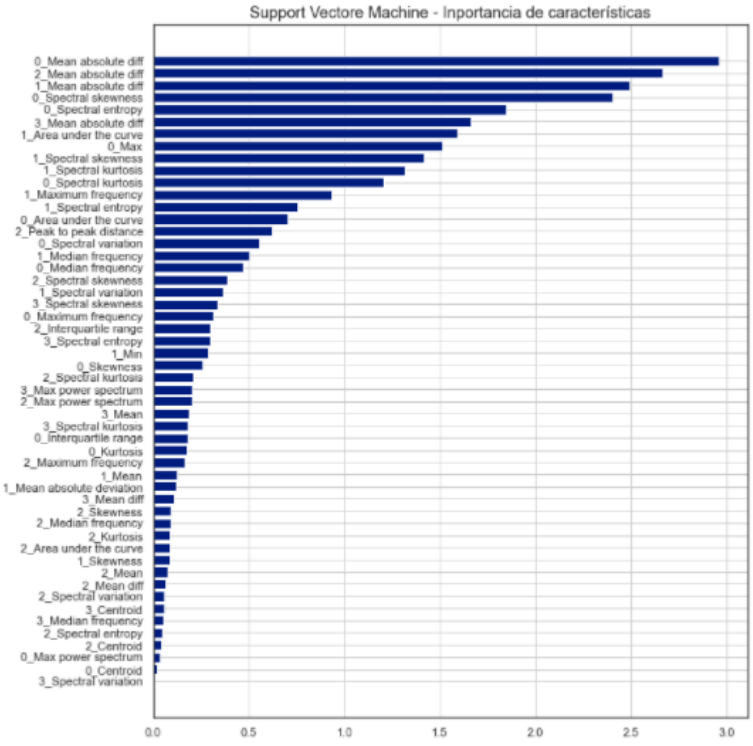
In [29]:
```
sns.set(style='white',palette='dark')
ax = Resultados_1.plot.bar(figsize=(12, 4)) # Grafica los resultados almacenados en resultados_1
for p in ax.patches: # Ubica los valores de ax en el grafico de barras, centra y redondea.
    ax.annotate(str(p.get_height().round(4)), (p.get_x()*0.995, p.get_height()*1.0005))
plt.ylim((0.96,1.006))
plt.xticks(rotation=0,fontsize=14) # Orientación de las etiquetas del eje x.
plt.title("Resultados de prueba",fontsize=15) # Título del gráfico.
plt.grid() # Mostrar cuadrícula.
plt.show() # Graficar.
```



Gráfica de importancia de caracteristcas seleccionadas.

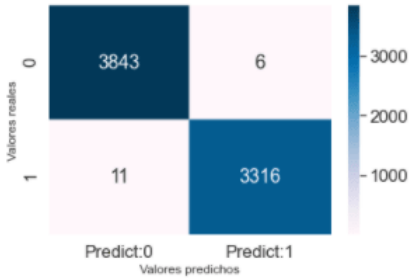
In [30]:
```
# Crea dos listas con el nombre y el algoritmo
clf_nombre = [ "Support Vector Machine"] # Define el nombre del clasificador
clf_kernel = [ SVC(kernel="linear")]  # Define el algoritmo
Diccionario= dict(zip(clf_nombre, clf_kernel)) # Combina las dos listas
for nombre, clf_2 in Diccionario.items():
    clf_2.fit(X_train_1, y_train_1) # Entrena el modelo con el algoritmo almacenado en la lista
    Diccionario[nombre] = clf_2 # identifica el nombre del algoritmo
```

```
fig=plt.figure(figsize=(10,12)) # Define el tamañp de la figura
sns.set(style='white', palette='dark', font_scale=1) # Establece el estilo y escala de la gráfica
importance_rfe = abs(Diccionario["Support Vector Machine"].coef_[0]) # Obtiene los coeficientes de peso
# Grafica las características de acuardo a sus coeficientes de peso.
plt.barh(X2.columns.values[importance_rfe.argsort()], importance_rfe[importance_rfe.argsort()])
ax = plt.axes()
plt.grid() # Mostrar cuadrícula
plt.title("Support Vectore Machine - Inportancia de características",fontsize=15) # Titulo y tamaño de letra
plt.show() # Mostrar
```



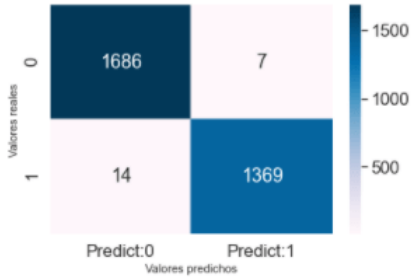
Matriz de confusión del modelo entrenado.

In [31]:
```
Modelo = SVC(kernel="rbf") # Define el estimador
Modelo.fit(X_train_1, y_train_1)  # Entrena el modelo con el conjunto de datos de entrenamiento etiquetado
y_pred = Modelo.predict(X_train_1)

sns.set(font_scale=1.5) # Escala
cm = confusion_matrix(y_train_1, y_pred) # Construye la matriz de confusión
cm_matrix = pd.DataFrame(data=cm, columns=['Predict:0', 'Predict:1'],  index=['0', '1'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='PuBu') # cmap='PuBu'= colores de la MC
plt.ylabel('Valores reales', size = 12) # Titulo del eje y
plt.xlabel('Valores predichos', size = 12) # Titulo del eje x
plt.show()
```



Matriz de confusión del modelo probado.

In [32]:
```
sns.set(font_scale=1.5, ) # Escala
y_test_pred= Modelo.predict(X_test_1) # Se prueba el modelo entrenado
cmp1 = confusion_matrix(y_test_1, y_test_pred) # Construye la matriz de confusión
cm_matrix = pd.DataFrame(data=cmp1, columns=['Predict:0', 'Predict:1'],  index=['0', '1'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='PuBu')
plt.ylabel('Valores reales', size = 11)    # Titulo del eje y
plt.xlabel('Valores predichos', size = 11) # Titulo del eje x
plt.show()
print(classification_report(y_test_1, y_test_pred)) # Imprime el reporte de clasificación de ambas clases
```




```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1693
         1.0       0.99      0.99      0.99      1383

    accuracy                           0.99      3076
   macro avg       0.99      0.99      0.99      3076
weighted avg       0.99      0.99      0.99      3076
```

Cálculo de métricas de evaluación de la matriz de confusión.

In [33]:
```
# Calcula la exactitud
exactitud = accuracy_score(y_test_1, y_test_pred)
exactitud= np.round(exactitud*100, decimals = 2)
# Calcula la precisión
precision = precision_score(y_test_1, y_test_pred)
precision= np.round(precision*100, decimals = 2)
# Calcula la sensibilidad
sensibilidad = recall_score(y_test_1, y_test_pred)
sensibilidad= np.round(sensibilidad*100, decimals = 2)
# Calcula la especificidad
Especificidad=1686/(1686+7)
Especificidad= np.round(Especificidad*100, decimals = 2)
# Calcula el pubtaje f1
puntajef1 = f1_score(y_test_1, y_test_pred)
puntajef1= np.round(puntajef1*100, decimals = 2)

# Crea un Dataframe donde se almacenaran los resultados
resultados = pd.DataFrame({
'CLASIFICADORES': ['Support Vector Machine'],
'Accuary': [exactitud], # Título de columna y valores asociados
'Precisión': [precision],
'Recall': [sensibilidad],
'Especificidad': [Especificidad],
'F1_score': [puntajef1]})
resultados
```

Out[33]:

	CLASIFICADORES	Accuary	Precisión	Recall	Especificidad	F1_score
0	Support Vector Machine	99.32	99.49	98.99	99.59	99.24

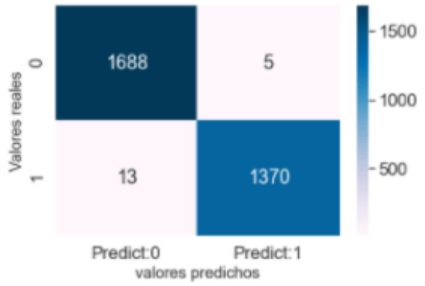
Optimización de hiperparámetros.

In [34]:
```
# Se aplica una validación cruzada de dos pliegues.
folds = KFold(n_splits = 5, shuffle = True, random_state = 42)
# Se establece la lista de hiperparámetros a optimizar
hyper_params = [ {'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 0.5 ,5,10],'C': [0.1,0.5,1,3,5,10,15, 25, 50, 100, 1000]}]
model= SVC(kernel="rbf") # Se establece el estimador
# Se selecciono como métrica la sensibilidad, dado que se quiere identificar la mayor cantidad de instancias positivas
model_cv = GridSearchCV(estimator = model, param_grid = hyper_params,scoring = 'recall',cv = folds,verbose = 1,
                        return_train_score=True,)
model_cv.fit(X_train_1, y_train_1) # Se ajusta el modelo al conjunto de datos de entrenamiento
y_pred_op = model_cv.predict(X_test_1) # Realiza las predicciones de cada pliegue.
gscores_recall = model_cv.cv_results_
best_score = model_cv.best_score_  # Mejor puntaje
best_hyperparams = model_cv.best_params_ # Mejores hiperparámetros
print(" El mejor puntaje es de {0} :hiperparámetros seleccionados  {1}".format(best_score, best_hyperparams))
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
 El mejor puntaje es de 0.9933746398790275 :hiperparámetros seleccionados  {'C': 5, 'gamma': 0.01}
```

Prueba del modelo con hiperparámetros optimizados.

In [35]:
```
sns.set(font_scale=1.5) # Escala
Modelo = SVC(kernel="rbf",C=5,gamma=0.01,random_state=42) # Estimador
Modelo.fit(X_train_1, y_train_1) # Entrena el modelo
y_test_op= Modelo.predict(X_test_1) # Prueba el modelo entrenado

# Construye la matriz de confusión
cmp1 = confusion_matrix(y_test_1, y_test_op )
cm_matrix = pd.DataFrame(data=cmp1, columns=['Predict:0', 'Predict:1'],  index=['0', '1'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='PuBu')
plt.ylabel('Valores reales', size = 15)
plt.xlabel('valores predichos', size = 15)
plt.show()
print(classification_report(y_test_1, y_test_op))
```



Cálculo de métricas de evaluación, para el modelo con hiperparámetros optimizados.

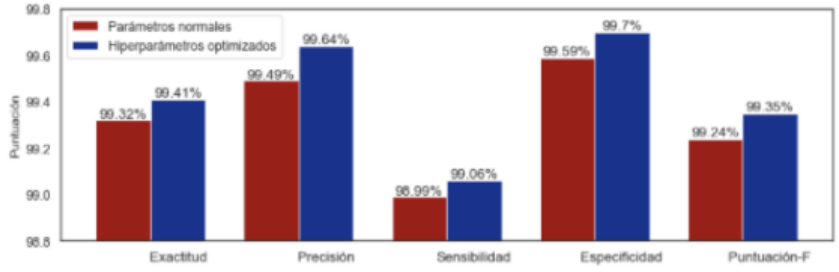
In [36]:
```
# Calcula la exactitud
exactitud_op = accuracy_score(y_test_1, y_test_op)
exactitud_op= np.round(exactitud_op*100, decimals = 2)
# Calcula la precisión
precision_op = precision_score(y_test_1, y_test_op)
precision_op= np.round(precision_op*100, decimals = 2)
# Calcula la sensibilidad
sensibilidad_op = recall_score(y_test_1, y_test_op)
sensibilidad_op= np.round(sensibilidad_op*100, decimals = 2)
# Calcula la especificidad
Especificidad_op=1688/(1688+5)
Especificidad_op= np.round(Especificidad_op*100, decimals = 3)
# Calcula el puntaje f1
puntajef1_op = f1_score(y_test_1, y_test_op)
puntajef1_op= np.round(puntajef1_op*100, decimals = 2)
# Crea un Dataframe donde se almacenaran los resultados
resultados_op = pd.DataFrame({
'CLASIFICADORES': ['Support Vector Machine'],
'Accuary': [exactitud_op], # Título de columna y valores asociados
'Precisión': [precision_op],
'Recall': [sensibilidad_op],
'Especificidad': [Especificidad_op],
'F1_score': [puntajef1_op],})
resultados_op
```

Out[36]:

	CLASIFICADORES	Accuary	Precisión	Recall	Especificidad	F1_score
0	Support Vector Machine	99.41	99.64	99.06	99.705	99.35

Gráfico de comparación de hiperparámetros optimizados y parámetros normales.

In [37]:
```
Datos1 = [99.32,99.49,98.99,99.59,99.24] # Resultados del modelo con parámetros normales
Datos2 = [99.41,99.64,99.06,99.70,99.35] # Resultados del modelo con Hiperparámetros optimizados
labels = ['Exactitud','Precisión','Sensibilidad',"Especificidad",'Puntuación-F']

plt.figure(figsize=(13,4)) # Tamaño de Figura
sns.set(style='white', palette='dark',font='sans-serif', font_scale=1.1) # Estilo y escala
bar_width = 0.37 # ancho de barras
# Dibujo
plt.bar(np.arange(5), Datos1, label = 'Parámetros normales', color = 'r', alpha = 0.9, width = bar_width)
plt.bar(np.arange(5)+bar_width, Datos2, label = 'Hiperparámetros optimizados', color = 'b', alpha = 0.9, width = bar_width)

plt.ylabel('Puntuación',fontsize=12)
plt.xticks(np.arange(5)+bar_width,labels,fontsize=13)
plt.ylim([98.8,99.8])
# Agregua etiquetas numéricas a cada gráfico de barras
for x1,y1 in enumerate(Datos1):
    plt.text(x1, y1+0.01,'%s' %y1+'%', ha='center')
for x2,y2 in enumerate(Datos2):
    plt.text(x2+bar_width, y2+0.01, '%s' %y2+'%', ha='center')
# Mostrar leyenda
plt.legend()
#plt.grid()
plt.show()
```



Validación cruzada.

In [38]:
```
clf= SVC(kernel="rbf",C=5,gamma=0.01,random_state=42)
# Realiza validación cruzada de 10 repeticiones.
cv_scores = cross_validate(estimator = clf, X = X_train_1, y= y_train_1, scoring= ('precision'), cv= 10,
                                                                    return_train_score = True)
```

```
cv_scores = pd.DataFrame(cv_scores) # Fuarda en una tabla los resultados de CV.
scores=cv_scores*100 # Multiplica por 100 los puntajes de validaciuón cruzada.
Puntajes = scores.drop( ['fit_time','score_time'], axis=1) # Elimina columnas que indica el tiempo de cálculo.
print(f"Precisión Promedio: {(cv_scores.mean())*100}") # Imprime la precisión promedio de entrenamiento y prueba.
# Imprime el valor de desviación estandar de cada valor con respecto a la media.
print(f"Desviación estandar: {cv_scores.std()}")
Puntajes= np.round(Puntajes, decimals = 2)
Puntajes
```

```
Precisión Promedio: fit_time       60.523093
score_time      6.637061
test_score     99.191398
train_score    99.923110
dtype: float64
Desviación estandar: fit_time       0.046393
score_time     0.004833
test_score     0.004641
train_score    0.000275
dtype: float64
```

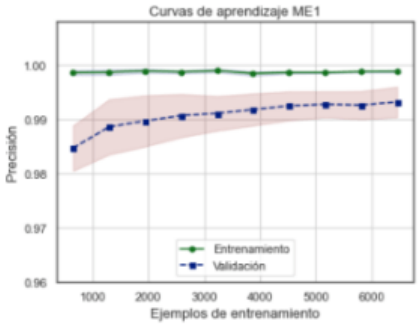
Out[38]:

	test_score	train_score
0	99.10	99.93
1	100.00	99.93
2	99.10	99.90
3	99.10	99.93
4	99.70	99.97
5	98.22	99.87
6	99.10	99.93
7	99.10	99.93
8	99.10	99.93
9	99.40	99.90

Curvas de aprendizaje.

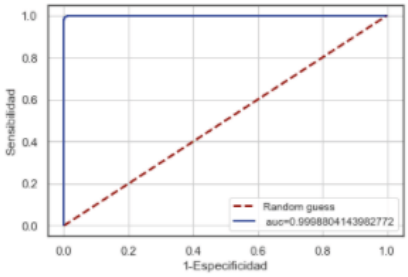
In [39]:
```
# Ejemplo Curvas de aprendizaje
clf= SVC(kernel="rbf",C=5,gamma=0.01,random_state=42)
# n_jobs=-1 significa usar todos los procesadores.
# cv Determina la estrategia de división de validación cruzada.
# train_sizes=np.linspace(0.1, 1.0, 10) Números relativos o absolutos de ejemplos de capacitación que se utilizarán para generar
train_sizes, train_scores, test_scores = learning_curve(estimator=clf,
                        X=X_train_1, y=y_train_1,
                        train_sizes=np.linspace(0.1, 1.0, 10), cv=10,
                        n_jobs=-1)

train_mean = np.mean(train_scores, axis=1) # Puntaje de precisión promedio de validación cruzada "entrenamiento"
train_std = np.std(train_scores, axis=1)   # Desviación estandar de los puntajes de precision con respecto a la media.
test_mean = np.mean(test_scores, axis=1)   # Puntaje de precisión promedio de validación cruzada "Prueba"
test_std = np.std(test_scores, axis=1)     # Desviación estandar de los puntajes de precision con respecto a la media.
```

In [40]:
```
# graficando las curvas
plt.figure(figsize=(6,4.5)) # Tamaño de gráfico
sns.set(style='white',palette='dark', font_scale=1) # Estilo  y escala de la gráfica
# TRAZA la curva de aprendizaje de entrenamiento
plt.plot(train_sizes, train_mean, color='g', marker='o', markersize=5,label='Entrenamiento')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='b')
# TRAZA la curva de aprendizaje de validación
plt.plot(train_sizes, test_mean, color='b', linestyle='--',  marker='s', markersize=5, label='Validación')
plt.fill_between(train_sizes, test_mean + test_std,  test_mean - test_std, alpha=0.15, color='r')
plt.gca().set_ylim([0.96, 1.008]) # Escala del eje y
plt.grid() # Mostrar cuadrícula en la gráfica
plt.title('Curvas de aprendizaje ME1',fontsize=13)  # Título y tamaño de letra
plt.legend(loc='lower center') # Posición de las leyendas
plt.xlabel('Ejemplos de entrenamiento',fontsize=13) # Título del eje x tamaño de letra
plt.ylabel('Precisión',fontsize=13) # Título del eje y tamaño de letra
plt.show() # Muestra
```



Curva ROC.

In [41]:
```
sns.set(style='white',palette='dark') # Estipo y escala de la gráfica.
clf = SVC(kernel='rbf',C=5,gamma=0.01,random_state=42, probability=True) # ESTIMADOR.
clf.fit(X_train_1, y_train_1) # Entrenamiento.
y_pred = clf.predict(X_test_1) # Prueba.
y_pred_proba = clf.predict_proba(X_test_1)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test_1,  y_pred_proba)
auc = metrics.roc_auc_score(y_test_1, y_pred_proba)
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', label='Random guess') # Etilo de gráfico.
plt.plot(fpr,tpr,label=" auc="+str(auc)) # Construye la gráfica.
plt.xlabel('1-Especificidad' )# Título del eje x tamaño de letra
plt.ylabel('Sensibilidad') # Título del eje y tamaño de letra
plt.legend(loc=4,fontsize=10) # Leyebdas y tamaño de letra.
plt.grid() # mostrar cuadrícula.
plt.show()
```



Comparación con otros modelos de Machine Learning.

Clasificadores.

- Random Forest
- Regresión Logística
- Support Vector Machine
- Albol de decisión
- k vecinos más próximos (KNeighbor)
- XGBoost

In [42]:
```
# Crea una lista que contiene a los clasificadores.
CLF = [
    RandomForestClassifier(),
    xgb.XGBClassifier(n_jobs = -1),
    LogisticRegression(),
    KNeighborsClassifier(),
    SVC(kernel="rbf"),
    DecisionTreeClassifier(),]
xgb.set_config(verbosity=0)
```

In [43]:
```
CLF_columns = []
CLF_compare = pd.DataFrame(columns = CLF_columns)
row_index = 0
for alg in CLF:
    predicted = alg.fit(X_train_1, y_train_1).predict(X_test_1) # Entrena y prueba los clasificadores de la lista CLF.
    fp, tp, th = roc_curve(y_test_1, predicted)
    CLF_name = alg.__class__.__name__
    CLF_compare.loc[row_index,'Algoritmos'] = CLF_name
    # Calcula las metricas de evaluación de cada clasificador.
    CLF_compare.loc[row_index, 'Train Exactitud'] = round(alg.score(X_train_1, y_train_1)*100, 2)
    CLF_compare.loc[row_index, 'Test Exactitud'] = round(alg.score(X_test_1, y_test_1)*100, 2)
    CLF_compare.loc[row_index, 'Precisión'] = round(precision_score(y_test_1, predicted)*100,2)
    CLF_compare.loc[row_index, 'Sensibilidad'] = round(recall_score(y_test_1, predicted)*100,2)
    CLF_compare.loc[row_index, 'F1 score'] = round(f1_score(y_test_1, predicted)*100,2)
    CLF_compare.loc[row_index, 'AUC'] = round(roc_auc_score(y_test_1, predicted)*100,2)
    row_index+=1
# Los resultados se ordenan de mayor a menor.
CLF_compare.sort_values(by = ['Test Exactitud'], ascending = False, inplace = True)
CLF_compare
```

Out[43]:

	Algoritmos	Train Exactitud	Test Exactitud	Precisión	Sensibilidad	F1 score	AUC
4	SVC	99.76	99.32	99.49	98.99	99.24	99.29
1	XGBClassifier	100.00	99.28	99.20	99.20	99.20	99.28
0	RandomForestClassifier	100.00	98.63	98.48	98.48	98.48	98.62
2	LogisticRegression	98.80	98.28	98.19	97.98	98.08	98.25
3	KNeighborsClassifier	98.51	97.07	97.50	95.95	96.72	96.97
5	DecisionTreeClassifier	100.00	96.39	95.49	96.53	96.01	96.40

In [44]:
```
fig=plt.figure(figsize=(10,4))
sns.set(style='white',palette='dark',font_scale=1.1)
# Construir datos
CL_results = CLF_compare["Test Exactitud"] # Genera un gráfico de barras de la columna Test Exactitud.
plt.bar(range(6), CL_results, width=0.50,align = 'center', color = ["dimgrey","gray","grey","darkgray","darkgrey","silver"],
        alpha = 1)
```

```
# Agregar etiqueta de ejes.
plt.ylabel('Precisión (%)',fontsize=15)
plt.xticks(range(6), ['Support Vector Machine', 'XGBClassifier',"RandomForestClassifier","LogisticRegression",
                      "KNeighborsClassifier", "DecisionTreeClassifier"])

plt.ylim([93,100.5]) # Escala del eje y.
# Agregue etiquetas numéricas a cada gráfico de barras
for x,y in enumerate(CL_results):
    plt.text(x, y+0.09, '%s' %round(y,3), ha='center')

plt.xticks(rotation=0,fontsize=15)
plt.show()
```

PRUEBA DE HIPÓTESIS (CÓDIGO GENERAL)

Librerías

In [1]:
```
# Store and organize output files
from pathlib import Path
# Manipulate data
import numpy as np
import pandas as pd
# Plots
import seaborn as sns
import matplotlib.pyplot as plt
# Machine learning
from sklearn.svm import SVC
from sklearn import preprocessing
#from sklearn.externals import joblib
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import permutation_test_score
import time
# Ignore WARNING
import warnings
warnings.filterwarnings('ignore')
```

HIPÓTESIS

La hipótesis planteada en esta investigación es la siguiente:

Utilizando el método de aprendizaje de máquinas support vector machine se detectan fallas en cajas de engranajes.

Para verificar la hipótesis planteada se aplicó la prueba de permutación a los conjuntos de datos utilizados para crear los tres modelos predictivos. Además, se planteó las siguientes hipótesis estadísticas.

- H0: El rendimiento de los modelos es por casualidad.
- H1: El rendimiento de los modelos no es por casualidad.

ME1

Lectura de datos

In [2]:
```
# Lecturta de características seleccionadas guardadas y etiquetas de clase
Características_1 = pd.read_csv("C:/Users/josel/OneDrive/Escritorio/Código_TIC/ME1/Características_seleccionadas.csv",header=0)
y_1 = pd.read_csv("C:/Users/josel/OneDrive/Escritorio/Código_TIC/ME1/Características.csv",header=0)
```

In [3]:
```
#  Caracterísitcas y etiquetas
X_1 = Características_1
Y_1 = y_1.falla
```

In [4]:
```
#Estandarización
columnas_1=X_1.columns
Estandarización_1 = preprocessing.StandardScaler()
X_1_nor= Estandarización_1.fit_transform(X_1)
X_1_nor=pd.DataFrame(X_1_nor, columns=[columnas_1])
```

Prueba de permutación.

In [5]:
```
# Se define el clasificador con hiperparámetros optimizados
clf_1 = SVC(kernel="rbf",random_state=42)
cv_1 = StratifiedKFold(3, shuffle=True, random_state=0)
```

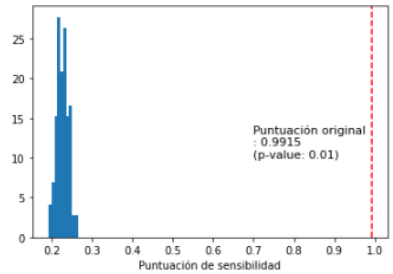
```
# Prueba de permutación
inicio = time.time()
score_datos, perm_scores_datos, pvalue_datos = permutation_test_score(
    clf_1, X_1_nor, Y_1 , scoring="recall", cv=cv_1, n_permutations=100)
fin = time.time()
print ( f"Puntuación con datos originales: {score_datos:.4f}\n(p-value: {pvalue_datos:.2f})" )
print ( f"Tiempo transcurrido: ",fin - inicio )
```

```
Puntuación con datos originales: 0.9915
(p-value: 0.01)
Tiempo transcurrido:  6273.841661691666
```

A continuación, trazamos un histograma de las puntuaciones de permutación (la distribución nula). La línea roja indica la puntuación obtenida por el clasificador sobre los datos originales.

In [6]:
```
fig, ax = plt.subplots()
ax.hist(perm_scores_datos, bins=10, density=True)
ax.axvline(score_datos, ls="--", color="r")
score_label = f"Puntuación original\n: {score_datos:.4f}\n(p-value: {pvalue_datos:.2f})"
ax.text(0.7, 10, score_label, fontsize=11)
ax.set_xlabel("Puntuación de sensibilidad")
```

Out[6]:
```
Text(0.5, 0, 'Puntuación de sensibilidad')
```



El p-valor de la prueba de permutación es menor al umbral de significancia de α=0.05 establecido por la prueba, lo que permite rechazar la hipótesis nula de que el rendimiento de los modelos es por casualidad y aceptar la hipótesis alternativa que indica que el rendimiento de los modelos no es por causalidad.