Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería Automotriz

Empleo de algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto

Trabajo de Integración Curricular o de Titulación, previo a la obtención del título de Ingeniero Automotriz

AUTOR:

Juan Pablo Medina Namicela

DIRECTOR:

Ing. Rogelio Santiago León Japa M.Sc.

Loja - Ecuador

2025

Certificación del director del trabajo de integración curricular o de titulación

Universidad Nacional de Loja

1859

Sistema de Información Académico Administrativo y Financiero - SIAAF

CERTIFICADO DE CULMINACIÓN Y APROBACIÓN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Yo, **LEON JAPA ROGELIO SANTIAGO**, director del Trabajo de Integración Curricular denominado **Empleo de algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto**, perteneciente al estudiante **JUAN PABLO MEDINA NAMICELA**, con cédula de identidad N° **1150922266**.

Certifico:

Que luego de haber dirigido el **Trabajo de Integración Curricular**, habiendo realizado una revisión exhaustiva para prevenir y eliminar cualquier forma de plagio, garantizando la debida honestidad académica, se encuentra concluido, aprobado y está en condiciones para ser presentado ante las instancias correspondientes.

Es lo que puedo certificar en honor a la verdad, a fin de que, de así considerarlo pertinente, el/la señor/a docente de la asignatura de **Integración Curricular**, proceda al registro del mismo en el Sistema de Gestión Académico como parte de los requisitos de acreditación de la Unidad de Integración Curricular del mencionado estudiante.

Loja, 4 de Febrero de 2025

ROGELIO SANTIAGO LEON JAPA

F) _____

DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR

Certificado TIC/TT.: UNL-2025-000351

1/1

Educamos para **Transformar**

Autoría del trabajo de integración curricular o de titulación

Yo, **Juan Pablo Medina Namicela**, declaro ser autor/a del presente Trabajo de Integración Curricular o de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular o de Titulación, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma:

Cédula de identidad: 1150922266

Fecha: 04 de febrero del 2025

Correo electrónico: juan.p.medina.n@unl.edu.ec

Teléfono: 0990098272

Carta de autorización del estudiante

Yo, **Juan Pablo Medina Namicela**, declaro ser autor/a del Trabajo de Integración Curricular o de Titulación denominado: **Empleo de algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto**, como requisito para optar por el título de **Ingeniero Automotriz**, autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular o de Titulación que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los 4 días del mes de febrero del año dos mil veinticinco.

Firma:

Autor/a: Juan Pablo Medina Namicela

Cédula de identidad: 1150922266

Dirección: Turunuma Alto, Loja, Ecuador

Correo electrónico: juan.p.medina.n@unl.edu.ec

Teléfono: 0990098272

DATOS COMPLEMENTARIOS:

Director del Trabajo de Integración Curricular o de Titulación: Ing. Rogelio Santiago León Japa M.Sc.

Dedicatoria

Dedico mi trabajo a mis padres, hermanos, familia y amigos; Dios los cuide y los bendiga.

“No existirá un mundo mejor si no lo construimos”

Juan Pablo Medina Namicela

Agradecimiento

Agradezco a Dios quien me dio la vida y me dio todo para vivirla, a mis padres a quienes no puedo agradecerles solo con palabras, a mi hermano Joe quien me acompañó y me motivó a no quedarme atrás, al Lic. Carlos Abraham quien me inculcó el deseo de aprender desde pequeño, a mi familia a quienes los llevo grabados en el corazón, a mis amigos, a mis compañeros/as quienes hicieron de mi estancia en la universidad el mejor lugar para aprender y a todos mis docentes de la escuela, colegio y universidad por sus enseñanzas tanto en lo profesional como en lo personal.

Extiendo mi sincero agradecimiento a mi director de tesis, el Ing. Rogelio Santiago León Japa, M.Sc. por su tiempo, dedicación y apoyo en el desarrollo de mi Trabajo de Titulación, sus conocimientos y experiencia fueron esenciales para poder cumplir con los objetivos de mi investigación.

Agradezco a la Universidad Nacional de Loja por brindarme el espacio y el lugar para llevar a cabo mi formación profesional y mi formación como persona. Agradezco a la carrera de Ingeniería Automotriz que me ha brindado su apertura en los laboratorios y talleres con los recursos necesarios para el desarrollo de mi tesis de grado.

Juan Pablo Medina Namicela

Índice de contenido

Índice de Figuras

Índice de Tablas

Índice de Anexos

1 Título

Empleo de algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto

2 Resumen

En la presente investigación se aplicaron algoritmos y modelos de Machine Learning dentro de Matlab, con el objetivo de clasificar 4 estados diferentes del funcionamiento del motor: en óptimas condiciones, con una bujía defectuosa, con un filtro de aire obstruido y con la aplicación de ambos fallos a la vez. Para ello, se recopilaron los datos de voltaje del sensor MAP de un motor Otto YESA 3133 haciendo uso de una tarjeta NI-USB 6009 y el software Labview. Posteriormente se extrajeron las características de cada una de las señales del sensor MAP que correspondían a 720° de giro del cigüeñal, de las cuales se extrajeron las más importantes que indicaron los análisis ANOVA, análisis de correlación y random forest. Las características elegidas, fueron entrenadas mediante algoritmos de redes neuronales artificiales (RNA) y máquinas de soporte vectorial (SVM), realizando varias modificaciones para optimizar sus parámetros; adicionalmente se hizo uso del toolbox "Classification Learner" de Matlab para analizar varios modelos de Machine Learning adicionales, destinados a la clasificación de datos. Dicho proceso se realizó para los datos del MAP en bruto y para los datos luego de pasar por un filtro Butterworth paso bajo. Los resultados de la investigación indicaron que la RNA es la más adecuada para utilizarse debido a que tiene una mejor precisión, la cual, luego de varias configuraciones de entrenamiento alcanzó una precisión del 96.05% sin mostrar síntomas de sobreajuste, además que tuvo un menor tiempo de entrenamiento con relación a los demás modelos de Machine Learning. Adicionalmente, la investigación demostró que cuando se tienen más características para su entrenamiento, la precisión de la RNA aumenta pasando de un 93.97% a un 95.53%.

Palabras clave: RNA, Machine Learning, MEP, SVM, sensor MAP y Classification Learner.

Abstract

In the present work, Machine Learning algorithms and models were applied within MATLAB to classify four different engine operating states: optimal conditions, faulty spark plug, obstructed air filter, and both faults combined. To achieve this, voltage data from the MAP sensor of a YESA 3133 Otto engine was collected using an NI-USB 6009 data acquisition board and LabVIEW software. Then, features were extracted from each MAP sensor signal corresponding to 720° of crankshaft rotation, the most relevant ones were selected based on ANOVA analysis, correlation analysis, and Random Forest. The selected features were trained using Artificial Neural Networks (ANN) and Support Vector Machines (SVM), with multiple configurations to optimize their parameters. Additionally, MATLAB's "Classification Learner" toolbox was used to implement various Machine Learning models for data classification. This process was applied for both the raw MAP sensor data and the MAP sensor data processed through a low-pass Butterworth filter. The work results indicated that ANN is the most suitable model for used because it has the best accuracy. After several configurations, it reached an accuracy of 96.05% without showing signs of overfitting, and also had the shortest training time compared to the other Machine Learning models. Additionally, the study demonstrated that increasing the number of training features improved ANN accuracy, raising it from 93.97% to 95.53%.

Keywords: ANN, Machine Learning, SI Engine, SVM, MAP sensor, Classification Learner.

3 Introducción

Actualmente, el parque automotor en el Ecuador sigue aumentando, en cifras generales, el INEC (2024) indica un crecimiento del 6.42% entre los años 2022 – 2023. Para el año 2022, de los 3,065 millones de vehículos matriculados, 1,816 millones correspondían a vehículos livianos y de ellos, el 90% utilizaban como combustible la gasolina (INEC, 2024). Dicha cifra ha aumentado, puesto que los vehículos livianos han aumentado hasta 2,624 millones para finales de mayo del 2024 (AEADE, 2024).

Adicionalmente, los vehículos deben encontrarse funcionando en óptimas condiciones para mantener los niveles de contaminación de ruido y emisiones al margen (Velepucha y Sabando, 2021), lo cual conlleva a aplicar los distintos tipos de mantenimientos al vehículo, esto ligado al crecimiento de los vehículos livianos a gasolina indica que se deben implementar mejoras en los servicios de mantenimiento, puesto que actualmente, existen falencias como la falta de herramientas actualizadas para el mantenimiento, subjetividad en el diagnóstico, elevados tiempos de servicio, ineficiencias en la localización de la falla y costos elevados, provocando que la calidad del servicio se perciba de mala calidad (Bimboza et al., 2023; Contreras et al., 2019).

Para el diagnóstico de fallas de sistemas automotrices, generalmente se utiliza un sistema de diagnóstico a bordo OBD el cual se encarga de detectar las fallas ocasionadas en el vehículo; sin embargo Castillo et al. (2021) mencionan que existen algunas fallas producidas en el sistema de admisión de aire y el sistema de encendido del vehículo que no son reconocidas por el sistema OBD.

Por este motivo para suplir las falencias del OBD se busca complementar los procesos de diagnóstico mediante la implementación de Machine Learning, por lo que actualmente se están aplicando las redes neuronales artificiales (RNA) y máquinas de soporte vectorial (SVM) para la detección de fallas en motores Otto (Contreras U. et al., 2019; Galindo et al., 2020; Li et al., 2023). De manera que logren clasificar distintas fallas de manera más precisa (Galindo et al., 2020).

De manera general, según Tuan et al. (2021) las RNA han recibido excelentes reseñas dentro de los campos de la ingeniería y han presentado grandes beneficios para resolver problemas no lineales en motores de combustión interna, por lo cual, las líneas de investigación actuales se enfocan en comparar distintos algoritmos en conjunto para determinar cuál es el

más eficiente, por ejemplo F. A. García et al. (2023) compararon distintos algoritmos de aprendizaje: XGBoost, SVMs, Random Forest, Regresión Logística y una combinación de todos ellos, consiguiendo que el mejor fue el algoritmo combinado, mientras que de manera individual el XGBoost y el SVMs fueron los mejores; Li et al. (2023) por su parte obtuvieron que el combinar los algoritmos de KNN, Random Forest y XGBoost resultó en una mejor eficiencia. En la investigación de Galindo et al. (2020) se utilizó como un código basado en RNA, SVM y RBF, para generar reglas de clasificación para un sistema difuso, en donde los algoritmos de SVM lineal y RBF fueron los más rápidos y eficientes.

Puesto que un algoritmo de clasificación necesita señales de entrada para su entrenamiento, se pueden extraer datos de varios sensores mediante el uso de una tarjeta DAQ NI-USB 6009 (Contreras U. et al., 2019). A partir de los sensores que utiliza un vehículo, los datos del sensor MAP sirven de utilidad para el diagnóstico de fallos de una bujía, como lo presenta la investigación de Yixin et al. (2022) cuando emplean CNN, un tipo de RNA para analizar los datos mediante el uso de la frecuencia de los datos del sensor MAP. Shahbaz y Amin (2021) utilizaron los datos del sensor MAP, el software Matlab y un algoritmo basado en una RNA para diseñar un sistema de control activo de tolerancia de fallas en el control de la mezcla aire-combustible de un motor Otto y mencionan que el uso de una RNA conlleva un menor costo computacional al trabajar con los datos del sensor MAP.

La investigación presenta beneficios para profesionales dentro del área del mantenimiento automotriz y los usuarios de un vehículo, puesto que se comparan dos algoritmos de Machine Learning: RNA y SVM en el diagnóstico de fallas al trabajar con los datos del sensor MAP, de esta manera se contribuye al proceso de implementación del Machine Learning en el mantenimiento automotriz. Dentro de la Universidad Nacional de Loja, la investigación aporta material de enseñanza para las materias de teoría del control, motores de encendido provocado y sistemas de inyección puesto que permite estudiar la metodología empleada y de igual manera propone futuras investigaciones acerca del uso de algoritmos de Machine Learning aplicados en la industria automotriz. Es por ello que la investigación cuenta con un objetivo general y tres objetivos específicos.

3.1 Objetivo General

Emplear algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto.

3.2 Objetivos Específicos

- Realizar la adquisición de datos mediante una técnica mínimamente invasiva en un motor Otto en diferentes condiciones de funcionamiento.
- Analizar los datos obtenidos mediante el empleo de algoritmos de Machine Learning para clasificar los diferentes tipos de fallos inducidos.
- Verificar los resultados obtenidos mediante el análisis estadístico de los algoritmos implementados identificando aquel que presente un menor margen de error.

El alcance de la presente investigación se basa en comparar dos métodos de Machine Learning destinados a la clasificación de fallos en un motor Otto, dichos métodos son Redes Neuronales Artificiales (RNA) y Máquinas de Soporte Vectorial (SVM) dentro del entorno de Matlab. Para ello se hizo uso de los datos de un sensor MAP obtenidos mediante una tarjeta DAQ 6009 en distintos comportamientos del motor: un funcionamiento normal, un funcionamiento cuando la distancia de separación de la bujía del cilindro número 1 es de 0 mm, un funcionamiento del motor con una obstrucción del 50% del conducto de admisión de aire y un funcionamiento cuando se simulan ambas fallas a la vez, la unidad experimental fue un banco de motor YESA 3133.

El trabajo está estructurado en un marco teórico, una metodología, resultados, una discusión, conclusiones y recomendaciones. En el marco teórico se explica el funcionamiento de las RNA y SMV, en la metodología se explica el proceso de la adquisición de datos del sensor MAP y el entrenamiento de los algoritmos, en los resultados se compara la eficiencia de ambos algoritmos, los resultados de rendimiento de Matlab con gráficas de cada algoritmo, variables de desempeño y estadística; en la discusión se comparan los resultados con estudios previos y finalmente se elaboran las conclusiones y recomendaciones a partir de los resultados obtenidos.

4 Marco Teórico

Para una mejor comprensión de la investigación a continuación se definen brevemente algunos conceptos que se utilizarán en su desarrollo.

4.1 Motor Otto

También conocido como “motor de encendido por chispa” o “motor de encendido provocado” (MEP o SI-ICE por sus siglas en inglés) es un motor de 4 tiempos (admisión, compresión, explosión y escape) que se encuentra dentro de la clasificación de los motores de combustión interna alternativos (MCIA), se caracteriza por su proceso de ignición producido al agregar una mezcla de aire/combustible encendida mediante el salto de chispa provocado por una bujía, en donde la combustión se propaga mediante un frente de llama (Muñoz y Rovira, 2015). La unidad experimental o banco de pruebas en donde se desarrolló la investigación corresponde a un banco de un MEP fabricado por la empresa YESA (Figura 1) en donde se encuentra un motor Otto de 4 cilindros de la marca Hyundai y el mismo permite realizar distintas mediciones de señal sin invadir directamente en el motor.

Figura 1. Unidad experimental para investigación, banco fabricado por la empresa YESA.

4.2 Bujía

Es el elemento encargado de generar la chispa para inflamar la mezcla de aire/combustible para su buen funcionamiento se requiere que la bujía soporte tensiones de hasta 30 kV, y presiones generadas de hasta 30 bar (generadas en motores Otto de inyección (Bosch, 1996).

La distancia de separación de los electrodos de la bujía debe ser lo suficientemente grande para activar un gran volumen de la mezcla aire/combustible y provocar la combustión

en el cilindro, pero también debe ser lo suficientemente pequeña para que la misma continúe trabajando incluso en situaciones desfavorables como el final de la vida útil de la bujía (Bosch, 1996). La Figura 2 (a) muestra una distancia de separación de electrodos óptima de 1 mm (Delgado, 2018).

Una distancia de separación de los electrodos muy pequeña o nula (Delgado, 2018), se muestra en la Figura 2 (b), no permite conseguir un correcto funcionamiento a ralentí; por su parte, una distancia muy grande requiere de una elevada tensión de encendido, lo que está asociado a un desgaste en los electrodos de la bujía (Bosch, 1996) .

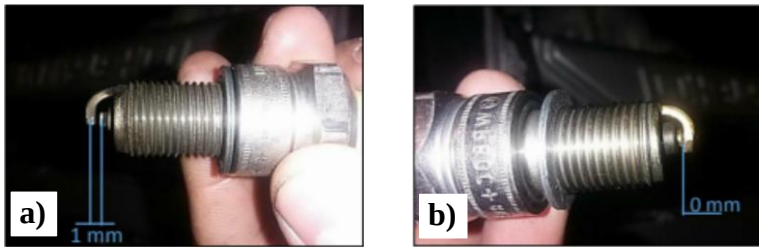


Figura 2. Distancia de separación de los electrodos de la bujía.

4.3 Sistema de admisión de aire

El sistema de admisión de aire o sistema del colector de admisión se encarga de dirigir el aire exterior hacia el interior del motor, mediante el uso de tuberías dirigidas a cada uno de los cilindros, sus componentes son el filtro de aire y el colector de admisión, este último consta de una mariposa de aceleración, EGR (si se implementa), un reservorio de aire (Plenum) los conductos de admisión o sobrealimentación y el puerto de admisión ubicado en la culata (Valdés, 2020); en algunos casos se suele utilizar una caja de filtros (Mena y Mena, 2023). En la Figura 3 se observan los componentes del sistema de admisión (Valdés, 2020).

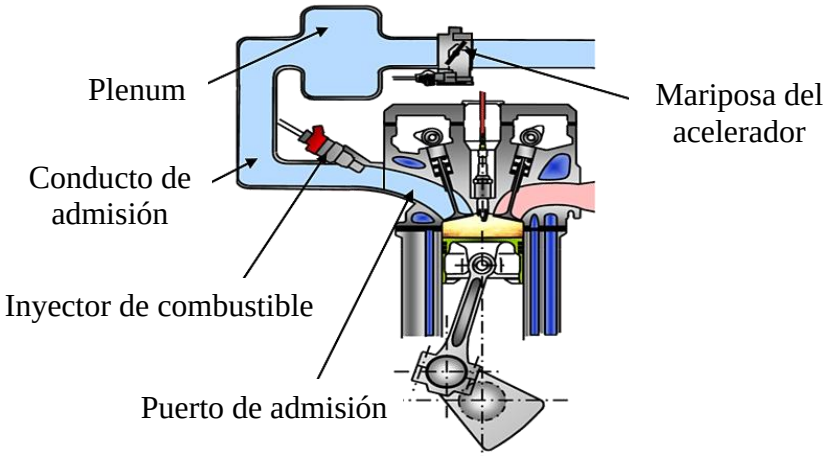


Figura 3. Sistema de admisión de aire.

La fase de admisión de un motor Otto funciona gracias al movimiento del pistón desde el punto muerto inferior (PMI) al PMS. El aire que ingresa al MEP debe ser limpio y con gran cantidad de oxígeno para obtener las mejores prestaciones en la combustión. Por ello, el filtro de aire constituye un papel importante en la retención del polvo y arena que puede ingresar a la cámara de combustión. Un filtro nuevo, brinda un suministro de aire en toda su capacidad, la misma que disminuye conforme su uso, según Bosch, en 1996, un contenido de 0.001 g/m^3 de polvo en el ambiente corresponde a que el motor consume hasta 50 g de polvo por cada 1000 km de recorrido.

De forma adicional, el sistema de admisión de aire incluye un conjunto de sensores y actuadores como el MAF, MAP o IAT con la finalidad de informar a la unidad de control electrónico (ECU) acerca del flujo de aire desde el exterior hacia el motor (Mena y Mena, 2023).

4.4 DAQ NI USB 6009

Es un dispositivo de adquisición de datos (DAQ) multifunción con interfaz de UBS de alta velocidad, empleado generalmente para la adquisición de datos de sensores. Cuenta con 8 entradas analógicas de un solo terminal (canales AI), 2 salidas analógicas (canales AO), 12 canales digitales de entrada y salida (DIO) con un contador de 32 bits. Permite adquirir datos analógicos con una resolución de 14 bits con un máximo de 48 kS/s (48 kHz) al trabajar con canales múltiples (Ver Figura 4) (National Instruments, 2015).

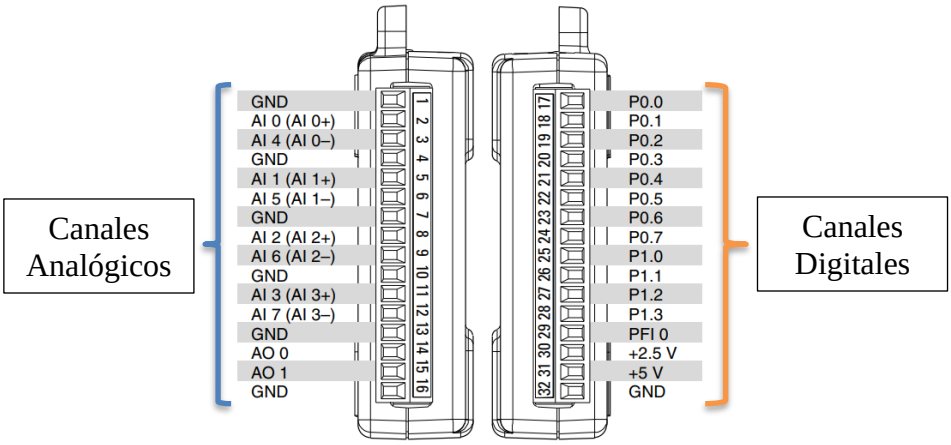


Figura 4. Canales analógicos y digitales del dispositivo DAQ NI 6009.

4.5 LabVIEW

Es un software desarrollado por la National Instruments (Ver Figura 5), el cual puede conectarse a cualquier instrumento para monitoreo y control de pruebas con varias funciones

de ingeniería y con un funcionamiento con lenguajes de programación (National Instruments, 2025), fue empleado por varias investigaciones como las de (Contreras U. et al., 2019; Yu et al., 2023) para la adquisición y registro de datos de sensores mediante el uso de una DAQ NI 6009.

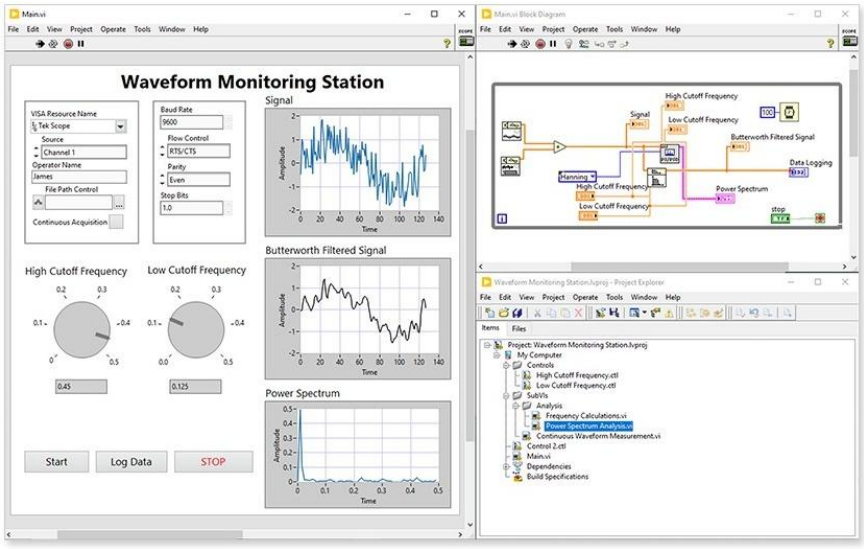


Figura 5. Entorno de LabVIEW.

4.6 Sensor MAP

El sensor MAP es el encargado de proporcionar una señal de tensión que indica la variación de la presión del aire en el colector de admisión de una manera proporcional, la presión varía dependiendo de la carga y las revoluciones por minuto (RPM) del motor, refleja el cambio de la posición de la mariposa de aceleración puesto que a mayor apertura de la mariposa se tiene una mayor presión en el colector, la señal del sensor MAP junto con el sensor de temperatura y de posición angular de la mariposa son utilizados para calcular el avance del encendido (E. Sánchez, 2008). Posee 4 pines cuando incorpora el sensor de temperatura y el sensor más empleado es el de tipo piezoeléctrico. Un sensor MAP de 4 pines generalmente tiene una tensión de salida de entre 1.3 a 1.9 V en ralentí y de 4 a 4.5 V a plena carga tal como se muestra en la Figura 6 la señal medida con el osciloscopio (E. Sánchez, 2008).

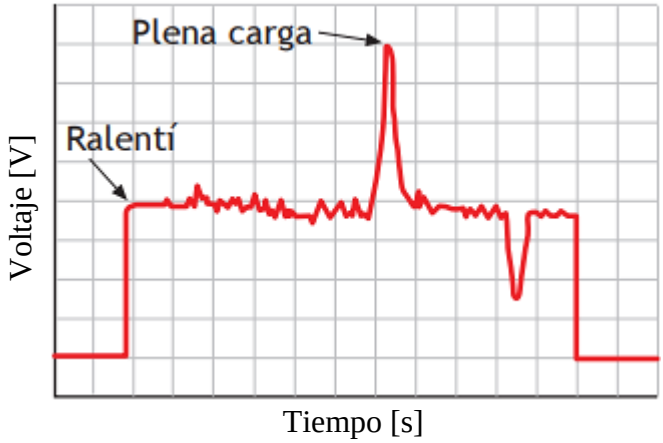


Figura 6. Voltaje de salida un sensor MAP.

4.7 Diseño factorial

Según H. Gutiérrez & Salazar (2008) forma parte del diseño de experimentos, es un método empleado para estudiar cómo influyen varios factores sobre un resultado, está compuesto por factores, niveles y repeticiones. Los factores son las variables que se estudian para ver si influyen o no a la respuesta y pueden ser cualitativos o cuantitativos, los niveles son los distintos estados que puede tener cada factor y las réplicas es el número de veces que se repetirá el experimento, su objetivo es determinar el número de combinaciones necesarias para un buen desempeño del proceso.

Se diseña de la siguiente manera: si tenemos dos factores cada uno con dos niveles, el número de combinaciones es $2x2 = 2^2$ que es igual a 4, si tenemos 3 factores, cada uno con 2 niveles, el diseño es $2x2x2 = 2^3$, si se tienen dos factores uno con 3 niveles y el otro con 2 niveles, el diseño corresponde a $3x2$, si tenemos 3 factores en donde uno tiene 4 niveles y los otros dos tienen 2 niveles, el diseño factorial es: $4x2x2 = 4x2^2$ (H. Gutiérrez y Salazar, 2008).

Según H. Gutiérrez y Salazar (2008) dentro de un diseño de experimentos encontramos las siguientes variables y factores.

- **Variables de respuesta:** Permiten observar los efectos de las pruebas realizadas.
- **Factores controlables:** Son variables o características que condicionan a un proceso y se les puede atribuir un nivel determinando su estado o magnitud de manera controlada.
- **Factores no controlables:** También llamados factores de ruido, son parámetros y métodos que no se pueden controlar al momento de la experimentación.

- **Factores de estudio:** Son aquellos parámetros que se van a investigar, con respecto a su influencia o no en las variables de respuesta.

4.8 Machine Learning

El Machine Learning o aprendizaje automático hace referencia al mejoramiento de un sistema a través del aprendizaje de una serie de datos mediante métodos computacionales, basa su principio en el comportamiento humano el cual aprende a través de la experiencia, de esta manera, se busca que los sistemas computacionales logren realizar predicciones y observaciones (Zhou, 2021). MathWorks (2024) considera la clasificación del Machine Learning tal como se muestra en la Figura 7.

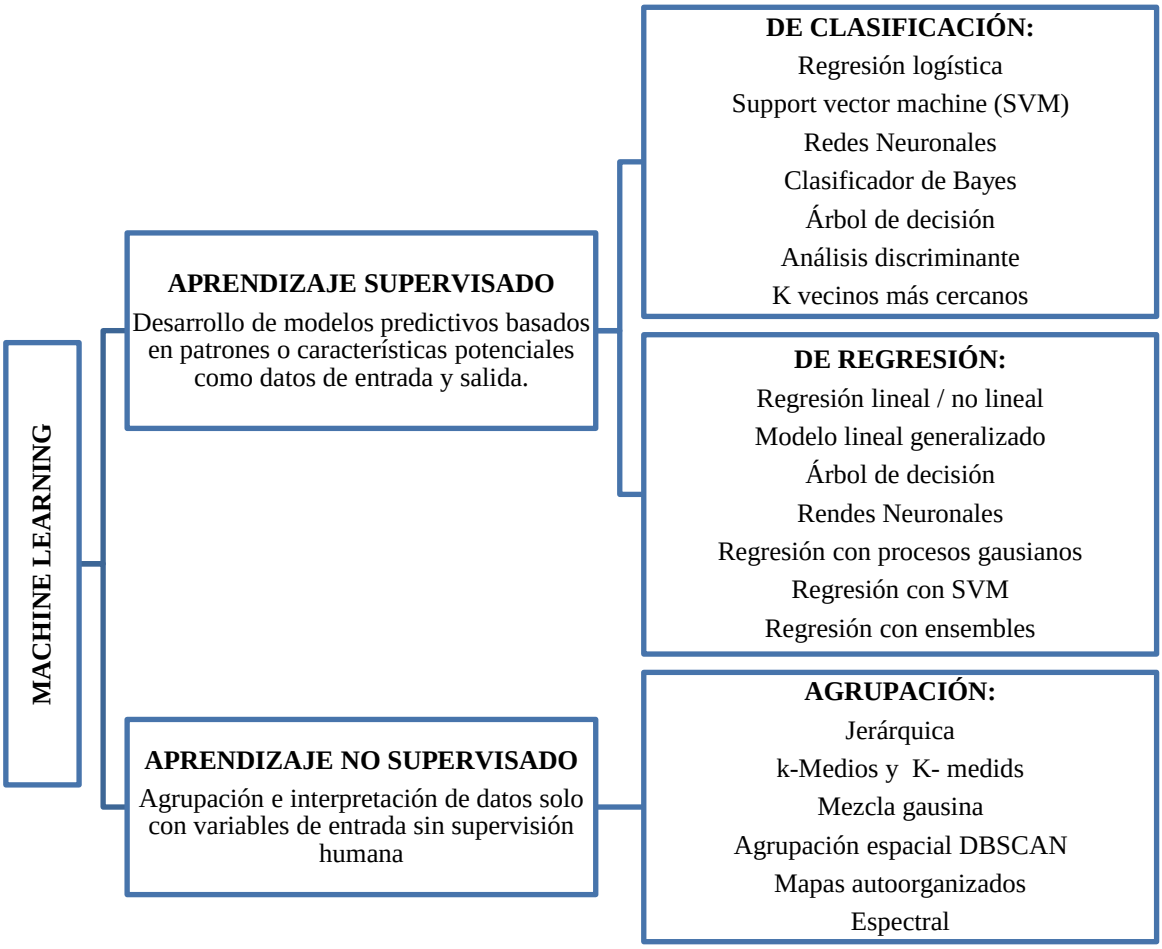


Figura 7. Clasificación de Machine Learning.

4.9 Redes neuronales artificiales (RNA)

Las redes neuronales artificiales (RNA o ANN por sus siglas en inglés) son una parte del Machine Learning inspiradas en modelar el funcionamiento del cerebro humano basándose en su estructura conformada por redes neuronales biológicas (Abeliuk y Gutiérrez, 2021). Para

ello, se representa una neurona biológica como un proceso simple e interconectado la cual opera de manera paralela (Guzmán, 2023).

4.9.1 Estructura de una red neuronal biológica

Una red neuronal biológica o sistema nervioso biológico funciona de la siguiente manera: las dendritas reciben los impulsos nerviosos, el soma mediante un procesamiento interno genera una activación o impulso, finalmente, se transmite como salida a lo largo del axón hacia la siguiente neurona (Ver Figura 8) (Guzmán, 2023).

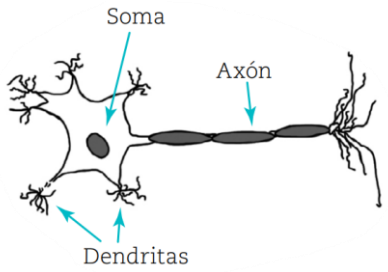


Figura 8. Estructura de una neurona biológica.

De manera análoga con una RNA, las dendritas corresponderían a la capa de entrada, el soma serían las capas ocultas y el axón la capa de salida.

4.9.2 Estructura de las redes neuronales artificiales

Las RNA generalmente constan de un conjunto de 3 capas: capa de entrada, capas ocultas y capa de salida. La capa de entrada permite el ingreso de todos los parámetros de entrada, las capas ocultas se encargan de calcular el peso de los datos para finalmente arrojar una salida mediante la capa de salida (Tuan et al., 2021). Dentro de las aplicaciones de RNA, la más utilizada corresponde a la red neuronal multicapa Backpropagation (Tello et al., 2021).

4.9.3 Perceptrón multicapa

El perceptrón es una neurona que puede aprender mediante un peso o ponderación asignado a cada dato de entrada y fue propuesto en 1958 por Frnak Rosenblatt (Abeliuk y Gutiérrez, 2021) además, es la unidad básica de una RNA (Guzmán, 2023). El conjunto de perceptrones totalmente conectados forma un perceptrón multicapa, en donde, que todas las salidas de una capa, llegan a cada nodo de la siguiente capa, además, es una red feedforward en donde la salida de cada neurona de una capa, es la entrada de una neurona de la siguiente capa (R. García, 2022). En la Figura 9 se observa su topología (Marín y Arriojas, 2021).

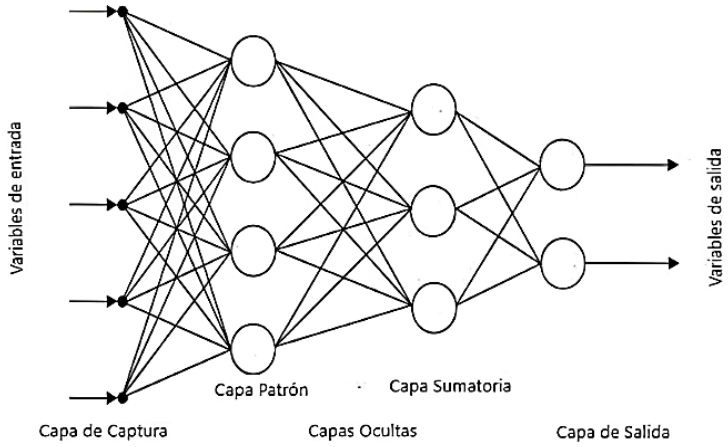


Figura 9. Disposición de las capas de un perceptrón multicapa.

Para entender su funcionamiento se debe comprender el funcionamiento de un perceptrón simple, el cual, se basa en una combinación lineal de un número de entradas $(x_1, x_2, x_3 \dots x_n)$ multiplicadas por un peso asignado a cada variable $(w_{1j}, w_{2j}, w_{3j} \dots w_{nj})$, (R. García, 2022), este peso es un valor aleatorio de entre 0 y 1 que se asigna aleatoriamente para iniciar el aprendizaje de la red neuronal (Raschka y Mirjalili, 2017). En la Figura 10 se puede observar el modelo de un perceptrón simple, en donde los valores de x corresponden a las variables de entrada, w corresponde a los pesos de cada variable, U_j indica la neurona artificial en donde mediante una sumatoria se procesará una variable v_j (Ecuación 1) la cual al pasar por medio de una función de decisión $f(v_j)$ genera una salida a_j de valor binario 0 o 1 (R. García, 2022).

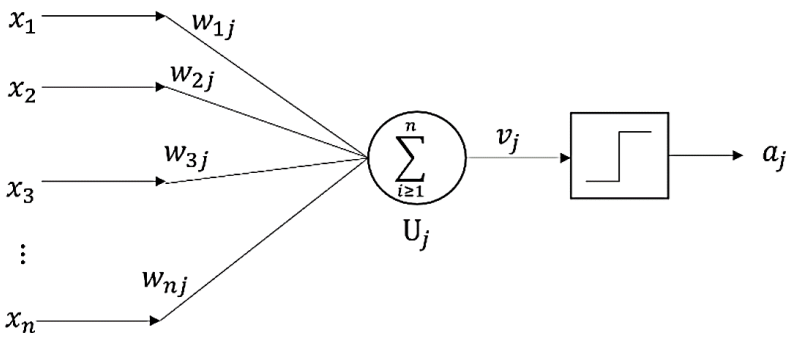


Figura 10. Variables de procesamiento de una neurona de un perceptrón simple.

El valor de la función v_j se define en la Ecuación 1 (Raschka y Mirjalili, 2017):

$$v_j = w_{1j}x_1 + w_{2j}x_2 + \cdots + w_{nj}x_n = \sum_{i=1}^{i=n} w_{ij}x_i = w^T x \qquad (1)$$

El perceptrón multicapa es superior al perceptrón monocapa, puesto que permite separar linealmente varias variables cuando se las agrupa entre sí, por ejemplo, si tenemos las clases A, B, C y D se pueden agrupar en las clases AB y CD, de manera que ambos grupos son linealmente separables entre sí, así como también se puede definir los grupos AD y BC que son linealmente separables, tal como se muestra en la Figura 11 (R. García, 2022).

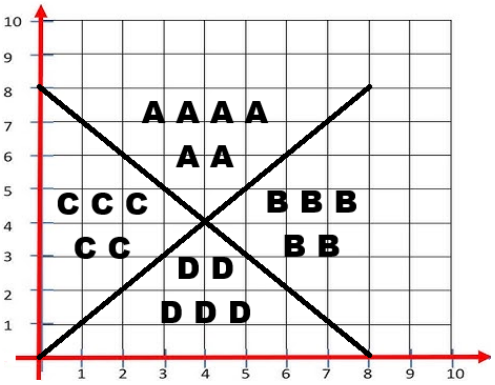


Figura 11. Clasificación lineal de los grupos AB, CD, AC y BD.

4.9.4 Aprendizaje de una red neuronal artificial

La RNA se caracteriza por ser un aprendizaje supervisado, en donde, la RNA buscará aprender una función de activación para clasificar variables a partir de un conjunto de datos (Umaña, 2021) la cual puede ser continua (lineal y sigmoidea), discreta (de tipo escalón) o híbrida (combina características de las otras funciones) (Guzmán, 2023). Dicho proceso está basado en 3 etapas: selección de los parámetros de entrada y salida, el tipo de entrenamiento y la prueba de la red (Tuan et al., 2021).

El entrenamiento de una red neuronal está basado en el ajuste del valor de sus pesos w_j que se realiza al comparar el valor de salida de la red neuronal a_j con el valor deseado y, en donde el nuevo vector de pesos se define mediane $w' = w + \alpha(y - a)x$ con $0 < \alpha < 1$, la variable α hace referencia al factor de aprendizaje, de esta manera se puede medir la velocidad que la red neuronal aprende (R. García, 2022).

De esta manera para el aprendizaje la red neuronal sigue el siguiente proceso:

1. Se define un peso w_{nj} para cada neurona.
2. Se toma un dato de entrada y un valor de salida esperado $[x, y]$.
3. Se evalúa el dato evaluado por la red neuronal a_j para la entrada x asignada.

4. Cuando $a_j \neq y$ entonces se debe aplicar el ajuste de los pesos según la fórmula de w' e ir al punto 5. Caso contrario se pasa al punto 2.
5. Cambiar el valor de w a w' y volver al punto 2.

Cuando el valor de w no cambie, se dice que el entrenamiento de la RNA ha culminado.

En un perceptrón multicapa para variar los pesos en cada iteración, se emplea el método del gradiente descendente, el cual, emplea funciones de activación diferenciables, en donde, la más utilizada generalmente corresponde a la función sigmoidea, también existen otros métodos como la regla delta y el backpropagation (R. García, 2022).

4.10 Máquinas de Soporte Vectorial (SVM)

Según Abdullah y Abdulazeez (2021), una máquina de soporte vectorial (support vector machine SVM) es un algoritmo de aprendizaje ampliamente utilizado para resolver problemas de clasificación. Su principio se basa en maximizar el margen de decisión m entre dos puntos que distancian dos clases (vectores de soporte) (Ver Figura 12) (Delgado, 2018).

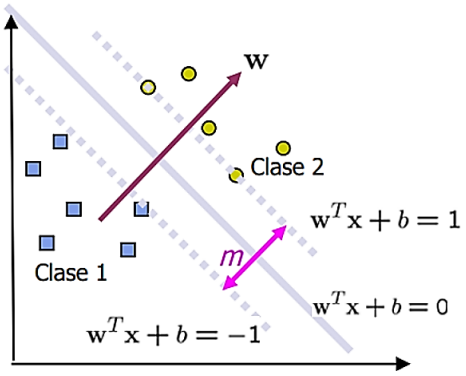


Figura 12. Margen de separación de una SVM

El problema que intenta resolver un algoritmo de SVM, se basa en determinar un hiperplano que separe los vectores de manera que esté lo más alejado de ellos y que, además, la suma de sus errores de clasificación sea minimizada (Valero-Carreras et al., 2023).

Los problemas de clasificación pueden corresponder a casos en donde los datos sean lineal o no linealmente separables (Figura 13 (a) y (b) respectivamente). Sin embargo, es común encontrar casos de variables no separables linealmente, es por ellos que se hace uso de dos métodos: Kernel o el método “Soft Margin Classification” (Delgado, 2018).

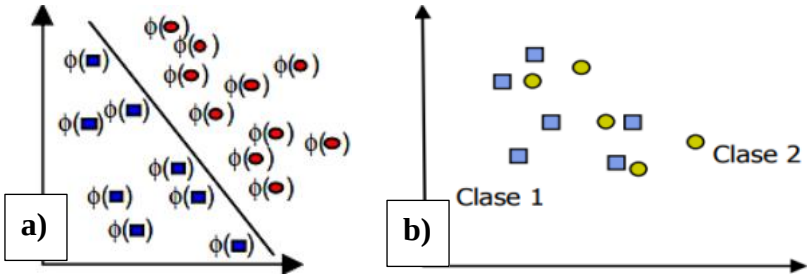


Figura 13. Diferencia de variables lineal y no linealmente separables.

4.10.1 Máquina de Soporte Vectorial con Soft margin Classification

Empleado como un método matemático para clasificar datos no linealmente separables, se hace uso de una constante C (parámetro C) para determinar la posición del hiperplano que clasifica las variables. El parámetro C representa el costo de clasificación errónea, puesto que, si su valor es muy grande da lugar a grandes penalizaciones por error, por lo que la distancia del margen es muy corta, volviéndola estricta a parámetros alejados de una clase (Ver Figura 14 (a)); mientras que, si el valor es muy pequeño el sistema es menos estricto frente a la clasificación errónea, en donde la distancia del margen aumenta (Ver Figura 14 (b)); por ende, el parámetro C es muy importante puesto que permite controlar el margen del hiperplano (Raschka y Mirjalili, 2017).

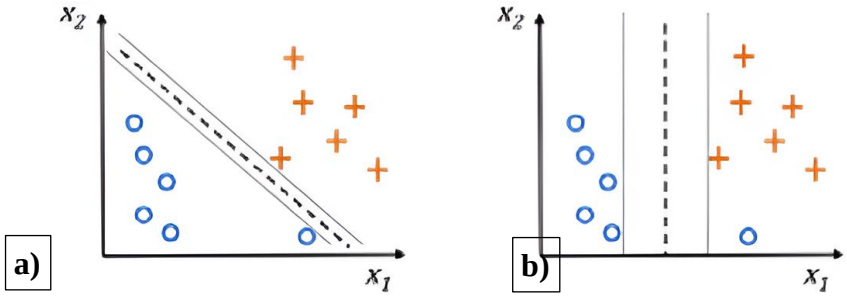


Figura 14. Comparación entre un valor muy grande y muy pequeño de C.

4.10.2 Máquina de Soporte Vectorial con función Kernel

Según Raschka y Mirjalili (2017) su principio se basa en crear combinaciones a partir de los vectores iniciales para luego, ser proyectados en una mayor dimensión. Por ejemplo, los datos escritos de manera bidimensional se pueden representar de manera tridimensional mediante la Ecuación 2.

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2) = (z_1, z_2, z_3) \quad (2)$$

En donde x_1 y x_2 son los datos iniciales de un vector de 2 dimensiones, y, los valores de z_n corresponden a las nuevas coordenadas que tendrá el vector inicial.

4.11 Matlab en Machine Learning

Matlab es un software que permite analizar, simular, modelar y graficar datos mediante cálculos y programación, utiliza un gran número de herramientas e incluye características para ajustar modelos de Machine Learning. Con las apps *Classification Learner* y *Regression Learner* se pueden explorar y seleccionar datos para entrenar un algoritmo de Machine Learning, compararlos y evaluarlos (MathWorks, 2024).

4.12 Matriz de confusión para la evaluación del desempeño

La matriz de confusión es una herramienta que sirve para evaluar el desempeño de un algoritmo de Machine Learning, permitiendo observar los aciertos y errores que comete al momento de su entrenamiento (Li et al., 2023). Por ejemplo, dentro de una clasificación binaria con las variables de entrada "Positivo" y "Negativo", el algoritmo las puede clasificar 4 maneras diferentes: verdadero positivo (VP), falso positivo (FP), falso negativo (FN) y verdadero negativo (VN), tal como se observa en la Tabla 1.

Tabla 1. Matriz de confusión.

Valores reales de entrada		*Positivo*	*Negativo*
Valores obtenidos por el algoritmo	**Positivo**	VP	FP
	Negativo	FN	VN

Nota. VP indica que el valor real era positivo y el algoritmo lo clasificó como positivo, FP la entrada real era negativa pero el algoritmo la clasificó como positivo, FN el valor real era positivo, pero se clasificó como negativo; VN el valor real era negativo y se clasificó como negativo. Adaptado de (Li et al., 2023).

A partir de la matriz de confusión nacen los términos de precisión, sensibilidad (también llamado "Recall") y la puntuación F1 (F1-Score), los cuales se calculan a partir de las Ecuaciones 3, 4 y 5 respectivamente (Li et al., 2023). La Ecuación 3 es la precisión e indica el número de veces que el algoritmo clasificó de manera correcta a una entrada.

$$Precisión(P) = \frac{VP}{VP + FP} \tag{3}$$

La Ecuación 4 es la sensibilidad o también llamado "Recall", en este caso representa las veces que clasificó de manera correcta todos los valores positivos de entrada.

$$Sensibilidad(P) = \frac{VP}{VP + FN} \quad (4)$$

La puntuación F1 o *F1-Score* es una métrica que combina tanto la precisión como la sensibilidad y se calcula mediante la Ecuación 5.

$$F_1(P) = \frac{2}{\frac{1}{Precisión(P)} + \frac{1}{Sensibilidad(P)}} \quad (5)$$

4.13 Software estadístico

4.13.1 Minitab

Minitab es un paquete de software estadístico el cual tiene un gran uso en la actualidad dentro de estadísticas industriales el cual de entre varias herramientas nos permite analizar la variabilidad mediante la estadística descriptiva cuando se trabajan con datos no constantes, pero que presentan un grado de variabilidad (Kenett et al., 2021).

4.14 Métodos de análisis estadísticos

4.14.1 Análisis de varianza (ANOVA)

El análisis de varianza permite cuantificar el nivel de variación entre los resultados de un estudio, el cual resulta ser mucho más efectivo que otros análisis estadísticos, determina si una variable es repetible o no dependiendo de la cercanía de los datos de una misma población (J. Gutiérrez et al., 2023). Dagnino (2014) mencionan que el método ANOVA comprende varias técnicas de evaluación dependiendo del diseño experimental que se utilice, pudiendo emplearlo en 4 distintas situaciones: cuando se desea comparar de dos a más grupos, cuando se miden datos de manera repetida, cuando una variable puede variar de una a más formas y al aplicar dos métodos de manera simultánea observando el efecto de cada uno por separado. Kenett et al. (2021) mencionan que este método como análisis de datos es muy empleado, sin embargo, el análisis e interpretación de los datos se los deben realizar de manera clara. H. Gutiérrez y Salazar (2008) indican las variables producto de un análisis ANOVA, las cuales se resumen en la Tabla 2.

Tabla 2. Variables producto de un análisis ANOVA.

Fuente de variabilidad	Grados de libertad	Suma de cuadrados (SC)	Cuadrado Medio (CM)	Valor estadístico de prueba (F_0)	Valor-p
Tratamientos	$k-1$	$SC\ del\ tratamiento$	$CM\ del\ tratamiento$	$\frac{CM_{Tratamiento}}{CM_{Error}}$	$P(F < F_0)$
Error	$N-k$	$SC\ del\ error$	$CM\ del\ error$		
Total	$N-1$	$SC\ total$			

Nota. Tabla adaptada para relacionarla con los resultados arrojados por Minitab. La variable *k* corresponde al número de tratamientos, *N* corresponde al número de datos total de todos los tratamientos, *F* corresponde al valor de 0.05 del nivel de significancia.

4.14.2 Coeficientes de determinación R^2 y $R^2_{ajustado}$

Se consideran los estadísticos más útiles los cuales comparan la variabilidad frente a la variación total. Cuando su finalidad es la de predicción de características, se recomienda un valor de $R^2_{ajustado}$ de al menos el 70%, y, cuando existen muchas variables, se prefiere regirse a este último valor antes que al R^2 (H. Gutiérrez y Salazar, 2008).

5 Metodología

5.1 Área de estudio

El área de estudio en la que se llevó a cabo la presente investigación fue la Universidad Nacional de Loja, la cual está ubicada en la ciudad de Loja (Ver Figura 15) y correspondiente a la región interandina o sierra del territorio ecuatoriano. Geográficamente, el área de estudio se encuentra a 2139 msnm, por lo cual, los valores de presión atmosférica y el porcentaje de oxígeno en el aire son distintos y varían acorde al nivel del mar, estos valores condicionan el comportamiento de un motor Otto en gran medida (Passo et al., 2024). Al realizar la investigación en la región interandina, los datos de presión obtenidos por el sensor MAP son característicos de la región y la ciudad. Generalmente, las condiciones de temperatura dentro de la ciudad varían desde los 11°C a 21°C en marzo, el mes más cálido, y de 9°C a 17°C en el mes más frío del año, julio (Weather Spark, 2025).

Figura 15. Área de estudio: Universidad Nacional de Loja.

5.2 Procedimiento

La investigación se basa en un método analítico – inductivo, puesto que se comparan analíticamente los algoritmos de RNA y SVM para extraer una conclusión acerca de cuál de estos algoritmos de Machine Learning presenta un menor margen de error, el margen de error está supeditado a la precisión del mismo al clasificar e identificar una falla en diferentes

condiciones de funcionamiento del motor: tanto del sistema de encendido y el sistema de admisión de aire. Para el efecto, se entrenaron con los datos de voltaje del sensor MAP, de manera que también se indaga acerca de si es posible o no de que ambos algoritmos puedan clasificar una falla a partir de los datos de entrada, a los cuales se los denominaron "Tratamientos". El enfoque de la investigación es mixto, puesto que abarca a las variables cualitativas dentro de los diferentes tratamientos realizados en la unidad de estudio, y las variables cuantitativas al momento de comparar las RNA y SVM mediante análisis estadístico.

5.2.1 Flujograma del desarrollo de la investigación

En la Figura 16 se presenta un flujograma general de la metodología aplicada dentro de la presente investigación.

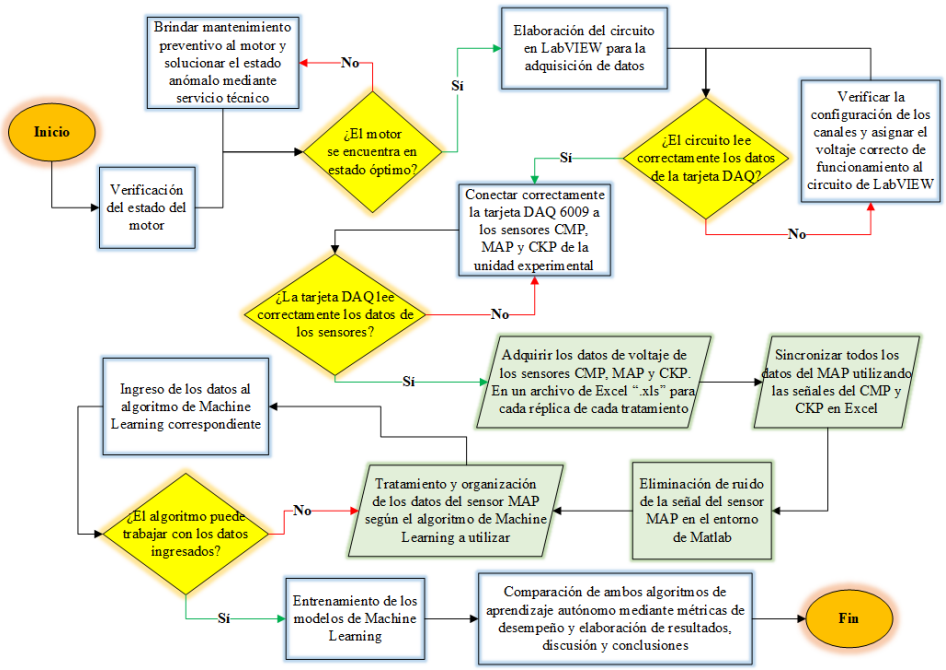


Figura 16. Flujograma de la metodología aplicada en la presente investigación.

Para la adquisición de datos mediante una técnica mínimamente invasiva, en un inicio, se tuvo previsto el uso de un dispositivo OBDLink Mx+ aplicando la metodología de Kumar y Jain (2023). Sin embargo, debido a la limitada frecuencia del software de adquisición de datos del OBDLink Mx+ de solamente 10 Hz y a la naturaleza de los datos del sensor MAP el cual posee picos de mayor frecuencia; la adquisición de datos se basó en la metodología empleada por Contreras U. et al. (2019). Quienes hicieron uso de un equipo DAQ NI 6009 para la adquisición de datos de los sensores CMP, MAP y CKP, y para su registro, un circuito en el software LabVIEW.

En la investigación se hizo uso de un diseño factorial realizado en el software Minitab siguiendo las recomendaciones que propone Delgado (2018), es por ello que se ha realizado un diseño 2x2 o 2^2 con dos factores y cada uno.

El tamaño de la muestra en la investigación corresponde al número de réplicas empleados en el diseño factorial, en la investigación se consideró el criterio de H. Gutiérrez y Salazar (2008) quienes mencionan que mayormente se utilizan de entre 5 a 10 réplicas, también se consideró lo mencionado por Li et al. (2023) en donde a mayor número de muestras, el entrenamiento de un algoritmo de clasificación mejora. Por ende, se utilizó un número de 10 réplicas para cada registrar datos de voltaje en diferentes condiciones de funcionamiento del motor a las cuales se las denominarán "Tratamientos". El muestreo empleado es de tipo sistemático puesto que el orden establecido para realizar cada tratamiento se obtuvo a partir de la creación del diseño de experimentos (DOE) realizado en el software Minitab.

Producto del diseño experimental, surgieron 4 estados del motor (tratamientos): el primero hace referencia al motor funcionando normalmente (bujía con calibración normal de 1 mm y flujo normal de aire); en el segundo, el motor presenta dos fallas (bujía a con calibración de 0 mm y presencia de una obstrucción en el paso de aire del 50%); en el tercer tratamiento, el motor funciona con una falla (obstrucción en el paso de aire del 50%); y finalmente, en el cuarto, el motor funciona con la otra falla (bujía a con calibración de 0 mm).

5.2.2 Unidad de estudio

Como unidad de estudio se eligió un banco de pruebas de un motor Otto de la empresa YESA modelo 3133, cuyas características se enumeran en la Tabla 3.

Tabla 3. Características de la unidad de estudio.

Vehículo base	**Hyundai Sonata EF**
Tipo de motor	Gasolina 2.0L DOCH (2002)
Cilindros	4 en línea
Transmisión	Transmisión automática
Potencia máxima (Vehículo base)	131 CV / 129 Hp / 96,4 kW @ 6000 rpm
Torque máximo (Vehículo base)	175 Nm @ 4600 rpm

Nota. Adaptado de (YES01, 2010).

Dicha unidad experimental se incluyó por el contexto del creciente parque automotor en el Ecuador, referente a los vehículos de gasolina, puesto que representan el 90% de los vehículos livianos matriculados para el 2024 (INEC, 2024) por lo cual, la investigación no comprende su estudio en un motor diésel.

La ubicación del sensor MAP de la unidad experimental se muestra en la Figura 17.

Figura 17. Ubicación del sensor MAP en la Unidad Experimental YESA 3133.

Para inducir los fallos para conseguir los 4 estados de funcionamiento del motor, se empleó una galga de espesores o “calibre fijo” para corroborar la calibración de la bujía a 1 mm cuando se requiere, también se utilizaron herramientas de mecánica para ejercer las fallas en el motor (Ver Anexo 1). En el Anexo 2 se observa la aplicación de los tratamientos al motor.

El registro de los datos de funcionamiento del motor a ralentí y a temperatura de régimen, se realizó gracias al circuito elaborado en el software Labview, el cual, permitió adquirir los datos de voltaje de los sensores CMP, MAP y CKP a una frecuencia de 10 kHz durante 5 segundos para cada réplica según la metodología empleada por (Contreras U. et al., 2019).

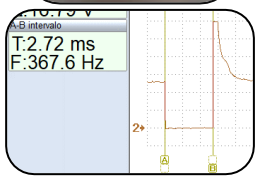
5.3 Procesamiento y análisis de datos

5.3.1 Adquisición de datos mediante una técnica mínimamente invasiva

5.3.1.1 Revisión del estado de la unidad Experimental. En primera instancia, se realizó la verificación del estado de funcionamiento de la unidad experimental YESA 3133.

La misma que es necesaria para el estudio, puesto que permite tener una base confiable del funcionamiento normal del motor, mejorando la calidad de los datos obtenidos y eliminando variables de ruido o señales erróneas que puedan afectar al desempeño de los algoritmos de Machine Learning. En la Tabla 4 se enlistan los componentes verificados, junto con su diagnóstico.

Tabla 4. Estado de los componentes del motor YESA.

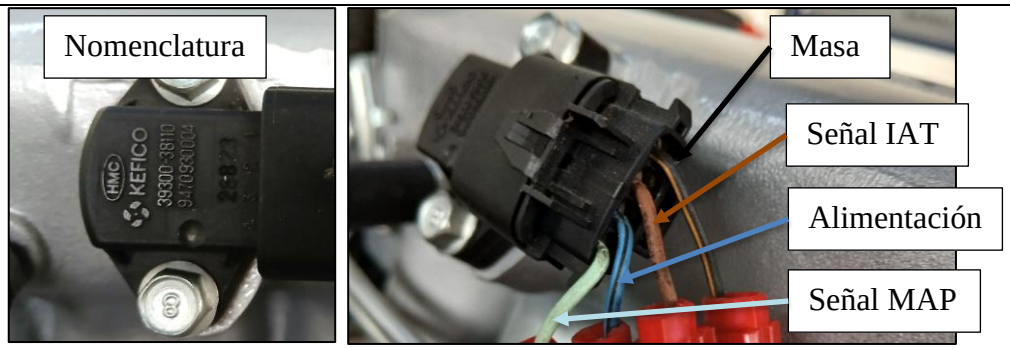
Componente o Parámetro	Valor obtenido	Datos técnicos de funcionamiento óptimo	Diagnóstico	Evidencias
Filtro de aire	(No aplica)	(Ninguna)	Buen estado	
Bujías	1 mm en todas las bujías	1 mm	Buen estado	
Presión de combustible	0.3 MPa = 43.51 psi	35 psi – 45 psi	Buen estado	
Tiempo de inyección en ralentí	C1: 2.68 ms C2: 2.72 ms C3: 2.64 ms C4: 2.64 ms	(Ninguna)	Buen estado	
Aceite del motor	(No aplica)	Nivel indicado y tez clara	Buen estado	

Componente o Parámetro	Valor obtenido	Datos técnicos de funcionamiento óptimo	Diagnóstico	Evidencias
Refrigerante del motor	(No aplica)	Nivel indicado y color apropiado	Buen estado	
Banda de accesorios	(No aplica)	Sin agrietamiento ni holgura excesiva	Buen estado	
Temperatura de funcionamiento	Electroventilador se enciende a los 97°C	90°C – 98°C	Buen estado	
Compresión de los cilindros	C1: 166 psi C2: 172 psi C3: 169 psi C4: 170 psi Diferencia entre cilindros: 6 psi	Diferencia máxima de compresión entre los cilindros: 10 psi	Buen estado	
Voltaje de carga de la batería	V vacío: 12.88 V V carga: 14.3 V	≥12.7 V	Buen estado	
DTC	Sistema OK	(Ninguna)	Buen estado	

5.3.1.2 Sensor MAP de la unidad experimental. El estudio comprendió el análisis de la señal de voltaje del sensor MAP.

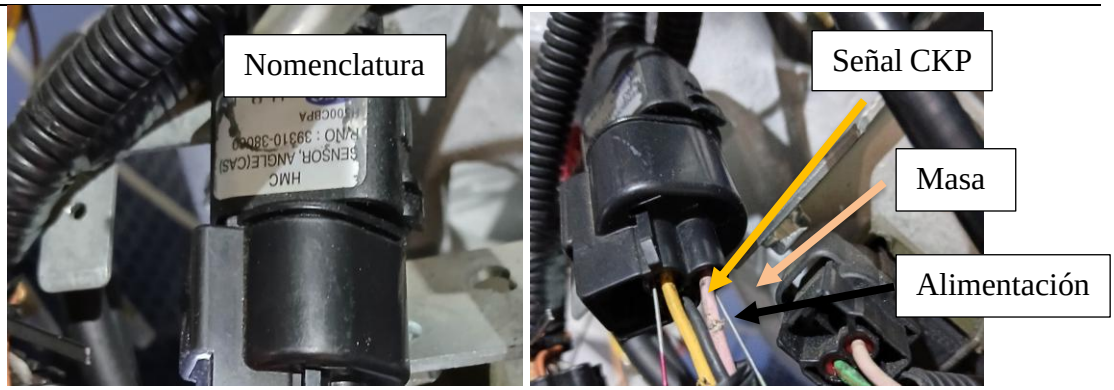
En la Tabla 5 se describen las características del sensor MAP empleado en la investigación.

Tabla 5. Características del sensor MAP de la unidad experimental.

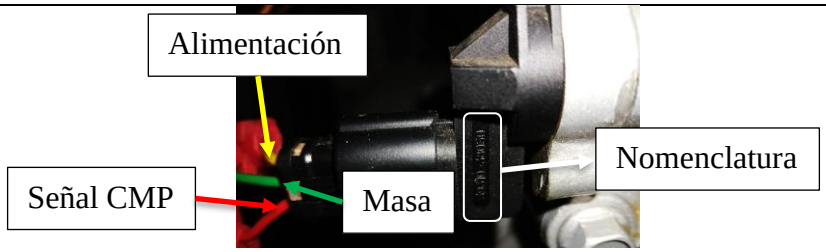
Sensor MAP	
	
Tipo de sensor MAP	MAP de 4 pines
Nomenclatura	39300-38110/9470930004
Pines	Alimentación: 12 V (Cable Azul con línea Negra) Señal del MAP: 5 V (Cable Turquesa) Señal IAT: 5V (Cable Café) Masa (Cable Negro con línea Café)

5.3.1.3 Sensor CMP y CKP de la unidad experimental. Las señales de los sensores CMP y CKP se utilizaron para poder realizar el tratamiento de los datos del sensor MAP. Sus características se describen en la Tabla 6.

Tabla 6. Pines de los sensores CKP y CMP.

Sensor CKP tipo Hall	
	
Nomenclatura	39310-38060
Pines	Alimentación: 12 V (Cable negro) Señal del CKP: 5 V (Cable Amarillo) Masa (Cable Rosado)

Sensor CMP tipo Hall



Nomenclatura	39310-38050
Pines	Alimentación: 12 V (Cable Amarillo) Señal del CMP: 5 V (Cable Verde) Masa (Cable Rojo)

5.3.1.4 Desarrollo del diseño Experimental. Previo a la adquisición de datos del sensor MAP, se realizó el diseño factorial 2^2 de los dos factores controlables dentro del estudio, cada uno con dos niveles.

El primer factor controlable corresponde a la bujía, dentro de este factor, en el primer nivel se consideró una separación de 1 mm alegando a un motor funcionando normalmente, dicho equivale a un promedio entre sistemas Otto convencionales y de alto rendimiento los cuales utilizan una calibración de bujías de 0.7 mm y 1.3 mm respectivamente (Aguirre y Campoverde, 2024) adicionalmente, la medida de 1 mm se evita tener una calibración muy grande con valores iguales o mayores a 1.13 mm (Carrión et al., 2022); el segundo nivel del primer factor, corresponde a una falla simulada en donde la separación de los electrodos estará completamente cerrada, simulando un estado de funcionamiento anómalo del motor considerando lo realizado en la investigación realizada por Contreras et al. (2019).

El segundo factor controlable corresponde a la simulación del estado del filtro de aire, en el primer nivel, el filtro de aire se considera limpio por lo cual permite una entrada normal de aire al motor, es decir, funciona correctamente; en el segundo nivel se simula un filtro sucio que se encuentra trabajando al 50% de su capacidad, para ello la investigación se basó en el trabajo realizado por Granda y Granda (2022) en el cual, mediante una empaquetadura obstruyen la entrada de aire en un 50% (Ver Anexo 3), sin embargo, en la presente investigación se diseñó una pieza de forma cónica, la cual genera un efecto similar al de la investigación anterior reduciendo el 50% el área de la entrada de aire en el conducto de admisión, la pieza se diseñó en el software Autodesk Inventor (Ver Anexo 4) y fue impresa en 3D (Ver Figura 18).

Figura 18. Cono de obstrucción del paso de aire impreso en 3D.

La ponderación para los factores y niveles del diseño experimental se muestra en la Tabla 7.

Tabla 7. Ponderación de los factores y sus niveles para el desarrollo del diseño experimental.

Factor	**Niveles**	**Ponderación**	
		Bajo (0)	**Alto (1)**
Bujía	2	Bujía con electrodos cerrados	Separación de electrodos de 1 mm
Filtro de aire	2	50% flujo de aire	100% flujo de aire

Una vez definidos los factores y niveles a ser empleados en la investigación, se utilizó el software Minitab para obtener las réplicas y el orden de los tratamientos los cuales serán aplicados al motor.

El proceso para el desarrollo del diseño factorial en Minitab se siguió el siguiente orden: Estadísticas → DOE → Factorial → Crear diseño factorial → (Selección de factorial de 2 niveles, 2 factores) → Factores → (Asignación del nombre y ponderación de los factores) → Diseño → (Selección de 10 réplicas) → Opciones → Aleatorizar corridas → Aceptar.

De esta manera se obtiene un orden aleatorio de los tratamientos puesto que se trata de una experimentación (H. Gutiérrez y Salazar, 2008), dicho proceso se representa en los pasos del 1 al 6 en la Figura 19.

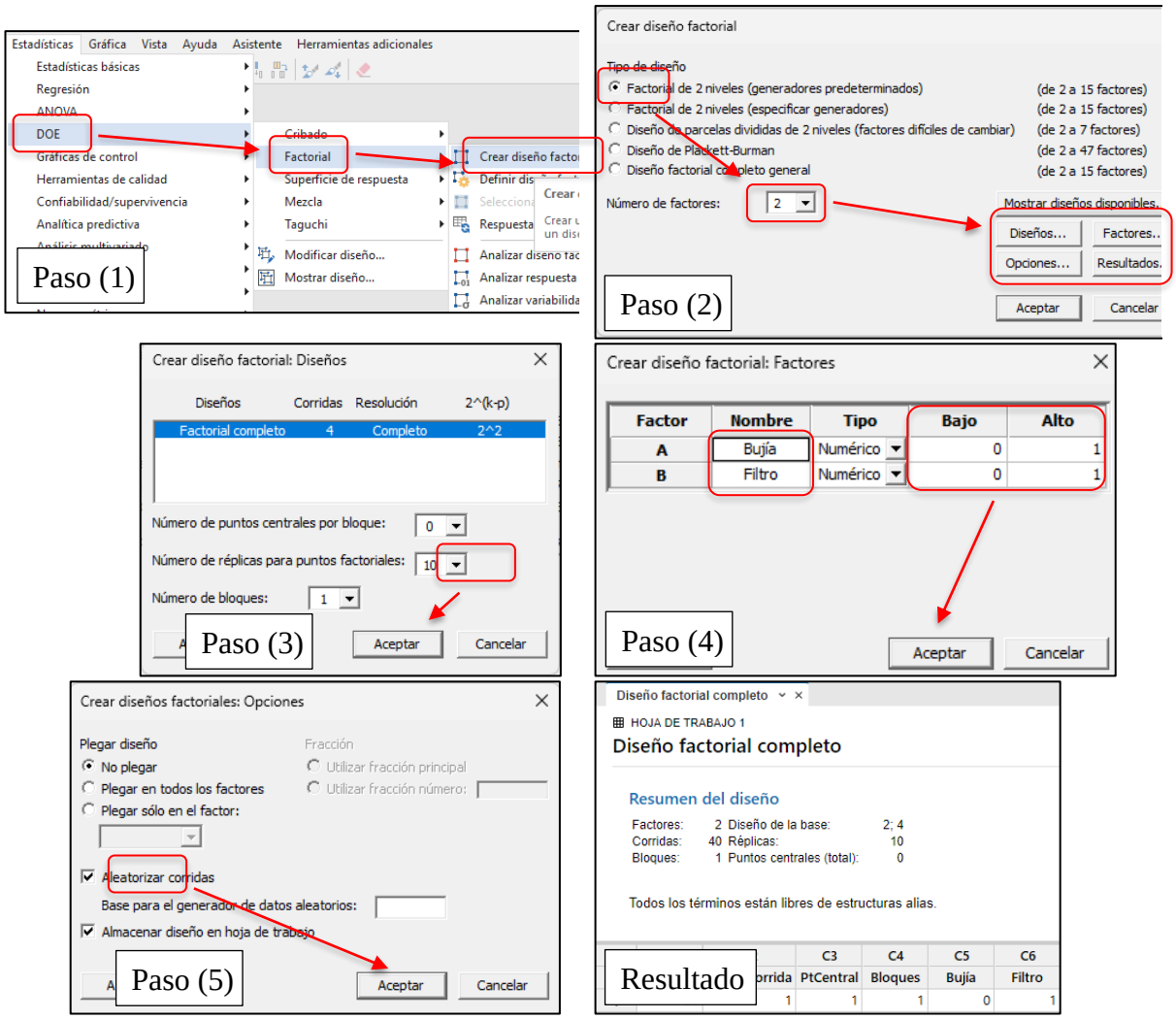


Figura 19. Proceso de desarrollo del diseño factorial en Minitab.

El diseño factorial al generar una sola réplica al diseño 2^2 da como resultado 4 tratamientos distintos los cuales se presentan en la Tabla 8.

Tabla 8. Factores, niveles y ponderación para el diseño factorial.

Tratamiento	Denominación	Descripción
1	Motor OK	Bujía del cilindro 1 con separación normal de 1 mm y sin obstrucción del paso de aire
2	Motor con ambas fallas	Bujía del cilindro 1 con separación anómala de 0 mm y con obstrucción del 50% del paso de aire
3	Motor simulando un filtro obstruido	Bujía del cilindro 1 con separación normal de 1 mm y con obstrucción del 50% del paso de aire
4	Motor simulando una bujía dañada	Bujía del cilindro 1 con separación anómala de 0 mm y sin obstrucción del paso de aire

El diseño factorial diseñado cuenta con 10 réplicas cuyo resultado se muestra en la tabla del Anexo 5, de esta forma, el diseño factorial brinda el orden para la adquisición de datos en los cuales se rige la presente investigación.

Los factores no controlables dentro del diseño experimental corresponden a las condiciones ambientales bajo las cuales se realizó la adquisición de datos. Las mismas, se registraron con el uso del sitio meteorológico (*wheater.com*), registrando los siguientes valores:

- Temperatura ambiente: 23°C
- Velocidad del viento: 13 km/h
- Humedad: 48%
- Condensación: 11°
- Presión atmosférica: 1009.1 mb

5.3.1.5 Configuración del circuito para la obtención de muestras. Luego de haber desarrollado el diseño experimental, se diseñó un circuito en LabVIEW, el cual permitió registrar y almacenar las magnitudes de voltaje obtenidos por la DAQ 6009 (Ver Figura 20).

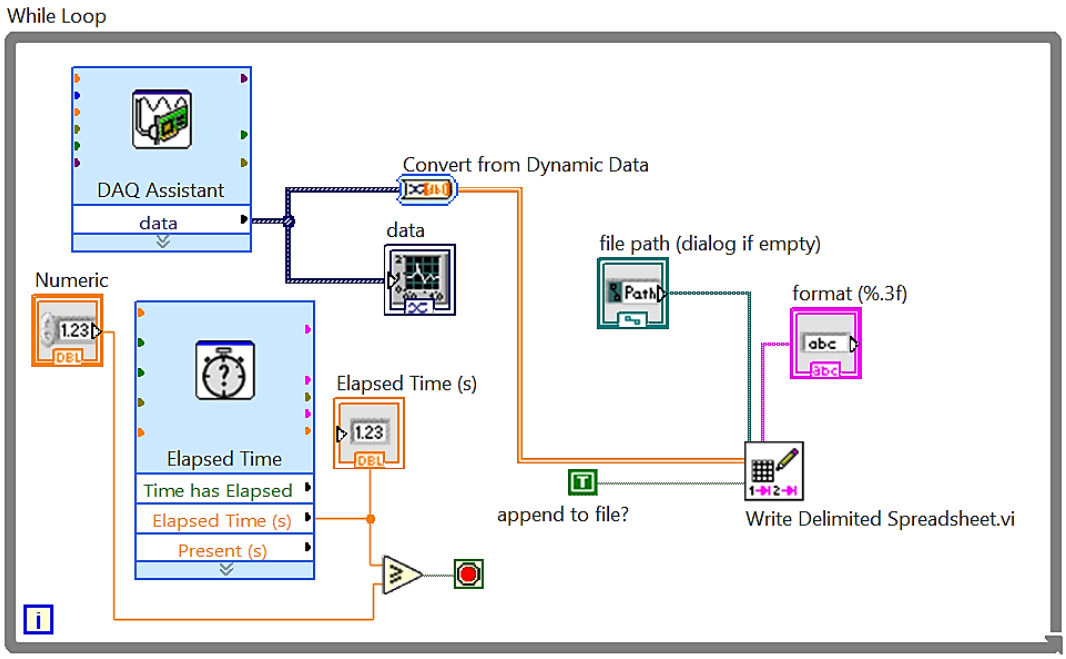
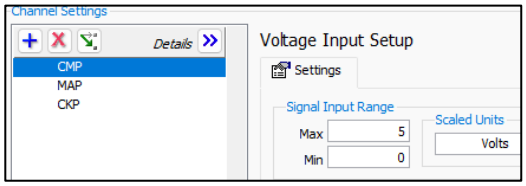
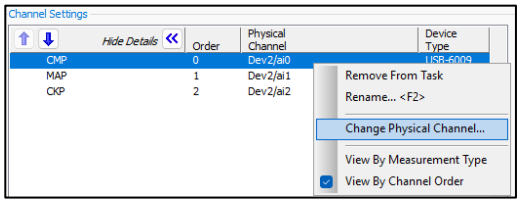
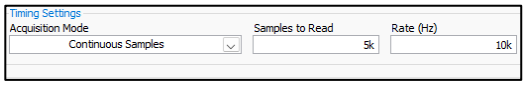
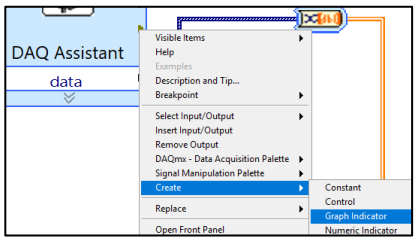
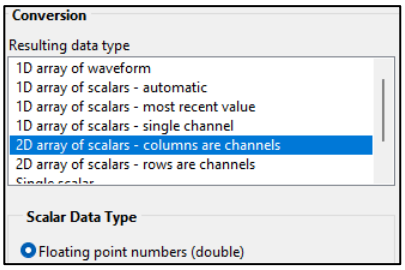
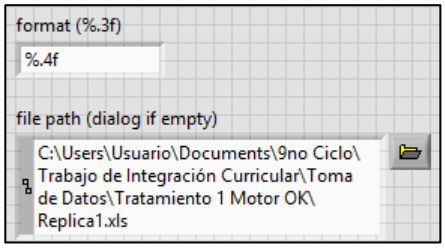
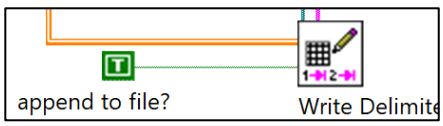
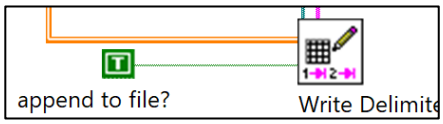
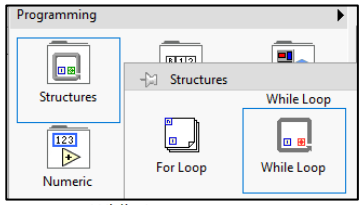
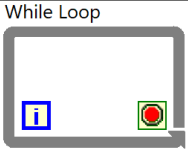
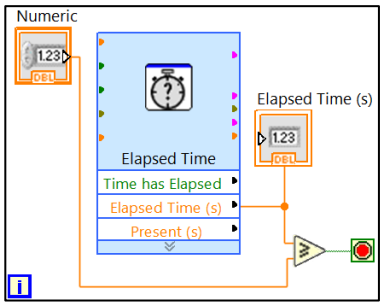
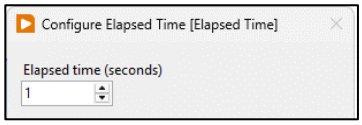
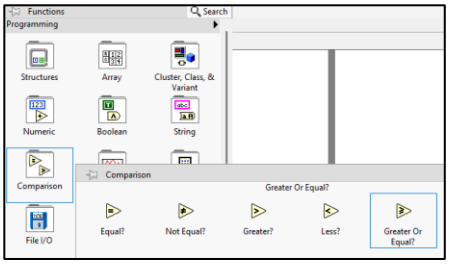
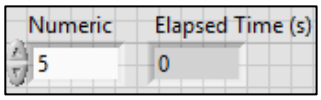


Figura 20. Circuito interno de LabVIEW para la adquisición y registro de datos a partir de la DAQ 6009.

El circuito consta de distintos elementos cuya configuración se resume en la Tabla 9.

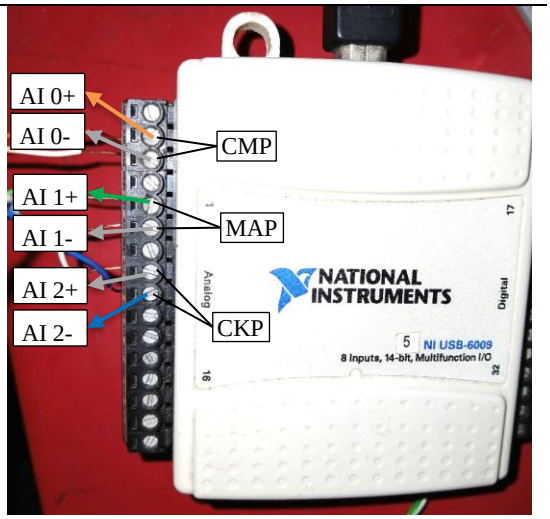
Tabla 9. Factores, niveles y ponderación para el diseño factorial.

Componente	Función	Configuración	Imagen
DAQ Assistnat	Leer los datos de la tarjeta DAQ 6009	Se asignó el voltaje de la señal de los sensores, en este caso para los tres sensores, el voltaje mínimo (Min) fue de 0 V y el máximo (Max) de 5 V. También se asignaron los canales de lectura, para el sensor CMP el canal de lectura seleccionado fue el (Dev2/ai0) haciendo referencia al nombre del dispositivo "Dev2" y a la entrada analógica 0 "ai0"; para el MAP (Dev2/ai1) y el CKP (Dev2/ai2). Se extrajeron 5000 muestras (Samples to Read) a una frecuencia (Rate) de 10 kHz. Se agregó un indicador gráfico "Graph Indicator" para observar la señal registrada.	   
Convert from Dynamic Data	Organiza los datos según se necesite	Se configuró con la opción "2D array of scalars – columns are channels" para guardar los datos de cada uno de los canales en una columna distinta.	
Write Delimited Spreadsheet.vi	Guarda la información en un documento	- File path: Crea un archivo de datos en la ubicación y con el formato especificados. - Format (%.3f): Configura la cantidad de decimales que tendrán los datos registrados, en este caso se optó por guardar 4 decimales (%.4f).	 

Componente	Función	Configuración	Imagen
Write Delimited Spreadsheet.vi	Guarda la información en un documento	- Append to file?: se configuró con el comando "True" o (T), lo cual permite escribir nuevos datos empezando desde el final del documento indicado en el "file path".	
While loop	Configura el circuito para que funcione en bucle	Se creó dicho componente el cual consta con un control (botón rojo) que detiene todos los procesos de los circuitos dentro de él. Este componente se lo colocó alrededor del circuito diseñado.	 
Control del tiempo	Graba los datos durante un tiempo establecido	El control de tiempo de muestreo se encuentra dentro del While loop, se configuró para contabilizar que marque 1 cuando haya transcurrido 1 segundo. Adicionalmente, se generó un control numérico "Numeric", al cual se le asignó un valor de 5, alegando que se leerán y registrarán los datos durante 5 segundos. Para controlar el fin de la lectura de datos, se colocó un comparador mayor o igual que (≥), para comparar la salida del "Elapsed Time" con el valor asignado en el control numérico. De esta forma, la condición de parada del bucle se activa cuando el "Elapsed Time" indique un valor mayor o igual a 5 deteniendo automáticamente la lectura y registro de los datos. El indicador "Elapsed Time (s)" permite visualizar en tiempo real el tiempo transcurrido.	   

5.3.1.6 Configuración de la DAQ 6009 para la toma de datos. Para adquirir la señal de los sensores, se conectaron los canales positivos de la tarjeta DAQ a la señal y los canales negativos a la tierra de cada sensor, tal como se indica en la Tabla 10.

Tabla 10. Conexión de la tarjeta DAQ con los sensores de la unidad experimental

Pin Analógico (DAQ)	Pin del sensor	Imagen de la conexión
AI0+	Señal del CMP	
AI0-	Masa del CMP	
AI1+	Señal del MAP	
AI1-	Masa del MAP	
AI2+	Señal del CKP	
AI2-	MAP del CKP	
Diagrama eléctrico de conexión del sensor CMP		

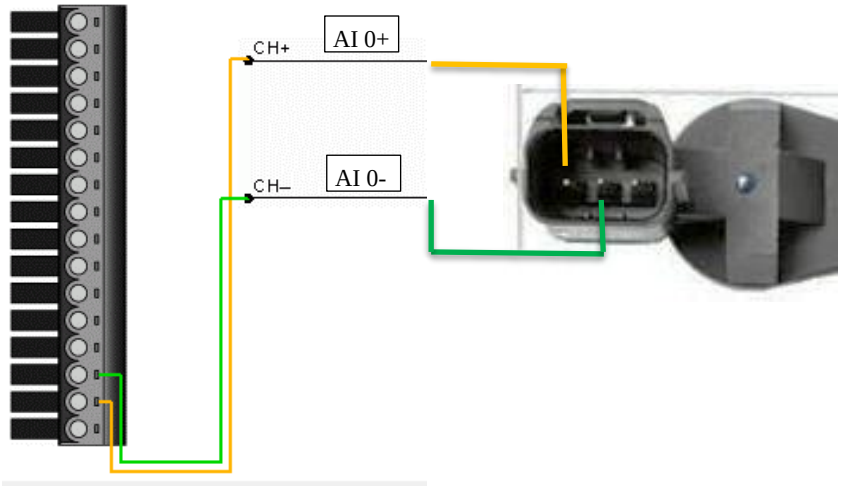


Diagrama eléctrico de conexión del sensor MAP

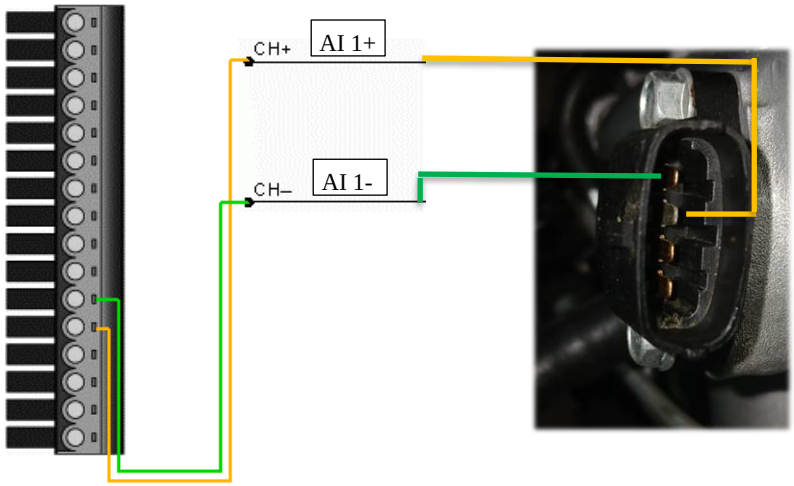
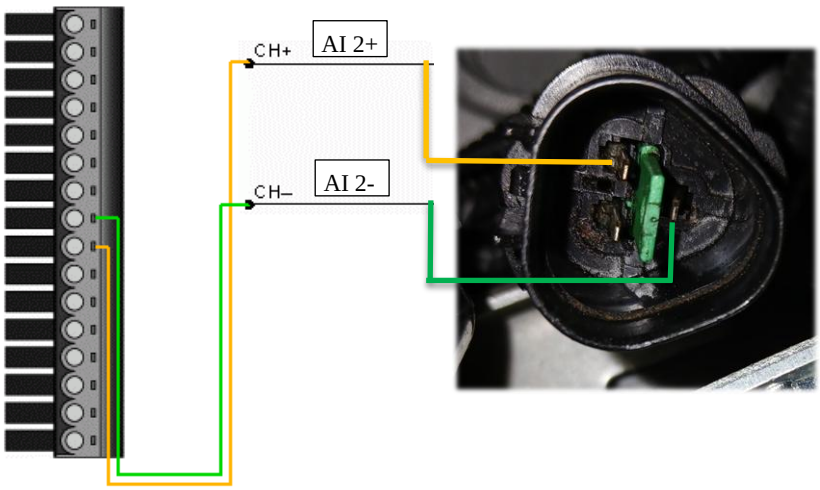


Diagrama eléctrico de conexión del sensor CKP



5.3.1.7 Protocolo de adquisición de datos. El protocolo de adquisición de los datos a partir de los tratamientos está referido a la forma en que se realizó la adquisición de datos en la presente investigación.

El protocolo se resume en el flujograma de la Figura 21.

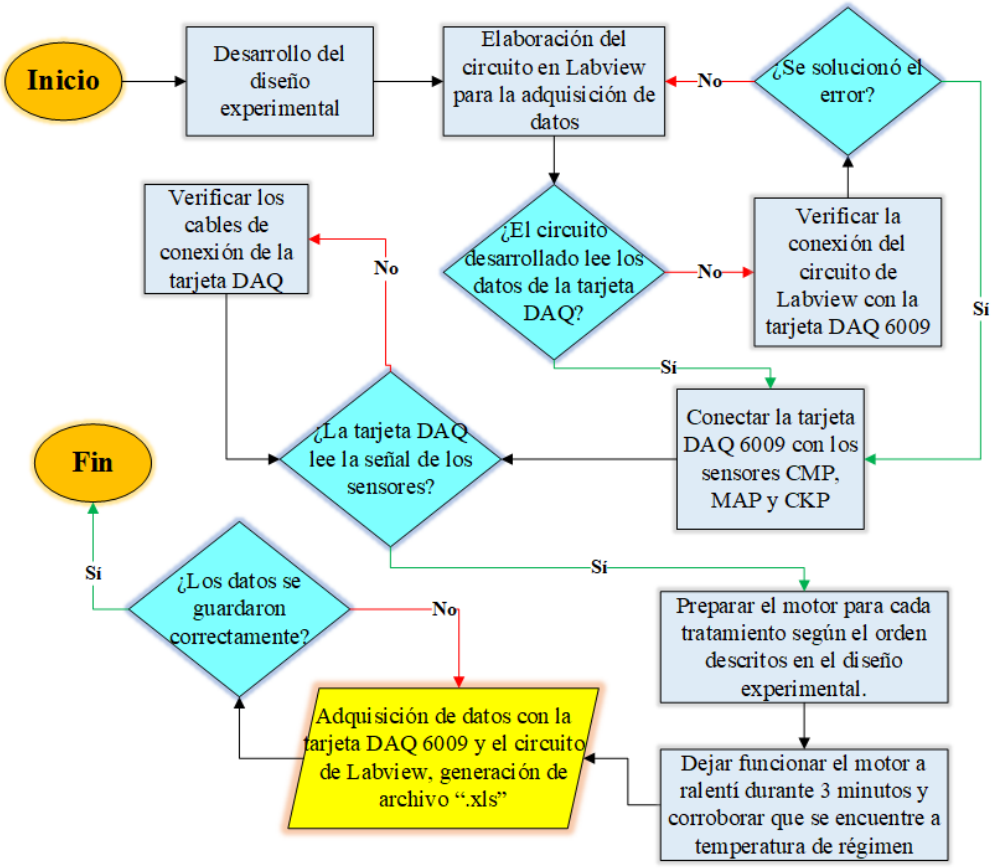


Figura 21. Flujograma empleado para la adquisición de datos del sensor MAP.

Los datos se registraron en distintos archivos con extensión ".xls" correspondientes a archivos de Excel, un archivo para cada réplica de cada tratamiento, la magnitud de los datos obtenidos corresponde al voltaje de los sensores en voltios [V], dichos datos se ordenaron en un archivo de Excel tal como se observa en la Figura 22, la primera, segunda y tercera columna corresponden a los datos de los sensores CMP, MAP y CKP respectivamente.

	A	B	C	D
	CMP	MAP	CKP	4.9367
1	4.9367	1.0734	-0.0288	
2	4.9449	1.074	-0.0212	
3	4.9468	1.0555	0.0323	
4	4.9297	1.06	0.0151	
5	4.9297	1.0606	0.0125	
6	4.9367	1.0721	-0.0346	
7	4.9443	1.0702	-0.0263	
8	4.9462	1.0651	0.0285	
9	4.9303	1.0511	0.0164	
10	4.9309	1.0587	0.0151	
11	4.9379	1.0676	-0.0301	
12	4.943	1.0759	-0.0206	
13	4.9449	1.0492	0.0335	

	CMP	MAP	CKP
54990			
54991			
54992	0.0133	1.1039	-0.0263
54993	0.0114	1.0886	0.0297
54994	0.0286	1.0906	0.024
54995	0.0305	1.0797	0.017
54996	0.0184	1.1033	-0.032
54997	0.0133	1.0906	-0.0269
54998	0.0114	1.0886	0.0259
54999	0.0292	1.081	0.0176
55000	0.0318	1.0721	0.0132
55001			
55002			
55003			

T1_Toma1

Figura 22. Excel de los datos obtenidos con la tarjeta DAQ.

5.3.2 Tratamiento de los datos y extracción de características

Los datos obtenidos por el circuito en LabVIEW no inician en un mismo ángulo de giro del cigüeñal, por lo tanto, el primer tratamiento que se realizó a los datos, correspondió a la selección y clasificación manual de los datos del sensor MAP: marcando el inicio de la señal el salto de chispa para la primera ignición y el final como el salto de chispa de la última ignición; repitiendo el proceso para las demás réplicas. Para ello, se referencia al manual de la unidad experimental correspondiente a la señal del sensor CMP, en donde la Figura 23 indica que los puntos 5 y 6 hacen referencia al PMS de los cilindros 1 y 4 respectivamente (YES01, 2010).

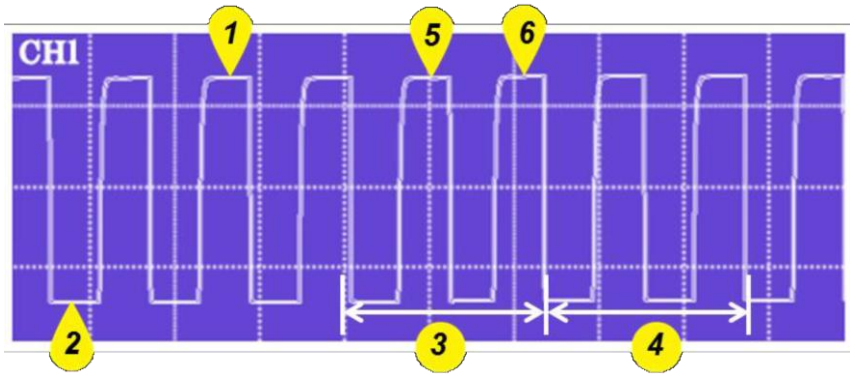


Figura 23. Señal del CMP indicando el PMS de los cilindros 1 y 4.

De esta forma, se tomaron las señales de los sensores MAP, CMP, CKP y el voltaje de la bobina del cilindro 1 con un osciloscopio, de esta forma en la Figura 24 se puede apreciar que el salto de chispa se produce a la par de la caída de la señal del sensor CKP a la vez que se encuentra dentro de la señal alta y de mayor duración del sensor CMP. Dicha condición, se corrobora con la Figura 23, puesto que el manual indica que el PMS del cilindro 1 se encuentra dentro de un intervalo alto de la señal del CMP. Adicionalmente, la Figura 24 indica que el fin del ciclo del motor también tiene la misma condición que el inicio, marcando un ciclo completo (720° de giro del cigüeñal) en el intervalo A-B los cuales indican el salto de chispa del cilindro 1. Adicionalmente, el osciloscopio indica que la duración de un ciclo completo en ralentí es de 162 ms.

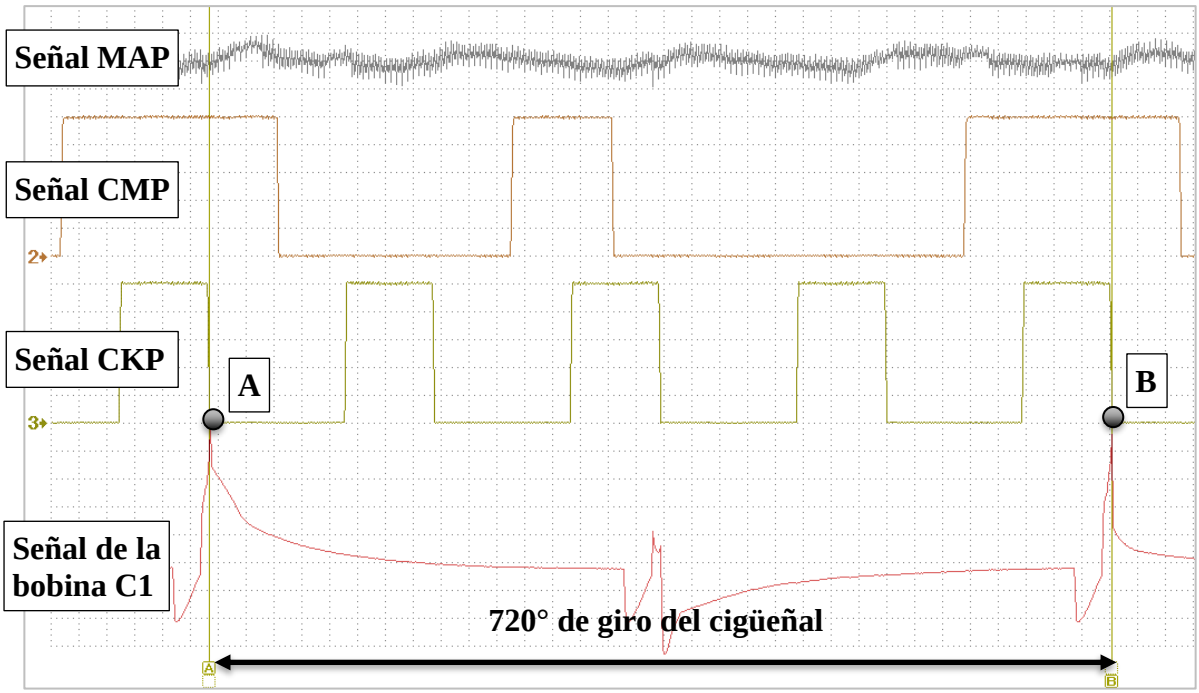


Figura 24. Definición del comienzo y el final de un ciclo completo del motor partiendo desde el PMS al final de la carrera de compresión.

Determinadas las condiciones de inicio y final, se extrajeron los datos del sensor MAP y el sensor CKP, de manera que, el inicio y el final de toda la señal cumplan las condiciones dadas del punto A y el punto B, respectivamente. El proceso se realizó manualmente en Excel, resaltando las celdas con el formato condicional, en donde, los valores mayores a 4V correspondientes al sensor CMP se resaltaron con color rojo, mientras que, los datos del sensor CKP menores a 1V se resaltaron con color amarillo (Ver Figura 25).

	A	D	E	H
1	4.9411	0288	4.9328	0272
2	CMP	MAP	CKP	CMP	MAP	CKP
3						
4	4.9265	1.0988	0.0145	4.9398	1.1262	-0.0269
5	4.9303	1.1077	-0.0244	4.943	1.1141	-0.0263
6	4.943	1.1224	-0.0314	4.9309	1.109	0.024
7	4.9443	1.1198	-0.0295	4.9347	1.1077	0.0145
8	4.9284	1.1065	0.0227	4.9303	1.1084	-0.0072
9	4.9297	1.1154	0.017	4.9411	1.1262	-0.032
10	4.9297	1.1084	-0.0161	4.9443	1.1205	-0.0282
11	4.9443	1.1326	-0.0288	4.9341	1.1154	0.0272
12	4.9455	1.1358	-0.0288	4.929	1.1116	0.0189
13	4.9297	1.102	0.0189	4.9309	1.1167	-0.0015
14	4.929	1.1326	0.0151	4.943	1.1249	-0.0263

Réplica 1

Réplica 2

Figura 25. Selección de los datos del sensor MAP desde el inicio del ciclo hasta el fin del ciclo del motor.

Los datos de los sensores MAP y CKP de las 10 réplicas se agruparon en la primera y segunda fila de un archivo ".mat" para cada tratamiento, y, finalmente los datos de los 4 tratamientos se guardaron en un único archivo denominado "MAP_CKP.mat". En la Figura 26 (a) el archivo "T1_MAP_CKP" corresponde a todos los datos de las 10 réplicas del Tratamiento 1, siguiendo una denominación similar para con los demás archivos; en adición, se realizó el ploteo de los datos del MAP en bruto obteniendo el resultado de la Figura 26 (b).

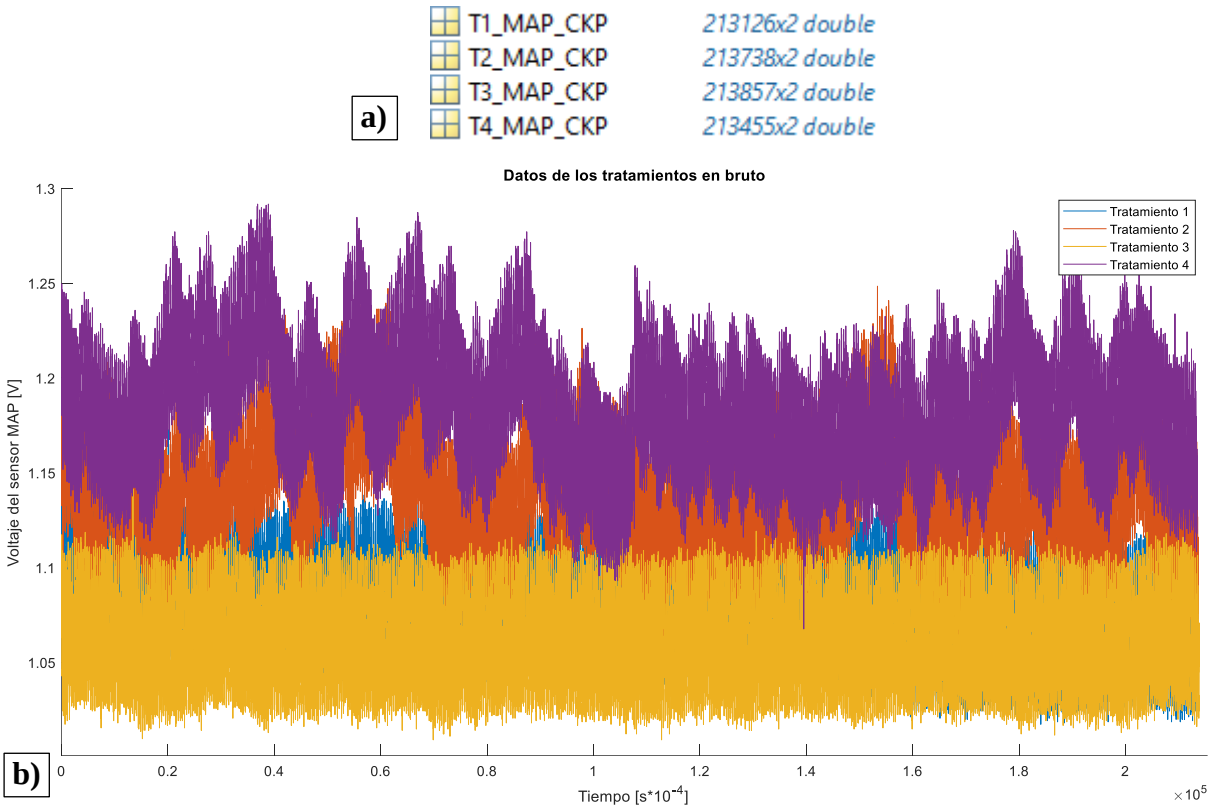


Figura 26. Datos de los 4 tratamientos en bruto.

5.3.2.1 Corte de los datos del MAP cada 720° de giro del cigüeñal. En la metodología empleada por Delgado (2018), se cortan fragmentos de los datos adquiridos cada 2 segundos, es por ello que en la presente investigación se realizó un proceso similar.

Se realizó los cortes a la señal del MAP para cada ciclo completo del motor. Recalcando que, no se consideró pertinente realizar un corte cada 1620 datos (equivalentes a los 162 ms) puesto que el ralentí del motor no era constante, más bien, era inestable. Sin embargo, como se observó en la Figura 24, cada 4 picos bajos de la señal del CKP corresponden a 720° de giro del cigüeñal, adicionalmente dichos picos bajos están situados después del último intervalo del pico alto del sensor CKP.

Se utilizó un código de Matlab para que detecte el primer pico bajo (< 0.5 V) del sensor CKP, con la condición de que este se encuentre luego del último intervalo mayor a 4 V. El pico alto pico alto se detectó luego de 1400 datos (dependiendo de cada tratamiento) contados a partir del pico bajo detectado previamente. Finalmente, se almacenaron los datos del sensor MAP que se encuentran dentro de dos picos bajos detectados. El código empleado fue el siguiente:

```
load MAP_CKP.mat % Se carga el archivo con los datos de los tratamientos
MAP = T4_MAP_CKP(:,1); % Señal del MAP del tratamiento correspondiente
CKP = T4_MAP_CKP(:,2); % Señal del CKP del tratamiento correspondiente
% Parámetros
segment_size = 1400; % Número de datos a considerar desde cada pico bajo
detectado
picos_detectados = []; % Almacena los índices de los picos bajos encontrados
inicio_busqueda = 1; % Índice inicial de la búsqueda
% Bucle condicionado a la longitud de datos del CKP
while inicio_busqueda < length(CKP)
    segmento = CKP(inicio_busqueda:min(inicio_busqueda+segment_size-1,
length(CKP))); % Extrae el segmento del CKP de 1400 datos, desde el dato
inicial

    if isempty(picos_detectados)
        idx = find(segmento < 0.5, 1, 'first'); % Detecta el primer pico bajo
que no está condicionado a ir después de un pico alto
    else
        % Se buscan los picos bajos que cumplan la condición
        encontrado_pico_alto = false;
        idx = []; % Se almacenan los picos bajos encontrados
        for j = 1:length(segmento)
            valor_actual = segmento(j); % Indica el segmento tomado del CKP
            if ~encontrado_pico_alto % Condición True del bucle
                if valor_actual > 2.5 % Busca un pico alto (> 2.5)
                    encontrado_pico_alto = true;
                end % Deja de buscar un pico alto
            else  % Condición False del bucle
                if valor_actual < 0.5 % Busca un pico bajo luego del pico alto
<0.5
                    idx = j; % Almacena el pico bajo que cumple la condición
                    break; % Deja de buscar los picos bajos
                end
            end
        end
    end

    % Si se encontró un pico bajo, se agrega al archivo “picos detectados”
    if ~isempty(idx)
        idx_global = inicio_busqueda + idx - 1; % Convertir a índice global
```

```
        picos_detectados = [picos_detectados; idx_global];
        inicio_busqueda = idx_global + segment_size; % Actualiza el inicio de
búsqueda al próximo bloque de 1400 datos
    else
        break; % Si no se encontró un pico bajo, terminar la búsqueda
    end
end

MAP_segmentos = cell(length(picos_detectados), 1); % Almacena los segmentos
del MAP en celdas diferentes
for k = 1:length(picos_detectados)-1 % Bucle de almacenamiento de los
segmentos
    MAP_segmentos{k} = MAP(picos_detectados(k):picos_detectados(k+1)-1);
End

ultimo_pico_idx = picos_detectados(end); % Índice del último pico bajo
detectado
MAP_segmentos{end} = MAP(ultimo_pico_idx:end); % Guarda el último segmento del
MAP (desde el último pico detectado hasta el final de los datos)
```

La Figura 27 (a) indica de manera representativa el ploteo de la señal del sensor CKP junto con los picos bajos detectados por el código, la Figura 27 (b) muestra un segmento de la señal del sensor MAP del tratamiento 4 que fue almacenado en la primera celda. El código para el ploteo no se incluye en la redacción puesto que la figura tiene la finalidad de otorgar una mayor comprensión al lector acerca de lo que realiza el código.

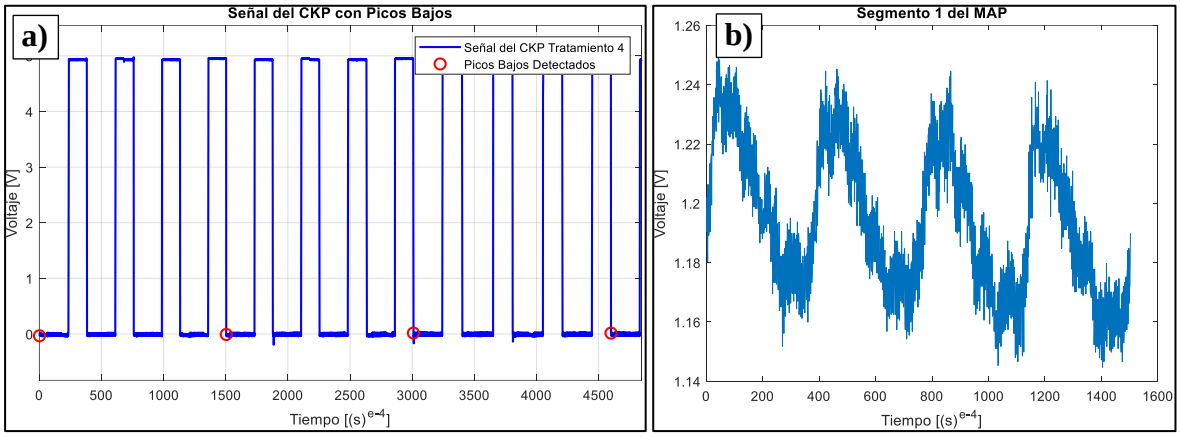


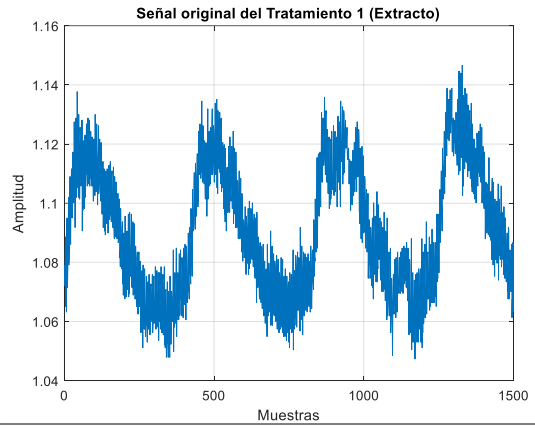
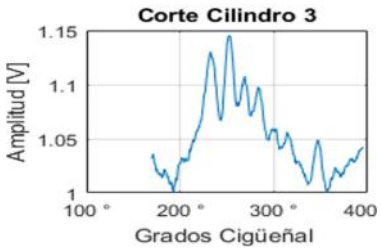
Figura 27. Representación gráfica del funcionamiento del código de detección de picos bajos y corte de la señal del MAP.

5.3.2.2 Aplicación de un filtro a las señales del sensor MAP. En la investigación se trabajó tanto con los datos en bruto como con los datos filtrados.

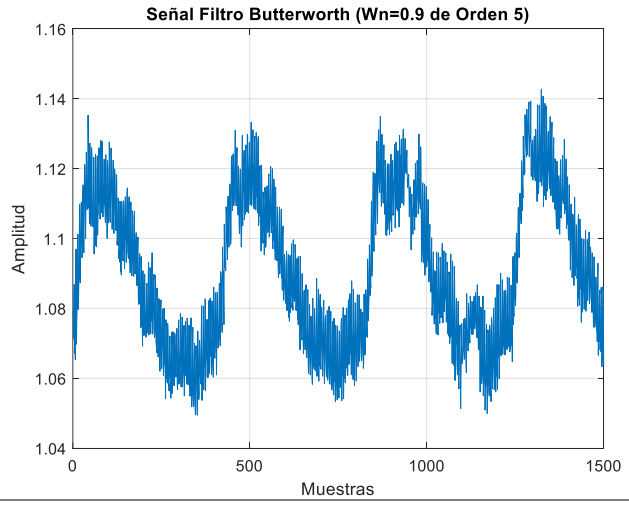
Su finalidad fue comparar el cómo se ve afectado el rendimiento de los algoritmos de Machine Learning al ingresar los datos originales (con presencia de ruido) y los datos filtrados (con menos ruido) método aplicado por Contreras U. et al. (2019).

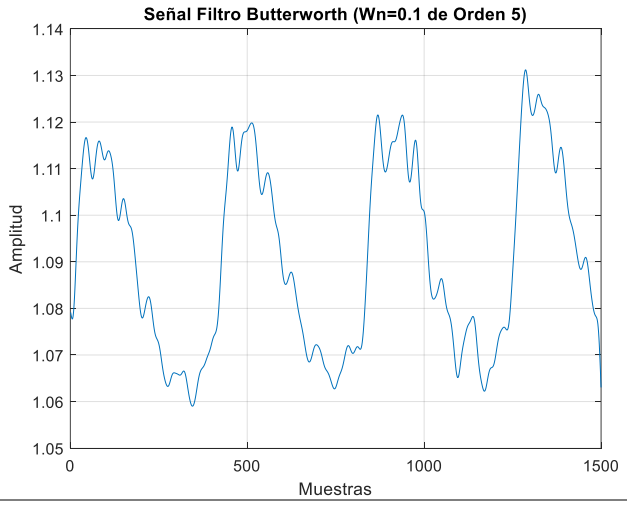
Se hizo uso de un filtro "Butterworth" pasa baja con distintas configuraciones orden del filtro y frecuencia de corte, con la finalidad de obtener una señal que conserve el espectro de la señal original, en la Tabla 11 se pueden observar las distintas señales obtenidas con el filtro, la señal original y la señal con la cual trabajan Contreras U. et al. (2019).

Tabla 11. Aplicación del filtro Butterworth con distintas configuraciones del número de orden y frecuencia de corte.

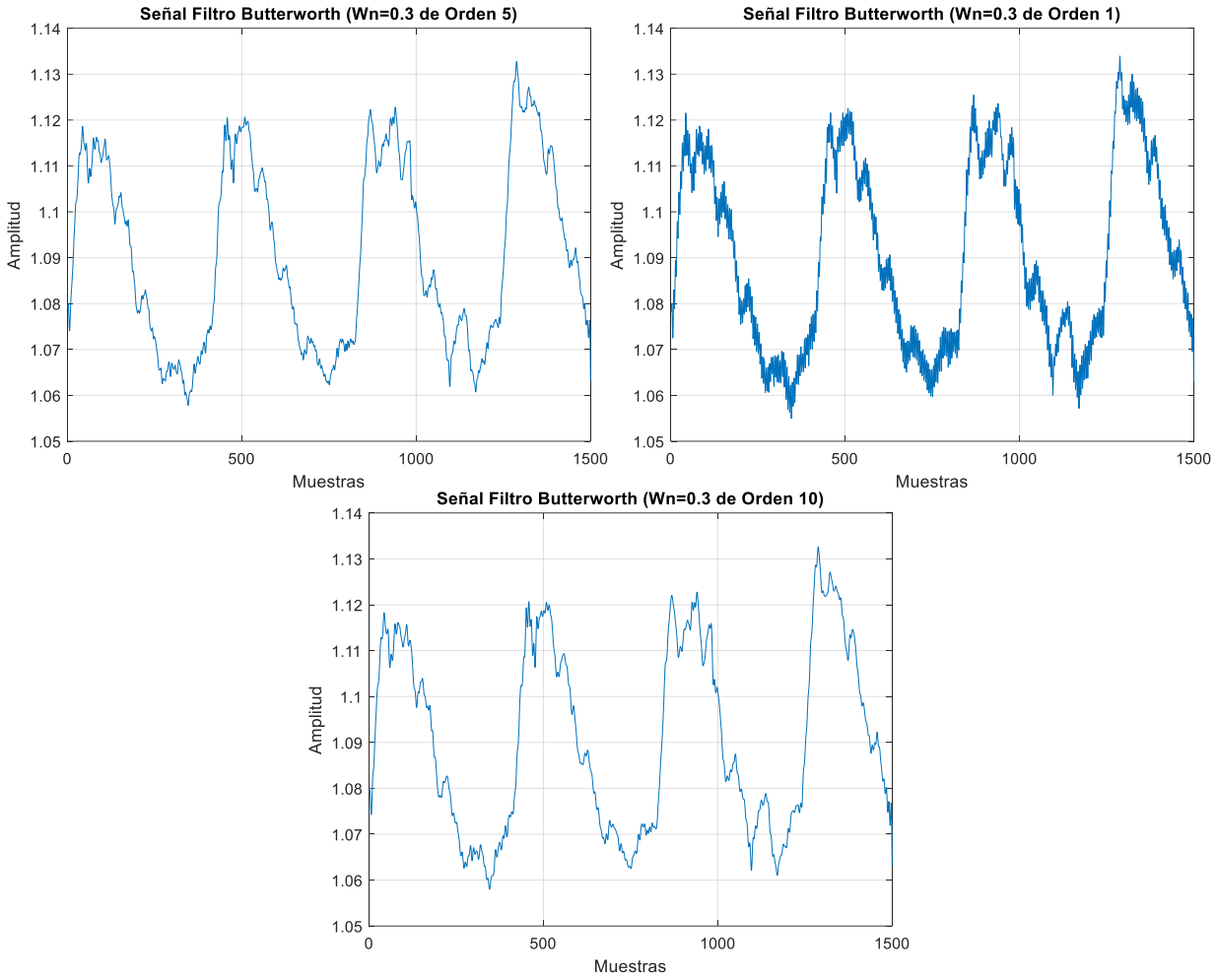
Señal original	**Señal del MAP (Estudio Previo)**
	

Señales filtradas con diferente configuración del filtro "Butterworth"





Señales filtradas con diferente configuración del filtro “Butterworth”



Al comparar el comportamiento de ambos tipos de eliminación de ruido de la señal, la configuración que mejor conservó el espectro de la señal original sin perder demasiada información fue la del filtro Butterworth de orden 5 con frecuencia de corte 0.3, es por ello que dicha configuración se aplicó a todas las señales de los demás tratamientos. Para ello, se modificaron las siguientes líneas de código al algoritmo explicado previamente:

```
MAP_o = T4_MAP_CKP(:,1); % Señal del MAP Original del tratamiento
correspondiente
% Diseño del Butterworth
w = 10000; % Frecuencia de los datos
n=5; % Orden del filtro
wn=0.3; % Frecuencia de corte
[b, a] = butter(n, wn, 'low'); % Filtro pasa baja
MAP = filtfilt(b, a, MAP_o); % Aplicar el filtro a la señal del MAP
```

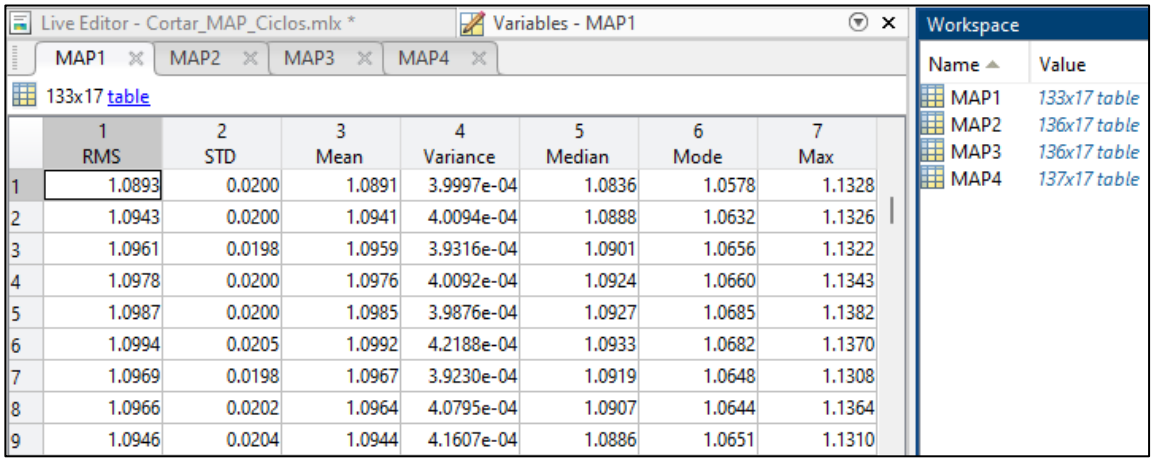
5.3.2.3 Extracción de características de las señales del sensor MAP. Según Omama et al. (2019), la extracción de características de las señales permite tener un mayor desempeño de los algoritmos de clasificación, método que es aplicado en su investigación al igual que en el trabajo realizado por Delgado (2018) y Contreras U. et al. (2019).

Por lo tanto, en la investigación se extrajeron los valores de: valor cuadrático medio (RMS), desviación estándar (STD), promedio, varianza, mediana, moda, el pico máximo, el pico mínimo, la diferencia entre el pico máximo y el pico mínimo, energía, asimetría, curtosis, factor de cresta, coeficiente de variación, la media recortada en un 10%, entropía de Shannon y el área bajo la curva para cada uno de los cortes de la señal. El código empleado para encontrar dichas características fue el siguiente:

```
num_segmentos = length(MAP_segmentos); Almacena los resultados en una tabla
MAP4 = table(); % Genera una tabla denominada “MAP4”

% Función para calcular la entropía de Shannon
calcular_entropia = @(x) -sum((x/sum(x)).*log2(x/sum(x) + eps)); % Se agrega
eps para evitar log(0)

for i = 1:num_segmentos % Inicia el bucle para cada segmento almacenado del
MAP
    segmento = MAP_segmentos{i}; % Tomar el segmento actual del MAP de la
celda

    % Extracción de características de cada corte o segmento
    rms_val = rms(segmento);                        % RMS
    std_val = std(segmento);                        % Desviación estándar
    mean_val = mean(segmento);                      % Promedio
    var_val = var(segmento);                        % Varianza
    median_val = median(segmento);                  % Mediana
    mode_val = mode(segmento);                      % Moda
    max_val = max(segmento);                        % Máximo
    min_val = min(segmento);                        % Mínimo
    peak_to_peak = max_val - min_val;               % Diferencia pico a pico
    energia = sum(segmento.^2);                     % Energía
    asimetria = skewness(segmento);                 % Asimetría
    kurtosis_val = kurtosis(segmento);              % Curtosis
    factor_crest = max_val / rms_val;               % Factor de cresta
    coef_var = std_val / mean_val;                  % Coeficiente de variación
    media_recortada = trimmean(segmento, 10);       % Media recortada (10% de
datos)
    entropia = calcular_entropia(abs(segmento));    % Entropía de Shannon
(función del código descrita anteriormente)
    area = trapz(segmento);                         % Área bajo la curva
```

```
    % Se agregan los resultados de las características a la tabla creada
"MAP4"
    MAP4 = [resultados; table(rms_val, std_val, mean_val, var_val,
    median_val, mode_val, max_val, min_val, peak_to_peak, energía, asimetria,
    kurtosis_val, factor_crest, coef_var, media_recortada, entropia, area)];
end % Fin del Bucle
% Se asignan nombres a las columnas de la tabla "MAP4"
MAP4.Properties.VariableNames = {'RMS', 'STD', 'Mean', 'Variance', 'Median',
'Mode', 'Max', 'Min', 'PeakToPeak', 'Energy', 'Skewness', 'Kurtosis',
'CrestFactor', 'CoeffVariation', 'TrimmedMean', 'Entropy', 'Area'};

save('Atributos_MAP4.mat', 'MAP4'); % Se Guarda la tabla en un archivo .mat
```

Producto del código se generaron los archivos ".mat" (MAP1, MAP2, MAP3 y MAP4) correspondientes a las características de cada uno de los cortes. Los datos se organizaron de tal manera que, cada una de las filas corresponde a cada uno de los cortes realizados a la señal del MAP correspondiente, y cada columna corresponde a los 17 atributos que se calcularon mediante el código (Ver Figura 28).



	1 RMS	2 STD	3 Mean	4 Variance	5 Median	6 Mode	7 Max
1	1.0893	0.0200	1.0891	3.9997e-04	1.0836	1.0578	1.1328
2	1.0943	0.0200	1.0941	4.0094e-04	1.0888	1.0632	1.1326
3	1.0961	0.0198	1.0959	3.9316e-04	1.0901	1.0656	1.1322
4	1.0978	0.0200	1.0976	4.0092e-04	1.0924	1.0660	1.1343
5	1.0987	0.0200	1.0985	3.9876e-04	1.0927	1.0685	1.1382
6	1.0994	0.0205	1.0992	4.2188e-04	1.0933	1.0682	1.1370
7	1.0969	0.0198	1.0967	3.9230e-04	1.0919	1.0648	1.1308
8	1.0966	0.0202	1.0964	4.0795e-04	1.0907	1.0644	1.1364
9	1.0946	0.0204	1.0944	4.1607e-04	1.0886	1.0651	1.1310

Figura 28. Características obtenidas para cada corte de la señal.

Posteriormente se creó una variable de identificación (target) para asignar a cada fila de datos, el número del tratamiento al que corresponde, para ello se tiene en cuenta el número de filas producto de cada tratamiento, es decir, como el tratamiento 1 contiene 133 filas, el "target" corresponderá al valor de 1 y se ubicará en la última columna del mismo; siguiendo el mismo proceso para los demás tratamientos. Los datos de los tratamientos se ubican en un solo archivo, los datos del tratamiento actual se colocan debajo de los datos del tratamiento previo. Para ello se hizo uso del siguiente código para ordenar y categorizar los datos obtenidos:

```
target = [ones(133, 1); 2*ones(136, 1); 3*ones(136, 1); 4*ones(137, 1); ]; % Se
crea un solo target para categorizar todas las filas de los 4 tratamientos
```

```
ClassTable = table(target, 'VariableNames', {'Class'}); % Se convierte el target en formato de tabla, cuyo nombre de la columna corresponde a “Class”
Atributos = [MAP1;MAP2;MAP3;MAP4]; % Se ordenan todos los atributos de los tratamientos, uno debajo del otro
Atributos_MAP_Target = [Atributos ClassTable]; % Se incluye el identificador de cada fila en la última columna de la matriz de los atributos
```

Finalmente obtiene una matriz final con nombre “Atributos_MAP_target.mat” con magnitud de 542x18, en donde las primeras 17 columnas corresponden a las diferentes características obtenidas de cada corte, mientras que la última columna identifica a las filas según el tratamiento al que pertenecen (Ver Figura 29). Dicho archivo se guardó con el mismo nombre.

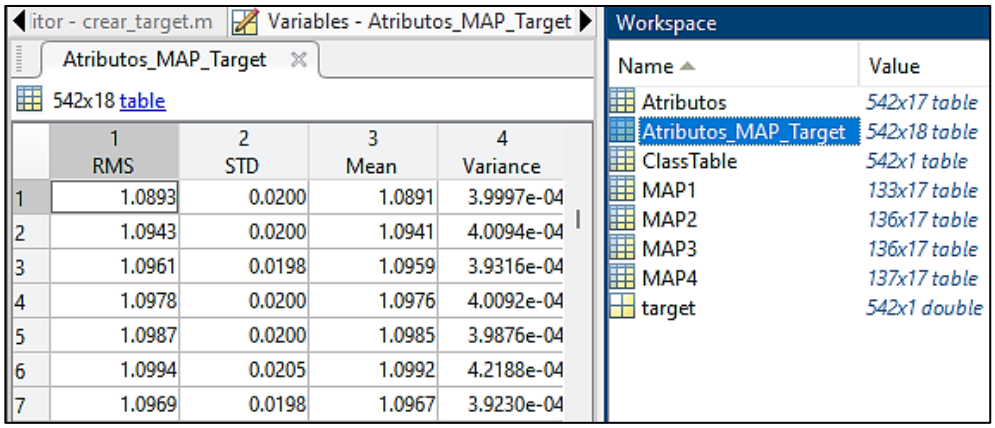


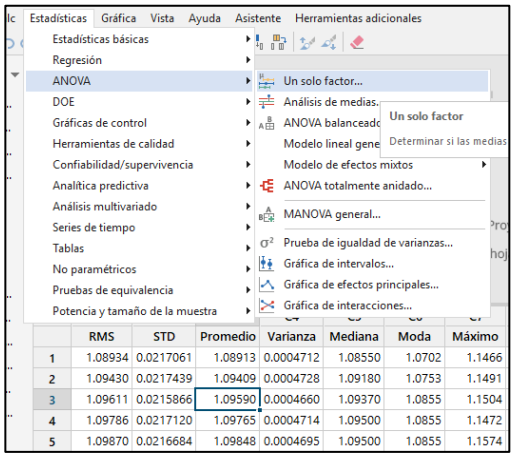
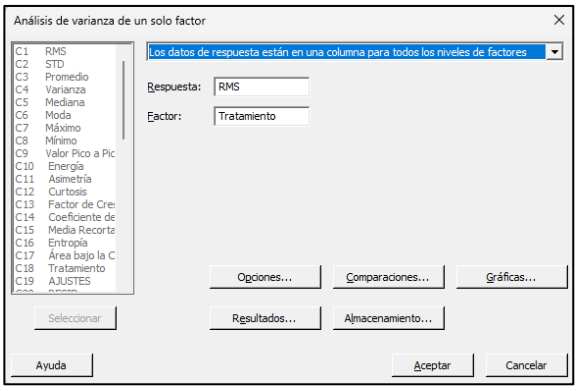
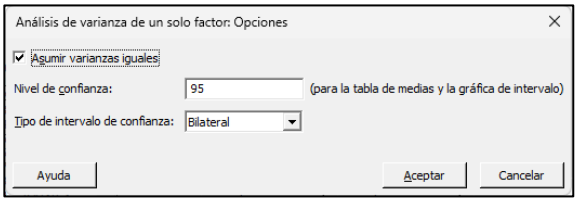
Figura 29. Matriz de las características de la señal con su identificador.

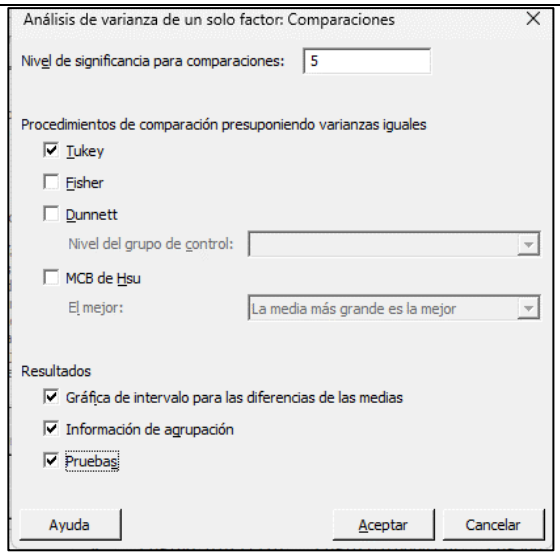
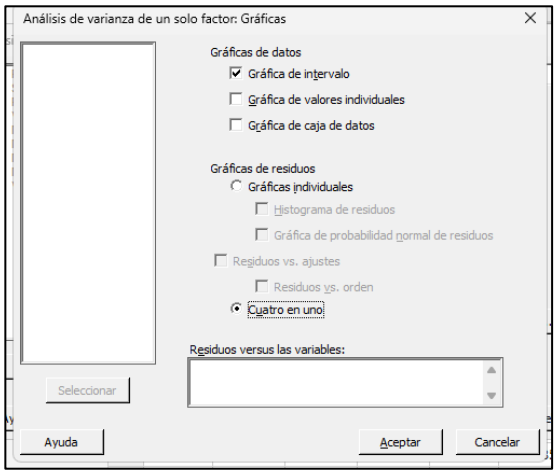
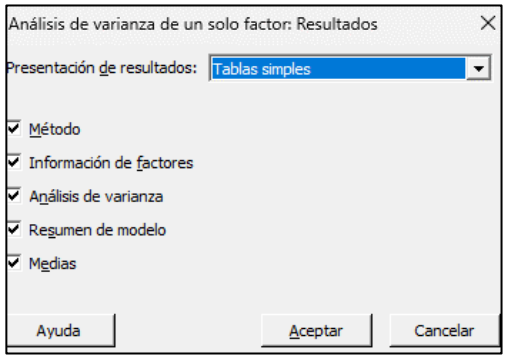
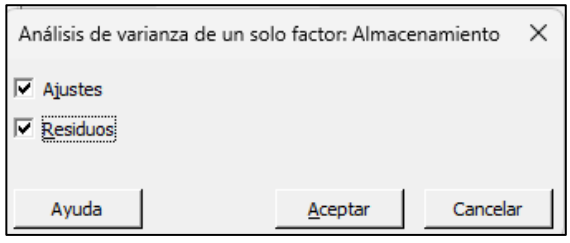
5.3.2.4 Identificación de las características de mayor importancia. Los autores R. V. Sánchez et al. (2024) hacen énfasis en que para el entrenamiento de un algoritmo de Machine Learning, es importante la selección de las características o atributos más relevantes para cada señal.

Para ello, se aplicó una metodología similar a la empleada por Contreras U. et al. (2019), quienes eligen las características más importantes teniendo en cuenta los diagramas de Pareto generados por 5 métodos estadísticos: ANOVA, análisis de correlación y aplicación de 3 métodos de Random Forest.

5.3.2.4.1 Identificación de la importancia de las características mediante el método ANOVA. A continuación, la Tabla 12 describe el proceso de obtención del diagrama de Pareto mediante el análisis estadístico ANOVA realizado en el software Minitab.

Tabla 12. Aplicación del análisis estadístico ANOVA a las características de la señal

Paso	Descripción	Imagen
1	En primer lugar, se copian los datos de la matriz "Atributos_MAP_target.mat" en el software Minitab. Se le asignan los nombres de las variables a cada columna. Se crea un ANOVA de un solo factor: • Estadísticas → ANOVA → Un solo factor	
2	Creado el diseño experimental, se selecciona: - "Los datos de respuesta están en una columna para todos los niveles de factores". - En "Respuesta" se selecciona el nombre de la columna, que corresponde a la característica a la que se le realizará el análisis ANOVA. Se configura el análisis paso a paso mediante la selección de las opciones: - Opciones - Comparaciones - Gráficas - Resultados - Almacenamiento En los pasos posteriores se describe cada apartado.	
3	Dentro de opciones, se asigna un nivel de confianza del 95%, se selecciona "Asumir varianzas iguales" y el tipo de intervalo es Bilateral. Al final, se selecciona "Aceptar".	

Paso	Descripción	Imagen
4	Dentro de “Comparaciones” se selecciona “Turkey” y en el apartado de “Resultados” se seleccionan las 3 opciones disponibles. Luego, se selecciona “Aceptar”.	
5	Dentro de “Gráficas” se selecciona: - Como gráfica de datos “Gráfica de intervalo” - Como gráfica para los resultados “Cuatro en uno” Para concluir con este apartado se selecciona “Aceptar”.	
6	Dentro de “Resultados” se selecciona como presentación de resultados “Tablas Simples” y se seleccionan todas las opciones disponibles: Método, información de factores, análisis de varianza, resumen del modelo y medias. Acto seguido, se selecciona aceptar.	
7	Dentro de “Almacenamiento” se seleccionan las dos opciones disponibles “Ajustes” y “Residuos”. Se selecciona “Aceptar” para guardar la configuración. Finalmente se selecciona la opción “Aceptar” del paso 2 para realizar el análisis ANOVA de la característica seleccionada.	

El análisis ANOVA para un solo factor, arroja los resultados estadísticos presentados en la Tabla 13.

Tabla 13. Aplicación del análisis estadístico ANOVA a las características de la señal

Imagen	Descripción
Análisis de Varianza Fuente GL SC Ajust. MC Ajust. Valor F Valor p Tratamiento 3 1.2583 0.419425 1781.74 0.000 Error 538 0.1266 0.000235 Total 541 1.3849 Resumen del modelo S R-cuad. R-cuad. (ajustado) R-cuad. (pred) 0.0153428 90.86% 90.80% 90.72%	**Análisis de varianza (ANOVA):** - El valor p indica que al menos dos tratamientos tienen un valor de RMS diferente. - El valor SC ajustado (variación entre tratamientos), indica que los datos tienen una variación total de 1.3849, de esta cantidad, 1.2583 se debe a la diferencia entre los valores de RMS entre los distintos tratamientos y 0.1266 corresponde a la diferencia entre los valores de RMS de un mismo tratamiento. - El valor del MC ajustado (cuadrados medios), indica que las diferencias entre los valores de RMS de los distintos tratamientos corresponden a 0.419425 y el error es de 0.000235, el primer valor es significativamente mayor al segundo, por lo que indica que las diferencias entre los valores de RMS de los distintos tratamientos son distintas. **R-Cuadrado:** Indica que el RMS explican en un 90.86% de la variabilidad entre los tratamientos.
confianza de 95% Tratamiento N Media Agrupación 4 137 1.18236 A 2 136 1.13767 B 1 133 1.07954 C 3 136 1.06079 D *Las medias que no comparten una letra son significativamente diferentes.*	La tabla de la agrupación en parejas de Turkey, establece que la variable RMS entre los tratamientos 1, 2, 3 y 4, son significativamente diferentes. Puesto que cada una corresponde a una letra diferente. Por lo cual el modelo de Machine Learning puede clasificar este parámetro.

Nota. La descripción de cada uno de los resultados del ANOVA para la selección de las mejores características, se basó en lo mencionado por H. Gutiérrez y Salazar (2008) y Contreras U. et al. (2019).

Se realizó el mismo proceso para cada una de las características, y se registraron los valores de R^2 y el valor p en la Tabla 14.

Tabla 14. Valores de p y R^2 para cada una de las características.

Factores estadísticos	R^2 (%)	Valor p	Factores estadísticos	R^2 (%)	Valor p
RMS	90.86	0	**Energía**	73.08	0
STD	55.72	0	**Asimetría**	11.84	0
Promedio	90.86	0	**Curtosis**	22.57	0
Varianza	52.47	0	**Factor de Cresta**	2.27	0.006
Mediana	91.07	0	**Coeficiente de Variación**	3.27	0
Moda	88.62	0	**Media recortada**	90.88	0
Máximo	90	0	**Entropía**	14.18	0
Mínimo	89.40	0	**Área bajo la curva**	52.12	0
Valor Pico a Pico	48.48	0			

A partir de los valores obtenidos, se realizó el diagrama de Pareto de los valores de R^2, con la finalidad de obtener las 10 características de mayor significancia. Para ello se hizo uso del siguiente código en Matlab.

```
R2 = [90.86;55.72;90.86;52.47;91.07;88.62; 90;89.40;48.48; 73.08;11.84;...
    22.57;2.27; 3.27;90.88;14.18;52.12] % Se escriben los datos del R-cuadrado
R2_n = R2/sum(R2)* 100; % Se normalizan los valores del R-cuadrado
nombres = {'1. RMS', '2. STD', '3. Promedio', '4. Varianza','5. Mediana',...
    '6. Moda','7. Máximo','8. Mínimo', '9. Pico a Pico', '10. Energía', ...
    '11. Asimetría','12. Curtosis', '13. F. Cresta', '14. C. Variación', ...
    '15. Media Recortada','16. Entropía','17. Área'}; % Nombre de
Características
pareto(R2_n,nombres); % Se comanda el ingreso del diagrama de Pareto
ylabel('Nivel de importancia [%]')
title('Diagrama de Pareto (ANOVA)')
grid on;
```

Dicho código permitió visualizar el diagrama de Pareto conseguido con el análisis ANOVA (Ver Figura 30). Las variables de mayor significancia se ordenaron de izquierda a derecha, en donde, la mediana (5) fue la variable más significante, mientras que la varianza (4) fue la de menor significancia.

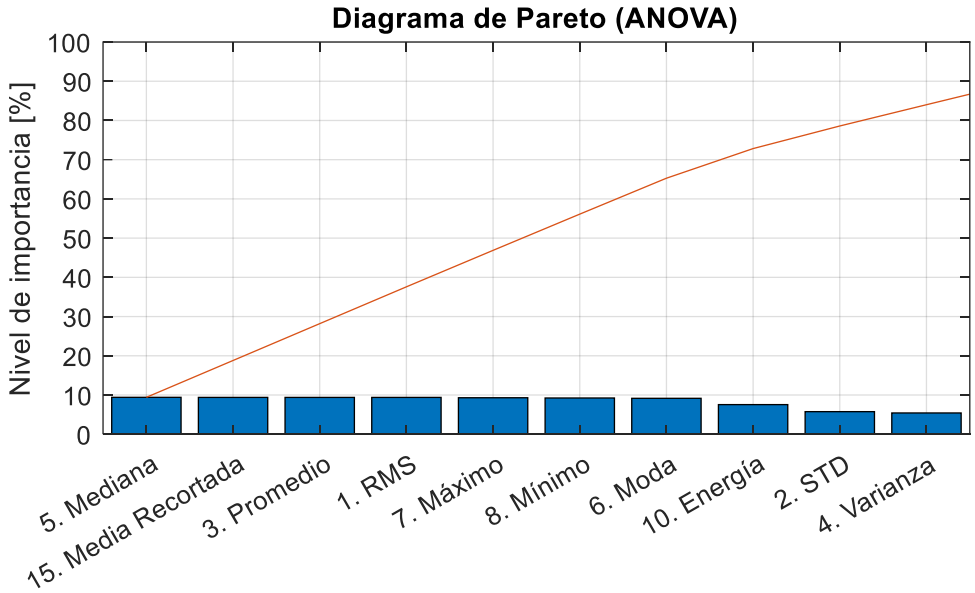


Figura 30. Diagrama de Pareto del análisis ANOVA.

5.3.2.4.2 Identificación de la importancia de las características mediante el análisis de correlación. En el siguiente apartado se realizó el análisis de la matriz de correlación junto con el diagrama de Pareto producto de este análisis.

Para obtener la matriz de correlación, se hizo uso del siguiente código en Matlab.

```
% Separación de variables de atributos y clases
load Atributos_MAP_Target.mat % Se carga el archivo de los datos de las
características y sus identificadores
X = Atributos_MAP_Target(:, 1:17); % Se seleccionan las primeras 17 columnas
correspondientes a los valores de las características
clases = Atributos_MAP_Target(:, 18); % Se seleccionan los datos de la última
columna (18) correspondiente a los identificadores de cada fila

M_correlacion = corr(X); % Se calcula la matriz de correlación: comando “corr”

figure; % Gráfica para la matriz de correlación
heatmap(M_correlacion, 'Colormap', jet, 'ColorbarVisible', true);
title('Matriz de Correlación');
xlabel('Atributos');
ylabel('Atributos');
```

La matriz de correlación de las características de los tratamientos se observa en la Figura 31. La denominación de los ejes X e Y corresponde a las características del estudio: valor cuadrático medio (RMS) (1), desviación estándar (2), promedio (3), varianza (4), mediana (5), moda (6), el pico máximo (7), el pico mínimo (8), la diferencia entre el pico máximo y el mínimo (9), energía (10), asimetría (11), curtosis (12), factor de cresta (13), coeficiente de variación (14), la media recortada en un 10% (15), entropía de Shannon (16) y el área bajo la curva (17).

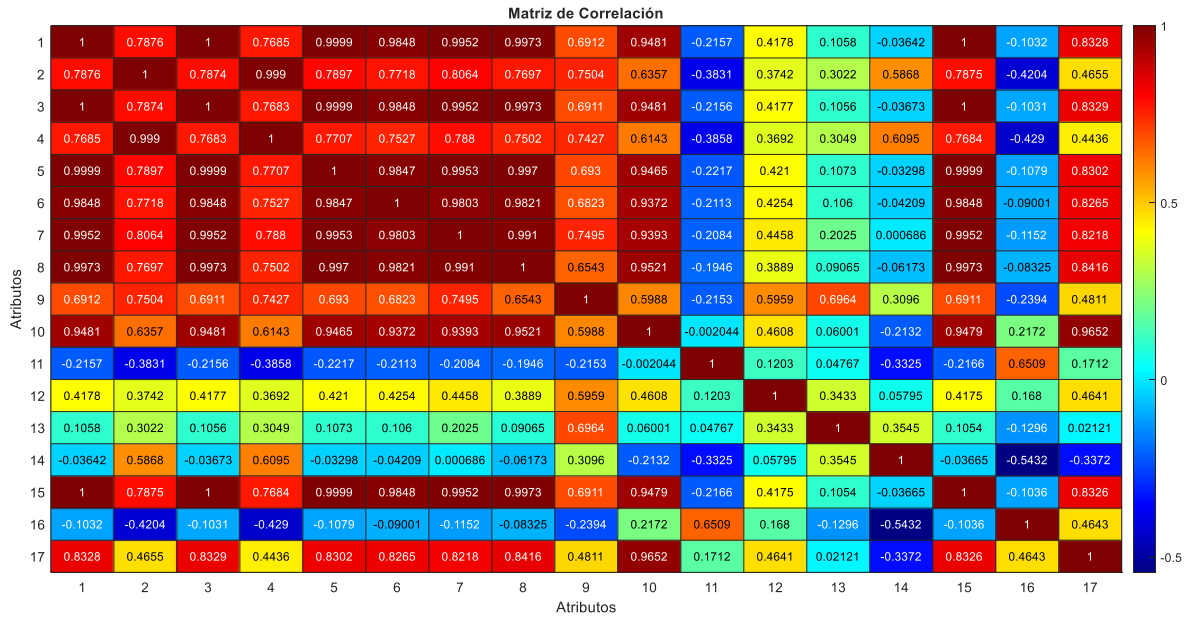


Figura 31. Matriz de correlación de las características de los tratamientos.

Para graficar el nivel de importancia con el diagrama de Pareto, se realizó la suma de los valores de correlación de cada una de las características realizó; dicha importancia se normalizó y se graficó el diagrama de Pareto mediante el siguiente código de Matlab:

```
importancia = sum(abs(M_correlacion), 2) - 1; % Se calcula la importancia de
cada característica, se resta 1 para excluir la correlación consigo misma
importancia_n = importancia / sum(importancia) * 100; % Se normalizan los
valores de la importancia y se extrae el porcentaje
nombres = {'1. RMS', '2. STD', '3. Promedio', '4. Varianza','5. Mediana',...
    '6. Moda','7. Máximo','8. Mínimo', '9. Pico a Pico', '10. Energía', ...
    '11. Asimetría','12. Curtosis', '13. F. Cresta', '14. C. Variación', ...
    '15. Media Recortada','16. Entropía','17. Área'}; % Nombre de
características
figure;
pareto(importancia_n,nombres); % Genera el diagrama de Pareto
title('Diagrama de Pareto (Análisis de Correlación)');
ylabel('Porcentaje de Importancia (%)');
grid on;
```

En la Figura 32 se observa el diagrama de Pareto obtenido por el código, se observa que el atributo de mayor importancia es el pico máximo (7) y el de menor importancia la varianza (4).

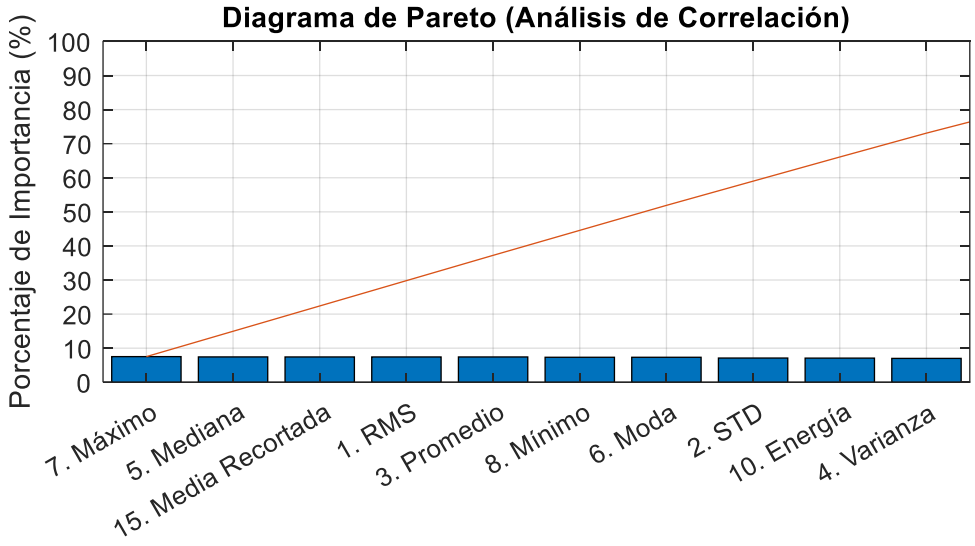


Figura 32. Diagrama de Pareto basado en el análisis de correlación.

5.3.2.4.3 Identificación de la importancia de las características mediante el método de Random Forest (RF). Los 3 métodos empleados para la elección de las características más importantes fueron: “Curvature Test”, “Standard CART” e “Interaction test”. Se configuraron 100 árboles para todos los modelos de Random Forest (RF).

Para el método de “Curvature Test” se hizo uso del siguiente código, en donde la importancia de cada una de las características se calculó mediante el ingreso del comando “'OOBPredictorImportance', 'on'” dentro del modelo del RF.

```
% Separación de variables de atributos y clases
load Atributos_MAP_Target.mat % Archivo de las características e
identificadores
X = Atributos_MAP_Target(:, 1:17); % Se seleccionan las columnas de
características
clases = Atributos_MAP_Target(:, 18); % Se seleccionan los identificadores de
cada fila
rng('default'); % (Selección automática de conjuntos de datos) Sirve para que
el script genere los mismos datos cuando se ejecute nuevamente el script

% Entrenar el modelo Random Forest
n_trees = 100; % Número de árboles
modeloRF = TreeBagger(n_trees, X, clases, 'PredictorSelection',
'curvature',...
    'OOBPredictorImportance', 'on', 'Method', 'regression', 'Surrogate',
'on');
    % Modelo de Random Forest ‘Curvature Test’
importancia = modeloRF.OOBPermutedPredictorDeltaError; % Visualizar la
importancia de las características
importancia_n = importancia / sum(importancia) * 100; % Normalizar la
importancia para el diagrama de Pareto

nombres = {'1. RMS', '2. STD', '3. Promedio', '4. Varianza','5. Mediana',...
    '6. Moda','7. Máximo','8. Mínimo', '9. Pico a Pico', '10. Energía', ...
    '11. Asimetría','12. Curtosis', '13. F. Cresta', '14. C. Variación', ...
    '15. Media Recortada','16. Entropía','17. Área'}; % Nombre de
características

% Graficar el diagrama de Pareto
figure;
pareto(importancia_n,nombres);
title('Diagrama de Pareto (RF Curvature Test)');
ylabel('Porcentaje de Importancia (%)');
grid on;
```

Obteniendo el diagrama de Pareto de la Figura 33 en donde, las características de mayor y menor importancia corresponde, a la entropía (16) y curtosis (12) respectivamente.

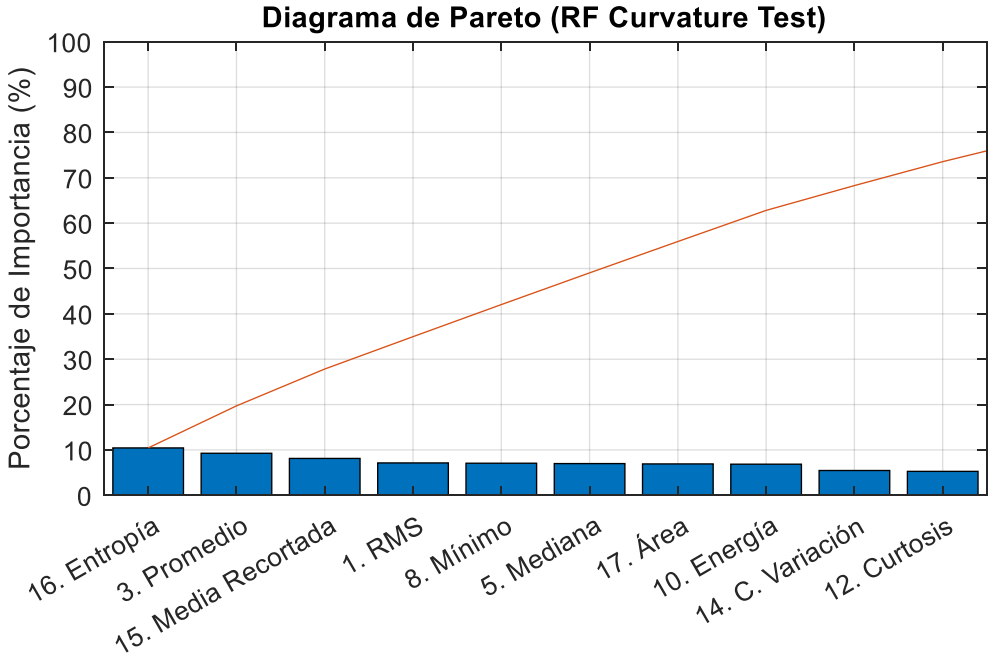


Figura 33. Diagrama de Pareto del modelo "Random Forest Curvature Test".

Para aplicar el Standard CART se modificó la variable "modeloRF" del código anterior, sustituyéndola por:

```
modeloRF = TreeBagger(n_trees, X, clases, 'OOBPredictorImportance', 'on', ...
    'Method', 'regression','Surrogate','on'); % Modelo de RF 'Standard CART'
```

Producto del código presentado y de la modificación del título de la figura, se obtuvo la Figura 34 del diagrama de Pareto para el segundo modelo de RF, en donde el atributo de mayor importancia corresponde a la entropía (16) y el de menor al valor pico a pico (9).

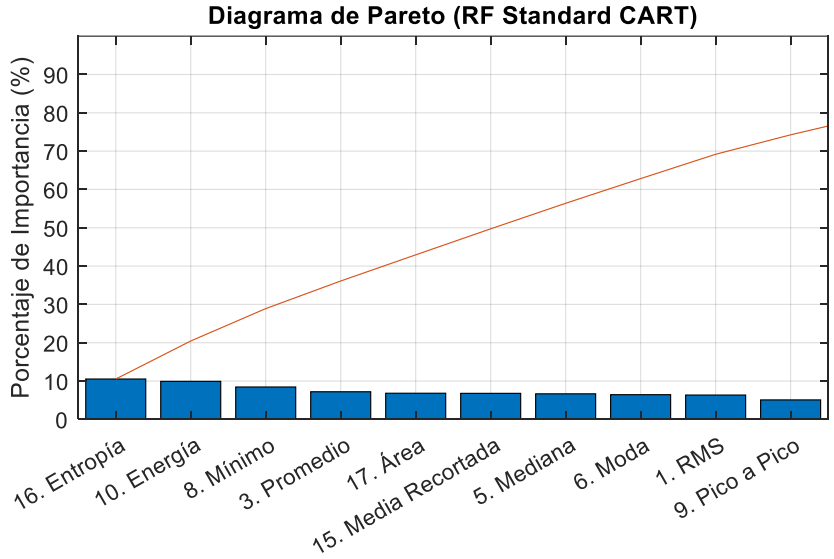


Figura 34. Diagrama de Pareto del modelo "Random Forest Standard CART".

Finalmente, para aplicar el modelo “Interaction-Curvature” se sustituyeron las líneas de código “rng”, “modeloRF” e “importancia” y se agregó la variable “L” correspondiente a los learners del RF. Dichas modificaciones se resumen en el siguiente código:

```
rng(1); % Genera un mismo resultado cada vez que se ejecuta el script (Con la
finalidad de comparar un trabajo previo)

n_trees = 100; % Número de árboles
L = templateTree('NumPredictorsToSample','all','PredictorSelection', ...
    'interaction-curvature','Surrogate','on') % Learners para el modelo de RF
modeloRF = fitrensemble(X, clases, 'Method','bag', 'NumLearningCycles', ...
    n_trees, 'Learners',L); % Modelo RF ‘Interaction-Curvature’
importancia = oobPermutedPredictorImportance(modeloRF); % Extraer la
importancia
```

El código resultó en el diagrama de Pareto de la Figura 35 el cual indica que las características de mayor y menor importancia corresponden a la entropía (16) y los puntos máximos respectivamente (7).

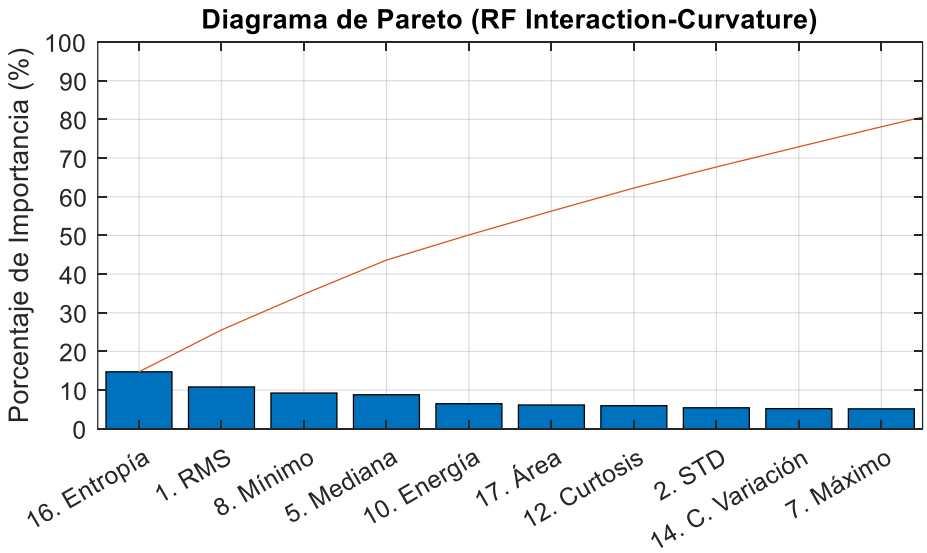


Figura 35. Diagrama de Pareto del modelo “Random Forest Interaction-Curvature”.

5.3.2.5 Extracción de los atributos de mayor importancia para el entrenamiento de la RNA y el SVM. A partir de las gráficas de Pareto obtenidas a partir de los métodos: ANOVA, matriz de correlación y de Random Forest, se realizó la Tabla 15 con los atributos de mayor importancia.

Tabla 15. Atributos de mayor importancia según los métodos estadísticos empleados.

Métodos estadísticos	Atributos de mayor importancia (Más importante a la izquierda)									
ANOVA	5	15	3	1	7	8	6	10	2	4
Correlación	7	5	15	1	3	8	6	2	10	4
Random Forest (Curvature)	16	3	15	1	8	5	17	10	14	12
Random Forest (Standart Cart)	16	10	8	3	17	15	5	6	1	9
Random Forest (Interaction Test)	16	1	8	5	10	17	12	2	14	7

Nota. Las diferentes categorías que se presentan en la tabla corresponden a: RMS (1), STD (2), promedio (3), varianza (4), mediana (5), moda (6), pico máximo (7), pico mínimo (8), energía (10), valor pico a pico (9), asimetría (11), curtosis (12), coeficiente de variación (14), la media recortada (15), entropía (16) y el área (17).

En la Tabla 16 se resaltan las variables de importancia que se repiten 4 o más veces, en los métodos estadísticos aplicados.

Tabla 16. Número de repeticiones como atributos de mayor importancia.

Valores estadísticos	Identificador	Número de repeticiones
RMS	1	5
STD	2	3
Promedio	3	4
Varianza	4	2
Mediana	5	5
Moda	6	3
Máximo	7	3
Mínimo	8	5
Valor Pico a Pico	9	1
Energía	10	5
Asimetría	11	0
Curtosis	12	2
Factor de Cresta	13	0
Coeficiente de Variación	14	2
Media recortada	15	4
Entropía	16	3
Área bajo la curva	17	3
TOTAL (Corroborar el conteo correcto)		50

Según el análisis de correlación, la matriz de correlación de la Figura 31 indica que tanto el RMS, el promedio como la media recortada, tienen una correlación de 1, por lo que solo se tomó en cuenta la característica del RMS, siguiendo la metodología aplicada por Contreras U. et al. (2019). De este modo, las características que se van a utilizar en el entrenamiento de la RNA y SVM serán:

- RMS
- Mediana
- Mínimo
- Energía

En base a estas características se creó el archivo “MAP_Extrac_Caract_Final.mat” el cual incluye los valores de RMS, mediana, mínimo y energía, así como también la columna para identificar cada clase.

Se realizó el mismo proceso con los datos filtrados, obteniendo los atributos de mayor importancia los cuales se detallan en la Tabla 17.

Tabla 17. Atributos de mayor importancia según los métodos estadísticos empleados (Datos con el filtro Butterworth).

Métodos estadísticos	Atributos de mayor importancia (Más importante a la izquierda)									
ANOVA	15	3	1	5	8	6	7	10	9	2
Correlación	7	5	1	3	15	8	6	9	2	4
Random Forest (Curvature)	16	8	6	10	11	17	3	5	15	1
Random Forest (Standart Cart)	16	8	6	7	17	10	1	5	3	15
Random Forest (Interaction Test)	6	16	1	17	5	2	10	12	7	15

Nota. Se categoriza cada característica de la misma manera que la Tabla 15.

Así mismo, en la Tabla 18 se resaltaron aquellas características que se repiten 4 o más veces como atributos de mayor importancia.

Tabla 18. Número de repeticiones como atributos de mayor importancia (Datos con el filtro Butterworth).

Factores estadísticos	Identificador	Número de repeticiones
RMS	1	5
STD	2	3
Promedio	3	4
Varianza	4	1
Mediana	5	5
Moda	6	5
Máximo	7	4
Mínimo	8	4
Valor Pico a Pico	9	2
Energía	10	4
Asimetría	11	1

Factores estadísticos	Identificador	Número de repeticiones
Curtosis	12	1
Factor de Cresta	13	0
Coeficiente de Variación	14	0
Media recortada	15	5
Entropía	16	3
Área bajo la curva	17	3
TOTAL (Corroborar el conteo correcto)		50

La matriz de correlación producto del análisis de correlación (Ver Anexo 6) indicó que tanto el RMS, el promedio como la media recortada, tienen una correlación de 1; de igual manera para con la moda y el mínimo. Por lo tanto, se utilizaron las características que más se repiten: RMS y la moda, descartando las demás características. Obteniendo finalmente las variables con las cuales se entrenarán la RNA y el SVM:

- RMS
- Mediana
- Moda
- Máximo
- Energía

En base a estas características se creó el archivo "MAP_Extrac_Caract_Final_f.mat" el cual incluye los valores de RMS, mediana, moda, máximo y energía de los datos con el filtro Butterworth, también se incluyó la columna para identificar cada clase.

5.3.3 Aplicación del algoritmo de RNA para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto

Para el diseño de la RNA se utilizó una RNA backpropagation referida a la metodología aplicada por Contreras U. et al. (2019), en dicha investigación se entrena un perceptrón multicapa (RNA con capa de entrada, capas ocultas y capa de salida). La RNA que se aplicará en el presente estudio tiene la finalidad de ser capaz de clasificar los 4 estados de funcionamiento del motor explicados en la Tabla 8.

El flujograma de la Figura 36 explica el proceso de aplicación de la RNA para la clasificación de las señales adquiridas y caracterizadas del sensor MAP, para la clasificación de fallas en un motor Otto.

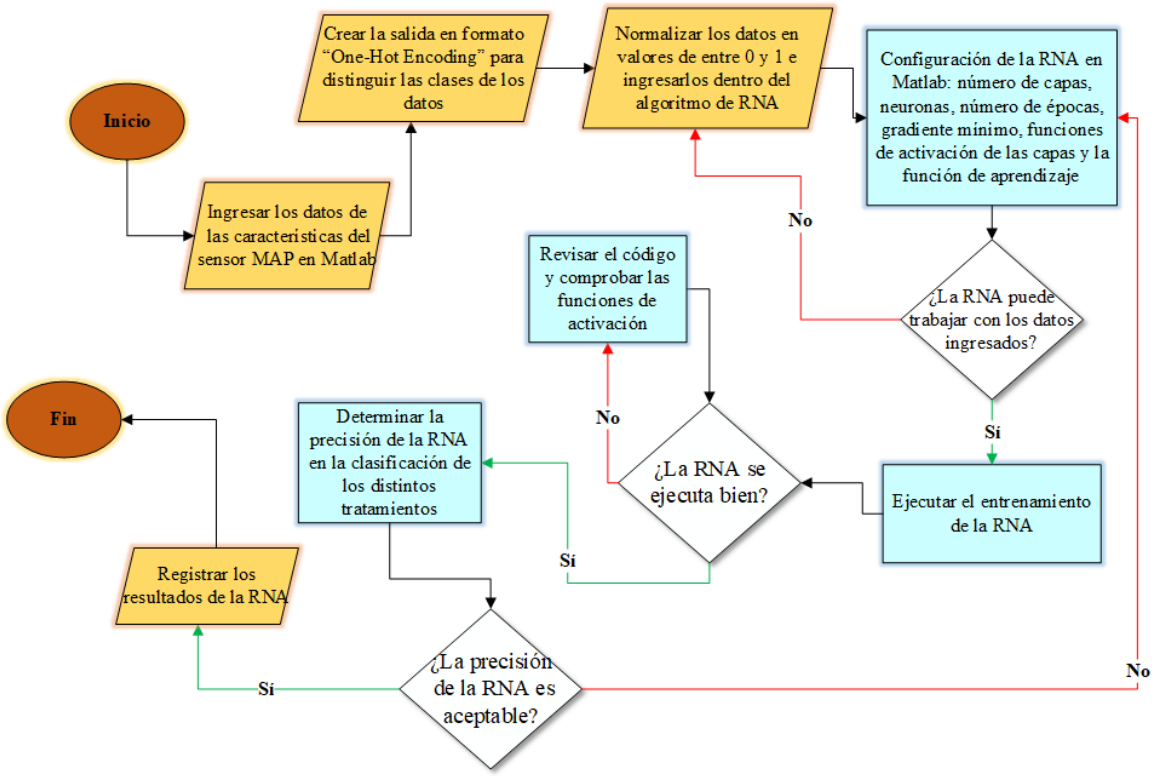


Figura 36. Flujograma de aplicación de la RNA.

5.3.3.1 Generación del archivo de los datos de entrada y salida para el algoritmo de RNA. El proceso se explica a continuación.

En primer lugar, se generó un archivo One-Hot Encoding denominado "Clases.mat" para que la red comprenda a qué tratamiento corresponden todos los datos de una columna de los datos de entrada. Por ello, los datos de entrenamiento se ingresaron mediante la transpuesta (formando una matriz de 4x542), influyendo en la magnitud del archivo "Clases.mat", el cual, de elaboró de la siguiente manera: el número de columnas tienen igual magnitud a los datos de entrada, y, el número de filas corresponde al número de tratamientos que se realizaron (1, 2, 3 y 4) por lo cual, el archivo "Clases.mat" corresponde a una matriz de 4x542.

El archivo One-Hot Encoding permite que la RNA realice la siguiente interpretación:

- Si el valor (1,5) de la matriz "Clases.mat" es 1, entonces los valores de la columna 5 que pertenecen a los datos de entrada son datos del tratamiento 1.
- Si el valor (2,5) es 0, entonces los valores de la columna 5 no corresponden al tratamiento 2.

En la Figura 37 se explica lo mencionado de manera práctica.

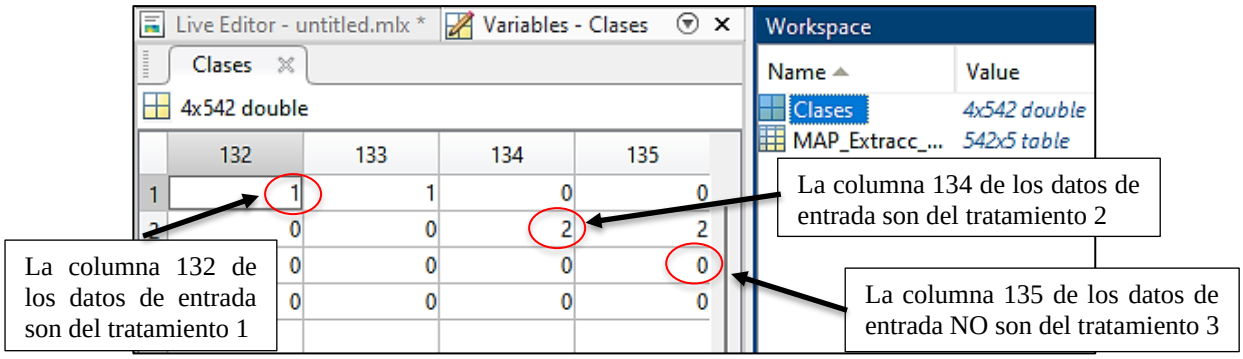


Figura 37. Archivo "Clases.mat" de tipo One-Hot Encoding.

Realizado el identificador para cada clase, se guardaron los archivos "Clases.mat" y "MAP_Extrac_Caract_Final" en un solo archivo denominado "Entrada_RNA_Caracteristicas.mat".

5.3.3.2 Configuración y entrenamiento del algoritmo de RNA. Al corroborar que los datos se organizaron correctamente, se ingresan al algoritmo de la RNA. La variable X corresponden a las características extraídas de todos los tratamientos que ingresarán para el entrenamiento de la RNA.

Además, Tuan et al. (2021) mencionan que es necesario normalizar los datos de entrada, por lo cual se normalizaron los datos en función de su máximo. Para ello se utilizó el siguiente código:

```
load Entrada_RNA_Caracteristicas.mat; % Se carga el archivo de los datos
X = table2array(MAP_Extracc_Caract_Final(:, 1:(end-1))); % Se seleccionan las
características de la señal
maximos=max(X); % Se encuentran los valores máximos de cada característica

for i=1:(size(X,2)) % Bucle para normalizar las características
    A(:,i)=X(:,i)/maximos(:,i);
end
V=A'; % Se guarda la traspuesta de los datos de las características ya
normalizadas
```

Como siguiente punto se configura la estructura de la RNA, en la investigación se utiliza una red neuronal multicapa feedforward backpropagation. El número de neuronas tanto de la capa de entrada como la capa de salida no se configuraron, puesto que su valor corresponde al número de entradas y salidas que se espera de la red (Marín y Arriojas, 2021; Rivas et al., 2018) y son generadas automáticamente.

El número de neuronas que se pueden configurar corresponden a las de las capas ocultas, puesto que no existe un modelo que garantice un número exacto de neuronas a utilizar (Contreras U. et al., 2019; Rivas et al., 2018) es por ello que su valor se modificó de manera manual de manera iterativa, asignando un valor de 80, 80 y 81 neuronas para la primera, segunda y tercera capa oculta respectivamente. Se asignó la tangente sigmoidea "tansig" como función de activación para las capas ocultas y la función "purelin" para la capa de salida consideranlo lo expuesto en la revisión literaria realizada por Tuan et al. (2021) y finalmente una línea de código para la visualización de la Red. Cabe recalcar que los pesos de las neuronas especificadas por Raschka y Mirjalili (2017) se generan automáticamente por el algoritmo. El código para el efecto, fue el siguiente:

```
hiddenLayerSizes = [80 80 81]; % Se asigna el número de neuronas de las capas
ocultas
red = feedforwardnet(hiddenLayerSizes, 'trainscg'); % Se crea la RNA
feedforward propagation
% Definir la función de activación de cada capa de la RNA
red.layers{1}.transferFcn = 'tansig'; % 1° capa oculta con función tansig
red.layers{2}.transferFcn = 'tansig'; % 2° capa oculta con función tansig
red.layers{3}.transferFcn = 'tansig'; % 3° capa oculta con función tansig
red.layers{end}.transferFcn = 'purelin'; % Capa de salida con función purelin

view(red); % Visualizar la red
```

La estructura de la red empleada se observa en la Figura 38 indicando la cantidad de datos para cada tratamiento las capas ocultas (Hidden Lawyer), la capa de salida (Output Lawyer) y el número de clases que va a clasificar la RNA, en este caso corresponden a los tratamientos 1, 2, 3 y 4.

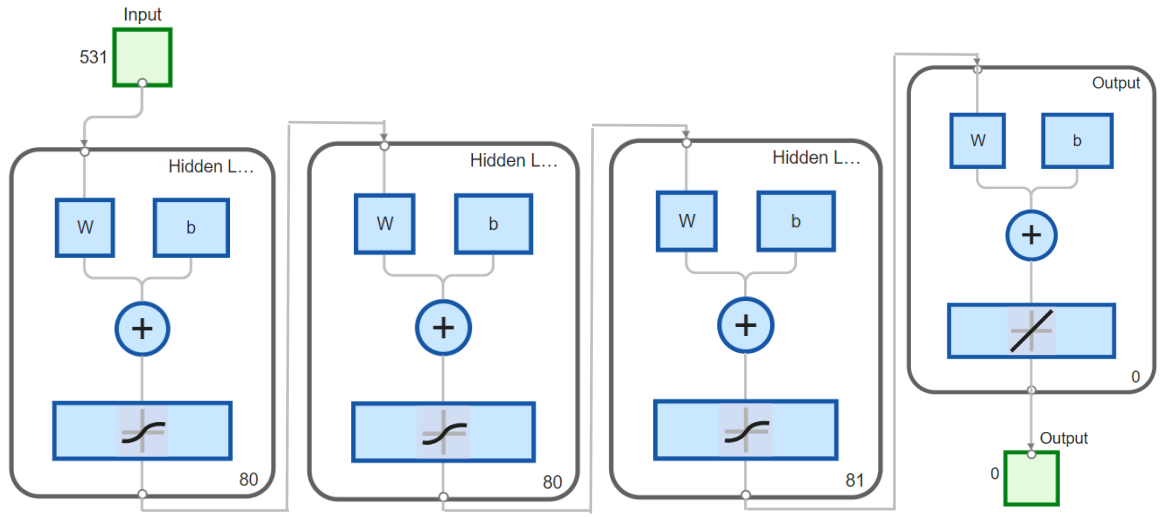


Figura 38. Estructura de la RNA utilizada.

Se dividieron aleatoriamente los datos en: 70% para entrenamiento, 15% para validación y 15% para probar la red entrenada referenciando la metodología empleada por Delgado (2018). Como función de aprendizaje u optimización se eligió la función “trainscg”. Se estableció un valor de 0 para el error mínimo y el gradiente con la finalidad de obtener el valor mínimo que puede alcanzar la RNA en 1000 épocas de entrenamiento, se asignó que se realicen hasta un valor máximo de 1000 iteraciones si el error no baja. Finalmente se estableció una línea para evitar el sobreajuste de la RNA. El código de la configuración de la RNA fue el siguiente:

```
% División de datos
red.divideFcn = 'dividerand'; % Divide los datos de forma aleatoria
red.divideParam.trainRatio = 0.7; % 70% para entrenamiento
red.divideParam.valRatio = 0.15;  % 15% para validación
red.divideParam.testRatio = 0.15; % 15% para prueba
red.performParam.regularization = 1e-6; % Regularización para evitar
sobreajuste

% Configuración del entrenamiento
red.trainFcn = 'trainscg';       % Función de optimización (Levenberg-
Marquardt)
red.trainParam.goal = 0;         % Valor deseado del error
red.trainParam.epochs = 1000;    % Número máximo de épocas
red.trainParam.max_fail = 1000;  % Máximo de fallos en validación
red.trainParam.min_grad = 0;     % Gradiente mínimo
```

Para iniciar el entrenamiento del algoritmo se realizó mediante la siguiente línea de código

```
[red, tr] = train(red, V, Clases); % Entrenar la red neuronal
```

Finalmente, para la visualización de la matriz de confusión y las métricas de rendimiento de la RNA, se emplea el siguiente código:

```
% Resultados de la red entrenada
outputs = red(V);                        % Obtiene las salidas de la red
perf = perform(red, Clases, outputs);    % Calcula el Error Cuadrático Medio
(MSE) de la red durante su entrenamiento, validación y prueba. Genera un valor
final

% Convierte las salidas de la red y las clases reales en variables categóricas
Tratamiento = vec2ind(Clases);          % Convierte las clases codificadas en
“One-Hot” en índices de clase (1, 2, 3 o 4)
Predicciones = vec2ind(outputs);    % Convierte las predicciones de la Red
codificadas como “One-Hot” en índices (1, 2, 3 o 4)

M_conf = confusionmat(Tratamiento, Predicciones); % Obtiene los valores de la
matriz de confusión
```

```
% Métricas del rendimiento de la red
numClasses = size(M_conf, 1);     % Determina el número de clases que tiene la
matriz de confusión
precision = zeros(1, numClasses); % Almacena la precisión de cada clase
recall = zeros(1, numClasses);    % Almacena el recall de cada clase
f1Score = zeros(1, numClasses);   % Almacena el F1 – Score de cada clase
for i = 1:numClasses            % Inicio del Bucle
    tp = M_conf(i, i);           % Determina la cantidad de verdaderos positivos
    fp = sum(M_conf(:, i)) - tp;  % Determina la cantidad de falsos positivos
    fn = sum(M_conf (i, :)) - tp; % Determina la cantidad de falsos negativos
    % Para los cálculos de la precisión, recall y F1-Score, se incluye el factor
    “eps”    (equivalente    a    aprox.    2.22e^16)    para    evitar    la
    indeterminación 0/0, en el caso de que tp, fp y fn sean 0. El valor es
despreciable
    precision(i) = tp / (tp + fp + eps); % Precisión para cada clase
    recall(i) = tp / (tp + fn + eps);    % Recall para cada clase
    f1Score(i) = 2 * (precision(i) * recall(i)) / (precision(i) + recall(i) +
eps); % F1-score para cada clase
end  % Fin del Bucle
% Calcular el promedio de todas las métricas
precisionMacro = mean(precision); % Precisión total del modelo
recallMacro = mean(recall);       % Recall total del modelo
f1Macro = mean(f1Score);          % F1-score total del modelo

% Se muestran los resultados, aunque se puede extraer su valor directamente
del Worspace de Matlab
fprintf('Precisión por clase: %s\n', mat2str(precision));
fprintf('Recall por clase: %s\n', mat2str(recall));
fprintf('F1-score por clase: %s\n', mat2str(f1Score));
fprintf('Precisión Macro: %.4f\n', precisionMacro);
fprintf('Recall Macro: %.4f\n', recallMacro);
fprintf('F1-score Macro: %.4f\n', f1Macro);

% Se grafica la matriz de confusión
figure;
confusionchart(confMat, {'Motor OK',...
    'Fallo de Bujía y Filtro', 'Fallo del Filtro', ...
    'Fallo de Bujía'}, 'Title', 'Matriz de Confusión', 'RowSummary', ...
    'row-normalized', 'ColumnSummary', 'column-normalized'); % Se modifican
los nombres de las clases y se generan los porcentajes de precisión de la red.
beep; % Indicador acústico del fin del entrenamiento de la red
```

Al ejecutar todo el código, se abre una ventana la cual permite observar el estado actual del entrenamiento de la RNA, permitiendo observar las gráficas del error cuadrático medio “Performance”, el estado de entrenamiento (Training State) y la regresión “Regression”. Adicionalmente indica el número de épocas del entrenamiento, el tiempo transcurrido, el error cuadrático medio, el gradiente y el número de validaciones realizadas (Ver Figura 39).

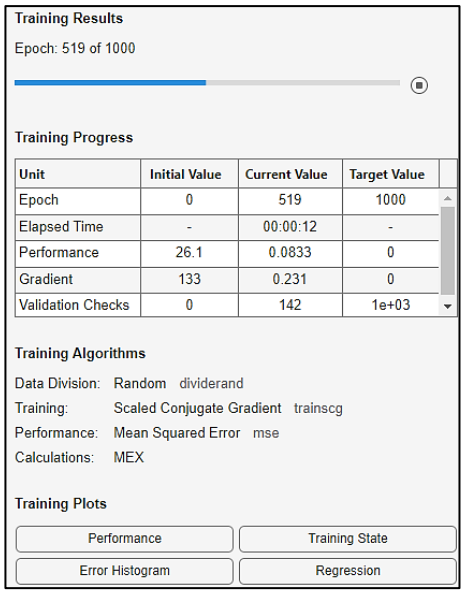


Figura 39. Ventana emergente del entrenamiento de la red.

5.3.4 Aplicación del algoritmo de SVM para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto

Para implementar el algoritmo de SVM se utilizó el toolbox de Matlab "Classification Learner" puesto que varias investigaciones (Biddle y Fallah, 2021; Cho et al., 2021; Delgado, 2018; Taylor et al., 2021) hacen uso de esta herramienta para implementar el SVM. La arquitectura para su ejecución fue similar a la RNA, puesto que, primeramente, se normalizaron los datos, luego se asignaron los datos de entrenamiento y los datos de respuesta, posteriormente se entrenó el modelo y al final se obtuvo la precisión del mismo. La Figura 40 indica el flujograma que se empleó para aplicar el modelo de SVM a los datos de las características del sensor MAP.

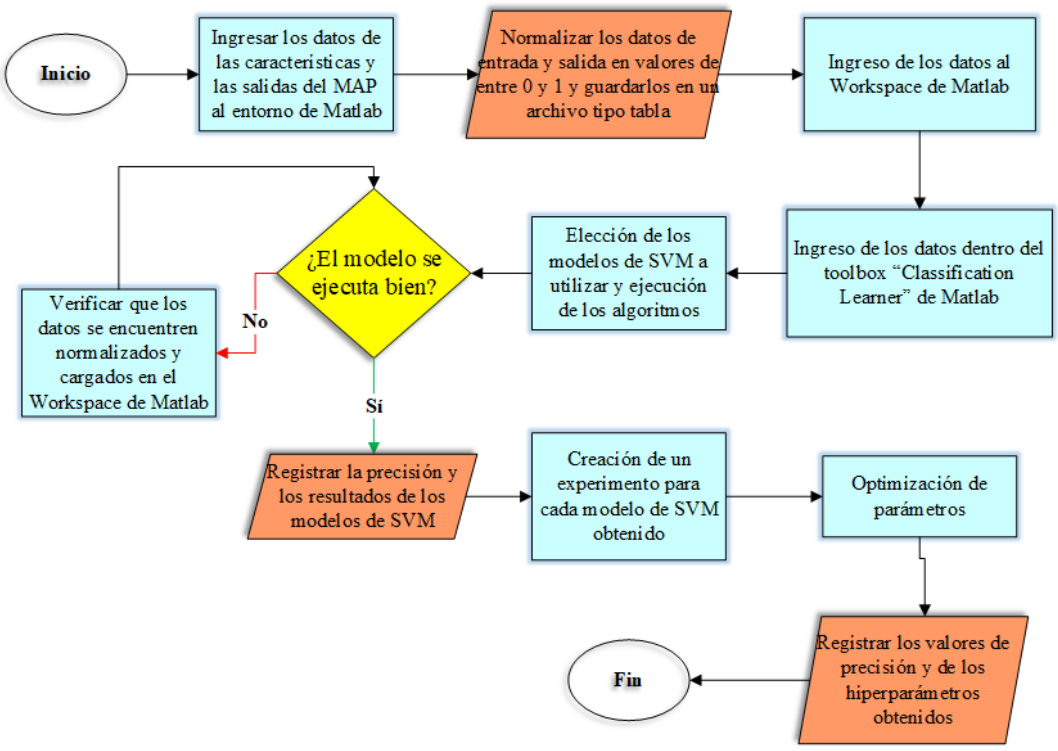


Figura 40. Flujograma de la implementación del SVM.

En primer lugar, se normalizaron los datos de entrada haciendo uso del siguiente código:

```
load MAP_Extrac_Caract_Final.mat; % Cargar los datos
% Normalizar los datos de entrada y salida para mejorar el entrenamiento
X = table2array(Atributos_MAP_Target); % Se convierte la tabla a un
archivo .mat
maximos=max(X); % Se encuentran los puntos máximos de cada característica
for i=1:(size(X,2)) % Bucle para normalización de los datos
    A(:,i)=X(:,i)/maximos(:,i); % Se normalizan los valores entre 0 y 1
end
MAP_Normalizado = array2table(A,"VariableNames", ...
{'RMS','Median','Mode','Max','Energy','Class'}); Se asigna el nombre a las
variables
save('MAP_Normalizado','MAP_Normalizado')
```

El archivo normalizado se guardó con el nombre de "MAP_Normalizado.mat".

Previa apertura del toolbox "Classification Learner" de Matlab, se carga el archivo de los datos normalizados "MAP_Normalizado" al Workspace de Matlab mediante el comando de "load('MAP_Normalizado.mat')". Posteriormente se abre el "Classification Learner" ubicado en el apartado de APPS en Matlab (Ver Figura 41).

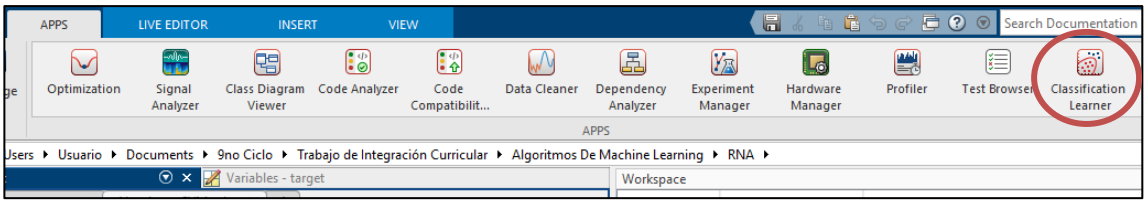
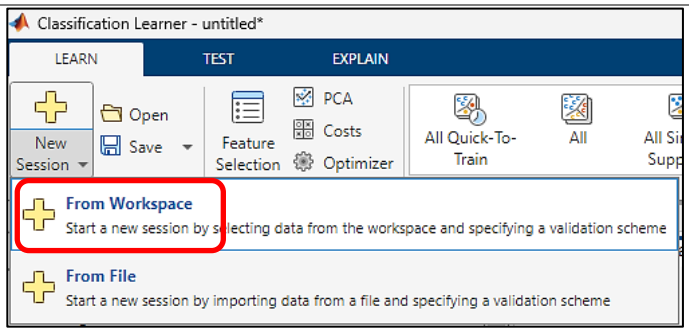
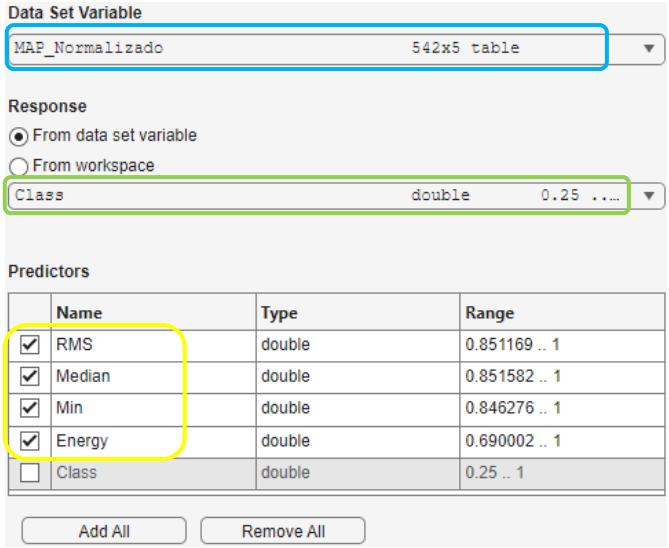
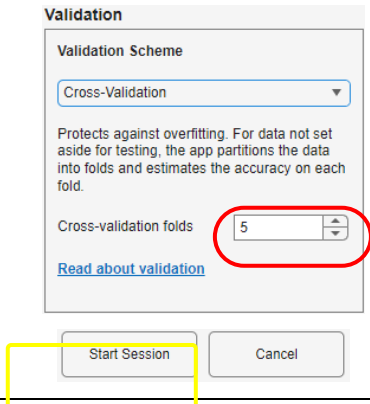
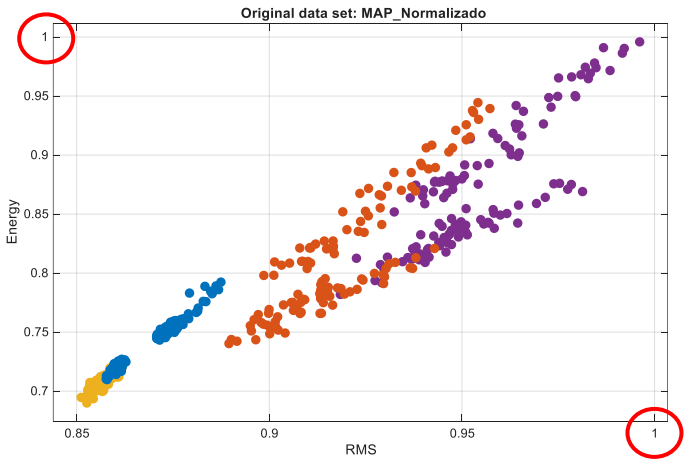
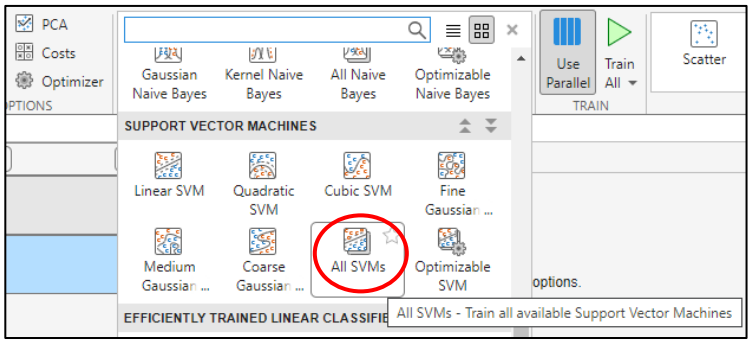
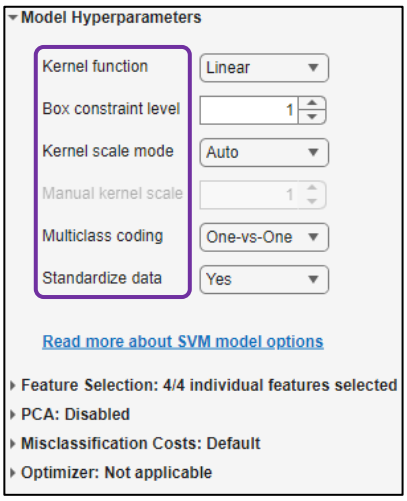
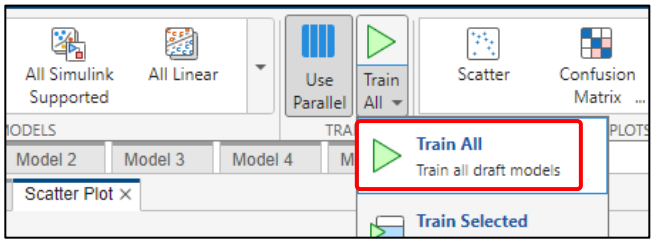


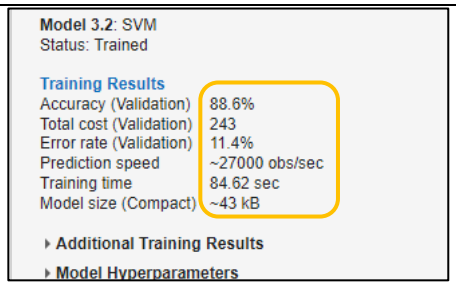
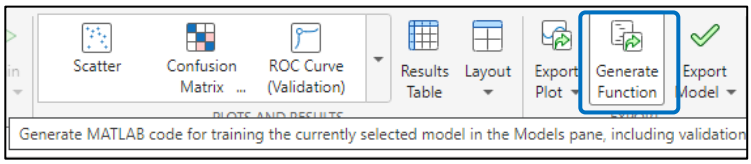
Figura 41. Ubicación del Classification Learner en Matlab.

Posteriormente, se sigue el proceso descrito en la Tabla 19.

Tabla 19. Uso del "Classification Learner" de Matlab.

Paso	Proceso	Imagen
1	Se crea una nueva sesión "New Session" y se selecciona la opción "From Workspace" indicando que los datos de entrada están cargados en el Workpace de Matlab.	
2	Automáticamente el programa asigna las características para entrenar el modelo (Data Set Variable), a todas las columnas de la tabla a excepción de la última, puesto que la última columna corresponde a la salida de los datos (Response). En el apartado "Predictors" se seleccionaron todas las características para entrenar el modelo.	
3	Se empleó la validación cruzada "Cross-Validation" con una partición de los datos en 5 carpetas diferentes. Finalmente se cargaron los datos dentro del Classification Learner mediante el "Start Session".	

Paso	Proceso	Imagen
4	Se observa la gráfica Scatter de los datos en 2 dimensiones, donde tanto el eje X e Y pueden tomar cualquier valor de las características. Se corrobora que los datos se encuentren normalizados, en donde, el valor máximo de ambos ejes no debe ser superior a la unidad ni tampoco inferior a 0.	
5	Se seleccionan los modelos a entrenar en la sección “Models” de la barra de opciones. En la investigación se seleccionaron varios modelos de clasificación para ser entrenados, dentro de ellos, todos los modelos SVM.	
6	Para el entrenamiento de los modelos SVM, se seleccionaron los modelos por defecto del programa, “SVM lineal” los cuales tuvieron diferentes características: Orden de la función Kernel, el parámetro C del SVM (Box constraint level), la escala del Kernel y la codificación del entrenamiento. Se asignó que los datos están estandarizados (Yes) puesto que se encuentran normalizados.	
8	Se entrenan todos los modelos de clasificación seleccionando la opción “Train All”.	

Paso	Proceso	Imagen
9	Se obtienen las características de: precisión (Accuracy), porcentaje de error (Error rate) y el tiempo de entrenamiento para cada modelo de SVM.	
9	Se exporta el algoritmo de SVM mediante la opción “Generate Function”.	

Al exportar la función generada, se obtiene el algoritmo base de los modelos de SVM con el que trabaja el Classification Learner, el mismo, se explica a continuación:

```
% Se ingresan los datos para el entrenamiento
inputTable = MAP_Normalizado; % Se carga la tabla normalizada de los atributos y las clases
predictorNames = {'RMS', 'Median', 'Min', 'Energy'}; % Se designa el nombre de las columnas de la tabla normalizada correspondientes a las características del MAP
predictors = inputTable(:, predictorNames); % Se seleccionan las columnas que previamente se designaron como datos de entrenamiento
response = inputTable.Class; % Selecciona como salida a la columna “Class” de la tabla normalizada cargada al principio
isCategoricalPredictor = [false, false, false, false]; % Indica que los datos de las características no son categóricos, si no, numéricos
classNames = [0.25; 0.5; 0.75; 1]; % Se designa el valor de las distintas clases, como los valores se encontraban normalizados, los valores 0.25; 0.5; 0.75 y 1 corresponden a los Tratamientos 1, 2, 3, y 4 respectivamente
% Train a classifier
% This code specifies all the classifier options and trains the classifier.
template = templateSVM(... % Se crea un modelo base de SVM
    'KernelFunction', 'polynomial', ... % Indica que la función tiene un grado de polinomio, “linear” corresponde a un modelo SVM lineal, “gaussian” corresponde a un SVM gausiano
    'PolynomialOrder', 3, ... % Se especifica el grado del polinomio kernel, [] indica que el modelo SVM es lineal o no tiene un orden (para el SVM gausiano), 2 indica que es cuadrático, 3 es cúbico
    'KernelScale', 'auto', ... % Ajusta automáticamente la escala del kernel, este valor se puede agregar manualmente. Para el kernel gausiano se tienen 3 categorías, la primera en donde el kernelScale es de 0.5, la segunda cuando su valor es de 2 y la tercera cuando su valor corresponde a 8
    'BoxConstraint', 1, ... % Por defecto, el software asigna el valor del BoxConstraint a 1
```

```
    'Standardize', true); % Indica que los datos de entrada están normalizados
para mejorar el rendimiento del SVM
classificationSVM = fitcecoc(... % Emplea el método Error-Conrrecting Output
Codes para ajustar el modelo SVM
    predictors, response, ... % Se asignan los datos de entrada y de salida
    'Learners', template, ... % Se carga el modelo base “template”
    'Coding', 'onevsone', ... % Se configura el código de entrenamiento que se
va a utilizar “uno vs uno” o “uno vs todos”
    'ClassNames', classNames); % Se asignan los nombres a las clases

predictorExtractionFcn = @(t) t(:, predictorNames); % Función para extraer los
predictores que se usarán para la validación, o en cuyo caso, para predicción
svmPredictFcn = @(x) predict(classificationSVM, x); % Función para asignar el
modelo entrenado
trainedClassifier.predictFcn = @(x)
svmPredictFcn(predictorExtractionFcn(x)); % Crea la función para predecir con
el SVM entrenado
trainedClassifier.classificationSVM = classificationSVM; % Se guarda el modelo
SVM entrenado

trainedClassifier.About = 'This struct is a trained model exported from
Classification Learner R2024b.'; % Indica la versión de Matlab que se utilizó

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.classificationSVM, 'KFold',
5); % Divide los datos en 5 partes, 4 utiliza para entrenar el modelo y 1
parte para la prueba

% Realiza la validación de las particiones
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError'); % Se calcula la precisión del modelo
```

Sin embargo, dentro de un modelo de SVM, lo más importante es encontrar los valores del parámetro C (denominado BoxConstraint dentro de Matlab) y la escala Kernel (denominado KernelScale dentro de Matlab) para lograr una mayor precisión, es por ello que el algoritmo se modificó para que en base al tipo de SVM que se utilice (ya sea lineal, cuadrático, polinomial o gaussiano) se encuentren los valores más óptimos del parámetro C y la escala Kernel.

Las líneas que se sustituyeron comprendieron al “template” y “classificationSVM”, manteniendo el mismo código al principio y al final. El código realiza 100 iteraciones en busca de los mejores parámetros del BoxConstraint y el KernelScale, el cual se explica a continuación:

```
template = templateSVM('KernelFunction', 'polynomial', 'PolynomialOrder', ...
2, 'Standardize', true); % Se designa el modelo SVM a entrenar: Para un SVM
lineal se sustituye “polynomial” por “linear” y “2” por “[]”; para un modelo
```

```
de gauss, se sustituye “polynomial” por “gaussian” y se elimina
“PolynomialOrder, 2,”.

classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template,  ... % Se carga el modelo SVM diseñado
        'Coding', 'onevsone', ... % Se desgina el método de aprendizaje
“onevsone” o “onevsall”
        'OptimizeHyperparameters', {'BoxConstraint', 'KernelScale'}, ... % Se
comanda al modelo para que optimice los parámetros “Boxconstraint” y
KernelScale”
        'HyperparameterOptimizationOptions', ...
        struct('UseParallel', true, 'ShowPlots', true, ...
               'MaxObjectiveEvaluations', 100)); % Se activa el parallel tool
para mejorar el rendimiento, se muestran las gráficas de la optimización y se
desgina el número de iteraciones a realizar
```

Para observar los resultados de la matriz de confusión, la precisión del modelo, el recall y el F1-Score, se implementó de manera adicional el siguiente código:

```
disp(validationAccuracy); % Imprime el valor de la precisión del modelo que
brinda por defecto el Classification Learner
% Se calcula y se grafica la Matriz de Confusión
confMat = confusionmat(response, validationPredictions);
figure;
confusionchart(confMat, {'Motor OK',...
    'Fallo de Bujía y Filtro', 'Fallo del Filtro', ...
    'Fallo de Bujía'}, 'Title', 'Matriz de Confusión', 'RowSummary', ...
    'row-normalized', 'ColumnSummary', 'column-normalized'); % Se asigna el
nombre a los tratamientos, para este caso, se utilizó el nombre de los estados
del motor

% Calcular métricas de desempeño
precision = diag(confMat) ./ sum(confMat, 1)'; % VP / (VP + FP)
recall = diag(confMat) ./ sum(confMat, 2);     % VP / (VP + FN)
F1 = 2 * (precision .* recall) ./ (precision + recall); % F1-Score
% Precisión General
accuracy = sum(diag(confMat)) / sum(confMat(:));
% Mostrar métricas
disp('Precisión por clase:'); disp(precision);
disp('Recall por clase:'); disp(recall);
disp('F1-Score por clase:'); disp(F1);
disp(['Precisión General: ', num2str(accuracy * 100), '%']);
% Calcular métricas promedio
precisionMacro = mean(precision)*100; % Precisión macro
recallMacro = mean(recall)*100;       % Recall macro
F1Macro = mean(F1)*100;          % F1-score macro
fprintf('Precisión Macro: %.4f\n', precisionMacro);
fprintf('Recall Macro: %.4f\n', recallMacro);
fprintf('F1-score Macro: %.4f\n', F1Macro);
```

```
% Calcular errores
errores = (response ~= validationPredictions);
% Graficar el histograma del error
errors = validationPredictions ~= response;
figure;
histogram(errors, 'BinWidth', 0.5);
title('Histograma del error de clasificación');
xlabel('Error (0 = Correcto, 1 = Incorrecto)');
ylabel('Frecuencia');
```

Se realizó el mismo proceso para con las características obtenidas a partir de los datos del MAP que fueron filtradas mediante el filtro Butterworth de 5to orden.

5.3.5 Aplicación de diferentes modelos de Machine Learning mediante Classification Learner de Matlab para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto

Se empleó el mismo archivo de los datos normalizados "MAP_Normalizado.mat" que fue utilizado para el entrenamiento del modelo de SVM, el cual correspondió a los datos de entrenamiento y de salida para el Classification Learner. Se siguieron los mismos pasos del 1 al 4 descritos en la Tabla 19, se seleccionó la opción "All" para entrenar todos los modelos de clasificación base con los que cuenta el Toolbox Classification Learner y finalmente se entrenó todo el modelo con "Train All" (Ver Figura 42).

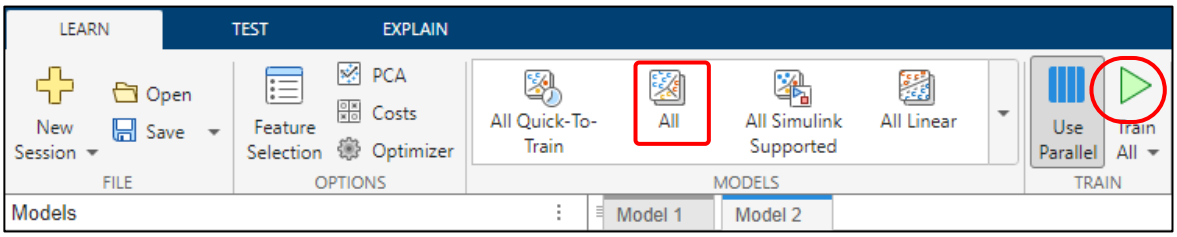


Figura 42. Selección de todos los modelos de Machine Learning de clasificación.

Luego del entrenamiento de cada tipo de algoritmo, se seleccionaron sus modelos optimizables el cual se configuró para que se realicen 100 iteraciones buscando mejorar la precisión del algoritmo base que tuvo un menor error (mayor precisión) y se procedió a su entrenamiento.

Se realizó el mismo proceso para con las características obtenidas a partir de los datos del MAP que fueron filtradas mediante el filtro Butterworth de 5to orden.

5.3.6 Análisis estadístico de los algoritmos de Machine Learning

Se tomó en cuenta los datos obtenidos a partir de la matriz de confusión, puesto que a partir de ella se puede aplicar el análisis estadístico de los algoritmos mediante la precisión, recall y la puntuación F1 tal como lo plantearon Li et al. (2023). Las ecuaciones empleadas se explicaron en el apartado teórico de la sección 4.12.

La matriz de confusión también permitió evaluar el desempeño del algoritmo y, por ende, validar que la precisión que indica el algoritmo es correcta.

El código para extraer los datos de la matriz de confusión, ya se incluye dentro de los algoritmos, la cual es:

```
M_conf = confusionmat(Tratamiento, Predicciones); % Obtiene los valores de la
matriz de confusión
```

El código para generar la matriz de confusión, igualmente, ya se incluye dentro de los algoritmos de RNA y SVM, dicha línea de código es:

```
figure;
confusionchart(confMat, {'Motor OK','Fallo de Bujía y Filtro', ...
 'Fallo del Filtro','Fallo de Bujía'}, 'Title', 'Matriz de Confusión', ...
'RowSummary','row-normalized', 'ColumnSummary', 'column-normalized'); % Se
modifican los nombres de las clases y se generan los porcentajes de precisión
y recall de la red.
```

El Toolbox denominado “Classification Learner” dentro de Matlab, calcula automáticamente los valores de precisión, recall, y F1-Score, de igual manera, en base a los datos de la matriz de confusión que obtiene el Toolbox.

6 Resultados

6.1 Resultados de clasificación mediante RNA

El algoritmo de RNA utilizado arrojó distintos resultados: gráficas de error cuadrático medio (ECM o MSE), variación del gradiente, comprobaciones de validación, la regresión obtenida y el histograma de error. Adicionalmente, el algoritmo brindó información de los valores iniciales y finales en el entrenamiento y el tiempo de entrenamiento de la red. Los resultados del algoritmo con la configuración descrita en la metodología se redactan a continuación.

Al inicio del entrenamiento de la red, la performance (MSE) total del modelo fue de 26.1 mientras que para la época 1000 fue de 0.0605, lo que indica que el modelo disminuyó el error significativamente. La Figura 43 indica que el menor valor obtenido para los datos de validación fue de 0.26851 y correspondió a la época número 377.

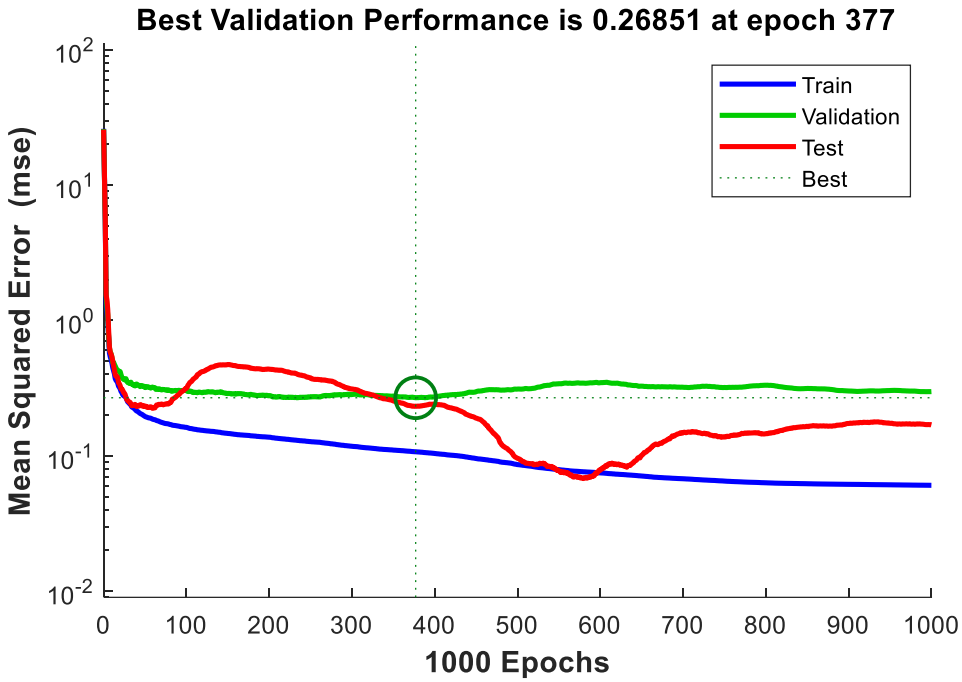


Figura 43. MSE del entrenamiento, validación y prueba de la red.

El gradiente obtenido en la época 1000 (Ver Figura 44), indica que la variación de los pesos de las neuronas fue de 0.033861. Al comienzo del entrenamiento, el valor del gradiente fue de 133.

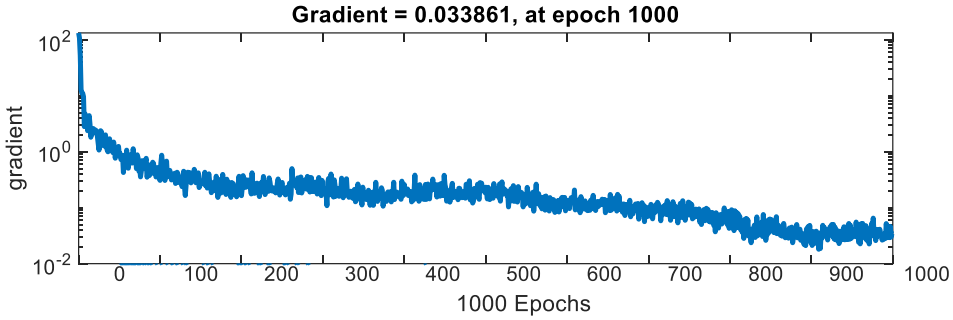


Figura 44. Gradiente y número de validaciones a lo largo del entrenamiento.

En la Figura 45 se indica que el error de validación no ha mejorado durante 623 épocas consecutivas hasta la época 1000.

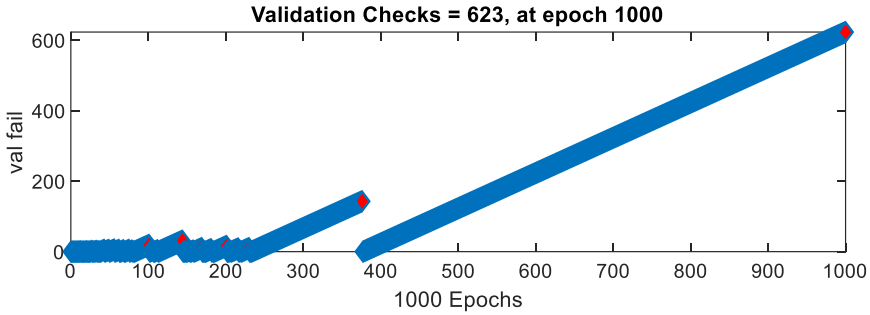


Figura 45. Gradiente y número de validaciones a lo largo del entrenamiento.

En la Figura 46 se observan los resultados de la regresión al final de las 1000 épocas, cuyo valor para el entrenamiento, validación y prueba corresponden a 0.963; 0.90693 y 0.92546 respectivamente. El valor general de la regresión corresponde a 0.9486.

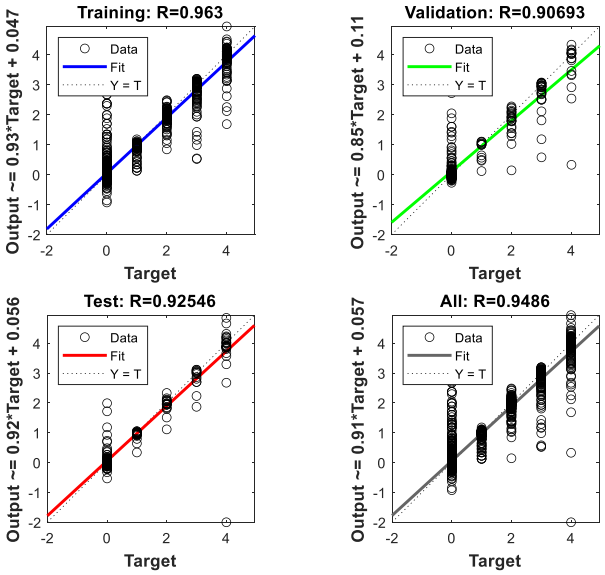


Figura 46. Resultados del gradiente de la RNA.

El histograma de error para el entrenamiento, validación y test se encuentran en un valor de 0.06387 y se indica en la Figura 47.

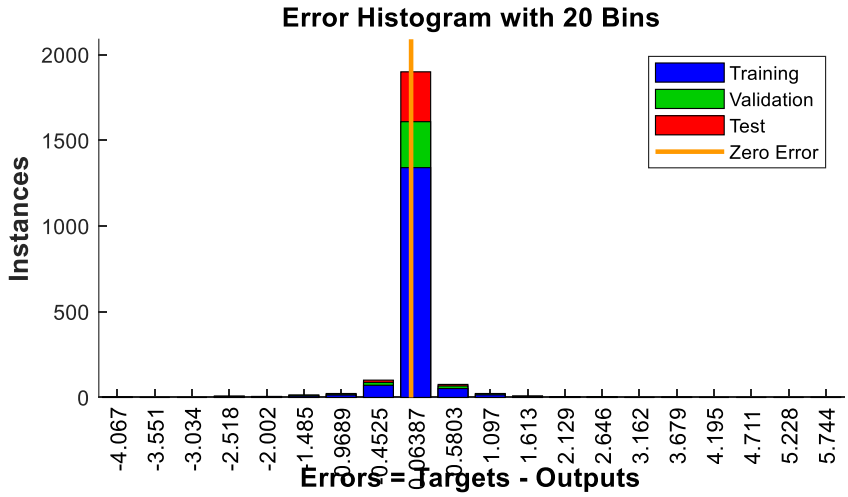


Figura 47. Histograma de error para los datos de entrenamiento, validación y prueba.

6.1.1 Validación del algoritmo de RNA

Para la validación del algoritmo se hizo uso de los resultados de clasificación obtenidos por la matriz de confusión de la Figura 48.

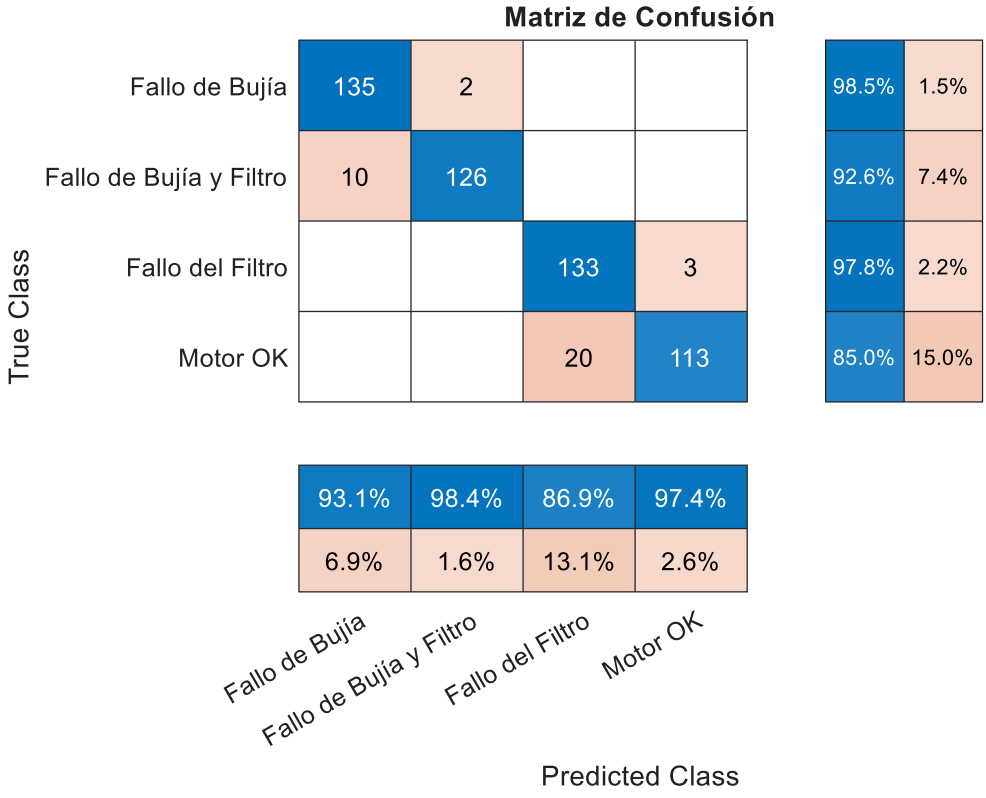


Figura 48. Matriz de confusión producto del algoritmo de la RNA.

La matriz de confusión indica que: de los 133 datos correspondientes al tratamiento 1 (Motor OK) que ingresaron para ser clasificados por la red, 113 fueron clasificados correctamente mientras que 20 de ellos fueron clasificados erróneamente dentro del tratamiento 3 (Fallo del filtro). Adicionalmente, de los 116 datos que la red clasificó como "Moto OK", 113 de ellos estuvieron bien clasificados y 3 fueron erróneos. De manera que también da a conocer los verdaderos positivos (VP), los cuales se ubican a lo largo de la diagonal de la matriz de confusión, para la clase 1 corresponde a 113; los falsos positivos (FP) se ubican en las columnas y corresponden al número de datos que el algoritmo clasificó como variables del tratamiento 1. Los falsos negativos (FN) por su parte se encuentran en las filas, correspondiendo a la cantidad de datos reales de entrada y su clasificación según lo entendió el algoritmo. Para los demás tratamientos, se realiza el mismo análisis.

Los porcentajes a la derecha indican el porcentaje de precisión y error del recall de cada clase de la red, por ejemplo, el tratamiento 1 (Motor OK) posee un recall del 85% con un error del 15%. Por su parte, los datos de la parte inferior corresponden a la precisión y el error de la RNA, en el caso del tratamiento 2 (Fallo de Bujía y Filtro) se obtuvo una precisión del 86.9% y un error del 13.1%. Cabe recalcar que los datos de dicha matriz están aproximados.

6.1.2 Precisión del algoritmo de RNA

Los valores de precisión obtenidos se resumen en la Tabla 20.

Tabla 20. Precisión obtenida del algoritmo de RNA.

Atributo	Magnitud	Atributo	Magnitud
Precisión T1	97.413 %	***F1 – Score T1***	90.763 %
Precisión T2	98.437 %	***F1 – Score T2***	95.454 %
Precisión T3	86.928 %	***F1 – Score T3***	92.041 %
Precisión T4	93.103 %	***F1 – Score T4***	98.745 %
Recall T1	84.962 %	***Precisión Total***	93.97 %
Recall T2	92.647 %	***Recall total***	93.49 %
Recall T3	97.794 %	***F1 – Score total***	93.5 %
Recall T4	98.54 %	***Tiempo de entrenamiento***	14 segundos

Nota. T1, T2, T3 y T4 hacen referencia a los tratamientos 1, 2, 3 y 4 respectivamente.

Además, se comparó el desempeño del algoritmo de RNA al utilizar distintas funciones de aprendizaje (Ver Tabla 21).

Tabla 21. Resultados de precisión de diferentes funciones de aprendizaje.

N	Función de aprendizaje	Precisión (%)	Recall (%)	F1-score (%)	Tiempo (s)
0	trainscg	93.97	93.49	93.5	14
1	trainb	88.34	86.60	86.4	15
2	trainbfg	43.61	35.01	38.74	737 (época 5)
3	traincgf	95.68	95.13	95.14	33
4	traincgp	95.13	94.39	94.4	26
5	traingd	91.29	89.75	89.7	14
6	traingda	90.82	89.2	89.15	9
7	traingdm	88.34	86.41	86.15	9
8	traingdx	91.94	90.30	90.25	8
9	trainlm	93.25	91.77	91.72	235 (época 5)
10	trainoss	93.07	92.18	92.19	31
11	trainrp	94.05	93.67	93.69	9
12	trains	88.12	86.23	85.98	16

Nota. El tiempo y el gasto computacional de entrenamiento de la RNA al utilizar las funciones de aprendizaje "trainbfg" y "trainlm" fue muy elevado, por lo cual se registró únicamente el tiempo transcurrido durante 5 épocas de entrenamiento (época 5).

En la Tabla 22, se compara el desempeño de la RNA al modificar el número de neuronas de las capas ocultas, se comparó el tiempo de entrenamiento y el histograma del error.

Tabla 22. Resultados de la modificación del número de neuronas diferentes funciones de aprendizaje.

N	Número de neuronas (capas ocultas)	Precisión (%)	Histograma de error	Tiempo (s)
0	80 80 81	93.97	0.06387	14
1	80 80 80	94.97	0.1768	16
3	80 160 81	95.12	0.08393	31
4	80 160 80	94.04	0.1066	23
5	80 80 162	94.99	0.05521	19
7	80 80 158	94.92	0.05321	17
8	80 80 165	94.79	-0.08472	21
9	80 80 180	91.41	-0.2022 y 0.2049	18

N	Número de neuronas (capas ocultas)	Precisión (%)	Histograma de error	Tiempo (s)
10	40 40 79	94.70	0.03971	9
11	40 40 81	95.68	0.007428	9
12	30 30 61	93.83	0.1195	7
13	160 80 81	92.65	0.1451 y -0.2006	20
14	160 80 80	96.05	-0.08657	17
15	40 80 162	91.08	0.1217	17
16	160 80 40	95.35	0.08827	15

6.1.3 Precisión del algoritmo de RNA entrenado con los datos filtrados

Finalmente se entrenó el algoritmo de RNA al ingresar los datos del sensor MAP con su ruido reducido mediante el filtro Butterworth (Sección 5.3.9.1). La comparación se resume en la Tabla 23.

Tabla 23. Precisión obtenida del algoritmo de RNA datos filtados.

Atributo	Magnitud	Atributo	Magnitud
Precisión T1	99.115 %	***Precisión Total***	95.53 %
Precisión T2	100 %	***Recall total***	94.95 %
Precisión T3	86.538 %	***F1 – Score total***	94.96 %
Precisión T4	96.478 %	***Gradiente (época 1000)***	0.060048
Recall T1	84.21 %	***MSE (época 1000)***	0.0682
Recall T2	96.323 %	***Histograma de error***	0.05613
Recall T3	99.264 %	***Regresión (Entrenamiento)***	0.96955
Recall T4	100 %	***Regresión (Validación)***	0.97795
F1 – Score T1	91.056 %	***Regresión (Prueba)***	0.95402
F1 – Score T2	98.127 %	***Regresión (Total)***	0.96839
F1 – Score T3	92.465 %	***Tiempo de entrenamiento***	13 segundos
F1 – Score T4	98.207 %	***Épocas de validación del error***	413

6.1.4 Resumen de resultados y comparación de las distintas configuraciones del algoritmo de RNA

Para comparar las distintas configuraciones realizadas al algoritmo, se consideró la precisión total, el tiempo de entrenamiento y el error del histograma resumidos en las tablas 20,

21, 22 y 23. Los valores arrojados por el algoritmo descrito en la metodología correspondieron al 93.97%, 14 segundos y 0.06387 respectivamente.

Al comparar distintas funciones de aprendizaje, se obtuvo que en una mayor precisión fueron las funciones “traincgf”, “traincgp” y “trainrp” cuya precisión fue de 95.68%, 95.13% y 94.05% respectivamente, y el tiempo de entrenamiento fue de 33, 26 y 9 segundos respectivamente.

Al realizar la modificación del número de neuronas de las capas ocultas para el algoritmo de RNA, se obtuvo que tanto la configuración propuesta: 40 40 81 $[(n)\ (n)\ (2n+1)]$ como la configuración 160 80 80 $[2n\ n\ n]$ presentaron una precisión del 95.68% y del 96.05% respectivamente, un valor del error en el histograma de 0.007428 y -0.08657; y, un tiempo de entrenamiento de 9 y 17 segundos respectivamente.

El utilizar los datos filtrados para entrenar el algoritmo inicial de RNA, este mejoró la precisión al 95.53%, disminuyó el tiempo de entrenamiento a 13 segundos y el error del histograma a 0.05613. Por lo cual se prevé que su precisión en general, es más alta.

De manera adicional, las tablas 21 y 24 indican que el algoritmo de RNA tiene una mayor precisión para clasificar el tratamiento 2 (Fallo de bujía y filtro), seguido del tratamiento 1 (Motor OK), el tratamiento 4 (Fallo de bujía) y finalmente el tratamiento 3 (Fallo del filtro de aire).

6.2 Resultados de clasificación mediante SVM

Producto de la ejecución el código, se obtuvo una gráfica del error mínimo objetivo vs el número de iteraciones realizadas comparando los valores obtenidos y los valores estimados, tal como se indica en la Figura 49. La figura indica los resultados obtenidos por el algoritmo optimizable de los factores del parámetro C y la escala Kernel, al entrenar un SVM cuadrático con codificación “onevsone” durante 100 épocas. Se puede apreciar que, con cada iteración, el algoritmo de SVM se ajusta para lograr un menor error, en donde, el error es menor a partir de la iteración 57.

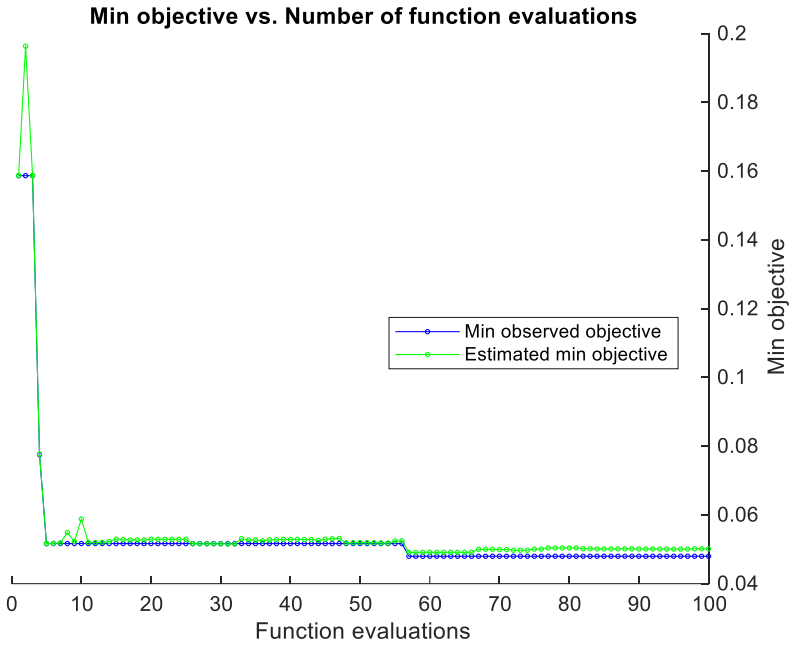


Figura 49. Gráfica del error según el número de iteraciones del modelo.

Adicionalmente, el algoritmo permite observar el resultado del hiperplano ajustado por el SVM para la clasificación de los parámetros, en la Figura 50 se puede apreciar una representación tridimensional del hiperplano producto de un modelo de SVM cuadrático. En los Anexos 7, 8 y 9 se representan los hiperplanos obtenidos para los demás modelos de SVM.

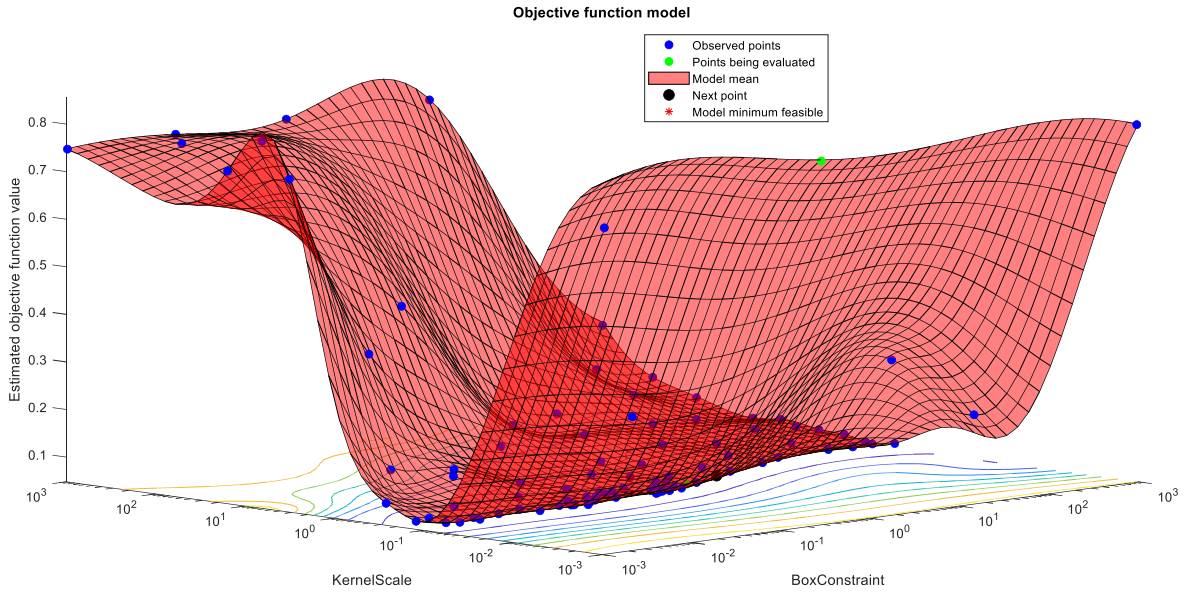


Figura 50. Hiperplano de SVM cuadrático.

6.2.1 Validación del algoritmo de SVM

Se valida la precisión de los modelos mediante la matriz de confusión producto del código. La Figura 51 corresponde al resultado del algoritmo de SVM cuadrático optimizable entrenado con codificación “onevsone”.

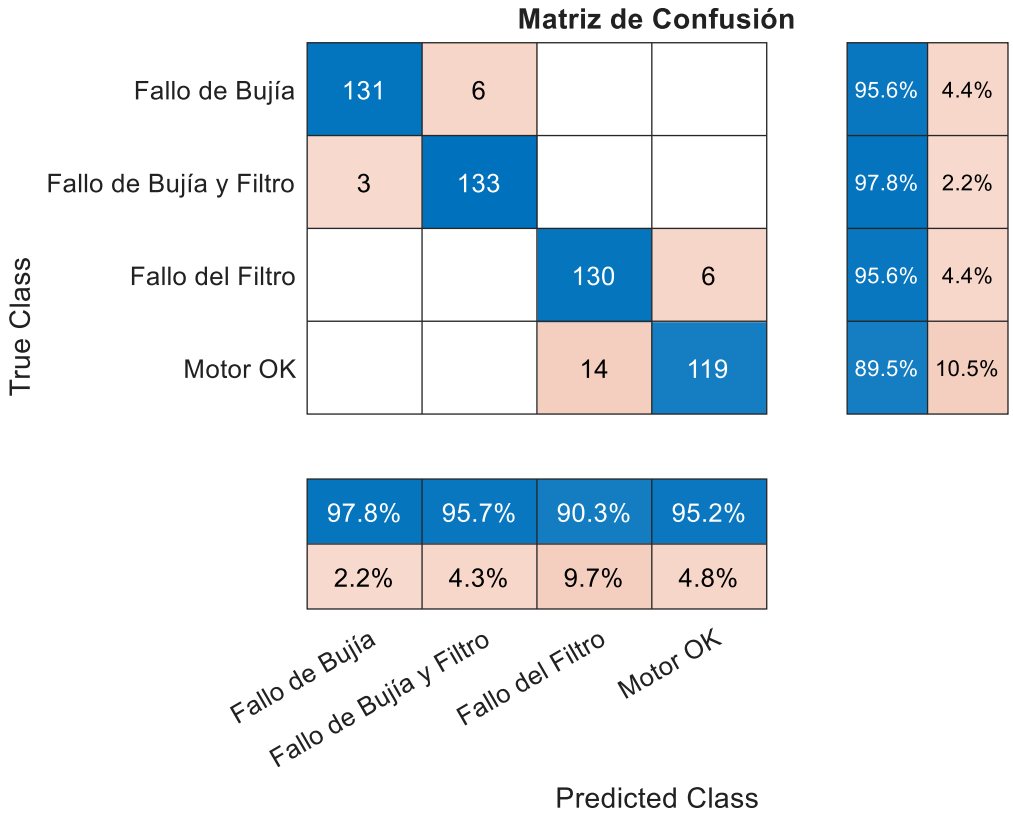


Figura 51. Matriz de confusión, del algoritmo de SVM cuadrático.

La matriz de confusión resultante del SVM, se interpreta de la misma manera que la matriz de confusión de la RNA. Se valida la precisión del algoritmo, puesto que de los 542 datos que ingresan para ser clasificados (suma total del número de datos), 513 (datos de color azul) fueron clasificados correctamente, lo cual corresponde al 94.6494%.

Los modelos realizados con el Classification Learner también se validaron de igual manera, puesto que el programa exporta la matriz de confusión para cada modelo.

6.2.2 Precisión del algoritmo de SVM

Los resultados de la precisión de los distintos tipos de SVM entrenados mediante el Classification Learner se describen en la Tabla 24, en donde el modelo con una mayor precisión correspondió al SVM optimizable (SVM Optimizable).

Tabla 24. Resultados de los modelos SVM del Classification Learner.

Función Kernel	Precisión	Recall	F1-Score	Tiempo (s)
Lineal (1)	90.6 %	90.5 %	90.6 %	15.541
Cuadrático (2)	94.1 %	94.1 %	94.1%	13.683
Cúbico (3)	94.5 %	94.4 %	94.5 %	12.807
Gaussiano (KS = 0.5)	93.5 %	93.3 %	93.3 %	11.53
Gaussiano (KS = 2)	92.3 %	92.2 %	91.9 %	10.565
Gaussiano (KS = 8)	88.4 %	87.0 %	83.9 %	9.4726
SVM Optimizable	95.0 %	95.0 %	95.0 %	137.01

Nota. Los valores (1), (2) y (3) representan el orden de la función polinomial

El SVM optimizable con el 95% de precisión correspondió al SVM cuadrático con escala Kernel 1, parámetro C igual a 998.2029 y la codificación multiclase “onevsone”.

En la Tabla 25 se registraron los resultados obtenidos del algoritmo de optimización de hiperparámetros el cual, se realizó durante 100 iteraciones con la codificación “onevsone”.

Tabla 25. Resultados del algoritmo de optimización de hiperparámetros con codificación “onevsone”.

Función Kernel	Parámetro C (Box constraint)	Escala Kernel (Kernel scale)	Precisión	Iteración	Tiempo (s)
Lineal (1)	0.058583	0.0010075	91.8819	89	78.001
Cuadrático (2)	0.6671	0.16515	94.6494 %	57	180
Cúbico (3)	132.54	1.8002	93.9114 %	24	94.5384
Gaussiano	339.25	2.2154	93.5424 %	36	35.1699

En la Tabla 26 se registran los resultados del algoritmo de optimización de hiperparámetros con la codificación “onevsall”.

Tabla 26. Resultados del algoritmo de optimización de hiperparámetros con codificación “onevsall”.

Función Kernel	Parámetro C (Box constraint)	Escala Kernel (Kernel scale)	Precisión (%)	Iteración	Tiempo (s)
Lineal (1)	94.56	0.60296	88.5609	27	129.8755
Cuadrático (2)	0.019437	0.044203	93.9114	50	323.1261
Cúbico (3)	11.536	0.78764	94.8339	71	227.187
Gaussiano	964.77	1.5816	94.6494	20	70.398

6.2.3 Precisión del algoritmo de SVM entrenado con los datos filtrados

Al entrenar el SVM con los datos filtrados de la sección 5.3.9.1 empleando el Classification Learner y el algoritmo de optimización del hiperparámetros con las codificaciones “onevsone” y “onevsall” se obtuvieron las tablas 27, 28 y 29.

Tabla 27. Resultados de los modelos SVM del Classification Learner entrenando con los datos filtrados.

Función Kernel	Precisión	Recall	F1-Score	Tiempo (s)
Lineal (1)	89.4 %	88.3 %	88.3 %	14.726
Cuadrático (2)	94.5 %	94.4 %	94.4 %	5.4884
Cúbico (3)	94.3 %	94.2 %	94.3 %	13.273
Gaussiano (KS = 0.5)	93.8 %	93.7%	93.7 %	12.051
Gaussiano (KS = 2)	91.7 %	91.5 %	91.5 %	11.325
Gaussiano (KS = 8)	88.4 %	87 %	86.9 %	10.474
SVM Optimizable	95.5 %	95.4 %	95.4 %	603.57

Nota. Los valores (1), (2) y (3) representan el orden de la función polinomial

Tabla 28. Algoritmo de optimización de hiperparámetros “onevsone” entrenado con los datos filtrados.

Función Kernel	Parámetro C (Box constraint)	Escala Kernel (Kernel scale)	Precisión	Iteración	Tiempo (s)
Lineal (1)	1.5295	0.025271	91.3284 %	20	128.7684
Cuadrático (2)	0.73363	0.49329	94.4649 %	12	154.3919
Cúbico (3)	0.33597	0.9997	94.2804 %	75	95.7713
Gaussiano	0.61826	0.26078	93.5424 %	48	50.4541

Tabla 29. Algoritmo de optimización de hiperparámetros “onevsall” entrenado con los datos filtrados.

Función Kernel	Parámetro C (Box constraint)	Escala Kernel (Kernel scale)	Precisión (%)	Iteración	Tiempo (s)
Lineal (1)	0.7588	0.10561	88.7454 %	25	138.086
Cuadrático (2)	0.35381	0.26585	95.3875 %	27	142.4452
Cúbico (3)	0.16583	0.58329	94.8339 %	83	245.4173
Gaussiano	33.823	1.8405	94.0959 %	37	46.0788

6.2.4 Resumen de resultados y comparación de las distintas configuraciones del algoritmo de SVM

El modelo del optimizable SVM del Classification Learner tuvo una mayor precisión, que fue del 95% y un tiempo de entrenamiento de 137.01 segundos. Los siguientes modelos con el porcentaje de precisión más alta correspondieron al SVM cuadrático y cúbico, cuyos valores de precisión fueron del 94.1% y 94.5% y su tiempo de entrenamiento fue de 12.807 y 13.683 respectivamente.

Al realizar el entrenamiento del SVM con codificación "onevsone", tanto los algoritmos de SVM lineal y cuadrático aumentaron su precisión, mientras que el SVM cúbico y gaussiano disminuyeron su precisión. Sin embargo, sucedió todo lo contrario al aplicar la codificación "onevsall". La mejor configuración fuera del SVM optimizable, fue el SVM cúbico con entrenamiento "onevsall" cuya precisión fue del 94.8339% correspondiente a la iteración 71 de 100 y con un tiempo de 227.187 segundos para realizar las 100 iteraciones.

Al comparar los resultados del entrenamiento con codificación "onevsone" y "onevsall" se puede observar que los tiempos de entrenamiento cuando se aplica el código "onevsall" es mayor a su contraparte, adicionalmente, sin embargo, la precisión máxima que alcanza esta codificación es 0.1845% mayor a la precisión de su contraparte.

Al entrenar los modelos de SVM con los datos filtrados, dentro de los modelos base del Classification Learner la precisión de los SVM aumentó, en donde, el SVM optimizable obtuvo un valor del 95.5% de precisión, seguido del SVM cuadrático con el 94.5% y el SVM cúbico con el 94.3%.

Para el caso del entrenamiento con los datos filtrados, al realizar el entrenamiento del SVM con codificación "onevsone", tanto los algoritmos de SVM lineal y cuadrático aumentaron su precisión, mientras que el SVM cúbico y gaussiano disminuyeron su precisión. Al aplicar la codificación "onevsall" los algoritmos SVM cuadrático y cúbico aumentaron su precisión, mientras que el SVM lineal y gaussiano disminuyeron su precisión. La mejor configuración fuera del algoritmo de ajuste del kernel y el parámetro C, fue el SVM cuadrático con entrenamiento "onevsall" cuya precisión fue del 95.3875%, encontrado en la iteración 27 de 100, y cuyo tiempo total de optimización para las 100 épocas fue de 142.4452 segundos.

6.3 Resultados de clasificación de distintos modelos de Machine Learning mediante el uso del Classification Learner

Luego de entrenar todos los modelos disponibles de Machine Learning (ML) junto con sus pertinentes algoritmos optimizables, se eligieron los dos submodelos con mayor precisión de cada modelo de ML, los cuales se resumen en la Tabla 30.

Tabla 30. Resultados de los modelos SVM del Classification Learner.

Modelos de ML	Tipo	Precisión (%)	Recall (%)	F1-Score (%)	Tiempo (s)
Árboles de decisión	Coarse Tree (splits: 4)	90.4	90.2	90.2	3.9378
	Optimizable Tree	91.2	91.2	91.2	38.093
Análisis discriminante	Quadratic discriminant (full)	91.6	91.3	91.3	2.0347
	Optimizable discriminant	91.6	91.3	91.3	46.262
Algoritmos eficientes	Efficient Logistic Regression (Auto)	79	79	78.7	7.1646
	Optimizable Efficient Linear	79.5	79.0	78.7	100.98
Naive Bayes	Kernel Naive Bayes (Kernel Gaussiano)	89.7	89.1	89.1	7.8289
	Optimizable Naive Bayes	89.7	89.1	89.1	58.233
SVM	Cubic SVM (BC=1)	94.5	94.4	94.5	8.0949
	Optimizable SVM	95.0	95.0	95.0	137.01
K-vecinos más cercanos	Cubic KNN (Neighbors: 10)	92.3	92.3	92.3	1.7881
	Optimizable KNN	92.3	92.3	92.2	55.635
Modelos de ensamblajes	Ensemble Bagged Trees (Splits: 541, learners:30)	91.2	91.1	91.1	10.938
	Optimizable Ensemble	92.1	92.1	92.1	257.23
RNA	Wide Neural Network (1° cap: 100 neuronas)	93.1	93	93	25.378
	Optimizable Neural Network	92.3	92.2	92.2	435.49

De los modelos de Machine Learning utilizados, se puede rescatar que todos los modelos optimizables, a excepción del modelo optimizable de redes neuronales, resultan en un

valor de precisión mayor a sus modelos base, sin embargo, el tiempo de procesamiento de los mismos es considerablemente alto y el aumento de la precisión no es superior al 0.9% para el mejor de los casos, que corresponde a los modelos ensamblados "Ensemble models", por lo que se considera oportuno, utilizar los parámetros empleados en los modelos base dentro del modelo optimizable, puesto que permite encontrar el modelo más preciso para luego mejorar su precisión mediante el "Optimizable".

Se realizó el mismo proceso para con los datos filtrados, dando como resultado los valores de la Tabla 31.

Tabla 31. Resultados de los modelos SVM del Classification Learner con los datos filtrados.

Modelos de ML	Método de clasificación	Precisión (%)	Recall (%)	F1-Score (%)	Tiempo (s)
Árboles de decisión	*Medium Tree*	90.3	90.2	90.2	10.604
	Optimizable Tree	90.3	90.2	90.2	144.93
Análisis discriminante	*Quadratic discriminant (full)*	89.9	89.6	89.7	4.1886
	Optimizable discriminant	89.9	89.6	89.7	2.4727
Algoritmos eficientes	*Efficient Logistic Regression (Auto)*	80.7	79.9	79.7	8.2864
	Optimizable Efficient	80.7	79.9	79.7	361.23
Naive Bayes	*Kernel Naive Bayes (Kernel Gaussiano)*	89.3	88.7	88.7	9.4306
	Optimizable Naive Bayes	89.3	88.7	88.7	214.14
SVM	*Quadratic SVM (BC=1)*	94.5	94.4	94.4	5.4884
	Optimizable SVM	95.5	95.4	95.4	603.57
K-vecinos más cercanos	*Weighted KNN (Neighbors: 10)*	93.6	93.5	93.5	1.3097
	Optimizable KNN	93.7	93.7	93.7	173.88
Modelos de ensamblajes	*Ensemble Bagged Trees (Splits: 541, learners:30)*	92.3	92.3	92.3	7.0307
	Optimizable Ensemble	91.7	91.7	91.7	151.28
RNA	*Wide Neural Network (1 cap: 100 neuronas)*	94.3	94.3	94.3	24.999
	Optimizable Neural Network	95.4	95.4	95.4	1704.7

Los resultados de precisión y tiempo de entrenamiento, indican los algoritmos de árboles de decisión, los discriminantes, algoritmos eficientes y Kernel de Naive Bayes, el uso del modelo optimizable no cambió la precisión de los modelos. En los modelos de SVM, vecinos más cercanos y RNA, mejoró la precisión desde el 94.5; 93.6 y 94.3% al 95.4; 93.7 y 95.4% respectivamente. Por su parte, el modelo optimizable para los algoritmos de ensamble, disminuyó su precisión desde 92.3% hasta 91.7%. Adicionalmente, el tiempo que tarda el algoritmo de optimización es mayor a todos los algoritmos de Machine Learning a excepción del algoritmo discriminante, el cual resulta ser más rápido en su entrenamiento.

6.4 Comparación general de los modelos de Machine Learning aplicados en la investigación

Se compararon los algoritmos de Machine Learning mediante su precisión, a mayor precisión, menor error. Se ordenaron de manera descendente los 10 algoritmos con la mayor precisión obtenida, los cuales se muestran en las Tablas 32 y 33.

Tabla 32. Modelos de Machine Learning con la mayor precisión.

N°	Algoritmo	Precisión (%)	Error (%)	Tiempo (s)
1	RNA multicapa [160 80 80]	96.05	3.95	17
2	RNA multicapa [40 40 81]	95.68	4.32	9
3	RNA con función “traincgf”	95.68	4.32	33
4	RNA multicapa [160 80 40]	95.35	4.65	15
5	RNA con función “traincgf”	95.13	4.87	26
6	RNA multicapa [80 160 81]	95.12	4.88	31
7	SVM Optimizable (Classification Learner)	95	5	137.01
8	RNA multicapa (3 distintas configuraciones)	94.92 – 94.99	5.01-5.08	16 - 19
9	SVM cúbico “onevsall” con optimización de hiperparámetros	94.8339	5.1661	227.187
10	SVM gaussiano “onevsall” y cuadrático “onevsone” con optimización del hiperparámetros	94.6494	5.3506	70.398 y 180
11	RNA sin modificación exhaustiva	93.97	6.03	14

Nota. Se incluyó el algoritmo base de RNA con el que se trabajó originalmente, el cual ocupa el puesto 17.

Tabla 33. Modelos de Machine Learning con la mayor precisión entrenados con los datos filtrados.

N°	Algoritmo	Precisión (%)	Error (%)	Tiempo (s)
1	RNA sin modificación exhaustiva	95.53	4.47	13
2	SVM optimizable (cuadrático con 100 iteraciones)	95.5	4.5	603.57
3	RNA optimizable	95.4	4.6	1704.70
4	SVM cuadrado “onevsall” con optimización de hiperparámetros	95.38	4.62	142.44
5	SVM cúbico “onevsall” con optimización de hiperparámetros	94.8339	5.1661	245.41
6	SVM optimizable (Classification Learner)	94.7	5.3	58.54
7	SVM cuadrático (Classification Learner)	94.5	5.5	5.4884
8	SVM cuadrático “onevsone” con optimización de hiperparámetros	94.46	5.54	154.3919
9	SVM cúbico (Classification Learner)	94.3	5.7	13.273
10	RNA grande (Classification Learner)	94.3	5.7	24.999

7 **Discusión**

A partir de los resultados de la investigación, se observa que la RNA fue la que tuvo el mayor valor de precisión para clasificar los distintos estados del motor con respecto a los demás algoritmos de Machine Learning. Lo cual concuerda con lo mencionado por Tellez et al. (2022) quienes concluyen luego de una revisión literaria que los modelos de RNA se consideran superiores frente a otros modelos de aprendizaje.

Según Cho et al. (2021), el aplicar la optimización bayesiana a los modelos de SVM y RNA del Classification Learner mejora la precisión de los mismos. Dicho criterio se demostró en la investigación, puesto que al trabajar con los datos de un sensor MAP, el modelo de SVM incrementó su precisión. Sin embargo, para el modelo de RNA, el criterio se cumplió solamente cuando el modelo fue entrenado con más características del sensor MAP, que fue el caso cuando se utilizaron las características extraídas a partir de los datos filtrados del sensor MAP.

Kurani et al. (2023) en su investigación, mencionan que cuando se implementan más características para el entrenamiento de los algoritmos de Machine Learning la precisión se incrementa. En dicha investigación se emplean datos de mercado para realizar predicciones, mientras que en la presente investigación se extrajeron las características de la señal del sensor MAP, resultando en un comportamiento similar al de la investigación mencionada. Cuando se utilizaron las características obtenidas de los datos en bruto, la precisión máxima alcanzada por los algoritmos de RNA y SVM sin modificación fue del 93.97 % y 95% respectivamente, mientras que, al utilizar las características obtenidas de los datos filtrados, la precisión aumentó, llegando a valores de precisión del 95.53% y 95.5% para la RNA y el SVM respectivamente, lo cual afirma que el hecho de agregar más características para el entrenamiento de los algoritmos, aumenta la precisión de los mismos cuando se trabaja con los datos del sensor MAP.

Tras varias configuraciones del algoritmo de RNA, se consiguió mejorar su precisión desde un 93.97% hasta un 96.05% cambiando el número de neuronas de las 3 capas ocultas, los cambios se realizaron teniendo en cuenta patrones simples, en donde, la configuración que brindó el valor de 96.05% corresponde la configuración 160 80 80 $[2n\ n\ n]$. Sin embargo, al considerar tanto el tiempo de entrenamiento como el histograma de error para el entrenamiento, validación y prueba de la RNA, se considera que la mejor configuración del número de neuronas para las 3 capas serían 40 40 81 $[n\ n\ (2n+1)]$ puesto que es el segundo mejor porcentaje de clasificación, con el 95.68%, con el menor error del histograma y con un tiempo de entrenamiento de 9 segundos.

Al realizar la configuración del algoritmo de RNA con distintas funciones de entrenamiento, la precisión obtenida entre ellas, tuvo una variación máxima del 7.56% (88.12% como valor mínimo y 95.68% como valor máximo), excluyendo los modelos de aprendizaje "trainlm" y "trainbfg" puesto que dichos modelos no completaron su entrenamiento debido a su excesivo tiempo de entrenamiento y se los consideró que no son aptos para aplicarse en una RNA cuando se trabaja con características extraídas del sensor MAP. Al contrastar esta información con la investigación de Contreras U. et al. (2019) el error que obtiene con una RNA con función de aprendizaje "trainscg" tiene un porcentaje de error del 1.89e-11%, lo cual equivale a una precisión de casi 100%, sin embargo, con la función "traingdm" obtiene un porcentaje de error del 649.81% que equivaldría a una precisión del 0%, lo cual daría una variación de la precisión entre funciones de entrenamiento máxima del 100%. Sin embargo, en la investigación realizada por Delgado (2018), los valores de precisión se asemejan en mayor medida a la investigación realizada, puesto que, al excluir las funciones "traingda" y "traingdx" se obtiene una variación entre las funciones de entrenamiento del 5.9251.

Lo mencionado previamente, puede ser producto de que en la investigación de Contreras U. et al. (2019), las funciones de activación para las capas, son todas del tipo "tanscg"; mientras que, tanto en la presente investigación como en la investigación de Delgado (2018), se hace incluye una función de activación "purelin", de manera que los resultados no varían en un gran rango para los distintos tipos de funciones de aprendizaje. Otro factor a considerar corresponde a que, en ambas investigaciones mencionadas, se utiliza una RNA de tipo "newff", mientras que, en la presente investigación, se hace uso de la RNA "feedforwardnet", ambas son una RNA feedorward-propagation, sin embargo "feedforwardnet" es una red más actual.

Un factor a considerar es el sobreajuste de los algoritmos de clasificación, que según Microsoft (2024) se puede realizar mediante el uso de la matriz de confusión, y lo cual se empleó en la investigación. A partir de ella, se obtuvieron los valores del Recall y el F1-Score (Grandini et al., 2020), en donde, los resultados indicaron que no existe una diferencia amplia entre la precisión de los modelos utilizados y el Recall, de manera que se considera que los algoritmos entrenados no están sobreajustados.

Los resultados de los algoritmos de clasificación, demostraron que el modelo de clasificación a utilizar depende precisamente de la naturaleza de los datos, puesto que en el caso de Guleria et al. (2022) el algoritmo que tiene una mayor precisión para la clasificación de características propias del hipotiroidismo corresponde a los árboles de decisión. Sin embargo,

al trabajar con características extraídas del sensor MAP, los árboles de decisión no destacaron, puesto que en el mejor de los casos presentaron un 91.2% de precisión. Dicho criterio se corrobora puesto que en la investigación realizada por (Gallegos et al., 2020) se utilizan señales de electroencefalografía (EEG) cuyo espectro de señal es similar a las señales del sensor MAP, en la mencionada investigación se obtienen resultados de clasificación similares a los de la presente investigación en donde la RNA prevalece con una mayor precisión.

8 Conclusiones

- Se realizó la adquisición de datos de voltaje del sensor MAP de un motor Otto mediante una técnica mínimamente invasiva con el uso de una tarjeta NI USB DAQ 6009.
- Se analizaron los datos obtenidos mediante el empleo de algoritmos de Machine Learning de clasificación, logrando resultados favorables para la clasificación de las diferentes condiciones de funcionamiento del motor
- Se verificaron los resultados obtenidos mediante el análisis estadístico de los algoritmos de Machine Learning implementados, se identificó que las RNA tienen el menor margen de error y una mayor precisión con respecto a los demás algoritmos de clasificación.
- Se logró extraer los datos a partir de un motor Otto el cual fue la unidad experimental YESA 3133, dichos datos correspondieron a diferentes condiciones de funcionamiento del motor: una condición normal y diferentes condiciones de funcionamiento simuladas: falla de una bujía, falla de obstrucción del filtro de aire y la aplicación de ambos fallos a la vez.
- Se extrajeron las características principales de RMS, mediana, el pico mínimo y la energía de cada una de las señales del sensor MAP correspondientes a 720° de giro del cigüeñal para el entrenamiento de los algoritmos de Machine Learning, Dichas características resultaron luego de la aplicación de un análisis de varianza, correlación y de random forest.
- Se realizó la comparación del desempeño de los distintos algoritmos y modelos de Machine Learning aplicados, para el efecto, se analizó estadísticamente las métricas de precisión, recall y F1-Score, infiriendo:
    - La RNA multicapa se desempeña mejor que el SVM en la clasificación de las fallas del sensor MAP, puesto que su tiempo de entrenamiento es menor y alcanza valores más altos de precisión.
    - A partir de las diferentes configuraciones al algoritmo de RNA, se considera que la mejor corresponde al uso de un perceptrón multicapa, cuyas capas ocultas contienen 40, 40 y 81 neuronas para la primera, segunda y tercera capa respectivamente.
    - El aplicar un filtro Butterworth, mejora la precisión y el tiempo de entrenamiento de la RNA, puesto que se extraen más características de la señal.
    - En general, la RNA obtuvo el menor margen de error igual al 3.95%, mientras que el menor error obtenido por el SVM fue del 4.5%.

9 Recomendaciones

- Tener en consideración el rango de voltaje en el que trabaja el sensor a medir, puesto que de ello dependerá el dispositivo de adquisición de datos a utilizar, en el caso la investigación se utilizó la tarjeta NI USB DAQ 6009 ya que trabaja en un rango de voltaje de entre ±10 voltios.
- Si se van a adquirir las señales de dos o más sensores en ralentí, se recomienda adquirir los datos de todos los sensores al mismo tiempo puesto que el ralentí del motor es inestable.
- Si se desea implementar el código para cortar los datos de cualquier sensor según la señal del MAP, tener en cuenta la frecuencia de adquisición de datos, para con ello encontrar la cantidad de datos que se registran por segundo o milisegundo y modificar el código según se requiera.
- Si desea saber un aproximado de la precisión que tendrán sus modelos, se recomienda utilizar como referencia el clasificador “Tree” dentro del Classification Learner, puesto que la precisión de este, será muy cercana a los algoritmos de clasificación SVM o RNA, además que el tiempo de su ejecución es significativamente menor cuando se utiliza un dataset muy grande.
- Utilizar el entorno y el lenguaje de programación con el que más se familiarice puesto que los códigos se pueden adaptar.
- Si se utiliza Matlab, tener en cuenta los Ad Ons que son requeridos para la ejecución del código y la extracción de características de la señal.
- Tener en cuenta que, por defecto, el Classification Learner ya tiene señalada la opción de que los datos se encuentran estandarizados, por lo cual previo ingreso de los datos, se recomienda normalizarlos.
- Aplicar el filtro Butterworth para conseguir más características relevantes de la señal del sensor MAP mejore y así mejorar la precisión de los algoritmos a utilizar.
- Emplear el algoritmo de RNA para trabajar con las características extraídas del sensor MAP puesto que fue el más preciso.

10 **Bibliografía**

Abdullah, D. M., & Abdulazeez, A. M. (2021). Machine Learning Applications based on SVM Classification A Review. *Qubahan Academic Journal*, *1*(2), 81–90. https://doi.org/10.48161/QAJ.V1N2A50

Abeliuk, A., & Gutiérrez, C. (2021). Historia y evolución de la inteligencia artificial. *Revista Bits de Ciencia*. https://revistasdex.uchile.cl/index.php/bits/article/download/2767/2700

AEADE. (2024). *Sector Automotor en cifras*. https://www.aeade.net/wp-content/uploads/2024/07/6.-Sector-en-Cifras-Resumen-Mayo.pdf

Aguirre, B. G., & Campoverde, N. M. (2024). Análisis de las principales emisiones contaminantes de un motor de combustión interna mediante la variación de la longitud de las bujías en la cámara de combustión. *UPS*. https://dspace.ups.edu.ec/bitstream/123456789/27147/1/UPS-CT011248.pdf

Biddle, L., & Fallah, S. (2021). A Novel Fault Detection, Identification and Prediction Approach for Autonomous Vehicle Controllers Using SVM. *Automotive Innovation*, *4*(3), 301–314. https://doi.org/10.1007/S42154-021-00138-0/FIGURES/8

Bimboza, J., Cárdenas, L., Mora, C., & Mancheno, M. (2023). Calidad del servicio y satisfacción del cliente. El caso del mantenimiento vehicular liviano. *Religación: Revista de Ciencias Sociales y Humanidades, ISSN-e 2477-9083, Vol. 8, Nº. 35, 2023 (Ejemplar Dedicado a: Issue in Progress | Dossier | Poetic-Affective Rationality: A Political Approach to the Contemporary Theatrical Scene.; E2301032), Pág. 15*, *8*(35), 15. https://doi.org/10.46652/rgn.v8i35.1019

Bosch. (1996). *Manual de la técnica del automóvil* (A. Cypra, A. Beer, H. Bauer, & F. E. Wolfgang Schuch, Eds.; 3rd ed.). Editorial Reverté, S.A. https://juliorestrepo.wordpress.com/wp-content/uploads/2013/08/bosch-manual-de-la-tecnica-del-automovil-tercera-edicion.pdf

Carrión, R., Castillo, J., Díaz, D., & Briceño, B. (2022). Análisis termográfico de un motor de encendido provocado, inducido a fallas mediante la aplicación de diseño de experimentos (DoE). *Información Tecnológica*, *33*(1), 181–192. https://doi.org/10.4067/S0718-07642022000100181

Castillo, J., Díaz, D., Carrión, R., & Rivera, D. (2021). Influence of Failures not Detected by the On-Board Diagnostic System on the Performance and Pollutant Emissions of a Spark Ignition Engine. *Advances in Intelligent Systems and Computing*, *1326 AISC*, 389–403. https://doi.org/10.1007/978-3-030-68080-0_29

Cho, H. U., Nam, Y., Choi, E. J., Choi, Y. J., Kim, H., Bae, S., & Moon, J. W. (2021). Comparative analysis of the optimized ANN, SVM, and tree ensemble models using Bayesian optimization for predicting GSHP COP. *Journal of Building Engineering*, *44*, 103411. https://doi.org/10.1016/J.JOBE.2021.103411

Contreras U., W., Maldonado O., J., & León J., R. (2019). Aplicación de una red neuronal Feer-Forward Backpropagation para el diagnóstico de fallas mecánicas en motores de encendido provocado. *Ingenius. Revista de Ciencia y Tecnología*, *2019*(21), 32–40. https://doi.org/10.17163/INGS.N21.2019.03

Dagnino, J. (2014). Análisis de Varianza. *Rev Chil Anest*, 306–310.

Delgado, E. H. (2018). *Desarrollo de un algoritmo de diagnóstico para la detección de fallas mecánicas en motores de encendido provocado basados en la transformada Wavelet*. http://dspace.ups.edu.ec/handle/123456789/15300

Galindo, E. A., Perdomo, J. A., & Figueroa-García, J. C. (2020). Comparative study among multiclass support vector machines, artificial neural networks and self-organized neuro-fuzzy inference system for classification problems. *Información Tecnológica*, *31*(1), 273–286. https://doi.org/10.4067/S0718-07642020000100273

Gallegos, A. E., Torres, M., Torres, A., & Ponce de León, E. (2020). Contrastación de algoritmos de aprendizaje automático para la clasificación de señales EEG Machine Learning Algorithms Testing for EEG Signal Classification. *Research in Computing Science*, *149*(8), 2020–2515.

García, F. A., Escobar, J. L., Gallegos, C. M., & Hernández, E. S. (2023). El enfoque de aprendizaje conjunto en la detección de fallas en cajas de engranajes. *Revista Universidad y Sociedad*, *15*(3), 325–333. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202023000300325&lng=es&nrm=iso&tlng=es

García, R. (2022). El perceptrón: Una red neuronal artificial para clasificar datos. *REVISTA DE INVESTIGACIÓN EN MODELOS MATEMATICOS APLICADOS A LA GESTION Y LA ECONOMIA*, *1*, 1–14. http://www.economicas.uba.ar/institutos_y_centros/revista-modelos-matematicos/

Granda, J. A., & Granda, J. D. (2022). *Análisis de fallas en la válvula EGR electrónica y en el Intercooler de un motor Diesel CRDI 2.2 mediante vibraciones y emisiones de gases*. http://dspace.ups.edu.ec/handle/123456789/23640

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: an Overview*. https://arxiv.org/abs/2008.05756v1

Guleria, K., Sharma, S., Kumar, S., & Tiwari, S. (2022). Early prediction of hypothyroidism and multiclass classification using predictive machine learning and deep learning. *Measurement: Sensors*, *24*, 100482. https://doi.org/10.1016/J.MEASEN.2022.100482

Gutiérrez, H., & Salazar, R. de la V. (2008). *Análisis y diseño de experimentos* (2nd ed.). MGraw-Hill Interamericana.

Gutiérrez, J., Chica, E., & Pérez, J. F. (2023). Análisis de varianza y coeficiente de variación como criterios de repetibilidad de una estufa de cocción basada en gasificación de pellets. *Revista UIS Ingenierías*, *22*(3), 115–134. https://doi.org/10.18273/REVUIN.V22N3-2023009

Guzmán, Y. (2023). *Redes neuronales artificiales*. https://doi.org/10.16925/GCGP.113

INEC. (2024). *Estadísticas de Transporte*. https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Economicas/Estadistica%20de%20Transporte/veh_matriculados/2023/2023_Resultados_Vehiculos.pdf

Kenett, Ron., Zacks, Shelemyahu., & Amberti, Daniele. (2021). *Modern industrial statistics : with applications in R, MINITAB and JMP*. https://books.google.com/books/about/Modern_Industrial_Statistics.html?hl=es&id=iYEqEAAAQBAJ

Kumar, R., & Jain, A. (2023). Driving behavior analysis and classification by vehicle OBD data using machine learning. *Volume 79, Issue 16, Pages 18800 - 18819*, *79*(16), 18800–18819. https://doi.org/10.1007/s11227-023-05364-3

Kurani, A., Doshi, P., Vakharia, A., & Shah, M. (2023). A Comprehensive Comparative Study of Artificial Neural Network (ANN) and Support Vector Machines (SVM) on Stock Forecasting. *Annals of Data Science*, *10*(1), 183–208. https://doi.org/10.1007/S40745-021-00344-X/METRICS

Li, X., Wang, N., Lyu, Y., Duan, Y., & Zhao, J. (2023). Data-Driven Fault Early Warning Model of Automobile Engines Based on Soft Classification. *Volume 12, Issue 3*, *12*(3). https://doi.org/10.3390/electronics12030511

Marín, T. D., & Arriojas, D. D. J. (2021, January 20). Ubicación de revistas científicas en cuartiles según SJR: Predicción a partir de estadística multivariante. *Anales de Documentación*, *24*. https://doi.org/http://dx.doi.org/10.6018/analesdoc.455951

MathWorks. (2024). *Machine Learning*. https://la.mathworks.com/solutions/machine-learning.html

Mena, J. A., & Mena, L. A. (2023). *Análisis del flujo de aire en el sistema de admisión de un vehículo*. https://repositorio.utn.edu.ec/handle/123456789/13871

Microsoft. (2024, September 3). *Evitar el sobreajuste y los datos desequilibrados con el ML automatizado*. https://learn.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls?view=azureml-api-2&utm_source=chatgpt.com

Muñoz, M., & Rovira, A. (2015). Motores de Combustión Interna. *UNED*. https://books.google.com.ec/books?id=-EfLCgAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

National Instruments. (2015, July). *NI USB-6008/6009 User Guide*. National Instruments. https://c7076-control.chem.sfu.ca/interlock_monitoring_system_resources/usb6008.pdf

National Instruments. (2025, January 15). *¿Qué es NI LabVIEW?* https://www.ni.com/es/shop/labview.html

Omama, Y., Haddad, C., MacHaalany, M., Hamoudi, A., Hajj-Hassan, M., Ali, M. A., & Hamawy, L. (2019). Surface EMG Classification of Basic Hand Movement. *International Conference on Advances in Biomedical Engineering, ICABME*, *2019-October*. https://doi.org/10.1109/ICABME47164.2019.8940352

Passo, R., Egas, J. P., Arguello, J. A., García, J. G., & Lisintuña, E. P. (2024). Aplicación de un modelo matemático para el análisis del comportamiento del ciclo otto MCI bajo manipulación de parámetros ambientales. *CONECTIVIDAD*, *5*(1), 1–19. https://doi.org/10.37431/CONECTIVIDAD.V5I1.111

Raschka, S., & Mirjalili, V. (2017). *Python Machine Learning* (Packt Publishing, Vol. 2). Packt Publishing Ltd. http://radio.eng.niigata-u.ac.jp/wp/wp-content/uploads/2020/06/python-machine-learning-2nd.pdf

Rivas, W., Bertha, A., & Olivo, M. (2018). Redes neuronales artificiales aplicadas al reconocimiento de patrones. *Redes 2017*. https://www.researchgate.net/profile/Bertha-Mazon-Olivo/publication/327703478_Capitulo_1_Generalidades_de_las_redes_neuronales_artificiales/links/5b9fe3c0299bf13e6038a1d8/Capitulo-1-Generalidades-de-las-redes-neuronales-artificiales.pdf

Sánchez, E. (2008). *Sistemas auxiliares del Motor* (Vol. 1). Macmillan.

Sánchez, R. V., Macancela, J. C., Ortega, L. R., Cabrera, D., García Márquez, F. P., & Cerrada, M. (2024). Evaluation of Hand-Crafted Feature Extraction for Fault Diagnosis in Rotating Machinery: A Survey. *Sensors 2024, Vol. 24, Page 5400*, *24*(16), 5400. https://doi.org/10.3390/S24165400

Shahbaz, M. H., & Amin, A. A. (2021). Design of Active Fault Tolerant Control System for Air Fuel Ratio Control of Internal Combustion Engines Using Artificial Neural Networks. *IEEE Access*, *9*, 46022–46032. https://doi.org/10.1109/ACCESS.2021.3068164

Taylor, T., Youssef, A., -, al, Ajevi, M., Furlanis, G., Buoite Stella, A., Zhang, R., Yu, A., & Nishi, M. (2021). Classification of Admission Data Using Classification Learner Toolbox. *Journal of Physics: Conference Series*, *1979*(1), 012043. https://doi.org/10.1088/1742-6596/1979/1/012043

Tellez, J. C., Ateeq, K., Rafiuddin, A., Alzoubi, H. M., Ghazal, T. M., Ahanger, T. A., Chaudhary, S., & Viju, G. K. (2022). AI-Based Prediction of Capital Structure: Performance Comparison of ANN SVM and LR Models. *Computational Intelligence and Neuroscience*, *2022*(1), 8334927. https://doi.org/10.1155/2022/8334927

Tello, L., Aguirre, M., Díaz, J. P., & Hernández, F. (2021). Evaluación de daños en pavimento flexible usando fotogrametría terrestre y redes neuronales. *TecnoLógicas*, *24*(50), 59–71. https://doi.org/10.22430/22565337.1686

Tuan, A., Nižetić, S., Chyuan, H., Tarelko, W., Viet, V., Hieu, T., Quang, M., & Phuong, X. (2021). A review on application of artificial neural network (ANN) for performance and emission characteristics of diesel engine fueled with biodiesel-based fuels. *Sustainable Energy Technologies and Assessments*, *47*, 101416. https://doi.org/10.1016/J.SETA.2021.101416

Umaña, G. (2021). *Aplicación empírica de metodologías de Machine Learning to Rank. Comparación con otras técnicas de aprendizaje supervisado en diferentes aplicaciones*. https://repositorio.utdt.edu/handle/20.500.13098/11881

Valdés, J. (2020, April 30). *Sistema de admisión*. Espirituvintage. https://espirituvintage.com/2020/04/30/sistema-de-admision/

Valero-Carreras, D., Alcaraz, J., & Landete, M. (2023). Comparing two SVM models through different metrics based on the confusion matrix. *Computers & Operations Research*, *152*, 106131. https://doi.org/10.1016/J.COR.2022.106131

Velepucha, J. M., & Sabando, L. F. (2021). Emisiones de gases contaminantes en vehículos livianos a gasolina. *Revista Científica INGENIAR: Ingeniería, Tecnología e Investigación. ISSN: 2737-6249.*, *4*(8), 78–95. https://doi.org/10.46296/IG.V4I8.0024

Weather Spark. (2025, January). *El clima y el tiempo promedio en todo el año en Loja Ecuador*. https://es.weatherspark.com/y/19339/Clima-promedio-en-Loja-Ecuador-durante-todo-el-a%C3%B1o

YES01. (2010). *YESA-3133 Automotive Engine Fault Diagnostic Simulator_Auto Fault 2002 EF SONATA User Manual* (1.6). YES01 U Learning Engineering Institute.

Yixin, E. S., Saeid, A. W., & Tjong, J. (2022). Fault Detection and Diagnosis of Engine Spark Plugs Using Deep Learning Techniques. *SAE International Journal of Engines*, *15*, 515–526. https://www.jstor.org/stable/27206785

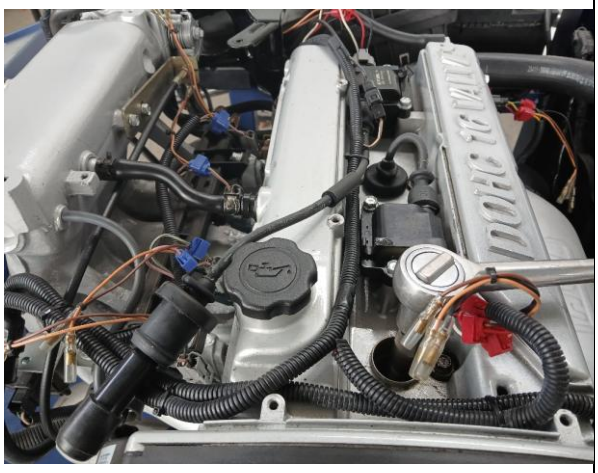
Yu, Z., Zhang, Y., Al-Haddad, L. A., & Abdulhady Jaber, A. (2023). An Intelligent Fault Diagnosis Approach for Multirotor UAVs Based on Deep Neural Network of Multi-

Resolution Transform Features. *Drones 2023, Vol. 7, Page 82*, *7*(2), 82. https://doi.org/10.3390/DRONES7020082

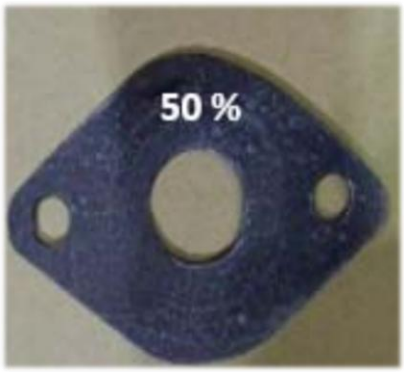
Zhou, Z. H. (2021). *Machine Learning* (Springer Nature, Ed.). Springer Nature. https://books.google.es/books?hl=es&lr=&id=ctM-EAAAQBAJ&oi=fnd&pg=PR6&dq=machine+learning&ots=o_KoW1Rz4n&sig=OIJMwO2jbDcPVrPZUaCgToyRHww#v=onepage&q&f=false

11 Anexos

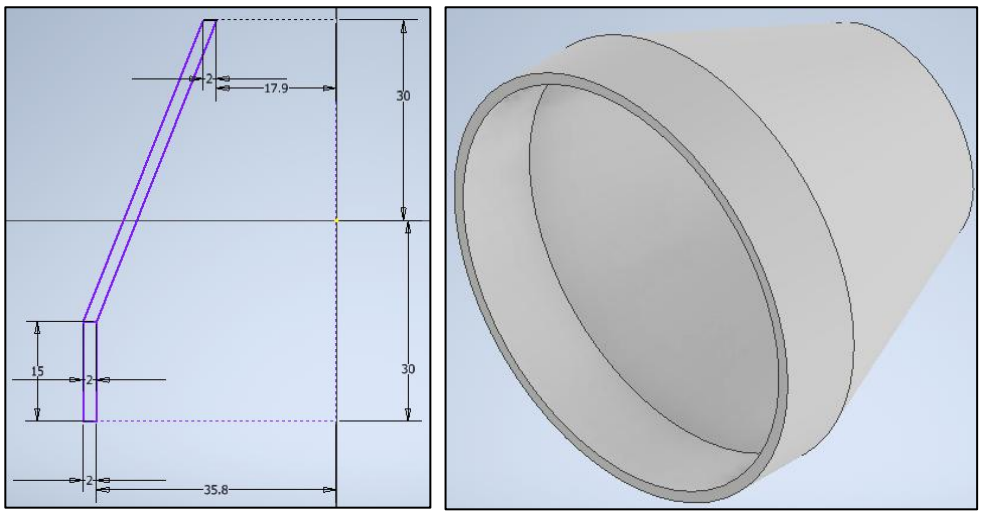
Anexo 1. Herramientas utilizadas para aplicar las fallas simuladas.

Motor OK	**Simulación de ambas fallas del motor**
	
Simulación de obstrucción del filtro de aire	**Simulación de falla en la bujía**
	

Anexo 2. Aplicación de los distintos tratamientos a la unidad experimental.



Anexo 3. Simulación de estrangulación del flujo de aire mediante una empaquetadura.

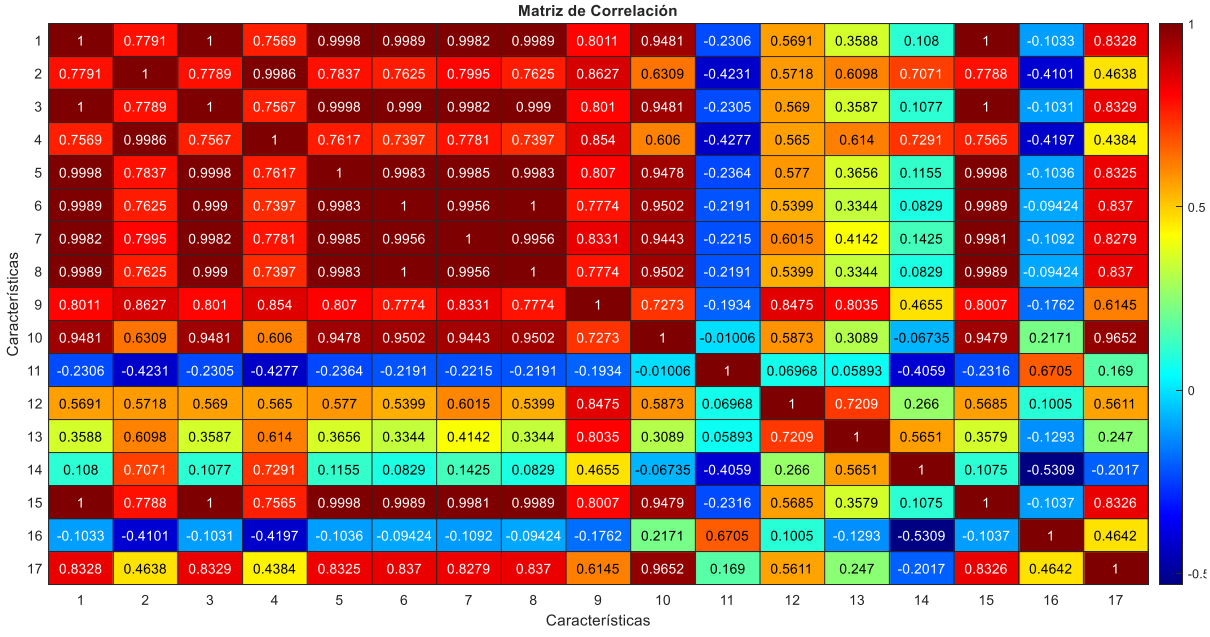


Anexo 4. Diseño de un cono en Autodesk Inventor para la obstrucción del 50% del conducto de admisión de aire.

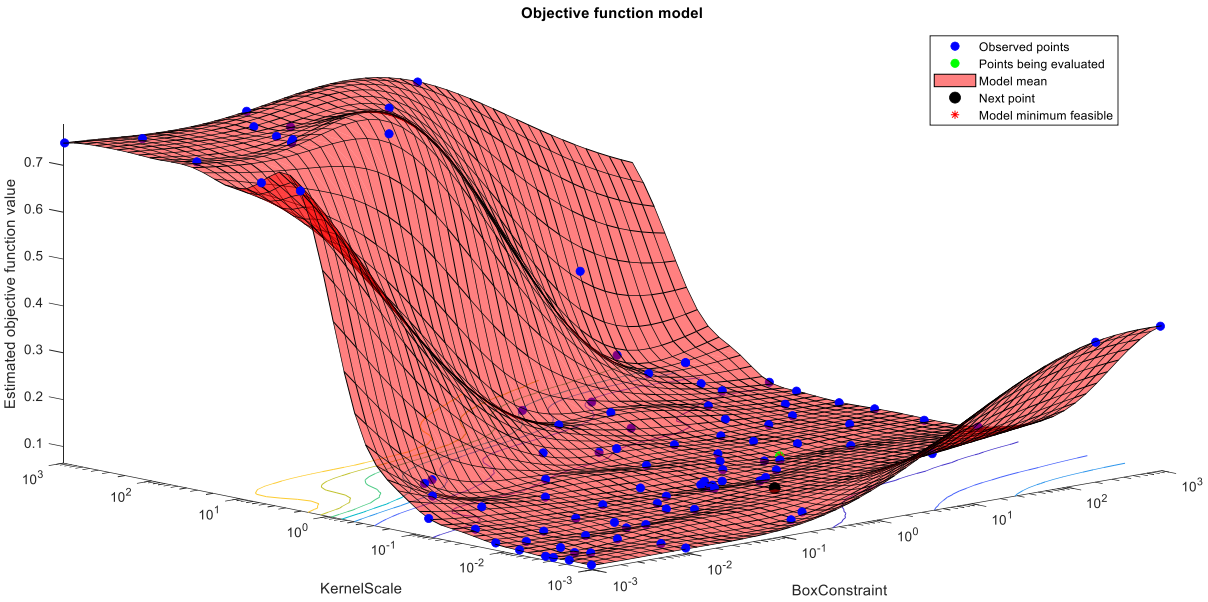
OrdenEst	OrdenCorrida	PuntoCentral	Bloques	Bujía	Filtro Aire
26	1	1	1	1	0
2	2	1	1	1	0
16	3	1	1	1	1
29	4	1	1	0	0
31	5	1	1	0	1
24	6	1	1	1	1
6	7	1	1	1	0
39	8	1	1	0	1
10	9	1	1	1	0
25	10	1	1	0	0
23	11	1	1	0	1
11	12	1	1	0	1
40	13	1	1	1	1
18	14	1	1	1	0
3	15	1	1	0	1
34	16	1	1	1	0

OrdenEst	OrdenCorrida	PuntoCentral	Bloques	Bujía	Filtro Aire
27	17	1	1	0	1
1	18	1	1	0	0
8	19	1	1	1	1
21	20	1	1	0	0
22	21	1	1	1	0
33	22	1	1	0	0
13	23	1	1	0	0
30	24	1	1	1	0
38	25	1	1	1	0
7	26	1	1	0	1
12	27	1	1	1	1
37	28	1	1	0	0
20	29	1	1	1	1
19	30	1	1	0	1
32	31	1	1	1	1
15	32	1	1	0	1
28	33	1	1	1	1
36	34	1	1	1	1
17	35	1	1	0	0
9	36	1	1	0	0
35	37	1	1	0	1
5	38	1	1	0	0
4	39	1	1	1	1
14	40	1	1	1	0

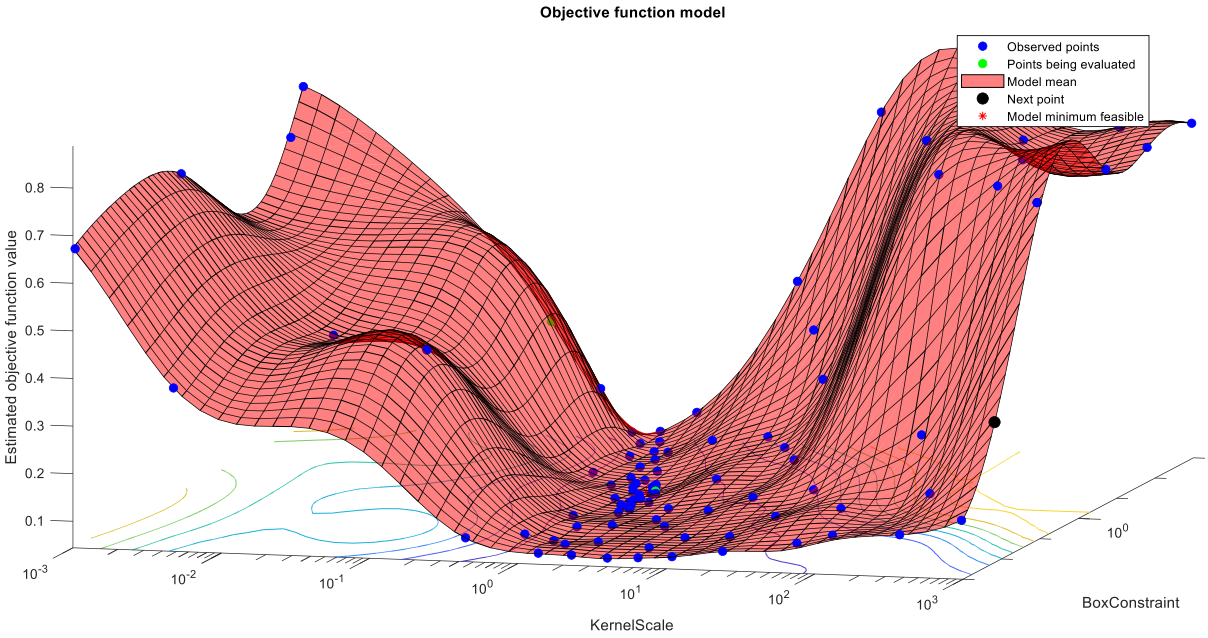
Anexo 5. Diseño factorial del funcionamiento del motor utilizado para la adquisición de datos del sensor MAP.



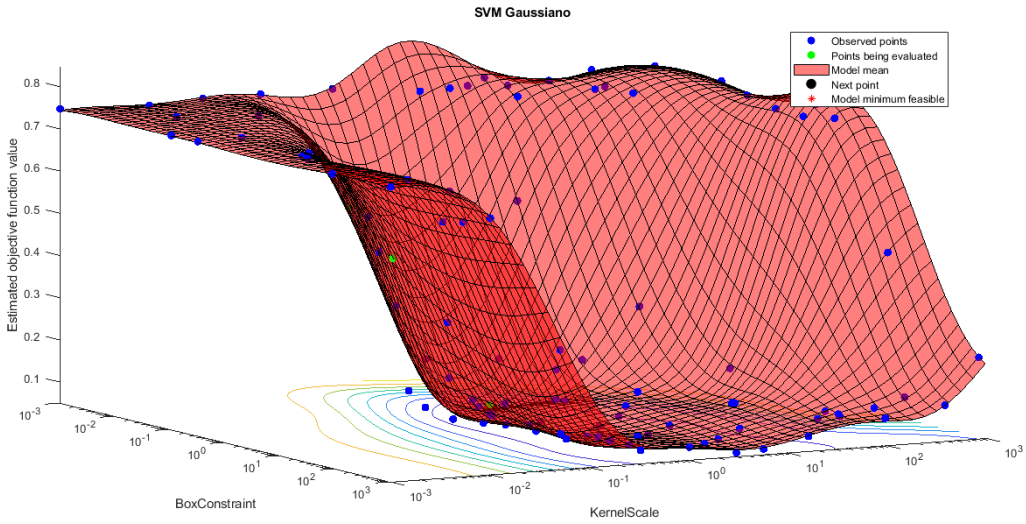
Anexo 6. Matriz de correlación de las características obtenidas a partir de los datos filtrados.



Anexo 7. Hiperplano de SVM lineal.



Anexo 8. Hiperplano de SVM cúbico.



Anexo 9. Hiperplano de SVM gaussiano.

Anexo 10. Certificado de traducción del Abstract

Loja, 4 de febrero de 2025

CERTIFICACIÓN DE TRADUCCIÓN

Doctora.
Erika Lucía González Carrión, Ph.D.

CERTIFICO:

Yo, Doctora Erika Lucía González Carrión, Ph.D., con cédula de ciudanía 1105820953, en mi calidad de traductora del idioma Inglés, con capacidades que pueden ser probadas a través de los siguientes documentos acreditativos:

1. **TÍTULO DE LICENCIADA EN CIENCIAS DE LA EDUCACION MENCION IDIOMA INGLES**, Registro SENESCYT Nro. 1008-16-1457913 (Anexo 1 documento SENESCYT): https://drive.google.com/file/d/1TrRikB37XkDSSXRhIsZDh4FhWbylYkFz/view?usp=sharing

2. **CERTIFICADO DE PROFICIENCIA DEL IDIOMA INGLES** (Anexo 2 documento): https://drive.google.com/file/d/1cNGWVEFjYH1E4eoHVDHGDkmLFEIIUYAT/view?usp=sharing

3. **CERTIFICADO INGLÉS NIVEL B2** (Anexo 3): https://drive.google.com/file/d/1i9QP22MCNrRMkflrKPO54003zE92tfMu/view?usp=sharing

4. **ACCESO A REVISTA COMUNICAR- BLOG ESCUELA DE AUTORES:** https://www.grupocomunicar.com/wp/school-of-authors/
(Al acceder en el enlace al blog, se podrá evidenciar la traducción realizada por quien certifica de cada entrada. Para mayor referencia observar la captura de pantalla adjunta ANEXO 4 : https://drive.google.com/file/d/1UjPj_R1ciRBxeW8UwNUHuNxICOXClr2f/view?usp=sharing

Con fundamento en la citada experiencia, numerales 1 al 4, **C E R T I F I C O** que la traducción del Resumen (Abstract) del Trabajo de Titulación denominado: **Empleo de algoritmos de Machine Learning para la detección de fallos en el sistema de encendido y admisión de aire en un motor Otto**, de autoría del estudiante: **Juan Pablo Medina Namicela** con CI: **11509922266**, es correcta y completa, según las normas internacionales de traducción de textos.

Es cuanto puedo certificar en honor a la verdad, facultando al interesado, **Juan Pablo Medina Namicela,** hacer uso legal del presente, según estime conveniente.

Atentamente,

Firmado electrónicamente por:
ERIKA LUCIA GONZALEZ CARRION

.....

Dra. Erika González Carrión. PhD.
C.I. 1105820953

- Registro SENESCYT Nro. 1008-16-1457913 - LICENCIADA EN CIENCIAS DE LA EDUCACION MENCION IDIOMA INGLES
- Registro SENESCYT Nro. 1031-15-1414538 - LICENCIADO EN COMUNICACION SOCIAL
- Registro SENESCYT Nro. 7242132304 - MASTER UNIVERSITARIO EN COMUNICACION Y EDUCACION AUDIOVISUAL
- Registro SENESCYT Nro. 7241182671 - DOCTORA DENTRO DEL PROGRAMA DE DOCTORADO EN COMUNICACION
- Registro Investigador SENESCYT acreditado: REG-INV-22-05714-Investigador Agregado 1.