Universidad Nacional de Loja

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

Carrera de Ingeniería Electromecánica

Aplicación de Técnicas de Inteligencia Artificial para la predicción de fallas en un aerogenerador de eje horizontal utilizando los datos del Sistema SCADA

Trabajo de Titulación previo a la obtención del título de Ingeniero Electromecánico.

AUTOR:

Jorge Alexander Puchaicela Castillo

DIRECTOR:

Ing. Jorge Luis Maldonado Correa Mg. Sc.

Loja – Ecuador

2023

Certificación

Loja, 2 de agosto de 2023

Ing., Jorge Luis Maldonado Correa Mg. Sc.
DIRECTOR DE TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Aplicación de Técnicas de Inteligencia Artificial para la predicción de fallas en un aerogenerador de eje horizontal utilizando los datos del Sistema SCADA,** previo a la obtención del título de **Ingeniero Electromecánico**, de la autoría del señor **Jorge Alexander Puchaicela Castillo**, con **cédula de identidad** Nro. **1105946972,** una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
JORGE LUIS
MALDONADO CORREA

Ing. Jorge Luis Maldonado Correa Mg. Sc
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Jorge Alexander Puchaicela Castillo**, declaro ser autor del presente Trabajo 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 mí del Trabajo de Titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

C.I.: 1105946972

Fecha: 31 de julio de 2022

Correo electrónico: japuchaicelac@unl.edu.ec

Teléfono: 0967263097

Carta de autorización por parte del autor para la consulta, reproducción parcial o total, y/o publicación electrónica del texto completo del Trabajo de Titulación.

Yo, **Jorge Alexander Puchaicela Castillo**, declaro ser autor del Trabajo de Titulación denominado **Aplicación de Técnicas de Inteligencia Artificial para la predicción de fallas en un aerogenerador de eje horizontal utilizando los datos del Sistema SCADA,** como requisito para optar el título de **Ingeniero Electromecánico**, 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 del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, al primer día del mes de agosto del dos mil veintitrés.

Firma: 

Autor: Jorge Alexander Puchaicela Castillo
Cédula: 1105946972
Dirección: Avenida Isidro Ayora, Calles La Habana y Salta.
Correo electrónico: japuchaicelac@unl.edu.ec
Teléfono: 0967263097

DATOS COMPLEMANTARIOS:
Director del Trabajo de Titulación: Ing. Jorge Luis Maldonado Correa Mg. Sc

Dedicatoria

El presente trabajo se lo dedico Dios, fuerza y divinidad del universo, sin su ayuda esto no hubiese sido posible.

A mis padres, este logro es de ellos, no me alcanzará la vida para devolverles lo que me han dado.

Jorge Alexander Puchaicela Castillo

Agradecimiento

Agradecimiento especial al Ing. Jorge Luis Maldonado Correa Mg. Sc., director de este Trabajo de Titulación, por sus enseñanzas, aportes y conocimientos brindados en la realización de esta investigación, sobre todo por su gran paciencia e interés en la misma.

A la Universidad Nacional de Loja, a la Facultad de la Energía las Industrias y los recursos Naturales no Renovables y a los docentes de la carrera de Ingeniería Electromecánica por sus sapiencias entregadas en cada una de las etapas de mi aprendizaje, y a mis compañeros de aula por ser parte del mismo.

Agradecimiento al Centro de Investigaciones Tecnológicas y Energéticas – CITE.

A mis hermanos, que alguna vez tuvieron que hacer lo impensado por mí y siempre estuvieron para apoyarme.

A mis familiares, amigos, conocidos, a todos quienes en su momento me brindaron una palabra de aliento para seguir adelante.

Jorge Alexander Puchaicela Castillo

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de Anexos:

Simbología:

AUC: Área bajo la curva

CNN: Red Neuronal Convolucional

DL: Deep Learning o Aprendizaje Profundo

FN: False Negative o Falso Negativo

FP: False Positive o Falso Positivo

HAWT: aerogenerador de eje horizontal pos sus siglas en inglés: Horizontal Axis Wind Turbine.

IA: Inteligencia Artificial

IR: Ratio de Desbalance

ML: Machine Learning o Aprendizaje Automático

MLP: Perceptrón multicapa

RNA: Redes Neuronales Artificiales

ROC: Característica operativa del receptor

SCADA: Supervision, control y adquisición de datos.

SMOTE: Técnica de sobremuestreo de minorías sintéticas

TN: True Negative o Verdadero Negativo

TP: True Positive o Verdadero Positivo

1. Título

Aplicación de Técnicas de Inteligencia Artificial para la predicción de fallas en un aerogenerador de eje horizontal utilizando los datos del Sistema SCADA

2. Resumen

En el presente trabajo, se realiza la predicción de fallas en el generador de un aerogenerador de eje horizontal mediante clasificación binaria, con el uso de técnicas de Inteligencia Artificial, específicamente, Machine Learning y Deep Learning. El principal objetivo de esta investigación es generar un modelo de predicción de fallas, mediante el análisis de los datos recogidos por el sistema SCADA (Supervisory Control And Data Acquisition). Para esto, en primer lugar, se realiza el análisis de los datos con técnicas de estadística descriptiva y visualización de datos, lo que permitió una adecuada selección de las variables, la preparación de la base de datos y la utilización de diferentes técnicas de remuestreo para eliminar el problema de desbalance de clases. Como resultado del análisis, se evidenció que, el submuestreo aleatorio presenta mejores resultados para la clasificación con el algoritmo de Árboles de Decisión, por lo que el conjunto equilibrado en sus clases de falla y estado normal, se emplea en la siguiente parte del estudio. A continuación, para aplicar de manera efectiva las técnicas de Machine Learning y Deep Learning, es necesario seleccionar adecuadamente aquellas que mejor se acoplen a la tarea de clasificación, para luego, una vez ajustados sus hiperparámetros, comparar su desempeño respecto a la que mejor predicción de fallas presente, trascendiendo el algoritmo Extra Trees con un valor de Recall igual a 0,973 y las redes neuronales convolucionales (CNN), aunque esta última trabaja con un conjunto de datos diferente alcanzando un valor de Recall de 0,79; resultando poco eficiente. Finalmente, la evaluación de los modelos se realiza con métricas derivadas de la matriz de confusión, especialmente Recall y el área bajo la curva ROC (AUC).

Palabras clave: Machine Learning, Deep Learning, clasificación desequilibrada, matriz de confusión, aerogenerador.

2.1 Abstract

In this work, the prediction of failures in a generator of horizontal axis wind turbine is performed by binary classification using Artificial Intelligence techniques, specifically Machine Learning and Deep Learning. The main objective of this research is to generate a fault prediction model by analyzing the data collected by the SCADA (Supervisory Control and Data Acquisition) system. First of all, the data analysis was performed with descriptive statistics and data visualization techniques, which allowed a good selection of the variables, the preparation of the database, and the use of different resampling techniques to eliminate the problem of class imbalance. As a result of the analysis, it showed that the random subsampling presents better results for the classification with the Decision Trees algorithm, so the balanced set in its failure classes and normal state is applied in the next part of the study. Next, in order to effectively apply Machine Learning and Deep Learning techniques, it is necessary to properly select those that are best suited to the classification task, and then, once their hyperparameters have been adjusted, compare their performance with respect to the one that best predicts failures, transcending the Extra Trees algorithm and convolutional neural networks (CNN). although the latter works with a different data set, reaching a Recall value of 0.79, which is not very efficient. Finally, the evaluation of the models is performed with metrics derived from the confusion matrix, especially Recall and the area under the ROC curve (AUC).

Key words: Machine Learning, Deep Learning, imbalanced classification, confusion matrix, wind turbine.

3. Introducción

El tiempo útil de funcionamiento de cualquier máquina, así como de una turbina eólica, depende de las condiciones en las que permanezcan sus componentes, siendo necesario el mantenimiento de los mismos por cualquier técnica que se ajuste a las características de la instalación, ya sea mantenimiento correctivo, preventivo, predictivo, etc. (García, 2003).

Según Loría, Villalobos & Piedra (2017), los costos de operación y mantenimiento (O&M) en la industria eólica son altamente representativos, siendo necesario su disminución para obtener una eficiente producción eléctrica.

La vida útil de los componentes de un aerogenerador influye en el funcionamiento óptimo de los mismos para la producción de energía eléctrica, una parada improvisada de los equipos implica costos por mantenimiento correctivo, así como pérdidas por falta de generación eléctrica en el tiempo establecido para el recambio de los componentes necesarios.

Los parques eólicos cuentan con programas y equipos especializados para el mantenimiento de los aerogeneradores, siendo las principales labores de mantenimiento las inspecciones visuales del nivel de aceite, toma de muestras del mismo para análisis, limpieza de algunos componentes y apriete de tornillos. También se realiza mantenimiento mediante técnicas no invasivas como endoscopías y termografías, esta última de mucha utilidad para el monitoreo del generador, elemento de gran importancia para el aerogenerador según Verma & Kusiak (2012), ya que, de producirse una avería, Pernia & Hernández (2014), afirman que el tiempo de parada es más extenso que el resto de componentes y, por tanto, los costos de mantenimiento son a gran escala.

Una práctica no invasiva para el monitoreo de condición del generador de una turbina eólica se presenta en el trabajo de Martínez (2017), quien emplea la técnica de estimación de estado no lineal NSET, cuyo modelo es un método estadístico que estima la ocurrencia de una avería a horas de producirse.

Otra herramienta no invasiva es la predicción de fallas con técnicas de Inteligencia Artificial, como lo demuestran Verma & Kusiak (2012), aplicando el algoritmo boosting trees para la predicción tempana de los fallos de las escobillas del generador hasta 12 horas antes de que se originen, lo que permite realizar a tiempo su mantenimiento o sustitución.

Politi (2022), genera un modelo LSTM (Long Short - Term Memory) para analizar la fluctuación de temperatura en el generador y detectar en tiempo real cualquier malfuncionamiento o alarma originada por el sistema de control.

Trabajos relacionados con algoritmos de clasificación de datos, pero perfectamente aplicables a análisis de fallas en aerogeneradores se presentan en Huertas (2020), Arnejo (2017)

y Espinar (2018), donde se hace énfasis en el remuestreo de datos para disminuir el desbalance de clases y mejorar la predicción. Así mismo, Pinto & Cerquitelli (2019) y Hasegawa, Saeki, Ogawa, & Nakano (2019), proponen el monitoreo de condiciones a través de redes neuronales convolucionales para tal fin.

Con la finalidad de reducir las paradas programadas de mantenimiento en una central eólica y mantener un nivel constante de generación de energía eléctrica, en este trabajo se realizó el tratamiento de datos recogidos por el sistema de control de un aerogenerador y se comparó diferentes técnicas de Inteligencia Artificial relacionadas con la clasificación para detectar fallas generadas por el aumento de la temperatura en los devanados del generador.

Las técnicas de Inteligencia Artificial proporcionarán al parque eólico del que se obtuvieron los datos para el respectivo análisis, un modelo de predicción de fallas basado en las observaciones recogidas por el sistema SCADA, para incorporar el mantenimiento predictivo generando ahorros por costo de recambio y paradas programadas o inesperadas.

Los objetivos que ayudaron a guiar la presente investigación se describen a continuación:

Objetivo general:

- Generar un modelo de predicción de fallas en un aerogenerador de eje horizontal aplicando técnicas de Inteligencia Artificial a los datos del Sistema SCADA.

  Objetivos específicos:

  - ✓ Examinar los datos históricos de las principales variables del aerogenerador que son recogidos por el sistema SCADA para determinar el comportamiento del generador.
  - ✓ Desplegar la arquitectura de un algoritmo para la predicción de fallas en el generador de la turbina eólica a partir del análisis de los datos.
  - ✓ Valorar el rendimiento y precisión del modelo de predicción de fallas en el generador utilizando métricas de rendimiento para la evaluación de aprendizaje automático.

4. Marco teórico

4.1 Capítulo I: Energías Renovables

Según Poveda, Ruiz & González (2017), "las energías renovables son aquellas que provienen de recursos naturales renovables que son teóricamente inagotables y pueden regenerase" (p. 2).

Las principales formas de energías renovables que existen son: la biomasa, hidráulica, eólica, solar, geotérmica y las energías marinas. Este tipo de energías provienen de forma directa o indirecta, de la energía del Sol; a excepción de la energía geotérmica y de las mareas (Schallenberg et al., 2008).

Estos recursos naturales y por tanto las energías renovables constituyen la base de los tres pilares del desarrollo sostenible: económico, social y calidad medioambiental, que requieren una garantía de desarrollo sostenible, es decir, utilizar un modelo energético que satisfaga las necesidades actuales, sin comprometer la capacidad de las generaciones futuras (Poveda et al., 2017).

4.1.1 Energía eólica

El viento es el resultado del movimiento de las masas de aire, que a su vez es causado por los cambios de temperatura que provoca la radiación solar sobre las distintas partes del planeta Tierra que originan diferencias de presión y densidad (Villarubia, 2013).

Considerando al viento como recurso energético y dadas sus condiciones de disponibilidad (componente aleatorio), es una fuente con importantes variaciones temporales, a pequeña y gran escala, y espaciales, tanto en superficie como en altura (Pueyo, Telmo, & Pérez, 2011).

Para el cálculo del potencial eólico, Pueyo et al. (2011), siguen el siguiente razonamiento:

La cantidad de aire que atraviesa una superficie S por unidad de tiempo se determina mediante la **Ecuación** 1:

$$m = \rho * A * U \quad (1)$$

Donde:

m = cantidad de aire que atraviesa una superficie S por unidad de tiempo [kg/s].

ρ = densidad del aire [kg/m^3].

A = área barrida por una cantidad de aire en un segundo [m^2].

U = velocidad de viento [m/s].

Debido a que la energía del viento es energía cinética, la potencia que atraviesa dicha superficie se da por la **Ecuación** 2:

$$P = 0.5 * m * U^2 = 0.5 * \rho * A * U^3 \quad (2)$$

Donde:

P = potencia eólica que atraviesa el área barrida A [W]

Los datos a destacar de esta ecuación son que la potencia es directamente proporcional al área barrida, y que es directamente proporcional al cubo de la velocidad del viento.

Si ρ se expresa en kg/m^3, S en m^2 y U en m/s, la potencia disponible Pd viene dada en unidades de W.

Debido a las acusadas variaciones temporales del viento, un modo de caracterizar el potencial eólico disponible en un determinado lugar es mediante la potencia media por unidad de área expuesta al viento, con lo que se hace independiente del tamaño de la máquina y prácticamente solo queda en función de la velocidad del viento como indica la **Ecuación** 3.

$$P_d/A = 0.5 * \rho * U^3 \quad (3)$$

Donde:

P_d = potencia disponible por unidad de área expuesta al viento.

Puesto que se debe cumplir la ecuación de conservación de la masa en el flujo de aire (condición de continuidad), no se puede extraer toda la potencia eólica disponible en el viento. "La cantidad de potencia eólica aprovechable depende, además de la potencia eólica disponible (velocidad del viento), de las características de funcionamiento de la máquina" (Pueyo et al., 2011, p.38).

4.2 Capítulo II: Aerogeneradores

El potencial eólico es aprovechable, entre otras aplicaciones, para la generación de energía eléctrica mediante el uso de aerogeneradores.

"Un aerogenerador es el dispositivo que transforma la energía cinética contenida en el viento en energía eléctrica en las condiciones idóneas para set utilizada" (Monge & Talayero, 2011, p. 69).

4.2.1 Clasificación

Los aerogeneradores se pueden clasificar de acuerdo a diversos criterios, según el autor considere, pueden ser rangos de potencia, tamaños, tecnologías y características externas.

Así, Villarubia (2013) clasifica estos dispositivos según su accionamiento:

- Aerogeneradores cuyo par motor se obtiene esencialmente de la fuerza de arrastre, como el aerogenerador de eje vertical tipo Savonius y los generadores multipala windmill o molinos americanos usados para el bombeo de agua.
- Aerogeneradores cuyo par motor se obtiene esencialmente de la fuerza de sustentación, como el aerogenerador de eje vertical tipo Darrieux y los modernos aerogeneradores de

eje horizontal tipo hélice (tripala o bipala). Sus álabes son de perfil aerodinámico en los que, de forma similar al ala de un avión, se desarrolla la fuerza de sustentación que produce el par motor en el eje del rotor (p. 104).

Otra manera de clasificar los aerogeneradores, según los criterios de Monge & Talayero (2011), son las siguientes consideraciones habituales:

- Según el tipo de eje: aerogenerador de eje horizontal, el cual está provisto de un rotor cuyo eje es sensiblemente paralelo a la dirección del viento; y aerogeneradores de eje vertical, en el que su eje es perpendicular a la dirección del viento y al suelo o superficie de montaje.
- Según la disposición del rotor: en las turbinas de eje horizontal el rotor puede estar situado a barlovento o a sotavento, es decir, con relación a la dirección del viento delante de la nacelle o góndola o detrás de la misma.
- Según el número de palas: el rotor puede poseer una, dos, tres y muchas palas.
- Según la velocidad del rotor: los aerogeneradores pueden tener velocidad de rotor fija girando a la velocidad nominal, manteniéndola mientras el aerogenerador esté conectado a la red y, aerogeneradores de velocidad variable que permiten que la velocidad del rotor varíe en régimen de funcionamiento.

4.2.1.1 Aerogeneradores de eje horizontal

El rotor de una turbina eólica captura la energía del viento y la convierte en energía mecánica. Las turbinas eólicas modernas tienen generalmente dos o tres palas construidas de fibra de vidrio y poliéster, las cuáles pueden girar alrededor de sus ejes. La energía mecánica es transmitida al generador a través de un sistema de transmisión que consiste generalmente de un eje rotor, freno mecánico y una caja multiplicadora que aumenta la velocidad de rotación del eje.

Las características descritas anteriormente encajan perfectamente con un aerogenerador de eje horizontal (HAWT), el cual se compone principalmente de una cimentación, torre, góndola (generador, freno mecánico, etc.), rotor con palas y buje, sistema de control y transformador (Villarubia, 2013).

4.2.2 Componentes de un aerogenerador de eje horizontal

La Figura 1 muestra un esquema de un HAWT tripala en el que se distinguen los siguientes elementos principales:

1. Bases y cimientos.
2. Punto de conexión a la estación transformadora de baja a alta tensión.

3. Torre de sustentación.
4. Escalera interior para acceso a la góndola.
5. Sistema de orientación del rotor hacia el viento.
6. Góndola.
7. Generador eléctrico (asíncrono o síncrono).
8. Anemómetro y veleta.
9. Freno para fijación del rotor.
10. Caja multiplicadora de velocidad (gear box).
11. Pala o álabe del rotor.
12. Punto de inserción de la pala en el buje.
13. Buje o nariz del aerogenerador.

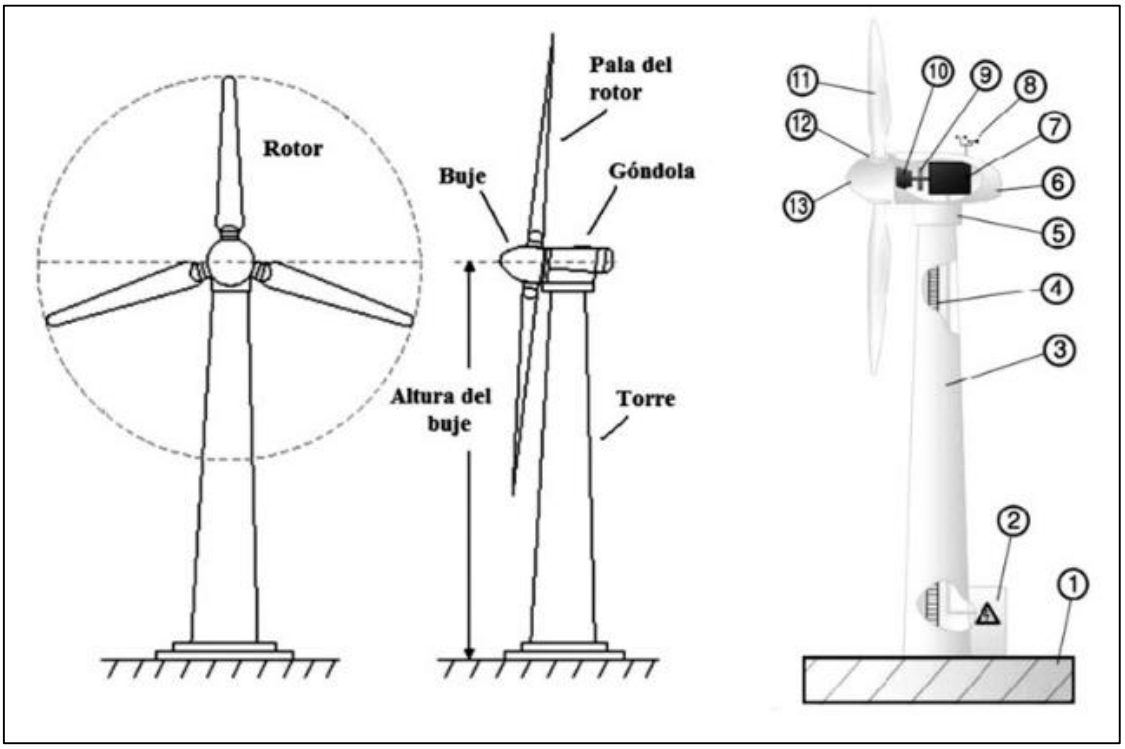

Figura 1. Esquema de un aerogenerador de eje horizontal (HAWT) tripala.
Fuente: Tomado de Villarubia (2013).

En la Figura 2 se aprecian los componentes alojados en el interior de la góndola, y con carácter general, se encuentran: eje, multiplicador, freno, generador eléctrico, transformador y parte del sistema de control, y en su parte superior externa la instrumentación, redundante, de medición de viento (anemómetro y veleta). El generador es el elemento fundamental en todas las tecnologías, pudiendo variar la presencia o posición del resto de los elementos (Monge & Talayero, 2011).

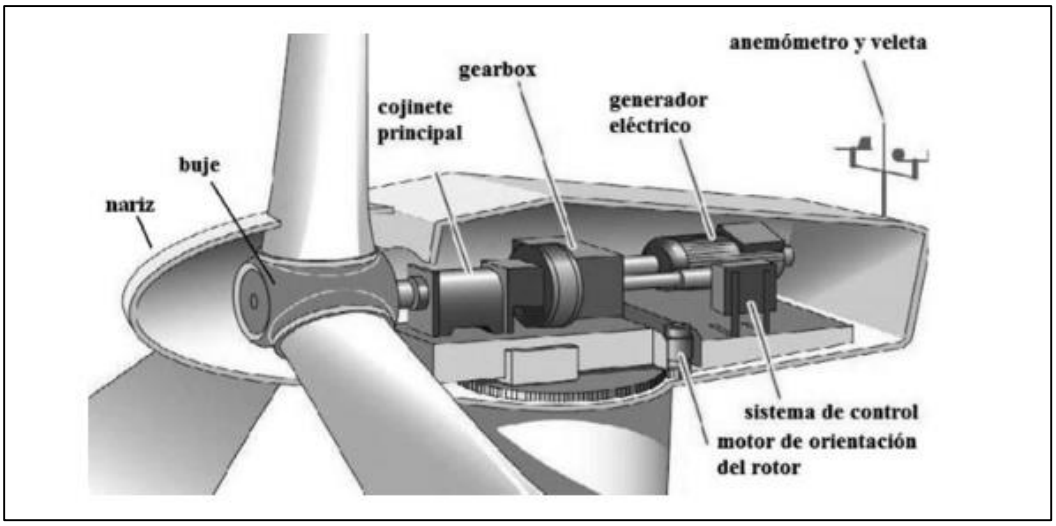

Figura 2. Vista del interior de la góndola en un HAWT.
Fuente: Tomado de Villarubia (2013).

4.2.3 Fallas en aerogeneradores

Martínez, Jiménez, Blanco & Sáenz (2014), mencionan que las averías más importantes que se pueden producir en un aerogenerador en cuanto al tiempo de parada y pérdida de producción son las relacionadas con alguno de los componentes principales del aerogenerador.

Debido al cambio continuo de los aerogeneradores con base en su configuración, tecnología y tamaño de los mismos, hace que se instalen en lugares de variadas condiciones, donde el mantenimiento de los mismos se dificulta por factores climáticos u orográficos. Esto hace que la necesidad de realizar predicciones de la fiabilidad, disponibilidad y vida útil de las máquinas aumente con el fin de evitar paradas inesperadas que supongan elevados costes de operación y mantenimiento (Jiménez, 2016).

Según Loría et al. (2017), los costos de operación y mantenimiento (O&M) en la industria eólica son altamente representativos, siendo necesario su disminución para obtener una eficiente producción eléctrica y su posterior comercialización.

En la Tabla 1 se muestran los costos de explotación generales para proyectos de eólicos, donde se evidencia que los costos por O&M son alrededor de 57%, cifra considerable con el resto de parámetros.

Tabla 1. *Costos de explotación generales para proyectos eólicos.*

Rubro		Operación y mantenimiento	Terrenos	Seguros e impuestos	Gestión y administración
Costo (%)		57	16	14	13

Fuente: Global Wind Energy Council, 2016

Las averías que afectan con más frecuencia a los aerogeneradores se puede visualizar en la Figura 3, aunque Jiménez (2016) señala que estos datos pueden llegar a ser engañosos y

que se refieren a una tasa anual de fallos, los principales están generados por el sistema eléctrico y la electrónica de control, sin embargo, estos desperfectos se solucionan relativamente rápido a los causados por el generador o por la multiplicadora, que necesitan más tiempo para que puedan repararse, paralizando la actividad de los aerogeneradores mucho más tiempo que otras averías comunes.

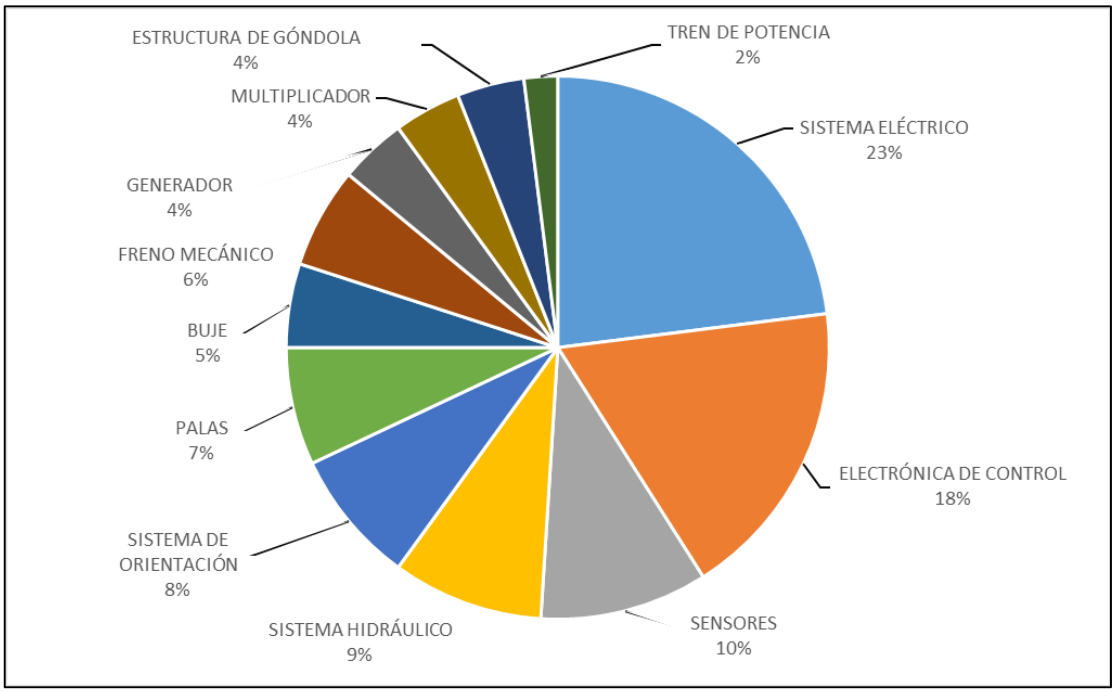

Figura 3. Frecuencia de fallas que afectan a los aerogeneradores.
Fuente: Tomado de Jiménez (2016).

4.3 Capítulo III: Mantenimiento Industrial

García (2003) define al mantenimiento como el "conjunto de técnicas destinadas a conservar equipos e instalaciones en servicio durante el mayor tiempo posible (buscando la más alta disponibilidad) y con el máximo rendimiento" (p. 16).

Para que una instalación de cualquier tipo continúe su funcionamiento normal es necesario contar con una organización de mantenimiento la cual se encargará de aplicar el tipo de mantenimiento adecuado.

4.3.1 Tipos de mantenimiento

A continuación, se describe cada uno de los tipos de mantenimiento clasificados por García (2003):

Mantenimiento correctivo: Es el conjunto de tareas destinadas a corregir los defectos que se van a presentando en los distintos equipos y que son comunicados al departamento de mantenimiento por los usuarios de los mismos.

Mantenimiento preventivo: Es el mantenimiento que tiene por misión mantener un nivel de servicio determinado por los equipos, programando las correcciones de sus puntos vulnerables en el momento más oportuno.

Mantenimiento Predictivo: Es el que persigue conocer e informar permanentemente el estado y operatividad de las instalaciones mediante el conocimiento de los valores de determinadas variables, representativas de tal estado y operatividad. Para aplicar este mantenimiento es necesario identificar variables físicas (temperatura, vibración, consumo de energía, etc.) cuya variación sea indicativa de problemas que puedan estar apareciendo en el equipo. Es el tipo de mantenimiento más tecnológico, pues requiere de medios técnicos avanzados, y de fuertes conocimientos matemáticos, físicos y técnicos.

Mantenimiento cero horas: Es el conjunto de tareas cuyo objetivo es revisar los equipos a intervalos programados bien antes de que aparezca ningún fallo, bien cuando la fiabilidad del equipo ha disminuido apreciablemente, de manera que resulta arriesgado hacer previsiones sobre su capacidad productiva. Dicha revisión consiste en dejar el equipo a cero horas de funcionamiento, es decir, como si el equipo fuera nuevo.

En estas revisiones se sustituyen o se reparan todos los elementos sometidos a desgaste. Se pretende asegurar, con gran probabilidad, un tiempo de buen funcionamiento fijado de antemano.

Mantenimiento en uso: es el mantenimiento básico de un equipo realizado por los usuarios del mismo. Consiste en una serie de tareas elementales (tomas de datos, inspecciones visuales, limpieza, lubricación, reapriete de tornillos) para las que no es necesario una gran formación, sino tan solo un entrenamiento breve. Este tipo de mantenimiento es la base del TPM (Total Productive Maintenance, Mantenimiento Productivo Total) (p. 32, 33)

4.4 Capítulo IV: Métodos Estadísticos

La Estadística es la ciencia que se encarga de recoger, organizar e interpretar los datos, es fundamental para muchas ramas de la ciencia y es esencial para interpretar los datos que se obtienen de la investigación científica (Garcia, Lopez, & Calvo, 2011)

Los métodos de recopilación y sintetización de la información de datos son muestreo y estadística descriptiva; el método estadístico que ayuda a obtener conclusiones a partir de datos se denomina estadística inferencial y el método que ayuda a medir la certeza o incertidumbre de un suceso o un experimento se llama probabilidad (Navidi, 2006).

4.4.1 Resumen Estadístico

La media de la muestra y la desviación estándar de la muestra son las dos cantidades más usadas en el resumen estadístico, el cual ayuda a tener una visión más clara de las características de una muestra constituida por una larga lista de números.

- *Media muestral o media aritmética.* Representa la suma de los números en la muestra, dividido entre la cantidad total de números que hay y se determina por la **Ecuación** 4.

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{4}$$

Donde:

$\bar{X}$ = media muestral

n = cantidad total de números existentes

X_i = cada una de las observaciones de la muestra

- *Desviación estándar.* Es la cantidad que mide el grado de dispersión en una muestra. Cuando la dispersión es grande, los valores de la muestra tenderán a alejarse de su media, pero cuando la dispersión es pequeña, los valores tenderán a acercarse a su media.

Existen desviaciones negativas y positivas que son calculadas como la distancia de cada valor de la muestra a la media de la muestra. Para que todas sean positivas se elevan al cuadrado, con lo que se obtiene las deviaciones al cuadrado, y a partir de ellas se puede calcular la varianza muestral como lo indica la **Ecuación** 5.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} (X_i - \bar{X})^2 \tag{5}$$

Donde:

s^2 = varianza muestral

Para obtener una medida de la dispersión en las mismas unidades que la de los valores de la muestra, se toma la raíz cuadrada de la varianza, esta cantidad se denomina desviación estándar muestral y se calcula por la **Ecuación** 6.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (X_i - \bar{X})^2} \tag{6}$$

Donde:

s = desviación estándar muestral

- *Datos atípicos.* Son puntos de una muestra que son mucho más grandes o pequeños que el resto y son el resultado de ingresar datos erróneamente como se ve en la Figura 4. Sin embargo, deben examinarse siempre y cualquiera de ellos que se encuentre debe corregirse o eliminarse, esto siempre que se esté seguro que se tratan de errores.

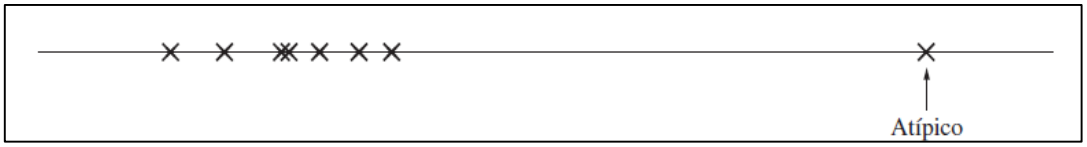

Figura 4. Conjunto de datos que contiene un dato atípico.
Fuente: Tomado de Navidi (2006).

- *Mediana muestral*. Representa una medida de tendencia central de los datos. Se calcula ordenando los valores de n números del más pequeño al más grande. La mediana es el número de en medio.
    - Si n es impar, la mediana muestral es el número en la posición $\frac{n+1}{2}$.
    - Si n es par, la mediana muestral representa el promedio de los números en las posiciones $\frac{n}{2}$ y $\frac{n}{2}+1$.
- *Cuartiles.* Así como la mediana divide la muestra a la mitad, los cuartiles la dividen tanto como sea posible en cuartos, por tanto, una muestra tiene tres de aquellos. El método más extendido para el cálculo es el siguiente:
    - Sea n el tamaño de la muestra.
    - Los valores de la muestra se ordenan del más pequeño al más grande.
    - Para encontrar el primer cuartil, calcule el valor $0.25(n+1)$. Si este es un entero, entonces el valor de la muestra en esta posición es el primer cuartil. Si no, tome entonces el promedio de los valores de la muestra de cualquier lado de ese valor.
    - El tercer cuartil se calcula de la misma manera, excepto que se usa el valor $0.75(n+1)$.
    - El segundo cuartil usa el valor $0.5(n+1)$ y es idéntico a la mediana.

4.4.2 Resúmenes gráficos

- *Diagramas de caja*. Un diagrama de caja constituye una gráfica que incluye la mediana, el primero y el tercer cuartil y cualquier dato atípico que se presente en una muestra.

Existe una pequeña terminología asociada a estos diagramas. El rango intercuartil es la diferencia entre el tercer y el primer cuartil. Debido a que 75% de los datos son menores que el tercer cuartil y que 25% de los datos son menores que el primer cuartil, la mitad de los datos

está entre el primero y el tercer cuartil. Por tanto, el rango intercuartil representa la distancia necesaria para atravesar la mitad de los datos de en medio.

Si denota IQR el rango intercuartil, entonces con el propósito de dibujar diagramas de caja, cualquier punto que está a más de 1.5 IQR por arriba del tercer cuartil, o que está a más de 1.5 IQR por debajo del primer cuartil, se considera un dato atípico.

En la Figura 5 se presenta un diagrama de caja con la indicación de sus partes. El diagrama consta de una caja cuyo lado inferior es el primer cuartil y el lado superior es el tercer cuartil. La mediana se dibuja como una línea horizontal. Los datos atípicos se grafican por separado y se indican por cruces en la figura. Los que se extienden desde la parte superior a la parte inferior de la caja son líneas verticales llamadas bigotes que terminan en los puntos más extremos que no son atípicos.

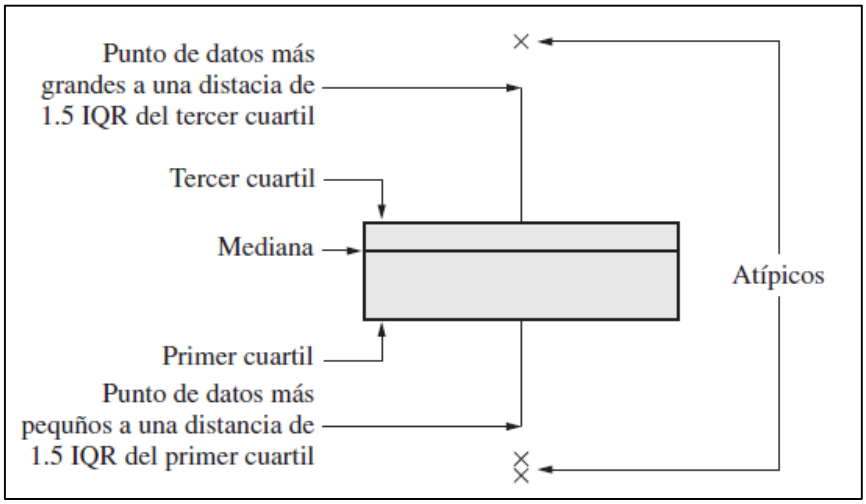

Figura 5. Anatomía de un diagrama de caja.
Fuente: Tomado de Navidi (2006).

La información de los párrafos anteriores se tomó de (Navidi, 2006).

- *Histogramas y polígonos de frecuencia.* Son dos representaciones gráficas de las distribuciones de frecuencias (Spiegel & Stephens, 2009). La distribución de frecuencias es la agrupación de datos en categorías mutuamente excluyentes que indican el número de observaciones en cada categoría.

Los histogramas frecuentemente clasifican los datos en varios contenedores o grupos de rango y cuentan cuántos puntos de datos pertenecen a cada uno de esos contenedores. Un contenedor está representado por un rectángulo que tiene *a)* sus bases sobre el eje horizontal con sus centros coincidiendo con las marcas de clase de longitudes iguales a la amplitud del intervalo de clase, y *b)* áreas proporcionales a las frecuencias de clase. En la Figura 6 se visualiza la representación de un histograma que ejemplifica la distribución de estatura de un grupo de personas.

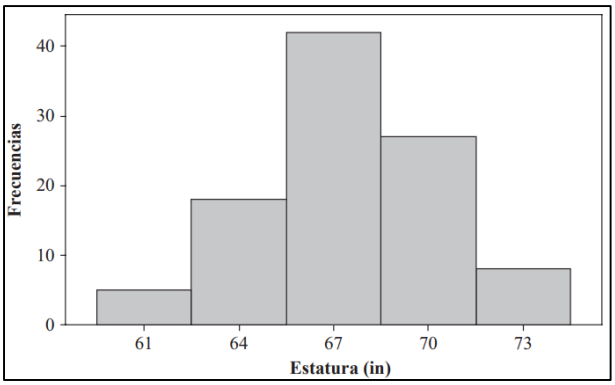

Figura 6. Ejemplo de histograma para la distribución de estaturas de un grupo de personas.
Fuente: Tomado de Spiegel & Stephens (2009).

Un polígono de frecuencias es una gráfica de línea que presenta las frecuencias de clase graficadas contra las marcas de cada clase. Se pueden obtener conectando los puntos medios de las partes superiores de los rectángulos de un histograma. En la Figura 7 se presenta el polígono de frecuencias para el ejemplo de estatura de un grupo de personas.

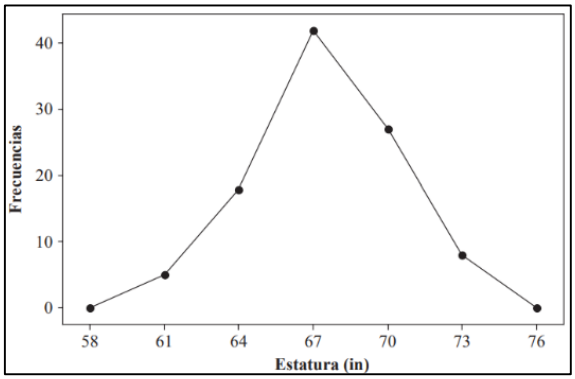

Figura 7. Ejemplo de polígono de frecuencias.
Fuente: Tomado de Spiegel & Stephens (2009).

En la Tabla 2 se presenta la diferencia de un histograma con la de un polígono de frecuencias considerando sus principales ventajas:

Tabla 2. *Diferencias entre Histograma y Polígono de frecuencias.*

Tipo	Ventajas
Histograma	Los rectángulos muestran cada clase de la distribución por separado.
	El área de cada rectángulo es proporcional al número total de observaciones en cada clase.
Polígono de frecuencias	Es más sencillo de elaborar que su histograma correspondiente
	Bosqueja con mayor claridad el perfil del patrón de comportamiento de los datos.

Fuente: *Adaptado de* (Khan Academy, n.d.)

- *Gráfica KDE*. Una gráfica de estimación de densidad kernel (KDE) es un método para visualizar la distribución de observaciones en un conjunto de datos, de forma análoga a un histograma.

KDE representa los datos utilizando una curva de densidad de probabilidad continua en una o más dimensiones. En relación con un histograma, KDE puede producir un gráfico menos desordenado y más interpretable, ya que, en lugar de utilizar contenedores discretos, suaviza las observaciones con un núcleo gaussiano (Seaborn, n.d.). La Figura 8 muestra un ejemplo de una gráfica KDE.

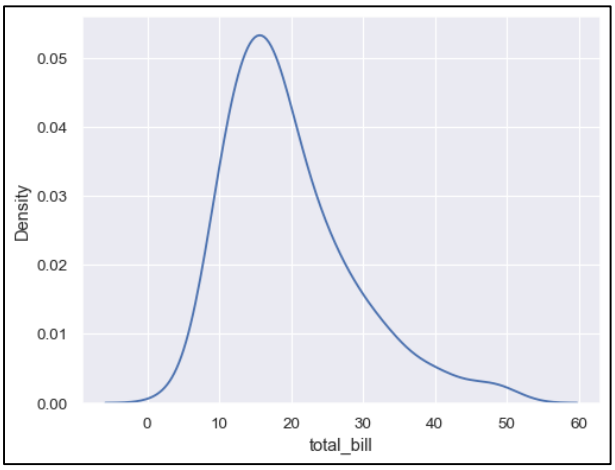

Figura 8. Representación de una gráfica KDE.
Fuente: Elaborado por el autor.

4.5 Capítulo V: Inteligencia Artificial

"La inteligencia artificial (IA) es una rama de las ciencias computacionales que se ocupa de los símbolos y métodos no algorítmicos para la resolución de problemas" (Ponce, 2010, p. 31).

Si bien, a lo largo de los años en los que la IA tuvo su desarrollo, no se ha logrado establecer un consenso general de lo que representa, dando lugar a diversas definiciones según la interpretación del autor. Ponce et al. (2014) recogen en su trabajo algunas de las definiciones iniciales de esta área dadas por diversos autores y se presentan en la Tabla 3.

Tabla 3. *Definiciones de Inteligencia Artificial*

Definiciones de IA	**Autor**
Estudio de la computación que observa que una máquina sea capaz de percibir, razonar y actuar.	(Winston, 1992)
Ciencia de la obtención de máquinas que logren hacer cosas que requerirían inteligencia si las hiciesen los humanos.	(Minsky, 1968)
Nuevo esfuerzo excitante que logre que la computadora piense… máquinas con mentes, en el sentido completo y literal.	(Haugeland, 1985)
Rama de la ciencia computacional preocupada por la automatización de la conducta inteligente.	(Luger and Stubblefield, 1993)
Máquina Inteligente es la que realiza el proceso de analizar, organizar, y convertir los datos en conocimiento, donde el conocimiento del sistema es información estructurada adquirida y aplicada para reducir la ignorancia o la incertidumbre sobre una tarea específica a realizar por esta.	(Pajares y Santos, 2006)

Fuente: Elaborado por el autor a partir de Ponce et al. (2014)

La IA abarca y está compuesta de diversos tópicos, y de la misma manera que su definición, existen múltiples formas de clasificarlos. Según Ponce, (2010), dentro de las IA's se pueden encontrar tres grandes ramas que la componen: lógica difusa, redes neuronales artificiales y algoritmos genéticos.

En Point (2018), se menciona que los ejemplos de IA incluyen el aprendizaje, el razonamiento y la autocorrección, siendo las principales aplicaciones el reconocimiento del habla, los sistemas expertos y el reconocimiento de imágenes y la visión artificial.

Una de las ramas de la IA es el Aprendizaje Automático (Machine Learning) que se ocupa de los sistemas y algoritmos que pueden aprender nuevos datos y patrones de datos. A su vez, el aprendizaje automático incluye una sección denominada Aprendizaje Profundo (Deep Learning).

En la Figura 9 se presenta un diagrama de Venn con la relación entre Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo, que como (Kim, 2017) señala "El Aprendizaje Profundo es un tipo de Aprendizaje Automático, y el Aprendizaje Automático es un tipo de Inteligencia Artificial."

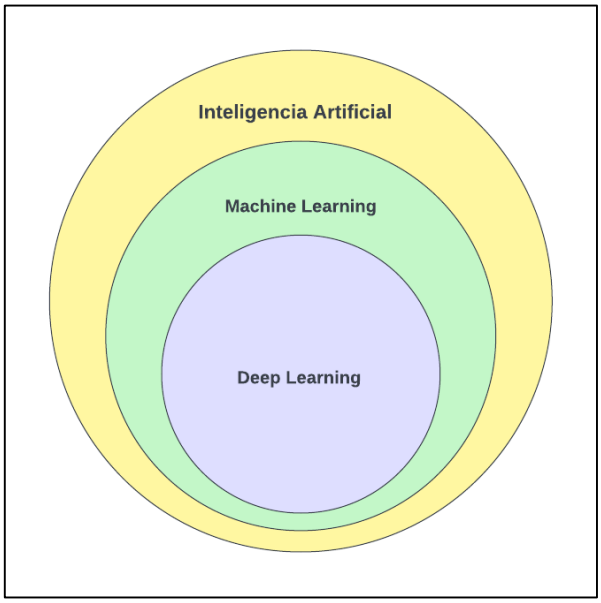

Figura 9. Relación entre IA, Aprendizaje Automático y Aprendizaje Profundo.
Fuente: Elaborado por el autor.

4.5.1 Machine Learning (Aprendizaje Automático)

Mitchell, (1997) describe de manera amplia y técnica el aprendizaje automático con la finalidad de incluir cualquier programa de computadora. Su definición es la siguiente:

> Se dice que un programa de ordenador aprende de la experiencia E con respecto a una clase de tareas T y una medida de desempeño P, si su rendimiento en las tareas de T, medido por P, mejora con la experiencia E. (p. 2)

"Se han desarrollado diferentes tipos de técnicas de Aprendizaje Automático para resolver problemas en varios campos" (Kim, 2017). Géron, (2019) clasifica estas técnicas en los siguientes tipos dependiendo del método de entrenamiento:

- *Aprendizaje Supervisado*. El conjunto de entrenamiento que se alimenta al algoritmo incluye las soluciones deseadas, llamadas etiquetas.
- *Aprendizaje No Supervisado:* Los datos de entrenamiento no están etiquetados, el sistema intenta aprender sin un maestro.
- *Aprendizaje Semisupervisado:* Algunos algoritmos pueden tratar con datos parcialmente etiquetados.
- *Aprendizaje por refuerzo:* El sistema de aprendizaje llamado agente en este contexto, puede observar el entorno, seleccionar y realizar acciones, y obtener recompensas a cambio. A continuación, debe aprender por sí mismo cuál es la mejor estrategia para obtener la mayor recompensa a lo largo del tiempo.

  En el presente documento únicamente se tratará el aprendizaje supervisado.

4.5.1.1 Aprendizaje Supervisado

El aprendizaje supervisado corresponde a la situación en que se tiene una variable de salida, ya sea cuantitativa o cualitativa, que se desea predecir basándose en un conjunto de características. Se establece un modelo que permite relacionar las características con la variable de salida. Luego se considera un conjunto de datos de entrenamiento en los cuales se observan tanto los valores de la variable de salida como de las características para determinados individuos (personas u otros). Usando tales datos se ajustan los parámetros del modelo, con lo cual es posible predecir valores de la variable de salida para nuevos individuos (Ponce et al., 2014).

El objetivo del aprendizaje supervisado es la predicción: Dado el valor de un vector de entrada X, generar una buena predicción $\hat{Y}$ de la salida Y como ilustra la Figura 10.

Figura 10. Método de Aprendizaje Supervisado.
Fuente: Tomado de Ponce et al., 2014.

Los dos tipos más comunes de aplicación de aprendizaje supervisado son la clasificación y la regresión.

La clasificación es la aplicación más frecuente del aprendizaje automático. Se centra en encontrar literalmente las clases a las que pertenecen los datos. Algunos ejemplos son:

- Servicio de filtrado de correo basura (spam): clasifica los correos electrónicos por regulares o spam.
- Servicio de reconocimiento de dígitos: clasifica la imagen de los dígitos dentro de 0-9.
- Servicio de reconocimiento de caras: clasifica la imagen de la cara en uno de los usuarios registrados.

La regresión en cambio no determina la clase. En su lugar estima un valor. Como ejemplo, si se tiene un conjunto de datos de edad e ingresos y se quiere encontrar el modelo que estima los ingresos por edad, se convierte en un problema de regresión.

El presente trabajo únicamente abarca el estudio del método de clasificación. En la Tabla 4 se exponen algunos de los algoritmos empleados en la tarea de clasificación resumidos por Huertas (2020), donde se detalla en qué consiste cada uno, así como sus limitaciones, fortalezas y los hiperparámetros de ajuste.

Tabla 4. *Algoritmos de Aprendizaje Automático para clasificación.*

Algoritmo	**Descripción**	**Fortalezas**	**Limitaciones**	**Hiperparámetros**
Decision Tree	Descompone un conjunto de datos en subconjuntos más pequeños con un aumento en la profundidad del árbol, el objetivo aumentar la predicción por medio de nodos de decisión.	Representación visual de todos los resultados posibles, requiere menos limpieza de datos, no está influenciado por valores atípicos, maneja variables numéricas y categóricas.	El cálculo puede ser más complejo lo que implica un mayor tiempo para entrenar el modelo, un pequeño cambio en los datos puede causar un gran cambio en la estructura del árbol.	Profundidad del árbol, número mínimo de muestras del nodo, criterio.
Random Forest y **Extra Trees**	Modelos compuestos por muchos árboles de decisión, al entrenar cada árbol se aprende de una muestra aleatoria de los puntos de datos y de un subconjunto de características. La diferencia entre Random Forest y Extra Trees es que muestrea sin reemplazo y los nodos se distribuyen en divisiones aleatorias, no mejores divisiones como Random Forest.	Las predicciones finales del bosque aleatorio se hacen promediando las predicciones de cada árbol individual reduciendo el problema de sobreajuste y varianza. Para el caso de Extra Trees dado que las divisiones se eligen al azar para cada característica, es menos costoso desde el punto de vista computacional que un Random Forest.	Genera muchos árboles lo que lo hace costoso computacionalmente, requiere más tiempo para entrenar en comparación con los árboles de decisión. Puede ajustarse en exceso a los conjuntos de datos que son particularmente ruidosos.	Número de árboles, profundidad del árbol, muestras necesarias para una hoja y divisiones de nodo interno.
Bagging	Algoritmo que genera varios subconjuntos de datos a partir de una muestra de entrenamiento elegida al azar con reemplazo.	Se centrará principalmente en obtener un modelo de conjunto con menos varianza, para producir modelos fuertes con menor sesgo.	Introduce una pérdida de interpretabilidad de un modelo, puede ser costoso computacionalmente.	Número de árboles y número máximo de muestras con reemplazo.
XGBoost	Algoritmo que recientemente ha dominado el aprendizaje automático aplicado, es una implementación de Gradient Boosting para maximizar velocidad de entrenamiento y el rendimiento de modelo.	Utilización de todos los núcleos de la CPU durante el entrenamiento, optimización de recursos de memoria y computación distribuida lo que permite manejar grandes conjuntos de datos.	La alta flexibilidad da como resultado muchos hiperparámetros que interactúan fuertemente en el comportamiento del modelo.	Número de árboles, Profundidad del árbol, taza de aprendizaje y muestreo de filas.
Gradient Boosting	Algoritmo de conjunto con optimización numérica donde el objetivo es minimizar la pérdida del modelo agregando secuencialmente árboles de decisión.	Excelente precisión predictiva, flexibilidad para ajustarse a diferentes clases de datos, las predicciones se hacen por mayoría de votos de los alumnos débiles.	El aumento del gradiente continuará mejorando con el propósito de minimizar todos los errores, lo que puede causar un excesivo sobreajuste.	Número de árboles, profundidad del árbol, taza de aprendizaje y muestreo de filas.

Algoritmo	Descripción	Fortalezas	Limitaciones	Hiperparámetros
Logistic Regresion	Algoritmo que se utiliza para problemas de clasificación binaria, la base de la regresión logística es la función logística (sigmoid) que toma cualquier número de valor real y le asigna a un valor entre 0 y 1.	La regresión logística es un algoritmo de clasificación simple pero muy efectivo, tiene una relación muy estrecha con las redes neuronales. El tiempo de entrenamiento es menor que otros algoritmos.	Dificultad para capturar relaciones complejas y que tiene una superficie de decisión lineal, en los conjuntos de datos de alta dimensión puede generar sobreajuste.	Regularización (C) y penalización a la función de pérdida L1 y L2.
Naive Bayes	Algoritmo de clasificación que aplica explícitamente el teorema de Bayes bajo el supuesto que todas las variables observadas son independientes.	Baja prospección al sobreajuste, entrenamiento y predicción rápida, uso modesto de capacidad de CPU y memoria ya que no hay gradientes o actualizaciones iterativas de parámetros para calcular.	El rendimiento es sensible a los datos asimétricos, es decir, cuando los datos de entrenamiento no son representativos de las distribuciones de clases en la población general.	No se ajustaron hiperparámetros
Support Vector Machine	Los SVM encuentran una línea o hiperplano entre diferentes clases de datos, calculan un límite de margen máximo que conduce a una partición homogénea de todos los puntos de datos.	Es útil tanto para datos separados linealmente como no separables linealmente, es eficaz en los casos en que varias dimensiones son mayores que la cantidad de muestras. Eficiente uso de memoria.	No funciona muy bien cuando el conjunto de datos tiene mucho ruido, elegir el kernel y los parámetros correctos puede ser costoso computacionalmente.	Parámetros núcleo, tipo núcleo (kernel)
k-Nearest Neighbors	Utiliza la similitud de características para predecir el clúster en el que caerá el nuevo punto. La idea principal es que el valor o la clase de una observación está determinado por las observaciones que lo rodean.	No hace una suposición sobre el patrón de distribución de datos subyacente, se agiliza el tiempo de entrenamiento ya que almacena el conjunto de entrenamiento y aprende de él solo al momento de hacer predicciones.	Computacionalmente costoso, los datos deben pre procesarse y escalarse, las observaciones se usarán solo en el momento de la predicción por lo que es un paso costoso, sensible a datos ruidosos y atípicos.	Número de vecinos

Fuente: Adaptado de Huertas (2020).

4.5.2 Redes Neuronales Artificiales

Para Ponce (2010), las RNA se definen como sistemas de mapeos no lineales cuya estructura se basa en principios observados en los sistemas nerviosos de humanos y animales. Constan de un número grande de procesadores simples ligados por conexiones con pesos. Las unidades de procesamiento se denominan neuronas. Cada unidad recibe entradas de otros nodos y genera una salida simple escalar que depende de la información local disponible, guardada internamente o que llega a través de las conexiones con pesos.

4.5.2.1 Elementos de una red neuronal artificial

Los elementos básicos de una RNA son:

- Conjunto de unidades de procesamiento (neuronas).
- Conexiones entre unidades (asociado a cada conexión un peso o valor)
- Funciones de salida o activación para cada unidad de procesamiento.

Un esquema básico de una red neuronal artificial se observa en la Figura 11.

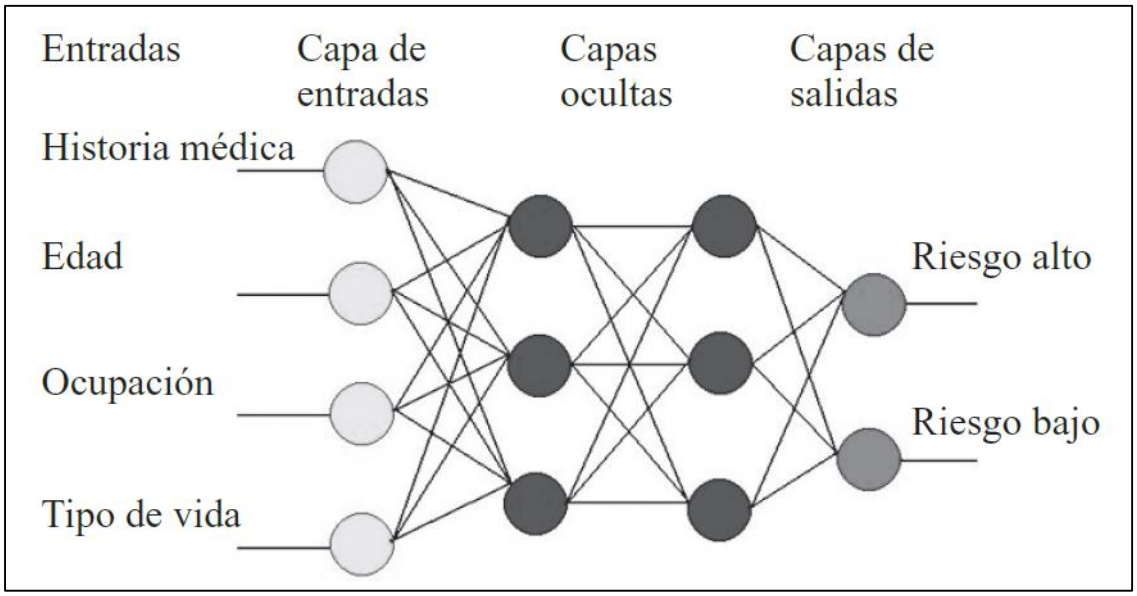

Figura 11. Esquema básico de una RNA.
Fuente: Tomado de P. Ponce (2010).

4.5.2.2 Perceptrón Multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés) es una red neuronal artificial simple que tiene la capacidad de resolver problemas que no son linealmente separables. Está compuesto de una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa, excepto la de salida, incluye una neurona de polarización y está totalmente conectada a la capa siguiente (Géron, 2019).

Los MLP también pueden utilizarse para tareas de clasificación. Para un problema de clasificación binaria, sólo se necesita una neurona de salida que utilice la función de activación

logística: la salida será un número entre 0 y 1, que se puede interpretar como la probabilidad estimada de la clase positiva. La probabilidad de la clase negativa es igual a uno menos ese número.

La función de activación es la regla que logra transmitir la información generada por la combinación lineal de los pesos y las entradas, es decir son la manera de transmitir la información por las conexiones de salida. A breves rasgos, es un filtro que modifica el valor resultado e impone un límite que se debe sobrepasar un umbral para poder proseguir a otra neurona.

Uno de los algoritmos más común para entrenar las redes neuronales es la retro-propagación. Géron (2019), lo resume así: para cada instancia de entrenamiento, el algoritmo hace primero una predicción (paso hacia adelante) y mide el error, luego recorre cada capa en sentido inverso para medir la contribución al error de cada conexión (paso hacia atrás) y, por último, ajusta los pesos de las conexiones para reducir el error (paso de descenso del gradiente).

4.5.3 Deep Learning

El Deep Learning o Aprendizaje Profundo trata con una red neuronal con más de dos capas. La red neuronal que tiene una simple capa oculta es llamada red neuronal superficial. Una red neuronal multi-capa que contiene dos o más capas ocultas es llamada una red neuronal profunda, más usada en aplicaciones prácticas en la actualidad (Kim, 2017).

En la Tabla 5 se resume las ramas de las redes neuronales en función de la arquitectura de capas.

Tabla 5. *Arquitectura de las redes neuronales en función de las capas ocultas.*

Red Neuronal de Capa Simple		Capa de Entrada – Capa de Salida
Red Neuronal Multi-Capa	Red Neuronal Superficial	Capa de Entrada – Capa Oculta - Capa de Salida
	Red Neuronal Profunda	Capa de Entrada – Capas Ocultas - Capa de Salida

Fuente: Adaptado de Kim (2017).

Las redes neuronales han tenido que trascender arquitectónicamente desde las redes anteriores (junto con mucha más potencia de procesamiento) antes de mostrar los resultados espectaculares que se han visto en los últimos años (Patterson & Gibson, 2017).

A continuación, se exponen algunas de las facetas de esta evolución de las redes neuronales:

- Más neuronas que las redes anteriores.
- Formas más complejas de conectar capas/neuronas en las NN.
- Explosión en la cantidad de potencia de cálculo disponible para el entrenamiento.

- Extracción automática de características.

Una de las ventajas del aprendizaje profundo es la extracción automática de características, que es el proceso de la red para decidir qué características de un conjunto de datos pueden utilizarse como indicadores para etiquetar esos datos de forma fiable, algo que con los algoritmos convencionales se tiene que realizar manualmente, tomando mucho tiempo y esfuerzo.

Los ejemplos de aprendizaje profundo incluyen aplicaciones como el reconocimiento de imágenes y el reconocimiento del habla, y los dos tipos más importantes de redes neuronales profundas son:

- Redes Neuronales Convolucionales (CNN)
- Redes Neuronales recurrentes (RNN)

4.5.3.1 Redes Neuronales Convolucionales (CNN)

Las CNNs están diseñadas para procesar datos a través de múltiples capas. Este tipo de redes neuronales es usado principalmente en aplicaciones como reconocimiento de imágenes o reconocimiento de rostros. La principal diferencia entre la CNN y cualquier otra red neuronal ordinaria, es que la CNN toma las entradas como una matriz bidimensional y opera directamente sobre las imágenes en lugar de centrarse en la extracción de características en la que se centran otras redes neuronales.

Sin embargo, otra aplicación que se da a las CNNs es en la clasificación de series de tiempo como menciona y demuestra Huertas (2020) en su trabajo.

4.6 CAPÍTULO VI: Métodos de muestreo de datos

En problemas de clasificación binaria es común que se presente desbalance entre clases. La baja fiabilidad de los resultados de la clasificación a partir de datos desequilibrados se produce debido al sesgo[1] del modelo predictivo hacia la clase mayoritaria, mientras que la clase minoritaria es casi ignorada o asumida como ruido por estar representada por muy pocas instancias, por consiguiente la precisión para la clase minoritaria suele ser mucho menor que para la que tiene el mayor número de muestras, lo que indica un modelo de predicción poco fiable, ya que la clase minoritaria es a menudo la más interesante en el área de aplicación, lo que agrava aún más el problema (Kraiem, 2020).

Según Rodríguez (2017), las soluciones para el problema de desbalance entre clases pueden dividirse en dos tipos: soluciones a nivel algorítmico y soluciones a nivel de datos. Las

[1] *Sesgo:* orientación o dirección que toma un asunto. / Estad. Error sistemático en el que se puede incurrir cuando al hacer muestreos o ensayos se seleccionan o favorecen unas respuestas frente a otras. /Estad. Diferencia entre el valor esperado de un estimador y el verdadero valor del parámetro.

soluciones a nivel algorítmico modifican los algoritmos para mejorar la clasificación en la clase minoritaria. Por otra parte, las soluciones a nivel de datos utilizan métodos de remuestreo para balancear el conjunto de datos.

Para Kraiem (2020), la forma más fácil de manejar el problema de desequilibrio de clases es la aplicación de los métodos de remuestreo básicos como son el submuestreo aleatorio, el sobremuestreo aleatorio y la combinación de ambos.

El submuestreo aleatorio elimina ejemplos de la clase mayoritaria para equilibrar el conjunto de datos, siendo su principal desventaja la pérdida de información potencialmente útil que puede ser importante para inducir a los clasificadores, conduciendo a un menor rendimiento de los modelos, por lo que es adecuado para conjuntos de datos con bajo desequilibrio.

El sobremuestreo aleatorio equilibra el conjunto de datos replicando los ejemplos de la clase minoritaria, con la ventaja de que no produce pérdida de información, sin embargo, la desventaja es que conduce a un sobreajuste e introduce un coste computacional adicional si el índice de desequilibrio del conjunto de datos es alto. También aumenta la proporción de solapamiento entre instancias de diferentes clases, debido a la duplicidad de datos.

A continuación, se presentan una serie de técnicas para el remuestreo de clases, enfocadas en mejorar los métodos de remuestreo aleatorio, o a su vez, disminuir sus desventajas. De acuerdo con Brownle,(2020a), se pueden clasificar de la siguiente manera:

4.6.1 Métodos de sobremuestreo

- *SMOTE: Synthetic Minority Over sampling Technique.* Es un método de sobremuestreo propuesto para abordar el problema del sobreajuste, haciendo más general el límite de decisión de la clase minoritaria. Este método, en lugar de replicar las instancias de la clase minoritaria, genera nuevos ejemplos interpolando los ya existentes, siendo especialmente eficiente para conjuntos de datos altamente desequilibrados y su rendimiento puede mejorarse combinándolo con técnicas de submuestreo.

Existen algunas extensiones de SMOTE que son más selectivas en cuanto a las muestras de la clase minoritaria que proporcionan la base para generar nuevos ejemplos sintéticos, algunas de las cuales son:

- *SMOTE Frontera (Borderline-SMOTE).* En esta alternativa, los elementos creados sintéticamente se localizan en la frontera, considerando tales casos objetivo del algoritmo, aquellos elementos de la clase minoritaria que tienen más de la mitad de sus k vecinos más cercanos en la clase mayoritaria.
- *Borderline-SMOTE SVM.* Es una extensión de Borderline-SMOTE en la se utiliza un algoritmo SVM en lugar de un KNN para identificar ejemplos más clasificados en el

límite de decisión. Se utiliza una SVM para localizar el límite de decisión denotado por los vectores de soporte y los ejemplos de la clase minoritaria que están cerca de estos vectores se convierten en el foco para generar ejemplos sintéticos.

- *Adaptative Synthetic Sampling.* El muestreo sintético adaptativo (ADASYN) es otro enfoque que consiste en generar muestras sintéticas inversamente proporcionales a la densidad de los ejemplos de la clase minoritaria, es decir, generar más ejemplos sintéticos en las regiones del espacio de características donde la densidad de ejemplos minoritarios es baja, y menos o ninguno donde la densidad es alta.

4.6.2 Métodos de Submuestreo

Hay muchos tipos diferentes de técnicas de submuestreo y pueden agruparse en las que seleccionan las muestras que se mantienen en el conjunto de datos transformado, las que seleccionan muestras que se eliminan y las híbridas que combinan ambos métodos.

4.6.2.1 Métodos que seleccionan muestras a conservar

- *Near Miss Undersampling.* Método de submuestreo que se basa en la distancia entre las muestras de la clase minoritaria y mayoritaria. Existen tres versiones de esta técnica: NearMiss-1, NearMiss-2 y NearMiss3.

  NearMiss-1 selecciona muestras de la clase mayoritaria que tengan la menor distancia media a las tres muestras más cercanas de la clase minoritaria. NearMiss-2 selecciona las muestras de la clase mayoritaria que tengan la menor distancia media a las tres muestras más alejadas de la clase minoritaria. NearMiss-3 consiste en seleccionar un número determinado de muestras de la clase mayoritaria por cada muestra de la clase minoritaria que estén más cerca. En este caso, la distancia se determina en el espacio de características utilizando la distancia euclidiana o similar.

- *Condensed Nearest Neighbor Rule Undersampling (CNN).* Es una técnica de submuestreo que busca un subconjunto de una colección de muestras que no suponga una pérdida de rendimiento del modelo, lo que se denomina un conjunto mínimamente coherente. Fue propuesto para reducir los requisitos de memoria del algoritmo k-Nearest Neigbors (KNN). El subconjunto se compone de todas las muestras del conjunto minoritario y sólo de las muestras del conjunto mayoritario que no se pueden clasificar correctamente.

4.6.2.2 Métodos que seleccionan las muestras a borrar

- *Tomek Liks.* Este método es una regla que encuentra pares de muestras, una de cada clase; juntos tienen la menor distancia euclidiana entre sí en el espacio de características. Esto significa que, en un problema de clasificación binaria con clases 0 y 1, un par

tendría un ejemplo de cada clase y serían los vecinos más cercanos en el conjunto de datos.

- *Edited Nearest Neighbors Rule for Undersampling (ENN).* Es otra regla para encontrar muestras ambiguas y ruidosas en un conjunto de datos y consiste en utilizar k=3 vecinos más cercanos para localizar los ejemplos de un conjunto de datos que están mal clasificados y eliminarlos antes de aplicar una regla de clasificación k=1.

4.6.2.3 Métodos de conservación y eliminación

- *One Side Selection.* La estrategia de selección unilateral (OSS) combina las técnicas Tomek Links y CNN. Específicamente, Tomek Links son puntos ambiguos en el límite de la clase y se identifican y eliminan en la clase mayoritaria. El método CNN es entonces usado para eliminar muestras redundantes de la clase mayoritaria que están lejos de la frontera de decisión.
- *Neighborhood Cleaning Rule (NCR).* Es una técnica que combina tanto la regla CNN para remover muestras redundantes y la regla ENN para remover ruido o muestras ambiguas. En NCR se eliminan menos muestras redundantes y se presta más atención a la limpieza de las que se conservan. La razón es que se centra menos en mejorar el equilibrio de la distribución de clases y más en la calidad de las muestras que son retenidas en la clase mayoritaria.

4.6.3 Sobremuestreo y submuestreo

Se pueden combinar los métodos de sobremuestreo y submuestreo aleatorio para igualar las clases. Brownlee (2020a), menciona que el orden en que se apliquen este método no tiene relevancia, pero si hay que tener en cuenta que al aplicar un primer remuestreo se haga un determinado porcentaje, dado que con el segundo método se logra igualar las clases. El mismo procedimiento se puede aplicar con la combinación de SMOTE y el submuestreo aleatorio.

Kraiem (2020), agrega que dos técnicas estándar híbridas para el remuestreo son incluyen SMOTE + Tomek y SMOTE + ENN, donde SMOTE se utiliza para sobremuestrear la clase minoritaria, mientras que Tomek y ENN, respectivamente, se utilizan para submuestrear la clase mayoritaria.

Como última consideración, de acuerdo con Brownlee (2020a) y Kraiem (2020), el remuestreo de datos se lleva a cabo sólo en el conjunto de entrenamiento que es el conjunto utilizado por un algoritmo para aprender un modelo. No se realiza en el conjunto de prueba, ya que el objetivo es asegurar que no haya problemas de sobreajuste y que los modelos puedan aplicarse a ejemplos reales diferentes del conjunto de entrenamiento.

4.7 CAPÍTULO VII: Medidas de desempeño del modelo de predicción

Para evaluar el desempeño de un clasificador supervisado se han propuesto diferentes medidas. Sin embargo, no todas las medidas de desempeño son útiles cuando se presenta el problema de desbalance entre clases, ya que algunas dan valores altos, aun cuando, no se haya clasificado correctamente ningún objeto de la clase minoritaria, la cual, es usualmente de mayor importancia.

Todas las medidas de desempeño pueden calcularse con base en la matriz de confusión, obtenida a partir de un conjunto de prueba que es un conjunto de datos etiquetados independiente del conjunto de entrenamiento, el cual es utilizado para evaluar el modelo de clasificación.

En la Tabla 6 se representa la matriz de confusión para dos clases, donde las columnas están asociadas a las clases asignadas por el clasificador y los renglones están asociadas a la etiqueta correcta de los objetos.

Tabla 6. *Matriz de confusión*

Número Total de Observaciones		**Predicción**	
		Negativo (N)	*Positivo (P)*
Observación	*Negativo (Clase mayoritaria)*	TN	FP
	Positivo (Clase minoritaria)	FN	TP

Fuente: Adaptado de Brownlee (2020a)

En la Tabla 5, TP (True Positive o Verdadero Positivo) y TN (True Negative o Verdadero Positivo), son el número de objetos correctamente clasificados de la clase minoritaria y mayoritaria respectivamente y, por otro lado, FP (False Positive o Falso Positivo) y FN (False Negative o Falso Negativo) son el número de objetos mal clasificados de la clase minoritaria y mayoritaria respectivamente (Rodríguez, 2017).

Las principales métricas de evaluación que se derivan de la matriz de confusión, de acuerdo con (Brownlee, 2020a) se pueden dividir en tres grupos útiles:

1. Métricas de umbral
2. Métricas de clasificación
3. Métricas de probabilidad

A continuación, se abordan los dos primeros grupos, ya que son de interés en la tarea de clasificación.

4.7.1 Métricas de Umbral para Clasificación Desequilibrada

Las métricas de umbral son aquellas que cuantifican los errores de predicción de la clasificación. Están diseñados para resumir la fracción, la proporción o la tasa de cuando una clase predicha no coincide con la clase esperada en un conjunto de datos reservado.

La métrica de umbral más utilizada es la Exactitud de la clasificación, que calcula el porcentaje de predicciones correctas entre todas las predicciones realizadas y se determina mediante la **Ecuación** 7:

$$\text{Exactitud} = \frac{\text{Predicciones correctas}}{\text{Predicciones totales}} = \frac{TN + TP}{TN + FP + FN + TP} \tag{7}$$

El complemento de precisión de clasificación se llama error de clasificación, calculada con la **Ecuación** 8:

$$Error = \frac{Predicciones\ incorrectas}{Predicciones\ totales} = \frac{FP + FN}{TN + FP + FN + TP} \tag{8}$$

La métrica Exactitud es inapropiada para la clasificación desequilibrada, puesto que se puede lograr una alta precisión (o un error bajo) mediante un modelo sin habilidades que solo predice la clase mayoritaria.

Existen dos grupos de métricas que pueden ser usadas para la clasificación desbalanceada porque se enfocan en una clase, estas son sensibilidad-especificidad y precisión-recall.

4.7.1.1 Métricas Sensibilidad-Especificidad

La Sensibilidad se refiere a la tasa de verdaderos positivos y resume lo bien que se predijo la clase positiva y se calcula a través de la **Ecuación** 9.

$$Sensibilidad = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \tag{9}$$

La Especificidad es el complemento a la sensibilidad, o la tasa de verdaderos negativos, y resume lo bien que se predijo la clase negativa y es calculada con la **Ecuación** 10.

$$Especificidad = \frac{Verdaderos\ Negativos}{Falsos\ Positivos + Verdaderos\ Negativos} \tag{10}$$

Para la clasificación desbalanceada, la sensibilidad podría ser más interesante que la especificidad. La Sensibilidad y la Especificidad pueden ser combinadas dentro de una única puntuación que equilibra ambas preocupaciones, denominado G-mean y determinado por la **Ecuación** 11.

$$G - mean = \sqrt{Sensibilidad \times Especificidad} \tag{11}$$

4.7.1.2 Métricas Precisión y Recall

La Precisión resume la fracción de los ejemplos asignados a la clase positiva que pertenecen a la clase positiva, dicho de otra manera, es el valor predictivo positivo (PPV por sus siglas en inglés). Esta métrica se calcula con la **Ecuación** 12.

$$Precisión\ (PPV) = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \tag{12}$$

La contraparte de PPV es el valor predictivo negativo o NPV que cuantifica las observaciones negativas verdaderas de todas las estimadas por el modelo de predicción. Está dado por la **Ecuación** 13:

$$NPV = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Negativos} \tag{13}$$

El Recall por su parte resume lo bien que la clase positiva fue predicha y es el mismo cálculo que para la Sensibilidad como demuestra la **Ecuación** 14.

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \tag{14}$$

La Precisión y el Recall se pueden combinar en una sola puntación que trata de equilibrar ambas preocupaciones y es llamada F-score o F-score y se determina por la **Ecuación** 15.

$$F - score = \frac{2\ x\ Precisón \times Recall}{Precisión + Recall} \tag{15}$$

La medida F-score es una métrica popular para la clasificación desbalanceada. La medida Fbeta-score (o $F\beta$) es una contemplación de F-score donde el balance de la precisión y el recall en el cálculo de la media armónica está controlado por un coeficiente llamado beta (β) como indica la **Ecuación** 16.

$$F\beta = \frac{(1 + \beta^2)\ \times Precisión\ \times\ Recall}{\beta^2\ \times Precisión + Recall} \tag{16}$$

Donde β es el coeficiente que controla el peso o el balance entre la Precisión y el Recall, siendo según Huertas (2020) los valores comunes asignados a β los siguientes:

- Medida F1 (β = 0.5): En esta medida se le asigna más peso a la Precisión y menos peso a el Recall, con este coeficiente se parte del supuesto que los Falsos Positivos son más importantes, para el caso de predicción de fallas en mantenimiento, se utiliza esta métrica cuando los costos de ejecutar múltiples mantenimientos preventivos o inspecciones son de mayor impacto que las clasificaciones erróneas que puedan generar un paro funcional de la máquina sin previo aviso.

- Medida F2 (ß = 1): Equilibra el peso entre Precisión y el Recall, es decir esta medida los falsos negativos y falsos positivos son igualmente importantes. En mantenimiento esta medida significa un balance entre costos generados por inspecciones o mantenimientos preventivos y posibles fallas funcionales que se presenten.
- Medida F2(ß = 2): En esta medida se le asigna más peso al Recall y menos peso a la precisión, con este coeficiente se parte del supuesto que los falsos negativos son más importantes, para el caso de predicción de fallas en mantenimiento, se utiliza esta métrica cuando los costos en que incurren por un paro funcional de la máquina, son mucho mayores que asumir costos por inspecciones o mantenimientos preventivos cuando se generan falsas alarmas.

4.7.2 Métricas de clasificación

La métrica de clasificación más utilizada es la curva de características operativas (ROC) y la correspondiente área bajo la curva ROC (AUC). La curva ROC permite la visualización de la compensación relativa entre beneficios, dado por la tasa de verdaderos positivos, y los constes, dado por la tasa de falsos positivos (Branco, Torgo, & Ribeiro, 2015). El rendimiento de un clasificador para una determinada distribución se representa mediante un único punto en el espacio ROC como se ve en la Figura 12. Una curva ROC consta de varios puntos, cada uno de los cuales corresponde a un valor diferente de un parámetro de decisión/umbral utilizado para clasificar un ejemplo como perteneciente a la clase positiva.

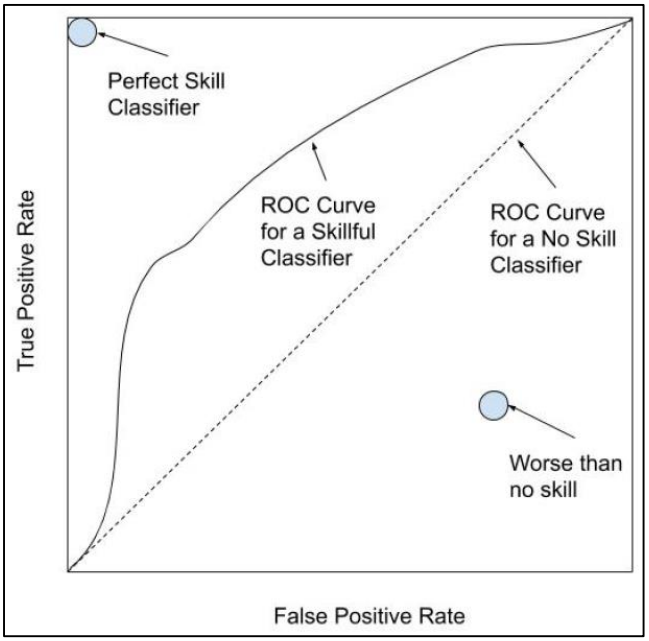

Figura 12. Gráfica de la curva ROC.
Fuente: Tomado de (Brownlee, 2020a)

La tasa de verdaderos positivos (TPR o True Positive Rate) es el Recall o Sensibilidad como indica la **Ecuación** 17:

$$Tasa\ de\ Verdaderos\ Positivos\ (TPR) = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \quad (17)$$

La tasa de falsos positivos (FPR o False Positive Rate) es calculada como indica la **Ecuación** 18:

$$Tasa\ de\ Falsos\ Positivos\ (FPR) = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \quad (18)$$

El AUC permite la evaluación del mejor modelo en promedio y se calcula con la **Ecuación** 19:

$$AUC = \frac{1 + TPR - FPR}{2} \quad (19)$$

De acuerdo con Torres (2017), se asocia el resultado de $AUC = 1$ a una clasificación perfecta, por otra parte, un $AUC = 0.5$ se asocia a una clasificación totalmente aleatoria, por último, un $AUC < 0.5$ refleja una muy mala clasificación.

4.7.3 Coeficiente de Correlación de Matthews

Para Chicco, Warrens, y Jurman (2021), otra métrica de clasificación para clases binaras desbalanceadas es el Coeficiente de Correlación de Matthews (MCC), conocido también como coeficiente ϕ, es decir, la raíz cuadrada de la media del estadístico x^2, $\sqrt{x^2/n}$, sobre n muestras observadas para una tabla de contingencia de 2 × 2 de un problema de clasificación, en este caso, para la matriz de confusión, resumiéndola en un único valor para las tareas binarias.

Dado los verdaderos positivos (TP), los verdaderos negativos (TN), los falsos negativos (FN), y los falsos positivos (FP), el cálculo del MCC se define por la **Ecuación** 20:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (20)$$

MCC es simétrico en cuanto a la clase: cambiar positivos y negativos conduce al mismo resultado. El valor mínimo de MCC es -1, lo que significa una predicción perfectamente errónea, en la que un clasificador etiqueta todos los positivos como negativos y todos los negativos como positivos. Por el contrario, el valor máximo es +1, que significa una clasificación perfecta.

Si el valor de MCC está en torno a 0 significa que la predicción realizada fue similar a una adivinación aleatoria. Cuando un par de valores de la matriz de confusión son ambos 0, el valor de correlación de Matthews puede ser indefinido.

5. Metodología

5.1 Área de trabajo

El objeto de estudio lo componen los datos recogidos por el Sistema SCADA de un aerogenerador Gamesa G90 de 2MW de un parque eólico ubicado en la provincia de Albacete en España. Por razones de confidencialidad, no se puede proporcionar más detalles del parque eólico en cuestión.

La base de datos contiene un total de 20 variables, listadas en la Tabla 7, abarcando una serie de tiempo desde inicios de noviembre de 2011 hasta inicios de septiembre de 2012.

Tabla 7. *Variables del aerogenerador recogidos por el Sistema SCADA.*

Nº	VARIABLE	DESCRIPCIÓN	UNIDADES
1	Ángulo Pitch	Medida del ángulo de las palas del aerogenerador con un rango de 0º a 90º	(º)
2	Corriente estator	Corriente producida en el estator del generador	(A)
3	Corriente red	Suma de la corriente producida en el estator y en el rotor del generador	(A)
4	Corriente rotor	Corriente producida en el rotor del generador	(A)
5	Dirección viento	Dirección del viento. Utilizada para orientar la nacelle. Se emplea el norte como referencia 0º y sur 180º	(º)
6	Factor de potencia	Cos φ de la potencia producida	-
7	Frecuencia red	Frecuencia de la red de suministro	(Hz)
8	Grados de posición de Nacelle	Posición en la que se encuentra la nacelle. Se emplea como referencia norte 0º y sur 180º	(º)
9	Potencia estator	Potencia producida en el estator	(kW)
10	Potencia rotor	Potencia producida en el rotor	(kW)
11	Potencia reactiva	Potencia reactiva total	(kVAr)
12	Potencia total	Potencia total producida por el aerogenerador	(kW)
13	Temperatura ambiente	Temperatura en el exterior de la nacelle	(ºC)
14	Temperatura anillos generador	Temperatura en el cuerpo de anillo del generador	(ºC)
15	Temperatura máxima devanado generador	Temperatura máxima registrada en un periodo de 30 segundos en los devanados del estator	(ºC)
16	Tensión Bus	Tensión en el bus de continua del convertidor	(V)
17	Tensión red	Tensión de la red de suministro	(V)
18	Velocidad generador PLC	Velocidad del rotor registrada por el PLC	(rpm)
19	Velocidad rotor	Velocidad del rotor registrada por la CCU del convertidor	(rpm)
20	Velocidad de viento	Velocidad del viento medido por el anemómetro NRG	(m/s)

Fuente: Elaborado por el autor.

5.2 Equipos y materiales

Para el desarrollo de las actividades de la presente investigación se utilizaron los siguientes recursos:

5.2.1 Equipos

Computadora Personal, Procesador Intel ® Core™ i7-8550U CPU @ 1.8GHz, tarjeta gráfica NVIDIA GeForce MX 150.

5.2.2 Materiales

- Bibliografía especializada y acceso a internet para obtener información relacionada al tema de estudio.

- Base de datos consistente en las variables de estudio, que son valores cuantitativos recogidos por el sistema SCADA de los aerogeneradores, así como los indicadores de falla de sus componentes.

- Software de programación, Python para el desarrollo de algoritmos de predicción de fallas.

- Librerías de Python: matplotlib, sklearn, pandas, seaborn.

- Software ofimático para la elaboración del informe del Trabajo de Titulación y sus respectivas láminas de presentación.

5.3 Procedimiento

El enfoque de la investigación es cuantitativo ya que se analizaron los datos con herramientas estadísticas para construir un modelo de predicción de fallas y buscar el que mejor se adapte de acuerdo a su desempeño y rendimiento. El enfoque cuantitativo utiliza la recolección de datos para probar hipótesis con base en la medición numérica y el análisis estadístico con el fin de establecer pautas de comportamiento y probar teorías.

El alcance de la investigación es correlacional ya que se conoció la relación entre las variables que actúan en un aerogenerador y su comportamiento para seleccionar y ajustar un modelo de predicción de fallas.

El método de estudio del presente trabajo es hipotético-deductivo porque se planteó como hipótesis que mediante el análisis de los datos de un aerogenerador se puede reducir la probabilidad de fallas en el mismo, permitiendo aplicar un mantenimiento predictivo y reducir costos de operación.

Para el desarrollo de esta investigación se usó el diseño no experimental longitudinal, ya que el conjunto de datos se modificó varias veces su tamaño con la finalidad de encontrar el número de muestras adecuado para la aplicación de algoritmos de Machine Learning y Deep Learning, técnicas que se compararon entre sí con el objetivo de encontrar el modelo de predicción de fallas que mejor se adapte.

La selección de muestras fue en un primer paso no probabilística puesto que se tomó como caso de estudio los datos recogidos por el sistema Scada de uno de los aerogeneradores del parque eólico en cuestión. Luego se hizo una selección probabilística del conjunto de datos de manera sistemática para la aplicación del modelo de predicción.

Los datos a utilizados en la investigación han sido recolectados por el sistema SCADA del aerogenerador a lo largo de casi diez meses en los que se recoge variables correspondientes a la velocidad viento, potencia producida en el generador y temperatura del mismo, así como del ambiente. Estos datos se analizaron mediante métodos estadísticos y técnicas de Inteligencia Artificial.

El cumplimiento de los objetivos propuestos en el presente trabajo se llevó a cabo con base en el flujograma metodológico que se indica en la Figura 13, donde se muestra de manera general el procedimiento seguido para la ejecución de la investigación. Posteriormente se plantean las actividades ejecutadas para cada objetivo.

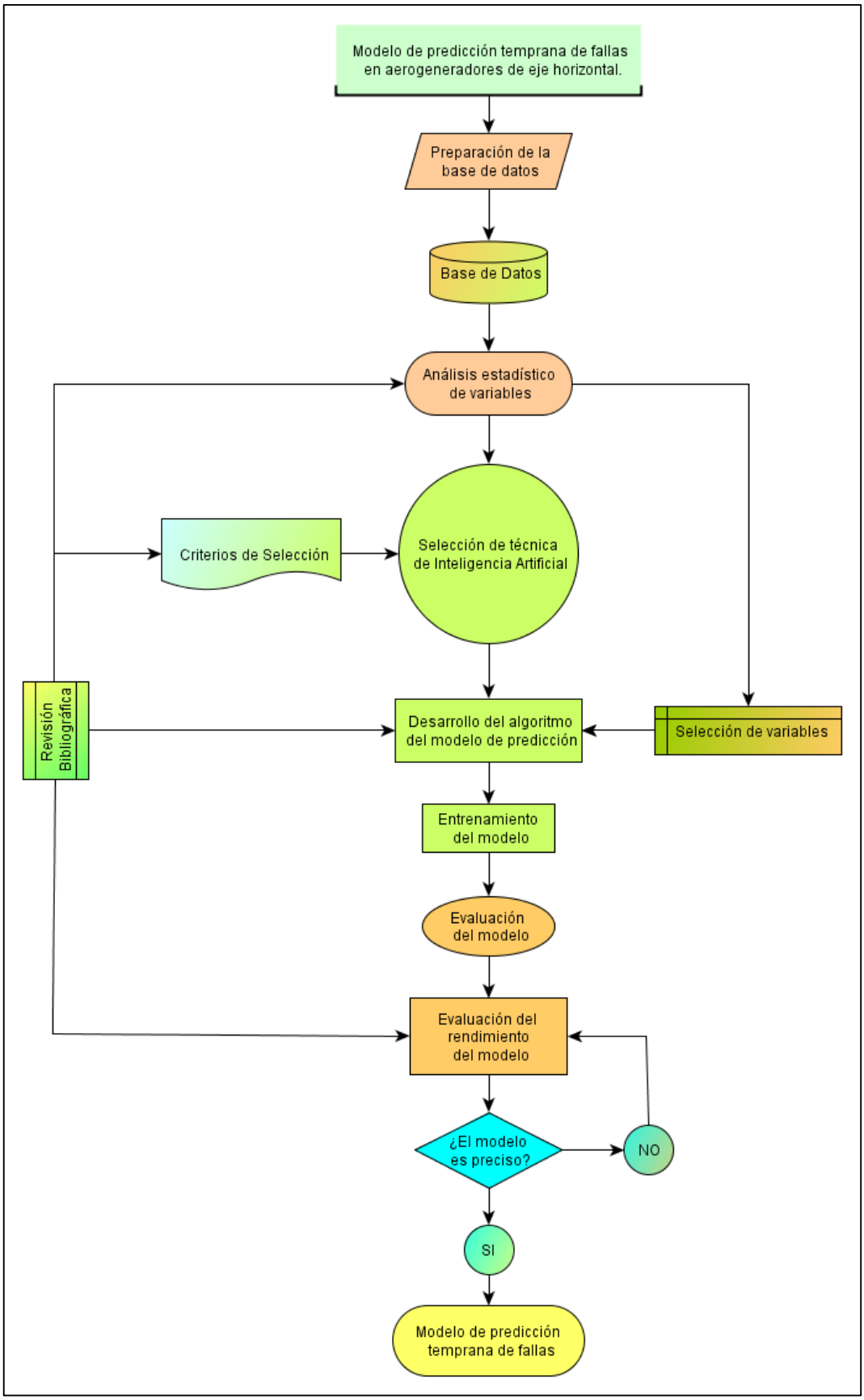

Figura 13. Flujograma metodológico utilizado.
Fuente: Elaborado por el autor.

5.4 Procesamiento y análisis de datos

5.4.1 Análisis de datos históricos de variables mecánicas y eléctricas presentes en el generador de la turbina eólica.

5.4.1.1 Exploración de la base de datos mediante gráficas

En Brownlee (2020b), se menciona que la visualización desempeña un papel importante en el análisis y la previsión de series temporales, ya que los gráficos de los datos pueden proporcionar valiosos diagnósticos para identificar estructuras temporales como tendencias, ciclos y estacionalidad que pueden influir en la elección del modelo. En este trabajo se emplean los siguientes:

- Mapas de calor
- Gráficas de Líneas
- Diagramas de Caja
- Histogramas y Polígonos de frecuencia
- Gráficas de Barras
- Gráficas de Dispersión

5.4.1.2 Selección de variables

Para obtener un mejor manejo del nombre de las variables presentes en la base de datos se procedió a realizar una abreviación de los nombres de cada variable como se indica en la Tabla 8.

Tabla 8. *Abreviado de los nombres de las variables*

Nº	Nombre original	Abreviatura	Nº	Nombre original	Abreviatura
1	Ángulo Pitch	A_pitch	11	Potencia reactiva	P_reactiva
2	Corriente estátor	C_estator	12	Potencia total	P_total
3	Corriente red	C_red	13	Temperatura ambiente	T_amb
4	Corriente rotor	C_rotor	14	Temperatura anillos generador	T_anillos_gen
5	Dirección viento	D_viento	15	Temperatura máxima devanado generador	T_estator
6	Factor de potencia	Cos_phi	16	Tensión Bus	Ten_bus
7	Frecuencia de red	F_red_hz	17	Tensión red	Ten_red
8	Grados posición de Nacelle	Pos_nacelle	18	Velocidad generador PLC	v_gen
9	Potencia estátor	P_estator	19	Velocidad rotor	v_rotor
10	Potencia rotor	P_rotor	20	Velocidad viento	V_viento

Fuente: Elaborado por el autor.

Con el objetivo de conocer la relación entre variables del conjunto original de datos se realizó un análisis correlacional entre ellas. Además, esto sirvió para seleccionar las principales variables que actúan en el generador con enfoque en la predicción de fallas relacionadas a la temperatura. Esto se llevó a cabo mediante matrices de correlación, en las que se muestra el coeficiente de correlación para cada par de variables. Posteriormente se lleva esta matriz a una gráfica de mapa de calor.

La correlación es una medida normalizada de asociación o co-variación lineal entre dos variables. Esta medida o índice de correlación r puede variar entre -1 y +1, ambos extremos indicando correlaciones perfectas, negativa y positiva respectivamente (Vinuesa, 2016).

El índice de correlación r es en sí mismo una medida del tamaño del efecto, que suele interpretarse de la siguiente manera:

- Correlación despreciable: r < |0,1|
- Correlación baja: |0,1| < r <= |0,3|
- Correlación mediana: |0,3| < r <= |0,5|
- Correlación fuerte o alta: r > |0,5|

5.4.1.3 Preparación de la base de datos

Para el entrenamiento de algoritmos de machine learning es necesario organizar los datos de acuerdo con su arquitectura. Así se tienen tablas, matrices, arreglos de matrices, etc.

Los datos pueden estructurarse en filas y columnas, como una tabla de base de datos o como una hoja de cálculo. Estos son conocidos como "estructura tradicional de datos" y son comunes en el campo del aprendizaje automático (Cuevas, Avalos, Emanuel, Valdivia, & Pérez, 2021). Los conceptos básicos para datos se definen a continuación:

- **Observación:** es la entidad más pequeña, con propiedades de interés para un estudio que puede ser registrado.
- **Características:** son las propiedades o atributos de las observaciones que pueden ser útiles para el aprendizaje.
- **Tipos de datos:** las características tienen un tipo de datos. Estos pueden ser de valor real o entero, o pueden tener un valor categórico u ordinal.
- **Conjunto de datos:** una colección de observaciones es un conjunto de datos y, cuando se trabaja con métodos de aprendizaje máquina, generalmente se requieren algunos conjuntos de datos para diferentes propósitos.
- **Datos de entrenamiento:** conforman un conjunto de datos que se incorpora al algoritmo de aprendizaje máquina para entrenar el modelo.

- **Datos de prueba:** constituyen un conjunto de datos utilizado para validar la precisión del modelo. Se lo conoce también como “conjunto de datos de validación”.

Las tablas y matrices son la configuración más simple que se puede disponer, en las que se puede encontrar distribuidos los datos por filas para las observaciones y columnas para las variables. Estos datos constituyen los parámetros de entrada para cualquier algoritmo de predicción que se utilice. Los datos de respuesta o datos de salida se pueden incluir en la misma tabla como muestra la Figura 14 o formar otra tabla o matriz con una única columna y el mismo número de filas que las observaciones de entrada.

	A	B	C	D	E	F		Q	R	S	T
1	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages		Total intl calls	Total intl charge	Customer service calls	Churn
2	KS	128	415	No	Yes	25		3	2.7	1	False
3	OH	107	415	No	Yes	26		3	3.7	1	False
4	NJ	137	415	No	No	0		5	3.29	0	False
5	OH	84	408	Yes	No	0		7	1.78	2	False
6	OK	75	415	Yes	No	0		3	2.73	3	False
7	AL	118	510	Yes	No	0		6	1.7	0	False
8	MA	121	510	No	Yes	24		7	2.03	3	False
9	MO	147	415	Yes	No	0		6	1.92	0	False
10	WV	141	415	Yes	Yes	37	...	5	3.02	0	False
11	RI	74	415	No	No	0		5	2.46	0	False
12	IA	168	408	No	No	0		2	3.02	1	False
13	MT	95	510	No	No	0		5	3.32	3	False
14	IA	62	415	No	No	0		6	3.54	4	False
15	ID	85	408	No	Yes	27		4	3.73	1	False
16	VT	93	510	No	No	0		3	2.19	3	False
17	VA	76	510	No	Yes	33		5	2.7	1	False
18	TX	73	415	No	No	0		2	3.51	1	False
19	FL	147	415	No	No	0		4	2.86	0	False
20	CO	77	408	No	No	0		6	1.54	5	True

Figura 14. Arreglo de datos en filas y columnas para datos de entrada y salida.
Fuente: Tomado de Gonzáles (s.f.).

Para las redes neuronales profundas como las Convolucionales (CNN) se requiere de una configuración de datos de entrada dispuestos en matrices o Arrays en 3 dimensiones, compuestas de muestras, pasos de tiempo y características (Brownlee, 2018).

Muestras: estas son las filas en la base de datos. Una muestra es una secuencia.

Pasos de tiempo: estas son las observaciones pasadas para una característica, como las variables de retardo.

Características: estas son las columnas en los datos.

En la Figura 15, las muestras se orientan a lo largo del eje 0, en el eje 1 los pasos de tiempo y en el eje 2 las características.

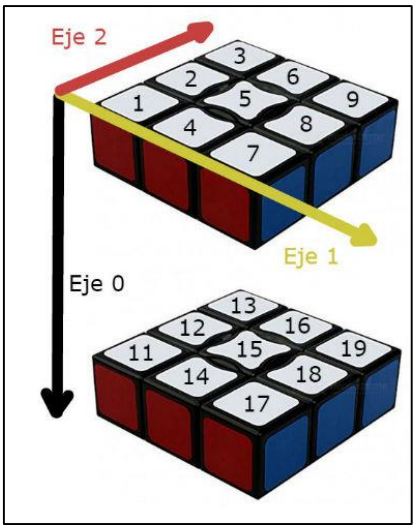

Figura 15. Arreglo de datos tridimensional
Fuente: Tomado de (Interactive Chaos, s.f.)

5.4.1.4 Análisis de variables

Con el objeto de conocer la tendencia de las variables a lo largo del tiempo de estudio es preciso realizar un análisis estadístico para determinar relaciones entre ellas y entre las observaciones de cada variable. Esto pude ayudar con ideas de escalado de datos e incluso limpieza de datos como preparación del conjunto de entrenamiento para la modelización.

5.4.1.5 Ingeniería de datos

División de conjunto de datos. Consiste en la separación de los datos de entrenamiento en dos grupos: uno para entrenamiento y el otro para validación. Como regla general, la relación entre el conjunto de entrenamiento y el conjunto de validación es de 8:2, como menciona Kim (2017), aunque esto puede variar dependiendo de la cantidad de datos disponible.

***Remuestreo de datos*.** Cuando las clases de los datos se encuentran desbalanceadas, se procede a realizar un remuestreo. Lo más común es que la clase negativa, que indica un estado normal de la máquina involucre la mayoría de las clases, mientras que la clase positiva todo lo contrario. En Rodríguez (2017), se define el problema de desbalance de la siguiente manera: Sea T un conjunto de entrenamiento dividido en dos clases, sea $|M|$ la cantidad de objetos en una clase y $|m|$ la cantidad de objetos en otra clase, de modo que se obtiene la **Ecuación** 21:

$$|T| = |M| + |m| \tag{21}$$

El conjunto T se considera desbalanceado si $|M| > |m|$.

Así mismo, Rodríguez (2017), menciona que no existe un consenso para saber el tamaño de una clase respecto a otra, por lo que una métrica para evaluar el nivel de desbalance es el Ratio de desbalance *IR* (Imbalance Ratio) que se define con la **Ecuación** 22:

$$IR = \frac{|M|}{|m|} \tag{22}$$

Donde $|M|$ es la cantidad de objetos en la clase mayoritaria y $|m|$ es la cantidad de objetos en la clase minoritaria. El *IR* representa la cantidad promedio de objetos en la clase mayoritaria por cada objeto de la clase minoritaria.

Es entonces cuando surge la necesidad de igualar las clases para ayudar al modelo a generalizar cuando se ingrese nuevos datos en el proceso de validación. Existen dos métodos de remuestreo: el submuestreo (undersampling) recorta las muestras de la clase mayoritaria, y el sobremuestreo (oversampling) que consiste en la adición de muestras a la clase negativa, disminuyendo la proporción de clases o igualándolas en ambos casos como se ilustra en la Figura 16.

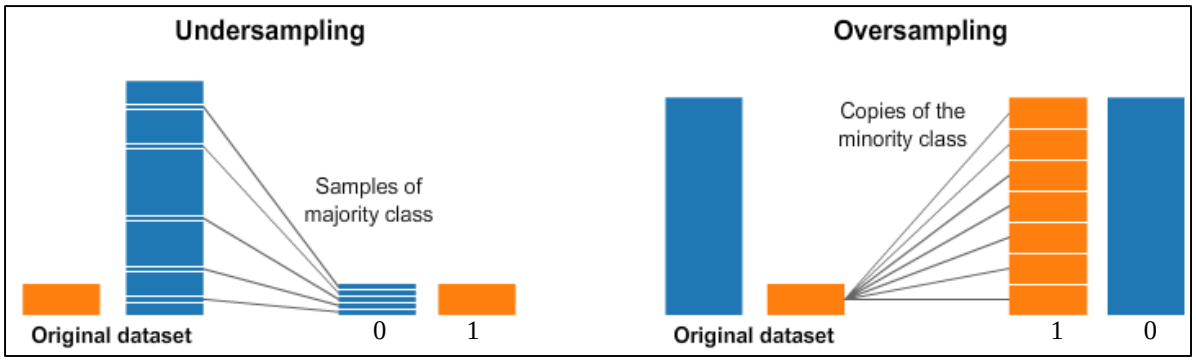

Figura 16. Representación del submuestreo y sobremuestreo.
Fuente: Adaptado de Fuchs (2020).

Técnicas de remuestreo de datos. En esta investigación, se realizó el remuestreo del conjunto de datos de entrenamiento con las variables seleccionadas previamente, aplicando cada uno de los métodos listados en la Tabla 9, esto con la finalidad de obtener un conjunto que se adapte mejor a los algoritmos de aprendizaje.

Tabla 9. Métodos de remuestreo

Métodos	**Tipos**	
Remuestreo aleatorio		- Sobremuestreo aleatorio - Submuestreo aleatorio
Métodos de Sobremuestreo		- Synthetic Minority Oversampling (SMOTE)
	SMOTE con generación selectiva de muestras	- Borderline –SMOTE - Borderline –SMOTE SVM - Adaptative Synthetic Sampling (ADASYN)
Métodos de Submuestreo	Métodos que seleccionan los ejemplos a conservar	- Near Miss Undersampling - Condensed Nearest Neighbor Rule Undersampling CNN
	Métodos que seleccionan los ejemplos a borrar	- Tomek Links - Edited Nearest Neighbors Rule for Undersampling ENN
	Combinación de métodos de conservación y eliminación	- One-Sided Selecction for Undersampling - Neighborhood Cleaning Rule for Undersampling
Sobremuestreo y submuestreo	Combinación manual de métodos de muestreo de datos	- Sobremuestreo y submuestreo aleatorios - SMOTE y submuestreo aleatorio
	Métodos estándar de muestreo combinado de datos.	- SMOTE y Tomek Links - SMOTE y Edited Nearest Neighbors

Fuente: Elaborado por el autor a partir de Brownlee (2020a) y Kraiem (2020).

De acuerdo con Kraiem (2020), todos los métodos de remuestreo de clases presentan ventajas e inconvenientes. Así, el Submuestreo aleatorio es de fácil aplicación y disminuye el tiempo de ejecución y de almacenamiento al reducir el número de instancias del conjunto de entrenamiento cuando este es muy grande. Sin embargo, el principal inconveniente es la pérdida de información importante, en especial cuando el índice de desbalance es alto.

Por el contrario, el Sobremuestreo aleatorio no conduce a la pérdida de información y es fácil de aplicar, pero puede causar un sobreajuste y un aumento del tamaño del conjunto de datos en el caso de alto desequilibrio, dando lugar a un mayor tiempo de entrenamiento del modelo.

La ventaja principal de SMOTE es mitigar el problema de sobreajuste causado por el sobremuestreo aleatorio, además, es eficiente cuando se combina con estrategias de submuestreo, pero no muy eficaz cuando se aplica a conjuntos de datos con alta dimensionalidad.

Las técnicas de submuestreo CNN, Tomek Links, ENN, OSS, NCL y NCR no se utilizan para devolver conjuntos de datos equilibrados sino para eliminar ruido en los datos, por lo que no causan pérdida de información, aunque tienen algunos inconvenientes que dependen de la

técnica particular. La combinación de SMOTE y los métodos de submuestreo reduce los efectos de instancias límite y el problema de solapamiento de clases.

Con el propósito de seleccionar la mejor técnica, se validó cada conjunto remuestreado con el modelo de Árbol de Decisión (Decision Tree), y se evalúo su rendimiento con las métricas ROC, Recall y F1-score. Se utilizó Recall ya que mide el número de predicciones positivas que acierta el modelo de todas las predicciones positivas correctas que podrían haberse detectado. Por su parte, ROC arroja puntajes entre 0.0 y 1.0 donde valores iguales o menores a 0.5 señalan que el modelo no cuenta con la suficiente habilidad para clasificar y el valor 1.0 indica una clasificación perfecta. F1-score en cambio equilibra el peso entre la predicción de falsos negativos y falsos positivos.

La selección del algoritmo Decision Tree estuvo motivado en sus fortalezas, las cuales son requerir menos limpieza de datos y no está influenciado por valores atípicos, como menciona Huertas (2020), pero además, Kraiem (2020) en referencia a Japkowicz & Stephen (2002), alude que este algoritmo es el más afectado por el desequilibrio de clases, y así lo demuestran en su trabajo Lemnaru & Potolea (2012).

El código para implementar los algoritmos de remuestreo se adaptó de Brownlee (2020a), y se empleó el lenguaje de programación Python en el entorno Spyder. Este código se presenta en el

Anexo 1, y ejemplifica el remuestreo con undersampling, sin embargo, para el resto de técnicas se cambió únicamente el nombre de la estrategia.

Escalado de datos. Según Martínez (2017), las variables utilizadas en el modelo de predicción tienen diferentes unidades (velocidad de viento, temperaturas, potencia, etc.) y sus valores absolutos son muy diferentes, los valores iniciales deben ser re-escalados en un rango de 0 a 1, de acuerdo con sus valores máximos y mínimos. En la Figura 17a se muestra la medida de dos variables en diferentes unidades, mientras que en la Figura 17b se ejemplifica el escalado de dichas variables entre 0 y 1.

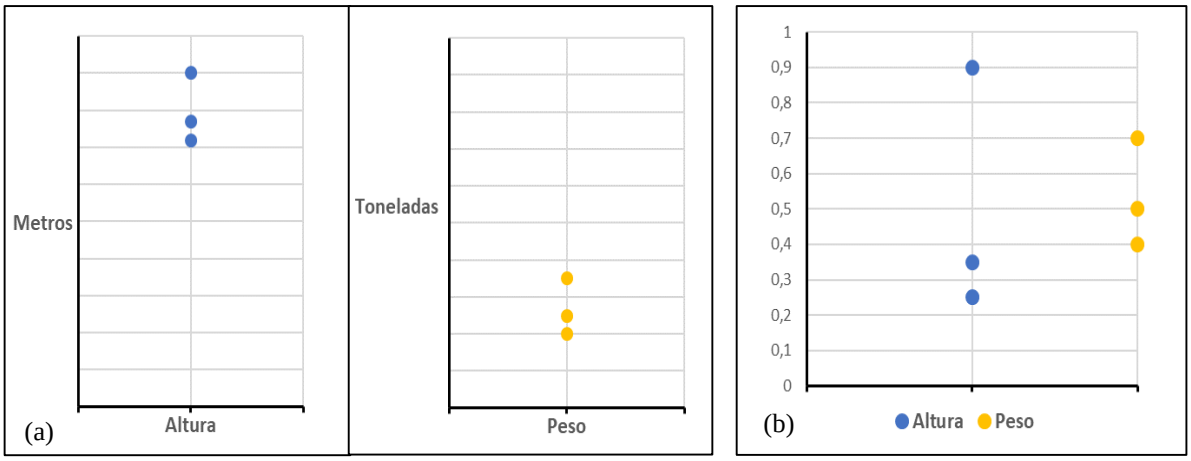

Figura 17. (a) Medida de dos variables con difrerentes unidades, y (b) reescalado de las variables entre 0 y 1.
Fuente: Elaborado por el autor.

Huertas (2020) menciona que el re-escalado de datos entre 0 y 1 se denomina normalización, pero además, para el ingreso de datos en un algoritmo de aprendizaje, también se puede optar por la estandarización, que implica cambiar la distribución de valores de cada variable para que la media de las observaciones sea 0 con una desviación estándar de 1.

La normalización está dada por la **Ecuación 23**:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{23}$$

Donde:

X_{norm} = observación normalizada

X = observación que se desea normalizar

X_{min} = observación con el valor mínimo de todos los datos de la variable

X_{max} = observación con el valor máximo de todos los datos de la variable

Por su parte, la estandarización se define por la **Ecuación** 24:

$$z = \frac{x_i - \bar{x}}{s} \tag{24}$$

Donde:

z = observación estandarizada.

x_i = observación que se desea estandarizar.

$\bar{x}$ = media muestral de la variable analizada.

s = desviación estándar de las variables analizada.

5.4.2 Desarrollo de algoritmo de predicción de fallas.

5.4.2.1 Selección de los algoritmos de Inteligencia Artificial.

Para la selección del algoritmo de IA a utilizar en este trabajo, en primer lugar, se identificó la tarea que se va a realizar. En la Tabla 10 se presentan algunas de las tareas más comunes relacionadas con la IA y los diferentes parámetros de trabajo.

Teniendo en cuenta el tema de investigación y la información presentada en la Tabla 10, se determinó que la tarea a realizar está relacionada con Predecir Salidas y una de sus aplicaciones principales es el Mantenimiento Predictivo, para lo cual es necesario entradas de datos numéricos registrados a través de sensores y almacenados por sistemas SCADA como el caso de la investigación en curso. Así, el enfoque más común que se da a esta tarea es el Aprendizaje Automático.

En el trabajo presentado por Maldonado, Martín, Artigao & Gómez (2020), se exponen las técnicas de Inteligencia Artificial más utilizadas en las investigaciones de la Condición de Monitoreo para aerogeneradores, siendo la más común las Redes Neuronales Artificiales y como mencionan Kavaz & Barutcu (2018), han demostrado ser potentes algoritmos para varios objetivos, incluida la clasificación, la estimación y la detección de fallas, ya que se adaptan a entornos de no linealidad y se pueden aplicar en tiempo real.

Tabla 10. *Tareas comunes que aplican IA y sus respectivos parámetros.*

Tareas comunes	**Aplicaciones**	**Entrada**	**Algoritmos comunes**	**Enfoque típico**	**Ejemplo**
Predecir una salida	- Mantenimiento predictivo - Comercio financiero - Sistemas de recomendación	- Datos del sensor - Datos financieros con marca de tiempo - Datos numéricos	- Regresión lineal - Árboles de decisión - Máquinas de vector soporte (SVM) - Redes Neuronales - Reglas de asociación	El aprendizaje automático es más común	Utilizar datos de sensores en tiempo real de un motor para predecir la vida útil de la máquina rotativa.
Identificar objetos	- Asistencia avanzada al conductor (ADAS) con detección de objetos - Robótica - Percepción de visión por computadora para reconocimiento de imágenes - Detección de actividad - Biometría de voz (huella de voz)	- Imágenes - Videos - Señales	- CNN - Agrupamiento - Viola-Jones	El aprendizaje profundo es más común	Crear una aplicación de visión artificial que pueda detectar vehículos.
Moverse físicamente en una simulación	- Sistemas de control - Robótica en la fabricación - Coches autónomos - Drones - Videojuegos	- Modelos matemáticos - Datos de sensores - Videos - Datos LIDAR	- Aprendizaje por refuerzo (redes Q profundas) - Redes Neuronales Artificiales (ANN) - CNN - Redes neuronales recurrentes (RNN)	El aprendizaje profundo es más común	Realizar una planificación de ruta robótica para conocer la mejor ruta posible a un destino
Descubrir tendencias	- Procesamiento de lenguaje natural para registros de seguridad - Investigación médica o de mercado - Análisis de sentimientos - Ciberseguridad - Resumen de documentos	- Transmisión de datos de texto - Datos de texto estáticos	- RNN - Regresión lineal - SVM - Naive Bayes - Asignación de Dirichlet latente - Análisis semántico latente - Word2vec	El aprendizaje automático es más común	Determinar cuántos temas están presentes en los datos de texto
Mejorar imágenes o señales	- Mejorar la resolución de la imagen - Eliminar el ruido de las señales de audio	- Imágenes - Datos de señal	- LSTM - CNN - Red neuronal VDSR	El aprendizaje profundo es más común	Crear imágenes de alta resolución a partir de imágenes de baja resolución.
Responder a voz o texto	- Llamadas de atención al cliente - Dispositivos inteligentes - Asistentes virtuales - Traducción automática - Dictado	- Datos acústicos - Datos de texto	- RNN (algoritmos LSTM en particular) - CNN - Word2vec	Se utilizan ambos enfoques (aprendizaje automático y/o aprendizaje profundo	Reconocer automáticamente comandos hablados como “encender”, “apagar”, “detener” y “continuar”.

Fuente: Elaborado por el autor a partir de la información presentada en www.mathworks.com

Además de aplicar Redes Neuronales, también se emplea otras técnicas de Machine Learning para la predicción de valores. Así, para la predicción de fallas en motores, Huertas (2020), trabaja en Python con los algoritmos más conocidos como son:

- Árboles de Decisión
- Gradient Boosting
- XGBoost
- Random Forest
- Extra Trees
- Regresión Logística
- Bagging
- Naive Bayes
- Máquinas de vector soporte
- K-Nearest Neighbors
- Perceptrón Multicapa (MLP)

Las técnicas listadas anteriormente están dirigidas a tareas de clasificación, que es una forma de predicción de datos. Algunas de estas son utilizadas con este propósito por Espinar (2018), Arnejo (2017), Huertas (2020), quien además trabaja con Redes Neuronales Profundas como CNN para la clasificación.

Si bien, los trabajos mencionados no están relacionados directamente con la predicción de fallas en aerogeneradores, se tomaron como guía para la elaboración de algoritmos para la clasificación de eventos normales o anormales en el generador de una turbina eólica de eje horizontal.

En la Tabla 11 se resumen las técnicas que se aplicaron a los datos del presente trabajo.

Tabla 11. *Técnicas de IA para la clasificación de fallas en aerogenerador.*

Enfoque Principal	Técnica	Software
Machine Learnig	- Árboles de Decisión - Gradient Boosting - XGBoost - Random Forest - Extra Trees - Regresión Logística - Bagging - Naive Bayes - Máquinas de vector soporte - K-Nearest Neighbors - MLP	Python
Deep Learnig	- CNN	Python

Fuente: Elaborado por el autor.

5.4.2.2 Aplicación de los algoritmos seleccionados sobre los datos disponibles

Una vez que se han elegido los modelos, se procede a entrenarlos con el conjunto de entrenamiento. El proceso a seguir se muestra en la Figura 18. Las etapas de implementación de un algoritmo de aprendizaje automático empiezan con la recopilación de datos que luego son preparados y pre-procesados como se menciona en páginas anteriores. Los datos de entrenamiento son requeridos por el modelo para aprender y generalizar el conocimiento

existente en tal conjunto y una vez que esto se ha logrado, el modelo es evaluado con nuevos datos conocidos como datos de prueba.

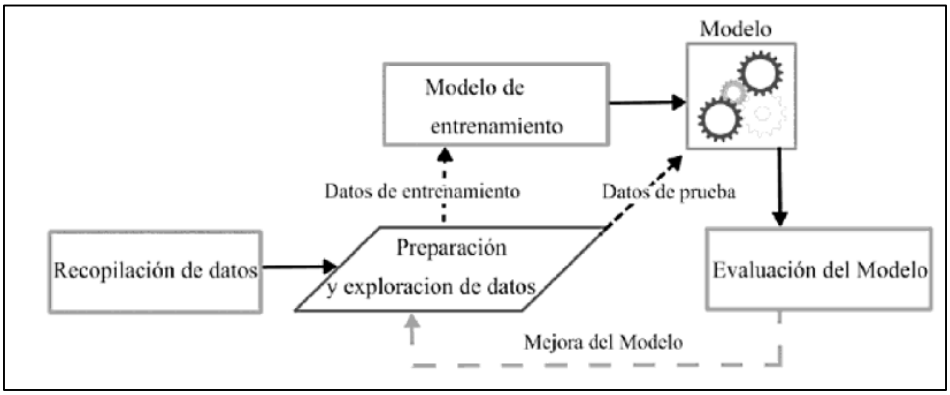

Figura 18. Etapas de implementación de un algoritmo de aprendizaje automático.
Fuente: Tomado de Cuevas et al. (2021).

Los algoritmos de predicción seleccionados se desarrollaron en los cuadernos Notebook Jupyter con el lenguaje Python. En una primera instancia se realiza la importación de librerías y datos, y su respectiva preparación como se muestra en el Anexo 2.

Seguidamente se aplica los algoritmos seleccionados y se evalúa la Precisión sobre los datos en bruto, es decir, sin aplicar ingeniería de datos. Esto se puede ver en el Anexo 3.

Luego se desarrolla el proceso para implementar las técnicas de Machine Learning con todos los datos preparados. En el Anexo 4 se muestran los códigos para este punto, de manera particular para el algoritmo Extra Trees. De igual forma se sigue el mismo proceso para el resto de algoritmos.

Para la Red Neuronal Convolucional o CNN, perteneciente al campo del Deep Learning se tuvo en cuenta otras consideraciones. Primero, al explorar el conjunto de datos original se encontró que los datos de la clase 1 son minoría, pero además, que se encuentran concentrados en las observaciones de pocos días, de todo el tiempo de estudio, los cuales se listan en la Tabla 12, por lo que se opta por la predicción mediante la clasificación de secuencias.

Tabla 12. *Días utilizados para el nuevo conjunto de datos.*

Conjunto	Día
Conjunto de entrenamiento	2011-11-10
	2012-08-10
	2012-08-11
	2012-08-12
Conjunto de prueba	2012-08-13
	2012-08-23

Fuente: Elaborado por el autor.

Según Brownlee (2017), la predicción de secuencias es diferente de otros tipos de problemas de aprendizaje supervisado, ya que la secuencia impone un orden a las observaciones que debe conservar al entrenar modelos y hacer predicciones, lo que difiere de los clasificadores de aprendizaje automático, donde cada muestra de los conjuntos de entrenamiento y prueba puede considerarse una observación del dominio, y no el orden de las mismas no es importante.

Existen una serie de problemas de predicción de secuencias que difieren según las secuencias de entrada y salida. En esta investigación se tratarán los problemas involucrados con la clasificación de secuencias, que consiste en predecir una etiqueta de clase para una secuencia de entrada dada. Por ejemplo, dado: 1, 2, 3, 4, 5, predecir: “Bueno” o “Malo”, como se ilustra en la Figura 19.

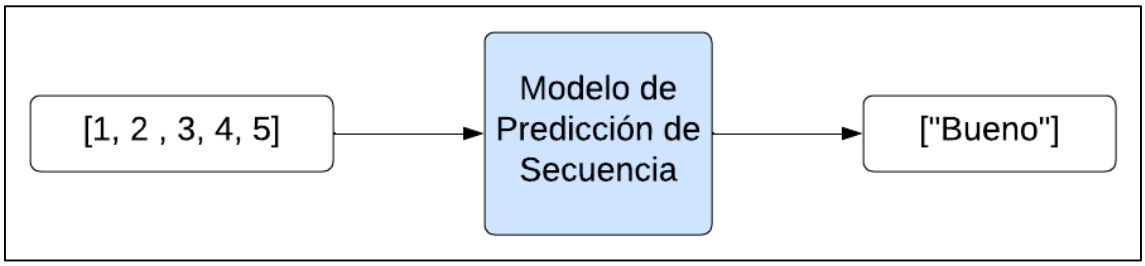

Figura 19. Ejemplo de un problema de clasificación de secuencia.
Fuente: Adaptado de (Brownlee, 2017)

El objetivo de la clasificación de secuencia es construir un modelo de clasificación utilizando un conjunto de datos etiquetados, para que el modelo pueda utilizarse para predecir la etiqueta de clase de una secuencia no vista.

Es en este punto donde se aplica el arreglo tridimensional a los datos antes presentados. En el Anexo 5 se incluye el algoritmo para la predicción con una red neuronal profunda CNN.

5.4.2.3 Ajustar los hiperparámetros de los modelos con mejores rendimientos.

Si el modelo no cumple con las expectativas, existe la posibilidad de mejorarlo. Una de las maneras de hacerlo es con el ajuste de hiperparámetros, que es el proceso de identificar el conjunto de parámetros que proporciona el mejor modelo como se muestra en la Figura 20. Los hiperparámetros controlan el modo en que un algoritmo de aprendizaje automático ajusta el modelo a los datos.

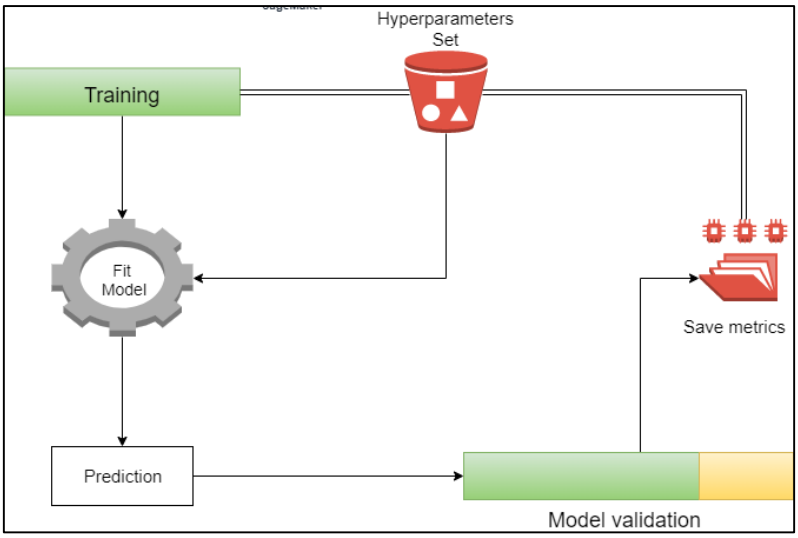

Figura 20. Mejora del modelo de aprendizaje automático.
Fuente: Tomado de Morrís (2020).

En la Tabla 4 se presentaron los hiperparámetros para los algoritmos de clasificación, siendo comunes la profundidad del árbol, el número de árboles y número de muestras para los modelos Decision Tree, Random Forest, Extra Trees, Bagging, XGBoost y Gradient Boosting. Para Logistic Regresion, se ajusta la regularización y la penalización a la función de pérdida, para Support Vector Machine se ajusta los parámetros núcleo, y, por último, para Naive Bayes no se realiza ningún ajuste.

Para las redes neuronales artificiales, de acuerdo con Géron (2019) y Politi (2022), los hiperparámetros más relevantes son:

- Tasa de aprendizaje: controla cuánto se ajusta el peso de la red neuronal respecto al gradiente de pérdida. Este representa la velocidad a la que la red neuronal se actualiza; valores bajos ralentizan el proceso de aprendizaje, pero permiten aprender un conjunto óptimo de pesos, mientras que los valores altos lo aceleran, pero aumentan el riesgo, para la red, de no calcular bien los pesos y, por tanto, de sobreajustar.
- Muestra: es un único elemento de un conjunto de datos. Puede ser, por ejemplo, una sola observación en una secuencia o serie temporal.
- Lote: es un conjunto de N muestras, procesadas independiente y paralelamente. El tamaño de lote es un hiperparámetro que define el número de muestras que se analizarán juntas durante el entrenamiento antes de actualizar los parámetros internos del modelo.
- Época: es un hiperparámetro que define el número de veces que el algoritmo de aprendizaje trabajará sobre el conjunto de entrenamiento completo. Se utiliza para

dividir el entrenamiento en múltiples etapas para realizar evaluaciones periódicas del modelo.

En el Anexo 6 se presenta el código pare ajustar del modelo mejor puntuado, aunque para el ajuste de otros modelos se sigue el mismo procedimiento.

5.4.3 Métricas de evaluación

Las métricas de evaluación aplicadas en este trabajo son las relacionadas con la matriz de confusión presentadas en la Tabla 6, que para fines de tareas de clasificación son las más adecuadas. Estas métricas son:

- Sensibilidad (Ecuación 9)
- Especificidad (Ecuación 10)
- Precisión o Valor Predictivo Positivo (PPV) (Ecuación 12)
- Valor Predictivo Negativo (NPV) (Ecuación 13)
- Recall (Ecuación 14)
- F-score (Ecuación 15)
- Curva ROC AUC (Ecuación 19)
- Coeficiente de correlación de Matthews (Ecuación 20)

Las métricas anteriores se emplean para calcular la predicción por cada clase, y, según Huertas (2020), también se puede calcular la medida "Macro" que corresponde a la media aritmética de las medidas de desempeño de las clases que se evalúan. Por ejemplo, para calcular la medida macro-F1, primero se mide la precisión y el Recall por cada clase, luego se calcula los puntajes F1 por clase y finalmente se obtiene la media aritmética de estos puntajes como se muestra en la **Ecuación** 25.

$$F1\ Score\ Macro = \frac{\sum_{i=0}^{n_c} F1Clase_i}{n_c} \tag{25}$$

Donde n_c es el número de clases y $F1Clase_i$ corresponde a los valores de *F1* por cada clase.

Espinar (2018), menciona que el método más simple y más usado para estimar el error de predicción es la validación cruzada, que consiste en dividir el conjunto de observación en dos partes, una para entrenamiento, y la otra parte será usada como conjunto de prueba. Esto se realiza para evitar el sobreajuste, que se deriva de un error metodológico que es entrenar los parámetros de una función de predicción y probarla con los mismos datos, es decir, un modelo que simplemente repita las etiquetas de las muestras que acaba de ver, tendría una puntuación perfecta pero no podría predecir nada útil sobre datos no vistos.

Al evaluar diferentes configuraciones de los hiperparámetros para los estimadores, aún existe el riesgo de sobreajuste en el conjunto de prueba porque los parámetros se pueden modificar hasta que el estimador funcione de manera óptima. Para resolver este problema, se puede presentar otra parte del conjunto de datos "conjunto de validación": el entrenamiento continúa en el conjunto de entrenamiento, después de lo cual se realiza la evaluación en el conjunto de validación y, cuando el experimento parece tener éxito, la evaluación final se puede hacer en el conjunto de prueba.

Sin embargo, como argumenta Bagnato (2020), al dividir los datos disponibles en tres conjuntos, reducimos drásticamente la cantidad de muestras que se pueden usar para aprender el modelo, y los resultados pueden depender de una elección aleatoria particular para el par de conjuntos de entrenamiento y validación. Una solución a este problema es la validación cruzada con partición en k-subconjuntos, aquí el conjunto de validación se mantiene dentro del conjunto de entrenamiento.

Para Huertas (2020), la validación cruzada con partición en k-subconjuntos conlleva a dividir el conjunto de datos en pliegues con el fin de producir una estimación más confiable del rendimiento del modelo. Un complemento a esta técnica es la validación cruzada estratificada, la cual es muy útil cuando se analizan datos desequilibrados, con la estratificación se divide el conjunto de datos preservando la distribución de la misma clase en cada pliegue, en consecuencia, la división de los datos coincide con la distribución en el conjunto de datos de entrenamiento completo, mitigando posibles variaciones significativas en la estimación del rendimiento del modelo.

En la Figura 21 se da una representación de la validación cruzada. Aquí el conjunto de entrenamiento se ha divido en 5 pliegues (Fold) de las cuales una se toma para validar cada iteración (Split). Una vez finalizado se emplea el conjunto de prueba para evaluar el modelo.

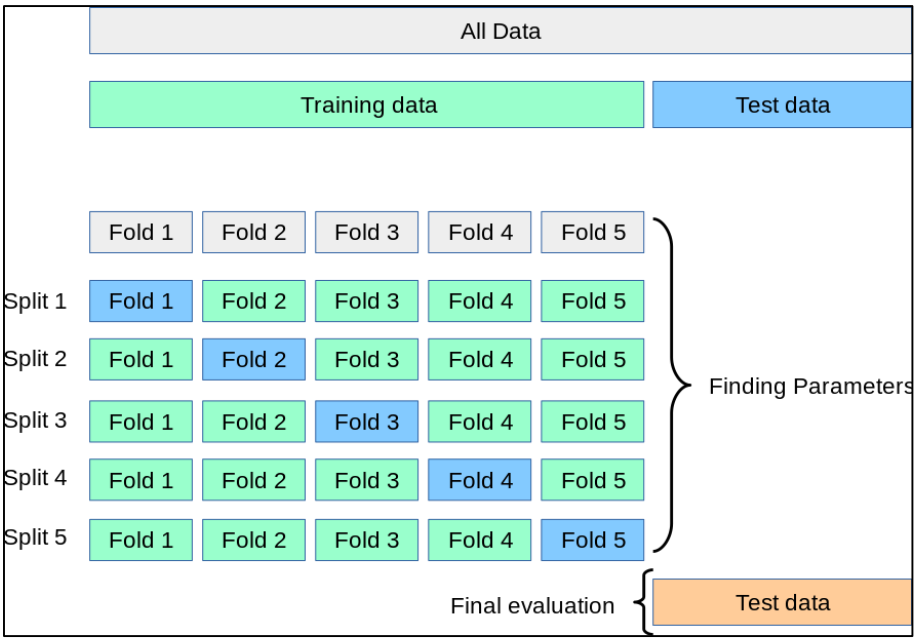

Figura 21. Entrenamiento de un conjunto de datos con validación cruzada estratificada.
Fuente: Tomado de (Scikit-learn, s.f.)

6. Resultados

6.1 Análisis Exploratorio

El presente estudio se enfoca en la predicción de fallas en el generador, por lo que se realizó la selección de las variables que actúan en este componente mediante el análisis correlacional entre la Temperatura del estátor, de gran interés, y las demás señales que recoge el sistema SCADA.

Se cuenta con 20 variables de las cuales se han seleccionado 7 teniendo en cuenta la Figura 22. Esta gráfica ayudó a identificar las variables que tienen relación lineal con la Temperatura del estator. Además, se acudió al criterio técnico para seleccionar otras variables relacionadas con temperatura.

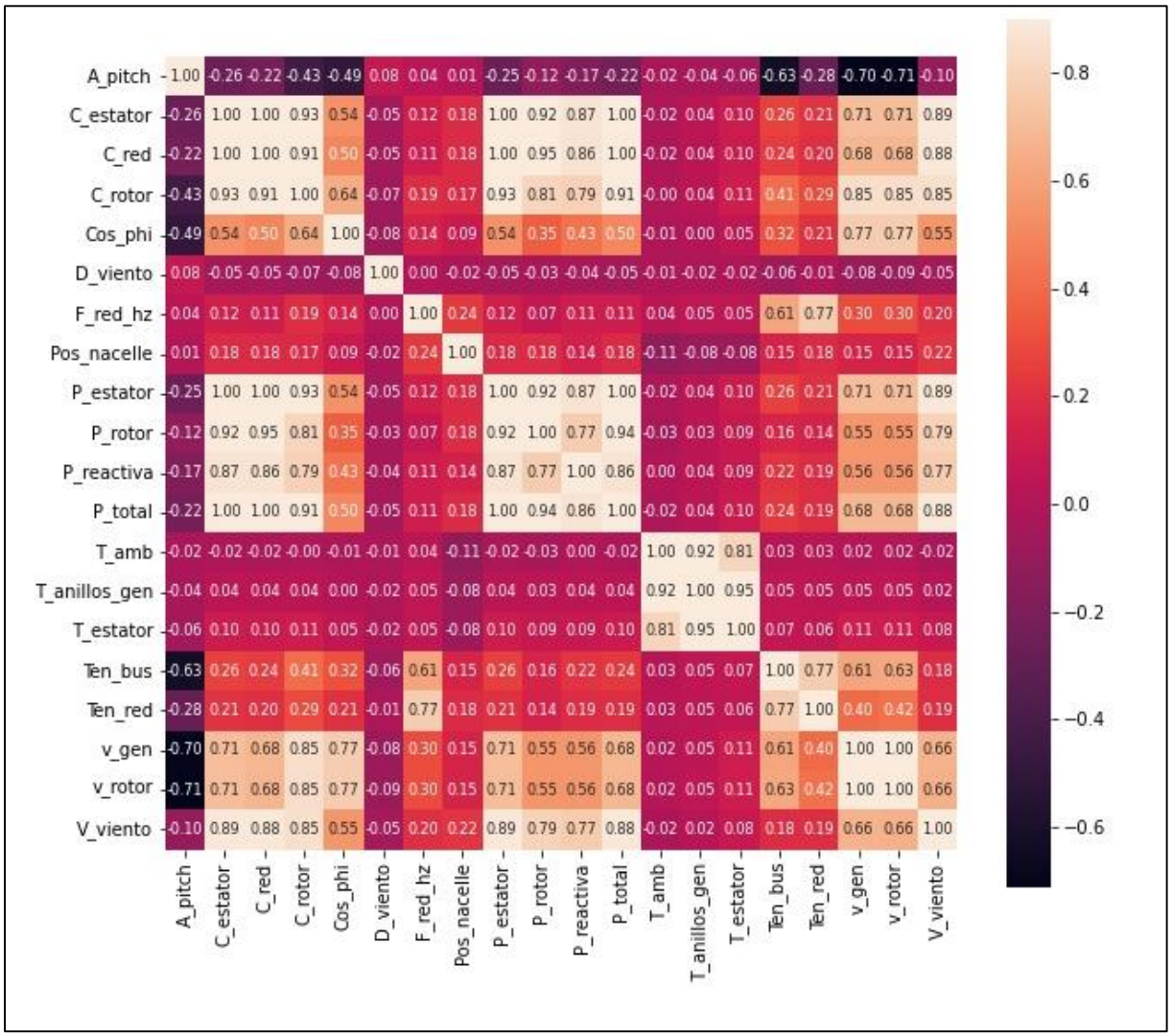

Figura 22. Análisis correlacional de las variables recogidas por el sistema SCADA del aerogenerador.
Fuente: Elaborado por el autor.

La base de datos con las que se trabajó recoge las observaciones de 7 variables que actúan en un aerogenerador tripala de eje horizontal para una duración de 10 meses, desde

noviembre de 2011 hasta agosto de 2012 como muestra la Figura 23, constando de 865 753 muestras que son registradas cada 30 segundos. Estas corresponden a las variables predictoras o de entrada para el algoritmo de predicción. Además, cuenta con 4 variables de respuesta que son de naturaleza binaria (0 y 1). De estas últimas se toma para el trabajo la variable denominada "alarma" como dato de salida, donde el valor 0 indica un estado normal del aerogenerador y el valor 1 da la señal de posible fallo. Las primeras filas de esta base de datos para las variables predictoras se pueden ver en la Tabla 13 y las variables de respuesta en la Tabla 14.

Tabla 13*. Variables predictoras de base de datos.*

	Fecha	v_rotor	P_estator	P_rotor	T_amb	T_estator	T_anillos_gen	V_viento
0	2011-11-02 00:00:00	11.812166	220.833335	-0.0125	12.477083	62.391669	25.025000	4.180000
1	2011-11-02 00:00:30	11.866142	303.890627	-0.0125	12.023959	62.405211	25.202086	5.173333
2	2011-11-02 00:01:00	11.865859	310.328127	-0.0125	12.582814	62.465105	25.465626	4.960000
3	2011-11-02 00:01:30	11.758332	291.963545	-0.0125	12.383855	62.562500	25.851044	4.633333
4	2011-11-02 00:02:00	11.879600	360.395837	-0.0125	11.477607	62.583333	25.810939	5.440000
…	…	…	…	…	…	…	…	…
865748	2012-08-28 14:34:00	11.792049	401.984377	-0.0125	15.007810	74.945310	28.296880	5.593333
865749	2012-08-28 14:34:30	11.825341	392.713545	-0.0125	15.007810	74.890630	28.257810	5.500000
865750	2012-08-28 14:35:00	11.832141	381.765627	-0.0125	15.070310	74.898440	28.195310	5.406667
865751	2012-08-28 14:35:30	11.845458	363.812503	-0.0125	15.085940	75.015630	28.242190	5.213333
865752	2012-08-28 14:36:00	11.818825	363.750002	-0.0125	15.093750	74.984380	28.226560	5.160000

865753 rows x 8 columns

Fuente: Elaborado por el autor.

Tabla 14. *Variables de respuesta de la base de datos.*

	funcionamiento	alarma	tipo_alarma	prioridad_alarma
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
…	…	…	…	…
865748	1	0	0	0
865749	1	0	0	0
865750	1	0	0	0
865751	1	0	0	0
865752	1	0	0	0

865753 rows x 4 columns

Fuente: Elaborado por el autor.

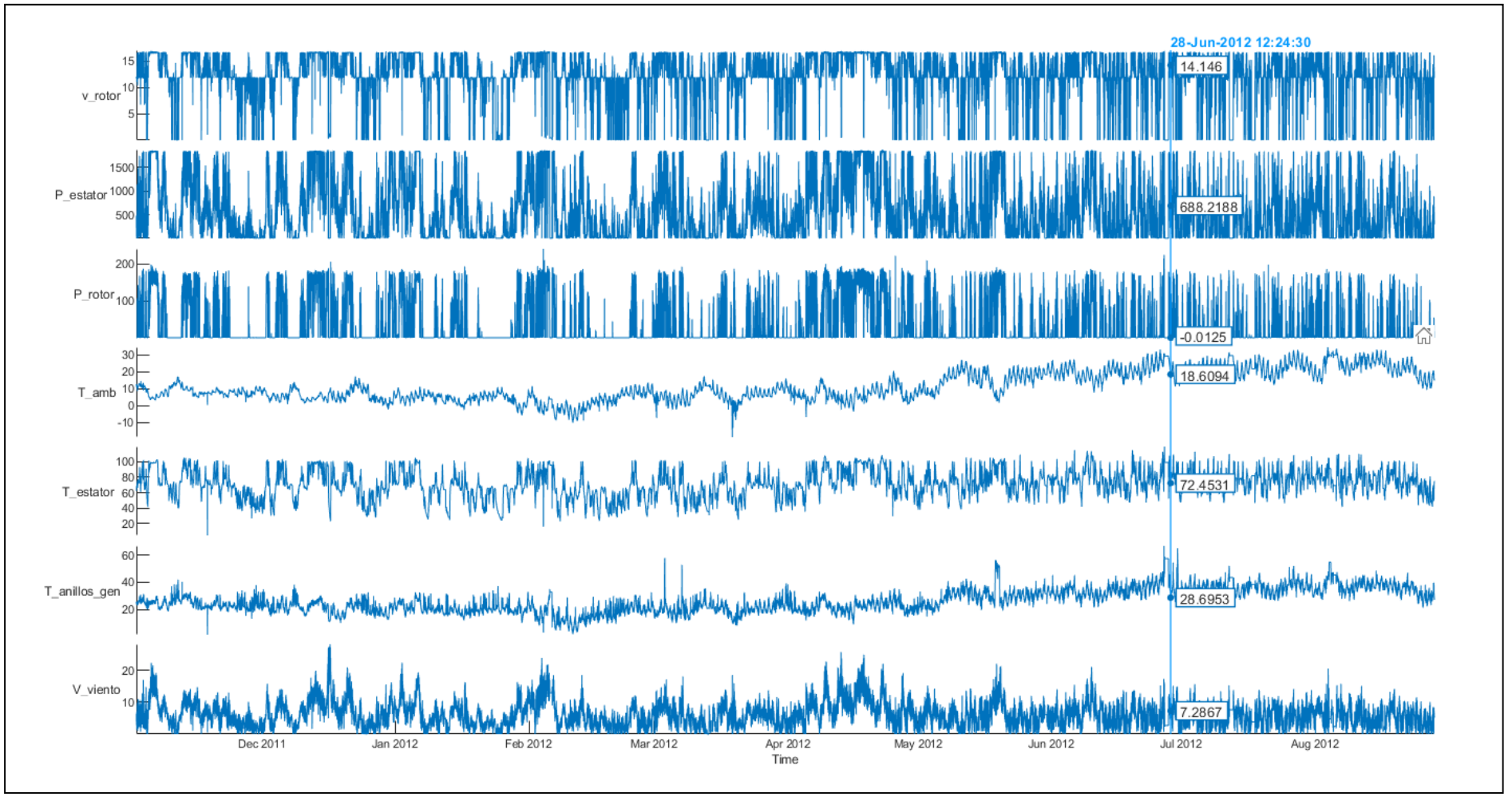

Figura 23.Tendencia de las variables predictoras a lo largo del tiempo de estudio.
Fuente: Elaborado por el autor.

Las variables predictoras que intervienen se enumeran en la Tabla 15.

Tabla 15. *Resumen de las variables predictoras*

Abreviatura	Variable	Unidad
v_rotor	Velocidad del rotor	rpm
P_estator	Potencia del estátor	kW
P_rotor	Potencia del rotor	kW
T_amb	Temperatura ambiente	ºC
T_estator	Temperatura del estator	ºC
T_anillos_gen	Temperatura anillos del generador	ºC
V_viento	Velocidad del viento	m/s

Fuente: Elaborado por el autor.

A continuación, se describen las variables a utilizar:

1) Potencia producida en el estator (P_estator). La potencia producida en el estátor del generador tiene una gran influencia en la temperatura de éste. Cuando la potencia producida es alta, la corriente del estátor también lo será, lo que conduce a una elevación de la temperatura.
2) Potencia producida en el rotor (P_rotor). Al igual que sucede con la potencia producida en el estátor, la potencia producida en el rotor tiene una gran influencia en la temperatura, es este caso del cuerpo de anillos. Cuando esta potencia sea alta, la corriente del estátor también lo será, y como consecuencia aumenta la temperatura en el cuerpo de anillos.
3) Temperatura ambiente (T_amb). Dado que la temperatura ambiente en la que trabajan los aerogeneradores experimentan grandes cambios en el corto (día y noche) y en el largo plazo (semanas a meses) debido al paso de frentes meteorológicos y periodos estacionales, debe tener cambios de temperatura de casi 30 ºC de un mes a otro.
4) Temperatura de los anillos del generador (T_anillos_gen).
5) Temperatura del devanado del estátor (T_estator).
6) Velocidad del rotor (v_rotor). La velocidad del rotor tiene un impacto directo en la temperatura del cuerpo de anillos. Además, una velocidad alta del rotor implica alta producción de energía, por lo que esta variable está estrechamente relacionada con la temperatura en el generador.
7) Velocidad de viento (V_viento). Aunque la velocidad del viento implica producción de energía, que a su vez significa elevación de las temperaturas en el generador, en el

estudio solamente se utiliza esta variable para comprobar, junto con las variables de potencia y velocidad del rotor, que las caídas a 0 estas variables no son producidas por la falta de recurso eólico, si no relacionadas con temperatura. Esta variable nos servirá para confirmar que la causa de la alarma registrada por el sistema SCADA, está relacionada con la temperatura.

6.2 Análisis Estadístico

6.2.1 Resumen Estadístico

En la Tabla 16 se presenta un resumen de las principales medidas de tendencia central de las variables de estudio para el presente trabajo de investigación, que sirvieron para construir los diagramas de caja a partir de la mediana y los cuartiles. También son base para realizar el escalado y estandarizado de los datos.

Tabla 16. *Resumen estadístico de las variables de estudio*

	V_rotor	P_estator	P_rotor	T_amb	T_estator	T_anillos_gen	V_viento
Max	17,00	1,875,23	240,85	34,61	119,53	66,73	28,16
Media	11,94	572,21	33,53	11,46	71,76	26,24	6,21
Mediana	11,87	401,28	-0,01	9,00	71,93	24,91	5,77
Mínimo	0,00	-3,03	-0,01	-18,49	5,00	1,47	0,00
Moda	0,00	0,00	-0,01	29,40	69,30	21,00	0,00
Desviación estándar	4,74	590,41	58,04	8,55	17,22	8,47	3,68
Desviación Media Absoluta	3,14	488,61	47,97	7,29	13,57	6,91	2,93
Varianza	22,43	348584,10	3369,15	73,14	296,44	71,77	13,55
25%	11,79	34,55	-0.01	5,05	61,55	20,00	3,32
50%	11,87	401,28	-0.01	9,00	71,92	24,91	5,77
75%	16,04	894,30	52,07	18,57	83,52	32,27	8,42

Fuente: Elaborado por el autor.

6.2.2 Resumen Gráfico

6.2.2.1 Diagramas de Caja

A continuación, se muestran los diagramas de cajas que ayudaron a resaltar aspectos de la distribución de los datos de las variables estudiadas, tales como la mediana, cuartiles, bigotes o límites superior e inferior y datos atípicos.

La Figura 24(a) presenta el diagrama de caja para la potencia del rotor, donde se puede observar que el primer cuartil y la mediana comparten el mismo punto, esto es -0,01 kW, y el tercer cuartil corresponde a 52,07 kW. A partir de estos valores se puede calcular el rango intercuartil (IQR) que es el resultado de restar el primer cuartil al tercer cuartil, dando un valor

de 52,08 kW. Con esto se determina la ubicación del bigote superior, que representa el punto de datos más grandes a una distancia de 1.5 IQR del tercer cuartil, siendo 130,19 kW, identificando de esta manera los puntos atípicos que van más allá de este valor, representados por puntos rojos en la gráfica.

Por su parte, en la Figura 24(b) se visualiza la distribución de la potencia del estator, donde la mediana corresponde al valor de 401,28 kW, el primer cuartil se encuentra en 34,54 kW, mientras que el tercer cuartil en 894,30 kW. El IQR es de 859,75 kW, por tanto, los valores para los límites superior e inferior son 2183,93 y -1255.08 kW respectivamente. Sin embargo, estas cantidades sobrepasan el valor máximo y mínimo de la variable medida, por lo que, en este caso, el valor máximo pasa a ser el límite superior y el valor mínimo el límite inferior, por tanto, no existen datos atípicos.

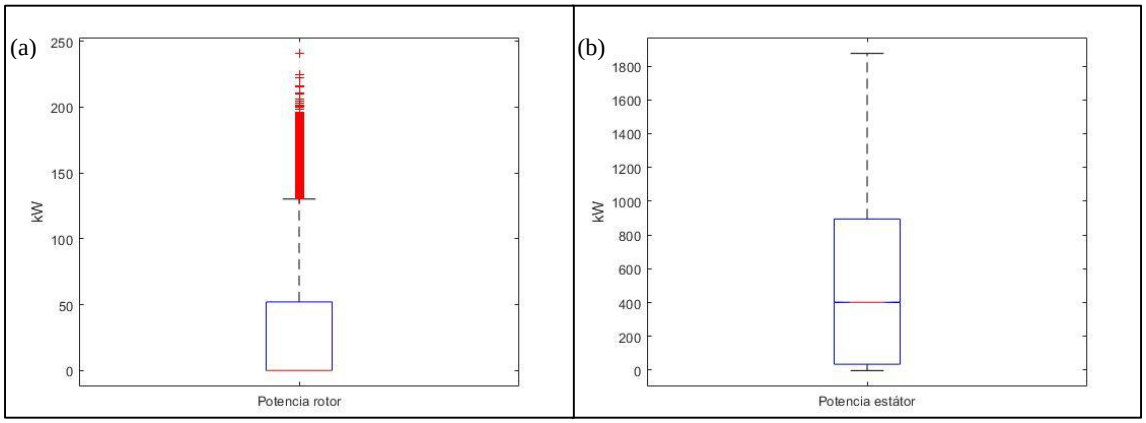

Figura 24. Diagramas de cajas para: (a) la potencia del rotor y (b) potencia del estátor.
Fuente: Elaborado por el autor.

La Figura 25 en cambio nos señala la existencia de datos atípicos en las variables de la velocidad del rotor, que se mide en rpm (revoluciones por minuto), y de la velocidad de viento, medida en m/s. Los cuartiles 1 y 3 para la velocidad del rotor son 11,79 y 16,03 rpm respectivamente, dando un IQR de 4,24 rpm, del cual se deriva el cálculo para el límite superior, 22,41 rpm, pero al existir observaciones de la variable hasta 17 rpm, este valor se toma como tal. El límite inferior es 5,45 rpm y, los valores menores a este se dibujan como atípicos. El cuartil 2 equivale a 11,87 rpm que es el mismo que la mediana.

Para la velocidad de viento, la mediana o cuartil 2 es 5,77 m/s, el primer y tercer cuartil son 3,32 y 8,24 m/s respectivamente, el rango intercuartil es 5, 09. El límite inferior es 0, igual al valor mínimo de la serie, ya que el que se calcula, -4,13 m/s, está por debajo. Los valores más allá del límite superior, 16,06 m/s, son considerados atípicos.

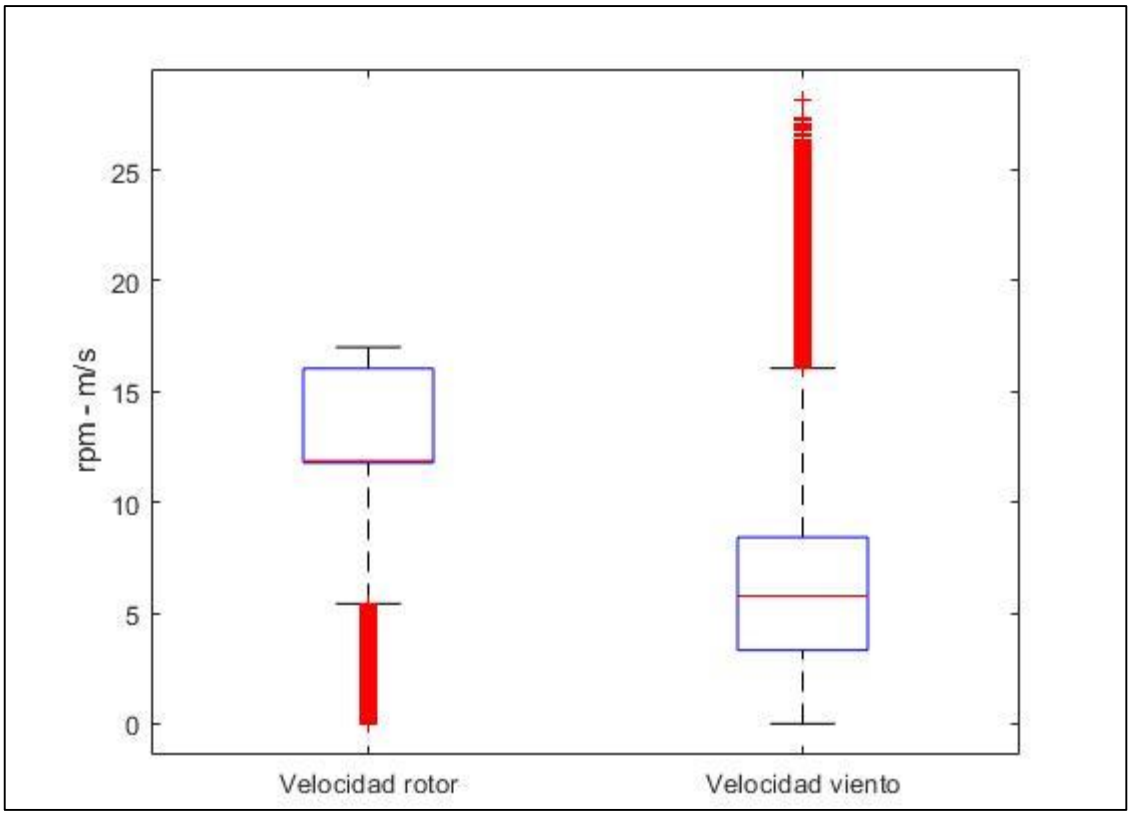

Figura 25. Datos atípicos en la velocidad de rotor y velocidad de viento.
Fuente: Elaborado por el autor.

Por último, en la Figura 26 se presentan las variables de temperatura, donde se aprecia que existen la mayoría de datos atípicos para la temperatura del estátor y del rotor. La mediana para la temperatura del estator es 71,92 ºC, mucho mayor que la mediana del cuerpo de anillos del generador, 24,90 ºC, que a su vez es mayor que la temperatura ambiente, 9 ºC. Los cuartiles 1 y 3 para la temperatura ambiente son 5,05 y 18,57 ºC respectivamente, para el estátor 61, 54 y 83,52 ºC y para el cuerpo de anillos, 20 y 32,27 ºC. los rangos intercuartil son 13,51 ºC para la temperatura ambiente, 21,97 ºC para el estátor y 12,27 ºC para el rotor.

El límite inferior para la temperatura ambiente corresponde a -15,21 ºC y el límite superior es 38,84 ºC. Para la temperatura del estator existen datos atípicos en los dos extremos de: el límite inferior cuyo valor es 28,58 ºC y el límite superior de 116,48 ºC. En el extremo adyacente al límite superior de la temperatura del cuerpo de anillos, 50,68 ºC, existen datos atípicos, mientras que son ausentes por debajo de 1,58 ºC que es el límite inferior.

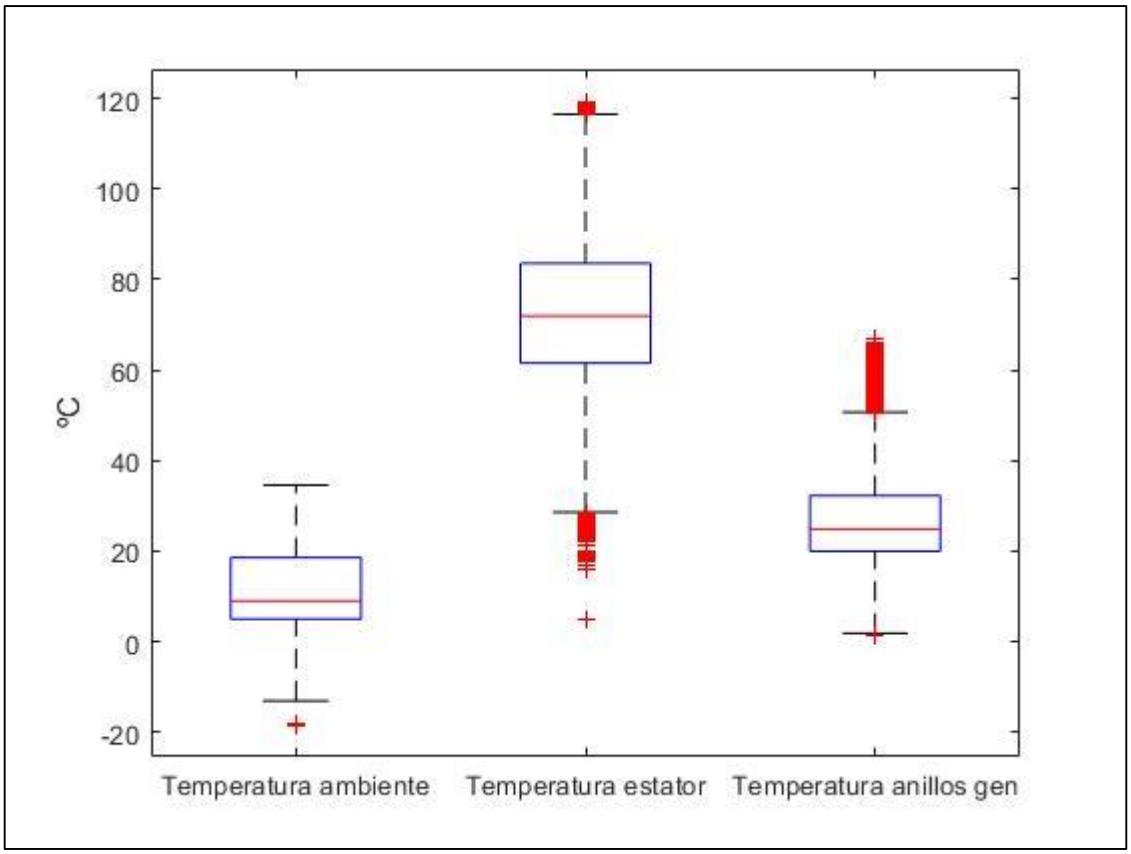

Figura 26. Datos atípicos en las variables de temperatura.
Fuente: Elaborado por el autor.

En la Tabla 17 se resumen los valores de los componentes principales de los diagramas de caja de las variables presentadas en este estudio

Tabla 17. *Valores de los principales componentes de los diagramas de caja presentados.*

	v_rotor	P_estator	P_rotor	T_amb	T_estator	T_anillos_gen	V_viento
Q1 (25%)	11,7915	34,5469	-0,0125	5,0547	61,5469	20,0000	3,3267
Q2 (50%)	11,8724	401.2813	-0,0125	9,0000	71,9297	24,9063	5,7733
Q3(75%	16,0389	894,3021	52,0725	18,5703	83,5234	32,2742	8,4200
IQR	4,2474	859,7552	52,0850	13,5156	21,9766	12,2742	5,0933
Li	5,4203	-1255,0859	-78,14	-15,2187	28,5820	1,5898	-4,3133
Ls	22,4101	2183,9348	130,2	38,8437	116,4882	50,6836	16,0599

Fuente: Elaborado por el autor.

6.2.2.2 Gráficos de distribución

En la Figura 27 se presenta la distribución de las observaciones para la temperatura del estátor en un histograma, donde se observa que la mayoría de datos se localizan en el rango de 60 a 90 ºC. También se puede corroborar que la distribución va desde alrededor de los 25 ºC hasta los 120 ºC y, que los datos atípicos, que son menores a 28.58 ºC no son visibles como sí lo son en el diagrama de caja y bigotes.

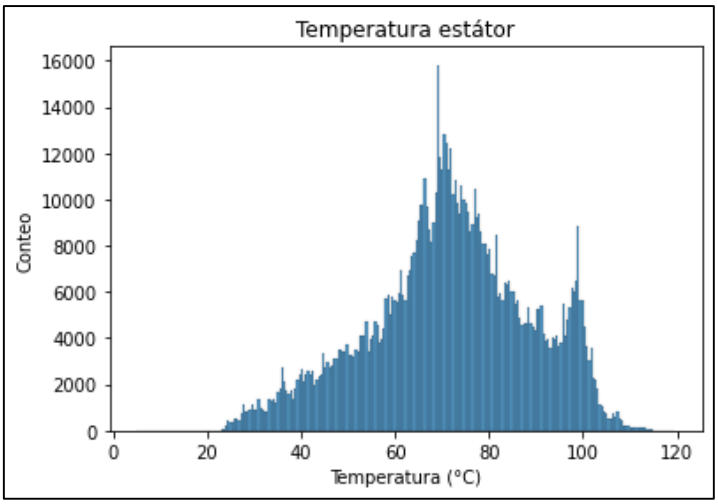

Figura 27. Distribución de la temperatura del estátor.
Fuente: Elaborado por el autor.

6.2.2.3 Gráficas KDE

El histograma presenta una gráfica con picos y, al existir una gran cantidad de datos, los rectángulos para cada clase de distribución se hacen cada vez más pequeños, entonces su visualización y un polígono de frecuencia no aportan mucho, por lo que se puede mejorar con un gráfico de estimación de densidad o KDE. La idea básica consiste en calcular para unos determinados puntos, la suma promediada de los Kernels o "elevaciones" centradas sobre las observaciones. En la Figura 28 se presenta la distribución de las variables estudiadas en gráficas KDE.

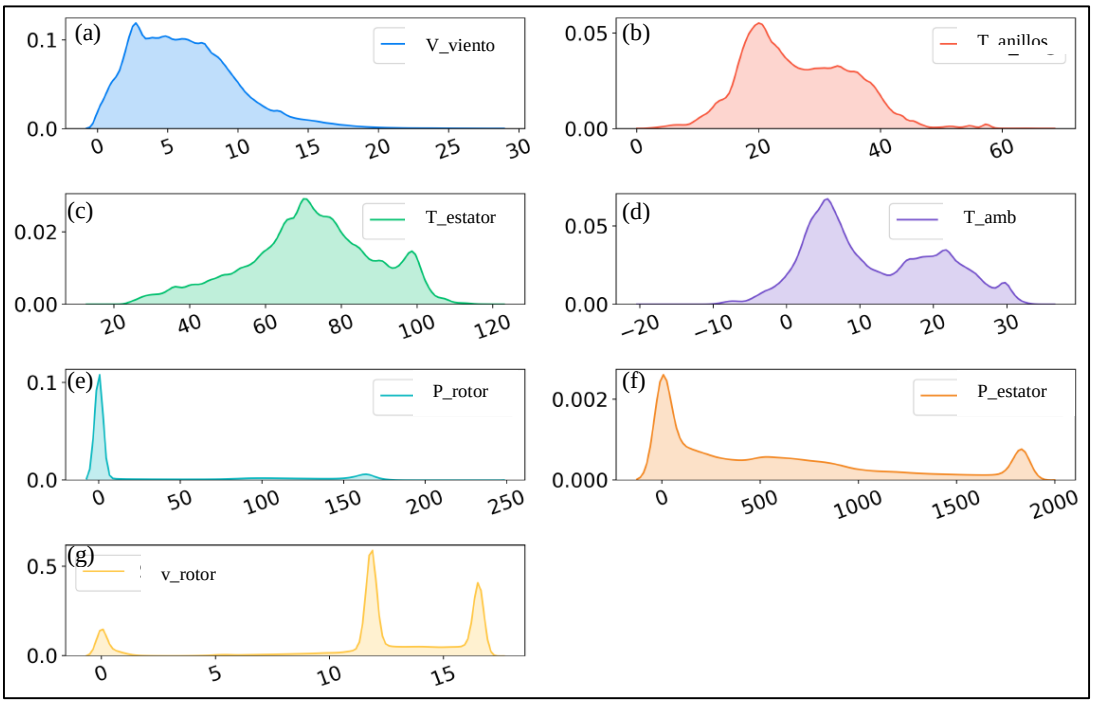

Figura 28. Gráficos de estimación de densidad o KDE para: (a) la velocidad de viento, (b) temperatura de anillos del generador, (c) temperatura del estator, (d) temperatura ambiente, (e) potencia del rotor, (f) potencia del estátor y, (g) velocidad del rotor.

Fuente: Elaborado por el autor.

6.3 Arreglo de la base de datos

6.3.1 División de datos

Al tratarse de un conjunto de datos con clases binarias se hizo el conteo de las mismas, cuyos datos se resumen a continuación:

Clase 0: 859791. Estado normal (Clase mayoritaria)

Clase 1: 5962. Estado Posible Falla (Clase minoritaria)

En la Figura 29 se ve claramente que existe un desbalance extremo de clases, siendo su ratio como sigue:

$$IR = \frac{|M|}{|m|}$$

$$IR = \frac{859791}{5962} = 144.21$$

Donde $|M|$ es la cantidad de objetos en la clase mayoritaria, $|m|$ es la cantidad de objetos en la clase minoritaria e IR indica que por cada objeto en la clase minoritaria, se tienen en promedio 144 objetos en la clase mayoritaria.

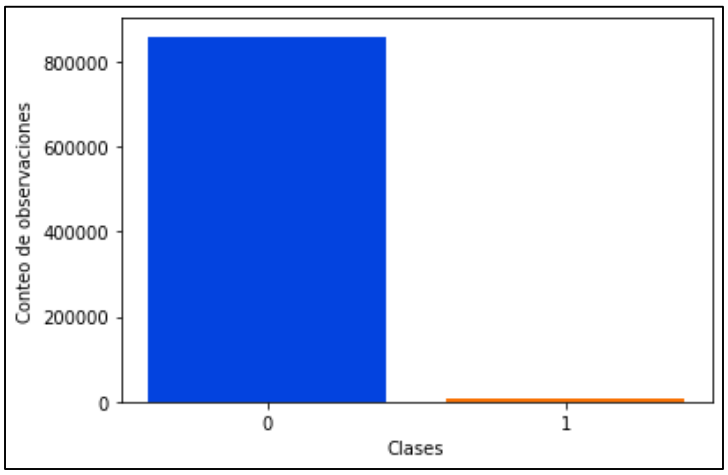

Figura 29. Desequilibrio de Clases.
Fuente: Elaborado por el autor.

A continuación, se realiza la división de datos en conjuntos de entrenamiento y prueba, con una razón de 70:30, es decir, el 70% de los datos para el entrenamiento y el 30% para la evaluación del modelo. Ambos conjuntos constan de una variable objetivo de dos clases para la predicción, siendo 0 la clase mayoritaria, la cual indica un funcionamiento normal de la

máquina y 1 la clase minoritaria que indica un posible estado de falla. En la Figura 30 se muestran estos conjuntos separados, además del conteo de clases para cada uno en la

Tabla ***18***.

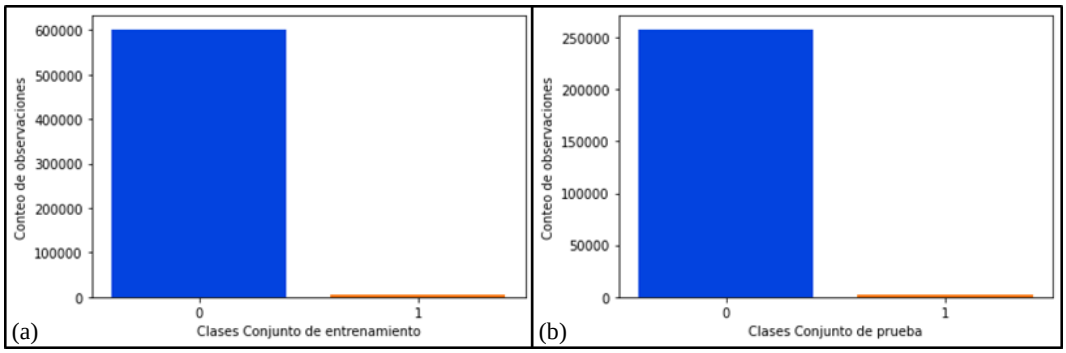

Figura 30. División del conjunto de datos en: (a) conjunto de entrenamiento y (b) conjunto de prueba.
Fuente: Elaborado por el autor.

Tabla 18. *Número de instancias para cada clase.*

	Conjunto de entrenamiento	**Conjunto de prueba**
Clase 0	601867	257 924
Clase 1	4 160	1802

Fuente: Elaborado por el autor.

6.3.2 Remuestreo de Datos Desequilibrados

Debido al desbalance existente en el conjunto de entrenamiento, los modelos de clasificación presentan una inclinación a favorecer la predicción de instancias de la clase mayoritaria, lo que se conoce como sesgo de aprendizaje de los modelos de machine learning.

Con el objeto de disminuir el sesgo de clasificación de datos, se realizó el remuestreo de los mismos en el conjunto de entrenamiento sobre diferentes métodos de submuestreo y sobremuestreo. La eficacia de éstos métodos se validó con el algoritmo de clasificación de Árboles de Decisión (Decision Tree), evaluando el rendimiento del clasificador con las métricas ROC AUC, Recall y F1-score.

La Tabla 19 revela que el submuestreo aleatorio brinda mejores resultados con referencia a la métrica Recall. El submuestreo aleatorio involucró la selección de ejemplos de

la clase mayoritaria para borrarlos del conjunto de datos. Para el presente caso se empleó este método de remuestreo únicamente en el conjunto de entrenamiento como evidencia la Figura 31 y no en el conjunto de prueba, para así disminuir el sesgo de clasificación al presentar nuevos datos. Sin embargo, el riesgo de perder información valiosa es indudable debido a la gran cantidad de datos que se eliminaron de la clase mayoritaria

Tabla 19. *Técnicas de remuestreo evaluadas en el algoritmo de clasificación Decision Tree.*

TÉCNICAS DE REMUESTREO	Conteo de muestras		Métricas de evaluación			IR
	Clase 0	Clase 1	AUC	Recall	F-1 score	
Submuestreo Aleatorio	4160	4160	0,952	0,956	0,198	1
Submuestreo Aleatorio (sampling strategy = 0,5)	8320	4160	0,951	0,938	0,260	2
Near Miss Undersampling (versión 1)	4160	4160	0,558	0,916	0,016	1
SMOTE y submuestreo aleatorio	120372	60186	0,938	0,887	0,398	2
SMOTE and Edited Nearest Neighbors Undersampling	600543	585313	0,930	0,873	0,453	1
SMOTE and Tomek Links Undersampling	601867	601834	0,913	0,839	0,503	1
Synthetic Minority Oversampling (SMOTE)	601867	601867	0,915	0,837	0,504	1
Adaptative Synthetic Sampling (ADASYN)	601867	601867	0,912	0,831	0,489	1
Sobremuestreo y submuestreo aleatorio	120372	60186	0,898	0,807	0,596	2
Borderline-SMOTE SVM	601867	601867	0,897	0,801	0,579	1
Borderline-SMOTE	601867	601867	0,895	0,785	0,567	1
Edited Nearest Neighbors Rule for Undersampling (ENN)	596017	4160	0,872	0,747	0,729	143
Neighborhood Cleaning Rule for Undersampling	595467	4160	0,862	0,719	0,678	143
One-Sided Selection for Undersampling	231870	4160	0,848	0,694	0,590	55
Tomek Links for Undersampling	600931	4160	0,844	0,676	0,676	144
Conjunto de datos desbalanceado	601867	4160	0,839	0,668	0,673	144
Sobremuestreo Aleatorio	601867	601867	0,830	0,664	0,694	1

Fuente: Elaborado por el autor.

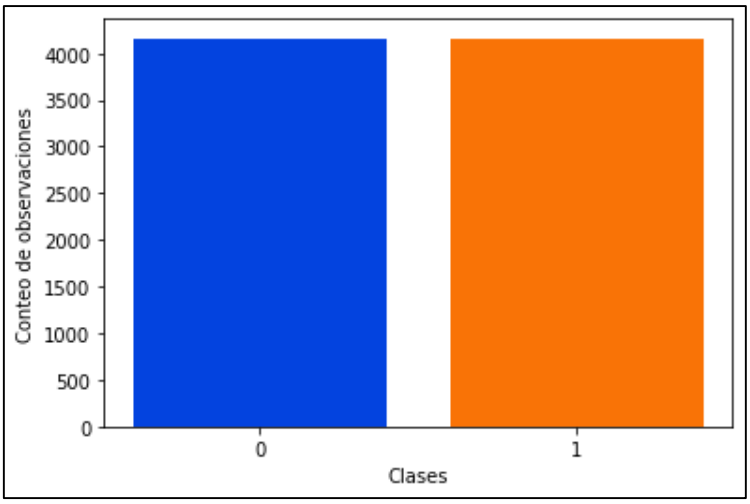

Figura 31. *Conjunto de datos de entrenamiento submuestreado.*
Fuente: *Elaborado por el autor.*

Para un mejor entendimiento del proceso de submuestreo ejecutado, en la Figura 32 se visualizan en diagramas de dispersión las variables de Temperatura medidas en el estátor del generador frente a la Potencia producida en el mismo. En la Figura 32(a) se grafican los puntos de la clase mayoritaria y de la clase minoritaria para conjunto de datos original. Puesto que el proceso consiste en disminuir la clase 0, en la Figura 32(b) se proyectan únicamente las observaciones de esta para conocer su dimensión.

Una vez llevado a cabo el submuestreo aleatorio, la Figura 32(c) manifiesta este hecho con las dos clases igualadas en representación de instancias, y de manera particular, en la Figura 32(d) se visualizan los datos de la clase 0 ya submuestreada, separados de la clase 1 que se grafican en la Figura 32(e).

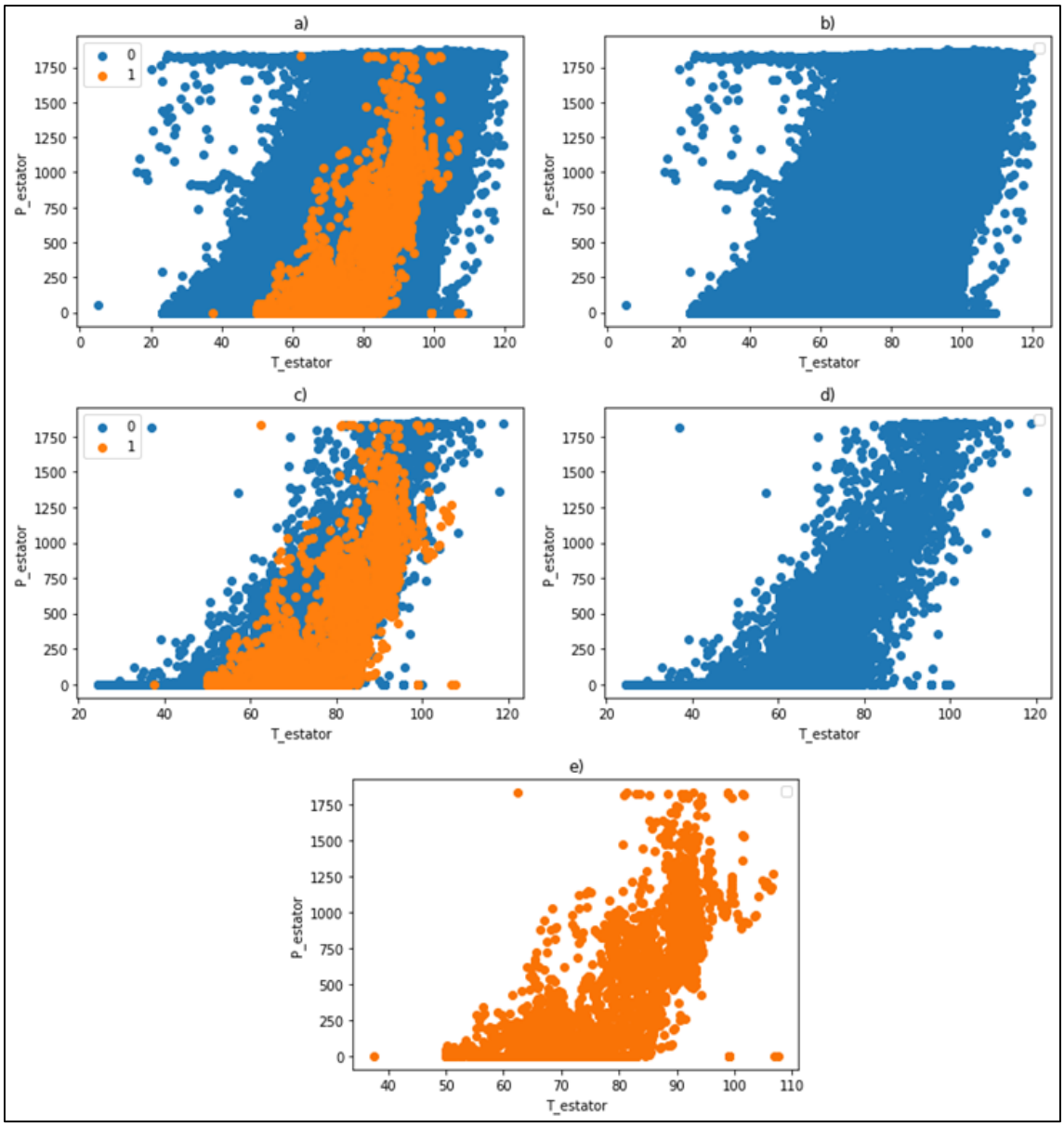

Figura 32. Proceso gráfico de submuestreo aleatorio: (a) representación de los datos del conjunto de entrenamiento original, (b) datos de la clase mayoritaria, (c) clases 0 y 1 igualadas, (d) instancias de la clase 0 submuestreada y, (e) instancias de la clase 1 sin modificar.

Fuente: Elaborado por el autor.

6.3.3 Escalado de datos

Las variables utilizadas tienen diferentes unidades (velocidad de viento, temperaturas, potencia, etc.) y sus valores absolutos son muy diferentes, los valores iniciales se re-escalaron en un rango de 0 a 1, de acuerdo con sus valores máximos y mínimos. En la Tabla 20 se muestran las primeras filas del conjunto de entrenamiento con este método de escalado también denominado normalización.

Tabla 20. *Normalización de datos del conjunto de entrenamiento.*

v_rotor	P_estator	P_rotor	T_amb	T_estator	T_anillos_gen	V_viento
0,5697	0,0011	0.0	0,0367	0,2737	0.0620	0,1556

0,5669	0,0011	0.0	0,0377	0.2695	0.0611	0,1564
0,5728	0,0011	0.0	0,0393	0.2654	0.0611	0,1591
0,5824	0,0011	0.0	0,0418	0.2628	0.0611	0,1644
0,5944	0,0011	0.0	0,0421	0.2579	0.0611	0,1657

Fuente: Elaborado por el autor.

La Figura 33(a) y la Figura 33(b) presentan gráficas de línea para las variables de temperatura del estátor y de la velocidad de viento respectivamente, constatando que ambas variables se miden en diferentes unidades y, por tanto, al aplicar normalización, estas se ajustan a la misma escala de 0 a 1, como demuestran la Figura 33(c) y la Figura 33(d).

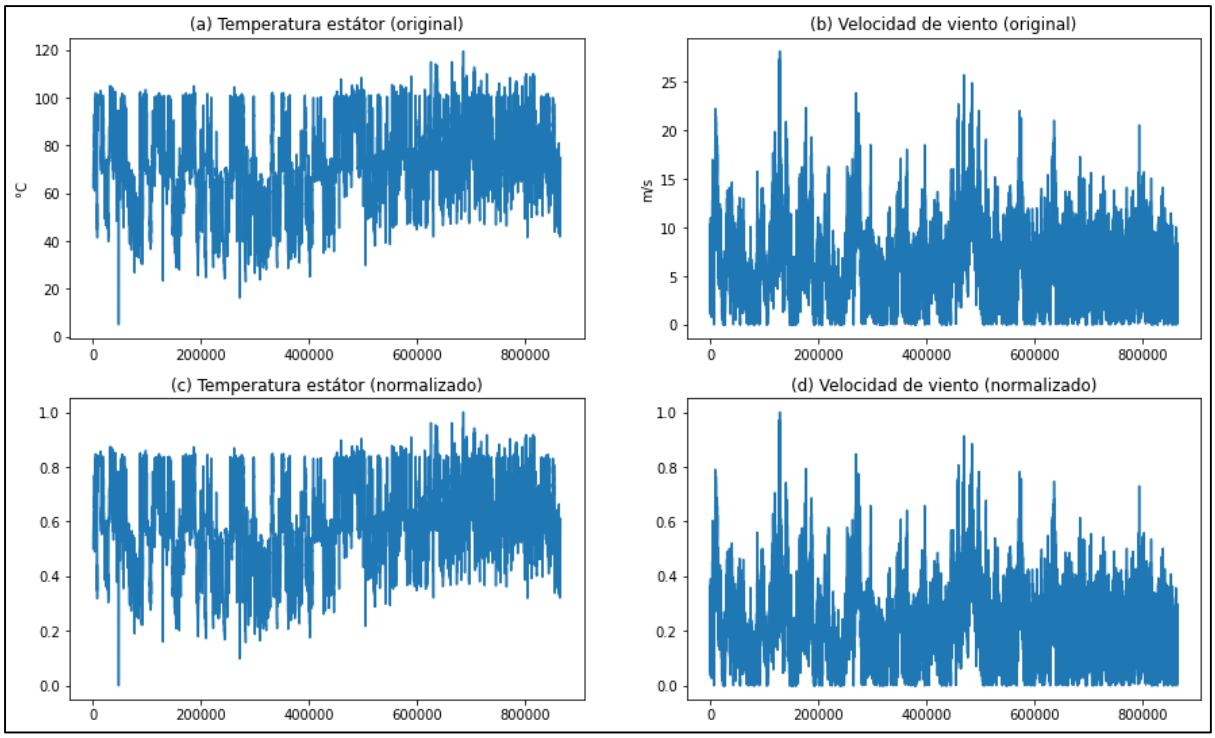

Figura 33. Gráficos de línea para: (a) datos originales de la temperatura del estátor, (b) datos originales de la velocidad de viento, (c) datos normalizados de la temperatura del estátor y, (d) datos normalizados de la velocidad de viento.

Fuente: Elaborado por el autor.

La estandarización por su parte implicó cambiar la distribución de valores de cada variable para que la media de las observaciones sea 0 con una desviación estándar de 1, como se presentan en Tabla 21 para las primeras filas.

Tabla 21. *Estandarización de datos del conjunto de prueba.*

v_rotor	P_estator	P_rotor	T_amb	T_estator	T_anillos_gen	V_viento
-0,0157	-0,4549	-0,5781	0,0665	-0,5423	-0,1219	-0,2815
-0,0158	-0,4440	-0,5781	0,1319	-0,5388	-0,0907	-0,3395
-0,0385	-0,4751	-0,5781	0,1086	-0,5331	-0,0452	-0,4283

-0,0129	-0,3592	-0,5781	0,0027	-0,5319	-0,0500	-0,2091
-0,0258	-0,4754	-0,5781	-0,0144	-0,5242	-0,1342	-0,2689

Fuente: Elaborado por el autor.

La normalización de datos se utilizó para el entrenamiento del algoritmo de aprendizaje con una Red Neuronal Convolucional, mientras que para los demás algoritmos de clasificación se usó tanto la estandarización.

6.4 Clasificación Binaria de Datos Desequilibrados

6.4.1 Algoritmos Clásicos de Machine Learning

En este acápite se evalúan un grupo de algoritmos de Machine Learning clásicos para clasificación y, con el propósito de analizar su desempeño, se utilizan las métricas de exactitud y precisión con los conjuntos de entrenamiento y prueba desequilibrados del caso de estudio, los modelos de clasificación binaria se entrenan utilizando las configuraciones predeterminadas, es decir, para este caso no se ajustan hiperparámetros para obtener las configuraciones apropiadas.

En la Tabla 22 se observa los resultados de los clasificadores. Se comparan las métricas de exactitud (accuracy) con la precisión de cada clase. Los datos muestran que los modelos de Bagging y MLP alcanzan puntuaciones aceptables. Sin embargo, se debe tener en cuenta el aprendizaje sensible al costo en mantenimiento por lo que se deben buscar métricas de evaluación alternativas como las basadas en la matriz de confusión.

Tabla 22. *Clasificación de los datos con las características originales.*

Algoritmo	**Exactitud**	**Precisión clase 0**	**Precisión clase 1**
Bagging	0,997	0,997	0,867
Random Forest	0,996	0,997	0,907
Extra Tress	0,996	0,996	0,956
Decision Tree	0,995	0,998	0,668
k-Nearest Neighbors	0,994	0,995	0,731
Gradient Boosting	0,994	0,994	0,714
Suport Vector Machine	0,993	0,993	0,000
Logistic Regresión	0,993	0,993	0,042
Naive Bayes	0,933	0,996	0,052
MLP	0,993	0,997	0,867

Fuente: Elaborado por el autor.

A continuación, se realiza la clasificación con los datos remuestreados con submuestreo aleatorio y el normalizado de los mismos. Además, se varía los hiperparámetros y se aplica el método de validación cruzada estratificada para evaluar el rendimiento del modelo durante el entrenamiento.

Para la comparación de los modelos de predicción se toma como referencia la métrica de evaluación Recall, que cuantifica el número de predicciones positivas correctas realizadas de todas las predicciones positivas que se podrían haber hecho.

Dado que se desea minimizar los falsos negativos, es decir, predecir para la clase positiva, en nuestro caso presencia de alarma, es decir, el 100% de las muestras, el Recall es el más indicado.

La precisión en cambio nos proporciona información sobre el rendimiento del modelo con respecto a los falsos positivos. Para los modelos presentados, la precisión para la clase 1 es cercana al 100%. Mientras que para la clase 0 los valores son bastantes bajos. Esto se da al mantenerse el conjunto de datos de prueba aún desbalanceado.

La Tabla 23 muestra que la mayoría de los algoritmos evaluados alcanzan valores de Recall y ROC AUC por encima del 90% de los que destaca el modelo Extra Trees con un 97.3% en ambas métricas de evaluación.

Tabla 23. *Evaluación del desempeño de los modelos de clasificación empleados.*

Algoritmo/Modelo	F1 score Macro	Recall Macro	Precisión Macro	Recall clase 1	Recall clase 0	Precisión clase 1	Precisión clase 0	F1 clase 1	F1 clase 0	ROC AUC	Exactitud
Random Forest	0,530	0,940	0,530	0,990	0,900	0,060	1,000	0,120	0,950	0,940	0,900
Extra Trees	0,616	0,973	0,573	0,987	0,959	0,145	0,999	0,253	0,979	0,973	0,959
XGBoost	0,570	0,960	0,547	0,986	0,934	0,095	0,999	0,174	0,966	0,960	0,935
Gradient Boosting	0,599	0,968	0,563	0,985	0,952	0,126	0,999	0,223	0,975	0,968	0,953
Baggin	0,598	0,968	0,562	0,985	0,951	0,125	0,999	0,221	0,975	0,968	0,952
Logstic Regression	0,498	0,899	0,521	0,950	0,847	0,042	0,999	0,080	0,917	0,899	0,848

Fuente: Elaborado por el autor.

La matriz de confusión del para el clasificador Extra Trees se presenta en la Tabla 24, donde se puede ver de manera directa la cantidad de observaciones estimadas por el modelo. Si se calcula la predicción positiva de observaciones para la clase 1, es decir, el Recall, se obtiene un valor de 0,987 en el rango de 0 a 1, entendiéndose por una buena predicción. El cálculo de Recall para la clase 0, es en realidad el valor predictivo negativo o NPV y su valor es 0,959. Con estos resultados se obtiene la media aritmética de Recall, es decir, el Recall Macro que toma en cuenta la certeza de predicción de ambas clases, siendo igualmente satisfactoria con un valor de 0,973.

Tabla 24. *Matriz de confusión para la predicción realizada por Extra Trees.*

Número total de observaciones 259726		**Predicción**		
		Negativo	**Positivo**	**Total**
Observación	**Negativo**	TN 247476	FP 10448	Clase 0 257924
	Positivo	FN 23	TP 1779	Clase 1 1802

Fuente: Elaborado por el autor.

La curva ROC del modelo se visualiza en la Figura 34. Esta confirma la excelente estimación del modelo con respecto a la tasa de verdaderos positivos, que es 0.97, y la tasa de falsos positivos tiene un valor de 0.04, lo bastante bajo como para ubicar el punto de rendimiento en la izquierda superior del espacio ROC dando a conocer por tanto un desempeño satisfactorio.

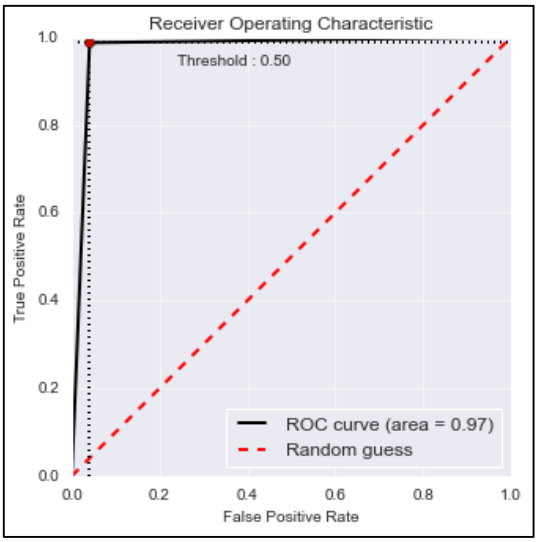

Figura 34. Curva ROC del modelo Extra Trees.
Fuente: Elaborado por el autor.

Los modelos fueron entrenados aplicando el método de validación cruzada estratificada repetida k veces, con 10 repeticiones y 20 pliegues. Los hiperparámetros que se modificaron se muestran en la Tabla 25.

Tabla 25. *Ajuste de Hiperparámetros del modelo Extra Trees.*

Hiperparámetros	**Valor**
Número de árboles	500
Número de características	7
Número de muestras	2

Fuente: Elaborado por el autor.

6.4.2 Clasificación con Red Neuronal Convolucional (CNN)

Para la clasificación de datos con la red CNN se utilizó otro conjunto en el que constan únicamente 6 días de observación donde el número de instancias de la clase positiva es mucho mayor que en todo el periodo del nuevo estudio.

Precisamente los días utilizados tanto para entrenamiento como para prueba se presentaron en la Tabla 12, esto luego de realizar una exploración de las variables a lo largo del tiempo como muestra la Figura 35, donde se señalan las instancias para cada clase de la variable temperatura del estátor, confirmando el agrupamiento de la clase positiva mayormente al inicio y final de la serie.

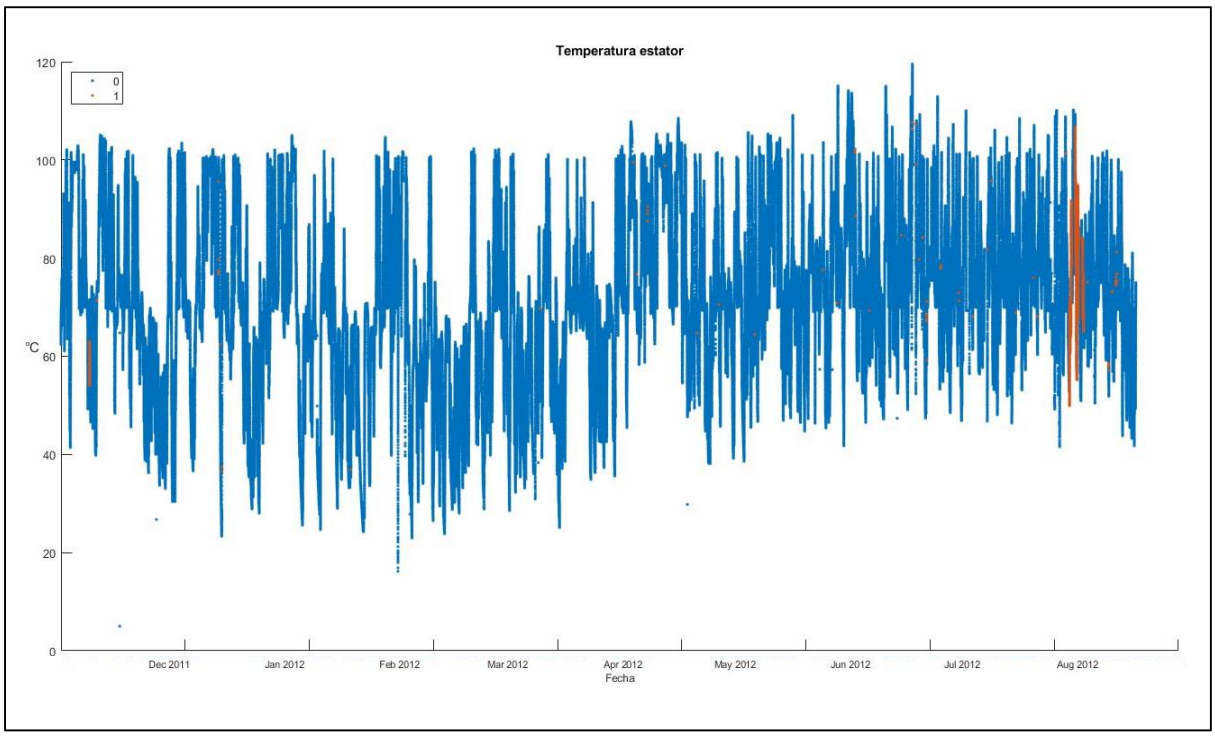

Figura 35. Visualización de las clases negativas y positivas en la variable Temperatura del estátor.
Fuente: Elaborado por el autor.

A continuación, los datos son normalizados y se arreglan de tal manera que sean una entrada tridimensional a la red CNN. La arquitectura dela red CNN se representa en la Figura 36 y consta de los siguientes elementos:

- Tres capas ocultas convolucionales con 64 filtros de salida por capa.
- Una capa de regularización del 50% (dropout).
- Una capa Maxpooling que agrupa la salida de la parte convolucional.
- Una capa densa con 64 neuronas.
- Una capa densa de salida con una neurona.

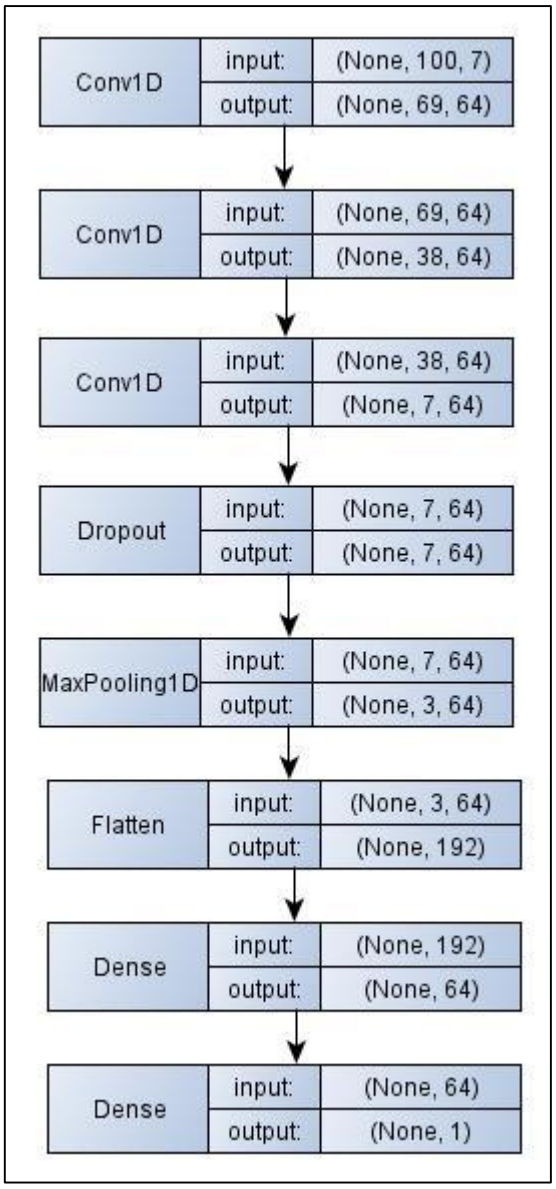

Figura 36. Arquitectura modelo CNN.
Fuente: Elaborado por el autor.

Se probaron diferentes configuraciones con respecto a los hiperparámetros de la red, como son pasos de tiempo, tamaño de lote y el número de épocas en cada iteración de entrenamiento como se listan en la Tabla 26.

Tabla 26. *Variación de hiperparámetros.*

Hiperparámetros	**Valor**
Pasos de tiempo	40, 80,100
Tamaño de lote	16, 32, 64, 128, 256, 512
Número de épocas	20, 50, 100

Fuente: Elaborado por el autor.

El mejor modelo de predicción se obtiene al aumentar el paso de tiempo hasta 100, disminuir el tamaño de lote a 16 y mantener un número de épocas en un valor de 20, con resultados de especificidad del 60,8% y Recall de 79,5% de acuerdo a la matriz de confusión presentada en la Tabla 27, aunque refleja una gran cantidad de falsos positivos y falsos negativos detectados.

***Tabla 27**. Matriz de confusión del modelo mejor puntuado para la red CNN.*

Nº total de observaciones 5758		**Predicción**		
		Negativo	**Positivo**	**Total**
Observación	**Negativo**	TN 29377	FP 1892	Clase 0 4829
	Positivo	FN 190	TP 739	Clase 1 929

Fuente: Elaborado por el autor.

Finalmente, en la Figura 37 se evalúa la curva de aprendizaje del modelo CNN, lo que corrobora un pobre desempeño del modelo, si bien el error de predicción para la validación disminuye a 1,65% en la época 7, se presenta un sobreajuste a partir de este punto por lo que se configura en el modelo una función de parada temprana para detener el entrenamiento.

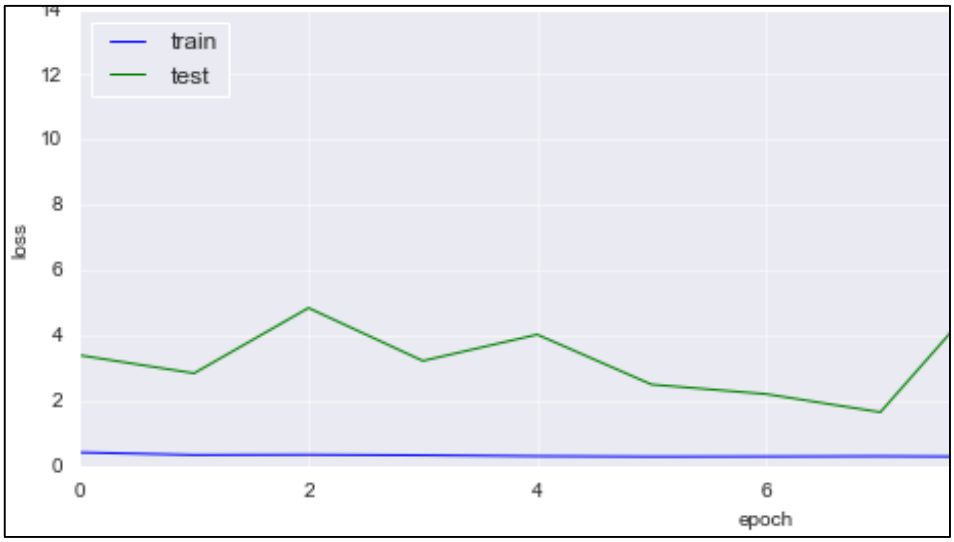

Figura 37. Curva de aprendizaje del modelo CNN.
Fuente: Elaborado por el autor.

7. Discusión

En el presente trabajo se realizó la predicción de datos del generador de una turbina eólica tripala de eje horizontal, aplicando técnicas de clasificación de Machine Learning y Deep Learning para analizar los datos de entrada y validar el estado normal de funcionamiento de la máquina o un posible fallo. Las variables analizadas en el generador son: potencia del estator, potencia del rotor, temperatura del estator, temperatura del rotor, temperatura ambiente, velocidad del rotor y velocidad de viento, mismos que son trabajados por Martínez (2017), donde emplea la técnica de estimación de estado no lineal NSET, cuyo modelo es un método estadístico que estima la ocurrencia de una avería en tiempo de estudio determinado, mientras que el que se despliega en esta investigación, clasifica las ocurrencias de falla o estado normal, por lo que no pueden compararse directamente; sin embargo, son dos procesos válidos para el monitoreo de un aerogenerador.

Martínez (2017), selecciona las variables valiéndose del criterio técnico, argumentando que se trabajan con aquellas que están estrechamente relacionadas con la temperatura del generador. La selección de las variables para este estudio se realizó mediante la correlación lineal tomando como variable dependiente la temperatura del estator, así, las variables independientes se seleccionan de acuerdo al índice de correlación. Además, se tiene en cuenta el criterio técnico para validar la selección de variables que interactúen directamente con la variable independiente.

Se llevó a cabo una comparativa de diferentes técnicas de remuestreo de datos, destacando el submuestreo aleatorio aplicado al conjunto de entrenamiento para eliminar el problema del desbalance de clases, cuyo índice es de 144, y evitar el sesgo de aprendizaje de los modelos de predicción, validando el desempeño con el clasificador Decision Tree que tiene relación con el algoritmo Random Forest utilizado por Kraiem (2020) en su trabajo para validar la eficacia de 7 estrategias de remuestreo de varios paquetes de datos, algunos de los cuales tienen tasas de desequilibrio de 129. Las métricas que este autor utiliza para evaluar el rendimiento del modelo son AUC, precisión y Recall siendo la última la que mejores valores presenta para el submuestreo aleatorio lo que se confirma en este trabajo junto con la métrica AUC.

Las técnicas de Inteligencia Artificial empleadas en esta investigación están fundamentadas en el trabajo de Huertas (2020), quien emplea algoritmos de machine learning y deep learning para predicción de fallos en el caso de simulación de un motor.

La predicción de fallas en el presente trabajo se hizo de dos maneras. Primero, se realizó el entrenamiento y validación de los datos sobre las técnicas de machine learning para

clasificación con los conjuntos de datos que abarcan los meses de estudio, de noviembre de 2011 a agosto de 2012 y que son tratados previamente con submuestreo aleatorio. El algoritmo que da mejores resultados es Extra Trees, con 98.7% de Recall en la predicción de la clase positiva, es decir, la clase de fallas, y 95.9% en la clase negativa, lo que da una buena impresión de la cantidad de datos detectados correctamente.

La métrica Recall empleada para evaluar y comparar el desempeño de los modelos mide el número de predicciones positivas que acierta el modelo de todas las positivas correctas que podrían haberse acertado, siendo de gran importancia, ya que es necesario tener mayores aciertos en lo que respecta a predicciones de fallas en un aerogenerador, teniendo en cuenta que una mala predicción provocaría una parada de la máquina con consecuente pérdida de tiempo y dinero.

El valor de Recall en Kraiem (2020) para clasificación con Decision Tree y Random Forest aplicando submuestreo aleatorio, es cercano al 80% en su estudio preliminar; en Huertas (2020) para Extra Trees se obtiene 86% de Recall, sin embargo opta por la métrica de F1-score, ya que únicamente busca minimizar la cantidad de falsos negativos detectados. Arnejo (2017) evalúa algoritmos de árboles de decisión con resultados bajos en Recall, pero con valores entre 55 y 75% para AUC; por su parte, Espinar (2018), también evalúa el clasificador Random Forest con la curva AUC, obteniendo un valor de 73% y es el mejor resultado tras aplicar submuestreo aleatorio frente métodos de sobremuestreo. Con lo expuesto anteriormente, se puede decir que en términos de Recall Macro, es decir, el valor de la media aritmética de Recall en ambas clases, el presente trabajo demuestra una evaluación superior con el 97%, igual valor que arroja la curva AUC, para la clasificación con el algoritmo Extra Trees, que en palabras generales pertenece a la familia de árboles decisión utilizados por los autores antes referenciados.

El segundo proceso consistió en la clasificación de datos con una Red Neuronal Convolucional, pero esta vez, el conjunto de datos, tanto para entrenamiento como para validación, fue modificado. Al hacer una exploración de los datos, se comprobó que la clase positiva, en su mayoría se agrupa en el día 10 del mes de noviembre, y en el mes de agosto en los días 10, 12, 13 y 23.

De tal manera que los 4 primeros días, en el orden descrito anteriormente, son tomados para entrenamiento, y los 2 restantes para validación. Los resultados obtenidos son los siguientes: de los 929 datos disponibles para la validación, 739 han sido clasificados correctamente como positivos, lo que da lugar al 79,5% en la métrica Recall, que, aunque es el mejor modelo de entre una serie de iteraciones con cambios en los hiperparámetros, no puede

considerarse como un modelo aceptable para la predicción de fallas en un aerogenerador. Cabe mencionar que, en este caso, no se realiza remuestreo alguno, ya que el desequilibrio de clases es bajo, y por su parte, Chen, Xu, Zhang & Zhang (2019) presentan una solución a nivel algorítmico para enfrentar el ratio de desbalance de clases cercano a 15 en tres casos de conjuntos de datos de turbinas eólicas para la detección de fallas en las palas.

8. Conclusiones

Con la finalización del presente trabajo investigativo, se ha llegado a las siguientes conclusiones:

- Se examinaron un total de 865753 datos del sistema SCADA, para determinar el comportamiento histórico del generador de un aerogenerador de eje horizontal. Al analizar los datos se encontraron valores atípicos, y se determinó el índice de desbalance IR= 144.2, el cual indica un marcado desbalance entre los datos o registros normales (sin fallas), y los datos anormales (fallas).

  Para solucionar el problema del desbalance de datos se utilizaron técnicas de remuestreo, y en las pruebas experimentales su obtuvo los mejores resultados con el método de submuestreo aleatorio.

  Luego de equilibrados los datasets, se normalizaron los datos con el propósito de igualar las magnitudes de las variables y contar con un conjunto de entrenamiento óptimo para aplicar los algoritmos de Machine Learning.

- Se analizaron seis diferentes arquitecturas de algoritmos de Machine Learning, y para encontrar el mejor desempeño se ajustaron sus hiperparámetros; siendo el algoritmo Extra Trees el que demostró el mejor desempeño en la clasificación de fallas, obteniendo en la fase de validación un Recall= 0.973, seguido de los modelos Gradient Boosting y Baggin que alcanzaron un Recall de 0.968.

  Además, para profundizar en el análisis de la clasificación y detección de fallas, se utilizaron redes neuronales convolucionales (CNN), las cuales se entrenaron con una muestra de datos diferente, y haciendo varias iteraciones en la búsqueda de los mejores hiperparámetros. Los resultados no superaron a los alcanzados por los algoritmos de Machine Learning, por lo que no es un modelo factible para la predicción de fallas en un generador de una turbina eólica.

- Se valoró el rendimiento y precisión de los algoritmos de Machine Learning y Deep Learning con ayuda de seis diferentes métricas derivadas de la matriz de confusión.

  Además, es necesario destacar que la métrica Recall, es la que más importancia tiene en este trabajo, ya que, es el indicador que mide el número de predicciones positivas correctas que aciertan los modelos entre el número total de predicciones positivas, lo que permite identificar fallas incipientes en el generador, reduciendo los costos de mantenimiento y mejorando la fiabilidad del aerogenerador.

9. Recomendaciones

A continuación, se presentan algunas recomendaciones para trabajos futuros a partir de los resultados actuales:

- Debido al riesgo de sobreajuste que se corre al entrenar los modelos de Machine Learning con el conjunto de entrenamiento submuestreado aleatoriamente, y, por tanto, perder una gran cantidad de información de la clase mayoritaria, se recomienda hacer un muestreo híbrido, es decir, primero hacer un submuestreo aleatorio al 50% y luego aplicar la técnica de sobremuestreo SMOTE.
- En el caso de entrenamiento y validación de datos para la clasificación de fallas con una red CNN, se sugiere revisar la longitud del kernel, ya que, si aumenta, se obtienen mejores resultados, así como también aumentar el número de capas convolucionales.
  También se podría aplicar modelos híbridos de redes neuronales profundas como es el caso de una red CNN-LSTM, con mayor enfoque en series de tiempo.

10. Bibliografía

Academy, K. (n.d.). Polígonos de frecuencia. Retrieved from Polígonos de frecuencia website: https://es.khanacademy.org

Arnejo, H. (2017). *Métodos para la mejora de predicciones en clases desbalanceadas en el estudio de bajas clientes (CHURN)*. Universidad de Santiago de Compostela.

Bagnato, J. (2020). Conjuntos de Entrenamiento, Prueba y Validación. Retrieved from Aprende Machine Learning website: https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/

Branco, P., Torgo, L., & Ribeiro, R. (2015). A Survey of Predictive Modelling under Imbalanced Distributions. *ArXiv*, 1–48. Retrieved from http://arxiv.org/abs/1505.01658

Brownlee, J. (2017). Long Short-Term Memory Networks With Python Develop Sequence Prediction Models With Deep Learning. In *Machine Learning Mastery* (v1.0).

Brownlee, J. (2018). *Deep Learning for Time Series Forecasting. Predict the Future with MLPs, CNNs and LSTMs in Python* (v1.6).

Brownlee, J. (2020a). *Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*. Machine Learning Mastery.

Brownlee, J. (2020b). *Introduction to Time Series Forecasting with Python. How to Prepare Data and Develop Models to Predict the Future* (v1.9). Machine Learning Mastery.

Chen, L., Xu, G., Zhang, Q., & Zhang, X. (2019). Learning deep representation of imbalanced SCADA data for fault detection of wind turbines. *Measurement: Journal of the International Measurement Confederation*, *139*, 370–379. https://doi.org/10.1016/j.measurement.2019.03.029

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen's Kappa and Brier Score in Binary Classification Assessment. *IEEE Access*, *9*(Mcc), 78368–78381. https://doi.org/10.1109/ACCESS.2021.3084050

Cuevas, E., Avalos, O., Emanuel, P., Valdivia, A., & Pérez, M. (2021). *Introducción al Machine Learning con MATLAB* (Primera Ed). Retrieved from https://books.google.es/books?id=Ek1OEAAAQBAJ

Espinar, R. (2018). *Modelos de Clasificación con datos no balanceados* (Universidad de Sevilla). Retrieved from https://idus.us.es/bitstream/handle/11441/77518/Espinar Lara Rocío TFG.pdf?sequence=1&isAllowed=y

Fuchs, M. (2020). Dealing with Imbalanced classes. Retrieved from https://michael-fuchs-python.netlify.app/2020/01/16/dealing-with-imbalanced-classes/

Garcia, J. G., Lopez, N. C., & Calvo, J. Z. (2011). *Estadistica Básica para estudiantes de Ciencias*. Madrid.

García, S. (2003). *Organización y gestión integral de mantenimiento* (Primera Ed). Madrid.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* (Second Edi; R. Roumeliotis & N. Tache, Eds.). Sebastopol: O'Reilly Media, Inc.

Gonzáles, A. (n.d.). Machine Learning: predicciones basadas en datos con BigML. Retrieved from cleverdata website: https://cleverdata.io/machine-learning-prediccion-basada-datos-bigml/

Hasegawa, T., Saeki, M., Ogawa, T., & Nakano, T. (2019). Vibration-based fault detection for flywheel condition monitoring. *Procedia Structural Integrity*, *17*, 487–494. https://doi.org/10.1016/j.prostr.2019.08.064

Huertas, A. (2020). *Algoritmos de aprendizaje supervisado utilizando datos de monitoreo de condiciones : Un estudio para el pronóstico de fallas en máquinas*. Universidad Santo Tomás.

Interactive Chaos. (n.d.). Arrays de tres dimensiones. Retrieved from interactivechaos.com

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem A systematic study fulltext.pdf. *Intelligent Data Analysis*, *6*, 429–449.

Jiménez, C. A. (2016). *Análisis de fallos de parques eólicos*. Universidad de Sevilla.

Kavaz, A. G., & Barutcu, B. (2018). Fault detection of wind turbine sensors using artificial neural networks. *Journal of Sensors*, *2018*, 11. https://doi.org/10.1155/2018/5628429

Kim, P. (2017). MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence. In *MATLAB Deep Learning*. https://doi.org/10.1007/978-1-4842-2845-6

Kraiem, M. (2020). *Resumen de tesis. Clasificación a partir de conjuntos de datos no equilibrados. Un marco para mejorar la aplicación de las estrategias de remuestreo* (Universidad de Salamanca). Retrieved from https://gredos.usal.es/handle/10366/145514%0Ahttps://gredos.usal.es/bitstream/handle/10366/145514/Kraiem%2C Mohamed %28v.r%29.pdf?sequence=1&isAllowed=y

Lemnaru, C., & Potolea, R. (2012). Imbalanced classification problems: Systematic study, issues and best practices. *Lecture Notes in Business Information Processing*, *102 LNBIP*(1), 35–50. https://doi.org/10.1007/978-3-642-29958-2_3

Loría, A. L., Villalobos, E. M., & Piedra, C. (2017). Modelo de toma de decisiones de mantenimiento basado en la predicción de vida útil para componentes de sistemas eólicos en Costa Rica. *Revista Tecnología En Marcha*, *30*(3), 129.

https://doi.org/10.18845/tm.v30i3.3279

Maldonado, J., Martín, S., Artigao, E., & Gómez, E. (2020). Using SCADA data for wind turbine condition monitoring: A systematic literature review. *Energies*, *13*(12). https://doi.org/10.3390/en13123132

Martínez, E., Jiménez, E., Blanco, J., & Sáenz, J. C. (2014). Predicción y detección de averías en aerogeneradores a partir de datos Scada. *Dyna (Spain)*, *89*(5), 10. https://doi.org/10.6036/7063

Martínez, L. J. (2017). *Monitorización de estado de los componentes eléctricos de aerogeneradores mediante el análisis de la temperatura del SCADA* (Universidad de Castilla-La Mancha). Retrieved from https://www.uv.es/qfinan/Tesis_Doctorales/PDF/tes_alvaro montealegre.pdf

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

Monge, L., & Talayero, A. (2011). Aerogeneradores. In A. Talayero & E. Telmo (Eds.), *Energía Eólica* (2a. Edició, pp. 69–118). Zaragoza: Prensas Universitarias de Zaragoza.

Morrís, M. (2020). Open Forecasting Engine. Retrieved from Amazon Web Service website: https://aws.amazon.com/es/blogs/aws-spanish/open-forecast-engine/

Navidi, W. (2006). *Estadística para ingenieros y científicos* (Primera Ed). Retrieved from http://repositorio.unan.edu.ni/2986/1/5624.pdf

Patterson, J., & Gibson, A. (2017). *Deep Learning. A practitioner's approach* (First Ed.; M. L. and T. McGovern, Ed.). O'Reilly Media, Inc.

Pernia, Y. D., & Hernández, Y. M. (2014). Estrategias de Operación y Mantenimiento de una Turbina Eólica. *ResearchGate*, (October 2014), 6.

Pinto, R., & Cerquitelli, T. (2019). Robot fault detection and remaining life estimation for predictive maintenance. *Procedia Computer Science*, *151*(2018), 709–716. https://doi.org/10.1016/j.procs.2019.04.094

Point, T. (2018). *TensorFlow*. https://doi.org/10.1007/978-3-322-94873-1

Politi, P. P. (2022). *Wind Turbine Condition Monitoring Through Artificial Neural Networks Using SCADA Data*. Politecnico di Torino.

Ponce, J. C., Torres, A., Quezada, F. S., Silva, A., & Martínez, E. U. (2014). *Inteligencia Artificial* (Primera Ed). Proyecto LATIn.

Ponce, P. (2010). *Inteligencia Artificial: con aplicaciones a la ingeniería* (Primera Ed; A. Herrera, Ed.). México: Alfaomega Grupo Editor, S.A. de C.V.

Poveda, G., Ruiz, K., & González, J. (2017). Desarrollo de Energías Renovables en el Ecuador del siglo XXI, Optimización de Recursos Económicos y Conservación del Medio

AMbiente. *Revista Observatorio de La Economía Latinoamericana*, 1–16. Retrieved from http://www.eumed.net/cursecon/ecolat/ec/2017/energias-renovables-ecuador.html

Pueyo, C., Telmo, E., & Pérez, J. (2011). Recurso Eólico. In A. Talayero & E. Telmo (Eds.), *Energía Eólica* (2a. Edició, pp. 29–67). Zaragoza: Prensas Universitarias de Zaragoza.

Rodríguez, F. (2017). *Smote-d, una versión determinista de smote*. Instituto Nacional de Astrofísica, Óptica y Electrónica.

Schallenberg, J. C., Gonzalo, R., Izquierdo, P., Hernández Rodríguez, C., Unamunzaga, P., Ramón, F., … Ortin, S. (2008). *Energías renovables y eficiencia energética* (Primera Ed). Canarias.

Scikit-learn. (n.d.). Validación cruzada: evaluando el rendimiento del estimador. Retrieved from https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Seaborn. (n.d.). Visualización de distribuciones de datos. Retrieved from seaborn.pydata.org/tutorial/distributions.html#tutorial-kde

Spiegel, M., & Stephens, L. (2009). *Estadística* (Cuarta Edi; A. Delgado, Ed.). México: Mc Graw Hill Educación.

Verma, A., & Kusiak, A. (2012). Fault monitoring of wind turbine generator brushes: A data-mining approach. *Journal of Solar Energy Engineering, Transactions of the ASME*, *134*(2), 1–9. https://doi.org/10.1115/1.4005624

Villarubia, M. Á. (2013). *Ingeniería de la energía eólica* (Primera Ed). México.

Vinuesa, P. (2016). *Tema 8 - Correlación : teoría y práctica*. 1–26. Retrieved from http://www.ccg.unam.mx/~vinuesa/

11. Anexos

Anexo 1. Código para el remuestreo de datos (Implementado en Spyder)

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 20 18:00:39 2021
@author: Lenovo
"""
# PREPARACIÓN DE DATOS
# Import libraries
import pandas as pd
import numpy as np
from collections import Counter
from matplotlib import pyplot
from numpy import where
# Import datasets
data_train = pd.read_csv('feat_train.csv')
label_train = pd.read_csv('targ_train.csv')
data_test = pd.read_csv('feat_test.csv')
label_test = pd.read_csv('targ_test.csv')
# Quitar las columnas que no se usan
data_train = data_train.drop(['Unnamed: 0', 'Time'], axis=1)
label_train = label_train.drop(['Unnamed: 0','funcionamiento','tipo_alarma',
'prioridad_alarma'], axis=1)
data_test = data_test.drop(['Time','Unnamed: 0'], axis=1)
label_test = label_test.drop(['Unnamed: 0','funcionamiento','tipo_alarma',
'prioridad_alarma'], axis=1)
# Preparar los datos para graficar
X = data_train[['V_viento','P_estator']].to_numpy()
y = label_train['alarma'].to_numpy()
# Summarize class distribution
counter = Counter(y)
print(counter)
# scatter plot of examples by class label
for label, _ in counter.items():
row_ix = where(y == label)[0]
pyplot.scatter(X[row_ix, 0], X[row_ix, 1], label=str(label))
pyplot.legend()
pyplot.show()
#------------------------------
# --------RANDOM UNDERSAMPLING--------------
from imblearn.under_sampling import RandomUnderSampler
# sampling_strategy='majority'
# define undersample strategy
undersample = RandomUnderSampler(sampling_strategy='majority')
# fit and apply the transform
X_under, y_under = undersample.fit_resample(data_train, label_train)
# summarize class distribution
counter_under = Counter(y_under)
print(counter_under)
# Preparar los nuevos datos para graficar
X2 = X_under[['V_viento','P_estator']].to_numpy()
y2 = y_under['alarma'].to_numpy()
# summarize the new class distribution
counter2 = Counter(y2)
print(counter2)
# scatter plot of examples by class label
for label, _ in counter2.items():
row_ix2 = where(y2 == label)[0]
```

```
pyplot.scatter(X2[row_ix2, 0], X2[row_ix2, 1], label=str(label))
pyplot.legend()
pyplot.show()
# example of evaluating a decision tree with random undersampling
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
# define pipeline
steps = [('under', RandomUnderSampler()), ('model', DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores1 = cross_val_score(pipeline, data_train, label_train, scoring='roc_auc',
cv=cv, n_jobs=-1
scores2 = cross_val_score(pipeline, data_train, label_train, scoring='recall',
cv=cv, n_jobs=-1)
scores3 = cross_val_score(pipeline, data_train, label_train, scoring='f1', cv=cv,
n_jobs=-1)
print('Mean ROC AUC: %.3f' % mean(scores1))
print('Mean Recall: %.3f' % mean(scores2))
print('Mean F-1: %.3f' % mean(scores3))

# sampling_strategy=0.5
# define undersample strategy
undersample5 = RandomUnderSampler(sampling_strategy=0.5)
# fit and apply the transform
X_under5, y_under5 = undersample5.fit_resample(data_train, label_train)
# summarize class distribution
counter_under5 = Counter(y_under5)
print(counter_under5)
# Preparar los nuevos datos para graficar
X3 = X_under5[['V_viento','P_estator']].to_numpy()
y3 = y_under5['alarma'].to_numpy()
## summarize the new class distribution
counter3 = Counter(y3)
print(counter2)
# scatter plot of examples by class label
for label, _ in counter3.items():
row_ix3 = where(y3 == label)[0]
pyplot.scatter(X3[row_ix3, 0], X3[row_ix3, 1], label=str(label))
pyplot.legend()
pyplot.show()
# example of evaluating a decision tree with random undersampling
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
# define pipeline
steps = [('under', RandomUnderSampler(sampling_strategy=0.5)), ('model',
DecisionTreeClassifier(
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
```

```
scores4 = cross_val_score(pipeline, data_train, label_train, scoring='roc_auc',
cv=cv, n_jobs=-1
scores5 = cross_val_score(pipeline, data_train, label_train, scoring='recall',
cv=cv, n_jobs=-1)
scores6 = cross_val_score(pipeline, data_train, label_train, scoring='f1', cv=cv,
n_jobs=-1)
print('Mean ROC AUC: %.3f' % mean(scores4))
print('Mean Recall: %.3f' % mean(scores5))
print('Mean F-1: %.3f' % mean(scores6))
# ============================================================================
# GUARDAR LOS DATASET EN LA RUTA DE TRABAJO
# ruta1 =
('C:/Users/USUARIO/alex/imbalacedClassification/random_undersampling/X_under.csv')
# ruta2 =
('C:/Users/USUARIO/alex/imbalacedClassification/random_undersampling/y_under.csv')
# ruta3 =
('C:/Users/USUARIO/alex/imbalacedClassification/random_undersampling/X_under5.csv'
)
# ruta4 =
('C:/Users/USUARIO/alex/imbalacedClassification/random_undersampling/y_under5.csv'
)
#
#
# X_under = X_under.to_csv(ruta1)
# y_under = y_under.to_csv(ruta2)
# X_under5 = X_under5.to_csv(ruta3)
# y_under5 = y_under5.to_csv(ruta4)
# ============================================================================
```

Anexo 2. Importación de Librerías y Preparación de datos (Implementado en Jupyter Notebook).

Cargar librerias

```
# librerias
import numpy as np
import math
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('classic')
%matplotlib inline
import seaborn as sns
#import pandas_profiling

from numpy.random import seed
seed(0)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, accuracy_score
from plot_metric.functions import BinaryClassification
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from yellowbrick.classifier import ConfusionMatrix
from sklearn.metrics import classification_report

from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Funciones

```
# BINARIA funcion para generar matriz de confusion y medidas de desempeño del modelos clasificacion binarios
def model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
    print('\n')
    print("MATRIZ DE CONFUSION")
    print('\n')
    matriz = confusion_matrix(label_test, label_pred)
```

```
41      #print(matriz)
42      classes = ["Funcional","Falla potencial"]
43      cm = ConfusionMatrix(modelo, classes=classes, label_encoder={0:"Funcional", 1:"Falla Potencial
"})
44      cm.fit(data_train, label_train)
45      cm.score(data_test, label_test)
46      cm.show()
47      # Medidas de desempeño del modelo
48      print('\n')
49      print("MEDIDAS DE DESEMPEÑO DEL MODELO")
50      print('\n')
51      print('Exactitud - Accuracy: {}'.format(modelo.score(data_test, label_test)))
52      print('ROC AUC: {}'.format (metrics.roc_auc_score(label_test, label_pred)))
53      print('-'*60)
54      print('Precisión por clase: {}'.format(metrics.precision_score(label_test, label_pred, average=None
)))
55      print('Sensibilidad - Recall por clase: {}'.format(metrics.recall_score(label_test, label_pred, averag
e=None)))
56      print('F1 por clase: {}'.format(metrics.f1_score(label_test, label_pred, average=None)))
57      print('-'*60)
58      print('F1 Macro: {}'.format(metrics.f1_score(label_test, label_pred, average='macro')))
59      print('F1 Micro: {}'.format(metrics.f1_score(label_test, label_pred, average='micro')))
60      print('F1 Weighted: {}'.format(metrics.f1_score(label_test, label_pred, average='weighted')))
61      print('-'*60)
62      print('Sensibilidad - Recall Macro: {}'.format(metrics.recall_score(label_test, label_pred, average=
'macro')))
63      print('Sensibilidad - Recall Micro: {}'.format(metrics.recall_score(label_test, label_pred, average='
micro')))
64      print('Sensibilidad - Recall Weighted: {}'.format(metrics.recall_score(label_test, label_pred, avera
ge='weighted')))
65      print('-'*60)
66      print('Precisión Macro: {}'.format(metrics.precision_score(label_test, label_pred, average='macro')
))
67      print('Precisión Micro: {}'.format(metrics.precision_score(label_test, label_pred, average='micro'))
)
68      print('Precisión Weighted: {}'.format(metrics.precision_score(label_test, label_pred, average='wei
ghted')))
69
70      TP = matriz[1][1]
71      TN = matriz[0][0]
72      FP = matriz[0][1]
73      FN = matriz[1][0]
74      print('-'*60)
75      print('Verdaderos positivos - True Positives:', TP)
76      print('Verdaderos negativos - True Negatives:', TN)
77      print('Falsos positivos - False Positives:', FP)
                                                                                                    44
78      print('Falsos negativos - False Negatives:', FN)
```

```
    # calculate accuracy
    conf_accuracy = (float (TP+TN) / float(TP + TN + FP + FN))
    # calculate mis-classification
    conf_misclassification = 1- conf_accuracy
    # calculate the sensitivity
    conf_sensitivity = (TP / float(TP + FN))
    # calculate the specificity
    conf_specificity = (TN / float(TN + FN))
    # calculate precision
    conf_precision = (TP / float(TP + FP))
    # calculate f_1 score
    conf_f1 = 2 * ((conf_precision * conf_sensitivity) / (conf_precision + conf_sensitivity))
    # calculate FPR tasa de falsos positivos
    conf_FPR = 1 - conf_specificity
    print('-'*60)
    print(f'Error de clasificacion: {round(conf_misclassification,3)}')
    print(f'Especificidad - Specificity: {round(conf_specificity,3)}')
    print(f'Tasa de falsos positivos FPR: {round(conf_FPR,3)}')
    print('-'*60)

    print(classification_report(label_test, label_pred))
    # curva ROC clasificacion binaria
    bc = BinaryClassification(label_test, label_pred, labels=["Class 0", "Class 1"])
    plt.figure(figsize=(5,5))
    bc.plot_roc_curve()
    print('\n')
    print("CURVA ROC")
    plt.show()
```

Preparar datos

```
# import dataset
datos = pd.read_csv('gen4.csv')
#check the shape of dataset
datos.shape
#create a visual plot to see the target distribution

#sns.countplot(x="funcionamiento", data=datos);
sns.countplot(x="alarma", data=datos);
#sns.countplot(x="tipo_alarma", data=datos);
#sns.countplot(x="prioridad_alarma", data=datos);
#check the target class distribution
datos['alarma'].value_counts()
# drop the ID variable as it is unique for all the transactions and does not have any meaningful information about the data
data = datos.drop(['Dia','Hora','funcionamiento', 'tipo_alarma', 'prioridad_alarma'], axis=1)
# extract features and target from the original dataset
```

```
features = data.drop(['alarma'], axis=1)
target = data['alarma']
#split the dataset into training and test set to train and evaluate the model respectively
from sklearn.model_selection import train_test_split
data_train, data_test, label_train, label_test = train_test_split(features, target, test_size=0.3, random_state=42)
```

Conteo de observaciones por clase

```
label_test.value_counts(sort=False)
label_train.value_counts(sort=False)
sns.countplot(label_test, palette=["#3498db", "#e74c3c"])
plt.title('Funcional (0)                    Falla potencial (1)')
plt.xlabel("Clases Base de prueba")
plt.ylabel("Numero de observaciones")
sns.countplot(label_train, palette=["#3498db", "#e74c3c"])
plt.title('Funcional (0)                    Falla potencial (1)')
plt.xlabel("Clases Base de entrenamiento")
plt.ylabel("Numero de observaciones")

#create some scatterplots based on features and can see some pattern in the data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
sns.scatterplot(x='velocidadViento', y='potenciaEstator', data=data, hue='alarma', style='alarma', palette="deep", ax=ax1);
sns.scatterplot(x='velocidadViento', y='potenciaRotor', data=data, hue='alarma', style='alarma', palette="deep", ax=ax2);

#create some scatterplots based on features and can see some pattern in the data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
sns.scatterplot(x='velocidadViento', y='potenciaEstator', data=data_train, hue=label_train, style=label_train, palette="deep", ax=ax1);
sns.scatterplot(x='velocidadViento', y='potenciaRotor', data=data_train, hue=label_train, style=label_train, palette="deep", ax=ax2);

#create some scatterplots based on features and can see some pattern in the data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
sns.scatterplot(x='velocidadViento', y='potenciaEstator', data=data_test, hue=label_test, style=label_test, palette="deep", ax=ax1);
sns.scatterplot(x='velocidadViento', y='potenciaRotor', data=data_test, hue=label_test, style=label_test, palette="deep", ax=ax2);
```

Anexo 3. Clasificación binaria con datos en bruto Implementado en Jupyter Notebook).

Modelos de ML clasificación binaria para evaluar la métrica exactitud

Modelo Random Forest

```
modelo1=RandomForestClassifier(random_state=0, n_estimators=100)
modelo1.fit(data_train, label_train)
label_pred1 = modelo1.predict(data_test)
# para evaluar cuales variables tiene mayor peso en el modelo
fi=modelo1.feature_importances_
fi
```

Evaluación del modelo

```
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo1, label_pred1)
# Graficar las predicciones
#set(label_test) # Clases distintas
label_test
colors1 = {0: "blue", 1:"red"}
label_test = label_test.map(colors1)
label_test
set(label_pred1)
label_pred1
colors2 = {0: "blue", 1:"red"}
label_pred1 = label_pred1.map(colors)
label_pred1
fig, ax = plt.subplots()
for alarma in set(datos.alarma):
    ax.scatter(
        datos.velocidadViento[datos.alarma == alarma],
        datos.potenciaEstator[datos.alarma == alarma],
        s = 30,
        c = colors[alarma],
        label = alarma)
plt.legend()
plt.show()
#create some scatterplots based on features and can see some pattern in the data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
sns.scatterplot(x='velocidadViento', y='potenciaEstator', data=data_test, hue=label_pred1, style=label_pred1, pal
ette="deep", ax=ax1);
sns.scatterplot(x='velocidadViento', y='potenciaRotor', data=data_test, hue=label_pred1, style=label_pred1, palet
te="deep", ax=ax2);
#create some scatterplots based on features and can see some pattern in the data
fig, ax = plt.subplots(1, 2, figsize=(16, 8))
sns.scatterplot(x='label_pred1', y='label_test', ax=ax1);
#sns.scatterplot(x='velocidadViento', y='potenciaRotor', data=data_test, hue=label_pred1, style=label_pred1, p
alette="deep", ax=ax2);
```

Clasificador Gausiano Naive Bayes

```
modelo2 = GaussianNB()
modelo2.fit(data_train, label_train);
label_pred2 = modelo2.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo2, label_pred2)
```

Clasificacion con Support Vector Machine

```
modelo3 = SVC()
modelo3.fit(data_train, label_train)
label_pred3 = modelo3.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo3, label_pred3)
```

k-NN KNeighborsClassifier

```
modelo5= KNeighborsClassifier()
modelo5.fit(data_train, label_train)
label_pred5 = modelo5.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo5, label_pred5)
```

Clasificacion con Decision tree

```
modelo6= tree.DecisionTreeClassifier()
modelo6.fit(data_train, label_train)
label_pred6 = modelo6.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo6, label_pred6)
```

Clasificacion por Logistic Regression

```
modelo7= LogisticRegression()
modelo7.fit(data_train, label_train)
label_pred7 = modelo7.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo7, label_pred7)
```

Bagging Classifier

```
modelo8= BaggingClassifier()
modelo8.fit(data_train, label_train)
label_pred8 = modelo8.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo8, label_pred8)
```

Extra Trees Classifier

```
modelo9=ExtraTreesClassifier(n_estimators=100)
modelo9.fit(data_train, label_train)
label_pred9 = modelo9.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo9, label_pred9)
```

Gradient Boosting Classifier

```
modelo10=GradientBoostingClassifier(n_estimators=100)
modelo10.fit(data_train, label_train)
label_pred10 = modelo10.predict(data_test)
```

```
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo10, label_pred10)
```

Clasificacion con red neuronal MLP

```
modelo11= MLPClassifier(random_state=0)
modelo11.fit(data_train, label_train);
label_pred11 = modelo11.predict(data_test)
# model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred):
model_evaluation(data_train, label_train, data_test, label_test, modelo11, label_pred8)
```

Red neronal MLP con grid search CV

```
model = MLPClassifier(random_state=0)
parameters= {'learning_rate':["constant", "invscaling", "adaptive"],
          'hidden_layer_sizes':[(8),(2,4),(1,8,1),(1,1,1),(2,8,2),(4,16,4), (1,2,1)],
          'activation': ["identity","logistic", "relu", "tanh"],
          'solver': ["lbfgs", "sgd", "adam"]
}
acc_scorer=make_scorer(accuracy_score)
grid = GridSearchCV(model, param_grid=parameters, cv=2, verbose=2, scoring=acc_scorer)
%time grid.fit(data_train, label_train)
#model = MLPClassifier(random_state=0)
#parameters= {'learning_rate':["constant", "invscaling", "adaptive"],
#          'hidden_layer_sizes':[(7),(7,14),(14,1)],
#          'activation': ["identity","logistic", "relu", "tanh"],
#          'solver': ["lbfgs", "sgd", "adam"]
#}
#acc_scorer=make_scorer(accuracy_score)
#grid = GridSearchCV(model, param_grid=parameters, cv=2, verbose=2, scoring=acc_scorer)
#%time grid.fit(data_train, label_train)
modelo12= grid.best_estimator_
modelo12.fit(data_train, label_train)
modelo12= MLPClassifier(activation='identity', alpha=0.0001, batch_size='auto',
          beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
          hidden_layer_sizes=(7,14), learning_rate='invscaling',
          learning_rate_init=0.001, max_iter=200, momentum=0.9,
          n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
          random_state=0, shuffle=True, solver='sgd', tol=0.0001,
          validation_fraction=0.1, verbose=False, warm_start=False)
#modelo12 = MLPClassifier(hidden_layer_sizes=(7, 14), random_state=0)
modelo12.fit(data_train, label_train)
label_pred12 = modelo12.predict(data_test)
# métricas de desempeño - evaluación del modelo
model_evaluation(data_train, label_train, data_test, label_test, modelo12, label_pred12)
```

Anexo 4. Clasificación binaria de datos con el modelo Extra Trees (Implementado en Jupyter Notebook).

Extra Trees Classifier hiperparametros por defecto

```
modelo9=ExtraTreesClassifier(n_estimators=100, random_state=0)
modelo9.fit(data_train, label_train)
label_pred9 = modelo9.predict(data_test)
# métricas de desempeño - evaluación del modelo
model_evaluation(data_train, label_train, data_test, label_test, modelo9, label_pred9)
```

Stratified k-fold cross-validation

Evaluaremos el modelo utilizando validación cruzada estratificada repetida de k veces, con 10 repeticiones y 20 pliegues. Informaremos la desviación media y estándar de la precisión del modelo en todas las repeticiones y pliegues

```
# evaluar extra trees algorithm para la clasifiacion binaria
model = ExtraTreesClassifier()
cv = RepeatedStratifiedKFold(n_splits=20, n_repeats=10, random_state=1)
n_scores = cross_val_score(model, data_train, label_train, scoring='accuracy', cv=cv, n_jobs=-1, error_sco
re='raise')
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

El resultado es una exactitud media de 0.976 con una desviacion estandar de 0.004

Anexo 5. Implementación del algoritmo de la Red Neuronal CNN

Preparar datos

```
train = pd.read_csv("data_train.csv")
test = pd.read_csv("data_test.csv")
train.drop(['Unnamed: 0','Time','funcionamiento','tipo_alarma','prioridad_alarma'], axis=1, inplace=True)
test.drop(['Unnamed: 0','Time','funcionamiento','tipo_alarma','prioridad_alarma'], axis=1, inplace=True)
train.head()
test.head()
variables=['v_rotor','P_estator','P_rotor','T_amb','T_estator','T_anillos_gen','V_viento']
objetivo='alarma'
# normalizado de datos sin incluir id, ciclo, ttf y clase
scaler = MinMaxScaler()
train[variables]=scaler.fit_transform(train[variables])
test[variables]=scaler.transform(test[variables])
train.head()
test.head()
print("numero de dimensiones =", train.ndim)
print("tamaño de la tabla =", train.shape)
print("total datos =", train.size)
print("numero de dimensiones =", test.ndim)
print("tamaño de la tabla =", test.shape)
print("total datos =", test.size)
# funcion para adaptar datos en 3d para CNN
def gen_sequence(id_df, seq_length, seq_cols):
    df_zeros=pd.DataFrame(np.zeros((seq_length-1,id_df.shape[1])),columns=id_df.columns)
    id_df=df_zeros.append(id_df,ignore_index=True)
    data_array = id_df[seq_cols].values
    num_elements = data_array.shape[0]
    lstm_array=[]
    for start, stop in zip(range(0, num_elements-seq_length), range(seq_length, num_elements)):
        lstm_array.append(data_array[start:stop, :])
    return np.array(lstm_array)
# funcion para generar etiquetas
def gen_label(id_df, seq_length, seq_cols,label):
    df_zeros=pd.DataFrame(np.zeros((seq_length-1,id_df.shape[1])),columns=id_df.columns)
    id_df=df_zeros.append(id_df,ignore_index=True)
    data_array = id_df[seq_cols].values
    num_elements = data_array.shape[0]
    y_label=[]
    for start, stop in zip(range(0, num_elements-seq_length), range(seq_length, num_elements)):
        y_label.append(id_df[label][stop])
    return np.array(y_label)
# definir pasos de tiempo
pasos_de_tiempo=100
columnas_secuencia=variables
# generar data_train
```

```
data_train=np.concatenate(list(list(gen_sequence(train[train['dia']==id], pasos_de_tiempo, columnas_secuencia)) for id in train['dia'].unique()))
# generar label_train
label_train=np.concatenate(list(list(gen_label(train[train['dia']==id], pasos_de_tiempo, columnas_secuencia,'alarma')) for id in train['dia'].unique()))
# generar data_test
data_test=np.concatenate(list(list(gen_sequence(test[test['dia']==id], pasos_de_tiempo, columnas_secuencia)) for id in test['dia'].unique()))
# generar label_test
label_test=np.concatenate(list(list(gen_label(test[test['dia']==id], pasos_de_tiempo, columnas_secuencia,'alarma')) for id in test['dia'].unique()))
print("numero de dimensiones Data_train =", data_train.ndim)
print("tamaño de la tabla Data_train =", data_train.shape)
print("total datos Data_train =", data_train.size)
print('-'*60)
print("numero de dimensiones Label_train =", label_train.ndim)
print("tamaño de la tabla Label_train =", label_train.shape)
print("total datos Label_train =", label_train.size)
print('-'*60)
print("numero de dimensiones Data_test =", data_test.ndim)
print("tamaño de la tabla Data_test =", data_test.shape)
print("total datos Data_test =", data_test.size)
print('-'*60)
print("numero de dimensiones Label_test =", label_test.ndim)
print("tamaño de la tabla Label_test =", label_test.shape)
print("total datos Label_test =", label_test.size)
```

Modelo CNN

```
model_path_clf = 'classification_model.h5'
verbose, epochs, batch_size = 0, 20, 32
n_features =data_train.shape[2]
n_timesteps =data_train.shape[1]
n_outputs= 1
model = Sequential()
model.add(Conv1D(64, 32, activation='relu', input_shape=(n_timesteps,n_features)))
model.add(Conv1D(64, 32, activation='relu'))
model.add(Conv1D(64, 32, activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(n_outputs, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
# entrenar la red
history = model.fit(data_train, label_train, epochs=epochs, batch_size=batch_size, validation_split=0.1, verbose=2,
```

```
        callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=15, verbose=2, mod
e='min'),
                     keras.callbacks.ModelCheckpoint(model_path_clf, monitor='val_loss', save_best_only=True, mod
e='min', verbose=2)])
print(history.history.keys())
# graficar curvas de aprendizaje
def history_plot(hist1, hist2, title, ylabl, xlabl):
    fig_acc = plt.figure(figsize=(12,4))
    plt.plot(history.history[hist1])
    plt.plot(history.history[hist2])
    plt.title(title)
    plt.ylabel(ylabl)
    plt.xlabel(xlabl)
    plt.legend(['train', 'test'], loc='upper left')
    return plt
history_plot('accuracy', 'val_accuracy', 'model accuracy', 'accuracy', 'epoch')
history_plot('loss', 'val_loss', 'model loss', 'loss', 'epoch')
# evaluar el modelo en train y test
_, train_acc = model.evaluate(data_train, label_train, batch_size=32, verbose=1)
_, test_acc = model.evaluate(data_test, label_test, batch_size=32, verbose=1)
print('Train: %.3f, Test: %.3f' % (train_acc, test_acc))
# prediccion de etiquetas de clase de los datos de prueba
label_pred=(model.predict(data_test)).astype("int32")
# Metricas de desempeño del modelo
model_evaluation(data_train, label_train, data_test, label_test, model, label_pred)
```

Anexo 6. Ajuste avanzado del modelo Extra Trees.

Ajuste de hiperparámetros

Explorar el numero de arboles

```
# funcion para evaluar lista de modelos
def get_models():
    models = dict()
    models['10'] = ExtraTreesClassifier(n_estimators=10)
    models['50'] = ExtraTreesClassifier(n_estimators=50)
    models['100'] = ExtraTreesClassifier(n_estimators=100)
    models['500'] = ExtraTreesClassifier(n_estimators=500)
    models['1000'] = ExtraTreesClassifier(n_estimators=1000)
    return models
# funcion para evaluar modelo usando CV
def evaluate_model(model):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, data_train, label_train, scoring='accuracy', cv=cv, n_jobs=-1, error_score='ra
ise')
    return scores
# modelos a evaluar
models = get_models()
# evaluar modelos y guardar resultados
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# graficar el desempeño del modelo para comparacion
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

Ajustar el Number of Features (max_features)

```
# lista de modelos a evaluar max_features entre 2 y 20
def get_models():
    models = dict()
    for i in range(1, 8):
        models[str(i)] = ExtraTreesClassifier(n_estimators=500, max_features=i)
    return models
# funcion para evaluar modelo usando CV
def evaluate_model(model):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, data_train, label_train, scoring='accuracy', cv=cv, n_jobs=-1, error_score='ra
ise')
    return scores
# modelos a evaluar
models = get_models()
# evaluar modelos y guardar resultados
```

```
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# graficar el desempeño del modelo para comparacion
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

Ajuste de Minimum Samples per Split

```
# evaluar entre 2 (default) y 14 muestras
def get_models():
    models = dict()
    for i in range(2, 15):
        models[str(i)] = ExtraTreesClassifier(min_samples_split=i, n_estimators=500, max_features=4)
    return models
# funcion para evaluar modelo usando CV
def evaluate_model(model):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, data_train, label_train, scoring='accuracy', cv=cv, n_jobs=-1, error_score='ra
ise')
    return scores
models = get_models()
# evaluar modelos y guardar resultados
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# graficar el desempeño del modelo para comparacion
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

Entrenar modelo con hiperparametros ajustados

```
modelo=ExtraTreesClassifier(min_samples_split=6, n_estimators=500, max_features=4, random_state=0)
modelo.fit(data_train, label_train)
label_pred = modelo.predict(data_test)
# métricas de desempeño - evaluación del modelo
model_evaluation(data_train, label_train, data_test, label_test, modelo, label_pred)
```

Anexo 7. Certificación de traducción de resumen.

CERTIFICADO DE TRADUCCIÓN DE RESUMEN

Ing. Pio Oswaldo Palacios Jimenez con certificación C2 en el idioma inglés.

CERTIFICA:

Que la traducción al idioma inglés del resumen del trabajo de titulación denominado **"Aplicación de técnicas de Inteligencia Artificial para la predicción de fallas en un aerogenerador de eje horizontal utilizando los datos del Sistema SCADA"**, correspondiente al señor egresado: Jorge Alexander Puchaicela Castillo, con cédula de identidad 1105946972, ha sido revisado y supervisado según se me ha solicitado, por lo cual cumple con la correcta traducción al idioma inglés.

Esto es lo que puedo mencionar y certificar en honor a la verdad para fines pertinentes.

Loja, 01 de agosto de 2023

Firmado electrónicamente por:
PIO OSWALDO PALACIOS JIMENEZ

Ing. Pio Oswaldo Palacios Jimenez

FINE-TUNED ENGLISH LANGUAGE INSTITUTE

Ministerio de Educación. Coordinación Zonal 7. Res. N° 471-14

CERTIFICATE OF ACADEMIC EXCELLENCE

AWARDED TO

PÍO OSWALDO PALACIOS JIMÉNEZ

For having achieved a high level of English Proficiency throughout the Program and showing dedication and commitment to our educational objectives. The Institution wishes you every success in your future.

Given in Loja, Ecuador on the 5' of February, 2016

Lcda. Aguilar de L., M.A. — **ACADEMIC DIRECTOR**

Edgar Abad Villavicencio, B.A. — **ADMINISTRATIVE DIRECTOR**

Liliana Heredia, B.A. — **ACADEMIC ADVISOR**

Jessica Morocho — **SECRETARY**