ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Análisis predictivo de funcionamiento de Sistema Híbrido Off Grid mediante Machine Learning

AUTOR: Laura Hernández Cubo

TITULACIÓN: Electrónica de Comunicaciones

DIRECTOR: Antonio Aragón Álvarez

TUTOR: Carlos Ramos Nespereira

DEPARTAMENTO: Ingeniería Electrónica y Telemática

Miembros del Tribunal Calificador:

PRESIDENTE: Yolanda Blanco Archilla

TUTOR: Carlos Ramos Nespereira

SECRETARIO: Hugo Alexer Parada Gélvez

Fecha de lectura:

Calificación:

El Secretario,

Agradecimientos

Antes de comenzar con el desarrollo del proyecto, quería reservar unas líneas para agradecer a todas las personas que me han acompañado durante todo este camino.

En primer lugar, agradecer a mi director de proyecto Antonio Aragón. Gracias por ser la primera persona en darme la oportunidad, enseñarme y guiarme en primeros pasos de mi carrera profesional. De igual forma, agradecer a mi tutor Carlos Ramos por estar siempre dispuesto a prestarme su ayuda durante la realización de este proyecto y animarme siempre a dar lo mejor. Gracias a los dos.

En segundo lugar, dar las gracias a mi familia. Sin duda, sois la mitad de todo este trabajo. A mis padres y a mi hermano, por dejarme siempre elegir mi camino y apoyarme para seguir adelante. Me habéis enseñado el valor del esfuerzo, la constancia y a no rendirme nunca. Gracias a mis amigos, mi otra familia. Gracias por aguantarme todos estos años de subidas y bajadas. En especial, gracias a Alba, Blanca y Lucía.

Por último, agradecer a los que me han acompañado en primera persona. Compartir con vosotros este tiempo ha sido lo mejor sin duda de esta etapa y me alegro de que vengáis conmigo a las siguientes. Gracias a Gerar, a Guti, a Ángel, a Keka, por compartir conmigo los triunfos, los fracasos, largas charlas para seguir o simplemente para desconectar y sobre todo tantas risas y cariño diario. Y en especial a Nacho. Gracias por tu ayuda infinita y por enseñarme que todo suma, eres la otra mitad de este trabajo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

CV	*Cross Validation*
DL	*Deep Learning*
DT	*Decision Tree*
GE	*Grupo Electrógeno*
GUI	*Graphical User Interface*
IA	*Inteligencia Artificial*
IP	*Internet Protocol*
L2TP	*Layer 2 Tunneling Protocol*
LOOCV	*Leave One Out CV*
ML	*Machine Learning*
NOC	*Network Operation Center*
RF	*Random Forest*
RL	*Reinforcement Learning*
SCADA	*Supervisory Control And Data Acquisition*
SIM	*Subscriber Identity Module*
SOC	*State Of Charge*
VPN	*Virtual Private Network*

Resumen

Con el aumento de dispositivos con capacidad de recopilar, procesar y trasmitir datos, se han popularizado el uso de herramientas de análisis de datos capaces de procesar automáticamente esta información sin perder fiabilidad. Este análisis permite obtener conocimiento de la información subyacente de los datos con el propósito de extraer conclusiones de funcionamiento y rendimiento de un equipo o un sistema formado por un conjunto de ellos.

De la disponibilidad de gran cantidad de datos recogidos por un sistema real nace la necesidad de este proyecto. El objetivo es analizar el funcionamiento de un sistema utilizando técnicas de aprendizaje automático. Estas técnicas se centrarán en algoritmos de aprendizaje supervisado, donde se evaluarán diferentes algoritmos de clasificación que permitan predecir el estado de un equipo del sistema.

Desde la captura de los datos, hasta la obtención de valores de precisión de un algoritmo entrenado, el proyecto abarca las diferentes fases de una metodología de análisis de datos.

Palabras clave: aprendizaje automático, algoritmos de clasificación, precisión.

Abstract

Due to the increase in the number of devices capable of collecting, processing, and transmitting data, the use of data analysis tools has become popular. These tools are capable of automatically processing the information without losing reliability. The analysis of data makes possible to gain of the underlying knowledge in the data in order to draw conclusions about the operation and efficiency of a device or a system composed of them.

The aim of this project comes from the availability of large amount of data collected by a real system. The objective is to analyze the operation of a system using machine learning techniques. These techniques will focus on supervised learning algorithms. Different classification algorithms will be evaluated to predict the state of a device in the system.

From data capture to obtaining accuracy values from a trained algorithm, the project covers the different phases of a data analysis methodology.

Keywords: machine learning, classification algorithms, accuracy.

Capítulo 1

1 INTRODUCCIÓN

Las estaciones base de telefonía móvil son elementos de las redes de comunicaciones móviles con equipamiento para cubrir el área donde se pretende prestar el servicio de cobertura. Con el desarrollo de las nuevas tecnologías de telefonía móvil, el despliegue de estas estaciones ha crecido de forma masiva. Las zonas donde se encuentran las estaciones base en entornos no urbanos suelen ser de difícil acceso, montañas o zonas donde no existe acometida de red eléctrica. Para el suministro eléctrico, de forma tradicional se utilizaban grupos electrógenos 24 horas, pero esta solución ha sido sustituida de forma progresiva por sistemas más eficientes que incluyen energías renovables, además de por su gasto operativo menor, por la reducción de vertidos y emisiones de CO_2 que estos suponen.

Existen numerosas empresas operadores de infraestructura que construyen estas estaciones con inversión propia y ofrecen la infraestructura en alquiler a operadores de telefonía móvil. Además de los servicios de suministro, estas empresas ofrecen servicios de mantenimiento y supervisión. Con el objetivo de cumplir estas prestaciones, es necesario el despliegue de equipos y servicios de monitorización que permitan un control total sobre la estación 24 horas, 365 días al año. Estos equipos se encargan de recoger datos de forma periódica para tomar acciones en tiempo real, pero también como base para la utilización de herramientas de análisis que permitan mejorar sus servicios. Este último punto será el foco de interés en el presente proyecto, desarrollado con el objetivo de analizar el funcionamiento de los sistemas que alimentan a estas infraestructuras a través de cantidad masiva de datos que se recogen, con el propósito de aprovechar sus datos y utilizarlos para identificar patrones de comportamiento de sus equipos.

1.1 Objetivos

El objetivo del presente proyecto es desarrollar un sistema que permita predecir el funcionamiento y controlar las fuentes de generación de energía de una estación base con técnicas de aprendizaje automático (en inglés *Machine Learning* o ML), a través de los datos que almacena un NOC (en inglés *Network Operation*

Center) junto con otras fuentes de información. Para conseguirlo, nos apoyamos en distintos objetivos específicos como pueden ser los siguientes:

- Modelar un algoritmo para la supervisión del estado de funcionamiento de los elementos de suministro de energía, más concretamente del grupo electrógeno.
- Evaluar el rendimiento de los distintos modelos de ML con el objetivo de analizar qué modelo se adapta mejor a la predicción del estado de este tipo de equipos.
- Poder controlar el funcionamiento de los equipos a través de modelos de predicción.

1.2 Estructura del documento

Para describir el desarrollo del proyecto, este informe se ha estructurado en diferentes capítulos.

Los primeros capítulos, tratan de aportar los conceptos necesarios para seguir la lectura de los capítulos posteriores, realizando un recorrido desde el marco general donde se desarrolla el proyecto hasta llegar al específico donde se aplica la solución.

En el tercer capítulo, se establecerán las especificaciones y restricciones de diseño.

En el cuarto capítulo, se explicará la metodología que siguen los proyectos relacionados con análisis de datos, describiendo detalladamente cada apartado que la constituye.

Durante el quinto capítulo, se realizará el desarrollo de la metodología expuesta y se analizarán los resultados obtenidos.

Finalmente, los últimos capítulos recogerán las conclusiones y posibles líneas futuras de trabajo junto con el presupuesto del proyecto.

Capítulo 2

2 MARCO TECNOLÓGICO

2.1 Introducción

Este capítulo permite obtener una perspectiva teórica del proyecto realizado. Para ello, primero se definirán las características del sistema donde se desarrolla el problema. Después, se hará una revisión de las diferentes ramas de tratamiento de datos, explicando de forma breve las diferentes vías que presenta y profundizando en las que se usarán para el desarrollo del proyecto. Por último, se explicarán las herramientas que se utilizarán.

2.2 Sistema Híbrido Off-Grid

En términos de suministro eléctrico, un sistema híbrido es aquel sistema compuesto por más de una fuente de generación de energía eléctrica. Estas fuentes de energía pueden ser energías renovables, como energía solar o eólica, o fuentes de energía convencionales, como el diésel o gas. Según su situación respecto a la red eléctrica pública se distinguen tres tipos de sistemas: sistemas acoplados a la red (en inglés *On-Grid*), sistemas aislados de la red (en inglés *Off-Grid*) y sistemas acoplados a red, pero con suministro inestable y cortes intermitentes, denominados *Bad-Grid*. El sistema que se va a analizar en este proyecto es un sistema híbrido solar, formado por un conjunto de paneles solares, baterías y un grupo electrógeno diésel de seguridad *(*en inglés *back-up)*. La función de este conjunto es alimentar una estación base de red móvil aislada de la red eléctrica. Este tipo de estaciones deben garantizar calidad y seguridad de suministro, ya que algunos nodos y sistemas dependen únicamente de una sola estación. Además, cuentan con un elevado consumo energético, ya que los equipos de banda ancha consumen una gran cantidad de energía, por lo que uno de los objetivos principales de este sistema híbrido es operar en términos de eficiencia energética.

2.3 Inteligencia Artificial, aprendizaje automático, aprendizaje profundo

Desde hace décadas, los seres humanos han tratado de entender cómo funciona la mente humana. Mucho antes de que existiera un campo de conocimiento llamado Inteligencia Artificial, incluso antes de que existieran los ordenadores, ya se pensaba en la posibilidad de crear inteligencia fuera del cuerpo [1].

En 1950, el matemático Alan Turing escribió un artículo titulado "Computing Machines and Intelligence [2]". En él se preguntaba, *¿pueden pensar las máquinas?* Tratando de responder esta pregunta propuso una prueba conocida actualmente como *El test de Turing*. Este ejercicio serviría como criterio de decisión para la validación de la inteligencia en un ordenador. La prueba planteaba una conversación de una persona con dos interlocutores a los que no ve, e intenta adivinar si se trata de un ser humano o un ordenador. De forma sencilla, si un juez humano era incapaz de decidir cuál de dos testigos ocultos era el ordenador y cuál el ser humano, la máquina se habría convertido en inteligente. El debate de Turing, relativo al significado de la prueba de Turing para la viabilidad de la Inteligencia Artificial y a lo que es o no es la inteligencia humana, ha sido uno de los principales ámbitos de investigación filosófica en las últimas décadas [3]. Si bien es cierto que el origen y los conceptos de la IA se le atribuyen a Alan Turing, la definición de "Inteligencia Artificial" se debe a John McCarthy:

> *"La inteligencia artificial es la ciencia y la ingeniería de la fabricación de máquinas inteligentes, especialmente de programas informáticos inteligentes* [4]*"*

En el año 1959, Arthur Samuel definió el aprendizaje automático como:

> *" El campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados explícitamente* [5]*"*

El aprendizaje automático es considerado una rama de la inteligencia artificial, que construye un modelo matemático basado en datos de muestra con el fin de hacer predicciones o tomar decisiones sin estar explícitamente programado para realizar la tarea [6]. Al contrario que los algoritmos tradicionales, donde se elabora un programa en el que introduciendo un dato de entrada se obtiene un dato de salida, en los algoritmos basados en ML, los datos de entrada y los valores

de salida esperados son conocidos y es el algoritmo el que es desconocido. De esta forma, el algoritmo de aprendizaje automático será capaz de determinar, en base a datos de entrada-salida conocidos, el mejor algoritmo que lo resuelva.

Dentro de esta rama, se encuentra el aprendizaje profundo (en inglés *Deep learning*) situado como un subcampo del ML y, por lo tanto, de la IA. El aprendizaje profundo consiste en aprender la lógica interna y el nivel de representación de los datos de la muestra. Este método, está estructurado en múltiples niveles de representación, denominados capas *(*en inglés *layers).* Cada capa realiza una función específica para ir logrando una abstracción jerárquica. Este tipo de algoritmos son altamente utilizados para detección y clasificación de objetos.

En la siguiente figura se muestra la relación de las tres disciplinas expuestas en el apartado.

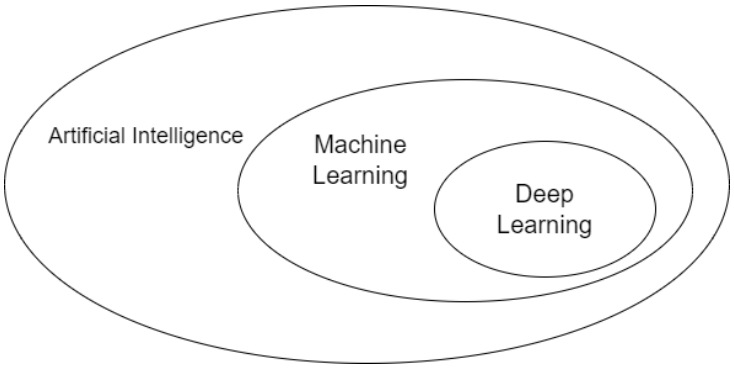

Figura 1. – Artificial Intelligence, Machine Learning and Deep learning

2.4 Clasificación de modelos de machine learning

Los algoritmos de aprendizaje automático se incorporan a las máquinas y a los flujos de datos proporcionados, de modo que, se extraen el conocimiento y la información para ser posteriormente introducidos al sistema, logrando así una gestión más rápida y eficaz de los procesos. El ML puede utilizarse para resolver una amplia variedad de problemas, pero es necesario contar con un canal de entrada de información al sistema para realizar los cálculos y la toma de decisiones adecuada para cada caso. De acuerdo con el problema que se quiere resolver se utiliza un tipo específico de ML. Se distinguen tres tipos de

aprendizaje: Aprendizaje Supervisado, Aprendizaje No Supervisado y Aprendizaje por Refuerzo.

2.4.1 Aprendizaje Supervisado

El aprendizaje automático supervisado implica un atributo de salida predeterminado además del uso de atributos de entrada. Los algoritmos intentan predecir y clasificar el atributo de salida, y sus precisiones y errores de clasificación, junto con otras medidas de rendimiento, dependen de los recuentos del atributo predicho o clasificado correctamente o no [7]. Los algoritmos requieren la especificación de ajustes máximos para el resultado deseado. El proceso de aprendizaje se detiene cuando el algoritmo alcanza un nivel aceptable de rendimiento.

Los algoritmos de aprendizaje supervisado se clasifican a su vez en algoritmos de clasificación y de regresión. Estos algoritmos serán explicados en los apartados 2.5 y 2.6 de este capítulo.

2.4.2 Aprendizaje No Supervisado

Los algoritmos de aprendizaje no supervisado trabajan de forma muy similar a los supervisados, con la diferencia de que éstos sólo ajustan su modelo predictivo tomando en cuenta los datos de entrada, sin importar los de salida. Al contrario que en el aprendizaje supervisado, el conjunto de datos no se encuentra etiquetado y no se tiene un resultado conocido proporcionado por un supervisor. Esta serie de algoritmos son adecuados para crear las etiquetas en los datos que posteriormente se utilizan para implementar tareas de aprendizaje supervisado [9].Es decir, los algoritmos identifican las agrupaciones inherentes a los datos no etiquetados y posteriormente asignan una etiqueta a cada valor de los datos.

Los algoritmos de aprendizaje no supervisado se clasifican a su vez en algoritmos de agrupamiento (en inglés *clustering*) y de reducción dimensional. El *clustering* es el proceso de agrupar los datos en un conjunto de clases disjuntas, llamadas *clusters*, de manera que los objetos de una clase tengan una gran similitud entre sí, mientras que los objetos de clases distintas sean más disímiles [10]. Los algoritmos de reducción dimensional mapean el conjunto de los datos a subespacios derivados del espacio original, de menor dimensión, que permiten hacer una descripción de los datos a un menor costo [11].

2.4.3 Aprendizaje Por Refuerzo

Los algoritmos de aprendizaje por refuerzo (en inglés *Reinforcement Learning* o RL) mapean situaciones definiendo modelos y funciones enfocadas en maximizar un escalar denominado señal de refuerzo o recompensa. Es una técnica de aprendizaje basada en prueba y error [12].

Este tipo de aprendizaje se refiere a uno o varios agentes interactuando con un entorno. El agente toma una decisión que implica la realización de una acción y esto conlleva un resultado en el entorno. Este resultado es visible para el agente en forma de observación y de recompensa inmediata. A medida que un agente va tomando acciones, recibirá a su vez el resultado de dichas acciones. Su objetivo es obtener el mejor resultado posible. Este resultado es análogo a que el agente maximice la recompensa acumulada a largo plazo.

RL es utilizado en el campo de la robótica, coches autónomos, y agentes para videojuegos.

2.5 Modelos de Regresión Lineal

El análisis de regresión es un enfoque estadístico para estudiar y modelar la relación entre variables, describir el estado de los datos, estimar los parámetros y ajustar el modelo para predecir y controlar [13]. Para crear estos modelos, se necesitan observaciones de las variables independientes $x_1, x_2 \dots x_n$ y de la variable dependiente y. Esta forma de análisis estima los coeficientes de la ecuación lineal, a partir de estas variables, que prevén mejor el valor de la variable dependiente. La regresión lineal se ajusta a una línea recta o a una superficie que minimiza las discrepancias entre los valores de salida previstos y reales. Si se considera una regresión lineal simple en la que solo hay una variable dependiente, donde la relación entre la variable dependiente e independiente esta expresada mediante un modelo lineal, la ecuación que le representa es la siguiente:

$$y = \beta_0 + \beta_1 X_1 \quad (1)$$

Donde β_0 corresponde a la constate de la recta (origen) y β_1 el parámetro de la pendiente de la recta. Cuando hay más de una variable explicativa (modelo de regresión lineal múltiple), se utiliza un subíndice para cada una de ellas, por ejemplo, para el caso de dos variables explicativas:

$$y_i = \beta_0 + \beta_1 X_i + \beta_2 X_2 \tag{2}$$

Teniendo en cuenta que los datos disponibles contienen errores, no siguen exactamente un modelo matemático lineal como este. En consecuencia, se define un modelo adecuado basado en los datos disponibles, denominado modelo predictivo, donde se tiene en cuenta la cantidad de error entre cualquier valor observado y estimado [13].

$$y_i = \beta_0 + \beta_1 X_i + \varepsilon_i;\ \mathrm{i} = 1,..\mathrm{n} \tag{3}$$

La ecuación (3) puede escribirse en notación más compacta de matrices:

$$Y = X.\beta + \varepsilon \tag{4}$$

Siendo,

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{5}$$

(5) la matriz de los valores de la variable objetivo del modelo para las *n* muestras del conjunto de datos.

$$X = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,m} \\ 1 & x_{2,1} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,m} \end{pmatrix} \tag{6}$$

(6) la matriz de los valores de las *m* características del conjunto de datos.

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \tag{7}$$

(7) la matriz del coeficiente que se van a aprender del modelo.

$$\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \tag{8}$$

(8) la matriz con el error cometido en cada una de las muestras del conjunto de datos.

El aprendizaje de este modelo consistirá en encontrar los mejores parámetros β para los datos disponibles que consigan minimizar la cantidad de error ε.

Los modelos de regresión presentan principalmente 3 problemas: sobreajuste, por no poder generalizar bien la línea de regresión por valores atípicos en los datos, escalabilidad por el rendimiento del modelo en función de la densidad del conjunto y del número de operaciones a realizar, y linealidad, por la correlación entre la variable objetivo y las características del conjunto de datos. Este tipo de errores se analizarán más adelante, enfocados a los modelos que se desarrollarán en el proyecto.

2.6 Modelos de Clasificación

De forma similar a los modelos de regresión, los modelos de clasificación producen una predicción de valor continuo que suele tener la forma de una probabilidad (es decir, los valores predichos de pertenencia a una clase para cualquier muestra individual están entre 0 y 1 y suman 1). Además de una predicción continua, los modelos de clasificación intentan predecir la etiqueta, grupo o categoría de las muestras del conjunto de datos. Aunque muchas de las técnicas de modelado de regresión también pueden utilizarse para la clasificación, la forma de evaluar el rendimiento del modelo es necesariamente diferente, ya que los parámetros de medición de la respuesta categórica son muy distintos [14].

Existen numerosos modelos de clasificación. En función del problema a resolver, la cantidad de datos o la estructura final que se quiera para el correcto entendimiento del problema seleccionado, se hace uso de cada uno de ellos. A continuación, se detallan los modelos elegidos para el desarrollo del proyecto.

2.6.1 Árboles de decisión

Un árbol de decisión (en inglés *Decision Tree* o DT) es una estructura de árbol invertido similar a un diagrama de flujo. Los modelos basados en árbol consisten en una o más sentencias *if-then* anidadas de las características que dividen los datos. Dentro de estas particiones, se utiliza un modelo para predecir el resultado. En la Figura 2 se muestra un ejemplo de árbol de decisión. Los principales componentes de este modelo son los nodos, las ramas y las hojas. Cada nodo interno representa una evaluación de una característica, cada rama representa el resultado de la evaluación y cada hoja contiene la etiqueta de una clase.

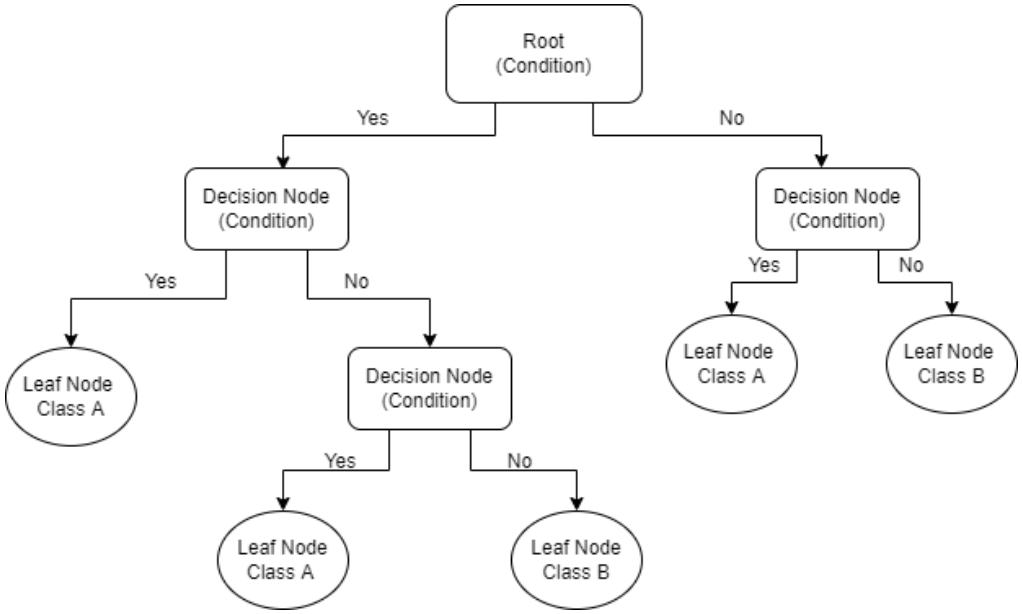

Figura 2. - Estructura árboles de decisión

Dentro de los nodos se diferencia el nodo superior o raíz (en inglés *root*), que representa a todo el conjunto de datos. Este nodo representa la primera división del conjunto de datos. Durante el proceso de aprendizaje del árbol, las muestras en cada nodo interior o nodo de decisión se dividen en subconjuntos basados en un atributo, y este proceso se repite en cada subconjunto derivado de una manera recursiva llamada "partición recursiva" [15]. La recursión finaliza cuando un subconjunto de un nodo tiene el mismo valor objetivo, cuando la división no mejora la predicción o cuando la división es imposible debido a restricciones definidas por el usuario. En cada paso del crecimiento de un árbol de decisión,

se selecciona una de las variables de entrada para dividir las muestras[1]. En función de la variable elegida, el punto de división se determina mediante una prueba de valor de atributos. La pureza de Gini y la entropía son las pruebas más utilizadas para esta función. Las ecuaciones (9) y (10) describen la entropía [16] y la impureza de Gini [17], respectivamente:

$$H_{entropy}(S) = -\sum_{y \in C} p(y) \log_2 p(y) \tag{9}$$

$$H_{gini}(S) = \sum_{y \in C} p(y)\,(1 - p(y)) = 1 - \sum_{y \in C} p(y)^2 \tag{10}$$

Donde,

- S representa el conjunto de datos
- C es un conjunto de clases
- $p(y)$ es la proporción del número de muestras con la etiqueta de clase en C.

El objetivo es medir la frecuencia con la que un elemento del conjunto elegido al azar sería etiquetado incorrectamente. Esta medida comprende valores entre 0 y 1, donde 0 se corresponde con la perfecta igualdad y 1 con la perfecta desigualdad. Tanto la impureza de Gini como la Entropía tienen valor 0 cuando sólo hay una clase en C y, por tanto, el error de clasificar esa muestra es nulo. Alcanzan el valor máximo, 0,5 cuando todas las clases son igualmente probables, presentando una forma cóncava. La diferencia entre ambos métodos es que la entropía en lugar de utilizar probabilidades simples utiliza base logarítmica 2 de las probabilidades.

En la Figura 3 se puede observar gráficamente el error de clasificación de ambos criterios en función de los valores de impureza de cada uno de ellos.

[1] Cada división óptima se identifica acorde al impacto que tiene en ese momento. No se tiene en cuenta si es la división que dará lugar a mejores árboles en futuras divisiones.

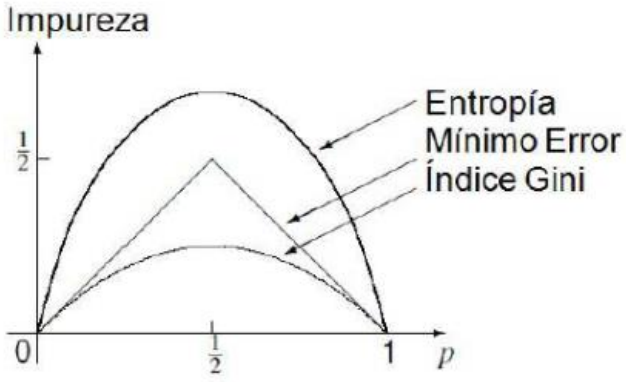

Figura 3. - Comparación de los criterios de Entropía e índice de Gini [18]

La regla de división está determinada por una reducción de la entropía y la impureza de Gini después de la división, lo que se denomina ganancia de información [15]:

$$G(r,S) = H(S) - \sum_{t} p(t)H(t) \tag{11}$$

Donde,

- r es la regla de división escogida
- t representa los nodos hijos inducidos por el conjunto S en el nodo actual
- p(t) es la proporción del número de muestras correspondientes a t

El atributo y el valor de la regla de división se determinan para obtener la máxima ganancia de información en el nodo actual. Tras el crecimiento del árbol completo, las clases de salida de las muestras se determinan en los nodos terminales. Cada nodo terminal decide un valor objetivo basado en la clase mayoritaria del nodo terminal. Cuando se observa una nueva muestra, se clasifica en uno de los nodos terminales del árbol en función de las variables de entrada y se predice que el objetivo es la clase mayoritaria en el nodo de la hoja.

El proceso de construcción de árboles mencionado tiende a reducir rápidamente el error de entrenamiento, es decir, el modelo se ajusta a las observaciones empleadas como entrenamiento. Como consecuencia, se genera un sobreajuste (en inglés *overfitting*) que reduce su capacidad predictiva al aplicarlo a nuevos datos. La razón de este comportamiento radica en la facilidad con la que los árboles se ramifican adquiriendo estructuras complejas. Existen dos estrategias

para prevenir el problema de *overfitting* de los árboles: limitar el tamaño del árbol (parada temprana, en inglés *early stopping*) y el proceso de podado (en inglés *pruning*). Estas estrategias se realizan ajustando los hiperparámetros del modelo. Este ajuste será explicado más adelante en el punto 4.2.3.5, cuando se introduzca el término hiperparámetro para la aplicación de estas técnicas.

La utilización de árboles de decisión para resolver problemas de clasificación se debe a las numerosas ventajas que presenta frente a otros modelos. En primer lugar, el árbol de decisión no es paramétrico. Durante el entrenamiento, crea reglas lógicas *if-then* que no requieren una suposición implícita sobre las distribuciones o relaciones subyacentes de las variables de entrada y salida. Por lo tanto, es apropiado para los problemas generales de análisis de datos en los que es común un conocimiento previo mínimo de los datos. En segundo lugar, el árbol de decisión es inherentemente no lineal. Puede utilizar la misma variable varias veces para crear reglas de división durante el crecimiento del árbol, revelando una relación no lineal entre las variables. Aunque la introducción de la no linealidad suele aumentar la complejidad del modelo y reduce la capacidad de interpretación en otras técnicas de análisis de datos, el árbol de decisión puede aplicar la no linealidad de forma sencilla [15]. Por último, la interpretación de los resultados en un árbol es muy sencilla. En la fase de prueba, el árbol es útil para predecir rápidamente las nuevas observaciones, y el resultado se interpreta fácilmente utilizando unas simples sentencias *if-then*.

2.6.2 Random Forest

El modelo de Bosques Aleatorios (en inglés *Random Forest* o RF), propuesto originalmente por Breiman (2001) [19], está formado por un conjunto (en inglés *ensemble*) de árboles de decisión individuales.

Todos los modelos de aprendizaje estadístico y machine learning sufren el problema de equilibrio entre sesgo (en inglés *bias*) y varianza. El término *bias* hace referencia a cuánto se alejan en promedio las predicciones de un modelo respecto a los valores reales, reflejando cómo de capaz es el modelo de aprender la relación real que existe entre los predictores y la variable respuesta. El término varianza hace referencia a cuánto cambia el modelo en función de los datos utilizados en su entrenamiento. Los métodos de *ensemble* combinan múltiples modelos en uno

nuevo con el objetivo de lograr un equilibrio entre estos términos, consiguiendo así mejores predicciones que cualquiera de los modelos individuales. Uno de los tipos de *ensemble* y el utilizado por RF es el *bagging* (***B**ootstrap **agg**regat**ing***) [20]. Este método ajusta múltiples modelos en paralelo con un subconjunto diferente de muestras de entrenamiento *bootstrap*[2]. La salida es el resultado de la votación de todos los árboles.

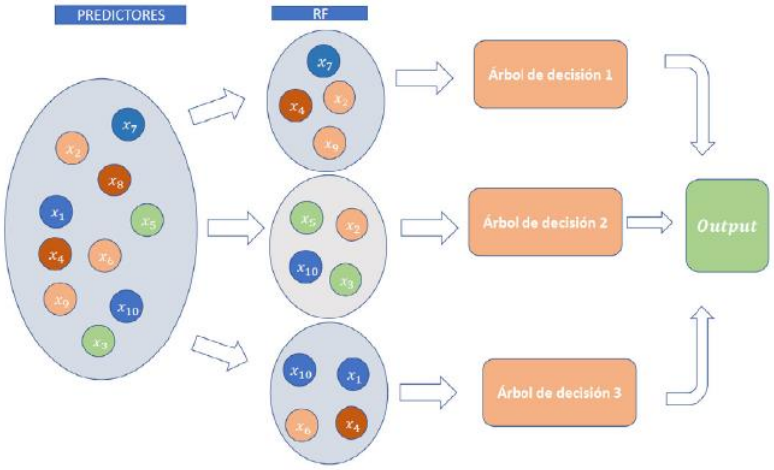

Figura 4. - Selección de predictores en RF [20]

Como se observa en la Figura 4, RF solo utiliza un número m de predictores p en cada uno de los árboles, siendo $m < p$. Además, la selección de m en cada árbol es aleatoria. Esto hace que, si en el conjunto de datos hay un predictor muy influyente, se puedan crear árboles con otros predictores menos influyentes, consiguiendo disminuir la correlación y la varianza. Al igual que DT, RF presenta problemas de sobreajuste. Para intentar evitar estos problemas, se somete al modelo a un ajuste de hiperparámetros.

2.6.3 AdaBoost

Al igual que RF, AdaBoost (***Ada**ptive **Boost**ing*) es un método de *ensemble*, que combina diferentes modelos en uno nuevo. Creado por Freund and Schapire (1996) [21], es un diseño mejorado del tipo de ensemble *Boosting*. Al contrario que *bagging*, donde los modelos simples se ajustaban de forma paralela, en los

[2] *Bootstrapping* es cualquier prueba o métrica que utiliza el muestreo aleatorio con reemplazo.

algoritmos de *boosting*, los modelos simples son utilizados secuencialmente. En la Figura 5 se representa una comparativa de los tres modelos.

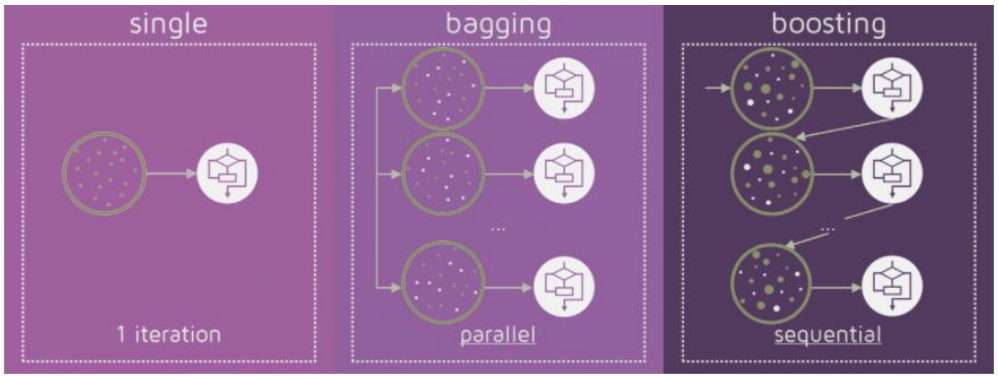

Figura 5. - Comparativa de construcción de modelos [22]

AdaBoost genera una secuencia de clasificadores débiles donde en cada iteración el algoritmo encuentra el mejor clasificador basado en los pesos de la muestra actual. A todos los datos se les asigna inicialmente el mismo peso. Este peso se va actualizando con cada iteración según los ejemplos mal clasificados. Las muestras que se clasifican incorrectamente en la iteración n reciben más peso en la iteración$(n + 1)$, mientras que las muestras correctamente clasificadas reciben menos peso en la siguiente iteración. Esto significa que las muestras difíciles de clasificar reciben pesos cada vez mayores hasta que el algoritmo identifique un modelo que las clasifique correctamente. Por lo tanto, cada iteración del algoritmo debe aprender un aspecto diferente de los datos, centrándose en las regiones que contienen muestras difíciles de clasificar.

La secuencia global de clasificadores ponderados se combina en un conjunto y tiene un gran potencial para clasificar mejor que cualquiera de los clasificadores individuales. Al igual que los anteriores modelos, los problemas que presenta AdaBoost son por *overfitting*.

2.6.4 Métricas de evaluación para clasificación

Como se desarrollará más adelante en la metodología del proyecto, los modelos una vez entrenados pasan por una etapa de evaluación. En este apartado se detallan las métricas que se utilizarán para medir el rendimiento de los modelos de clasificación utilizados.

- Matriz de error: también conocida como matriz de confusión, muestra los resultados reales de clasificación de la muestra contra los resultados

de predicción clasificados por el modelo. En nuestro caso, la clasificación es binaria, por lo que la matriz de confusión tendrá el siguiente aspecto.

		VALORES REALES	
		Clase 0	Clase 1
PREDICCIÓN	Clase 0	TP	FP
	clase 1	FN	TN

Figura 6. - Matriz de confusión

- TP (*True positive*): Hace referencia a las muestras clasificadas por el modelo como clase 0 y el valor real de la muestra pertenece a la clase 0. Se ha acertado la predicción en este conjunto.
- FP (*False positive*): Hace referencia a las muestras clasificadas por el modelo como clase 0 y el valor real de la muestra pertenece a la clase 1. No se ha acertado la predicción en este conjunto.
- FN (*False negative*): Hace referencia a las muestras clasificadas por el modelo como clase 1 y el valor real de la muestra pertenece a la clase 0. No se ha acertado la predicción en este conjunto.
- TN (*True negative*): Hace referencia a las muestras clasificadas por el modelo como clase 1 y el valor real de la muestra pertenece a la clase 1. Se ha acertado la predicción en este conjunto.

A partir de esta matriz, se obtienen diferentes métricas de evaluación:

- Exactitud (*Acurracy* en inglés): calcula la proporción de predicciones clasificadas de forma correcta por el modelo.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

- Precisión (*Precision* en inglés): calcula la exactitud de las predicciones de cada clase.

$$Precision_0 = \frac{TP}{TP + FP};\ Precision_1 = \frac{TN}{TN + FN} \tag{13}$$

- Sensibilidad (*Recall* en inglés): calcula la tasa de aciertos de cada clase.

$$Recall_0 = \frac{TP}{TP + FN};\ Recall_1 = \frac{TN}{TN + FP} \tag{14}$$

- F1: El valor F1 se utiliza para combinar las medidas de *precision* y *recall* en un sólo valor. Esta medición es muy habitual para poder comparar de forma más sencilla el rendimiento del modelo.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{15}$$

2.7 Python y librerías de uso

Durante la evolución de la IA, ML y proyectos en general de análisis de datos, se han desarrollado numerosas herramientas y lenguajes que se adaptan a las soluciones que estos proyectos buscan. A la hora de elegir el mejor lenguaje se tienen en cuenta requisitos como, buen rendimiento en tiempo de ejecución, buen soporte de herramientas y una gran comunidad de programadores.

Entre la comunidad de científicos de datos, las herramientas preferidas son Python y R. Ambas herramientas destacan por su enorme catálogo de paquetes que facilitan su utilización para manejo, análisis y representación de datos, además de incluir paquetes para la elaboración y evaluación de los modelos.

Para este proyecto se ha decidido utilizar el lenguaje Python [23] por su sencillez y, en especial, por el conjunto de librerías que ofrece para la utilización de modelos de aprendizaje. A continuación, se nombran las principales librerías utilizadas:

- *NumPy* [24]: De forma similar a Matlab, admite una gran cantidad de matrices dimensionales avanzadas y operaciones de matriz, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- *Pandas* [25]: herramienta basada en *NumPy*, creada para resolver tareas de análisis de datos. Incorpora una gran cantidad de bibliotecas y algunos modelos de datos estándar, proporcionando las herramientas necesarias para manipular de manera eficiente grandes conjuntos de datos.

- *Matplotlib* [26]*:* biblioteca para la generación de gráficos a partir de datos contenidos en listas o *arrays.* También se ha hecho uso de la librería *Seaborn* [27], basada en *Matplotlib* para algunos gráficos.
- *Sckit-Learn* [28]*:* biblioteca para el entrenamiento, validación y evaluación de los modelos. Incluye algoritmos de los diferentes tipos de aprendizaje además de sus métricas de evaluación.

La herramienta software de código abierto utilizada será la distribución Anaconda. Para su utilización se instalará la interfaz gráfica de usuario (*graphical user interface* en inglés o GUI) *Anaconda Navigator*.El entorno de trabajo utilizado será *JupyterLab* [29].

Capítulo 3

3 ESPECIFICACIONES Y RESTRICCIONES DE DISEÑO

Para la realización de este proyecto se han establecido las siguientes especificaciones y restricciones de diseño:

- El proyecto se realizará mediante software *open source*.
- Utilización de la distribución Anaconda del lenguaje de programación Python como herramienta software. El entorno de trabajo será *JupyterLab*.
- Se utilizará un conjunto de datos reales, obtenidos de la estación base en un periodo aproximado de 12 meses.
- Se utilizarán otras fuentes de información que recojan datos de la radiación solar.
- Se deben filtrar valores anómalos que puedan afectar potencialmente al conjunto de datos de entrada al modelo.
- Se deben eliminar los valores procedentes de fallos puntuales de comunicación en los equipos del sistema.
- Se deben implementar diferentes modelos de machine learning.
- Todos los algoritmos utilizados deben presentar métricas de evaluación

Capítulo 4

4 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción

En un proyecto de desarrollo software existe un ciclo de vida estándar con varias fases o pasos: análisis, especificaciones, diseño e implementación. En muchas ocasiones, se utilizan modelos en cascada, de modo que solo cuando una de estas fases termina se continua con la siguiente. En el caso de los proyectos relacionados con análisis de datos, estos no siguen este ciclo de vida estándar, si no que tienen su propio ciclo de vida de análisis de datos.

Como se observa en Figura 7, el proceso es dinámico e iterativo, por lo que la ejecución de los procesos no es estricta y con frecuencia se puede pasar de uno a otro proceso, de atrás hacia delante y viceversa. Estas transiciones dependen del resultado de cada fase o la planificación de la siguiente tarea por ejecutar [30].

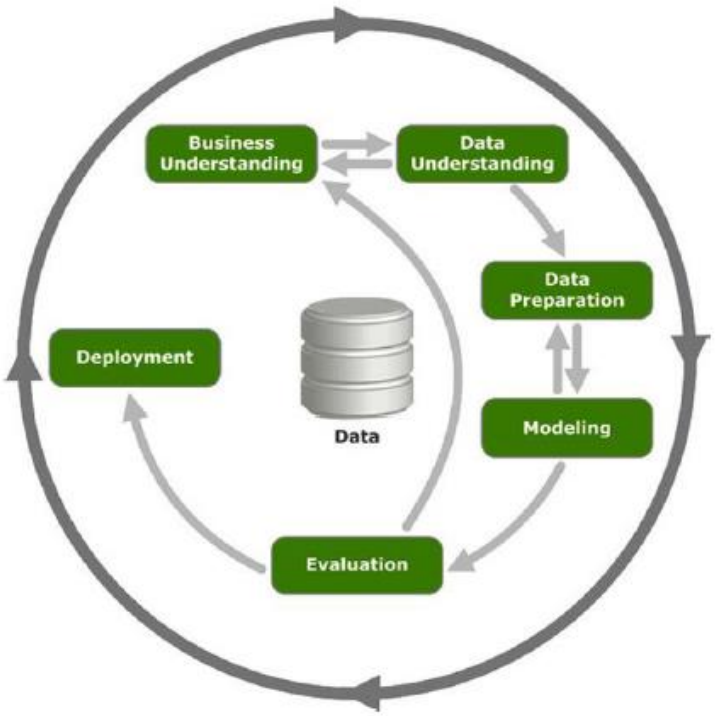

Figura 7. - Metodología de un proyecto de análisis de datos [31]

Desde la comprensión del negocio o problema hasta la implementación, se encuentran las fases de comprensión y preparación de los datos, modelado y evaluación de algoritmos de aprendizaje. La dirección a seguir se representa con flechas.

4.2 Metodología propuesta

Este proyecto trata de resolver un problema de aprendizaje automático. Este capítulo está centrado en cuatro de las seis fases de la Figura 7. El proceso comenzará con la recuperación y extracción de datos. Estos datos pasarán por diferentes fases de preparación, modelado y evaluación, hasta alcanzar el conocimiento adecuado sobre el comportamiento de los mismos. La Figura 8 muestra el diagrama de flujo de esta serie de pasos.

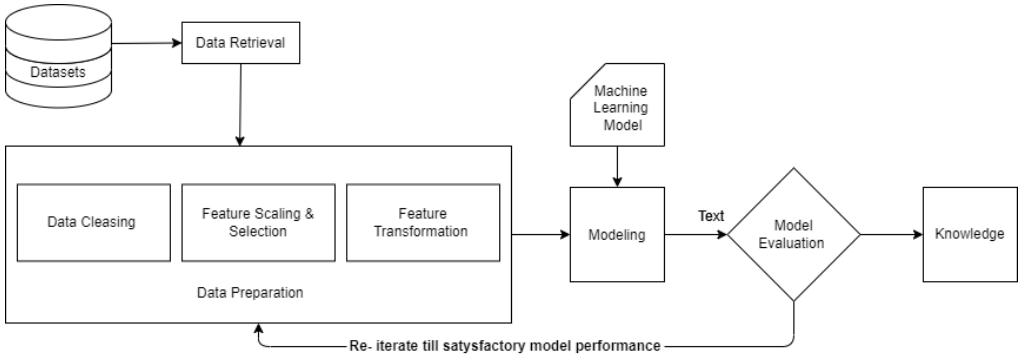

Figura 8. - Diagrama de flujo de los datos. Basada en [32]

A lo largo de todo este apartado se irán describiendo, de forma ordenada y detallada, las diferentes partes que componen este proceso.

4.2.1 Obtención de datos

Los datos utilizados para entrenar los modelos de ML se han extraído de dos fuentes diferentes. Por un lado, se encuentran los datos relacionados con la infraestructura de red móvil. Estos datos se han descargado del portal de monitorización de la empresa operadora de infraestructura. Este monitor web obtiene los datos de la infraestructura a través de los portales SCADA[3](*Supervisory Control And Data Acquisition*) de cada uno de los equipos del sistema. Por otro lado, se encuentran valores de radiación solar descargados de SISIFO[4], herramienta de simulación del Instituto de Energía Solar de la Universidad Politécnica de Madrid.

[3] SCADA: software para ordenadores que permite controlar y supervisar procesos industriales a distancia.
[4]Acceso al simulador a través de la página: Home Page - Simulación fotovoltaica (sisifo.info)

De ambos portales se han descargado datos de un periodo de 12 meses, tomados cada hora.

4.2.2 Preprocesamiento de datos

Para lograr que un algoritmo de aprendizaje supervisado sea capaz de extraer conocimiento de un conjunto de datos es fundamental que los datos sean de calidad. El término datos de calidad en Machine Learning se mide en relación con la cantidad de datos disponibles, si son o no son representativos y si no están sesgados [31]. La limpieza de datos (*data cleansing* en inglés) es un proceso necesario para asegurar la calidad de los datos. Este proceso incluye operaciones de descubrimiento, filtrado y corrección de datos erróneos.

A continuación, se detallan las etapas de preparación sobre el conjunto de datos disponible, mostradas en la Figura 8.

4.2.2.1 Eliminar fallos

En el conjunto de datos obtenidos del monitor web se observan datos erróneos, vacíos o incorrectos. Estos datos deben eliminarse o sustituirse ya que pueden distorsionar los resultados y con ellos las conclusiones que podemos sacar. Se distinguen dos tipos de errores en el conjunto:

- **NaN**: Los *NaN* ("Not a Number ") son valores vacíos no computables. Este dato aparece en nuestro conjunto cuando el valor que proporciona el equipo es nulo. Dependiendo de la fuente de datos donde aparece esta constante, se hará un tratamiento distinto para rellenar o eliminar el dato, ya que puede hacer referencia a que un equipo esté en parada, como puede ser el inversor fotovoltaico en horas nocturnas, o el grupo electrógeno en horas diurnas con suficiente potencia fotovoltaica. En este caso, el valor se rellenará con un 0, ya que el modelo necesita conocer esos valores de potencia nulos. Si, por el contrario, esta constante aparece de forma consecutiva en determinadas características que no puedan hacer referencia a la parada de un equipo, como puede ser la temperatura o el estado de carga de las baterías, este error se clasificará como error de comunicación y se eliminarán estas filas del conjunto de datos.
- '**-**': Los valores registrados por el monitor web con el símbolo '- 'aparecen cuando hay comunicación con el equipo correspondiente, pero en el

momento de registrar el dato, el equipo no tiene ese dato listo. Este tipo de errores se clasificarán como fallos de comunicación puntual y se rellenarán haciendo la media con el dato de la hora anterior y posterior. Si estos datos tampoco estuvieran disponibles, se eliminarán estas filas del conjunto de datos.

4.2.2.2 Eliminar outliers

Un valor atípico (en inglés *outlier*) es un dato que difiere significativamente de los otros de una muestra de datos. Esta desviación puede indicar una anomalía o error ocurrido en la medición de la variable, que puede provocar una distorsión de los resultados y, por tanto, un aprendizaje del modelo con datos que no se asemejan a la realidad del sistema. Por el contrario, estas desviaciones pueden representar comportamientos anómalos ocurridos de forma real en el sistema y que son representativos de su funcionamiento, y, por tanto, válidos y necesarios para el aprendizaje del modelo.

Para determinar si estos valores son errores que debemos eliminar del conjunto de datos o desviaciones representativas del sistema que deben mantenerse, se deben gestionar de forma individual en cada variable del sistema. Para su tratamiento se realizan dos técnicas de detección:

- Visualización de los datos de forma gráfica utilizando los métodos disponibles en *MatPlotLib*, *box plot* y *scatter plot*. Un diagrama de caja (en inglés *box plot*) es un método para representar gráficamente un conjunto de datos a través de sus cuartiles. Los valores atípicos se representan como puntos individuales. Un diagrama de dispersión (en inglés *scatter plot*) es un tipo de diagrama que utiliza coordenadas cartesianas para mostrar los valores de dos variables para un conjunto de datos. Estas técnicas de representación permiten visualizar de forma rápida y sencilla los valores atípicos del conjunto de datos de cada variable, por lo que será el primer paso para la detección de estos valores.
- Cálculo del rango intercuartil o IQR. El IQR es la diferencia entre el percentil 75 y el percentil 25. Todos los valores que se alejen más de un 50% de este intervalo se considerarán outliers y deberán ser analizados.

Esta técnica se utilizará de forma posterior a la visualización gráfica cuando se hayan detectado outliers en los gráficos.

Una vez identificados los outliers en cada una de las variables se tomarán dos decisiones: si por la naturaleza del sistema híbrido o de la propia variable, los valores considerados outliers son desviaciones válidas para el aprendizaje del modelo, se mantendrán estos valores. Si, por el contrario, se trata de errores en la fase de medición, se sustituirá este valor con la media de los valores de la hora anterior y posterior, de igual forma que se hizo con los fallos de comunicación.

4.2.2.3 Selección de variables

En ML, la selección de variables es un proceso de selección de las características más relevantes de un conjunto de datos previa a la construcción del modelo. El objetivo es reducir las dimensiones creando un subconjunto de características mínimo que represente la cantidad máxima de varianza en los datos. Menor cantidad de datos se traduce a un menor tiempo de entrenamiento y en consecuencia de computación. Esta reducción mejorará la capacidad predictiva de nuestro modelo, proporcionando resultados más rápidos y eficientes.

Se utilizan técnicas de selección de variables debido a que, en ocasiones, los datos contienen algunas características que son redundantes y/o irrelevantes y, en consecuencia, pueden eliminarse sin perder información significativa. Con el objetivo de eliminar las características redundantes del conjunto de datos disponibles, se analizará la correlación existente entre las variables de nuestro set de datos.

Actualmente, el conjunto de datos está formado por 13 variables. Para el análisis de las variables, se representa la matriz de correlación utilizando el coeficiente de correlación lineal de Pearson [33]. Este índice mide el grado de covarianza entre distintas variables relacionadas linealmente. El coeficiente de correlación de Pearson, r_{xy}, viene definido por la siguiente expresión:

$$r_{xy} = \frac{\Sigma_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\Sigma_{i=1}^{n}(y_i-\bar{y})^2}} \quad (16)$$

donde,

- n es el tamaño de la muestra.

- x_i, y_i son las muestras individuales indexadas con i.
- $\bar{x}$ es la media muestral definida por $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- $\bar{y}$ de forma análoga a $\bar{x}$.

Los valores de correlación de Pearson oscilan entre [-1,1], especificando el signo, la dirección de tal valor:

- Si $r_{xy} = 1$, existe una correlación perfecta positiva. Indica una dependencia total entre las dos variables, de forma que, cuando una de ellas aumenta o disminuye, la otra lo hace en la misma proporción. Valores cercanos a 1 indican una fuerte correlación positiva.
- Si $r_{xy} = 0$, o valores cercanos a 0, indica la ausencia de correlación entre las variables.
- Si $r_{xy} = -1$, existe una correlación perfecta negativa. Indica una dependencia total, pero en este caso inversa, de forma que cuando una de ellas aumenta, la otra disminuye en la misma proporción y viceversa. Valores cercanos a -1 indican una fuerte correlación negativa.

Se analizará la matriz de correlación, eliminando del conjunto de datos aquellas variables que presenten valores de r_{xy} cercanos a 1 o -1.

4.2.2.4 Transformación de datos

Los datos de entrada al modelo no disponen de una escala homogénea, por lo que, cada variable se mueve en un rango diferente del resto. Mientras que los valores de potencia del grupo electrógeno varían entre [0-5000] (W), los valores de temperatura lo hacen entre [0–40] (°C). Es por ello, que es necesario acotarlos a un rango de valores que permita compararlos entre sí independientemente de su naturaleza.

Los algoritmos de ML tienen mejor rendimiento cuando las características se encuentra en la misma escala y, por tanto, tiene el mismo peso dentro del modelo. Las técnicas más comunes son:

- Estandarización: proceso a partir del cual un conjunto de datos, que siguen una distribución normal, son transformados a una distribución

normal con media 0 y desviación típica 1. Para ello, se utiliza la siguiente fórmula:

$$x_i' = \frac{x_i - \mu}{\sigma} \quad (17)$$

Donde x_i es el dato que queremos estandarizar, μ es el valor medio de todos los datos y σ es la desviación típica de todos los datos.

- Normalización: se trata de otra alternativa a la estandarización de las características, que consiste en ajustarlas a un rango predefinido, normalmente en el rango [0,1]. Esta será la técnica que utilizaremos para nuestro modelo. Utiliza la siguiente fórmula:

$$X_{normalizada} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (18)$$

Una vez aplicada esta técnica, finaliza el proceso de preparación del conjunto de datos disponibles, considerando que los datos actuales son datos de calidad e inteligibles para el modelo a aplicar.

4.2.3 Procesamiento de los datos

Una vez preparado el conjunto de datos inicial, se hará uso de algoritmos de ML con el objetivo de encontrar patrones que expliquen el comportamiento de los datos y de los que se extraerán las conclusiones del proyecto. Los algoritmos se utilizarán para resolver un problema de clasificación binaria, donde la variable objetivo será el estado del grupo electrógeno (ON/OFF) y las demás características se utilizarán como predictores para obtener esta clasificación.

Los algoritmos que se utilizarán son: *Decision Tree*, *Random Forest* y *AdaBoost*.

Los tres algoritmos seguirán el mismo ciclo de modelado. A continuación, se muestra en la Figura 9 el diagrama de flujo que seguirán y posteriormente se detalla cada uno de los pasos.

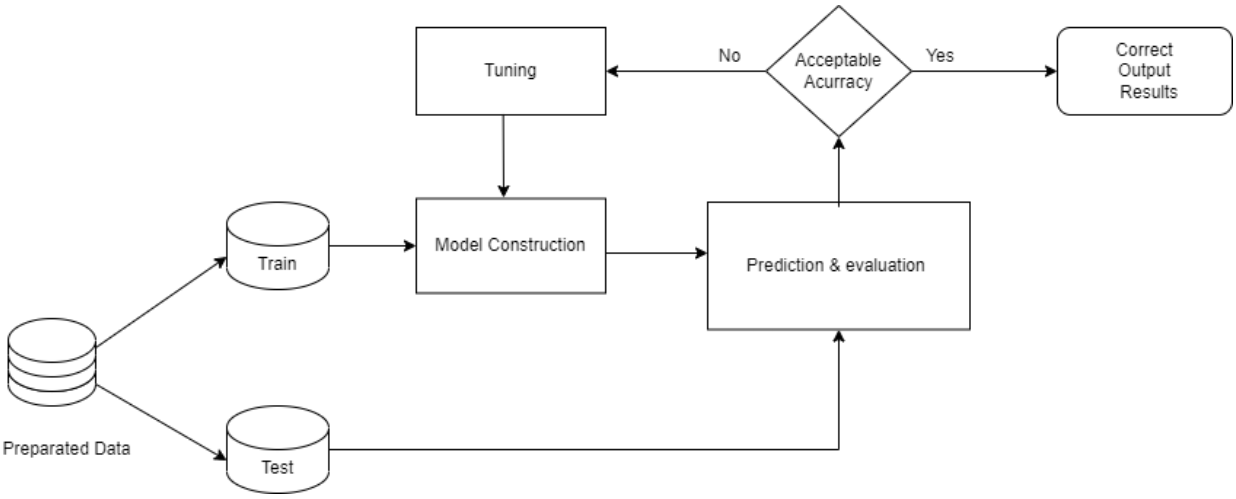

Figura 9. - Diagrama de flujo del procesamiento de datos

4.2.3.1 División del conjunto de datos

El primer paso en el procesado es la división del conjunto de datos en subconjuntos. Una parte de los datos se utilizará para construir y ajustar el modelo y otra parte para medir el rendimiento. El objetivo de esta división es poder cuantificar el error de predicción.

El set de datos utilizado para aprender el modelo se conoce como conjunto de datos de entrenamiento (*Train*). Los registros que componen este grupo se denominan muestras de entrenamiento y se seleccionan aleatoriamente de la población de la muestra. El modelo es construido colectivamente a partir del conjunto de entrenamiento. El set de datos utilizado para medir la calidad del modelo se conoce como conjunto de datos de prueba (*Test*). Los datos de prueba son un grupo de ejemplos que se utilizan únicamente para evaluar el rendimiento del clasificador seleccionado [34]. Las muestras de prueba se seleccionan aleatoriamente y son independientes de las muestras de entrenamiento.

4.2.3.2 Construcción del modelo

Una vez seleccionado el conjunto de entrenamiento, se realiza la construcción del modelo. En esta fase, se selecciona el algoritmo de ML que se utilizará como clasificador y se entrena. El algoritmo examina el conjunto de datos de entrenamiento y comienza a aprender las combinaciones óptimas de variables que generarán un buen modelo predictivo [35].

Cada algoritmo de ML tiene sus propios parámetros que determinarán el aprendizaje del modelo. Estos parámetros serán explicados de forma más detallada en el punto 4.2.3.5. Inicialmente, se dejan valores por defecto.

4.2.3.3 Predicción

Con el modelo construido, se realizan prediciones de valores de salida. Estas predicciones se realizan con el conjunto de prueba seleccionado aleatoriamente en la división de datos. Este grupo de datos, son muestras que el modelo no ha "visto" clasificadas para su construcción y que permitirán estimar su precisión. Las predicciones obtenidas se comparan con las clasificaciones reales de los ejemplos en la fase de evaluación.

4.2.3.4 Evaluación

Como se mencionó en el apartado 2.6.2, en ML, aumentar el *bias* provoca disminuir la varianza y viceversa, es por ello necesario buscar el equilibrio de forma que se minimice el error total[5]. Analizar el error permitirá evaluar el rendimiento del modelo.

Durante el entrenamiento del clasificador seleccionado, se cometen errores. Si el error del entrenamiento es elevado, el modelo tiene problemas para aprender. Esto se denomina desajuste (en inglés *underfitting*) y sucede cuando el modelo de ML construido es demasiado simple, presentando un elevado sesgo. Esto puede ocurrir por datos insuficientes, incompletos o con mucho ruido. Para solucionar problemas de *underfitting* es necesario volver a la fase de preparación de datos. Si, por el contrario, el error de entrenamiento es bajo, el modelo ha sido capaz de aprender la relación entre los datos de entrada y los resultados. En este caso, es necesario analizar el error de las predicciones realizadas en el conjunto de prueba. El sobreajuste ocurre cuando el modelo aprende las respuestas, en vez de generalizar patrones en el comportamiento de los predictores. Esto significa que el ruido o las fluctuaciones aleatorias en los datos de entrenamiento son recogidos y aprendidos por el modelo. En este caso, el modelo presenta una alta varianza, siendo muy cambiante en función de los datos de entrenamiento seleccionados. En el apartado 4.2.3.5 se detallarán métodos para evitar *overfitting*.

[5] El error total se calcula con la siguiente fórmula: $Error\ total = Bias^2 + Varianza$.

Para la evaluación de los errores indicados se utilizarán las métricas mencionadas en el apartado 2.6.4.

4.2.3.5 Ajuste de hiperparámetros.

Muchos modelos tienen parámetros importantes que no pueden estimarse directamente a partir de los datos [14]. Estos parámetros se denominan hiperparámetros en ML y son los que determinan el proceso de aprendizaje. En el caso de Árboles de Decisión alguno de sus hiperparámetros son la profundidad total del árbol, el criterio de clasificación o el número mínimo de muestras en un nodo.

El valor de los hiperparámetros debe establecerse antes de que comience el proceso de aprendizaje, por lo que, de forma inicial, estos parámetros se ajustan manualmente en la fase de construcción, introduciendo valores aleatorios o utilizando los que el modelo tiene por defecto. Tras un primer entrenamiento y evaluación, se realiza de forma recursiva un ajuste de hiperparámetros hasta obtener un rendimiento óptimo. Como se ha visto en el apartado 4.2.3.1, de forma inicial, el conjunto de datos se divide en datos de entrenamiento y datos de prueba. Este proceso puede provocar la aparición de *overfitting* debido a que la división de los datos se realiza de forma aleatoria, por lo que, si repetimos varias veces el proceso siguiendo esta división, el conjunto de test incluirá alguna muestra que el modelo ya ha "visto" anteriormente y aprenderá los resultados.

Una forma de evitar el sobreajuste mencionado es utilizar métodos de validación cruzada. La validación cruzada (en inglés *cross validation* o CV) es un método estadístico para evaluar y comparar algoritmos de aprendizaje dividiendo los datos de entrenamiento a su vez, en dos segmentos: uno utilizado para aprender o entrenar un modelo y otro para validarlo [36]. La Figura 10 muestra la comparativa de la división de datos.

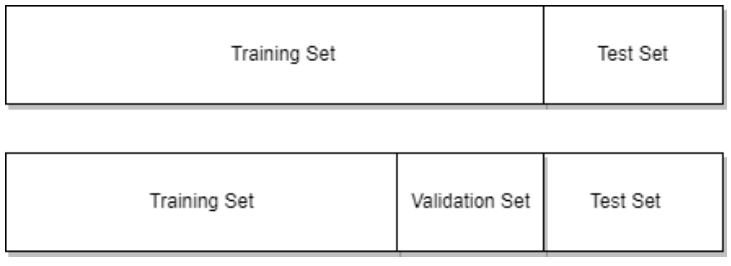

Figura 10. - División del conjunto de datos

El modelo se ajusta empleando un subconjunto de muestras del conjunto de entrenamiento y se evalúa con el conjunto de validación. Este proceso se repite múltiples veces y los resultados se añaden y promedian. Gracias a las repeticiones, se compensan las posibles desviaciones que puedan surgir por el reparto aleatorio de las muestras. La diferencia entre métodos de CV se determina por la forma en que se generan los subconjuntos de entrenamiento/validación. A continuación, se detallan algunos de ellos:

- *k-Fold CV*: Las muestras se dividen aleatoriamente en *k*-grupos de tamaño aproximadamente igual, *k*-1 grupos se emplean para entrenar el modelo y el grupo restante se emplea como validación. Este proceso se repite *k* veces utilizando un grupo distinto como validación en cada iteración. El proceso genera *k* estimaciones del error cuyo promedio se emplea como estimación final.

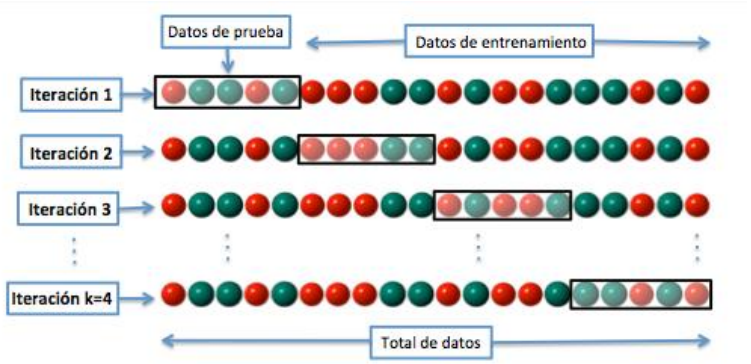

Figura 11[6]. - Ejemplo de técnica k-Fold CV para k=4 [37].

- *Leave One Out CV (LOOCV)*: Se emplea como conjunto de entrenamiento todas las observaciones disponibles excepto una, que se excluye para emplearla como validación. Si se emplea una única observación para calcular el error, este varía mucho dependiendo de qué observación se haya seleccionado. Para evitarlo, el proceso se repite tantas veces como observaciones disponibles, excluyendo en cada iteración una observación distinta, ajustando el modelo con el resto y calculando el error con dicha observación.

[6] Los datos de prueba que identifica la figura hacen referencia al conjunto de validación y son independientes del conjunto de test como se muestra en la Figura 10.

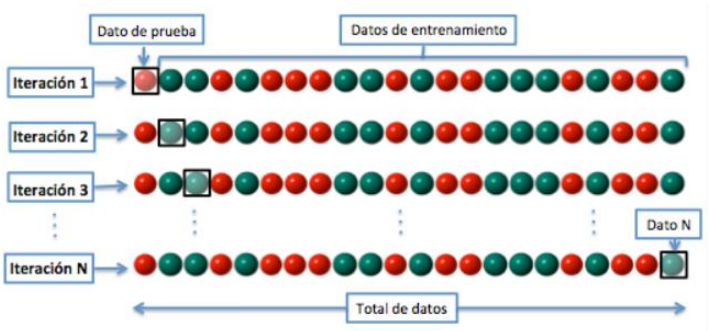

Figura 12. - Técnica LOOCV [37]

La técnica que se utilizará como método de validación será k-*Fold CV*. El ajuste de hiperparámetros se realizará de forma manual y automática. Ambas formas se explicarán detalladamente en el capítulo 5.

Capítulo 5

5 DESARROLLO Y ANÁLISIS DE RESULTADOS

5.1 Desarrollo de la metodología

Durante este apartado se expone el desarrollo práctico de la metodología expuesta en el capítulo 4 siguiendo el orden marcado anteriormente. El capítulo finaliza con el análisis de los resultados obtenidos.

5.1.1 Obtención de datos

Para el entrenamiento de los modelos de clasificación binaria, se han utilizado las dos fuentes de información mencionadas en el punto 4.2.1. Por un lado, se ha generado un archivo *Date_site.xlsx* con los datos de la estación base. Para obtener estos datos se ha desarrollado un *script* que recorre, y almacena en este archivo, los datos recogidos por el monitor web durante un año completo en una estación base concreta localizada en la Comunidad de Madrid. La Figura 13 recoge un esquema del conexionado de los equipos de alimentación y comunicación.

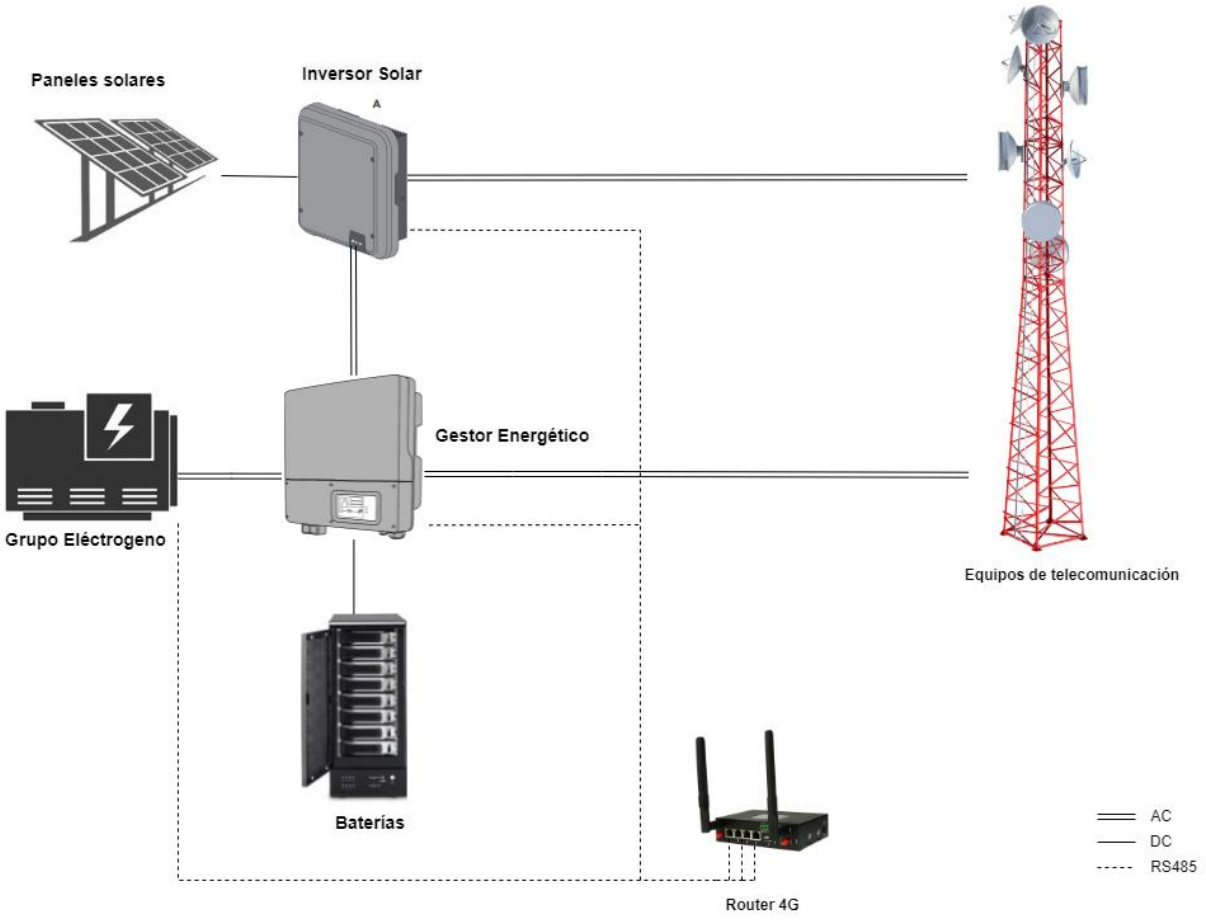

Figura 13. - Diagrama del Sistema Híbrido

Como se observa en la Figura 13, la alimentación de los equipos de telecomunicación conmuta entre el inversor solar, al que se encuentran conectados los paneles solares, y el gestor energético, donde se encuentra conectado el GE y el banco de baterías. Para la captura de datos, los equipos se encuentran conectados a un router 4G con tarjeta SIM multioperador. En este equipo se encuentra configurado una red VPN (*Virtual Private Network*), a través del protocolo L2TP (*Protocolo de Layer 2 Tunneling Protocol*). Cada equipo tiene configurado una dirección IP (*Internet Protocol*) estática. Por protocolo Modbus, el router se comunica con los equipos a través del bus RS-485, y el monitor Web recoge los datos a través de una tabla de enrutado a cada dispositivo configurada en el router. Los datos de la estación se recogen cada 5 minutos. El script generado registra datos en el archivo *Date_site.xlsx* cada hora, haciendo una media de los datos recogidos cada 5 minutos. La **¡Error! No se encuentra el o rigen de la referencia.** resume los variables recogidas.

Tabla 1. - Variables archivo Date_site.xlsx

Variable	***Descripción***
FECHA	Referencia temporal. Formato *aa-mm-dd hh:mm*
CHARGE.BAT	Estado de carga de las baterías (en inglés *State Of Charge* o SOC). Es una medida adimensional. Se expresa en %.
POT.BAT	Potencia de las baterías (W)
STATUS.BAT	Estado de funcionamiento de las baterías (CHARGE-DISCHARGE)
POT.GE	Potencia del grupo electrógeno (W)
FUEL.GE	Litros de combustible del GE (L)
TIME.GE	Horas de funcionamiento del GE(h)
GE (ON/OFF)	Estado de funcionamiento del GE(ON-OFF)
POT.SOLAR	Potencia de los paneles solares(W)
STATUS.SOLAR	Estado de funcionamiento del inversor solar (ON-OFF)

Variable	***Descripción***
POT.TOTAL	Potencia total de la estación (W). Realiza la suma de las tres fuentes de potencia. POT.TOTAL= POT.BAT+ POT.GE+ POT.SOLAR
TEMPERATURA	Temperatura ambiente de la caseta (ºC)

Por otro lado, se ha generado otro archivo *sisifo_data.xlxs* obtenido de la plataforma SISIFO con los datos de una simulación en línea. En esta simulación se han configurado parámetros geográficos, rellenado las coordenadas de la localización de esta estación, y parámetros temporales. Se pueden ver estas configuraciones en la Figura 14 y Figura 15.

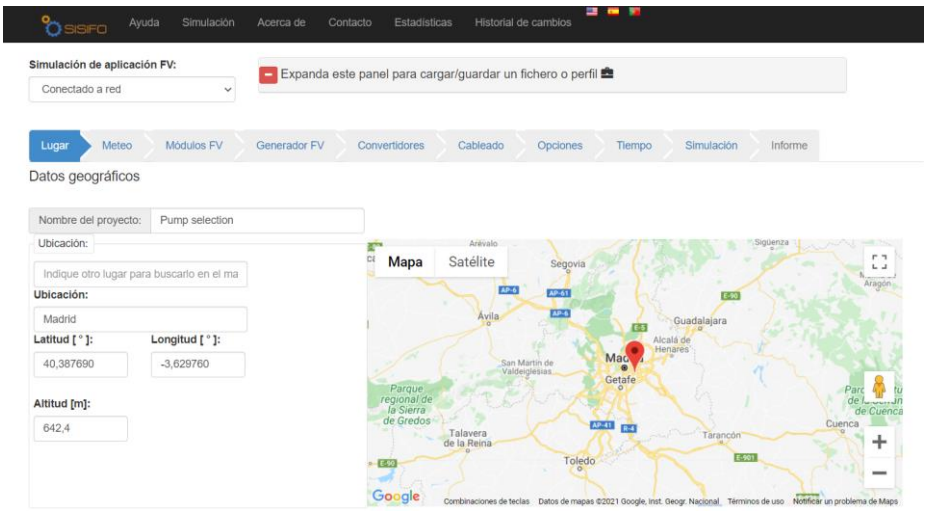

Figura 14. - Configuración de parámetros geográficos

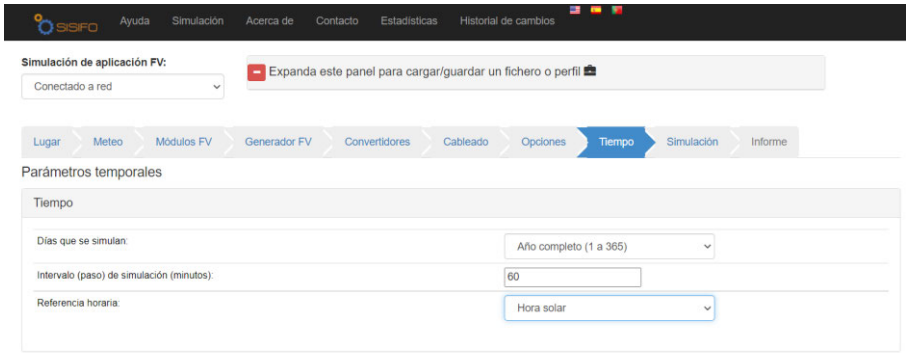

Figura 15. - Configuración de parámetros temporales

Del archivo obtenido de la simulación, se extraen las variables que recoge la **¡Error! No se encuentra el origen de la referencia.**.

Tabla 2. - Variables archivo sisifo_data.xlsx

Variable	**Descripción**
Horizontal	Radiación solar sobre un ángulo de 90º con respecto al panel fotovoltaico [kWh/m2]
Incidente	Radiación solar incidente con el ángulo óptimo de inclinación de panel fotovoltaico [kWh/m2]
Effective with total shadows and spectrum	Radiación solar efectiva sobre el panel fotovoltaico [kWh/m2]

Ambos archivos se cargan en el entono *JupyterLab* y se organizan como dos *dataframes*. Un *dataframe* es una estructura de datos donde cada fila corresponde a un objeto de la muestra y cada columna a una variable. Esta estructura de datos es ofrecida a través de la librería de Python *Pandas*. Su uso es frecuente ya que admite diferentes tipos de variables de entrada. De esta forma se organizan en el entorno los *dataframes, df_site*, correspondiente a los datos de la estación base, y *data_SISIFO*, correspondiente a los de la simulación.

5.1.2 Preprocesamiento de los datos

De forma inicial, cada variable tiene un formato diferente. Para poder realizar operaciones de filtrado, transformación y reducción es necesario transformar las variables a un formato adecuado para que pueda ser entendido por el modelo.

```
df_site.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8334 entries, 0 to 8333
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   FECHA          8334 non-null   object
 1   CHARGE.BAT     8217 non-null   float64
 2   POT.BAT        8217 non-null   float64
 3   STATUS.BAT     8217 non-null   object
 4   POT.GE         5234 non-null   object
 5   FUEL.GE        8217 non-null   object
 6   TIME.GE        8217 non-null   float64
 7   GE (ON/OFF)    8217 non-null   object
 8   POT.SOLAR      4159 non-null   object
 9   STATUS.SOLAR   8217 non-null   object
 10  POT.TOTAL      8217 non-null   float64
 11  TEMPERATURA    8217 non-null   object
dtypes: float64(4), object(8)
memory usage: 781.4+ KB
```

Figura 16. - Formato inicial de variables *df_site*

Los datos recogidos de la estación base se sitúan entre el 01/09/2020 a las 00:00 hasta el 01/09/2021 a las 23:00. Los datos de la simulación están ordenados desde de 01/01 00:00 de un año natural, hasta el 31/12 23:00. Del conjunto de las tres variables que muestra el archivo de simulación, se ha seleccionado la 3ª, *Effective with total shadows, and spectrum*, debido a que es la información que mejor recoge valores reales de radiación.

Para poder unificar en un único *dataframe* ambos archivos, es necesario ordenar los datos.

- Se transforma en *df_site* la fecha recogida por su correspondiente hora ordinal del año, es decir, los valores de fecha irán: de la 0 - 8783 (24*366[7]). Para ello, se añade una nueva columna **OrdinalDate**.
- Se añade la columna de radiación, **RAD_SIMULATE**, en su fecha correspondiente, manteniendo el orden de 1/09/2020 00:00 - 1/09/2021 23:00 de *df_site.*

Una vez recogidas y ordenadas todas las variables en un solo *dataframe*, *df_site_v0*, se procede a la limpieza de los datos.

[7] El año 2020 es un año bisiesto, por lo que hay que tener en cuenta que el archivo tiene 24 entradas adicionales. Como la simulación se hace de un año natural, se han añadido 24 horas adicionales realizando la media aritmética del día correspondiente al 28 de febrero y el 1 de marzo.

5.1.2.1 Eliminar fallos

Como se mencionó en el apartado 4.2.2.1, el conjunto de datos presenta valores *NaN* además de errores señalados por el monitor como '- '.

```
df_site_v0.isnull().sum()

CHARGE.BAT       117
POT.BAT          117
STATUS.BAT       117
POT.GE          3100
FUEL.GE          117
TIME.GE          117
GE (ON/OFF)      117
POT.SOLAR       4175
STATUS.SOLAR     117
POT.TOTAL        117
TEMPERATURA      117
OrdinalDate        0
RAD_SIMULATE       0
dtype: int64
```

Figura 17. - Conjunto de valores NaN

Como se ve en la Figura 17 el conjunto presenta de forma inicial un total de 8328 celdas con valor *NaN*. Estos valores son sustituidos por 0 o por un valor medio, según la naturaleza del error, hasta eliminar todos estos valores.

```
df_site_v0.isnull().sum()

CHARGE.BAT      0
POT.BAT         0
STATUS.BAT      0
POT.GE          0
FUEL.GE         0
TIME.GE         0
GE (ON/OFF)     0
POT.SOLAR       0
STATUS.SOLAR    0
POT.TOTAL       0
TEMPERATURA     0
OrdinalDate     0
RAD_SIMULATE    0
dtype: int64
```

Figura 18. - Conjunto de valores *NaN* una vez eliminados los fallos

De forma análoga, se sustituyen los valores '- 'clasificados como fallos de comunicación puntual.

Una vez eliminados los errores del *dataframe,* la matriz actual está formada por un conjunto de 8216 muestras, frente a las 8334 iniciales. Después, se transforman

las variables al formato adecuado. En el caso de las variables cuantitativas, al eliminar los fallos mencionados, estas variables toman un formato final del tipo *float* o *int*. En el caso de las variables cualitativas, este tipo de datos suelen ser problemáticos para la gran mayoría de algoritmos de machine learning ya que los algoritmos se fundamentan en expresiones matemáticas que requieren operar con las características de las muestras. Por lo tanto, cuando se quiere realizar el producto de un dato que representa, por ejemplo, "CHARGE" con uno que representa "DISCHARGE" sencillamente es imposible, ya que no existen operaciones matemáticas definidas para esos valores. Para estos valores, ON-OFF, CHARGE-DISCHARGE, se han asignado los valores 1 para ON y CHARGE y valores 0 para OFF-DISCHARGE.

La Figura 19 recoge el formato final de cada variable.

```
df_site_v0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8216 entries, 0 to 8333
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   CHARGE.BAT    8216 non-null   float64
 1   POT.BAT       8216 non-null   float64
 2   STATUS.BAT    8216 non-null   int64
 3   POT.GE        8216 non-null   int64
 4   FUEL.GE       8216 non-null   float64
 5   TIME.GE       8216 non-null   float64
 6   GE (ON/OFF)   8216 non-null   int64
 7   POT.SOLAR     8216 non-null   float64
 8   STATUS.SOLAR  8216 non-null   int64
 9   POT.TOTAL     8216 non-null   float64
 10  TEMPERATURA   8216 non-null   float64
 11  OrdinalDate   8216 non-null   int64
 12  RAD_SIMULATE  8216 non-null   float64
dtypes: float64(8), int64(5)
memory usage: 1.2 MB
```

Figura 19. - Formato final de las variables

5.1.2.2 Eliminar outliers

Como se ha comentado en el apartado 4.2.2.2, los *outliers* se visualizan a través de los métodos *box plot* y *scatter plot.* Si de forma gráfica se detectan estos outliers, se realiza el cálculo IQR para sustituir estos valores. Este proceso se realiza de forma individual en cada variable. A continuación, se muestra un ejemplo de este proceso en una variable. El resto de los gráficos de las diferentes variables se encuentran en los anexos.

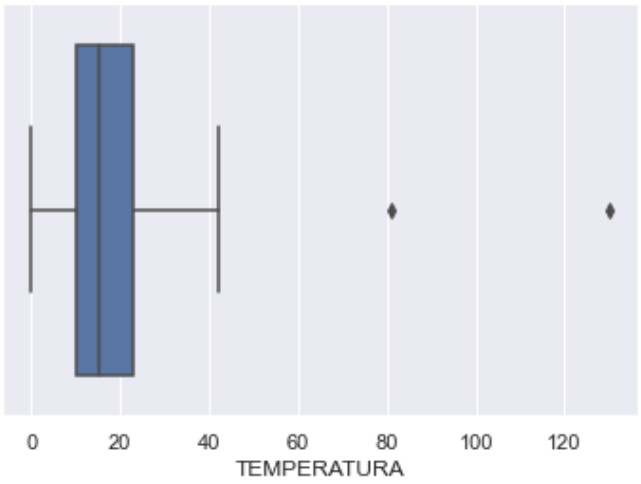

Figura 20. - *BoxPlot* de la variable temperatura

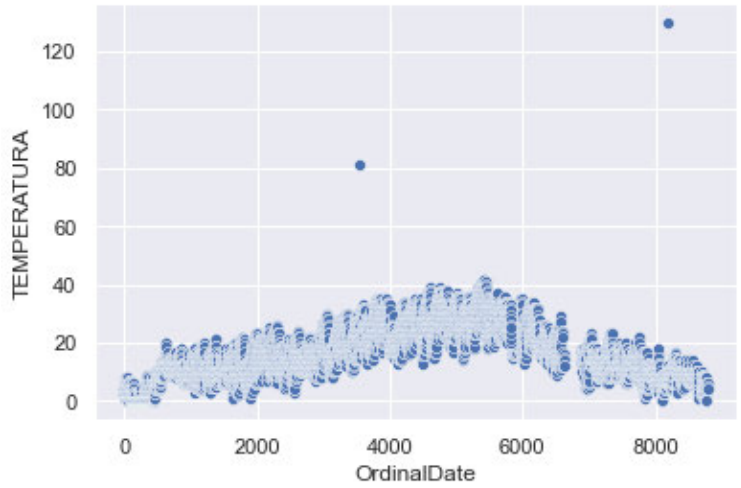

Figura 21. - *Scatterplot* de la variable temperatura en función de la hora ordinal del año

Como se observa en la Figura 20 y Figura 21 existen dos puntos que presentan una desviación significativa. Se ha realizado el cálculo IRQ, siendo el rango de valores donde no se considera outlier [-3, 36]. Como se observa en la Figura 21, durante los meses de verano, *OrdinarDate* entre 4000-6000 aproximadamente, las temperaturas superan en numerosas ocasiones 40ºC. Aunque son valores fuera del intervalo, son valores de temperatura considerados válidos, debido a las temperaturas alcanzadas en estos meses en la Comunidad de Madrid, por lo que para eliminar los outliers, se han filtrado para valores superiores a 45ºC. Los demás valores se consideran errores en la sonda de temperatura y son sustituidos por la media del valor anterior y posterior.

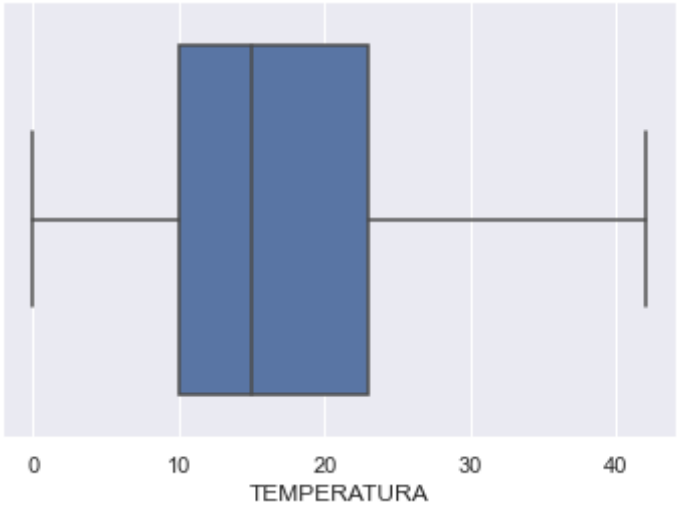

Figura 22. - *Boxplot* de la variable temperatura tras eliminar los outliers

De la misma forma que valores de temperatura tan elevados como 100ºC son errores de los equipos que no pueden ocurrir en circunstancias normales, valores como 42ºC, que se encontraban fuera del rango IRQ, son válidos. Por esta razón, se ha realizado un filtrado exhaustivo de los valores en cada variable, determinando si cada desviación presente puede ser o no real, y si puede ser o no interesante para el modelo.

5.1.2.3 Selección de variables

Con el objetivo de reducir las dimensiones del *dataframe*, se visualiza la matriz de correlación en la Figura 23.

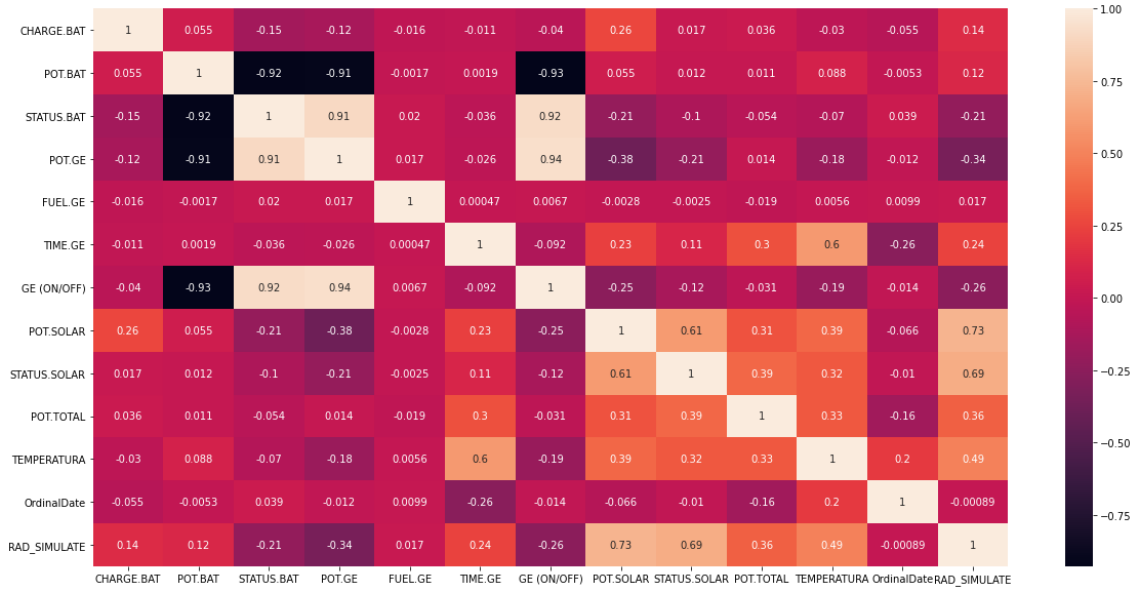

Figura 23. - Matriz de correlación inicial

De forma inicial, el conjunto está formado por 13 variables. El código de colores de la matriz permite realizar un análisis de la correlación de las variables. Se observa que:

- Presentan una alta correlación POSITIVA:
  - STATUS BAT – POT.GE
  - STATUS BAT – GE(ON/OFF)
  - POT.GE – GE(ON/OFF)
  - STATUS SOLAR – POT SOLAR
  - STATUS SOLAR – RAD_SIMULATE

- Presentan una alta correlación NEGATIVA:
  - STATUS BAT – POT.BAT
  - POT.BAT – POT.GE
  - POT.BAT – GE(ON/OFF)

Por el contrario, se observa que la variable en la que se encuentra menos correlación y por tanto es la más independiente del *dataframe* es FUEL.GE.

Tras el análisis, se eliminan las variables STATUS BAT, STATUS SOLAR, POT.GE, por ser las variables con mayor correlación y que, por tanto, tras eliminarlas no se elimina información significativa para el modelo.

La matriz de correlación tras eliminar estas variables se muestra en la Figura 24.

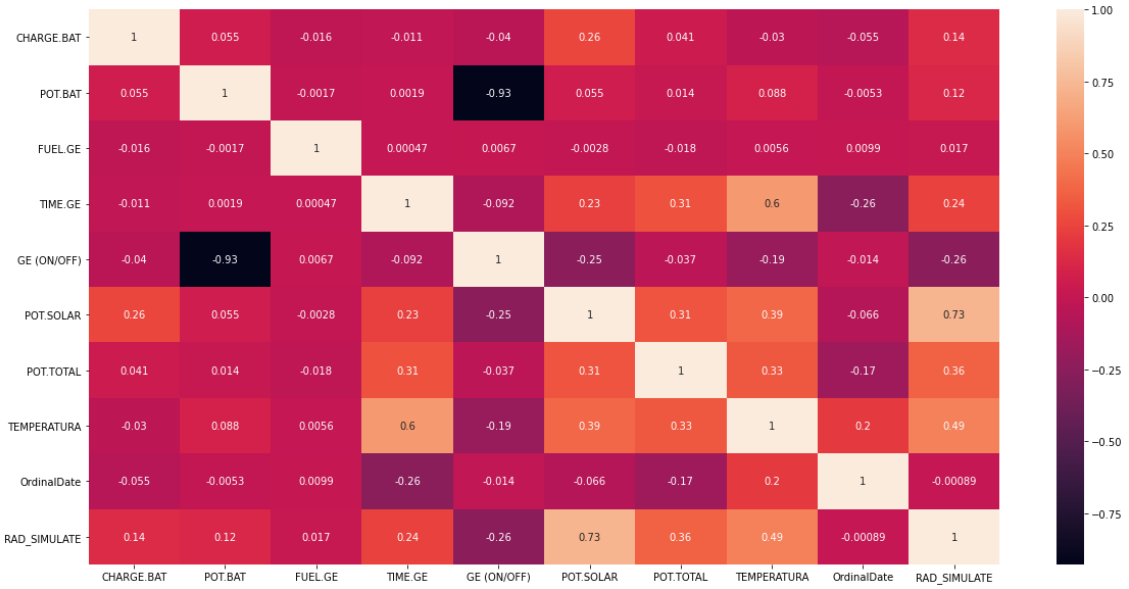

Figura 24. - Matriz de correlación final

5.1.2.4 Transformación de datos

La escala de valores de cada variable se mueve en un rango en función de su naturaleza. En la Figura 25 se representa mediante un diagrama de barras los valores máximos que alcanza cada variable.

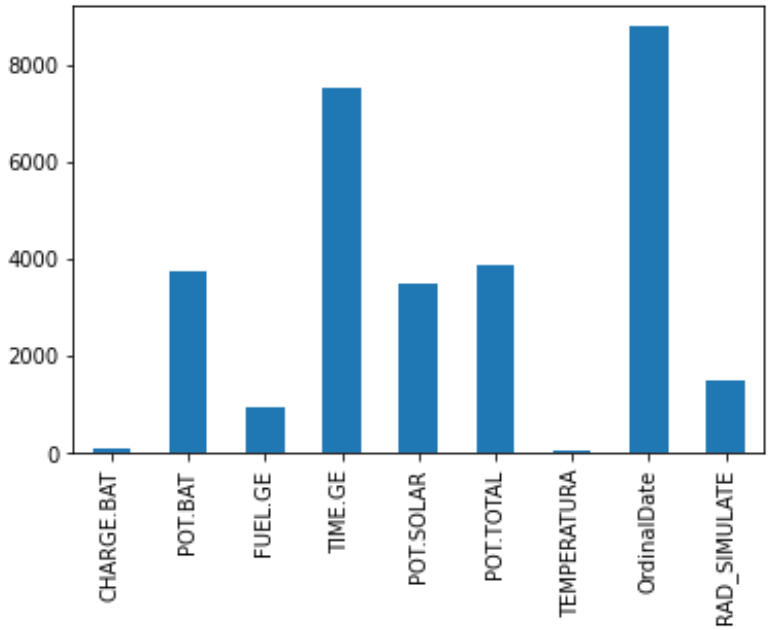

Figura 25. - Valores máximos de cada variable

Se puede observar que los valores máximos de las variables sufren una gran desviación unos sobre otros. Se realiza la normalización mencionada en el

apartado 4.2.2.4 al conjunto de datos. En la Figura 26 y Figura 27 se muestra una comparativa de un conjunto de muestras del *dataframe* anterior y posterior al normalizado.

	CHARGE.BAT	POT.BAT	FUEL.GE	TIME.GE	POT.SOLAR	POT.TOTAL	TEMPERATURA	OrdinalDate	RAD_SIMULATE
0	52	-579	585.1	0	0.0	2921.0	16.0	5856	0
1	59	-487	584.2	1	0.0	3113.0	16.0	5857	0
2	67	-549	582.3	2	0.0	2851.0	16.0	5858	0
3	75	-589	580.4	3	0.0	2811.0	15.0	5859	0
4	82	-601	579.4	4	0.0	2799.0	15.0	5860	0

Figura 26. - Conjunto de muestras del df antes de la normalización

	CHARGE.BAT	POT.BAT	FUEL.GE	TIME.GE	POT.SOLAR	POT.TOTAL	TEMPERATURA	OrdinalDate	RAD_SIMULATE
0	0.477273	0.245879	0.530678	0.000000	0.0	0.275319	0.380952	0.666743	0.0
1	0.556818	0.261843	0.529518	0.000133	0.0	0.419355	0.380952	0.666856	0.0
2	0.647727	0.251085	0.527069	0.000266	0.0	0.222806	0.380952	0.666970	0.0
3	0.738636	0.244144	0.524620	0.000399	0.0	0.192798	0.357143	0.667084	0.0
4	0.818182	0.242061	0.523331	0.000533	0.0	0.183796	0.357143	0.667198	0.0

Figura 27. - Conjunto de muestras del df después de la normalización

5.1.3 Procesamiento de los datos

La variable objetivo de los modelos (en inglés *Target*) será la columna que indica ON/OFF del GE. El resto de las variables serán las características (en inglés *features*), que se comportarán como predictores para la clasificación del estado de funcionamiento de GE.

5.1.3.1 Decision Tree

Para el entrenamiento de Árboles de Decisión se ha utilizado el modelo *DecisionTreeClassifier*[8]*()*, incluido en la librería *Scikit-Learn*. Este modelo tiene numerosos hiperparámetros que pueden ser configurados. A continuación, se describen aquellos parámetros que han sido de interés para el entrenamiento.

- **Criterion {"gini", "entropy"}:** función para medir la calidad de una división. Ambos criterios explicados en el apartado 2.6.1.
- **max_depth**: profundidad máxima del árbol.

[8] En los anexos de los códigos Phyton se encuentran todas las librerías que se deben importar para la utilización de este y los demás modelos.

- **min_samples_split:** número mínimo de muestras necesarias para dividir un nodo interno.
- **random_state:** control de la aleatoriedad del estimador. Este parámetro se ha añadido en todos los modelos con el fin de poder reproducir siempre el mismo escenario.

Para el entrenamiento del modelo, se han realizado diferentes ajustes de estos hiperparámetros. Primero se han ajustado de forma manual, introduciendo valores aleatorios, y después con técnicas de ajuste automático. Para esta última prueba se ha hecho uso de la clase *GridSearchCV* incluido en *Scikit-Learn*, con la que se realiza una búsqueda exhaustiva de valores de parámetros especificados para un estimador. Los parámetros del estimador utilizados se optimizan mediante una búsqueda con validación cruzada sobre una cuadrícula de parámetros. Esta búsqueda implementa un método de "ajuste" y "puntuación". Aunque el análisis individual de los hiperparámetros es útil para entender su impacto en el modelo e identificar rangos de interés, la búsqueda final no debe hacerse de forma secuencial, ya que cada hiperparámetro interacciona con los demás. Es más efectivo la utilización de *GridSearch* para analizar varias combinaciones de estos. Con el objetivo de identificar la profundidad óptima que consigue reducir la varianza y aumentar la capacidad predictiva del modelo, se ha sometido al árbol a un proceso de *pruning* con esta técnica. Así, se deja crecer el árbol al máximo posible identificando los valores óptimos de crecimiento, y posteriormente se predice el modelo con los valores identificados.

La Tabla 3 muestra los resultados de las métricas de evaluación obtenidas a partir de la matriz de confusión. En los anexos se encuentran representadas todas las matrices de confusión.

Tabla 3. - Resultados del entrenamiento de DecisionTreeClassifier

	Accuracy Train	**Accuracy Test**	**F1 Score**
Ajuste Manual	99.93%	99.8175%	99.8174%
Ajuste por *GridSearchCV*	100%	99.8781%	99.8779%

Como se observa en la Tabla 3 los resultados obtenidos con ajustes de *GridSearch* tienen mayor precisión que con el ajuste manual, aunque queda reflejado que el árbol ha crecido lo suficiente como para ajustarse de forma total a las muestras utilizadas durante el entrenamiento, obteniendo una precisión del 100% en este proceso.

A continuación, se muestra en la Figura 28 el árbol final obtenido.

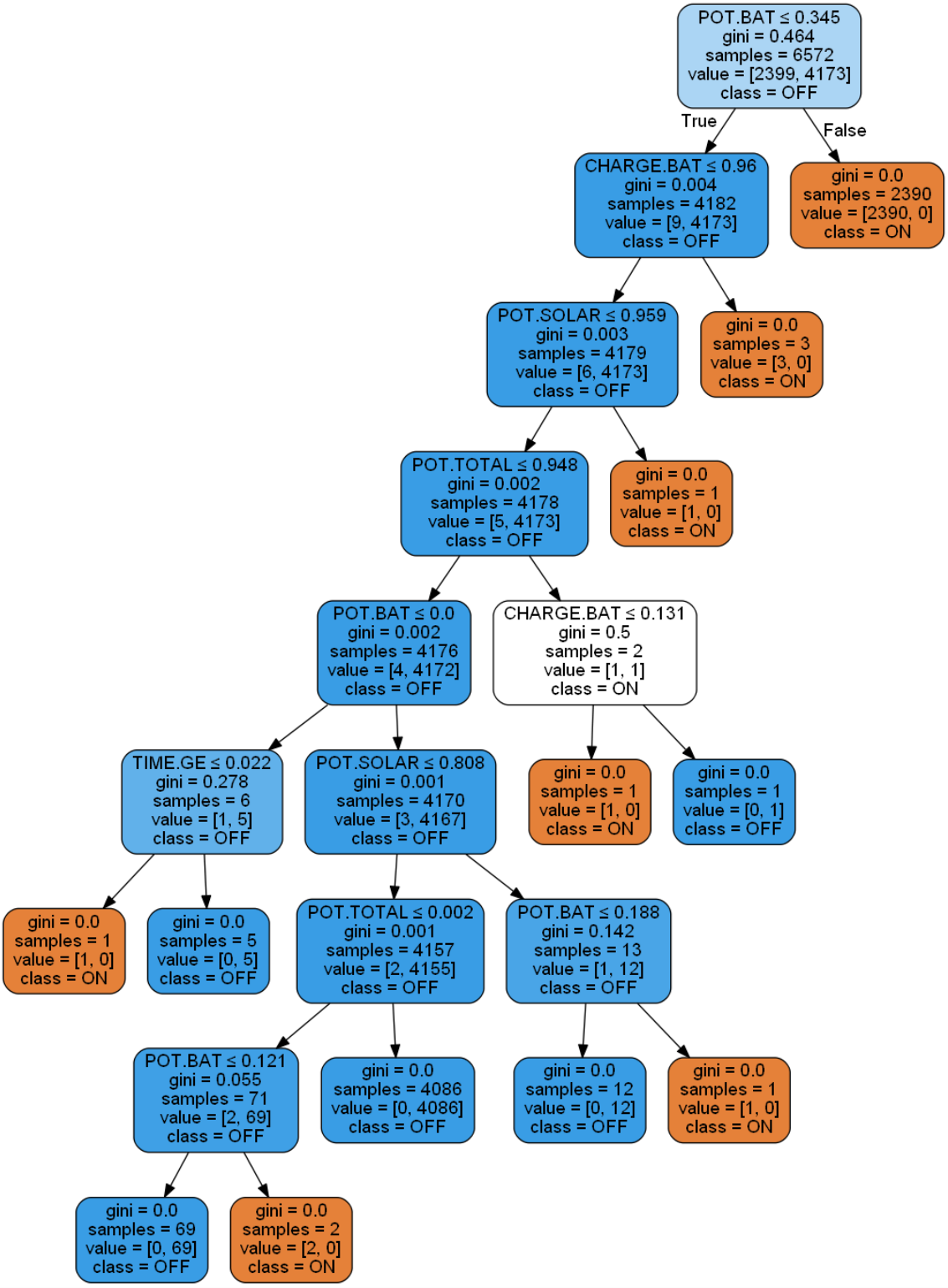

Figura 28. - Árbol de decisión final obtenido por *GridSearchCV*

Con la técnica *GridSearch,* se ha determinado que el modelo presenta las mejores predicciones con ciertos valores de algunos de sus hiperparámetros: profundidad máxima (*max_depth*) de 8 y con el criterio de división (*criterion*) *Gini,* obteniendo

un árbol final con un conjunto de 11 nodos terminales para clasificar el conjunto de muestras.

Para la mejor interpretación de este árbol, se describen algunos aspectos:

- Cada color representa una clase. Naranja ->ON, Azul ->OFF
- El color es más intenso cuanto más seguro está el modelo que la clasificación es correcta. Así el primer nodo, *root,* tiene un azul menos intenso, reflejando que es menos segura la clasificación, y blanco cuando son igualmente probables.
- Información de cada nodo:
  - *samples*: número de muestras que satisfacen las condiciones necesarias para llegar a este nodo.
  - *value*: número de muestras de cada clase que llegan a cada nodo.
  - *class*: clase que les asigna a las muestras que llegan a cada nodo.

A partir de la representación del árbol, se pueden identificar las variables que se evalúan en cada nodo para la división de las muestras que se quieren clasificar. Este modelo, además, permite calcular la importancia total que tienen cada uno de los predictores en el modelo. Así en la Figura 29 se puede observar que con más de un 99% de importancia, la variable *POT.BAT* es el predictor que mayor relevancia tiene en la predicción del estado de funcionamiento del GE. Y que, predictores como *TIME.GE, FUEL.GE, OrdinalDate* y *Temperatura* no intervienen en la predicción, con importancia 0.

Importancia de los predictores en el modelo

	predictor	importancia
1	POT.BAT	0.996720
0	CHARGE.BAT	0.001967
4	POT.SOLAR	0.000656
8	RAD_SIMULATE	0.000329
5	POT.TOTAL	0.000328
2	FUEL.GE	0.000000
3	TIME.GE	0.000000
6	TEMPERATURA	0.000000
7	OrdinalDate	0.000000

Figura 29. - Importancia de los predictores

Esta información, comparándola con el árbol de la Figura 28, determina que, durante el primer nodo de división, de las 6572 muestras se han clasificado 2390 muestras en ON, en función de la variable *POT.BAT* con un valor de Gini de 0.0, es decir, un error de clasificación de 0. Para el resto de las muestras, la clasificación ha sido en función de diferentes variables del *dataframe*, con una importancia muy baja. Estos datos explican la alta precisión de los modelos obtenidos para la clasificación del conjunto de datos, además de aportar información muy relevante para entender el funcionamiento del GE dentro del sistema híbrido, ya que ha quedado confirmada la directa relación del funcionamiento del GE en función de la potencia a la que se encuentren las baterías.

5.1.3.2 Random Forest

Para el entrenamiento de *Random Forest* se ha utilizado el modelo *RandomForestClassifier(),* incluido en la librería *Scikit-learn.* Este método tiene numerosos hiperparámetros que se pueden configurar y la mayoría son relativos a hiperparámetros del árbol, por lo que algunos son los mismos mencionados en el apartado anterior.

- **n_estimators:** número de árboles del bosque.
- **criterion {"gini", "entropy"}:** función para medir la calidad de la división.
- **max_depth**: profundidad máxima de cada árbol del bosque.

- **min_samples_split:** número mínimo de muestras necesarias para dividir un nodo interno.
- **random_state:** control de la aleatoriedad del estimador.

De igual forma que en DT, se han ajustado de forma manual, introduciendo valores aleatorios y por defecto, y después con técnicas de ajuste automático haciendo uso de *GridSearchCV*.

Tabla 4. - Resultados del entrenamiento de RandomForestClassifier

	Accuracy Train	**Accuracy Test**	**F1 Score**
Ajuste Manual	100%	99.8475%	99.8472%
Ajuste por *GridSearchCV*	100%	99.8783%	99.8782%

Como se observa en la Tabla 4, con técnicas de ajuste automático, se consigue de nuevo una precisión mayor que con ajuste manual, aunque la diferencia es bastante reducida. Con este clasificador, se buscaba evitar la importancia tan elevada de uno de los predictores reflejada en DT, ya que RF no utiliza todos los predictores en cada árbol, si no que utiliza pequeños conjuntos de predictores diferentes en cada árbol escogidos de forma aleatoria. Aun así, no se ha conseguido evitar una precisión del 100% durante el entrenamiento, de forma que el modelo se ajusta de forma total a las muestras utilizadas durante el entrenamiento.

5.1.3.3 AdaBoost

Para el entrenamiento de AdaBoost se ha utilizado el modelo *AdaBoostClassifier()*, incluido en la librería *Scikit-Learn.* Este método tiene un número de hiperparámetros más reducidos que los anteriores modelos. Estos son:

- **n_estimators:** El número máximo de estimadores en el que se termina el *boosting*. En caso de ajuste perfecto, el procedimiento de aprendizaje se detiene antes.

- **learning_rate:** Peso aplicado a cada clasificador en cada iteración de *boosting*.
- **random_state:** control de la aleatoriedad del estimador.

De igual forma que en los anteriores modelos, se han ajustado de forma manual, introduciendo valores aleatorios, y después con técnicas de ajuste automático haciendo uso de *GridSearchCV*.

Tabla 5. - Resultados del entrenamiento de AdaBoostClassifier

	Accuracy Train	**Accuracy Test**	**F1 Score**
Ajuste Manual	100%	99.8175%	99.8174%
Ajuste por *GridSearchCV*	100%	99.9330%	99.9329 %

Como se observa en la Tabla 5, y comparando estos resultados con los obtenidos por los demás modelos, AdaBoost, consigue el mejor valor de precisión en sus predicciones con un 99.93%. Este modelo, como se explicó anteriormente, es un modelo de *ensemble* que se va ajustando el error cometido de forma secuencial, asignando pesos a las diferentes muestras según su clasificación correcta o incorrecta, por lo que al final del entrenamiento ha conseguido un modelo casi perfecto de predicción. Sin embargo, tampoco consigue evitar un sobreajuste durante el entrenamiento de igual forma que ha ocurrido con los demás modelos.

5.1.4 Análisis de resultados

La importancia de la interpretabilidad de los modelos de ML es crucial para poder justificar y comprender las predicciones y/o resultados obtenidos. En ocasiones, muchos de los modelos de ML muestran resultados que no explican sus predicciones de una manera sencilla. En cambio, si se utilizan modelos intrínsecamente interpretables, éstos proporcionan sus propias explicaciones, que son fieles a lo que el modelo realmente calcula [38]. De igual forma, es importante entender el funcionamiento real de los sistemas donde se pretende dar uso a estos modelos para poder comparar los resultados obtenidos.

Atendiendo a los resultados de precisión obtenidos tras el entrenamiento y ajuste de los hiperparámetros de los diferentes modelos de ML utilizados, se observan valores muy altos de precisión, incluso llegando en algunos casos a valores de un 100%. Para dar respuesta a estos resultados se ha realizado un análisis funcional sobre el sistema para poder realizar una comparativa entre el funcionamiento real y el funcionamiento predicho por los algoritmos de ML.

Se ha realizado la consulta de esquemas eléctricos, hojas de características y de más documentación técnica del sistema y de cada uno de sus componentes. Se ha comprobado que el funcionamiento electrónico de control del GE viene determinado por un parámetro de configuración en uno de los equipos, relativo al *SOC* de las baterías. Este valor es directamente proporcional a la potencia de estas. Es decir, la acción de arranque o parada del grupo viene determinado por el valor de la potencia de las baterías. Como se determinó en el apartado 5.1.3.1 y se puede ver en la Figura 29, DT determinó la importancia de los predictores, asignando un valor del 99% de importancia a la potencia de las baterías y realizando una clasificación de más de 2000 muestras en función de un valor concreto de este predictor, revelando la relación directa que encontraba el algoritmo entre este predictor y la variable objetivo. Relación que, finalmente, se ha comprobado que es la encargada del funcionamiento real del sistema.

La comparativa de ambos análisis funcionales determina que los algoritmos de ML utilizados en el marco de este proyecto son altamente eficaces en el modelado automático de algoritmos a través de los datos de entrada. Además, los altos valores de precisión obtenidos se justifican a través de la sencillez del algoritmo de control real de funcionamiento y, por tanto, el alto acoplamiento con los diferentes modelos de ML entrenados a partir de los datos de entrada proporcionados del sistema.

Capítulo 6

6 PRESUPUESTO

Para la realización de este proyecto se han empleado herramientas de código abierto (en inglés *Open-Source*), y gracias a los convenios que posee la Universidad Politécnica de Madrid, la licencia del paquete Microsoft Office es gratuita para estudiantes de la escuela. Por lo tanto, no existe ningún tipo de coste asociado a la distribución de software.

Respecto a los costes hardware, solo se tiene él cuenta el coste del ordenador portátil. Este coste se ha calculado como un coste asociado a la amortización durante el periodo de realización de proyecto [39].

$$Coste\ de\ amortización = \frac{Coste\ total \cdot Periodo\ de\ uso}{Vida\ util} \tag{19}$$

Para un ordenador portátil, se estima un tiempo de vida útil de 4 años. Para este cálculo, se tiene en cuenta que el número de horas totales utilizadas para la realización del proyecto ha sido de 320h, repartidas en 8h/día. Junto a este, únicamente se añaden los costes derivados del esfuerzo humano invertido en la realización de cada tarea. En la Tabla 6 se recoge un resumen de estos gastos.

Tabla 6. - Presupuesto

Material	Unidades	Precio/Ud.	Precio
Ordenador MSI	1	1200 €	37,00 €
Distribución Anaconda	-	-	0,00 €
Licencia Microsoft Office	-	-	0,00 €
Material de Impresión	-	-	100,00€
Mano de obra	320 h	20 €/h	6.400,00 €
TOTAL			**6537,00 €**

Capítulo 7

7 CONCLUSIONES Y TRABAJO FUTURO

Una vez finalizado el proyecto, se procede a analizar el grado de cumplimiento de los objetivos planteados y extraer las conclusiones pertinentes.

La realización del proyecto ha resultado un camino extenso de aprendizaje. En primer lugar, fue necesario estudiar y comprender los aspectos básicos de la IA y ML. Después, identificar las características de cada uno de los tipos de aprendizaje que existían para poder identificar cual podía ser el que cumpliera con los objetivos que se buscaban. Una vez identificado, fue necesario conocer y analizar las herramientas que permitieran el desarrollo de la solución. El último paso de este aprendizaje fue conocer los aspectos de la metodología que siguen los proyectos de análisis de datos, ya que no coinciden con la metodología de software tradicional aprendida durante los estudios de grado. En ocasiones, este aprendizaje ha resultado complicado y costoso, pero ha resultado una experiencia enriquecedora y de gran interés y actualidad.

Analizando los resultados obtenidos, se puede afirmar que los objetivos propuestos se han cumplido en un grado muy satisfactorio, cumpliendo con las especificaciones y restricciones definidas para el proyecto. Se ha concluido que el entrenamiento de los modelos seleccionados ha aportado información relevante del funcionamiento del sistema híbrido, identificando las relaciones de las variables de funcionamiento. Respecto a los modelos, DT ha revelado de la forma más clara y sencilla esta información del sistema, pero AdaBoost ha conseguido los mejores resultados de precisión.

Sin embargo, aunque los objetivos del proyecto han sido conseguidos, existen posibles líneas futuras de trabajo. La gran variedad de modelos de aprendizaje disponibles, permitirían analizar el sistema desde diferentes perspectivas:

- Manteniendo el aprendizaje como un problema de clasificación, podrían probarse diferentes modelos a los utilizados en el proyecto.
- Los algoritmos estudiados también tienen la posibilidad de usarse para modelos de regresión lineal. Se propone modificar la variable objetivo para resolver un problema de estas características, utilizando los mismos datos para diferentes variables objetivo.

- Sería interesante, analizar la división de los datos y el entrenamiento de estos en función del tiempo para estas nuevas variables objetivo, como puede ser la radiación solar o la potencia de los equipos.

Capítulo 8

8 REFERENCIAS BIBLIOGRÁFICAS

8.1 Referencias

[1] M. D. Calvo-Flores, «La Inteligencia Artificial,» *Universidad de Granada*, 1997.

[2] A. M. Turing, «Computing Machinery and Intelligence,» Mind, New Series, Vol. 59, No. 236 (Oct., 1950), pp. 433-460.

[3] P. Castro, «Computing Machinery, Intelligence and Undecidability,» *J Theor Comput Sci 4: 160. doi:10.4172/2376-130X.1000160.*

[4] J. McCarthy, «WHAT IS ARTIFICIAL INTELLIGENCE?,» http://www-formal.stanford.edu/jmc, Computer Science Department Stanford University, 2007 Nov 12.

[5] M. M. .. I. E. N. I. L. R. M. M. El Naqa I., « What Is Machine Learning?,» de *Machine Learning in Radiation Oncology*, (2015), pp. https://doi.org/10.1007/978-3-319-18305-3_1.

[6] Z. XD., Machine Learning. In: A Matrix Algebra Approach to Artificial Intelligence, Springer, Singapore. https://doi.org/10.1007/978-981-15-2770-8_6 , (2020).

[7] S. B. Kotsiantis, «A review of classification techniques,» de *Supervised machine learning*, (2007), p. 249–268.

[8] E. Alpaydın, Introduction to Machine Learning, 2º Edition, Cambridge: The MIT Press, 2020.

[9] T. Hofmann, «learning by probabilistic latent semantic analysis,» de *Machine Learning,*, 2001, p. 177–196.

[10] C. T. A. Z. Daxin Jiang, «Cluster Analysis for Gene Expression Data: A Survey,» Department of Computer Science and Engineering, State University of New York at Buffalo, 2004.

[11] J. Arroyo-Hernández, «Métodos de reducción de dimensionalidad: Análisis comparativo de los métodos APC, ACPP y ACPK,» *UNICIENCIA* , Vols. %1 de %2Vol. 30, No. 1, pp. 115-122., 2016.

[12] M. L. L. A. W. M. L. P. Kaelbling, «Reinforcement Learning: A Survey».*Journal of Artificial Intelligence Research, Vol 4, (1996).*

[13] F. H. M. S. Mohammad Hossein Dehghan, «Study of Linear Regression Based on Least Squares and Fuzzy Least Absolutes Deviations andits Application in Geography,» University of Sistan and Baluchestan, 2015.

[14] M. K. ·. K. Johnson, Applied Predictive Modeling, Springer.

[15] K. Kim, «A hybrid classification algorithm by subspace partitioning throughsemi-supervised decision tree,» *Seoul National University of Science & Technology,* December 2015.

[16] C. E. Shannon, «A mathematical theory of communication,» *The Bell System Technical Journal,* vol. vol. 27, nº 3, July 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[17] J. F. R. O. a. C. S. L. Breiman, «Classification and Regression Trees,» *Wadsworth International Group,* Belmont, CA (1984).

[18] E. Acuna, «Arboles de Decision,» de *Mineria de Datos*, Universidad de Puerto Rico-Mayaguez.

[19] L. Breiman, «RANDOM FORESTS,» *University of California Berkeley,* 2001.

[20] B. L, «Bagging Predictors,» de *Machine Learning, Vol 24(2)*, 1996, pp. pp 123-140..

[21] Y. S. Robert E. Schapire, «Improved Boosting Algorithms Using Confidence-rated Predictions,» 1996.

[22] «What is the difference between Bagging and Boosting?,» *ETS Asset Management Factory,* 2016.

[23] Phyton, «Phyton Offical Web,» [En línea]. Available: https://www.python.org/.

[24] NumPy, «NumPy Official Web,» [En línea]. Available: https://numpy.org/.

[25] Pandas, «Pandas Official Web,» [En línea]. Available: https://pandas.pydata.org/.

[26] Matplotlib, «Matplotlib Official Web,» [En línea]. Available: https://matplotlib.org/.

[27] seaborn, «Seaborn Official Web,» [En línea]. Available: https://seaborn.pydata.org/index.html.

[28] Scikit-learn, «Scikit-learn Official Web,» [En línea]. Available: https://scikit-learn.org/.

[29] Jupyter, «Jupyter Official Web,,» [En línea]. Available: https://jupyter.org/.

[30] S. R. H.-A. I. C.-Z. S. J. Timarán-Pereira, El proceso de descubrimiento de conocimiento en bases de datos, En Descubrimiento de patrones de desempeño académico con árboles decisión en las competencias genéricas de la formación profesionales(pp. 63-86), 2016.

[31] R. B. T. S. Dipanjan Sarkar, Practical Machine Learning with Python, APRESS.

[32] E. E. Houby, «A survey on applying machine learning techniques for management of diseases,» *J Appl Biomed,* pp. vol. 16, pp. 165-74, 2018 .

[33] J.-F. E.-C. Juan Hernández-Lalinde, «Sobre el uso adecuado del coeficiente de correlación de Pearson: definición, propiedades y suposiciones,» *Universidad Simón Bolívar (Colombia),* December 2018.

[34] W. Sarle, «Neural network,» *Periodic posting to the Usenet newsgroup comp.ai.neural-nets,* . (Ed.) (1998). .

[35] D. Larose, «Descubriendo el conocimiento en los datos: una introducción a la minería de datos.,» 2014.

[36] L. T. L. PAYAM REFAEILZADEH, Cross-Validation, Arizona State University.

[37] F. Joanneum, «Cross-Validation Explained.,» (2005-2006).

[38] C. Rudin, «Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and UseInterpretable Models Instead».*Duke University.*

[39] P. f. P. S. a. M. L. J. Unpingco, USA: Springer, 2019.

[40] T. Hastie, R. Tibshirani y J. Friedman, The Elements of Statistical Learning, Springer, 2009.

[41] M. TM, Machine learning, New York: McGraw-Hill, 1997.

[42] D. A.-J. J. M. A. H. A. J. A. Mohamed Alloghani, «A Systematic Review on Supervisedand Unsupervised Machine LearningAlgorithms for Data Science,» 2020, pp. https://doi.org/10.1007/978-3-030-22475-2_1.

[43] Y.-Y. Song, «Decision tree methods: applications for classification and prediction,» *Shanghai Archives of Psychiatry 27(2):130-5,* April 2015.

Capítulo 9

9 ANEXOS

Anexo 1. Código Python "Preprocesado.py"

```
#Carga de ficheros
#fichero de site
import pandas as pd
import numpy as np
filename = 'DATOS_SITE_ML.xlsx'
data_site = pd.read_excel(filename, header=0)

df_site.info()

#Carga de ficheros
#fichero de SISIFO
filename = 'sisifo_data.xlsx'
data_sisifo = pd.read_excel(filename, header=0)

# Se eliminan las columnas y se queda únicamente la 3º
data_sisifo_t = data_sisifo.transpose()
data_sisifo_t=data_sisifo_t.drop([1], axis=1)
data_sisifo_t=data_sisifo_t.drop([2], axis=1)
data_sisifo_t=data_sisifo_t.drop([4], axis=1)
data_sisifo_t=data_sisifo_t.drop(0,axis=0)
data_sisifo_t[3] = data_sisifo_t[3].apply(lambda x: x*1000)

# Cambio de formato de fecha a hora ordinal
from datetime import datetime
from datetime import date

fecha_ordinal = []
for i in range(len(df_site)):
    fecha_dt = pd.to_datetime(df_site.iloc[i, 0], format='%d-%m-%Y
%H:%M')
    fecha_ordinal.append((date.toordinal(fecha_dt) -
date(fecha_dt.year, 1, 1).toordinal())*24+fecha_dt.hour)
df_site = df_site.assign(OrdinalDate = fecha_ordinal)


df_site_v0=df_site.drop(['FECHA'], axis=1)
df_site_v0.info()

radiacion = []
for i in range(len(df_site_v0)):
    valor=int(df_site_v0.iloc[i,11])
    radiacion.append(data_sisifo_t.iloc[valor,1])

df_site_v0 = df_site_v0.assign(RAD_SIMULATE = radiacion)
df_site_v0.isnull().sum()

# Búsqueda de fallos
fallo=False
pos_inicial=0
pos_final=0
drop_fallos=[]
for i in range(len(df_site_v0)):
    if np.isnan(df_site_v0.iloc[i, 9]):
        if pos_inicial<i and pos_final>=i:
            break
        else:
            for j in range(len(df_site_v0)):
                pos_inicial=i
```

```
                drop_fallos.append(i+j)
                if np.isnan(df_site_v0.iloc[i+j,9])==False:
                    pos_final=i+j
                    fallo = True
                    print(i, j)
                    break
print('El fallo comienza en la posicion ', pos_inicial, ' y acaba en
la posicion ', pos_final)
df_site_v0=df_site_v0.drop(drop_fallos)
df_site_v0.isnull().sum()

# BUSQUEDA DE FALLOS POR POTENCIA NULA EN LOS EQUIPOS:
# Se rellenan con valores 0
df_site_v0['POT.GE']=df_site_v0['POT.GE'].fillna(0)
df_site_v0['POT.SOLAR']=df_site_v0['POT.SOLAR'].fillna(0)
df_site_v0.isnull().sum()

# BUSQUEDA DE FALLOS PUNTUALES POR COMUNICACIÓN
dat_err = []
for i in range(len(data_site)):
    if data_site.iloc[i,0] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(data_site)):
    if data_site.iloc[i,1] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(data_site)):
    if data_site.iloc[i,2] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(data_site)):
    if data_site.iloc[i,3] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,4] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

# Se rellenan por valores medios
dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,4] == '-':

        if df_site_v0.iloc[i-1,4] != '-' and df_site_v0.iloc[i+1,4]
!= '-':
```

```
            df_site_v0.iloc[i,4] = (float(df_site_v0.iloc[i-1,4]) +
float(df_site_v0.iloc[i+1,4]))/2.0

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,4] == '-':
        dat_err.append(i)
df_site_v0 = df_site_v0.drop(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,4] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,5] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,6] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,7] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,7] == '-':

        if df_site_v0.iloc[i-1,7] != '-' and df_site_v0.iloc[i+1,7]
!= '-':
            df_site_v0.iloc[i,7] = (int(df_site_v0.iloc[i-1,7]) +
int(df_site_v0.iloc[i+1,7]))/2.0

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,7] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)
df_site_v0 = df_site_v0.drop(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,8] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)
```

```
dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,9] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,10] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,10] == '-':

        if df_site_v0.iloc[i-1,10] != '-' and
df_site_v0.iloc[i+1,10] != '-':
            df_site_v0.iloc[i,10] = (float(df_site_v0.iloc[i-1,10])
+ float(df_site_v0.iloc[i+1,10]))/2

dat_err = []
for i in range(len(df_site_v0)):
    if df_site_v0.iloc[i,10] == '-':
        dat_err.append(i)
len(dat_err)
print(dat_err)


# FORMATO ADECUADO PARA CADA VARIABLE
df_site_v0['STATUS.BAT']=df_site_v0['STATUS.BAT'].replace({'CHARGE':
1, 'DISCHARGE': 0})

df_site_v0['GE (ON/OFF)']=df_site_v0['GE (ON/OFF)'].replace({'ON':
1, 'OFF': 0})

df_site_v0['STATUS.SOLAR']=df_site_v0['STATUS.SOLAR'].replace({'ON':
1, 'OFF': 0})

df_site_v0['POT.GE'] = df_site_v0['POT.GE'].replace('-', )

df_site_v0['TEMPERATURA']=pd.to_numeric(df_site_v0['TEMPERATURA'],do
wncast='integer' )

df_site_v0['FUEL.GE']=pd.to_numeric(df_site_v0['FUEL.GE'])

df_site_v0['POT.SOLAR']=pd.to_numeric(df_site_v0['POT.SOLAR'])

df_site_v0['RAD_SIMULATE']=pd.to_numeric(df_site_v0['RAD_SIMULATE'])

df_site_v0['POT.GE']=pd.to_numeric(df_site_v0['POT.GE'])
df_site_v0.info()
df_site_v0.to_excel('df_prueba.xlsx')
```

Anexo 2. Código Python "Outliers.py"

```
import pandas as pd
import numpy as np
import seaborn as sns

filename ='df_prueba.xlsx'
data_preprocesado_1= pd.read_excel(filename, header=0)
data_preprocesado= data_preprocesado_1.drop(['Unnamed: 0'], axis=1)

time = data_preprocesado['OrdinalDate']

SOC = data_preprocesado['CHARGE.BAT']
sns.boxplot(x=SOC)
sns.scatterplot(data=data_preprocesado, x=time, y=SOC)


pot_bat = data_preprocesado['POT.BAT']
sns.boxplot(x=pot_bat)

pot_ge = data_preprocesado['POT.GE']
sns.boxplot(x=pot_ge)
sns.scatterplot(data=data_preprocesado, x=time, y=pot_ge)

pot_sol = data_preprocesado['POT.SOLAR']
sns.boxplot(x=pot_sol)
sns.scatterplot(data=data_preprocesado, x=time, y=pot_sol)


q75_ps, q25_ps = np.percentile(pot_sol, [75 ,25])
IQR_ps=q75_ps-q25_ps
min_ps=q25_ps-IQR_ps
max_ps=q75_ps+IQR_ps

#Si tenemos en cuenta, que, en esta columna de datos, más del 50% de
las horas del día la potencia solar tiene un valor 0, es normal que
los valores superiores a 2000 W aparezcan como outliers. El sistema
tiene un conjunto de 10 paneles con una potencia de 440W. Teniendo
en cuenta la mejor situación, es decir, un día de verano sobre las 3
de la tarde, funcionando los paneles a un rendimiento del 90%, la
potencia que obtendrían sería de unos 4000 W, por lo que estos
valores que el diagrama señala como outliers se mantendrán en el set
de datos, ya que son desviaciones que se consideran válidas para el
aprendizaje del modelo.

pot_tot= data_preprocesado['POT.TOTAL']
sns.boxplot(x=pot_tot)
sns.scatterplot(data=data_preprocesado, x=time, y=pot_tot)
q75_pt, q25_pt = np.percentile(pot_tot, [75 ,25])
IQR_pt=q75_pt-q25_pt
min_pt=q25_pt-IQR_pt
max_pt=q75_pt+IQR_pt

#Se rellenan los outliers con valores medios
for i in range(len(data_preprocesado)):
    if data_preprocesado.iloc[i,9] > max_pt or
data_preprocesado.iloc[i,9] < min_pt :
            data_preprocesado.iloc[i,9] =
(float(data_preprocesado.iloc[i-1,9]) +
float(data_preprocesado.iloc[i+1,9]))/2.0
```

```
pot_tot= data_preprocesado['POT.TOTAL']
sns.boxplot(x=pot_tot)

for i in range(len(data_preprocesado)):
    if data_preprocesado.iloc[i,9] < min_pt :
            data_preprocesado.iloc[i,9] = min_pt

pot_tot= data_preprocesado['POT.TOTAL']
sns.boxplot(x=pot_tot)

temp= data_preprocesado['TEMPERATURA']
sns.boxplot(x=temp)
sns.scatterplot(data=data_preprocesado, x=time, y=temp)
q75, q25 = np.percentile(temp, [75 ,25])
IQR_temp=q75-q25
max_temp=q75+IQR_temp
min_temp=q25-IQR_temp

#Se rellenan los outliers con valores medios para temperaturas
mayores de 45ºC
for i in range(len(data_preprocesado)):
    if data_preprocesado.iloc[i,10] > 45:
            data_preprocesado.iloc[i,10] =
(float(data_preprocesado.iloc[i-1,10]) +
float(data_preprocesado.iloc[i+1,10]))/2.0


sns.boxplot(x=temp)
rad= data_preprocesado['RAD_SIMULATE']
sns.boxplot(x=rad)
sns.scatterplot(data=data_preprocesado, x=time, y=rad)

data_preprocesado.to_excel('df_prueba2.xlsx')
```

Anexo 3. Código Python "Seleccion_variables.py"

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

filename ='df_prueba2.xlsx'
data_preprocesado= pd.read_excel(filename, header=0)
data_preprocesado= data_preprocesado.drop(['Unnamed: 0'], axis=1)
mtx_corr=data_preprocesado.corr(method="pearson")

plt.figure(figsize=(20,10))
mtx=sns.heatmap(mtx_corr, annot = True)
plt.savefig("Matriz de correlacion.jpg")

# Eliminamos las variables STATUS BAT, STATUS SOLAR y POT.GE del df.

data_preprocesado= data_preprocesado.drop(['STATUS.BAT'], axis=1)
data_preprocesado= data_preprocesado.drop(['POT.GE'], axis=1)
data_preprocesado= data_preprocesado.drop(['STATUS.SOLAR'], axis=1)
data_preprocesado.to_excel('df_to_proces.xlsx')
```

Anexo 4. Código Python "Procesado.py"

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

filename ='df_to_proces_v2.xlsx'
data_pre_norm= pd.read_excel(filename, header=0)
data_pre_norm= data_pre_norm.drop(['Unnamed: 0'], axis=1)

# La variable objetivo (Target) del modelo será la columna que
indica ON/OFF del grupo electrógeno. Denominamos a X la matriz de
caracteristicas (features) y a Y la variable objetivo

Y = data_pre_norm['GE (ON/OFF)']
X=data_pre_norm.drop(['GE (ON/OFF)'], axis=1)
maximos=X.max()
maximos.plot(kind = 'bar')


import sklearn
from sklearn import preprocessing
min_max_scaler = sklearn.preprocessing.MinMaxScaler()
X[['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE','POT.SOLAR','POT.TOTAL
','TEMPERATURA','OrdinalDate','RAD_SIMULATE']]=
min_max_scaler.fit_transform(X[['CHARGE.BAT','POT.BAT','FUEL.GE','TI
ME.GE','POT.SOLAR','POT.TOTAL','TEMPERATURA','OrdinalDate','RAD_SIMU
LATE']])

#  Árboles de decisión
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from sklearn import metrics
from sklearn.tree import export_graphviz
from pydotplus import graph_from_dot_data
import matplotlib.pyplot as plt
import matplotlib as pltl
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedKFold
from six import StringIO
from IPython.display import Image
import pydotplus

os.environ['PATH'] =
os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\gra
phviz"

X_train, X_test, y_train, y_test = train_test_split(X,
Y,test_size=0.2, random_state=123)
clf = DecisionTreeClassifier(max_depth=5,criterion='gini',
random_state=123)
clf = clf.fit(X_train,y_train)
print(f"Profundidad del árbol: {clf.get_depth()}")
print(f"Número de nodos terminales: {clf.get_n_leaves()}")

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,
```

```
feature_names=['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE','POT.SOLAR
','POT.TOTAL','TEMPERATURA','OrdinalDate','RAD_SIMULATE'],class_name
s=['ON', 'OFF'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
print("Accuracy:", clf.score(X_train,y_train)*100)
print("Accuracy:", metrics.accuracy_score(y_test,
clf.predict(X_test))*100)
print("F1:", metrics.f1_score(y_test, clf.predict(X_test),
average='weighted')*100)

plot_confusion_matrix(clf, X_test, y_test, display_labels=['ON',
'OFF'], cmap=plt.cm.Blues)

plt.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,
clf.predict(X_test),target_names=['ON', 'OFF']))

print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores = pd.DataFrame(
                          {'predictor':
['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE','POT.SOLAR','POT.TOTAL',
'TEMPERATURA','OrdinalDate','RAD_SIMULATE'],
                           'importancia':
clf.feature_importances_}
                          )
importancia_predictores.sort_values('importancia', ascending=False)
nmp=X.to_numpy()
caract=nmp.shape[1]
plt.barh(range(caract), clf.feature_importances_)
plt.xlabel('Importancia de las caracteristicas')
plt.yticks(range(caract),['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE'
,'POT.SOLAR','POT.TOTAL','TEMPERATURA','OrdinalDate','RAD_SIMULATE']
)
plt.ylabel('Caracteristicas')
plt.figure(figsize=(80,80))
plt.show()

max_depth_range = list(range(1, 15))

accuracy_test = []
accuracy_train= []
for depth in max_depth_range:

    clf_pr = DecisionTreeClassifier(max_depth = depth, random_state
= 123)
    clf_pr.fit(X_train, y_train)
    score =  metrics.accuracy_score(y_test, clf_pr.predict(X_test))
    score_train =  metrics.accuracy_score(y_train,
clf_pr.predict(X_train))
    accuracy_test.append(score)
    accuracy_train.append(score_train)

accuracy_train
accuracy_test
```

```
import seaborn as sns
clf_3 = DecisionTreeClassifier(random_state=0)
path = clf_3.cost_complexity_pruning_path(X_train, y_train)
alphas = path['ccp_alphas']

accuracy_train, accuracy_test=[],[]

for i in alphas:
    clf_alp= DecisionTreeClassifier(ccp_alpha=i)
    clf_alp.fit(X_train, y_train)
    y_train_predict=clf_alp.predict(X_train)
    y_test_predict=clf_alp.predict(X_test)

accuracy_train.append(metrics.accuracy_score(y_train,y_train_predict
))

accuracy_test.append(metrics.accuracy_score(y_test,y_test_predict))

plt.figure(figsize=(14,7))
sns.lineplot(y=accuracy_train, x= alphas, label="Train accuracy")
sns.lineplot(y=accuracy_test, x=alphas, label="Test accuracy")
plt.show()

# Post pruning (cost complexity pruning) por validación cruzada
# ------------------------------------------------------------------------
------------
# Valores de ccp_alpha evaluados
param_grid = {'ccp_alpha':np.linspace(0, 5, 10)}

# Búsqueda por validación cruzada
grid = GridSearchCV(
        # El árbol se crece al máximo posible antes de aplicar el
pruning
        estimator = DecisionTreeClassifier(
                            max_depth         = None,
                            min_samples_split = 2,
                            min_samples_leaf  = 1,
                            random_state      = 123
                        ),
        param_grid = param_grid,
        scoring    = 'accuracy',
        cv         = RepeatedKFold(n_splits=4,random_state=123),
        refit      = True,
        return_train_score = True
      )

grid.fit(X_train, y_train)

fig, ax = plt.subplots(figsize=(6, 3.84))
scores = pd.DataFrame(grid.cv_results_)
scores.plot(x='param_ccp_alpha', y='mean_train_score',
yerr='std_train_score', ax=ax)
scores.plot(x='param_ccp_alpha', y='mean_test_score',
yerr='std_test_score', ax=ax)
ax.set_title("Error de validacion cruzada vs hiperparámetro
ccp_alpha");
grid.best_params_

# Búsqueda por validación cruzada
param_grid = {'ccp_alpha':np.linspace(0, 5, 10)}
grid = GridSearchCV(
```

```
        # El árbol se crece al máximo posible antes de aplicar el
pruning
        estimator = DecisionTreeClassifier( ccp_alpha = 0
                          ),
        param_grid = param_grid,
        scoring    = 'accuracy',
        cv         = RepeatedKFold(n_splits=4,random_state=123),
        refit      = True,
        return_train_score = True
       )

grid.fit(X_train,y_train)

# Estructura del árbol final
# ------------------------------------------------------------------
------------
modelo_final = grid.best_estimator_
print(f"Profundidad del árbol: {modelo_final.get_depth()}")
print(f"Número de nodos terminales: {modelo_final.get_n_leaves()}")
print("Accuracy:", metrics.accuracy_score(y_train,
modelo_final.predict(X_train)))
print("Accuracy:", metrics.accuracy_score(y_test,
modelo_final.predict(X_test))*100)
print("F1:", metrics.f1_score(y_test, modelo_final.predict(X_test),
average='weighted')*100)

plot_confusion_matrix(estimator, modelo_final.predict(X_test),
y_test, display_labels=['ON', 'OFF'], cmap=plt.cm.Blues)

plt.show()
dot_data = StringIO()
export_graphviz(modelo_final, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,

feature_names=['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE','POT.SOLAR
','POT.TOTAL','TEMPERATURA','OrdinalDate','RAD_SIMULATE'],class_name
s=['ON', 'OFF'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())

print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores = pd.DataFrame(
                            {'predictor':
['CHARGE.BAT','POT.BAT','FUEL.GE','TIME.GE','POT.SOLAR','POT.TOTAL',
'TEMPERATURA','OrdinalDate','RAD_SIMULATE'],
                             'importancia':
modelo_final.feature_importances_}
                            )
importancia_predictores.sort_values('importancia', ascending=False)


# # RANDOM FOREST
import statsmodels.api as sm
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import ParameterGrid
```

```
from sklearn.inspection import permutation_importance
import multiprocessing
X_train_RF, X_test_RF, y_train_RF, y_test_RF = train_test_split(X,
Y,test_size=0.4, random_state=123)
clf_RF = RandomForestClassifier(max_features=5,random_state=123)
clf_RF = clf_RF.fit(X_train_RF,y_train_RF)
print("Accuracy:", clf_RF.score(X_train_RF,y_train_RF))
print("Accuracy:", metrics.accuracy_score(y_test_RF,
clf_RF.predict(X_test_RF))*100)
print("F1:", metrics.f1_score(y_test_RF, clf_RF.predict(X_test_RF),
average='weighted')*100)
plot_confusion_matrix(clf_RF, X_test_RF, y_test_RF,
display_labels=['ON', 'OFF'], cmap=plt.cm.Oranges)

plt.show()

# Grid Search basado en validación cruzada
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20],
              'criterion'   : ['gini', 'entropy']
             }

# Búsqueda por grid search con validación cruzada
grid = GridSearchCV(
        estimator  = RandomForestClassifier(random_state = 123),
        param_grid = param_grid,
        scoring    = 'accuracy',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5,random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train_RF, y = y_train_RF)
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param*|mean_t|std_t)')     .drop(columns
= 'params')     .sort_values('mean_test_score', ascending = False)
.head(4)
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
modelo_final_RF = grid.best_estimator_
predicciones = modelo_final_RF .predict(X = X_test_RF)
print("Accuracy:", metrics.accuracy_score(y_train_RF,
modelo_final_RF .predict(X_train_RF)))
print("Accuracy:", metrics.accuracy_score(y_test_RF, modelo_final_RF
.predict(X_test_RF)))
print("F1:", metrics.f1_score(y_test_RF, modelo_final_RF
.predict(X_test_RF), average='weighted'))
plot_confusion_matrix(modelo_final_RF , X_test_RF, y_test_RF,
display_labels=['ON', 'OFF'], cmap=plt.cm.Oranges)
plt.show()


# # AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier
```

```
X_train_ada, X_test_ada, y_train_ada, y_test_ada =
train_test_split(X, Y,test_size=0.2, random_state=123)
clf_ada = AdaBoostClassifier(n_estimators=50,learning_rate=1)
clf_ada = clf_ada.fit(X_train_ada,y_train_ada)
print("Accuracy:", metrics.accuracy_score(y_train_ada,
clf_ada.predict(X_train_ada)))
print("Accuracy:", metrics.accuracy_score(y_test_ada,
clf_ada.predict(X_test_ada))*100)
print("F1:", metrics.f1_score(y_test_ada,
clf_ada.predict(X_test_ada), average='weighted')*100)
plot_confusion_matrix(clf_ada, X_test_ada, y_test_ada,
display_labels=['ON', 'OFF'], cmap=plt.cm.Greens)
plt.show()

# Grid de hiperparámetros evaluados
param_grid = {'n_estimators': [25,50,100],
              'learning_rate': [0.5, 1.0, 1.5]
             }

# Búsqueda por grid search con validación cruzada
#
==========================================================================
==========
grid = GridSearchCV(
        estimator  = AdaBoostClassifier(random_state = 123),
        param_grid = param_grid,
        scoring    = 'accuracy',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5,random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train_ada, y = y_train_ada)
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param*|mean_t|std_t)')     .drop(columns
= 'params')     .sort_values('mean_test_score', ascending = False)
.head(4)
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)

modelo_final_AB = grid.best_estimator_

print("Accuracy:", metrics.accuracy_score(y_train_ada,
grid.predict(X_train_ada)))
print("Accuracy:", metrics.accuracy_score(y_test_ada,
grid.predict(X_test_ada))*100)
print("F1:", metrics.f1_score(y_test_ada, grid.predict(X_test_ada),
average='weighted')*100)
```

Anexo 5. Figuras obtenidas

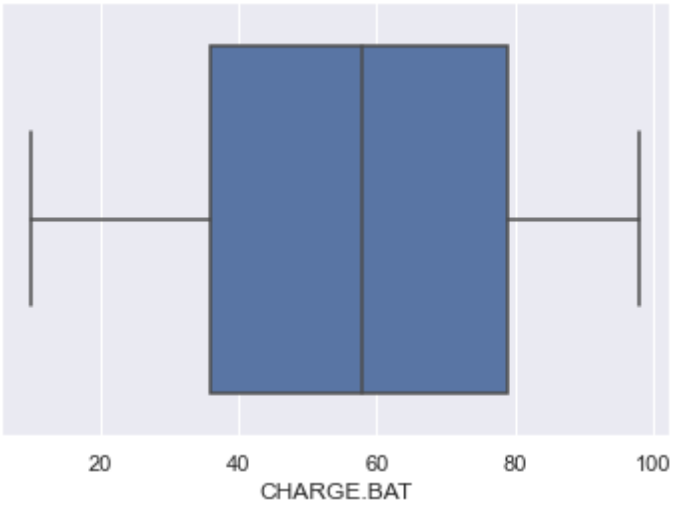

Figura 30. - ***BoxPlot*** **de la variable CHARGE.BAT**

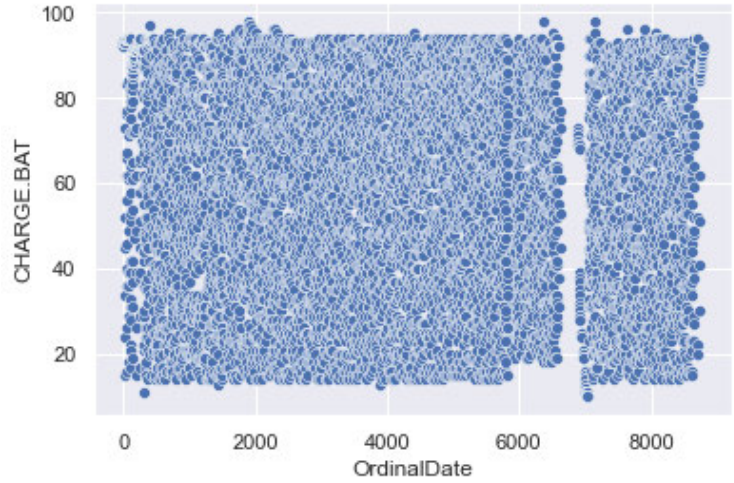

Figura 31. - ***Scatterplot*** **de la variable CHARGE.BAT en función de la hora ordinal del año**

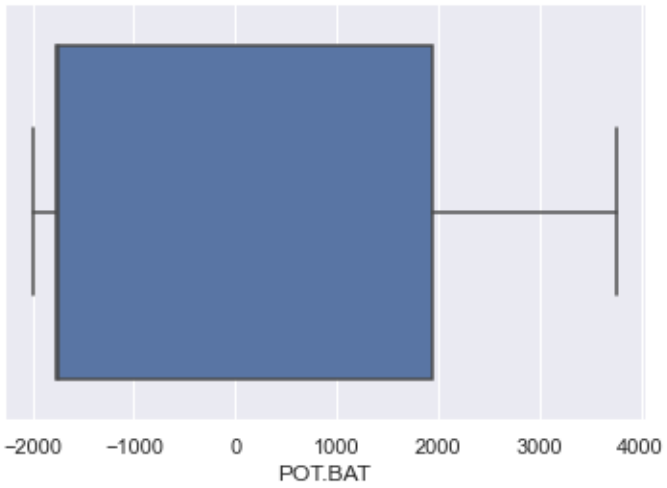

Figura 32. - BoxPlot de la variable POT.BAT

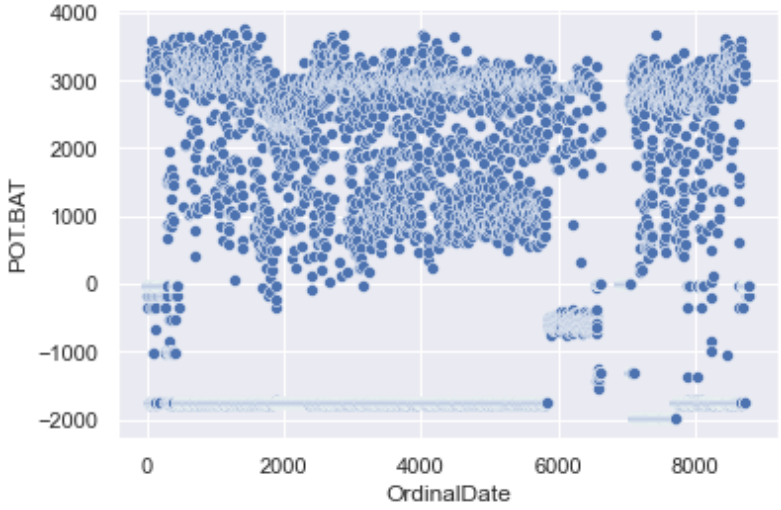

Figura 33. - *Scatterplot* de la variable POT.BAT en función de la hora ordinal del año

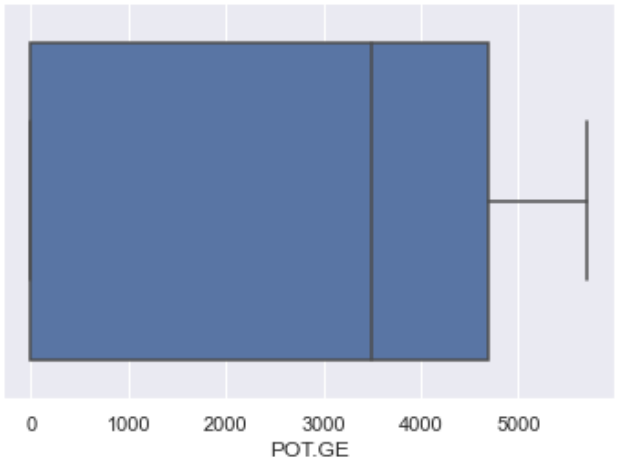

Figura 34. - *BoxPlot* de la variable POT.GE

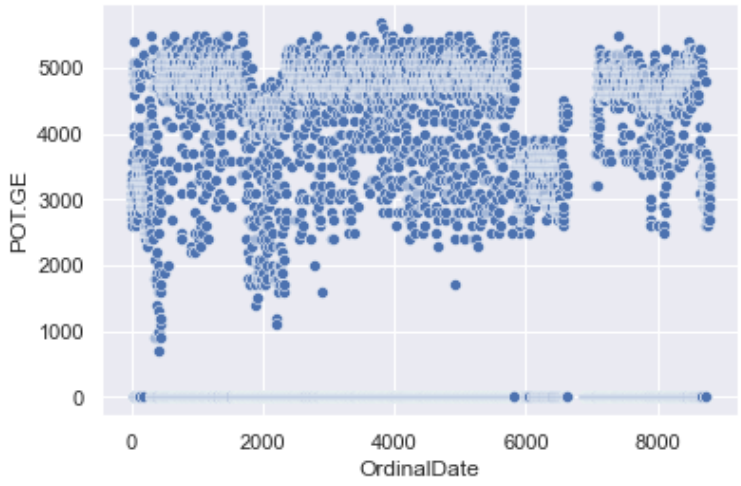

Figura 35. - ***Scatterplot*** **de la variable POT.GE en función de la hora ordinal del año**

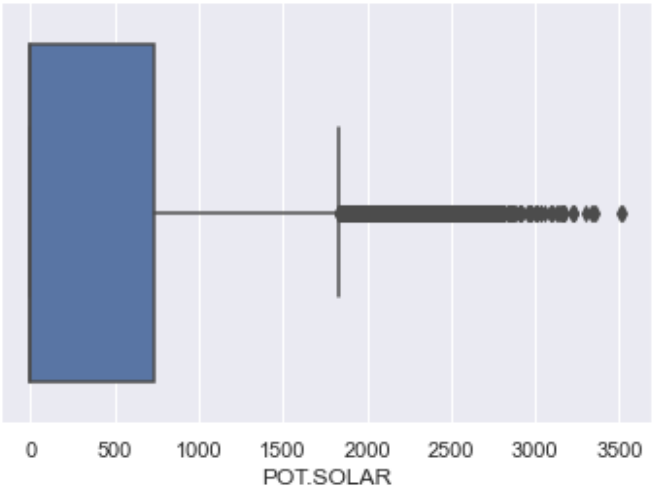

Figura 36. - ***BoxPlot*** **de la variable POT.SOLAR**

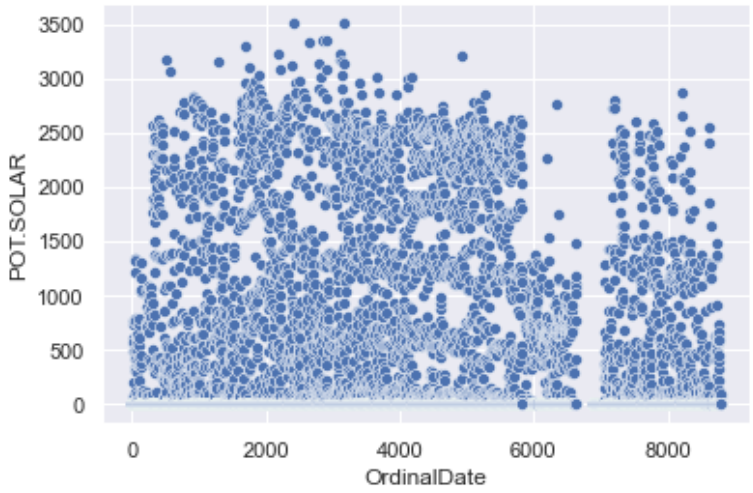

Figura 37. - ***Scatterplot*** **de la variable POT.SOLAR en función de la hora ordinal del año**

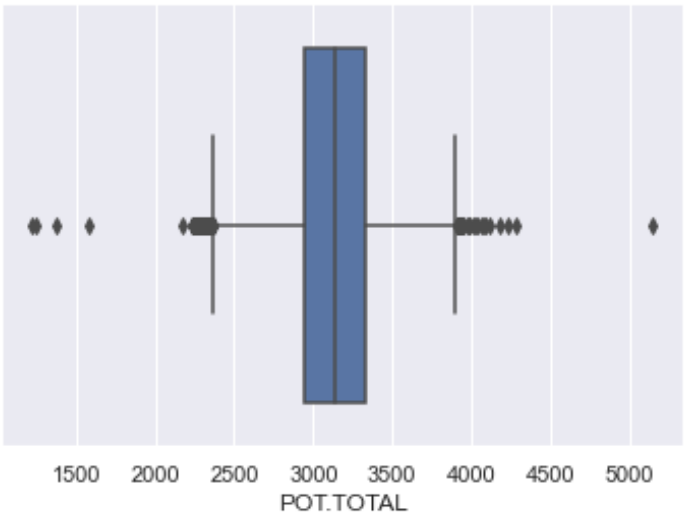

Figura 38. - ***BoxPlot*** **de la variable POT.TOTAL**

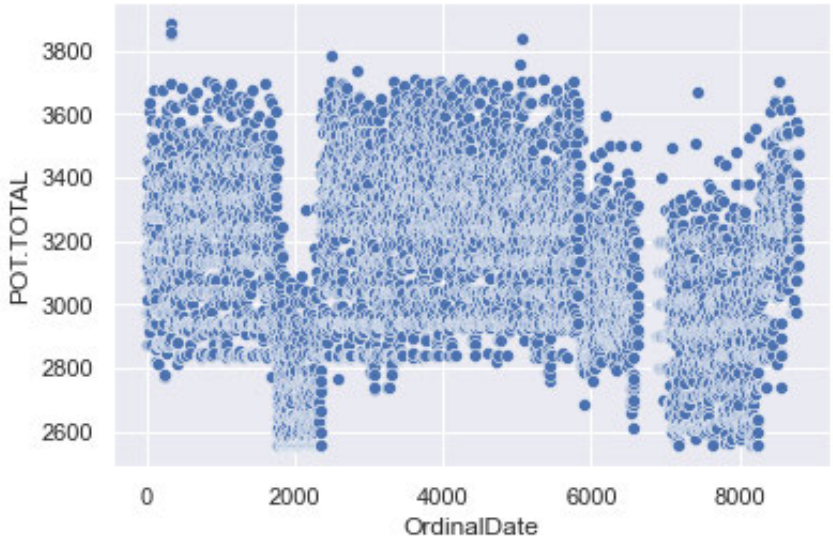

Figura 39. - ***Scatterplot*** **de la variable POT.TOTAL en función de la hora ordinal del año**

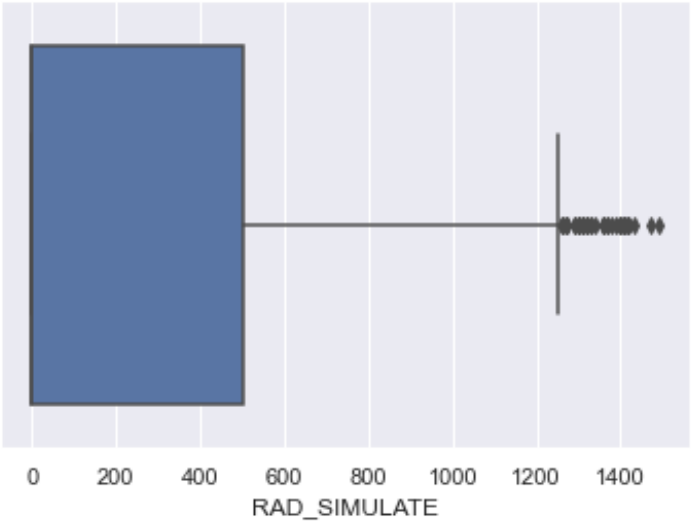

Figura 40. - ***BoxPlot*** **de la variable RAD_SIMULATE**

Figura 41. - ***Scatterplot*** **de la variable RAD_SIMULATE en función de la hora ordinal del año**

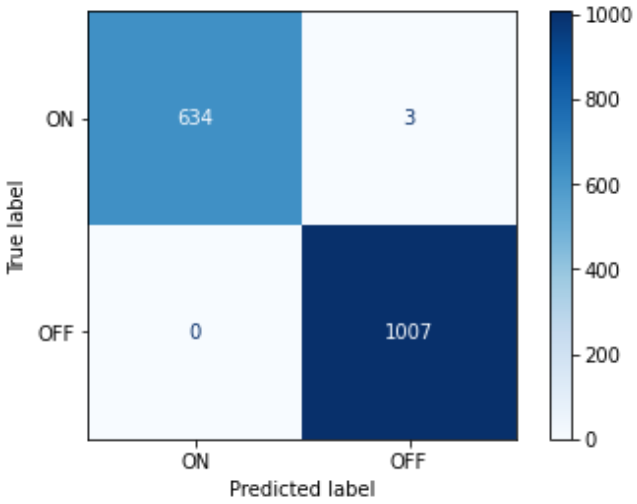

Figura 42. - Matriz de confusión DT

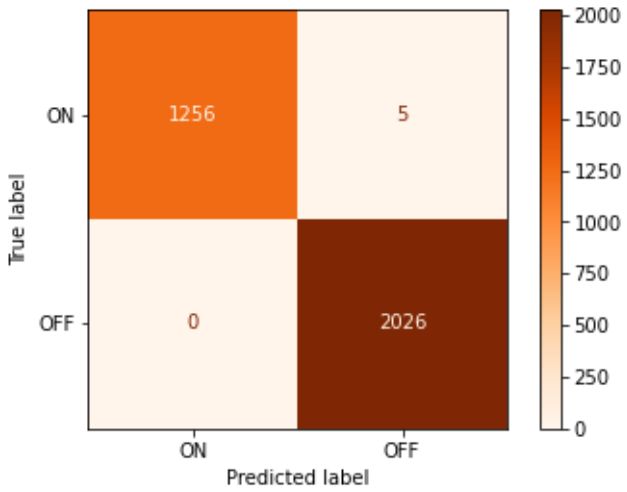

Figura 43. - Matriz de confusión RF

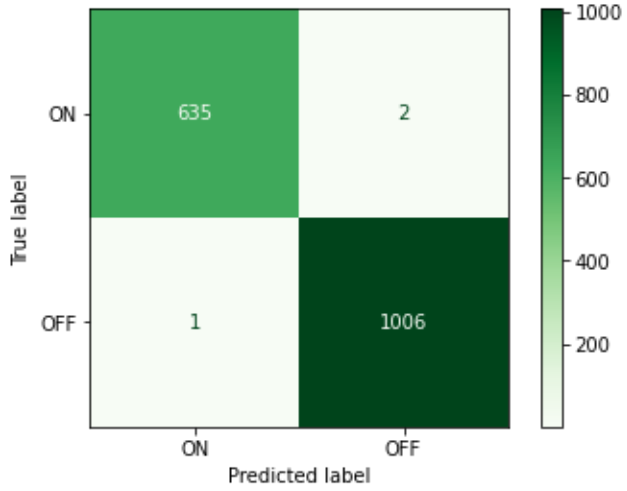

Figura 44. - Matriz de confusión AdaBoost