Universitat Oberta de Catalunya

uoc.edu

Aplicación de Machine Learning al consumo eléctrico de edificios inteligentes

Mónica Meneses Díez
Máster Universitario en Ingeniería de Telecomunicación
Smart Cities

David Crespo García
Carlos Monzo Sanchez

27 de diciembre de 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación de Machine Learning al consumo eléctrico de edificios inteligentes*
Nombre del autor:	*Mónica Meneses Díez*
Nombre del consultor/a:	*David Crespo García*
Nombre del PRA:	*Carlos Monzo Sanchez*
Fecha de entrega (mm/aaaa):	12/2021
Titulación:	*Máster Universitario en Ingeniería de Telecomunicación*
Área del Trabajo Final:	*Smart Cities*
Idioma del trabajo:	*Español*
Palabras clave	*Consumo eléctrico, Machine Learning, eficiencia energética*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En la actualidad, el consumo de la electricidad es necesaria en el día a día de todas las personas. Por lo tanto, el consumo eléctrico se ha convertido en una necesidad de la sociedad. En este trabajo se estudia la predicción del consumo eléctrico de los edificios a partir de *Machine Learning.* Así pues, a partir de una gran cantidad de datos de consumo eléctricos en edificios se podrá realizar una predicción del consumo, empleando *Machine Learning*.

El objetivo es estudiar los distintos modelos de aprendizaje supervisado y no supervisado que se pueden aplicar a este tipo de datos de consumo para poder obtener la mejor predicción puesto que muchos de estos datos tienen valor cero, por lo tanto, puede verse sesgado el modelo que se aplique en este tipo de predicciones.

Además, a partir de los datos obtenidos se podrá mejorar la eficiencia energética y tomar medidas para sacar el máximo partido de las energías renovables reduciendo los costes asociados al consumo eléctrico tradicional.

Abstract (in English, 250 words or less):

Nowadays, the consumption of electricity is necessary in every person's daily life. Therefore, electricity consumption has become a necessity for society. In this work, we study the prediction of electricity consumption in buildings using Machine Learning. Thus, from a large amount of electricity consumption data in buildings, it will be possible to make a prediction of consumption using Machine Learning.

The aim is to study the different supervised and unsupervised learning models that can be applied to this type of consumption data in order to obtain the best prediction, given that many of these data have zero value, and therefore the model applied in this type of predictions may be biased.

Furthermore, the data obtained can be used to improve energy efficiency and take measures to make the most of renewable energies by reducing the costs associated with traditional electricity consumption.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Las *Smart cities* han sido toda una revolución en el sector tecnológico. Estas ciudades detectan las necesidades de sus ciudadanos y reaccionan ante estas a partir del conocimiento de sistemas y elementos del sector público en tiempo real o incluso anticipándose. Un ejemplo de ello es la eficiencia energética en este tipo de ciudades.

Hoy en día la eficiencia del consumo eléctrico en edificios es muy relevante en la sociedad, puesto que se cuida más el medioambiente y se reducen costes. Esto último es lo que más preocupa a la población ya que en el último año se han producido altas subidas de precios en el consumo de electricidad.

A partir de una gran cantidad de datos de consumo eléctrico en edificios se podrá realizar una predicción del consumo, empleando Machine Learning. Se creará un modelo que relacionará datos meteorológicos de un lugar con la predicción del consumo eléctrico de un edificio para conocer si es posible generar energía renovable suficiente para satisfacer el consumo.

Como se ha comentado, se empleará Machine Learning puesto que es una de las tecnologías en auge de los últimos años, y a partir de ella se pueden crear modelos automáticos para que se analicen una gran cantidad de datos complejos y ofrezcan resultados rápidos y precisos.

Este estudio ayudará en la toma de decisiones para mejorar la eficiencia energética de un edificio y reducir costes derivados del consumo, puesto que la aplicación indicará si el edificio es capaz de abastecerse a partir de energía renovable empleando medidas como el uso de paneles solares, etc.

1.2. Objetivos del Trabajo

Los objetivos que se realizarán en este trabajo serán:

- Búsqueda y recopilación de información sobre el consumo eléctrico en edificios.

- Búsqueda y recopilación de una gran cantidad de datos (*dataset*) del consumo eléctrico en edificios.

- Búsqueda y recopilación de información sobre los tipos de sensores meteorológicos que se pueden instalar en edificios.

- Búsqueda y recopilación de una gran cantidad de datos (*dataset*) meteorológicos en la misma ubicación de dónde se han recogido los datos del consumo eléctrico.

- Búsqueda y recopilación de información sobre la tecnología *Machine Learning* que se va a emplear en este estudio.

- Búsqueda y recopilación de información sobre energías renovables.

- Análisis de los datos obtenidos y tratados para poderse aplicar a procesos de *Machine Learning*.

- Crear un modelado para poder realizar una predicción sobre el consumo eléctrico en edificios a partir de *Machine Learning* relacionándolo con datos meteorológicos.

- Creación de una simulación, a modo piloto, de una monitorización de un edificio inteligente donde se observará si el edificio podrá abastecerse únicamente de energía renovable y si es así, a partir de cuándo.

- Aplicación de los conocimientos aprendidos en las distintas asignaturas cursas en el Máster Universitario en Ingeniería de Telecomunicación.

1.3. Enfoque y método seguido

Para poder cubrir las necesidades de este estudio se desarrollará un nuevo producto. Se realizará un tratamiento de datos, a partir de Machine Learning, como se ha comentado en el apartado anterior.

Es necesario desarrollar un nuevo producto, en lugar de adaptar un producto ya existente, ya que hay que adaptar los datos que se han recopilado y crear una relación entre estos. Por tanto, esta estrategia es la más apropiada para poder conseguir los objetivos.

1.4. Planificación del Trabajo

1.4.1. Recursos necesarios

Los recursos necesarios para poder realizar el trabajo son:

- Grandes cantidades de datos meteorológicos.

- Grandes cantidades de datos de consumos eléctricos de edificios.

- Un ordenador para poder montar el estudio a partir de *Machine Learning* de los datos y sus modelados.

1.4.2. Tareas

Las tareas que se realizarán en este trabajo serán:

- Búsqueda y análisis de los datos meteorológicos.

- Búsqueda y análisis de los datos de consumo eléctrico.

- Adaptación de los datos.

- Crear un modelado para poder realizar una predicción sobre el consumo eléctrico en edificios a partir de *Machine Learning* relacionándolo con datos meteorológicos

- Obtener conclusiones sobre la ejecución.

- Creación de una simulación, a modo piloto, de una monitorización de un edificio inteligente donde se observará si el edificio podrá abastecerse únicamente de energía renovable y si es así, a partir de cuándo.

- Realizar entregas de evaluación continua (PEC).

1.4.3. Planificación

La planificación de este trabajo se divide en cinco fases, una por cada entrega PEC:

- **Definición** (PEC1 15/09/2021 – 26/09/2021):
  Se definirá el tema y los aspectos fundamentales del proyecto, como el alcance, objetivos y planificación temporal de las tareas.

- **Estado del arte** (PEC2 27/09/2021 – 10/10/2021):
  Se realizará un estudio completo de los proyectos y técnicas que se emplearán en el trabajo.

- **Diseño, implementación, análisis y/o propuesta de un concepto novedoso** (PEC3 11/10/2021 – 03/12/2021):
  Se desarrollará el producto del proyecto.

- **Memoria** (PEC4 04/12/2021 – 27/12/2021):
  Se documentará todo el proyecto en una memoria explicando qué se ha realizado.

- **Defensa** (PEC5 28/12/2021 – 05/01/2021):
  Se presentará y defenderá el TFM delante de un tribunal.

También se incorporarán hitos con el tutor del TFM, por tanto, la representación de esta planificación en el diagrama de Gantt es:

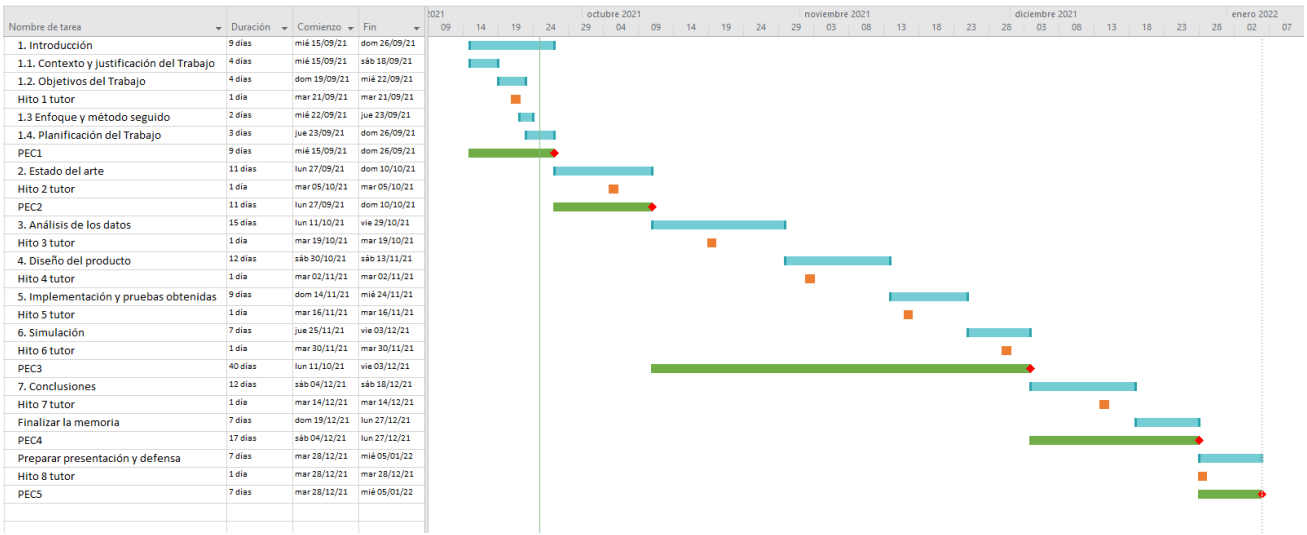


Figura 1: Diagrama de Gantt

1.5. Breve sumario de productos obtenidos

Los productos obtenidos en este trabajo son:

- Un modelado poder realizar una predicción sobre el consumo eléctrico en edificios a partir de *Machine Learning* relacionándolo con datos meteorológicos.

1.6. Breve descripción de los otros capítulos de la memoria

Los otros capítulos que completan la memoria del trabajo son: el estado del arte, el análisis de los datos, diseño del producto, implementación y pruebas obtenidas y una simulación.

- **El estado del arte**: se detallará la información de base del proyecto junto con el contexto.

- **Análisis de los datos**: se detallará el análisis de los datos que se tratarán con la aplicación.

- **Diseño del producto**: se detallará un diseño de la aplicación de *Machine Learning* que se desarrollará.

- **Implementación y pruebas**: se detallará el desarrollo y se probará.

2. Estado del arte

2.1. Smart Cities

El concepto de *Smart City* engloba diferentes estrategias urbanas en torno a las TIC y las nuevas infraestructuras, que persiguen un crecimiento económico bajo en carbono, inclusivo y participativo y una ruptura radical con el urbanismo del siglo XX. La *Smart City* o ciudad inteligente se compone tanto de infraestructuras físicas, por ejemplo, sensores, informática ubicua, etc., como de infraestructuras inmateriales, por ejemplo, nuevas formas de gobernanza, un cambio hacia la participación privada y de la sociedad civil, y nuevos procesos de innovación. [1]

Así pues, la ciudad inteligente se convierte en una plataforma digital que permite maximizar la economía, la sociedad, el entorno y el bienestar de las ciudades. Además, facilita el cambio hacia un comportamiento más sostenible entre los usuarios, empresas y administración de la ciudad. [2]

Para que esto sea posible es necesario que deban desarrollarse unos potentes sistemas de comunicación que no solo almacenen e intercambien datos entre sí, sino que además puedan analizar dichos datos aplicando algoritmos adecuados y extraigan información de gran valor para la mejora de los procesos y para la toma de las mejores decisiones que repercutan en el bienestar del ciudadano. [2]

Los principales elementos que componen una *Smart City* son [2][3]:

- ***Smart mobility*:** es como se conoce a la movilidad inteligente. Este concepto se refiere al uso de medios de transporte alternativos al uso individual del vehículo privado. En la práctica, *Smart mobility* puede darse de formas diferentes, como compartir el coche, desplazarse en transporte público, los patinetes eléctricos, caminar o ir en bicicleta, entre otros. [4]

- ***Smart people:*** este concepto engloba los aspectos y hace referencia a las siguientes características: e-habilidades, trabajo habilitado mediante la IT, tener acceso a la educación y a la formación, recursos humanos y capacidad de gestión, en una sociedad inclusiva que mejora la creatividad y la innovación. [5]

- ***Smart economy:*** se define como la base principal del desarrollo urbano en una comunidad inteligente. Este modelo se basa en una serie de conceptos para impulsar el desarrollo, la sostenibilidad y el atractivo para nuevas inversiones, los principales son: e-business, e-commerce,

incremento de la productividad, empleo e innovación en TI y generación de servicios y nuevos productos, nuevos modelos y oportunidades de negocio y emprendimiento. [6]

- ***Smart environment:*** se centra en el uso de la *Green TI* (*Green Computing and Information Technology*) para desarrollar un entorno inteligente, capaz de optimizar los recursos naturales, preservar y proteger el medio ambiente, reducir los gases y residuos de manera sostenible, y de controlar y racionalizar el consumo de energía. [7]

- ***Smart governance:*** es el principal elemento para el desarrollo de una ciudad inteligente ya que su función principal es desarrollar políticas que promuevan la incorporación de las TI en la ciudad alservicio del ciudadano y de las sinergias entre los diferentes actores de la sociedad. Las TI no son el objetivo sino el medio para que la ciudad disponga de los elementos necesarios para que progresivamente sea una ciudad inteligente, sostenible y para el beneficio del ciudadano para mejorar su calidad de vida. [8]

- ***Smart living:*** hace referencia a los nuevos estilos de vida mediante las TI, el comportamiento y el consumo. Los servicios inteligentes *Smart Living* actúan en los ámbitos de: salud, seguridad ciudadana, cultura, domótica en viviendas, *smart buildings,* proporcionando servicios inteligentes como e-salud, e-accesibilidad y e-turismo, todo ello con el objetivo de incrementar los niveles cohesión social, el capital y la seguridad. [9]

2.2. Smart Buildings

Un edificio inteligente utiliza su inteligencia para recopilar datos procesables de los dispositivos de los usuarios, los sensores, los sistemas y los servicios de las instalaciones. La aplicación de esos datos mediante la inteligencia artificial y el aprendizaje automático (IA/ML) hace que el edificio sea programable y responda a las necesidades de los usuarios y del gestor del edificio. [10]

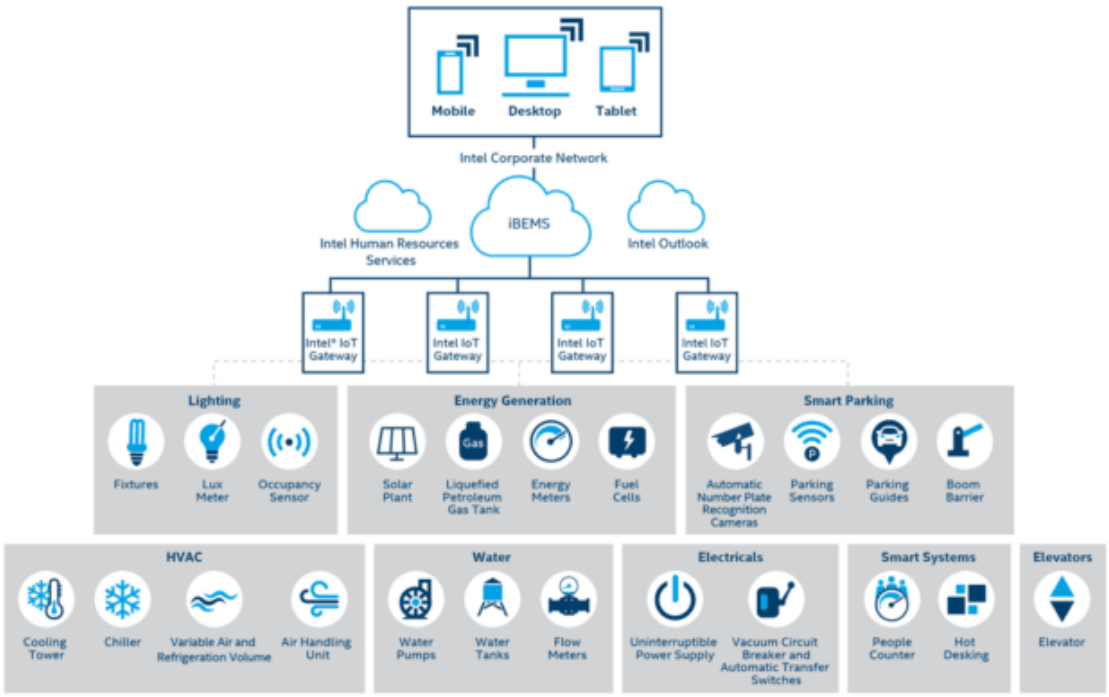


Figura 2: Arquitectura *Smart building*. Fuente: www.insight.tech

Como se puede observar en la Figura 2, la arquitectura de los edificios inteligentes contiene numerosos sensores IoT. Estos requisitos de los edificios inteligentes para su consideración como nodo IoT, se recoge en la norma UNE 178108:2017 Ciudades inteligentes. [11]

Por lo tanto, las siguientes consideraciones sirven de base para el diseño y la implementación de un prototipo de edificio inteligente IoT. [12]

2.2.1. Clima

La temperatura en las casas y oficinas es un factor importante: para un confort y una productividad óptimos, la temperatura de las habitaciones debe estar entre 15°C y 25°C. Tradicionalmente, la calefacción y el aire acondicionado se han controlado manualmente o por control remoto, pero para el control automático hay que considerar dos escenarios: la calefacción central y la calefacción independiente en cada habitación. [12]

2.2.2. Iluminación

La iluminación artificial puede ser un factor de coste importante en muchos edificios. La utilización de sensores para controlar los niveles de iluminación de una habitación, combinada con sensores para detectar la presencia de personas en ella, ofrece oportunidades para un importante ahorro de costes y una reducción del consumo de energía. [12]

2.2.3. Consumo eléctrico

Los edificios tradicionales suelen tener un solo contador de electricidad. Por ello, las personas no pueden seguir fácilmente su consumo de electricidad ni analizar cómo la utilizan. En un edificio inteligente, la estructura del sistema de gestión de la energía permite a las personas seguir su consumo de energía en una aplicación o en Internet. La medición generalizada de la corriente consumida por los dispositivos individuales permite a los usuarios ver cómo se consume la energía y reducir su uso energético y el posible gasto resultante. [12]

2.2.4. Consumo de agua

La visualización de los niveles de uso del agua y la indicación de los costes de dicho uso podrían acabar motivando la adopción más generalizada de técnicas como la recogida de agua de lluvia para reducir el consumo de agua. [12]

2.2.5. Detección de incendios, humo y monóxido de carbono

El fuego es un riesgo importante para la vida y la propiedad en los edificios. La detección de incendios y la activación de alarmas mediante sensores en cada habitación es una necesidad en un edificio inteligente. Además, los sensores de monóxido de carbono son otra forma de evitar accidentes mortales. [12]

2.2.6. Detección de fugas

Los sensores de agua colocados cerca de lavabos, duchas y otros puntos de agua pueden indicar si hay riesgo de inundación y el edificio debe alertar a sus propietarios de los riesgos y activar las contramedidas si es posible.

2.2.7. Estado de las ventanas

Las ventanas abiertas pueden ser un problema de seguridad si hay niños en el edificio o un problema de seguridad en caso de posibles intrusos. Se podría enviar una alerta a los usuarios si los sensores de movimiento permiten a un edificio inteligente detectar un caso en el que un ocupante sale del edificio pero deja una ventana abierta. [12]

2.3. Sensores

Los sensores desempeñan un papel importante en la automatización de cualquier aplicación midiendo y procesando los datos recogidos para detectar cambios en las materias físicas. Cuando se produce un cambio en cualquier condición física para la que se fabrica un sensor, produce una respuesta medible. [13]

Existen numerosos protocolos de encaminamiento para redes de sensores inalámbricas, algunos de estos protocolos de encaminamiento son: LEACH, TEEN, HEED, PEGASIS y ERMSS. [14]

2.3.1. LEACH

LEACH es un protocolo de encaminamiento jerárquico en redes de sensores inalámbricas (WSN, del inglés Wireless Sensor Network). En el protocolo LEACH la selección de la cabeza del clúster es aleatoria y cada nodo tiene la oportunidad de convertirse en cabeza del clúster. El consumo de energía de los nodos sensores es diferente. El funcionamiento de LEACH se divide en ciclos. Cada ciclo contiene una etapa de configuración, en la que cada nodo sensor elige un número aleatorio entre 0 y 1 para decidir si se convertirá en cabeza de clúster o no. Si un nodo con muy poca energía se convierte en cabeza del clúster, la vida útil de la red se reduce.[15]

2.3.2. TEEN

TEEN del inglés Threshold sensitive Energy Efficient sensor Network protocol, significa protocolo de red de sensores sensible al umbral con eficiencia energética. TEEN se ha desarrollado para redes reactivas. TEEN es un protocolo de enrutamiento jerárquico sensible al umbral. En este protocolo el cabeza del clúster envía un umbral duro y un umbral blando a sus miembros. TEEN proporciona un consumo eficiente de energía en aplicaciones de detección de tiempo crítico. [16]

2.3.3. HEED

HEED es un algoritmo de agrupación distribuida. Distribuye el consumo de energía en toda la red para aumentar la vida útil de la misma. A la hora de seleccionar la cabeza del clúster, HEED también tiene en cuenta la energía residual de los nodos sensores de la WSN. [17]

2.3.4. PEGASIS

PEGASIS es una mejora de LEACH. PEGASIS utiliza un algoritmo ambicioso para establecer la distancia más corta entre los nodos vecinos, de forma que cada nodo pueda enviar información al menor coste. La selección de cabeza del clúster en PEGASIS es aleatoria. [18]

2.3.5. ERMSS

ERMSS minimiza el número de enlaces maximizando el solapamiento de los trayectos por los que se encaminan los datos desde un determinado origen a un determinado sumidero. Para un alto grado de solapamiento entre los trayectos fuente-sumidero, este protocolo utiliza un esquema adaptativo. [19]

2.3.6. Tipos de sensores

Existen diferentes tipos de sensores que pueden ir desde los más sencillos hasta los más complejos. La clasificación de los sensores puede basarse en sus especificaciones, su método de conversión, el tipo de material utilizado, su fenómeno físico de detección, las propiedades que mide y el campo de aplicación. Los sensores más destacados se comentarán a continuación. [13][20]

- **Sensores de proximidad**

La posición de cualquier objeto cercano puede detectarse fácilmente sin ningún contacto físico con el sensor de proximidad. Al emitir radiación electromagnética como los infrarrojos, encuentra la presencia de un objeto simplemente buscando cualquier variación en la señal de retorno. [21]

- **Sensores de posición**

El sensor de posición detecta la presencia de personas u objetos en una zona determinada detectando su movimiento. [22]

- **Sensores de ocupación**

El sensor de ocupación detecta la presencia de personas u objetos en una zona determinada. Se puede utilizar para la monitorización remota a través de varios parámetros como la temperatura, la humedad, la luz y el aire. [23]

- **Sensores de movimiento**

Un sensor de movimiento es un dispositivo que se utiliza para detectar todo el movimiento cinético y físico en el entorno. [24]

- **Sensores de velocidad**

Es un sensor que calcula la velocidad de cambio en la medición de la posición constante y los valores de posición a intervalos conocidos. [25]

- **Sensores de temperatura**

Los sensores de temperatura son útiles para detectar los cambios físicos en el cuerpo midiendo la energía térmica. [13]

- **Sensores de presión**

Los sensores de presión detectan la cantidad de fuerza y la convierten en señales. Este tipo de sensores puede utilizarse en la vigilancia de la salud. [26]

- **Sensores químicos**

Un sensor químico es un dispositivo analítico utilizado para medir la composición química del medio ambiente. El control de la calidad del aire puede realizarse mediante una red de sensores químicos inalámbricos, controlando los vapores químicos en el medio ambiente. [27]

- **Sensores de humedad**

Un sensor de humedad mide la temperatura del aire así como la humedad y señala la humedad del ambiente. [13]

- **Sensores de calidad del agua**

Los sensores de calidad del agua se utilizan para el control de los iones. La calidad del agua se mide mediante sensores de calidad del agua. [28]

- **Sensores infrarrojos**

Los sensores de infrarrojos emiten o detectan radiaciones infrarrojas para percibir algunas características de determinados objetos. También pueden medir la emisión de calor. Este tipo de sensores puede utilizarse en la automatización del hogar para supervisar y controlar los electrodomésticos, como el encendido y apagado de las luces. [29]

2.4. Machine Learning

Machine Learning (ML) se define como el uso de algoritmos y estadísticas computacionales para aprender de los datos sin ser programados explícitamente. Es una subsección del ámbito de la inteligencia artificial dentro de la informática. Aunque el campo del aprendizaje automático no explotó hasta hace poco, el término se acuñó por primera vez en 1959 y las investigaciones más fundamentales se realizaron a lo largo de los años 70 y 80. El auge del aprendizaje automático se debe a la abundancia de datos, a un almacenamiento de datos más eficiente y a ordenadores más rápidos. [30]

Machine learning ofrece una manera eficiente de capturar el conocimiento mediante la información contenida en los datos, para mejorar de forma gradual el rendimiento de modelos predictivos y tomar decisiones basadas en dichos datos. Se ha convertido en una tecnología con una amplia presencia, y actualmente está presente en: filtros *anti-spam* para correo electrónico, conducción automática de vehículos o reconocimiento de voz e imágenes. [31]

2.4.1. Tipos de Machine Learning

- **Aprendizaje Supervisado**

Se refiere a un tipo de modelos de Machine Learning que se entrenan con un conjunto de ejemplos en los que los resultados de salida son conocidos. Los modelos aprenden de esos resultados conocidos y realizan ajustes en sus parámetros interiores para adaptarse a los datos de entrada. Una vez el modelo es entrenado adecuadamente, y los parámetros internos son coherentes con los datos de entrada y los resultados de la batería de datos de entrenamiento, el modelo podrá realizar predicciones adecuadas ante nuevos datos no procesados previamente. [31]

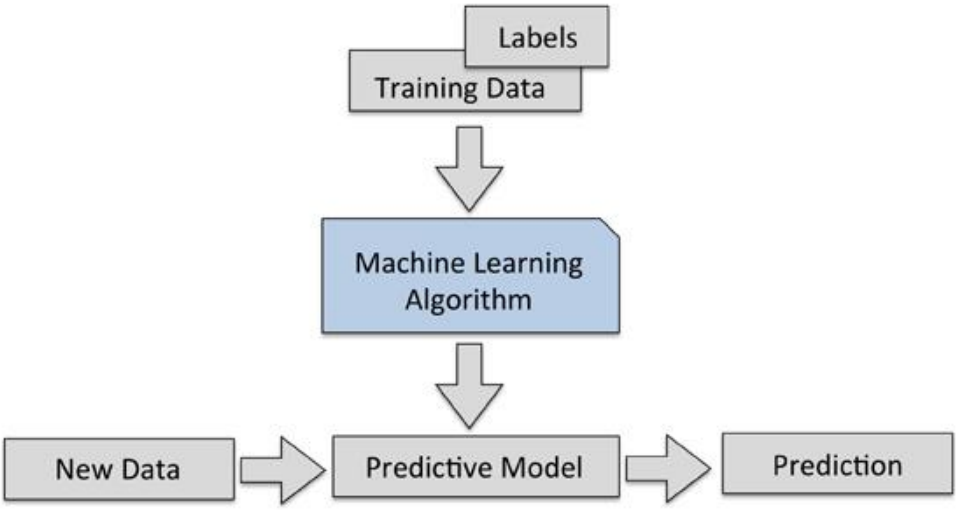


Figura 3: Aprendizaje supervisado. Fuente: www.medium.com

Hay dos aplicaciones principales de aprendizaje supervisado: clasificación y regresión:

1. **Clasificación:**

   Clasificación es una sub-categoría de aprendizaje supervisado en la que el objetivo es predecir las clases categóricas (valores discretos, no ordenados, pertenencia a grupos). El ejemplo típico es la detección de correo spam, que es una clasificación binaria (un email es spam — valor "1"- o no lo es — valor "0" -). También hay clasificación multi-clase, como el reconocimiento de caracteres escritos a mano (donde las clases van de 0 a 9). [31]

2. **Regresión:**

   La regresión se utiliza para asignar categorías a datos sin etiquetar. En este tipo de aprendizaje tenemos un número de variables predictoras (explicativas) y una variable de respuesta continua (resultado), y se tratará

de encontrar una relación entre dichas variables que nos proporciones un resultado continuo. [31]

- **Aprendizaje no supervisado**

En el aprendizaje no supervisado, trataremos con datos sin etiquetar cuya estructura es desconocida. El objetivo será la extracción de información significativa, sin la referencia de variables de salida conocidas, y mediante la exploración de la estructura de dichos datos sin etiquetar. [31]

Hay dos categorías principales: agrupamiento y reducción dimensional.

1. **Agrupamiento ó Clustering:**

   El agrupamiento es una técnica exploratoria de análisis de datos, que se usa para organizar información en grupos con significado sin tener conocimiento previo de su estructura. Cada grupo es un conjunto de objetos similares que se diferencia de los objetos de otros grupos. El objetivo es obtener un numero de grupos de características similares. [31]

2. **Reducción dimensional:**

   Es común trabajar con datos en los que cada observación se presenta con alto número de características, en otras palabras, que tienen alta dimensionalidad. Este hecho es un reto para la capacidad de procesamiento y el rendimiento computacional de los algoritmos de Machine Learning. La reducción dimensional es una de las técnicas usadas para mitigar este efecto.
   La reducción dimensional funciona encontrando correlaciones entre las características, lo que implica que existe información redundante, ya que alguna característica puede explicarse parcialmente con otras (por ejemplo, puede existir dependencia lineal). Estas técnicas eliminan “ruido” de los datos (que puede también empeorar el comportamiento del modelo), y comprimen los datos en un sub-espacio más reducido, al tiempo que retienen la mayoría de la información relevante. [31]

- **Aprendizaje reforzado**

El aprendizaje reforzado es una de las ramas más importantes del aprendizaje profundo. El objetivo es construir un modelo con un agente que mejora su rendimiento, basándose en la recompensa obtenida del entorno con cada interacción que se realiza. La recompensa es una medida de lo correcta que ha sido una acción para obtener un objetivo determinado. El agente utiliza esta recompensa para ajustar su comportamiento futuro, con el objetivo de obtener la recompensa máxima. [31]

2.4.2. Tecnologías

Las tecnologías más destacables que se pueden emplear en desarrollo de modelos de aprendizaje automático son los siguientes.

- Keras: Se trata de una librería de código abierto que se centra en simplificar la creación de modelos de aprendizaje profundo. Está escrita en Python. Se ejecuta de forma óptima tanto en CPUs como en GPUs, además de ser conocida por su facilidad de uso así como la rapidez en la creación de prototipos. [32]

- Torch: Es una de las tecnologías más antiguas, creada en 2002. Se trata de una librería de aprendizaje automático de código abierto que ofrece una variedad de algoritmos para el aprendizaje profundo. Su objetivo es tener la máxima flexibilidad y velocidad en la construcción de sus algoritmos científicos al tiempo que hace el proceso extremadamente simple. [33]

- Caffe: Se trata de una de las tecnologías más recientes, que inspira un grado de innovación con una arquitectura expresiva junto con la provisión de una comunidad vibrante. Se centra en la velocidad, la expresividad y la funcionalidad. [34]

- TensorFlow: Se creó por Google en 2015, es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de AA fácilmente. [35]

- Theano: Se trata básicamente de una biblioteca de código abierto de Python que se puede utilizar para crear varios modelos de aprendizaje automático. Al ser una de las bibliotecas más antiguas, se considera un estándar de la industria. Simplifica el proceso de optimización, definición y evaluación de expresiones matemáticas. [36]

- Microsoft Cognitive Toolkit (CNTK): es un conjunto de herramientas de código abierto para el aprendizaje profundo distribuido de nivel comercial. Describe las redes neuronales como una serie de pasos computacionales a través de un gráfico dirigido. CNTK permite al usuario realizar y combinar fácilmente los tipos de modelos más populares, como las redes neuronales de avance, las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN/LSTM). [37]

2.5. Almacenamiento en la nube

El almacenamiento en la nube es un almacenamiento externo mantenido por un tercero. El almacenamiento en la nube guarda sus datos de forma segura en una base de datos remota, de modo que no tiene que guardar sus datos y archivos en el disco duro de su ordenador o en otro dispositivo de almacenamiento. [38]

El almacenamiento en la nube tiene muchas ventajas sobre el almacenamiento en el disco duro. En primer lugar, no tienes que poseer físicamente el dispositivo de almacenamiento como harías con una unidad flash, por ejemplo, así que no corres el riesgo de perder datos irreemplazables. En segundo lugar, el almacenamiento en la nube hace que sea fácil compartir el contenido. [38]

En la computación en la nube, los servicios de almacenamiento en la nube funcionan como una red de servidores de datos conectados que se utilizan colectivamente para compartir y acceder a sus archivos en todos los dispositivos. Los proveedores de almacenamiento en la nube poseen y mantienen los servidores externos que componen esta red en sus centros de datos. Los usuarios pueden subir archivos a los servidores y acceder a sus datos en la nube a través de un sitio web, una aplicación de escritorio o una aplicación móvil. [38]

2.5.1. Bases de datos en la nube

Hay dos tipos principales de modelos de implementación de bases de datos en la nube.

- El **tradicional,** que es muy similar a una base de datos in-situ y administrada internamente, con la única diferencia del suministro de la infraestructura. En este caso, una organización compra espacio de máquina virtual de un proveedor de servicios en la nube y la base de datos se implementa en la nube. La organización se encarga de la supervisión y la gestión de la base de datos. [39]

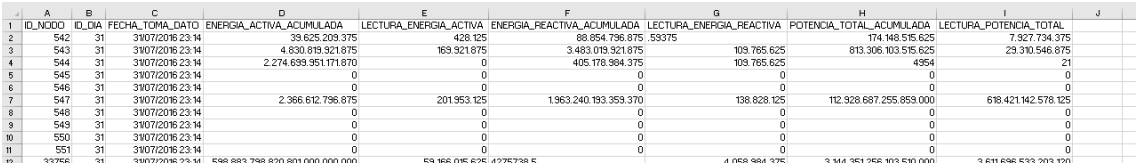
- La **base de datos como servicio (DBaaS),** en el que una organización contrata un servicio de suscripción de pago con un proveedor de servicios en la nube. El proveedor de servicios ofrece diferentes tareas operativas, de mantenimiento, administrativas y de gestión de bases de datos en tiempo real para el usuario final. La base de datos se ejecuta en la infraestructura del proveedor de servicios. Este modelo de uso suele incluir la automatización en las zonas de suministro, copia de seguridad, escalabilidad, alta disponibilidad, seguridad, aplicación de parches y supervisión del estado. [39]

3. Análisis de los datos

La parte más importante en el desarrollo de un modelo de *machine learning* es el análisis de los datos que se van a procesar. Este conjunto de datos denominado *dataset* debe ser de gran volumen para poder entrenar el modelo de *machine learning* que se creará y configurará.

En este trabajo se va a estudiar la aplicación de un modelo de *machine learning* capaz de predecir el consumo eléctrico de un edificio. Para ello, se van a emplear varios *datasets* de lecturas de contadores eléctricos de la provincia de Málaga para predecir el valor de la lectura de energía activa, es decir, la energía consumida. [40]

Un ejemplo de los *dataset* que se han empleado es:



	A	B	C	D	E	F	G	H	I	J
1	ID_NODO	ID_DIA	FECHA_TOMA_DATO	ENERGIA_ACTIVA_ACUMULADA	LECTURA_ENERGIA_ACTIVA	ENERGIA_REACTIVA_ACUMULADA	LECTURA_ENERGIA_REACTIVA	POTENCIA_TOTAL_ACUMULADA	LECTURA_POTENCIA_TOTAL	
2	542	31	31/07/2016 23:14	39.625.209.375	428.125	88.854.796.875	59375	174.148.515.625	7.927.734.375	
3	543	31	31/07/2016 23:14	4.830.819.921.875	169.321.875	3.483.019.921.875	109.765.625	813.306.103.515.625	29.310.546.875	
4	544	31	31/07/2016 23:14	2.274.639.951.171.870	0	405.178.984.375	109.765.625	4354	21	
5	545	31	31/07/2016 23:14	0	0	0	0	0	0	
6	546	31	31/07/2016 23:14	0	0	0	0	0	0	
7	547	31	31/07/2016 23:14	2.366.612.796.875	201.953.125	1.963.240.193.359.370	138.828.125	112.928.687.255.859.000	618.421.142.578.125	
8	548	31	31/07/2016 23:14	0	0	0	0	0	0	
9	549	31	31/07/2016 23:14	0	0	0	0	0	0	
10	550	31	31/07/2016 23:14	0	0	0	0	0	0	
11	551	31	31/07/2016 23:14	0	0	0	0	0	0	

Figura 4: Ejemplo de dataset.

Los campos del conjunto de datos que vienen informados son:

- **ID_NODO:** Identificador del nodo del cuadro eléctrico.
- **ID_DIA:** Identificador del día de la toma (número de día de mes)
- **FECHA_TOMA_DATO**: Fecha y hora de la toma del dato.
- **ENERGIA_ACTIVA_ACUMULADA:** Energía activa acumulada.
- **LECTURA_ENERGIA_ACTIVA:** Energía activa en el momento de la toma.
- **ENERGIA_REACTIVA_ACUMULADA:** Energía reactiva acumulada.
- **LECTURA_ENERGIA_REACTIVA:** Energía reactiva en el momento de la toma.
- **POTENCIA_TOTAL_ACUMULADA:** Acumulado de la potencia total manejada por el cuadro (expresada en W).
- **LECTURA_POTENCIA_TOTAL:** Potencia total manejada por el cuadro (expresada en W).

Estos datos se han preparado para que los pudiera emplear el modelo de *machine learning*. Por lo tanto, para poder validar la preparación se ha filtrado por uno de los nodos, en este caso el **101914** para poder comprobar su predicción, así pues, la importación del *dataset* se ha realizado con la siguiente sentencia:

```
df = pd.read_csv('./nodo_101914.csv', sep=';', encoding='utf-8')
```

Una vez cargado, se comprueban las primeras filas con el comando:

```
df.head()
```

	ID_NODO	ID_DIA	FECHA_TOMA_DATO	ENERGIA_ACTIVA_ACUMULADA	LECTURA_ENERGIA_ACTIVA	ENERGIA_REACTIVA_ACUMULADA	LECTURA_ENERGI
0	101914	1	01/01/2016 0:00	2.185159e+10	2.406250e+05	1.971071e+10	1
1	101914	1	01/01/2016 0:15	2.185181e+10	2.187500e+04	1.971086e+10	
2	101914	1	01/01/2016 0:30	2.185204e+10	2.312500e+04	1.971101e+10	
3	101914	1	01/01/2016 0:45	2.185213e+11	8.906250e-01	1.971107e+11	
4	101914	1	01/01/2016 1:00	2.185229e+10	1.609375e+06	1.971117e+11	

Figura 5: Cabecera del dataset

Como se puede observar en la imagen, se toman datos cada quince minutos.

A partir del siguiente comando se puede conocer la información y los tipos de campos que componen el *dataset*:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34941 entries, 0 to 34940
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID_NODO                    34941 non-null  int64
 1   ID_DIA                     34941 non-null  int64
 2   FECHA_TOMA_DATO            34941 non-null  object
 3   ENERGIA_ACTIVA_ACUMULADA   34941 non-null  float64
 4   LECTURA_ENERGIA_ACTIVA     34941 non-null  float64
 5   ENERGIA_REACTIVA_ACUMULADA 34941 non-null  float64
 6   LECTURA_ENERGIA_REACTIVA   34941 non-null  float64
 7   POTENCIA_TOTAL_ACUMULADA   34941 non-null  float64
 8   LECTURA_POTENCIA_TOTAL     34941 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 2.4+ MB
```

Figura 6: Información del dataset

La columna **FECHA_TOMA_DATO** contiene datos categóricos por lo que es necesario asignarle el formato correspondiente para que el proceso reconozca los datos como formato fecha. Para ello, es necesario ejecutar el siguiente comando:

```
df["FECHA_TOMA_DATO"] = pd.to_datetime(df["FECHA_TOMA_DATO"],format='%d/%m/%Y %H:%M')
```

Figura 7: Asignación de formato de fecha

De esta forma a la columna **FECHA_TOMA_DATO** se le aplica el formato de fecha y el proceso reconoce sus datos:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34941 entries, 0 to 34940
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID_NODO                    34941 non-null  int64
 1   ID_DIA                     34941 non-null  int64
 2   FECHA_TOMA_DATO            34941 non-null  datetime64[ns]
 3   ENERGIA_ACTIVA_ACUMULADA   34941 non-null  float64
 4   LECTURA_ENERGIA_ACTIVA     34941 non-null  float64
 5   ENERGIA_REACTIVA_ACUMULADA 34941 non-null  float64
 6   LECTURA_ENERGIA_REACTIVA   34941 non-null  float64
 7   POTENCIA_TOTAL_ACUMULADA   34941 non-null  float64
 8   LECTURA_POTENCIA_TOTAL     34941 non-null  float64
dtypes: datetime64[ns](1), float64(6), int64(2)
memory usage: 2.4 MB
```

Figura 8: Información del dataset con el formato de fecha aplicado

A continuación, se analiza las relaciones que hay entre las variables para poder conocer qué datos serán útiles y cuáles no se podrán emplear en el método.

- Gráficos de dispersión:
  A partir de los gráficos de dispersión se obtendrá la relación existente entre las dos variables que se están analizando, es decir, la dependencia de una variable, en este caso la del eje Y, respecto a la otra variable del eje X.
  En este caso, se han analizado la relación de las siguientes variables:

```
# Gráfico dispersión
# ==============================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.ENERGIA_ACTIVA_ACUMULADA, y=df.LECTURA_ENERGIA_ACTIVA, alpha= 0.8)
ax.set_xlabel('Energía activa acumulada')
ax.set_ylabel('Lectura energía activa');
```

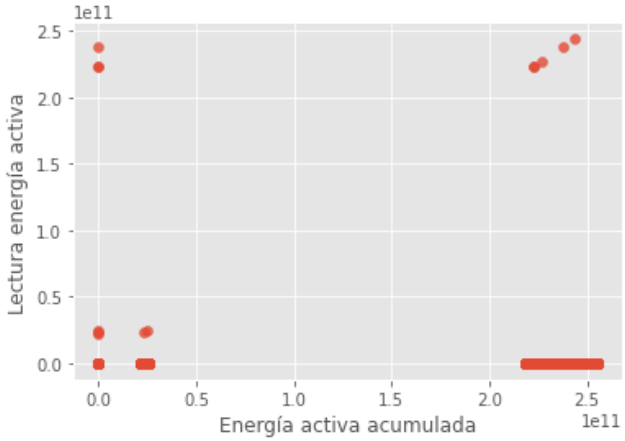


Figura 9: Gráfico de dispersión (Lectura energía activa/Energía activa acumulada)

Como se puede observar en la Figura 9, el gráfico de dispersión de las variables Lectura energía activa y Energía activa acumulada indica que estas variables son independientes entre sí. Puesto que en la mayoría de datos, una de las variables tiene valor cero mientras que la otra variable tiene valor distinto de cero.

```
# Gráfico dispersión
# ==============================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.POTENCIA_TOTAL_ACUMULADA, y=df.LECTURA_POTENCIA_TOTAL, alpha= 0.8)
ax.set_xlabel('Potencia total acumulada')
ax.set_ylabel('Lectura potencia total');
```

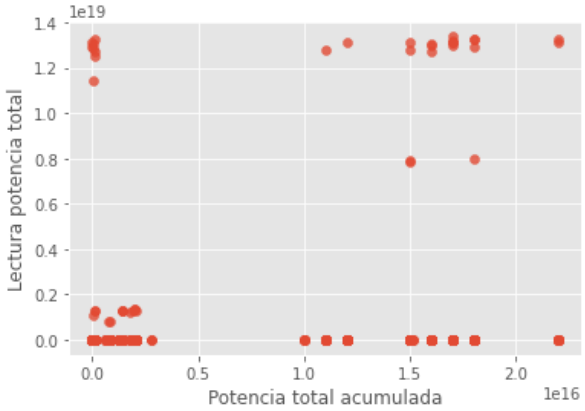


Figura 10: Gráfico de dispersión (Lectura potencia total/Potencia total acumulada)

Al igual que la gráfica anterior, entre las variables Lectura potencia total y la Potencia total acumulada no hay ninguna relación como se puede observar en el gráfico de dispersión.

```
# Gráfico dispersión
# ==============================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.LECTURA_POTENCIA_TOTAL, y=df.LECTURA_ENERGIA_ACTIVA, alpha= 0.8)
ax.set_xlabel('Lectura potencia total')
ax.set_ylabel('Lectura energía activa');
```

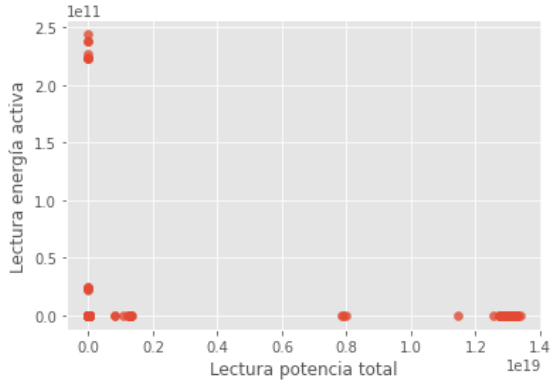


Figura 11: Gráfico de dispersión (Lectura energía activa/Lectura potencia total)

En este gráfico de dispersión de las variables de la Lectura energía activa y Lectura potencia total se aprecia que no hay ningún tipo de relación, puesto que cuando una de las variables tiene valor cero, la otra tiene valor distinto de cero.

```
# Gráfico dispersión
# ==============================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.POTENCIA_TOTAL_ACUMULADA, y=df.ENERGIA_ACTIVA_ACUMULADA, alpha= 0.8)
ax.set_xlabel('Potencia total acumulada')
ax.set_ylabel('Energía activa acumulada');
```

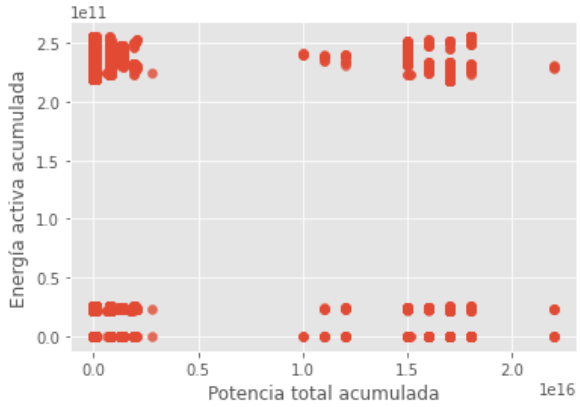


Figura 12: Gráfico de dispersión (Energía activa acumulada/Potencia total acumulada)

En este gráfico de dispersión entre las variables Energía activa acumulada y Potencia total acumulada hay una mínima relación lineal aunque como se puede observar no hay ninguna dependencia.

- Correlación:
  Otro método para analizar la relación de las variables es la correlación. Este método indica la asociación que hay entre dos variables numéricas y evalúa su tendencia en los datos.
  Para ello, se obtendrá la relación entre las variables numéricas (float64) donde se ha creado una nueva variable con los datos llamada **datos**.

```
# Variables numéricas tipo float64
datos = df.select_dtypes(include=['float64'])
```

```
# Matriz de correlación
# ==============================================================================
corr_matrix = datos.corr(method='pearson')
corr_matrix
```

	ENERGIA_ACTIVA_ACUMULADA	LECTURA_ENERGIA_ACTIVA	ENERGIA_REACTIVA_ACUMULADA	LECTURA_ENERGIA_REAC
ENERGIA_ACTIVA_ACUMULADA	1.000000	0.005008	0.010565	-0.004
LECTURA_ENERGIA_ACTIVA	0.005008	1.000000	-0.009165	0.434
ENERGIA_REACTIVA_ACUMULADA	0.010565	-0.009165	1.000000	0.000
LECTURA_ENERGIA_REACTIVA	-0.004278	0.434320	0.000686	1.000
POTENCIA_TOTAL_ACUMULADA	0.030896	-0.003226	0.005183	-0.002
LECTURA_POTENCIA_TOTAL	0.007511	-0.000471	-0.001511	-0.000

Figura 13: Datos numéricos y correlación

Para que la matriz de correlación sea más visual, se ha generado la siguiente gráfica:

```
# Heatmap matriz de correlaciones
# ==============================================================================
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5))

sns.heatmap(
    corr_matrix,
    annot     = True,
    cbar      = False,
    annot_kws = {"size": 8},
    vmin      = -1,
    vmax      = 1,
    center    = 0,
    cmap      = sns.diverging_palette(20, 220, n=200),
    square    = True,
    ax        = ax
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 10)
```

Figura 14: Matriz de correlación

Con estas gráficas se puede observar que la lectura de energía activa, la variable que queremos predecir, está relacionada con la lectura de energía reactiva. Sin embargo, con las demás variables no tiene ningún tipo de relación.

Para poder tener más detalle del conjunto de datos que contiene el *dataset* se ha observado su distribución de los datos de Lectura de energía activa, la variable que se pretende predecir, y los datos de la Lectura de potencia:

```
# Gráfico distribución variables
# ==============================================================================
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

axs[0].hist(x=df.LECTURA_ENERGIA_ACTIVA, bins=130, color="#3182bd", alpha=0.5)
axs[0].plot(df.LECTURA_ENERGIA_ACTIVA, np.full_like(df.LECTURA_ENERGIA_ACTIVA, -0.01), '|k', markeredgewidth=1)
axs[0].set_title('Distribución lectura energía activa')
axs[0].set_xlabel('Lectura energía activa')
axs[0].set_ylabel('counts')

axs[1].hist(x=df.LECTURA_POTENCIA_TOTAL, bins=130, color="#3182bd", alpha=0.5)
axs[1].plot(df.LECTURA_POTENCIA_TOTAL, np.full_like(df.LECTURA_POTENCIA_TOTAL, -0.01), '|k', markeredgewidth=1)
axs[1].set_title('Distribución lectura potencia total')
axs[1].set_xlabel('Lectura potencia total')
axs[1].set_ylabel('counts')


plt.tight_layout();
```

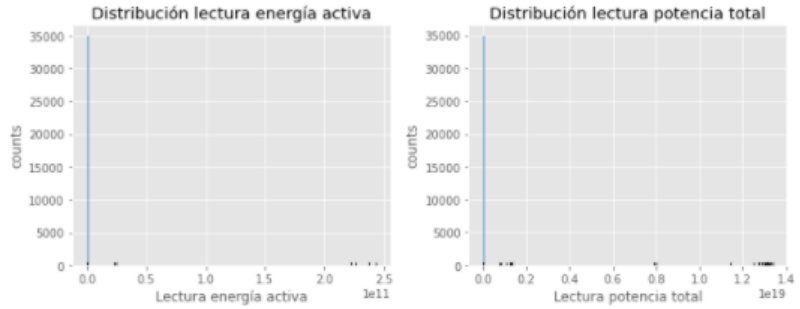


Figura 15: Distribución de los datos de lectura de energía activa y potencia

Como conclusión, podemos afirmar que un número elevado de datos de lectura de energía activa tiene valor cero, esto es debido a que muchas de las lecturas realizadas tienen valor cero ya que el edificio no está consumiendo energía como por ejemplo, un fin de semana.

4. Diseño del producto

Una vez los datos se han preparado y validado, ahora ya se pueden procesar en los modelos de *machine learning*.

Dentro de los modelos de *machine learning* que se han descrito en el apartado de Estado del arte, hay dos agrupaciones, aprendizaje supervisado y aprendizaje no supervisado.

En este trabajo se ha decidido ejecutar un modelo de cada tipo para analizar y estudiar el comportamiento de la predicción. Así se puede comprobar que método es más adecuado para la predicción de este conjunto de datos.

En el aprendizaje no supervisado se ha empleado el modelo de *Random Forest* y para el aprendizaje supervisado se ha empleado el modelo de SARIMA (Seasonal Auto Regressive Integrated Moving Average).

4.1. Random Forest

Esta técnica de aprendizaje automático tiene una capacidad de generalización muy alta para errores, es decir, el proceso aprende muy bien los datos de entrenamiento, pero su predicción no es tan buena.

Como se ha comentado en el anterior apartado, los datos de energía reactiva tenían una correlación alta con los datos de la energía activa, por lo tanto, se han quitado del conjunto de datos con los que se va a entrenar.

```
num_df = df.select_dtypes(include=['float64']).copy().drop(['ENERGIA_REACTIVA_ACUMULADA','LECTURA_ENERGIA_REACTIVA'],axis=1) # S
num_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34941 entries, 0 to 34940
Data columns (total 4 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ENERGIA_ACTIVA_ACUMULADA  34941 non-null  float64
 1   LECTURA_ENERGIA_ACTIVA    34941 non-null  float64
 2   POTENCIA_TOTAL_ACUMULADA  34941 non-null  float64
 3   LECTURA_POTENCIA_TOTAL    34941 non-null  float64
dtypes: float64(4)
memory usage: 1.1 MB
```

Figura 16: Configuración del dataset para el modelo Random Forest

Por tanto, los datos que se van a entrenar tienen las siguientes características:

```
num_df.describe()
```

	ENERGIA_ACTIVA_ACUMULADA	LECTURA_ENERGIA_ACTIVA	POTENCIA_TOTAL_ACUMULADA	LECTURA_POTENCIA_TOTAL
count	3.494100e+04	3.494100e+04	3.494100e+04	3.494100e+04
mean	1.033829e+11	5.654223e+07	5.135042e+14	1.043079e+16
std	1.084310e+11	3.488657e+09	2.562156e+15	3.553647e+17
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	2.241414e+10	0.000000e+00	1.500000e+01	0.000000e+00
50%	2.442226e+10	0.000000e+00	1.295882e+11	0.000000e+00
75%	2.325172e+11	1.203125e+06	8.268104e+13	2.000000e+00
max	2.551367e+11	2.438058e+11	2.202271e+16	1.338028e+19

Figura 17: Descripción del dataset configurado para Random Forest

Se preparan los datos para que puedan ser aplicados en el modelo:

```
X = np.array(num_df.drop(['LECTURA_ENERGIA_ACTIVA'],1))
y = np.array(num_df['LECTURA_ENERGIA_ACTIVA'])
X.shape
```

```
(34941, 3)
```

```
lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(y)
print(utils.multiclass.type_of_target(y))
print(utils.multiclass.type_of_target(y.astype('int')))
print(utils.multiclass.type_of_target(encoded))
```

```
continuous
multiclass
multiclass
```

Figura 18: Preparación de los datos para Random Forest

El modelo de *Random Forest* se ha configurado con 120 árboles porque era el número máximo que permitía el proceso, ya que cuanto mayor número de árboles más aproximado y mejor puede ser la predicción:

```
from sklearn.ensemble import RandomForestClassifier
# El modelo se ha creado con 120 árboles
model = RandomForestClassifier(n_estimators=120,
                               bootstrap = True, verbose=2,
                               max_features = 'auto')
# Entrenamiento
model.fit(X, encoded)
```

```
building tree 105 of 120
building tree 106 of 120
building tree 107 of 120
building tree 108 of 120
building tree 109 of 120
building tree 110 of 120
building tree 111 of 120
building tree 112 of 120
building tree 113 of 120
building tree 114 of 120
building tree 115 of 120
building tree 116 of 120
building tree 117 of 120
building tree 118 of 120
building tree 119 of 120
building tree 120 of 120
[Parallel(n_jobs=1)]: Done 120 out of 120 | elapsed:   18.3s finished
RandomForestClassifier(n_estimators=120, verbose=2)
```

Figura 19: Modelado de Random Forest

Se comprueba la calificación del modelo de entrenamiento que es de un 0.97:

```
model.score(X,encoded)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 120 out of 120 | elapsed:    5.4s finished
0.9691479923299275
```

Figura 20: Calificación de Random Forest

Es necesario validar el modelo que se ha entrenado para ello, se ejecutará un 20% de los datos:

```
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, encoded, test_size=validation_size, random_st
```

Figura 21: Validación del modelado de Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# Validación
model.fit(X_train, Y_train)
```

```
building tree 105 of 120
building tree 106 of 120
building tree 107 of 120
building tree 108 of 120
building tree 109 of 120
building tree 110 of 120
building tree 111 of 120
building tree 112 of 120
building tree 113 of 120
building tree 114 of 120
building tree 115 of 120
building tree 116 of 120
building tree 117 of 120
building tree 118 of 120
building tree 119 of 120
building tree 120 of 120
[Parallel(n_jobs=1)]: Done 120 out of 120 | elapsed:   13.0s finished
RandomForestClassifier(n_estimators=120, verbose=2)
```

Figura 22: Ejecución de la validación del modelado

Se comprueban las predicciones que ha generado con el 20% de los datos:

```
predictions = model.predict(X_validation)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 120 out of 120 | elapsed:    1.1s finished
```

Figura 23: Predicción de la validación de Random Forest

```
print(classification_report(Y_validation, predictions))
```

111	0.00	0.00	0.00	0
112	0.00	0.00	0.00	2
113	0.00	0.00	0.00	1
114	0.00	0.00	0.00	2
115	0.00	0.00	0.00	1
116	0.00	0.00	0.00	0
117	0.00	0.00	0.00	1
118	0.00	0.00	0.00	2
119	0.00	0.00	0.00	1
120	0.00	0.00	0.00	1
122	0.00	0.00	0.00	0
128	0.00	0.00	0.00	1
129	0.00	0.00	0.00	0
accuracy			0.55	6989
macro avg	0.04	0.03	0.03	6989
weighted avg	0.51	0.55	0.53	6989

Figura 24: Clasificación de la validación de Random Forest

Se obtiene una media ponderada de precisión de un 51%, por lo tanto, este algoritmo no se puede aplicar para este caso.

Esto es debido a que, como se ha visto en la distribución de los datos, la mayoría de los datos que se pretender predecir tienen valor cero, por lo tanto,

no es real que el 51% de la predicción sea correcta puesto que los únicos valores que ha predicho bien son los que tienen valor cero. Así pues, este modelo no es válido para predecir el consumo eléctricos en edificios.

4.2. SARIMA

Como se ha explicado en el anterior modelo, la mayoría de los datos tienen valor cero, por lo tanto, se ha decidido emplear en el modelo de tipo de aprendizaje supervisado el modelo SARIMA (Seasonal Auto Regressive Integrated Moving Average) puesto que del consumo eléctrico se puede obtener una serie a partir de los hábitos que se realizan en el edificio.

Los modelos SARIMA son modelos ARIMA (Auto Regressive Integrated Moving Average, en español, media móvil integrado autorregresiva) con un componente estacional.

Se trata de un modelo de regresión lineal que utiliza sus propios desfases como predictores. Los modelos de regresión lineal funcionan mejor cuando los predictores no están correlacionados y son independientes entre sí.

Según la fórmula **SARIMA(p,d,q)x(P,D,Q,s),** los parámetros de este tipo de modelos son los siguientes: [41]

- **p** y **P** estacional: indican el número de términos autorregresivos (desfases de la serie estacionaria)
- **d** y **D** estacional: indican la diferenciación que hay que hacer para estacionar la serie
- **q** y **Q** estacional: indican el número de términos de media móvil (desfases de los errores de previsión)
- **s**: indica la longitud estacional de los datos, en este caso será 15 porque las muestras son cada 15 minutos.

Por lo tanto, eliminando las columnas del identificador del nodo y del número del día, puesto que estos valores no aportan nada a la predicción, y estableciendo como índice del conjunto de datos la fecha de las lecturas registradas:

```
datos = df.drop(['ID_NODO','ID_DIA'],axis=1)
datos.set_index("FECHA_TOMA_DATO")
```

FECHA_TOMA_DATO	ENERGIA_ACTIVA_ACUMULADA	LECTURA_ENERGIA_ACTIVA	ENERGIA_REACTIVA_ACUMULADA	LECTUR
2016-01-01 00:00:00	2.185150e+10	2.406250e+05	1.971071e+10	
2016-01-01 00:15:00	2.185181e+10	2.187500e+04	1.971088e+10	
2016-01-01 00:30:00	2.185204e+10	2.312500e+04	1.971101e+10	
2016-01-01 00:45:00	2.185213e+11	8.906250e-01	1.971107e+11	
2016-01-01 01:00:00	2.185229e+10	1.609375e+06	1.971117e+11	
...	...	...	...	
2016-12-31 21:46:00	2.551283e+11	2.593750e+05	2.191248e+11	
2016-12-31 22:01:00	2.551311e+10	2.796875e+06	2.191265e+05	
2016-12-31 22:16:00	2.551343e+11	3.203125e+06	2.191284e+10	
2016-12-31 22:31:00	2.551367e+11	2.406250e+05	2.191299e+10	
2016-12-31 22:46:00	2.551394e+10	2.703125e+06	2.191315e+05	

34941 rows × 6 columns

Figura 25: Configuración del dataset para el modelo SARIMA

Para tener más detalle de la lectura de energía activa que hay que predecir se ha representado los primeros días de consumo:

```
plt.figure(figsize=(20, 10), dpi=150)
plt.plot(datos.FECHA_TOMA_DATO[:2000], datos.LECTURA_ENERGIA_ACTIVA[:2000], color='tab:blue')
plt.gca().set(title='Lectura de energía activa',
xlabel='Fecha', ylabel='Vatios (W)')
plt.show()
```

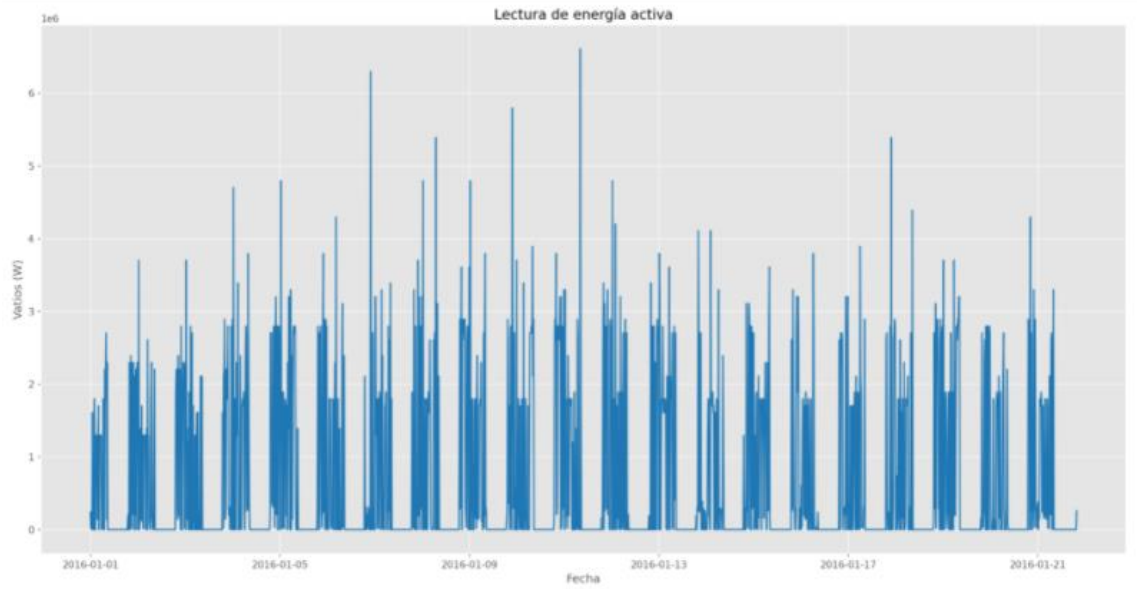


Figura 26: Representación de los primeros días de consumo

Para poder escoger los mejores parámetros del modelo de SARIMA para la predicción de la variable de lectura de energía activa se han configurado los rangos de los posibles valores de los parámetros:

```
import itertools

p = range(1,3)
q = range(1,3)
d = range(1,3)
#generate patterns from p,q,r
pdq = list(itertools.product(p, d, q))

seasonal_pdq = [(x[0], x[1], x[2], 15) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for SARIMA..')
print('SARIMAX: {} * {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} * {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} * {}'.format(pdq[2], seasonal_pdq[2]))
print('SARIMAX: {} * {}'.format(pdq[3], seasonal_pdq[2]))
```

```
Examples of parameter combinations for SARIMA..
SARIMAX: (1, 1, 2) * (1, 1, 2, 15)
SARIMAX: (1, 1, 2) * (1, 1, 2, 15)
SARIMAX: (1, 2, 1) * (1, 2, 1, 15)
SARIMAX: (1, 2, 2) * (1, 2, 1, 15)
```

Figura 27: Configuración de rangos para los parámetros de SARIMA

Una vez establecidos los rangos de los posibles valores de los parámetros, se ejecutará la lógica necesaria que calculará cual de todas las posibles combinaciones tiene un menor coste, por lo tanto, a partir de esta medición se podrá obtener la mejor opción para poder predecir el consumo eléctrico.

El código que se ha ejecutado para obtener las mediciones anteriormente mencionadas es:

```
import warnings
import statsmodels.api as sm

warnings.filterwarnings("ignore")
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(
                datos.LECTURA_ENERGIA_ACTIVA,
                order=param, seasonal_order=param_seasonal,
                enforce_stationarity=False,
                enforce_invertibility=False)
            results = mod.fit()
            print('SARIMA{}*{}15 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

Puesto que el resultado de esta ejecución es muy extenso se ha extraído y filtrado en un fichero aparte. A partir de estos datos, se ha obtenido las quince combinaciones que poseen un menor coste para poder establecer cuál es la mejor opción para poder realizar la predicción:

1. SARIMA(1, 1, 2)*(2, 1, 1, 15)15 - AIC:535856.592608996
2. SARIMA(0, 0, 2)*(0, 0, 2, 15)15 - AIC:1625701.7067387768
3. SARIMA(2, 0, 2)*(0, 0, 2, 15)15 - AIC:1625702.874615015
4. SARIMA(0, 0, 2)*(1, 0, 2, 15)15 - AIC:1625703.706737965
5. SARIMA(0, 0, 2)*(1, 0, 2, 15)15 - AIC:1625703.706737965

6. SARIMA(2, 0, 2)*(1, 0, 2, 15)15 - AIC:1625704.874604205
7. SARIMA(0, 0, 2)*(2, 0, 2, 15)15 - AIC:1625705.7067380133
8. SARIMA(1, 0, 2)*(1, 0, 2, 15)15 - AIC:1625705.7067596703
9. SARIMA(2, 0, 2)*(2, 0, 2, 15)15 - AIC:1625706.8745920188
10. SARIMA(1, 0, 2)*(2, 0, 2, 15)15 - AIC:1625707.7067914524
11. SARIMA(2, 0, 2)*(2, 0, 0, 15)15 - AIC:1625748.4497396993
12. SARIMA(2, 0, 2)*(2, 0, 1, 15)15 - AIC:1625750.449720191
13. SARIMA(2, 0, 2)*(2, 1, 1, 15)15 - AIC:1631775.5314466937
14. SARIMA(1, 0, 2)*(2, 0, 1, 15)15 - AIC:1625796.8569953842
15. SARIMA(0, 0, 2)*(2, 0, 0, 15)15 - AIC:1625838.4321221502

De todas estas combinaciones se ejecutó su modelado, se realizó la predicción y se calculó su error cuadrático medio obteniéndose un mejor resultado la combinación 12 SARIMA(2, 0, 2)*(2, 0, 1, 15).

El resultado de la combinación 12 SARIMA(2, 0, 2)*(2, 0, 1, 15) es la siguiente:

```
mod = sm.tsa.statespace.SARIMAX(
    datos.LECTURA_ENERGIA_ACTIVA,
    order=(2, 0, 2), seasonal_order=(2, 0, 1, 15),
    enforce_stationarity=True,
    enforce_invertibility=False)
results = mod.fit()
results.summary()
```

SARIMAX Results

Dep. Variable:	LECTURA_ENERGIA_ACTIVA	**No. Observations:**	34941
Model:	SARIMAX(2, 0, 2)x(2, 0, [1], 15)	**Log Likelihood**	-813596.705
Date:	Sun, 26 Dec 2021	**AIC**	1627209.411
Time:	19:37:25	**BIC**	1627277.102
Sample:	0	**HQIC**	1627230.974
	- 34941		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	3.684e-05	0.068	0.001	1.000	-0.132	0.132
ar.L2	-0.0180	0.001	-25.636	0.000	-0.019	-0.017
ma.L1	0.0010	0.062	0.016	0.988	-0.120	0.122
ma.L2	0.5064	0.001	705.039	0.000	0.505	0.508
ar.S.L15	8.638e-05	0.302	0.000	1.000	-0.591	0.591
ar.S.L30	0.0002	0.099	0.003	0.998	-0.193	0.194
ma.S.L15	-5.565e-05	0.302	-0.000	1.000	-0.591	0.591
sigma2	9.882e+18	1.68e-18	5.87e+36	0.000	9.88e+18	9.88e+18

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	24790957002.49
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.37	**Skew:**	55.43
Prob(H) (two-sided):	0.00	**Kurtosis:**	4128.04

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.67e+50. Standard errors may be unstable.

Figura 28: Modelado SARIMA mejor opción

La ejecución de su predicción de los seis últimos meses del año 2016:

```
plt.figure(figsize=(20, 10), dpi=150)
ax = datos.LECTURA_ENERGIA_ACTIVA[:27000].plot(label='Actual', color='tab:blue')
pred.predicted_mean[:20000].plot(ax=ax, label='Forecast', color='tab:red')

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.05)
ax.set_xlabel('Date')
ax.set_ylabel('Lectura energia activa')
plt.legend()
plt.xlim(0,10000)
plt.ylim(-1,7e6)
plt.show()
```

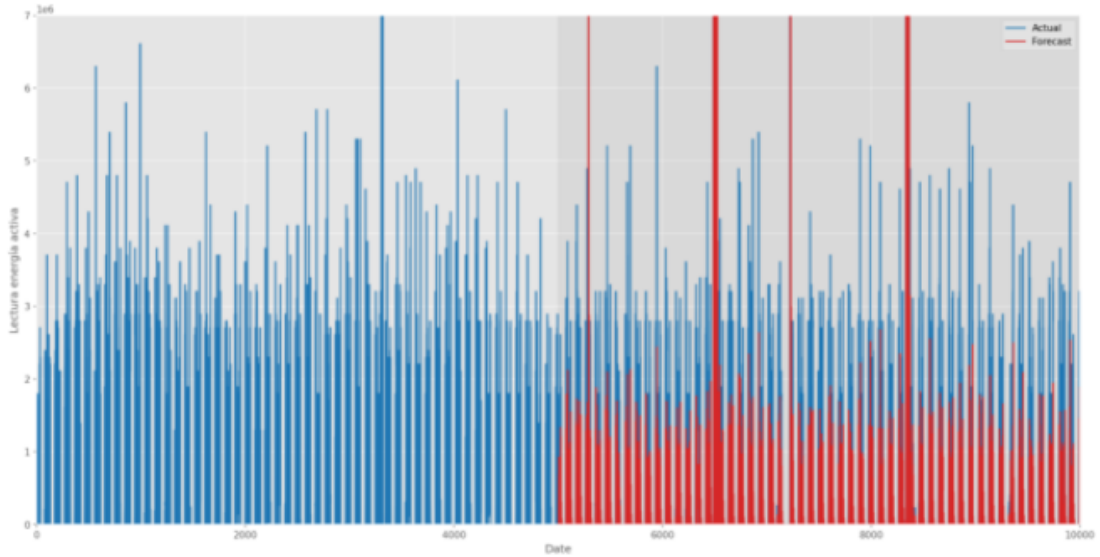


Figura 29: Predicción de la mejor combinación SARIMA

Realizando un aumento en los primeros días de la predicción se puede observar:

```
plt.figure(figsize=(20, 10), dpi=150)
ax = datos.LECTURA_ENERGIA_ACTIVA[4000:8000].plot(label='Actual', color='tab:blue')
pred.predicted_mean[:3000].plot(ax=ax, label='Forecast', color='tab:red')

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.05)
ax.set_xlabel('Date')
ax.set_ylabel('Lectura energia activa')
plt.legend()
plt.xlim(4000,8000)
plt.ylim(-1,7e6)
plt.show()
```

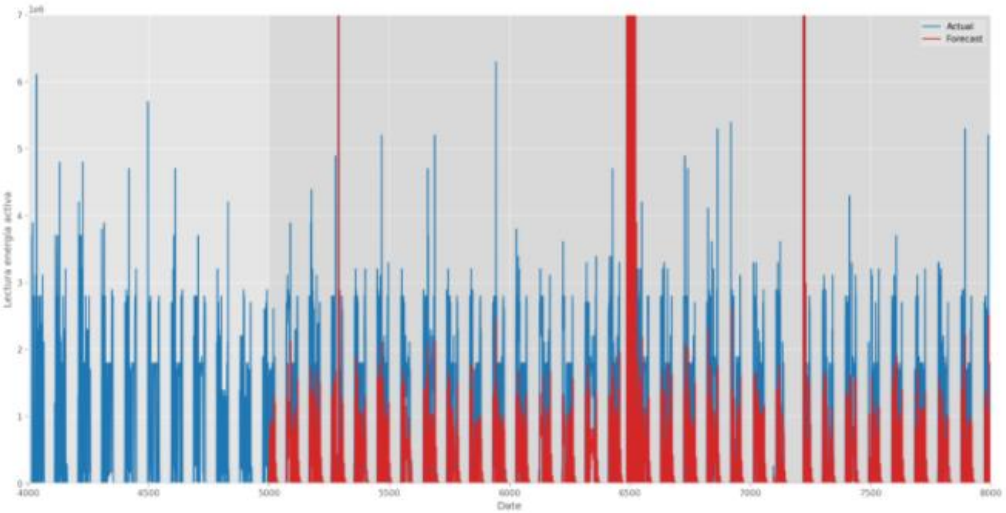


Figura 30: Predicción aumentada mejor opción SARIMA

A continuación, se calcula el error cuadrático medio para conocer el promedio de los errores al cuadrado, es decir, la diferencia entre los valores del conjunto de datos y la predicción:

```
from sklearn.metrics import mean_squared_error
from numpy import sqrt
y_forecasted = pred.predicted_mean
y_truth = datos.LECTURA_ENERGIA_ACTIVA[5000:]
rmse = sqrt(mean_squared_error(y_truth, y_forecasted).mean())
print('The RMSE error of forecast prediction is {}'.format(round(rmse, 2)))
```

```
The RMSE error of forecast prediction is 2657939644.15
```

Figura 31: Error cuadrático medio de la mejor combinación SARIMA

El error es alto debido a que el valor de las predicciones de la lectura de energía activa tiene menos valor que los valores reales. Aunque hay que destacar que el modelo predice correctamente las lecturas intermedias donde tienen valor cero. Por lo tanto, este modelo ajustando las configuraciones que se han comentado anteriormente sería válido para predecir el valor de las lecturas de energía activa, es decir, el consumo eléctrico.

Para poder realizar la comparación con el mejor resultado, la combinación 12, se mostrará el resultado de las anteriores ejecuciones con la combinación que tiene el menor coste, en este caso, SARIMA(1, 1, 2)*(2, 1, 1, 15).

Se configura el modelado y se establece que la predicción comience en la posición 3000:

```
mod = sm.tsa.statespace.SARIMAX(
    datos.LECTURA_ENERGIA_ACTIVA,
    order=(1, 1, 2), seasonal_order=(2, 1, 1, 15),
    enforce_stationarity=False,
    enforce_invertibility=False)
results = mod.fit()
```

```
pred = results.get_prediction(start=3000,
dynamic=False, full_results=True)
pred_ci = pred.conf_int()
```

Figura 32: Configuración de otra opción SARIMA

En la siguiente gráfica se puede observar que la predicción de esta combinación no es correcta, aunque su coste sea menor:

```
plt.figure(figsize=(20, 10), dpi=150)
ax = datos.LECTURA_ENERGIA_ACTIVA[:10000].plot(label='Actual', color='tab:
pred.predicted_mean[:7000].plot(ax=ax, label='Forecast', color='tab:red')

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.05)
ax.set_xlabel('Date')
ax.set_ylabel('Lectura energía activa')
plt.legend()
plt.xlim(0,10000)
plt.ylim(-1,7e6)
plt.show()
```

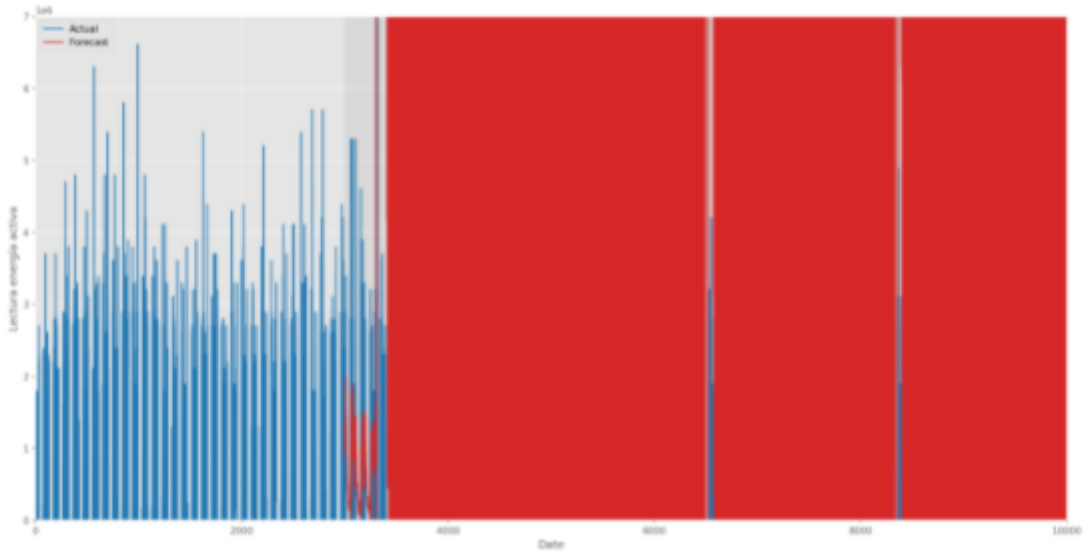


Figura 33: Predicción de otra opción SARIMA

Como se pueden observar en las figuras anteriores, aunque es un buen método escoger la combinación con menor coste a veces no es la mejor opción para realizar la predicción de datos, como es en este caso, el consumo eléctrico.

5. Conclusiones

Las conclusiones que se han obtenido en este trabajo es que no todos los aprendizajes o modelos que existen en *machine learning* se pueden aplicar para realizar la predicción de un conjunto de datos. En este caso, como es la predicción del consumo eléctrico donde hay un gran número de valores del conjunto de datos con valor cero, hay muchos modelos que no se pueden aplicar puesto que se sesga la información y la predicción es incorrecta ocasionándose que siempre se prediga el valor cero, por lo tanto, un gran porcentaje de datos lo realiza correctamente pero no es capaz de los valores.

Además, el conjunto de datos es necesario tratarlo y establecer su formato correspondiente para que se pueda aplicar la lógica de los modelos. También es necesario realizar un análisis previo de la información que se va a tratar puesto que hay datos que dependen unos de otros y puede sesgar el aprendizaje del modelo.

De los objetivos que se han planteado inicialmente no se ha podido cumplir la última parte planteada que era el realizar una simulación con un visionado de los datos puesto que ha dedicado mucho más tiempo a encontrar un modelo que pudiera predecir mejor los valores del consumo energético ya que como se ha comentado, no todos los modelos se pueden aplicar y este conjunto de datos posee un gran número de datos con valor cero.

El seguimiento de la planificación se ha realizado correctamente, siempre se han realizado las entregas a tiempo. Hubiera sido mejor haber podido realizar las entregas con más tiempo para poderlas revisarlas con más detalle con el tutor.

Las líneas de trabajo futuro sería poder realizar un cruce de este modelo con datos meteorológicos para poder mejorar la predicción de consumo, puesto que de un año a otro dependiendo de la meteorología el consumo puede cambiar.

6. Glosario

Machine Learning: aprendizaje automático.
Smart Cities: ciudad inteligente.
Smart Buildings: edificios inteligentes.
Dataset: conjunto de datos.
Random Forest: modelo de machine learning que se conocen como bosques aleatorios.
SARIMA: modelo de machine learning que se trata de una extensión estacional del modelo de ARIMA.
ARIMA: modelo de machine learning que se basa en modelo dinámicos que emplea series temporales.

7. Bibliografía

[1] March Corbella, Hug, “The smart city and other ICT-led techno-imaginaries: Any room for dialogue with degrowth?”, http://openaccess.uoc.edu/webapps/o2/handle/10609/112946

[2] Vidal Tejedor, Narcís, “La smart city: las ciudades inteligentes del futuro”, 1ª Edición, Editorial UOC, Barcelona, 2015.

[3]https://www.idae.es/uploads/documentos/documentos_Borrador_Smart_Cities_18_Abril_2012_b97f8b15.pdf (01/10/2021)

[4]https://www.universidadviu.com/es/actualidad/nuestros-expertos/smart-mobility-concepto-y-principios (02/10/2021)

[5]https://web.ua.es/smart/smart-people-comunidad-senspeople.html (02/10/2021)

[6]https://web.ua.es/es/smart/smart-economy-economia-inteligente.html (02/10/2021)

[7]https://web.ua.es/es/smart/smart-environment-un-entorno-de-calidad-de-vida.html (02/10/2021)

[8]https://web.ua.es/es/smart/smart-government-gobernanza-del-futuro.html (02/10/2021)

[9]https://web.ua.es/es/smart/smart-living-microentorno-de-calidad.html (02/10/2021)

[10]https://www.cisco.com/c/en/us/solutions/smart-building/what-is-a-smart-building.html (04/10/2021)

[11]https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0059483 (10/10/2021)

[12] Nicolas Havard, Sean McGrath, Colin Flanagan, Ciaran MacNamee; “Smart Building Based on Internet of Things Technology”; Published in 2018 12th International Conference on Sensing Technology (ICST); IEEE Xplore.

[13] Deepti Sehrawat, Nasib Singh Gill; “Smart Sensors: Analysis of Different Types of IoT Sensors”; Published in 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI); IEEE Xplore

[14] Manoj Kumar Sah; D.K. Gupta; Pooja Rani; “Energy Efficient Routing Protocol for Wireless Sensor Networks with Multiple Sinks”; Published in: 2015 Second International Conference on Advances in Computing and Communication Engineering; IEEE Xplore

[15] W. Heinzelman and A. Chandrakasan, "Communication Protocol for Wireless Microsensor Networks", Published in: 33rd Annual Hawaii International Conference on System Sciences (HICSS), pp. 3005-3014, Jan. 2000.

[16] A. Manjeshwar and D. P. Agarwal, "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks", 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, 2001.

[17] P. Ding, J. Holliday and A. Celik, "Distributed Energyefficient hierarchical clustering for wireless sensor networks", Lecture notes in Computer Science, vol. 3560, pp. 322-339, 2005.

[18] S. Lindsey and C. Raghavendra, "PEGASIS PowerEfficient Gathering in Sensor Information Systems", IEEE Aerospace Conference Proceedings, vol. 3, pp. 1125-1130, 2002.

[19] Pietro Ciciriello and Luca Mottola, "An Efficient Routing Multiple Sources to Multiple Sinks", Wireless Sensor Networks Computer Science and Engineering, vol. 373, pp. 34-50, 2007.

[20]https://medium.com/@siddharth.parakh/the-complete-list-of-types-of-sensors-used-in-iot-63b4003ab6b3 (05/10/2021)

[21] K.T.V.GrattanT.SunDr. “Fiber optic sensor technology: an overview”; Sensors and Actuators A: Physical; Volume 82, Issues 1–3, 15 May 2000, Pages 40-61;

[22] Sourav Kumar Dhar; Suman Sankar Bhunia; Nandini Mukherjee. "Interference aware scheduling of sensors in IoT enabled health-care monitoring system", Published in: 2014 Fourth International Conference of Emerging Applications of Information Technology, IEEE Xplore

[23] Nashreen Nesa; Indrajit Banerjee “IoT-Based Sensor Data Fusion for Occupancy Sensing Using Dempster–Shafer Evidence Theory for Smart Buildings”; Published in: IEEE Internet of Things Journal (Volume: 4, Issue: 5, Oct. 2017), IEEE Xplore

[24] Aamir Nizam Ansari; Mohamed Sedky; Neelam Sharma; Anurag Tyagi; “An Internet of things approach for motion detection using Raspberry Pi” Published in: Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, IEEE Xplore

[25] Rahul B. Pendor; P. P. Tasgaonkar; “An IoT framework for intelligent vehicle monitoring system”; Published in: 2016 International Conference on Communication and Signal Processing (ICCSP), IEEE Xplore

[26] Guigang Zhang; Chao Li; Yong Zhang; Chunxiao Xing; Jijiang Yang; “SemanMedical: A kind of semantic medical monitoring system model based on the IoT sensors”; Published in: 2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom); IEEE Xplore

[27] Akshata Tapashetti; Divya Vegiraju; Tokunbo Ogunfunmi; “IoT-enabled air quality monitoring device: A low cost smart health solution”; Published in: 2016 IEEE Global Humanitarian Technology Conference (GHTC); IEEE Xplore

[28] N Vijayakumar; R Ramya; “The real time monitoring of water quality in IoT environment”; Published in: 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]; IEEE Xplore

[29] D. Pavithra; Ranjith Balakrishnan “IoT based monitoring and control system for home automation”; Published in: 2015 Global Conference on Communication Technologies (GCCT); IEEE Xplore

[30]https://medium.datadriveninvestor.com/what-is-machine-learning-55028d8bdd53 (08/10/2021)

[31] https://medium.com/datos-y-ciencia/introduccion-al-machine-learning-una-gu%C3%ADa-desde-cero-b696a2ead359 (08/10/2021)

[32] https://keras.io/ (10/10/2021)

[33] http://torch.ch/ (10/10/2021)

[34] https://caffe.berkeleyvision.org/ (10/10/2021)

[35] https://www.tensorflow.org/?hl=es-419 (10/10/2021)

[36] https://pypi.org/project/Theano/ (10/10/2021)

[37] https://docs.microsoft.com/en-us/cognitive-toolkit/ (10/10/2021)

[38] https://www.dropbox.com/features/cloud-storage (08/10/2021)

[39] https://www.oracle.com/es/database/what-is-a-cloud-database/ (08/10/2021)

[40]https://datosabiertos.malaga.eu/organization/medio-ambiente-y-sostenibilidad

[41]https://towardsdatascience.com/time-series-forecasting-with-a-sarima-model-db051b7ae459

8. Anexos

En los anexos se adjunta el código de Python que se ha ejecutado para poder realizar el análisis de la predicción del consumo energético de los edificios inteligentes.

8.1. Modelado Random Forest

```
#!/usr/bin/env python
# coding: utf-8

#Importación de las librerías necesarias
import pandas as pd
import numpy as np
from datetime import datetime
import seaborn as sb
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import utils
# Gráficos
#
================================================================================
========
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y análisis
#
================================================================================
========
import statsmodels.api as sm
from scipy import stats
from scipy.stats import pearsonr

# Configuración matplotlib
#
================================================================================
========
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from time import time
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA as sklearnPCA
from sklearn import preprocessing
from sklearn import datasets
from sklearn.cluster import MeanShift
from sklearn.cluster import AffinityPropagation
from sklearn.metrics.pairwise import pairwise_distances
```

```
from sklearn.cluster import KMeans
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report,
auc, precision_recall_curve, roc_curve

get_ipython().run_line_magic('matplotlib', 'inline')
RANDOM_SEED = 42

#Importación del dataset
df = pd.read_csv('./nodo_101914.csv', sep=';', encoding='utf-8')

#Comprobación de las primeras filas
df.head()

#Información del dataset. Se comprueban los tipos de datos que hay
(float64, enteros y categóricos)
df.info()

#Se comprueba la variable categórica de la fecha para poder validar su
formato
df["FECHA_TOMA_DATO"]

#Se configura el formato de la varible para que sea una fecha
df["FECHA_TOMA_DATO"] =
pd.to_datetime(df["FECHA_TOMA_DATO"],format='%d/%m/%Y %H:%M')

#Se comprueba que la variable tiene el formato correctamente aplicado
df["FECHA_TOMA_DATO"]

#Se vuelve a extraer la información del dataset para validar que ya no
hay datos categóricos
df.info()

# Gráfico dispersión ENERGIA_ACTIVA_ACUMULADA/LECTURA_ENERGIA_ACTIVA
#
==========================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.ENERGIA_ACTIVA_ACUMULADA, y=df.LECTURA_ENERGIA_ACTIVA,
alpha= 0.8)
ax.set_xlabel('Energía activa acumulada')
ax.set_ylabel('Lectura energía activa');

# Gráfico dispersión POTENCIA_TOTAL_ACUMULADA/LECTURA_POTENCIA_TOTAL
#
==========================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.POTENCIA_TOTAL_ACUMULADA, y=df.LECTURA_POTENCIA_TOTAL,
alpha= 0.8)
ax.set_xlabel('Potencia total acumulada')
ax.set_ylabel('Lectura potencia total');

# Gráfico dispersión POTENCIA_TOTAL_ACUMULADA/LECTURA_POTENCIA_TOTAL
#
==========================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.POTENCIA_TOTAL_ACUMULADA, y=df.LECTURA_POTENCIA_TOTAL,
alpha= 0.8)
ax.set_xlabel('Potencia total acumulada')
ax.set_ylabel('Lectura potencia total');

# Gráfico dispersión LECTURA_POTENCIA_TOTAL/LECTURA_ENERGIA_ACTIVA
```

```
#
==========================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.LECTURA_POTENCIA_TOTAL, y=df.LECTURA_ENERGIA_ACTIVA,
alpha= 0.8)
ax.set_xlabel('Lectura potencia total')
ax.set_ylabel('Lectura energía activa');

# Gráfico dispersión POTENCIA_TOTAL_ACUMULADA/ENERGIA_ACTIVA_ACUMULADA
#
==========================================================================
fig, ax = plt.subplots(1, 1, figsize=(6,4))
ax.scatter(x=df.POTENCIA_TOTAL_ACUMULADA,
y=df.ENERGIA_ACTIVA_ACUMULADA, alpha= 0.8)
ax.set_xlabel('Potencia total acumulada')
ax.set_ylabel('Energía activa acumulada');

# Gráfico distribución variables
#
==========================================================================
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

axs[0].hist(x=df.LECTURA_ENERGIA_ACTIVA, bins=130, color="#3182bd",
alpha=0.5)
axs[0].plot(df.LECTURA_ENERGIA_ACTIVA,
np.full_like(df.LECTURA_ENERGIA_ACTIVA, -0.01), '|k',
markeredgewidth=1)
axs[0].set_title('Distribución lectura energía activa')
axs[0].set_xlabel('Lectura energía activa')
axs[0].set_ylabel('counts')

axs[1].hist(x=df.LECTURA_POTENCIA_TOTAL, bins=130, color="#3182bd",
alpha=0.5)
axs[1].plot(df.LECTURA_POTENCIA_TOTAL,
np.full_like(df.LECTURA_POTENCIA_TOTAL, -0.01), '|k',
markeredgewidth=1)
axs[1].set_title('Distribución lectura potencia total')
axs[1].set_xlabel('Lectura potencia total')
axs[1].set_ylabel('counts')


plt.tight_layout();

# Gráfico Q-Q
#
==========================================================================
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

sm.qqplot(
    df.LECTURA_ENERGIA_ACTIVA,
    fit   = True,
    line  = 'q',
    alpha = 0.4,
    lw    = 2,
    ax    = axs[0]
)
axs[0].set_title('Gráfico Q-Q LECTURA_ENERGIA_ACTIVA', fontsize = 10,
fontweight = "bold")
axs[0].tick_params(labelsize = 7)

sm.qqplot(
```

```
    df.LECTURA_ENERGIA_ACTIVA,
    fit   = True,
    line  = 'q',
    alpha = 0.4,
    lw    = 2,
    ax    = axs[1]
)
axs[1].set_title('Gráfico Q-Q df.LECTURA_ENERGIA_ACTIVA', fontsize =
10, fontweight = "bold")
axs[1].tick_params(labelsize = 7)

# Matriz de correlación
#
======================================================================
corr_matrix = datos.corr(method='pearson')
corr_matrix

# Heatmap matriz de correlaciones
#
======================================================================
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5))

sns.heatmap(
    corr_matrix,
    annot     = True,
    cbar      = False,
    annot_kws = {"size": 8},
    vmin      = -1,
    vmax      = 1,
    center    = 0,
    cmap      = sns.diverging_palette(20, 220, n=200),
    square    = True,
    ax        = ax
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 10)

#Definición de dataset con variables numéricas y sin las variables no
necesarias
num_df =
df.select_dtypes(include=['float64']).copy().drop(['ENERGIA_REACTIVA_A
CUMULADA','LECTURA_ENERGIA_REACTIVA'],axis=1)
num_df.info()

#Descripción del nuevo dataset
num_df.describe()

#Distribución de las variables agrupando por LECTURA_ENERGIA_ACTIVA
print(num_df.groupby('LECTURA_ENERGIA_ACTIVA').size())

#Se elimina del nuevo dataset la variable que queremos predecir
num_df.drop(['LECTURA_ENERGIA_ACTIVA'],1).hist()
plt.show()

#Se asigna las variables para poder realizar la predicción
```

```
X = np.array(num_df.drop(['LECTURA_ENERGIA_ACTIVA'],1))
y = np.array(num_df['LECTURA_ENERGIA_ACTIVA'])
X.shape

#Configuración
lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(y)
print(utils.multiclass.type_of_target(y))
print(utils.multiclass.type_of_target(y.astype('int')))
print(utils.multiclass.type_of_target(encoded))

from sklearn.ensemble import RandomForestClassifier
# El modelo se ha creado con 120 árboles
model = RandomForestClassifier(n_estimators=120,
                               bootstrap = True, verbose=2,
                               max_features = 'auto')
# Entrenamiento
model.fit(X, encoded)

predictions = model.predict(X)
print(predictions)

model.score(X,encoded)

#Validación del entrenamiento del modelo
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, encoded,
test_size=validation_size, random_state=seed)

from sklearn.ensemble import RandomForestClassifier

# Validación
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print(confusion_matrix(Y_validation, predictions))

print(classification_report(Y_validation, predictions))

model.score(X_train,Y_train)
```

8.2. Modelado SARIMA

```
#!/usr/bin/env python
# coding: utf-8

#Importación de las librerías necesarias
import pandas as pd
import numpy as np
from datetime import datetime
import seaborn as sb
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import utils
```

```
# Gráficos
#
# ==============================================================================
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y análisis
#
# ==============================================================================
import statsmodels.api as sm
from scipy import stats
from scipy.stats import pearsonr

# Configuración matplotlib
#
# ==============================================================================
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from time import time
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA as sklearnPCA
from sklearn import preprocessing
from sklearn import datasets
from sklearn.cluster import MeanShift
from sklearn.cluster import AffinityPropagation
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import KMeans
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report,
auc, precision_recall_curve, roc_curve

get_ipython().run_line_magic('matplotlib', 'inline')
RANDOM_SEED = 42

#Importación del dataset
df = pd.read_csv('./nodo_101914.csv', sep=';', encoding='utf-8')

#Comprobación de las primeras filas
df.head()

#Información del dataset. Se comprueban los tipos de datos que hay
(float64, enteros y categóricos)
df.info()

#Se configura el formato de la varible para que sea una fecha y sea el
índice del dataset
df["FECHA_TOMA_DATO"] =
pd.to_datetime(df["FECHA_TOMA_DATO"],format='%d/%m/%Y %H:%M')
df.set_index("FECHA_TOMA_DATO")

#Se comprueba que el formato y el índice se ha aplicado correctamente
df.info()

#Se configura un nuevo dataset eliminando las columnas que no son
necesarias para la aplicación de este modelo
datos = df.drop(['ID_NODO','ID_DIA'],axis=1)
```

```
datos.set_index("FECHA_TOMA_DATO")

# Matriz de correlación
#
# ==============================================================================
corr_matrix = datos.corr(method='pearson')
corr_matrix

# Heatmap matriz de correlaciones
#
# ==============================================================================
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5))

sns.heatmap(
    corr_matrix,
    annot     = True,
    cbar      = False,
    annot_kws = {"size": 8},
    vmin      = -1,
    vmax      = 1,
    center    = 0,
    cmap      = sns.diverging_palette(20, 220, n=200),
    square    = True,
    ax        = ax
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 10)

#Comprobación de las primeras filas
df.head()

#Gráfica de los primeros meses de lectura
plt.figure(figsize=(20, 10), dpi=150)
plt.plot(datos.FECHA_TOMA_DATO[:2000],
datos.LECTURA_ENERGIA_ACTIVA[:2000], color='tab:blue')
plt.gca().set(title='Lectura de energía activa',
xlabel='Fecha', ylabel='Vatios (W)')
plt.show()

#Se configuran los rangos de los posibles valores que puede obtener el
modelo
import itertools

p = range(1,3)
q = range(1,3)
d = range(1,3)
pdq = list(itertools.product(p, d, q))

seasonal_pdq = [(x[0], x[1], x[2], 15) for x in
list(itertools.product(p, d, q))]

print('Examples of parameter combinations for SARIMA..')
print('SARIMAX: {} * {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} * {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} * {}'.format(pdq[2], seasonal_pdq[2]))
```

```
print('SARIMAX: {} * {}'.format(pdq[3], seasonal_pdq[2]))

#Bucle que con las configuraciones anteriormente establecidas para
conocer la mejor opción de predicción
import warnings
import statsmodels.api as sm

warnings.filterwarnings("ignore")
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(
                datos.LECTURA_ENERGIA_ACTIVA,
                order=param, seasonal_order=param_seasonal,
                enforce_stationarity=False,
                enforce_invertibility=False)
            results = mod.fit()
            print('SARIMA{}*{}15 - AIC:{}'.format(param,
param_seasonal, results.aic))
        except:
            continue

#Modelo de SARIMA
mod = sm.tsa.statespace.SARIMAX(
    datos.LECTURA_ENERGIA_ACTIVA,
    order=(2, 0, 2), seasonal_order=(2, 0, 1, 15),
    enforce_stationarity=True,
    enforce_invertibility=False)
results = mod.fit()
results.summary()

#Predicción a partir de la posición 5000 para validar la predicción
pred = results.get_prediction(start=5000,
dynamic=False, full_results=False)
pred_ci = pred.conf_int()

#Gráfica de la lectura y su predicción
plt.figure(figsize=(20, 10), dpi=150)
ax = datos.LECTURA_ENERGIA_ACTIVA[:27000].plot(label='Actual',
color='tab:blue')
pred.predicted_mean[:20000].plot(ax=ax, label='Forecast',
color='tab:red')

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.05)
ax.set_xlabel('Date')
ax.set_ylabel('Lectura energía activa')
plt.legend()
plt.xlim(0,10000)
plt.ylim(-1,7e6)
plt.show()

#Error cuadrático medio para validar el error de la predicción
from sklearn.metrics import mean_squared_error
from numpy import sqrt
y_forecasted = pred.predicted_mean
y_truth = datos.LECTURA_ENERGIA_ACTIVA[5000:]
rmse = sqrt(mean_squared_error(y_truth, y_forecasted).mean())
print('The RMSE error of forecast prediction is {}'.format(round(rmse,
2)))
```

```
#Predicción de toda la serie
pred = results.get_prediction(
dynamic=False, full_results=True)
pred_ci = pred.conf_int()

#Gráfica de la predicción de toda la serie
plt.figure(figsize=(20, 10), dpi=150)
ax = datos.LECTURA_ENERGIA_ACTIVA.plot(label='Actual',
color='tab:blue')
pred.predicted_mean.plot(ax=ax, label='Forecast', color='tab:red')

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.05)
ax.set_xlabel('Date')
ax.set_ylabel('Lectura energía activa')
plt.legend()
plt.ylim(-1,7e6)
plt.show()
```