Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Ánalisis de patrones de comportamiento en una red de sensores IoT

Autor: Francisco Manuel Mouro Santos

Tutor: Pablo Muñoz Martínez

2022

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Análisis de patrones de comportamiento en una red de sensores IoT.

Francisco Manuel Mouro Santos
2022

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Ánalisis de patrones de comportamiento en una red de sensores IoT.

Autor: Francisco Manuel Mouro Santos
Tutor: Pablo Muñoz Martínez

TRIBUNAL:

Presidente: Rosa Estriégana Valdehita

Vocal 1º: Manuel Prieto Mateo

Vocal 2º: Pablo Muñoz Martínezz

Índice

Lista de Figuras

Resumen

En este trabajo de fin de grado se ha desarrollado una aplicación relacionada con el mundo de la domótica, la cual es capaz de detectar situaciones anómalas dentro de una vivienda con el objetivo de lanzar un mensaje de alerta al usuario de ser necesario.

Para llevar esto a cabo, se han utilizado un conjunto de datos procedentes de varios sensores IoT distribuidos a lo largo de la vivienda, los cuales fueron sometidos a un preprocesamiento y posteriormente procesados utilizando técnicas de Machine Learning, concretamente árboles de decisión, regresión lineal y regresión logística.

Palabras clave:
Domótica, sensores IoT, preprocesamiento, Machine Learning, Árboles de decisión.

Abstract

In this degree dissertation, an application related to the field of home automation has been developed, which is capable of detecting abnormal situations within a home with the aim of sending an alert message to the user if necessary.

To carry this out, a set of data from several IoT sensors distributed throughout the house has been used, which were subjected to pre-processing and subsequently processed using Machine Learning techniques, specifically decision trees, linear regression and logistic regression.

Key Words:
Home automation, IoT sensors, preprocessing, Machine Learning, Decision Trees.

Resumen Extendido

Antaño, el concepto de domótica era algo totalmente impensable. No existía forma alguna de poder automatizar y/o monitorizar ciertas tareas relacionadas con el hogar, como pueden ser la iluminación, el consumo energético o la seguridad.

Hoy en día, gracias al gran avance que ha sufrido la tecnología y, especialmente en las últimas décadas, el mundo de las comunicaciones, surge el concepto de IoT ("Internet of Things" o, por su traducción al castellano, "el Internet de las cosas"). El IoT nos permite poder monitorizar diversos objetos de uso cotidiano sin necesidad de ningún tipo de interacción física. Gracias a ello, multitud de tareas pueden realizarse de forma inalámbrica desde un dispositivo conectado a la red. Por ejemplo, mediante una aplicación instalada en el móvil sería posible regular la intensidad y los colores de la iluminación de una habitación. Incluso, este tipo de tareas podrían llegar a ser sometidas a algún tipo de algoritmo que se encargase de automatizarlas, eliminando la necesidad de interacción humana. Esto último es lo que se conoce como Machine Learning, y es en lo que se centrará este proyecto.

En este trabajo de fin de grado se ha desarrollado una aplicación relacionada con lo descrito en el anterior párrafo, cuya finalidad es la de detectar dentro de una vivienda situaciones anómalas de manera autónoma, como podría ser un intento de robo. Para ello se han distribuido en la vivienda varios sensores IoT de diversos tipos que se encargan de recoger información acerca de diferentes parámetros. como puede ser el grado de luminosidad en ciertas partes de la vivienda, la presión en la cama o en el sofá, la existencia de movimiento en el salón, en los pasillos, etc..

Para poder construir dicha aplicación a partir de los datos recogidos por los sensores es necesario llevar a cabo primero un preprocesamiento de estos. El preprocesamiento de datos es el proceso mediante el cual, a partir de los datos en bruto iniciales, se extraen otros nuevos que permiten un manejo más sencillo y adecuado para la tarea final, así como se eliminan aquellos que resultan innecesarios.

Una vez realizado el preprocesamiento se separan los nuevos datos en datos de entrenamiento y datos de prueba. Con los datos de entrenamiento se construye el algoritmo que se desea implementar, y con los datos de prueba, tal y como el propio nombre indica, se comprueba el funcionamiento del algoritmo ya construido. En la mayoría de casos, los datos de entrenamiento corresponden a un total del 85 % del total de datos, mientras que los datos de prueba corresponden al 15 % restante.

Existen varios tipos de algoritmos dentro del Machine Learning. En este caso los utilizados para resolver el problema propuesto han sido la regresión lineal, la regresión logística y, especialmente, árboles de decisión. Un árbol de decisión es una especie de mapa en el que se muestra cada una de las opciones de decisión posibles y sus resultados. En función de los datos de entrada al árbol, que en este caso son el ID del sensor y el momento específico del día, se recorre el árbol siguiendo un camino determinado, y se predice la salida del sensor en cuestión.

Existen varios tipos de árboles de decisión. En este caso se crearon árboles de tipo CART, de tipo ID3, de regresión lineal y de regresión logística. El más fiable de todos es el de tipo CART, por lo que ha sido el utilizado para crear la aplicación final, que es la encargada de generar el mensaje de alerta en función de la predicción obtenida al aplicar dicho algoritmo a los datos entrantes.

El lenguaje de programación utilizado a lo largo de todo el trabajo ha sido Python, debido a la simplicidad que ofrece para asuntos relacionados con el Machine Learning y el tratamiento de altos volúmenes de información, gracias a ciertas librerias como scikit-learn o pandas.

1 Introducción

En esta introducción a este trabajo de fin de grado se expondrá la motivación por la cual se ha propuesto su desarrollo, los objetivos que se pretenden cumplir y finalmente la estructuración que presentará el proyecto.

1.1. Motivación

El mundo ha cambiado más en los últimos doscientos años que lo que lo había hecho en los miles de años anteriores de la historia del ser humano. Desde el comienzo de la revolución industrial se han ido produciendo cambios de forma exponencial que no sólo han permitido un aumento en la esperanza de vida de las personas, sino también en la calidad de esta.

La máquina de hilar, el barco de vapor, el ferrocarril... son sólo unos de los muchos inventos que han permitido la reducción de trabajo de las personas y el aumento de la eficiencia en todo tipo de tareas y ámbitos. Sin embargo, la mayor revolución tecnológica, que ha conseguido cambiar la forma en la que actuamos, en la que trabajamos e incluso en la que nos relacionamos ha sido la relacionada con los avances en el mundo de las telecomunicaciones y, en especial, Internet.

Quinientos años atrás, para transmitir un mensaje de un lugar concreto del planeta a otro en particular, se podría demorar entre días y meses, dependiendo de lo remoto que fuese el destino. Con el paso de los años y con un simple ordenador, podemos comunicarnos en tiempo real con cualquier entidad sin importar su lugar de residencia.

Un ejemplo más particular, como el garantizar la seguridad de una vivienda o de un local, estaban al alcance de unos pocos afortunados que pudiesen tolerar el esfuerzo económico que supone mantener un personal de seguridad adecuado. Posteriormente, los avances tecnológicos permitieron el introducir el concepto de cámara de seguridad, aunque esta seguía necesitando la participación directa de al menos una persona que pudiese controlar y monitorizar la información aportada por dicha cámara. Pero, ¿qué ocurriría si se pudiese utilizar todas estas nuevas tecnologías como el Internet, las telecomunicaciones y la programación a favor para poder conseguir finalmente un sistema que no requiriese de la intervención humana para poder llevar a cabo este tipo de tareas?

Esa es la motivación de este TFG: conseguir crear un sistema autónomo que sea capaz de detectar situaciones anómalas. Para lograr esto, es primordial tener una serie de sensores instalados y distribuidos estratégimanete que transmitan información acerca de determinadas partes del local o vivienda. Posteriormente esta información se preprocesará y se le aplicará algoritmos de Machine Learning que permitan construir el sistema autónomo buscado.

1.2. Objetivos

El objetivo principal es construir un sistema de detección de situaciones extrañas que funcione de manera automática. Con el fin de lograr esto, se establecen una serie de objetivos secundarios, los cuales son:

- Estudiar la información procedente de todos los sensores distribuidos a lo largo de la vivienda, examinando cómo está estructurado el dataset en el que viene recogida dicha información, la forma en la que se identifica cada sensor y el tipo de dato del que se trata (valor numérico continuo, valor alfabéticos, valores booleanos, etc.).
- Preprocesar la información con el fin de eliminar aquellas partes que resultan indiferentes para la tarea final, así como modificar ciertos datos para ahorrar potencia de cómputo o adecuar la información a un formato con el que resulte más sencillo trabajar.
- Aplicar a este dataset preprocesado diferentes algoritmos de Machine Learning que tengan como objetivo la predicción de un atributo objetivo.
- Realizar una comparación de eficiencia entre todos los algoritmos utilizados para ver cuál es la mejor opción para la aplicación final.

1.3. Estructuración de la memoria

A continuación se expondrán de manera ordenada todos los puntos de la memoria, así como se explicará brevemente el contenido que se abordará en cada uno:

- Introducción: En este apartado se habla sobre cuál es la motivación para realizar el trabajo, el objetivo principal y los secundarios de este mismo y la estructarición de la memoria.
- Estado del arte: Se explican conceptos básicos sobre Internet of Things y Machine Learning y cuáles son los tipos principales de estos mismos.
- Dataset inicial y su preprocesamiento: Se explica la naturaleza del dataset en crudo y los cambios a los que se le somete con el preprocesamiento.
- Elección del algoritmo ideal: Se describe el resultado de la aplicación de algoritmos de distintos tipos al nuevo dataset creado en el anterior punto y se escoge el mejor de todos.
- Elaboración del programa final: Se explica la creación del programa final que cumple con el objetivo principal del trabajo, es decir, automatiza el proceso de detección de situaciones anómalas en la vivienda.
- Conclusiones y futuros trabajos: Se valora si se han cumplido adecuadamente los objetivos y sugiere futuros trabajos acerca del tema tratado en este TFG.
- Presupuesto: Trata el presupuesto que ha sido necesario para llevar a cabo este trabajo.

2 Estado del arte

En este apartado se abordará el estado del arte de todo lo que engloba al **IoT**: qué es exactamente, por qué está en auge, los tipos de sensores que pueden existir...

También se desarrollará el concepto de **Machine Learning**, así como los tipos de este que existen y del preprocesamiento de datos.

2.1. Internet of Things

Explicado de forma simple y concisa, el IoT ("Internet of Things") es el concepto de conectar cualquier dispositivo (siempre que tenga un switch de on/off) a la red y a otros dispositivos que también se encuentren conectados. Todos estos dispositivos tienen la capacidad de recolectar información del medio y compartirla con entre ellos o con la red, sin la necesidad de que haya algún tipo de conexión humano-humano o humano-máquina.

Este concepto está cada vez más extendido y se pueden observar multitud de ejemplos en el mundo cotidiano. Un ejemplo de esto son los relojes que recogen el número de pasos que una persona realiza a lo largo del día, así como su actividad cardíaca. Todos estos datos se utilizan para hacer algún tipo de sugerencia, para obtener otros nuevos, como puede ser el gasto calórico, para subir dicha información a la red con el fin de compartirla, registrarla o almacenarla en algún tipo de base de datos que permita llevar un registro diario, etc.

2.1.1. IoT por niveles

La idea de IoT se puede aplicar a tres niveles: el doméstico, el empresarial y el industrial.

El escenario más simple es el **doméstico**, puesto que es el más pequeño y el que menos dispositivos presenta. Un ejemplo de IoT en el hogar podría ser el mostrado en la figura 2.1.

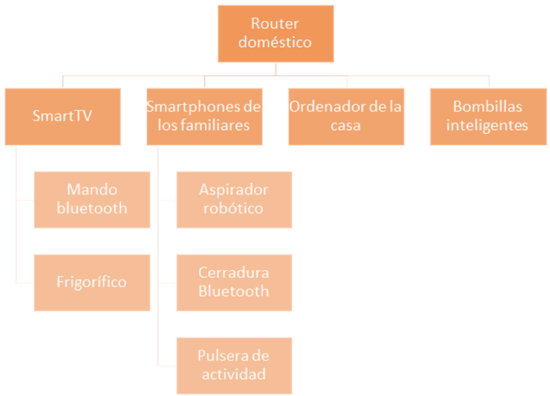

Figura 2.1: Ejemplo de IoT doméstica. Fuente: Blog de Orange [1].

El router es el elemento principal de la red, y a este se conectan todos los demás dispositivos: el televisor inteligente, los smartphones, las bombillas inteligentes. Todo se encuentra conectado mediante WiFi, aunque también existen otras subredes que utilizan Bluetoth.

Una red IoT **empresarial** sería más compleja. ya que cuenta con más routers, a parte de otros sistemas, como podría ser la climatización o el control de accesos. Todos estos routers y sistemas están conectados a través de la propia red IoT.

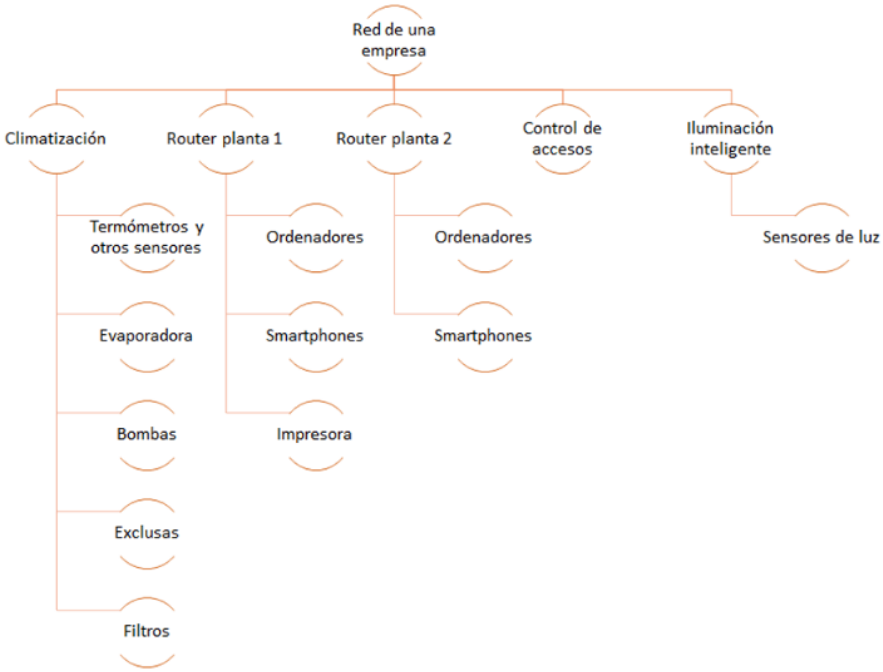

Figura 2.2: Ejemplo de IoT empresarial. Fuente: Blog de Orange [1].

En cuanto a la industria, aparece el concepto de **IIoT** ("Industrial Internet of Things"), también conocido como "Industry 4.0" o internet industrial, y representa el uso del IoT con el objetivo de mejorar la manufacturación y los procesos industriales.

El mundo del automóvil, el del petróleo o el de la agricultura y ganadería son sólo unos pocos de los muchos beneficiados por estas nuevas tecnologías. Por ejemplo, respecto a este último, la información recolectada sobre el tiempo o el suelo son utilizados para avisar al granjero sobre si es una decisión correcta el cosechar o no. También, el implantar chips a animales ayuda a los trabajadores a localizar la posición exacta de cada uno de ellos, así como mantener un estado de las condiciones de estos mismos (peso, altura, estado de salud...).

2.1.2. Sensores IoT

Para poder monitorizar y compartir información, es primeramente necesario recolectarla. De esto se encargan precisamente los sensores IoT.

Un **sensor IoT** es un dispositivo que, a parte de medir una magnitud determinada, lleva integrado un circuito destinado a la comunicación con otros sensores y con la red, cumpliendo ciertos estándares.

Los sensores pueden ser analógicos o digitales: los analógicos pueden tomar cualquier valor, mientras que los digitales sólo toman dos (1 o 0, True o False). Dependiendo de las circunstancias y de la información que sea preciso monitorizae, la proporción de cada uno variará.

Existen multitud de sensores, pero los más comunes en estos ámbitos son [4]:

- Sensores de temperatura: Miden la energía térmica y son analógicos. Suelen ser de gran importancia. Siguiendo con el ejemplo descrito en el apartado anterior sobre la agricultura, en este campo se suelen introducir sensores de este tipo para medir la temperatura del suelo (véase un ejemplo de sensor de temperatura en la figura 2.3).
- Sensores de humedad: Miden la cantidad de vapor de agua y, al igual que los de temperatura, son analógicos. Se suelen usar en sistemas de aire acondicionado.
- Sensores de presión: Detectan cambios en la presión, y son utilizados sobre todo en sistemas hidráulicos para detectar fluctuaciones o cáidas en este parámetro.
- Sensores de proximidad: Detectan objetos que se encuentren cerca de donde están instalados.
- Sensores de nivel: A diferencia de los anteriores, no funcionan por cercanía sino por contacto, y son empleados para medir el nivel de sustancias como líquidos respecto a un contenedor.
- Acelerómetros: Como el propio nombre indica, miden la aceleración de objetos, así como cambios en la gravedad.
- Giroscopio: Calculan la velocidad angular de un cuerpo. Resultan útiles en sistemas de navegación o en sistemas de control de estabilidad (véase un ejemplo de giroscopio en la figura 2.4).
- Sensores de infrarrojo: Detectan radiación infrarroja.
- Sensores de gas: Detectan cambios en el aire (su calidad, presencia de gases tóxicos, etc.).

Figura 2.3: Sensor de temperatura.

Figura 2.4: Giroscopio.

2.1.3. Por qué el IoT está en auge

Según un informe de Globaldata ("Thematic Research: Internet of Things"), el mercado del IoT alcanzará un valor en ingresos de 1'1 billones de USD en 2024 [5]. Teniendo en cuenta que en 2020 fue de 622.000 millones de USD y en 2019 de 586.000, esto supone un gran crecimiento para su mercado (véase la figura 2.5).

Este crecimiento económico y aumento de interés en el sector se refleja en el aumento del número de dispositivos IoT (véase la figura 2.6). Como se puede observar, mientras que la cantidad de dispositvos no-IoT se espera constante, el número de dispositovs IoT conectados tiende a incrementarse más y más cada año.

Aunque existen muchas razones por las que la implementación de estas tecnologías resulta beneficioso para individuos y empresas, estas son las más destacables [6]:

- Cuanta más información, mejor: El hecho de poder utilizar sensores IoT que midan multitud de magnitudes respecto a un área en concreto implica que mayor cantidad de información se encuentra disponible para someterla a estudio y producir mejoras en el ámbito en cuestión.

- Posibilidad de monitorizar y seguir la pista a todo tipo de entidades: Si el anterior punto era una razón mayor para el empleo de IoT a nivel industrial, esta razón representaría una ventaja para el individuo. El IoT permite mantener un registro acerca del estado de dispositivos electrónicos del hogar (o incluso de animales mediante chips) y tomar decisiones para con ellos.

- Reducción de la carga de trabajo: A parte de la monitorización, el IoT también puede utilizarse para la automatización de tareas, lo que implica que las máquinas pueden llegar a tener capacidad de actuación sin necesidad de intervención humana (o, por lo menos, reduciendo en un gran porcentaje esta). Todo esto implica una disminución del esfuerzo humano.

- Mayor eficiencia: Un aumento en la eficiencia suele conllevar una reducción en el gasto económico. Por ejemplo, si se consigue elaborar un sistema que fuese capaz de detectar presencia humana a la vez que mide los niveles de luz en una habitación, activando la iluminación artificial de ser necesario (es decir, que haya presencia humana y a la vez bajos niveles de luz natural) se conseguiría un sistema totalmente eficiente a nivel energético, que no es vulnerable al error humano, como podría ser el dejar las luces encendidas.

- Mayor sostenibilidad: Al fomentar una mayor eficiencia y menor gasto innecesario de recursos, sólo se emplean los recursos realmente necesarios.

- Mayor calidad de vida: Realmente todos los puntos anteriores conducen hasta este último, el cual es el fin que la mayoría de avances tecnológicos persiguen. Cuantas más tareas sea el ser humano capaz de adjudicar a una máquina, menores esfuerzos mentales y físicos habrá de realizar, reduciendo así la posibilidad de error humano.

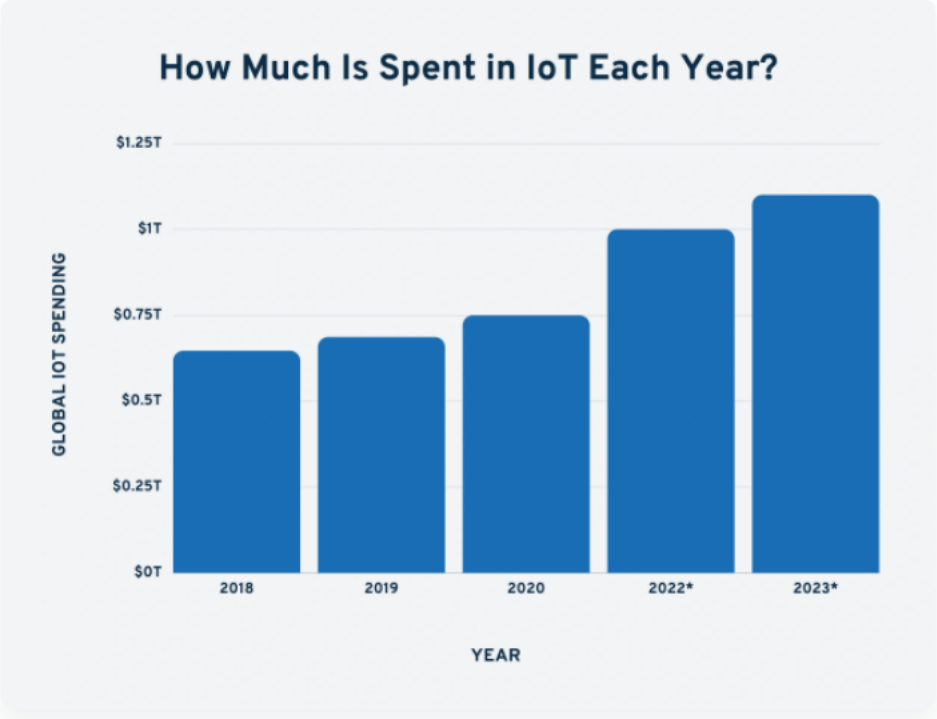

Figura 2.5: Gasto en IoT por año. Fuente: Exploding Topics [2].

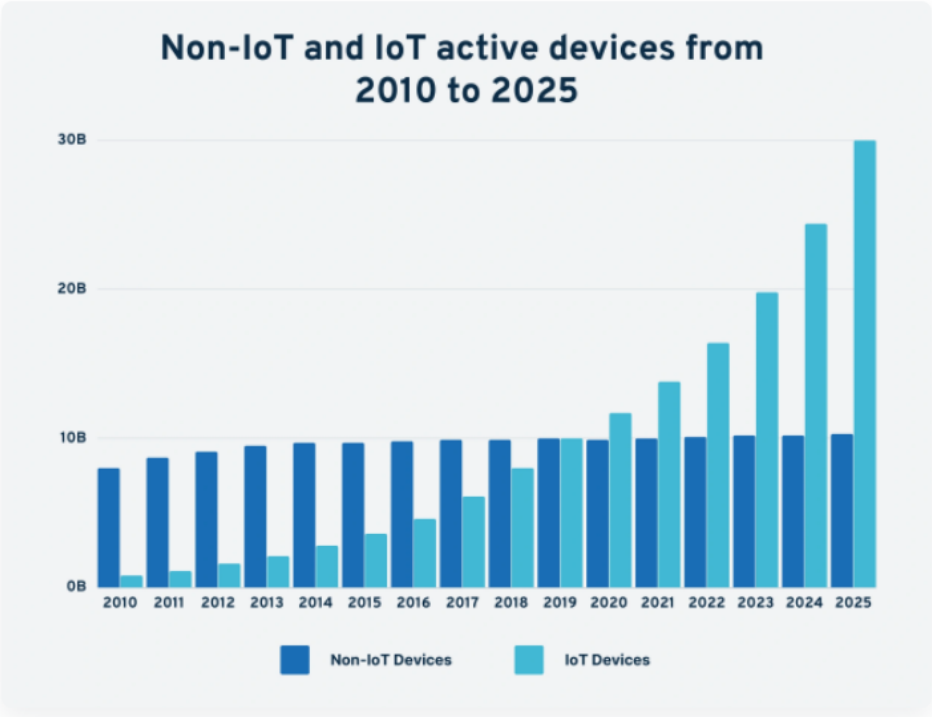

Figura 2.6: Número de dispositivos no-IoT frente a dispositivos IoT. Fuente: Exploding Topics [2].

2.1.4. Desventajas del IoT

Como todo, el IoT también presenta a día de hoy ciertas desventajas o puntos negativos. Los más considerables son los siguientes:

- Falta de seguridad: La mayoría de redes IoT no cuentan con un sistema de cifrado, lo que las expone a la posibilidad de ser hackeadas, y la información podría ser robada o vulnerada.
- Gran inversión económica inicial: Aunque el mercado tiende al alza, lo que implica más empresas procurando sacar a la venta productos más eficientes y menos caros, a día de hoy introducir toda esta tecnología supone un esfuerzo financiero elevado, el cual no todo el mundo es capaz de asumir.
- Menor intimidad: Esto se puede producir de múltiples formas: un uso incorrecto de ciertos dispositivos, como cámaras de seguridad, o la publicación de datos personales recogidos por los sensores distribuidos a lo largo de la vivienda. Especialmente este último caso habría que tenerlo en cuenta, puesto que los datos de toda persona están expuestos a ser registrados y sometidos a estudio.
- Brecha tecnológica: No todo el mundo podría acceder a esta tecnología en corto-medio plazo. Por ejemplo, ciudades muy avanzadas como pueden ser Nueva York o Toronto se beneficiarán enormemente de esta tecnología cada vez más a lo largo de los próximos años, mientras que países subdesarrollados apenas podrán importar un porcentaje ínfimo de lo que estas ciudades inviertan.
- Falta de compatibilidad: Aunque sí existen ciertos estándares, es muy común a día de hoy la incompatibilidad entre varios dispositivos, que fueron programados de distintas maneras y esto impide que puedan trabajar de forma conjunta.

2.2. Machine Learning

Una vez descrita la idea de IoT, los diferentes niveles a los que se aplica, su situación actual en el mercado y sus puntos negativos, se procederá a explicar el concepto de Machine Learning.

El Machine Learning es un tipo de inteligencia artificial que se centra en el uso de datos y algoritmos con el objetivo de imitar la conducta humana de la manera más precisa posible, tomando decisiones de forma similar a estos. Lo que se pretende en Machine Learning es conseguir que las máquinas tomen decisiones en base a la experiencia, y no a las instrucciones.

Cuando los humanos tomamos una decisión, solemos cumplir con el siguiente *modus operandi*: Primero, recordamos situaciones pasadas similares, procedemos a formular una regla general que aplique a la mayoría de situaciones anteriores y usamos esta regla para tratar de predecir qué consecuencias tendrá el tomar cierta decisión. Este proceso se podría resumir en "recordar-formular-predecir". En el caso de una máquina, para "recordar" se guarda en memoria una tabla de datos que describa la mayor cantidad de situaciones ocurridas respecto a una variable en concreto que se quiera predecir, "formular" es aplicar una serie de reglas y fórmulas a esta tabla de datos, buscando la que más se adecue a esta, y "predecir" es la aplicación del conjunto de reglas y fórmulas anteriores a los datos futuros, que serán los que se tratarán de predecir [7].

Existen diversos tipos de Machine Learning, los cuales se pueden clasificar según el tipo de aprendizaje o según el algoritmo utilizado.

2.2.1. Tipos de aprendizaje

Existen tres formas de entrenar algoritmos de Machine Learning [8], y cada una de ellas presentará una serie de pros y contras.

- Aprendizaje supervisado: Es un tipo de Machine Learning en el que la información se encuentra etiquetada. Funciona de tal manera que al algortimo se le aporta una parte del conjunto de datos conocida como datos de entrenamiento, que normalmente es un conjunto de entradas escogidas al azar con el objetivo de conseguir una apariencia similar al dataset completo. Posteriormente el algoritmo utilizará estos datos para encontrar relaciones entre parámetros, buscando un conjunto de "causas-efectos" entre las variables del dataset. Una vez encontrada la solución final, se podrá utilizar para la predicción. Además, el algoritmo seguirá mejorando a medida que se van introduciendo nuevos datos, encontrando nuevos conjuntos de "causas-efectos".

  Este tipo de aprendizaje es el más común hoy en día y es el que usan aplicaciones de reconocimiento facial, filtros de spam, etc.

- Aprendizaje no supervisado: Este tipo de aprendizaje es el que se lleva a cabo cuando los datos no están etiquetados, es decir, no existe ningún objetivo el cual tratar de predecir. El hecho de que estos no posean una etiqueta significa que no se necesita el trabajo de la persona para hacer el dataset legible para la máquina, y permite datasets mucho más grandes. Al no tener ningún parámetro en concreto el cual tratar de predecir, las relaciones que el algoritmo establezca entre los datos serán percibidos por el propio algoritmo de una manera abstracta.

  Se suele usar para agrupar la información en grupos según la similitud de los datos entre ellos, y para reducir las dimensiones del dataset, es decir, simplificarlo y describirlo con menos parámetros.

- Aprendizaje por refuerzo: Es el que más se inspira en cómo los humanos aprenden de la información. El algoritmo va aprendiendo más y más a medida que avanza el tiempo a través de la prueba y el error; las decisiones correctas son reforzadas, y las incorrectas, castigadas. Para esta labor existen un intérprete, que decide para

cada iteración si el resultado es favorable o no, y un sistema de recompensas. De ser favorable, recompensa al algoritmo; si no lo es, obliga a este a repetir el proceso hasta que obtenga un mejor resultado.

Los programas que buscan la mejor ruta desde un punto A a un punto B utilizan este tipo de aprendizaje.

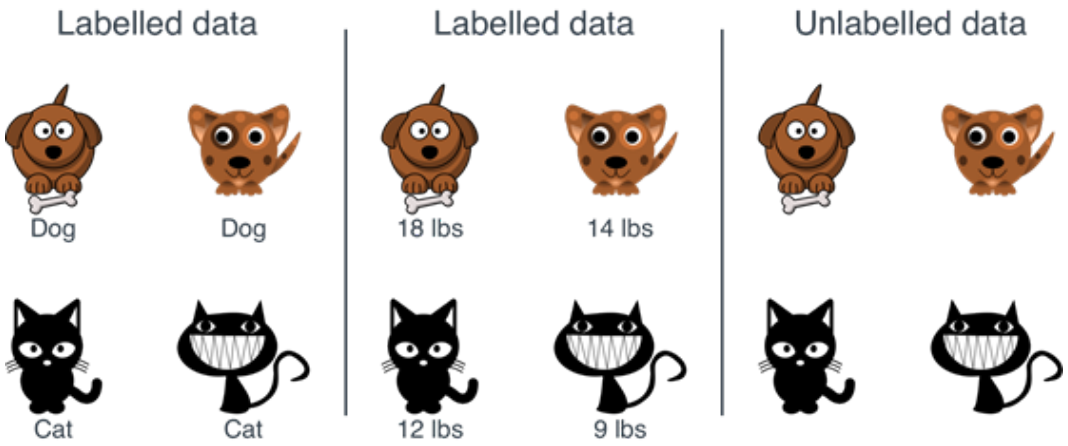

Figura 2.7: Diferencia entre información etiquetada e información sin etiquetar. Fuente: Grokking Machine Learning [3].

2.2.2. Algoritmos más empleados

Aunque existen más, estos son los algoritmos que más se utilizan dentro del mundo del Machine Learning [9]:

Regresión:

En este tipo de tareas, el programa obtiene una serie de relaciones existentes entre la información entrante con el fin de predecir una variable dependiente de las anteriores. Al ser necesaria la existencia de tal variable dependiente la cual predecir, es preciso que la información se encuentre etiquetada.

La regresión puede ser de dos tipos:

- Lineal: Fue creada en el siglo XIX por Francis Galton y en él la variable predecida es continua. Para explicar esto, se escogerá el caso más sencillo posible (véase figura 2.8): una variable de entrada "x" y una variable dependiente "y" (que será la que se tratará de predecir). El algoritmo se encargaría de construir una recta (en dos dimensiones en este caso) que se adecue lo más posible a cada uno de los puntos de la gráfica.
- Logística: Se usa para resolver problemas de clasificación, en los cuales la solución no es numérica. La solución puede tomar dos valores: SÍ/NO. En este caso, siguiendo con el ejemplo de una variable "x" y otra "y", la regresión ya no sería representada

por una simple línea recta, sino que presentaría una forma similar a la mostrada en la figura 2.9.

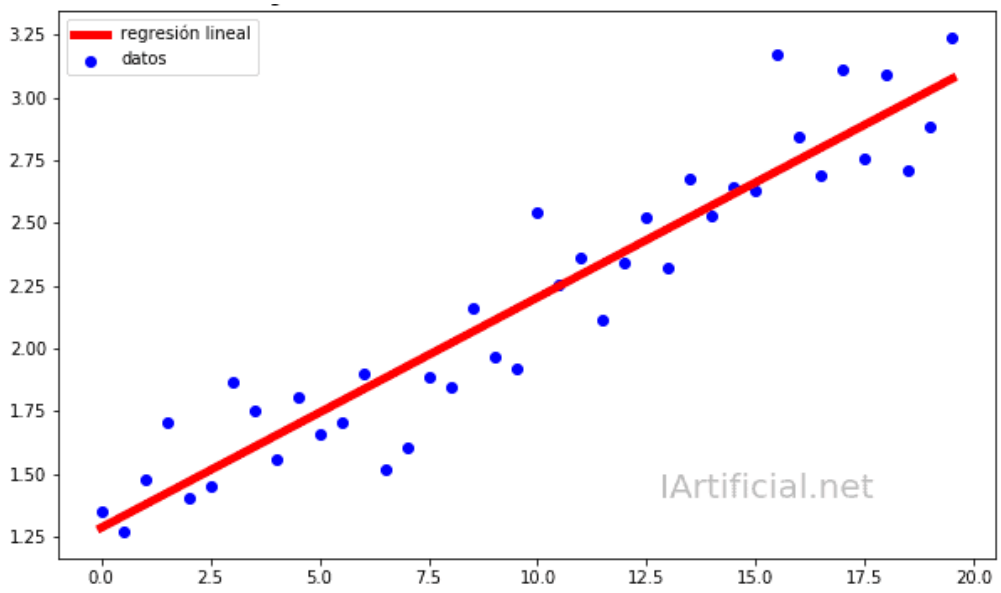

Figura 2.8: Regresión lineal.

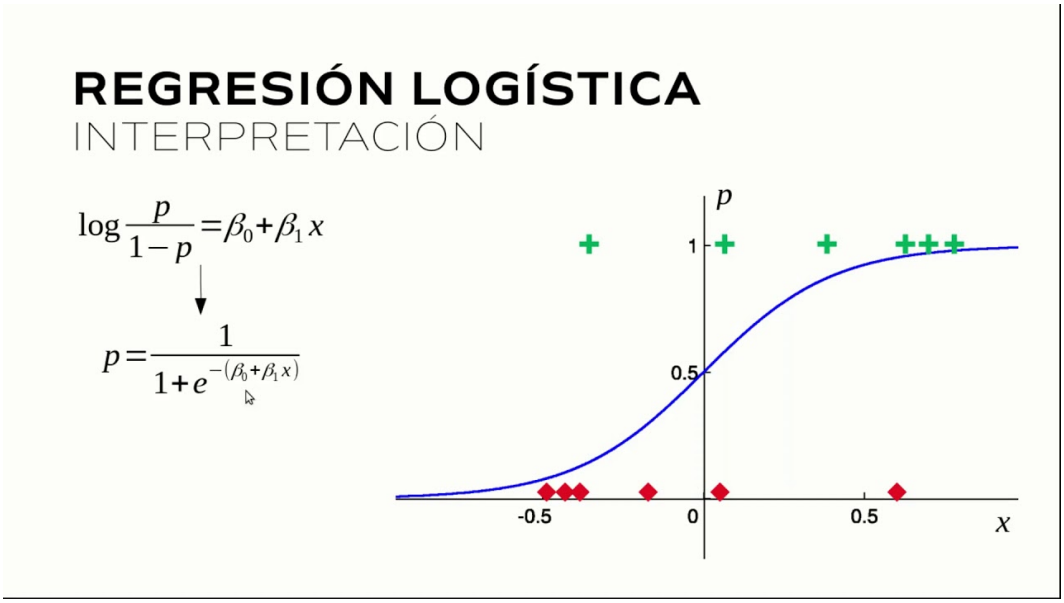

Figura 2.9: Regresión logística.

Algoritmos de agrupación o clustering:
Es una tarea de Machine Learning no supervisada, es decir, tal y como se ha descrito en el apartado anterior, destinada a datasets sin etiquetar. Se encarga de agrupar los datos según determinados parecidos entre estos. Se pueden dividir en cuatro tipos [10]:

- Partición estricta: cada objeto sólo puede pertenecer a un cluster o agrupación

- Partición estricta con outliers: Lo mismo que el anterior pero se añade el concepto de outlier, el cual define a aquel objeto que no pertenece a ningún cluster.
- Clustering con superposiciones: Algunos clusters se superponen, por lo que existe la posibilidad de que un mismo objeto cumpla los requisitos necesarios para pertenecer a dos clusters al mismo tiempo.
- Clustering jerárquico: Existe un cluster padre del que van surgiendo el resto de clusters, por lo que si un objeto pertenece a un cluster determinado, tambíen lo hará a su padre.

El algoritmo más representativo es "K-means", que es un clustering de partición estricta (véase la figura 2.10). Este algoritmo tratará de encontrar K particiones para el conjunto de datos total, de las cuales cada objeto pertenecerá a la agrupación cuyo centroide se encuentre más próximo.

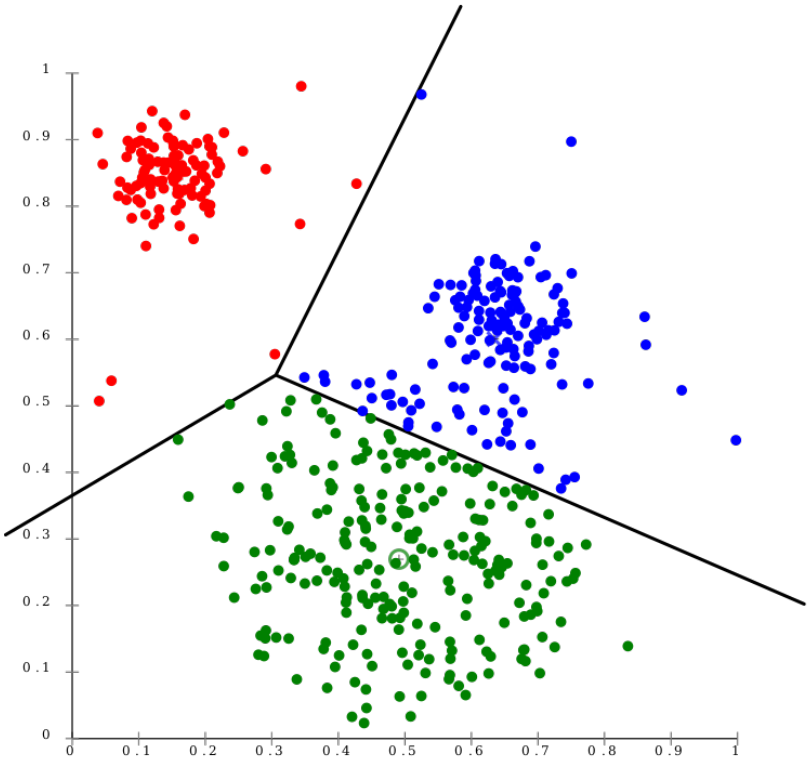

Figura 2.10: K-means, modelo de centroides.

Algoritmos de reducción de dimensionalidad:

Como los anteriores, también son utilizados con datasets sin etiquetar. Su principal función es el conseguir hacer las operaciones menos engorrosas, de forma que eliminan aquellas características las cuales no estén relacionadas o simplemente no presenten una correlación estricta con la variable objetivo.

La reducción de dimensiones puede producirse manteniendo sólo las características más importantes o combinando dos o más características para formar una nueva. Respecto al primer método, el algortimo más utilizado son los bosques aleatorios ("Random Forests" en inglés). En cuanto al segundo, es posible llevarlo a cabo mediante métodos lineales o no lineales, dependiento de la naturaleza del dataset.

Algoritmos bayesianos

Son un tipo de algoritmo de aprendizaje supervisado, basado en el teorema de Bayes. Se usa para resolver problemas de clasificación, basándose en la probabilidad de un evento, la cual se calcula con la siguiente fórmula:

$$P(A|B) = \frac{(P(B|A) * P(A)}{P(B)}$$

En dicha fórmula, P(A—B) representa la probabilidad a posteriori (probabilidad de que ocurra A dado por hecho que ocurre B), P(B—A) es la probabilidad condicional (probabilidad de que ocurra B dado por supuesto que ha ocurrido A), P(A) es la probabilidad a priori (probabilidad de que ocurra A) y P(B) es la probabilidad total (probabilidad de que ocurra B).

Aunque es bastante simple, funciona sorprendentemente con mucha precisión.

Existen tres tipos de modelos de algoritmos bayesianos: el gaussiano (asume que cada atributo sigue una distribución normal), multinominal (se usa cuando la información está distribuida multinominalmente) y el clasificador Bernouilli, que trabaja parecido al multinominal pero las variables predictoras son booleanas indeppendientes.

Este tipo de algoritmos se usan sobre todo en clasificadores médicos, en filtros de spam, en aplicaciones que realicen predicciones en tiempo real, etc.

Redes Neuronales:

Este algoritmo, tal y como su nombre deja entrever, está basado en el funcionamiento de nuestras neuronas. El cuerpo humano contiene millones de millones de neuronas, las cuales se comunican entre sí mediante un proceso denominado sinapsis. Este es el concepto específico que se toma para formular la red neuronal artifical, la cual está constituída por:

- Nodos de entrada: Son aquellos nodos que reciben información del exterior.
- Nodos de salida: Son aquellos que vuelcan la información al exterior.
- Nodos ocultos: No mantienen ningún contacto con el exterior y sólo intercambian información. Estos se organizan por capas conectándose entre sí.

La red "aprende" generando una predicción para cada entrada, realizando ajustes a las ponderizaciones cuando se realiza una predicción incorrecta. Al principio todas estas predicciones son completamente aleatorias, pero la red va mejorando más y más a base de entrenamiento con entradas para las cuales ya se conoce su resultado.

Pueden ser usadas tanto para problemas de regresión como de clasificación, y existen cinco tipos [11]:

- Red neuronal monocapa: Sólo presenta una capa intermedia entre los nodos de entrada y los de salida.
- Red neuronal multicapa: Presenta varios nodos intermedios entre ls de entrada y los de salida.
- Red neuronal convolucional (CNN): A diferencia de la anterior, en la que cada nodo se une con todas las capas siguientes, sólo lo hace con un subgrupo determinado de ellas, reduciendo el número de neuronas utilizadas (véase la figura 2.11 como ejemplo).
- Red neuronal recurrente (RNN): No tienen estructura de capa, sino que las conexiones son aleatorias enter las neuronas, añadiendo la posibilidad de crear ciclos para introducir el concepto de memoria a la red (véase la figura 2.12 como ejemplo).
- Redes de base radial (RBF): Calculan la salida dependiendo de la distancia al "centro" de la red (véase la figura 2.13 como ejemplo).

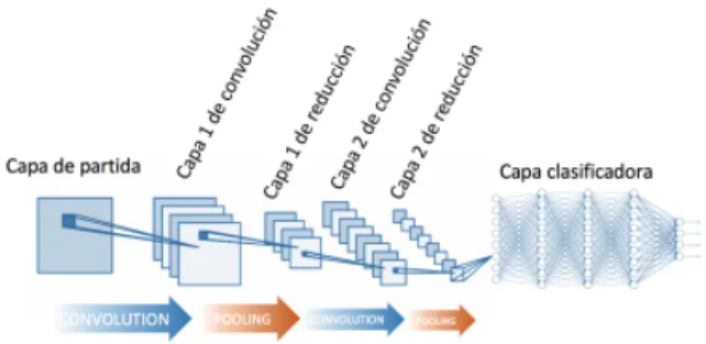

Figura 2.11: Red neuronal convolucional.

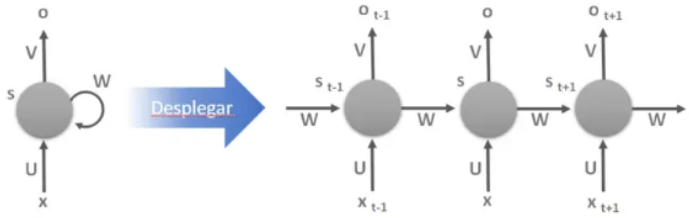

Figura 2.12: Red neuronal recurrente.

Figura 2.13: Red neuronal radial.

Árboles de decisión:

Es un tipo de Machine Learning de aprendizaje supervisado que se usa para hacer predicciones basadas en como fue "contestada" una pregunta anterior.

Estos algoritmos deben su nombre a su forma, la cual se asemeja precisamente a un árbol, y están compuestos por:

- Nodo raíz: Es la base del árbol de decisión
- Nodo intermedio o de decisión: Es un nodo el cual dirige a otros nodos.
- Nodo "hoja" o terminal: No dirige a otros y representa una posible solución

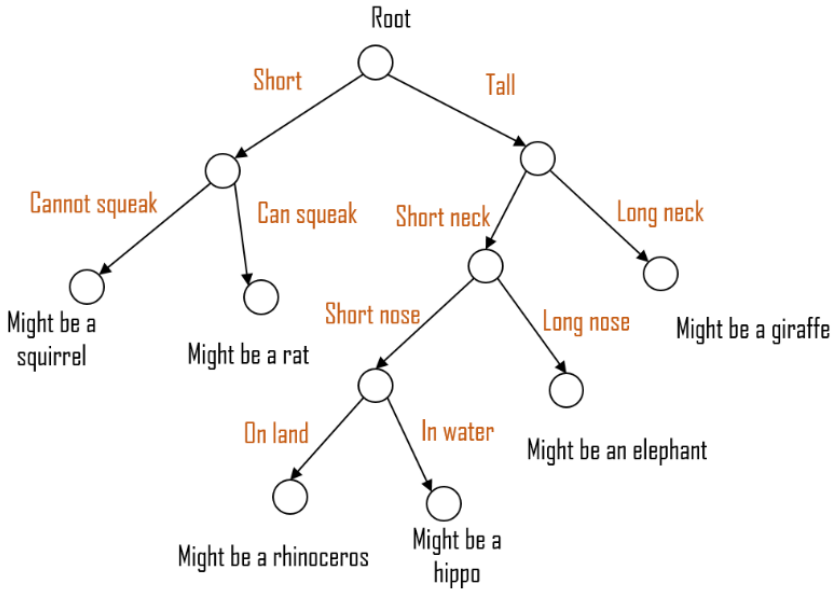

Figura 2.14: Ejemplo de árbol de decisión.

Su funcionamiento es bastante sencillo de comprender y para explicarlo se utilizará como ejemplo el árbol de la figura 2.14:

Supongamos que se le aporta al árbol de decisión el siguiente array de datos: {Tall, Short neck, Long nose, On land, Cannot squeak}, que describe a un animal el cual se quiere "adivinar" (precedir) cuál es. Es necesario que este array se encuentre estructurado de tal forma que el árbol pueda entender, de forma que tiene que seguir una estructura determinada y cada atributo sólo tome ciertos valores. Para predecir de qué animal se trata, el primer atributo que el algoritmo estudiará es el referente al tamaño del animal, puesto que es el atributo que se analiza nada más aplicarse el algoritmo, es decir, en el nodo raíz. Como la respuesta es "Tall", se avanza hacia el nodo intermedio de la derecha. Si la

respuesta hubiese sido “Short” el siguiente atributo a estudiar hubiese sido si el animal en cuestión chirría o no, pero como ha sido “Tall” se analiza la longitud del cuello. En este caso es “Short neck”, por lo que se avanzaría por la ramificación izquierda. Este proceso se repite continuamente hasta llegar a un nodo terminal o nodo hoja, el cual aporta una solución al problema planteado. Siguiendo el proceso descrito, al comparar el valor de la longitud de la nariz con el que se ha introducido al principio, el algoritmo llegaría a la conclusión de que se trata de elefante.

Este tipo de Machine Learning tiene como ventajas que puede trabajar tanto con valores numéricos como categóricos y requiere menor preprocesamiento de datos que otros algoritmos, pero por otro lado se ven muy afectados por el ruído y no trabajan demasiado bien con datasets muy largos.

Los tipos de árboles de decisión más comunes son:

ID3:

Utiliza el concepto de ganancia de información, que a su vez se calcula con la entropía, que mide la cantidad de incertidumbre dentro de un dataset. Para cada atributo se calcula su ganancia de información, y aquel que presente una mayor ganancia de información será el que forme el nodo raíz. La cantidad de ramificaciones que este presente dependerá de la cantidad de valores distintos que dicho atributo pueda tomar, y para cada una de ellas se aplicará el mismo proceso, buscando el atributo con mayor ganancia de información, esta vez realizando el cálculo para el caso específico concerniente. Para explicar esto, se utilizará el árbol ID3 de la figura 2.16. En este caso, el atributo con mayor ganancia de información ha sido “Outlook”, entonces el árbol crea tres ramas que corresponden a los tres posibles valores que “Outlook” puede tomar. Para el valor “overcast”, se observaría que todas las opciones del dataset indican que se ha jugado al tenis, por lo que no es necesario seguir estudiando los otros atributos. Para “Sunny”, al escoger todos esas entradas del dataset en las que “Outlook” vale “Sunny” se observa que el atributo que mayor ganancia de información aporta es “Humidity”, mientras que para “Rain” sería “Wind”.

Este tipo de árboles sólo se pueden usar como árboles de clasificación, es decir, la variable objetivo toma valores no numéricos.

Tras ID3, surgieron C4.5 y C5.0, que son evoluciones de este mismo las cuales mejoran el rendimiento.

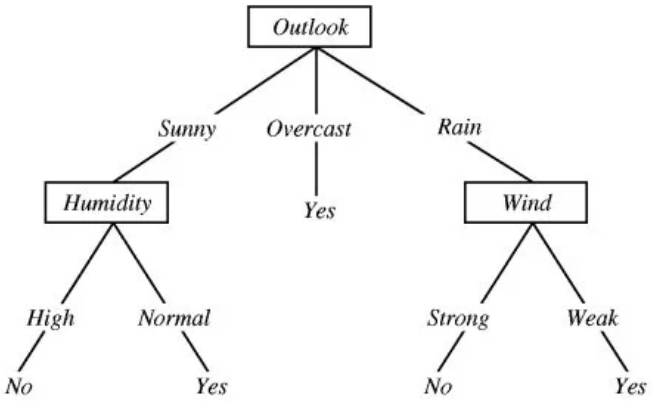

Figura 2.15: Árbol de decisión de tipo ID3 utilizado para predecir si se juega al tenis o no.

CART:

Estos son los árboles que más se utilizan puesto que son los que mejores resultados arrojan en cuanto a precisión. Se basan en el índice Gini, que responde a la siguiente fórmula:

$$Gini = 1 - \sum_{j} p_j^2$$

A diferencia del anterior árbol, se puede utilizar también en tareas de regresión (permite variables objetivo numéricas) y sólo puede tener dos ramificaciones por nodo.

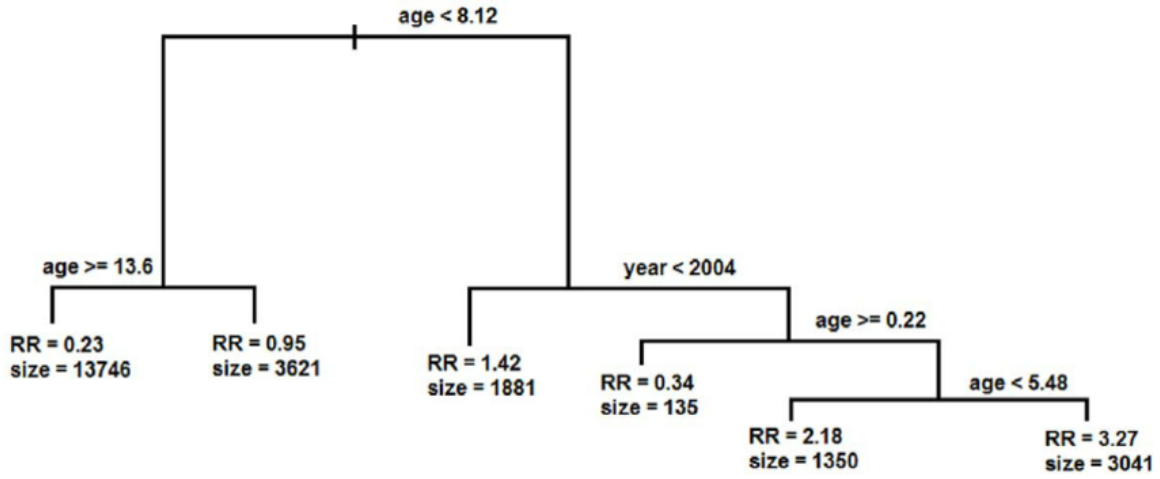

Figura 2.16: Árbol de decisión de tipo CART.

2.2.3. El preprocesamiento de datos

Tras explicar tanto el concepto de Machine Learning y describir los algoritmos más comunes, es importante hacer mención al preprocesamiento de datos. El preprocesamiento de datos es el proceso mediante el cual se modifica el dataset "crudo" (el que llega directamente desde las distintas fuentes de información) para adecuarlo a su uso en el algoritmo con el que se vaya a trabajar.

Este, pese a poderse considerar como un paso preliminar, es muy importante, ya que de él dependerá el éxito y la eficiencia del programa, a la vez que permitirá reducir la potencia de cómputo requerida al optimizar el uso de información.

En el preprocesamiento de información, son muy habituales las siguientes metodologías:

- Omisión de información perdida, ya sea eliminando la columna entera y introduciendo un valor determinado (normalmente 0) en sustitución de dicha información.
- Eliminación de aquellos atributos que no guarden una correlación relativamente alta con el atributo objetivo.
- Normalización de la información.
- Discretización de la información, con el objetivo de trabajar con rangos de valores en vez de valores continuos.
- Combinaciones de distintos atributos en uno sólo.

3 Dataset inicial y su preprocesamiento

Antes de estudiar los posibles algoritmos que puedan ser utilizados para acatar el objetivo principal del trabajo, que es analizar los patrones de comportamiento en una red de sensores IoT para detectar situaciones anómalas, es necesario ver qué datos tenemos, cómo están estructurados y como podemos modificarlos para conseguir una mayor sencillez a la hora de trabajar con ellos.

Debido a la simplicidad que Python aporta a la hora de programar algoritmos de Machine Learning, este será precisamente el lenguaje de programación empleado a lo largo de todo el trabajo.

3.1. Descripción de los datos en crudo

El dataset con el que se va a trabajar corresponde a una serie de grabaciones realizadas por un conjunto de sensores distribuidos por una casa inteligente [12]. Se ha escogido este dataset en concreto porque es uno de los pocos que se encuentran disponibles en Internet y es utilizado en la literatura científica.

Estos sensores son de varios tipos (de contacto, de presión, de movimiento, de luz, de humedad, de temperatura y de estado) y la distribución de ellos dentro de la vivienda está representada en la figura 3.1.

La vivienda está dividida en cinco partes, y los sensores se reparten a lo largo del domicilio de la siguiente manera:

- Cada una de ellas presenta un sensor de movimiento, salvo el dormitorio, el cual cuenta con dos.
- En el hall existe un sensor de contacto instalado en la puerta de entrada.
- El baño cuenta con un sensor de luz, otro de humedad y otro de temperatura.
- El dormitorio tiene dos sensores de presión: uno en la cama y otro en la báscula.
- La cocina presenta un sensor de luz.
- En el salón se ubican tanto un sensor de presión en el sofá como otro de contacto en la puerta que conecta con el balcón y otro de luz en la televisión.
- Los principales electrodomésticos que no han sido mencionados cuentan con sensores de estado que indican si estos se encuentran activos o no. Por ejemplo, la cafetera, la aspiradora, etc.

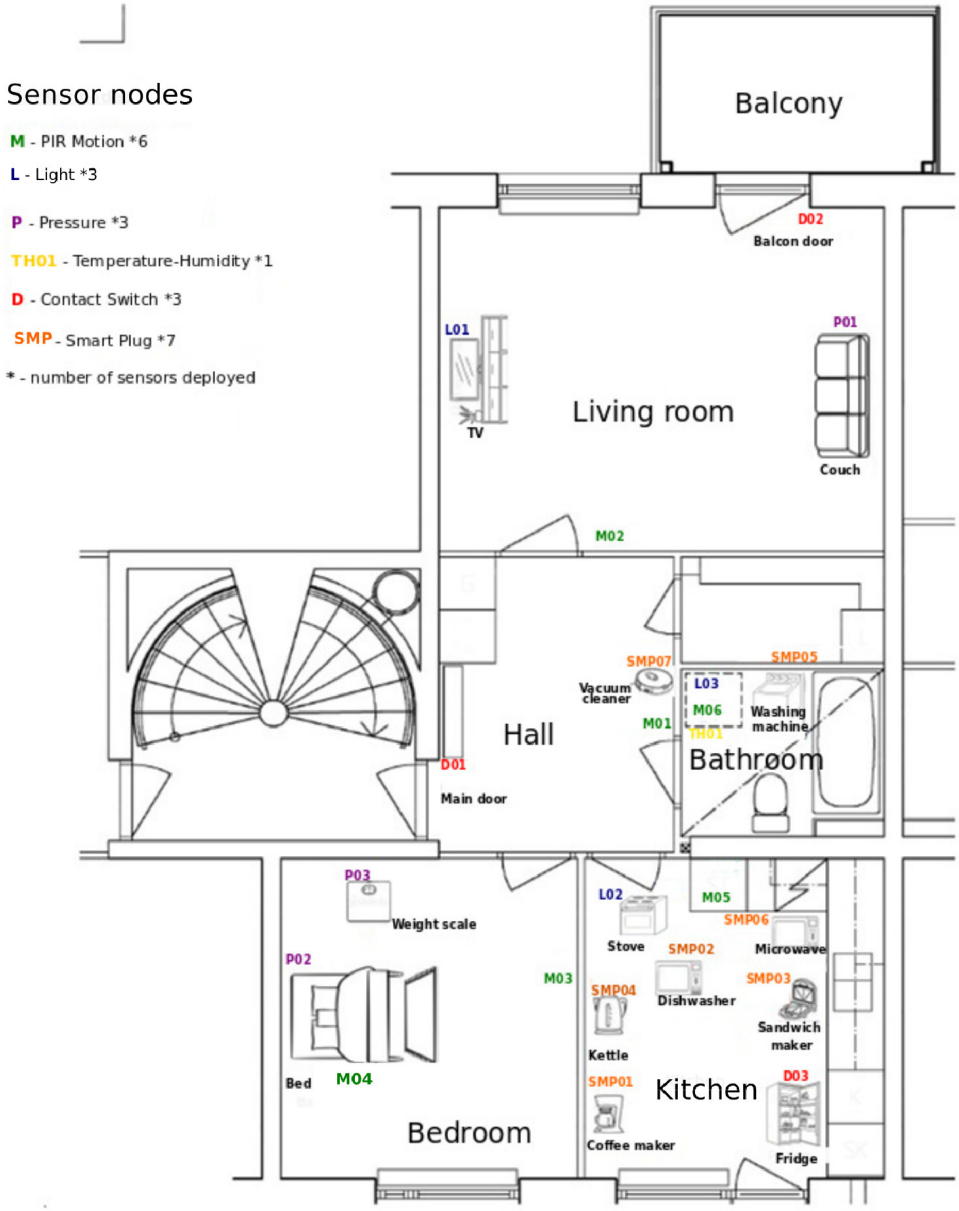

Figura 3.1: Plano de la distribución de los sensores dentro de la vivienda

Una vez descritos los distintos sensores existentes y su distribución, se procederá a explicar cómo está compuesto el dataset, el cual cuenta con tres archivos:

- Sensor.csv, que recoge información acerca de los distintos tipos de sensores instalados.
- Sensor_sample_int.csv, que contiene las medidas de todos los sensores que arrojan información con formato int.
- Sensor_sample_int.csv, que contiene las medidas de todos los sensores que arrojan información con formato float.

Si se imprime el contenido del primer archivo por pantalla, se puede observar que tiene la siguiente forma:

	sensor_id	node_id	type	name
0	5894	541150	INT	corridor/ambience/motion
1	5895	541151	INT	bathroom/ambience/motion
2	7125	542381	INT	bathroom/ambience/light
3	5896	541152	INT	bedroom/bed/pressure
4	6127	541383	INT	livingroom/tv/light
5	6220	541444	INT	balcon/door/contact
6	6253	541509	INT	kitchen/fridge/contact
7	6632	541888	FLOAT	kitchen/coffeemaker/current
8	6633	541889	FLOAT	kitchen/sandwichmaker/current
9	6634	541890	FLOAT	kitchen/dishwasher/current
10	6635	541891	FLOAT	kitchen/kettle/current
11	6636	541892	FLOAT	bathroom/washingmachine/current
12	6896	542152	FLOAT	kitchen/microwave/current
13	6686	541885	INT	bedroom/ambience_under_the_bed/motion
14	6687	541883	INT	bedroom/weightscale/pressure
15	7139	542395	FLOAT	corridor/ilifeRobot/current
16	6222	541141	FLOAT	bathroom/ambience/humidity
17	6223	541141	FLOAT	bathroom/ambience/temperature
18	5887	541143	INT	kitchen/stove/light
19	5888	541144	INT	entrance/door/contact
20	5889	541145	INT	livingroom/couch/pressure
21	5891	541147	INT	livingroom/ambience/motion
22	5892	541148	INT	bedroom/ambience/motion
23	5893	541149	INT	kitchen/ambience/motion

Figura 3.2: Contenido de sensor.csv

Podemos ver que hay un total de 24 filas (desde la número 0 a la 23) que corresponden cada una a un sensor distinto. Cada sensor será representado por ID en los otros dos archivos, el cual es el número correspondiente a la columna "sensor_id", mientras que la columna "node_id" no tendrá presencia alguna. La columna "type" representa el formato que tomará la información medida por cada sensor, por lo que los valores tomados por los sensores que sean tipo INT se encontrarán en el archivo sensor_sample_int.csv, mientras que los que sean de tipo FLOAT estarán en el archivo sensor_sample_float.csv. Por último, en la columna "name" se describe esquemáticamente cada sensor mediante un formato X/Y/Z, donde X representa la parte de la casa en la que se encuentra instalado, Y describe si se trata de un mueble, electrodoméstico o un valor particular del ambiente en sí de la parte de la casa X; Z explica el tipo de sensor del que se trata (si es de luz, de presión, de movimiento, etc) o el estado de funcionamiento si Y describe a un electrodoméstico en cuestión.

Respecto a los archivos contenedores de todas las medidas tomadas por los sensores, ambos tienen el mismo formato y es el mostrado en la figura 3.3.

Cada fila cuenta con cuatro columnas distintas, las cuales son: "value_id", que es único para cada entrada; "sensor_id", que es el identificador de cada sensor que se encuentra descrito en el primer archivo; "timestamp", que indica la fecha y hora en la que se ha tomado cada valor y "value", que es el valor de la medición en sí.

En las figuras 3.4 y 3.5 se muestra una representación gráfica de los valores tomados por cada columna en cada archivo, salvo los valores de "timestamp", los cuales no son numéricos y no se pueden plasmar sobre una gráfica.

Out[4]:

	value_id	sensor_id	timestamp	value
0	315318	6222	2020-02-26 11:22:36.417725	17.78
1	315319	6223	2020-02-26 11:22:36.426686	26.51
2	315320	6223	2020-02-26 19:36:36.586117	27.27
3	315321	6222	2020-02-26 19:36:36.657636	51.52
4	315322	6223	2020-02-26 19:36:37.704286	27.26

Figura 3.3: Ejemplo de cinco entradas del archivo sensor_sample_float.csv

En el caso de la figura 3.4 se observa que "value_id" presenta multitud de valores dispares, algo que se tratará más adelante. En la gráfica de "sensor_id" se representa el número de veces que aparece dentro del dataset cada sensor que arroja información de tipo INT. El pico de esta gráfica estaría representado por el número total de filas del array de datos en cuestión en las cuales el ID del sensor es menor a 6000. Finalmente, la gráfica relacionada con los valores de la columna "value" muestra que la gran mayoría de valores se encuentran cercanos a 0 (en este caso, serán sobre todo 1's y 0's).

Respecto a la figura 3.5, las dos primeras gráficas siguen el mismo razonamiento que en el caso anterior. La principal diferencia se encuentra en la gráfica de la columna "value", la cual está totalmente distorsionada debido a la presencia de ciertos valores muy altos en términos absolutos.

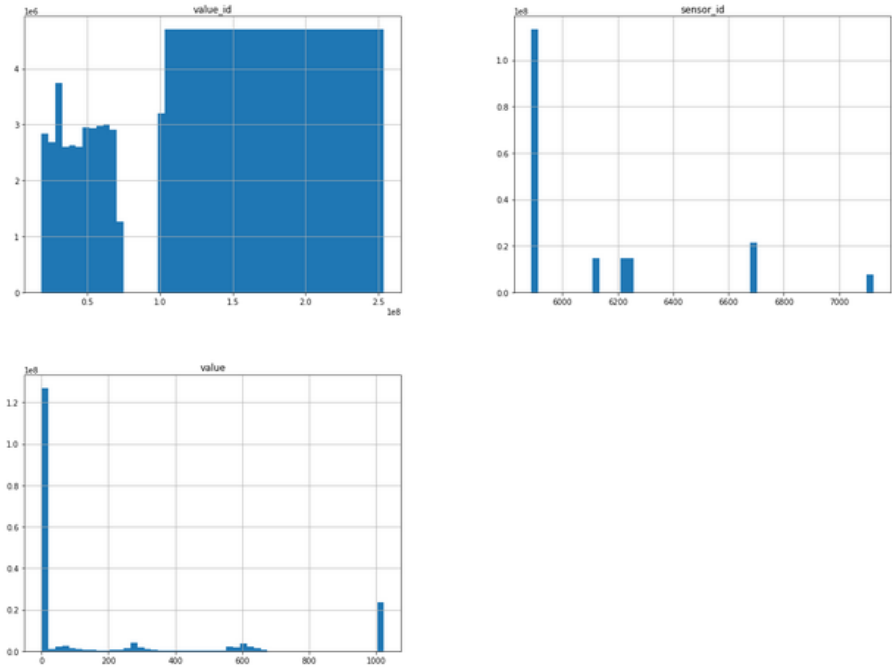

Figura 3.4: Representación gráfica de la distribución de valores por columna en el archivo sensor_sample_int.csv.

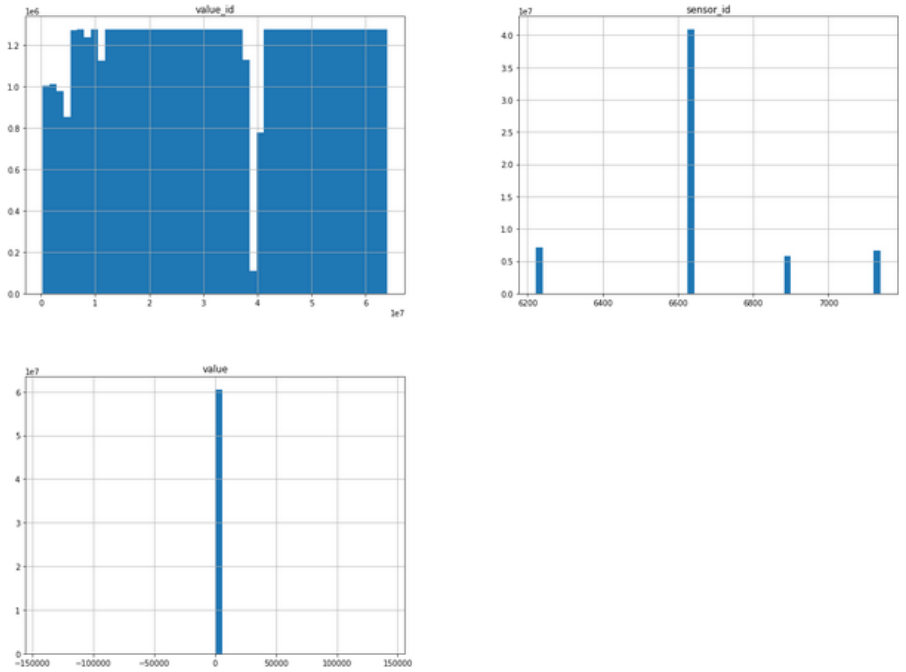

Figura 3.5: Representación gráfica de la distribución de valores por columna en el archivo sensor_sample_float.csv.

3.2. Preprocesamiento

Lo primero que se ha de hacer es decidir qué sensores resultan interesantes para desarrollar la aplicación deseada, que en este caso es un sistema de detección de situaciones extraordinarias. Si se aniliza cuáles son aquellos sensores que guardan sus datos en formato tipo float vemos que corresponden todos con electrodomésticos como la cafetera, la sandwichera, el lavavajillas o el microondas, así como la humedad y la temperatura del lavabo. Obviamente ninguno de estos sensores aportan información realmente útil para la tarea final, por lo que resulta muy conveniente descartar el archivo "sensor_sample_float.csv" entero, de tal forma que se pueda trabajar únicamente con uno, sin necesidad de realizar un paso preliminar que unifique los dos datasets.

En cuanto al segundo archivo, el que contiene datos en formato INT, también existen algunos sensores que resultan indiferentes, los cuales son:

- El sensor que mide la actividad de la TV (si está encendido o no). No guarda ninguna correlación con un posible allanamiento.
- El sensor que mide el contacto con el frigorífico. De la misma forma que un posible ladrón no interactuaría con la TV, tampoco lo haría con el frigorífico.
- El sensor que mide la presión con la báscula.

Una vez llevado a cabo este proceso se obtiene un nuevo dataset el cual es menos pesado y permite realizar operaciones sobre él de forma más fluída y menos demandante para la computadora.

El siguiente paso que se lleva a cabo es eliminar la columna de "value_id". Esta columna es independiente para cada entrada, es decir, tiene un valor único. De ello se deduce que en ningún caso podrá tener algún tipo de correlación con la variable objetivo, la cual es "value", por lo que se procede a eliminarla del dataset. Al igual que el paso anterior, esto supone una reducción del dataset al 75 % del tamaño, puesto que se ha suprimido una de las 4 columnas al completo.

Tras la supresión de la columna y los sensores mencionados, se procede a aplicar una función a la columna "timestamp" que permita trabajar con ella. "timestamp" es realmente importante, ya que, como se menciona en el subapartado anterior, aporta información acerca de la hora a la que se realiza una medición en concreto. El problema es que dicha información es de tipo objeto, cuyo formato es "año-mes-dia hora:minuto:segundo.milésimas", con el que es complejo trabajar.

Para solucionar esto, primero se elude lo relacionado a la fecha. La razón de esto es que eliminaría cualquier tipo de correlación con el valor final, porque el hecho de tenerla en cuenta supondría hacer único cada valor de timestamp; el día concreto al que corresponda cada medición no es lo realmente importante, sino que lo que conviene es estudiar el momento del día en cuestión. Por ejemplo, a la hora de comparar los distintos valores que toma el sensor de presión de la cama, no es primordial el saber exactamente el día del año, sino más bien la hora, porque una persona en concreto tiende a mantener una serie de costumbres y rutinas, que son precisamente las que permitirían acatar el objetivo principal de este trabajo. Respecto a esto, sí podría llegar a ser interesante separar los datos que correspondan a días no laborales y festivos, pero esto supondría un esfuerzo mucho más exigente a nivel computacional para simplemente lograr un ligero aumento en el porcentaje de acierto, debido a que no se conoce el calendario laboral del individuo que habita en la vivienda (los calendarios y horarios laborales pueden llegar a variar mucho en función del tipo de trabajo de cada persona, y aventurarse a asumir uno en cuestión podría ser incluso perjudicial para el resultado buscado).

Después de descartar la fecha correspondiente a cada entrada, se divide el tiempo del día en minutos. es decir, en 1440 unidades de tiempo (24 horas del día multiplicado por los 60 minutos que contiene una hora), de tal forma que no se tendrán en cuenta segundos ni microsegundos. Para ello, se aplica a la columna "timestamp" la siguiente fórmula de conversión:

$$timestamp = hora * 60 + minuto$$

De esta forma, el dataset pasa de tener la forma ilustrada en la figura 3.3 a la mostrada en la figura 3.6.

Obviando la primera columna, que es una especie de residuo que surge al guardar el dataset y que posteriormente se elimina antes de aplicar cualquier algoritmo de Machine Learning, se puede observar que efectivamente los valores de "timestamp" son ahora numéricos. Por ejemplo, el primer valor mostrado es 1325, el cual equivaldría a las 22:05,

puesto que 1325/60 es 22'08333, siendo la hora la parte par del cociente y 5 el resultado de multiplicar la parte decimal por 60. Siguiendo con los otros valores, 905 equivaldría a las 15:05, 1070 a las 17:50, etc...

	Unnamed: 0	sensor_id	timestamp	value_mod
70032548	70032548	6220	1325	0
93803946	93803946	6220	905	0
88413930	88413930	6220	1070	0
55093516	55093516	6220	1337	0
121111739	121111739	6220	1219	0

Figura 3.6: Ejemplo de cinco valores tomados por el sensor 6220 tras las modificaciones anteriores

Finalmente, después de haber modificado todas las columnas que corresponden a las variables independientes, la única que queda por estudiar y cambiar es la del valor objetivo. Para trabajar correctamente y que el esfuerzo computacional no sea prácticamente infinito en algunos algoritmos de Machine Learning o incluso inaplicable en otros, es necesario que todos los sensores arrojen sus datos en binario. Esto se cumple por defecto en la mayoría, pero hay cuatro en particular que no, y son los sensores de luz de la cocina y del lavabo y los sensores de presión del sofá y de la cama.
Respecto a los sensores de luz, la distribución de valores en ambos es similar y es de la forma mostrada en la figura 3.7:

Se puede ver que la mayoría de valores corresponden a uno en concreto que se encuentra cerca del 1000, así que se imprimen un conjunto de filas aleatorias con el objetivo de averiguar cuál es dicho número. Al llevar esto a cabo, se observa que dicho número es el 1024 (véase la figura 3.8). Como no existe ningún tipo de información acerca de cómo funcionan exactamente los sensores, la única opción es tratar de extraer una conclusión lógica a este fenómeno, por lo que se deduce que si la gran mayoría de valores equivalen a 1024, ese ha de ser el estado natural de la luz en la cocina y en el baño, siendo cualquiera de los otros valores que pueda tomar dichos sensores una representación de cambio en el estado base de la iluminación, por lo que se aplica una función al dataset que hace que, para todo valor correspondiente a los sensores cuyos ID sean 7125 o 5887, se sustituirá 1024 por 1, mientras que de lo contrario se sustituirá por 0.

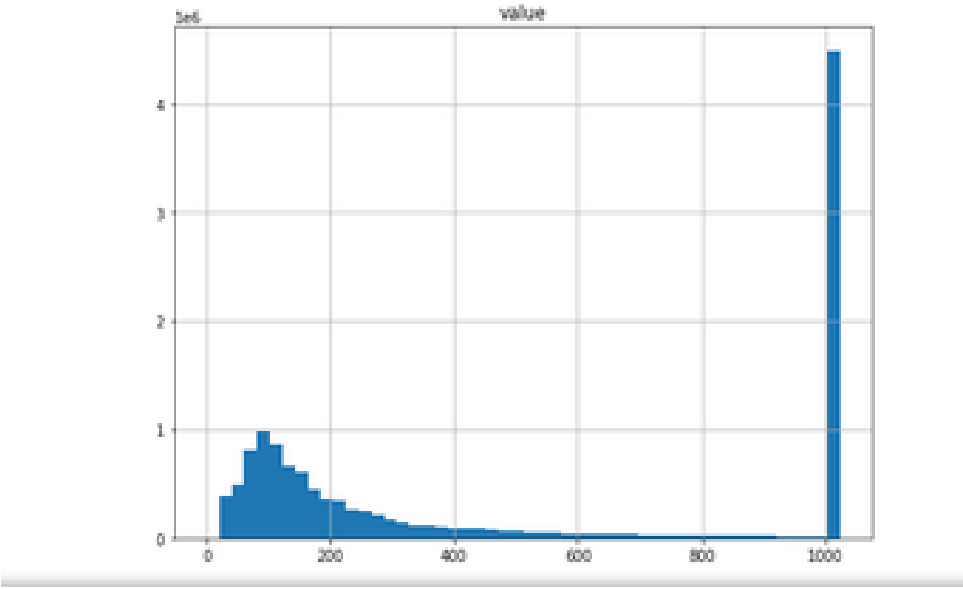

Figura 3.7: Distribución de valores del sensor 5887, el correspondiente a la luz de la cocina

En cuanto a los otros dos sensores, el procedimiento a seguir es más complicado, ya que no presentan ningún patrón en cuanto a la distribución de valores que pueda servir de ayuda para realizar este paso de la forma más adecuada posible, así que ambos se tratan por separado. En cuanto al sensor de presión del sofá (véase la figura 3.9), se aprovecha ese inicio de "pendiente" en el valor 200 para suponer que cualquier valor superior a este será sustituído por 1, mientras que los otros por 0. No existe ninguna razón matemática para escoger este límite en cuestión, pero cualquier otro método relacionado, por ejemplo, con la media o la mediana supondría un esfuerzo extra que arrojaría datos tanto o más imprecisos debido a la falta de un patrón definido en dicha gráfica.
Respecto al sensor 5896 (véase la figura 3.9), ocurre exactamente lo mismo: valores aparentemente distribuidos de forma aleatoria. En este caso se toma el valor más común, el cual es 620, para marginar a los futuros 1's y 0's.

Cabe mencionar que llevar este paso a cabo comprometerá en gran medida el resultado final, puesto que en cada aproximación o suposición que se hace se está perdiendo correlación con el resultado. Por ejemplo, quizás un valor arrojado por el sensor de presión del sofá que sea superior a 200 no tiene por qué implicar un 1 (es decir, "presión detectada").

Para rematar el preprocesamiento, se crea un nuevo dataset, el cual es idéntico al que se ha obtenido tras aplicar todos los cambios comentados hasta este punto, con la diferencia de que se sustituye el 1 por "Yes" y el 0 por "No". Aunque el significado del dataset quede intacto, esto se realiza porque uno de los algoritmos que se aplicarán será un árbol de decisión de tipo ID3, el cual sólo puede trabajar con valores no numéricos. Además, para poder ejecutar el programa que se estudiará posteriormente ha sido necesario cambiar la columna de timestamp de tal manera que la unidad de tiempo no sea un

minuto, sino 15 minutos (o cuarto de hora), es decir, antes el tiempo podía tomar 1440 valores (24*60), ahora puede tomar 96 (1440/15).

	value_id	sensor_id	timestamp	value
119407527	186404026	7125	2020-06-19 17:25:30.973162	1024
141465926	208462425	7125	2020-07-14 13:31:52.974238	1024
137740643	204737142	7125	2020-07-11 06:13:54.300738	1024
153966971	220963470	7125	2020-07-26 03:31:05.746132	1024
183711335	250707834	7125	2020-08-23 12:00:05.925563	598
140934695	207931194	7125	2020-07-14 02:21:08.310308	1024
131375978	198372477	7125	2020-07-05 12:13:26.055205	29
142689506	209686005	7125	2020-07-15 15:19:06.418630	1024
166913301	233909800	7125	2020-08-08 07:09:02.844371	1024
176766748	243763247	7125	2020-08-17 03:51:04.697498	1024
104083222	171079721	7125	2020-06-06 10:39:52.399083	841
160650449	227646948	7125	2020-08-01 13:34:14.317355	1024
123916263	190912762	7125	2020-06-23 15:19:43.310205	742
81069617	148066116	7125	2020-05-16 11:31:34.861571	480
168713932	235710431	7125	2020-08-09 20:58:36.813944	1024
176672618	243669117	7125	2020-08-17 01:47:21.020678	1024
100904811	167901310	7125	2020-06-03 16:43:23.515231	1024
135494206	202490705	7125	2020-07-09 03:44:40.360149	1024
124367946	191364445	7125	2020-06-24 00:44:00.632640	1024

Figura 3.8: Varias filas aleatorias del sensor 7125, el correspondiente a la luz en el baño. Mencionar que esta captura es anterior al inicio del preprocesamiento, por eso las columnas no han sido modificadas aún.

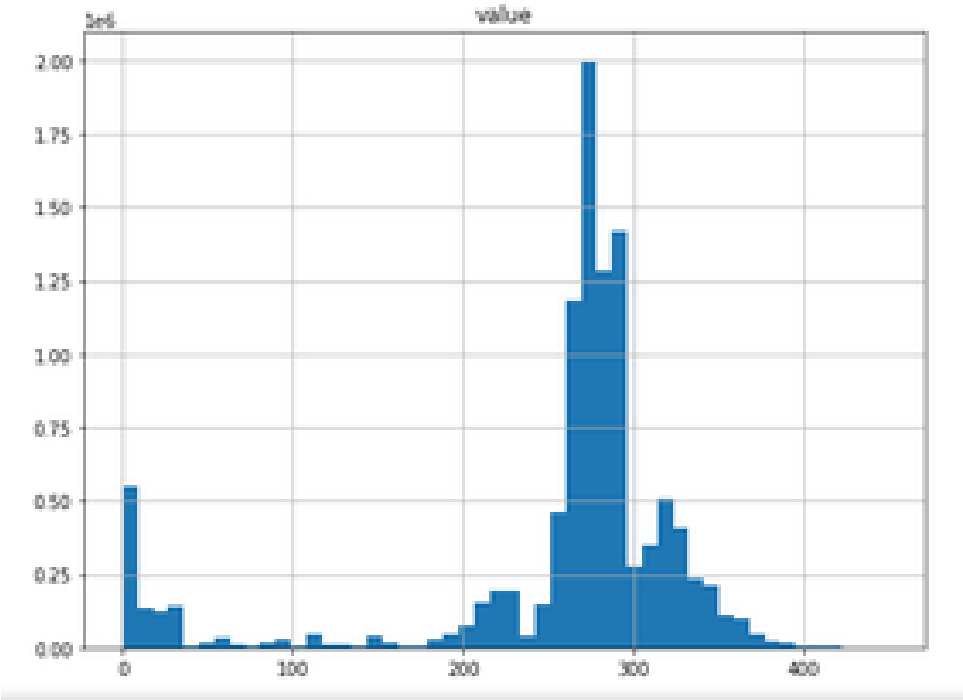

Figura 3.9: Distribución de valores del sensor 5889, el correspondiente a la presión en el sofá.

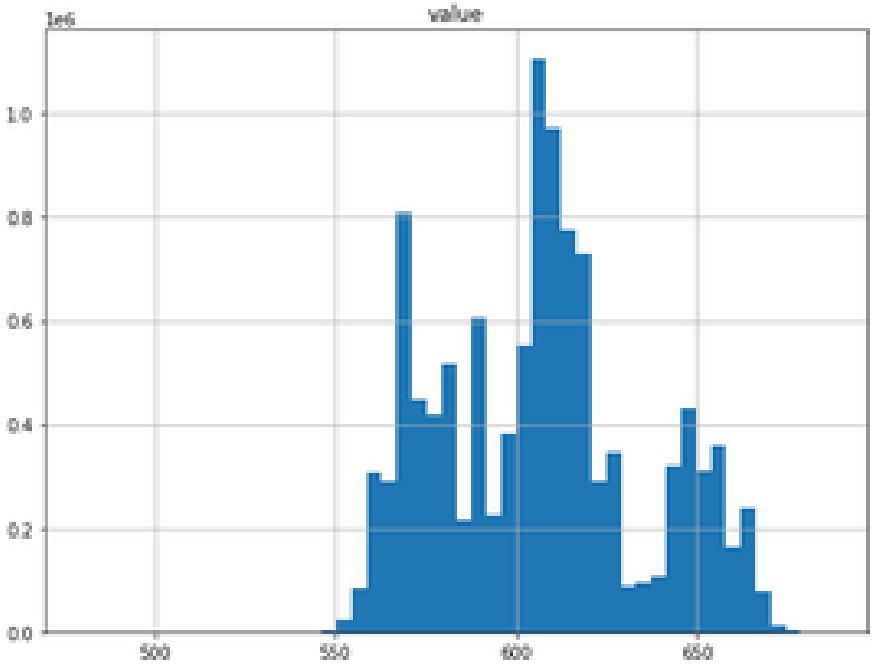

Figura 3.10: Distribución de valores del sensor 5896, el correspondiente a la presión en la cama.

4 Elección del algoritmo ideal

En este apartado se describirán cada uno de los algoritmos utilizados y su eficiencia en términos de porcentaje de acierto respecto a un dataset de prueba formado en base a la extracción de entradas aleatorias del dataset total.

4.1. Árbol de decisión de tipo ID3

Tal y como se explica en el apartado del estado del arte, los árboles de tipo ID3 se basan en los conceptos de entropía y de ganancia de información para ser construidos. El principal problema con este tipo de árboles es que no existe ninguna librería que los implemente, por lo que para crearlo se utilizó un código publicado en Internet [13], al cual se realizaron un conjunto de modificaciones para lograr un correcto funcionamiento.

La secuencia que sigue el programa para crear el árbol es la siguiente:

1. Se definen las funciones que calculan la entropía y la ganancia de información para cada dataframe.

2. Se aplican dichas funciones al dataframe para ver cuál es el atributo con mayor ganacia de información, es decir, el que "mejor clasifica" el valor de los sensores. En este caso, se trata de la columna "sensor_id".

3. Para cada nodo se aplica el mismo proceso, en el que el dataframe utilizado es aquel que cumple con los valores ya definidos en las anteriores ramas, y calculando la ganancia de información para todos los atributos restantes, aquellos que no se han utilizado para clasificar aún. Esto último se lleva a cabo si se dan distintos valores para la variable objetivo; de lo contrario, se asigna directamente a esta rama el único valor que puede tomar. Por ejemplo, teniendo en cuenta que la raíz de este árbol en concreto es "sensor_id", para aquella rama en la que este toma el valor 7125, el dataframe utilizado serán todas las filas del dataframe completo en las cuales el atributo anterior siempre valga 7125. Como ya no existen más atributos aparte del timestamp, este será el último que se utilizará para crear subramas.

4. Una vez se han construido todos los nodos y se han establecido todas las secuencias de nodos (caminos), el valor será el más común dentro del dataframe construído en base a las especificaciones establecidas a lo largo de dicha secuencia de nodos. Siguiendo con el ejemplo anterior, en el que "sensor_id" valía 7125, para un valor de timestamp cualquiera, tal como 64, se calcularía el porcentaje de presencia de Yes y el porcentaje de No, ambos respecto a la totalidad de filas. El valor que predicirá el árbol cuando las variables de entrada coincidan con estas dos (sensor_id=7125, timestamp=64) será el que mayor porcentaje presente, el valor más común de los dos. Cabe señalar que se está dando por supuesto que dicho sensor toma distintos valores; de tratarse del caso contrario no se realizarían más cálculos, sino que directamente se asumiría que si el sensor es el 7125 siempre se tomará dicho valor unánime.

Finalmente, el árbol creado tiene la forma monstrada en la figura 4.1:

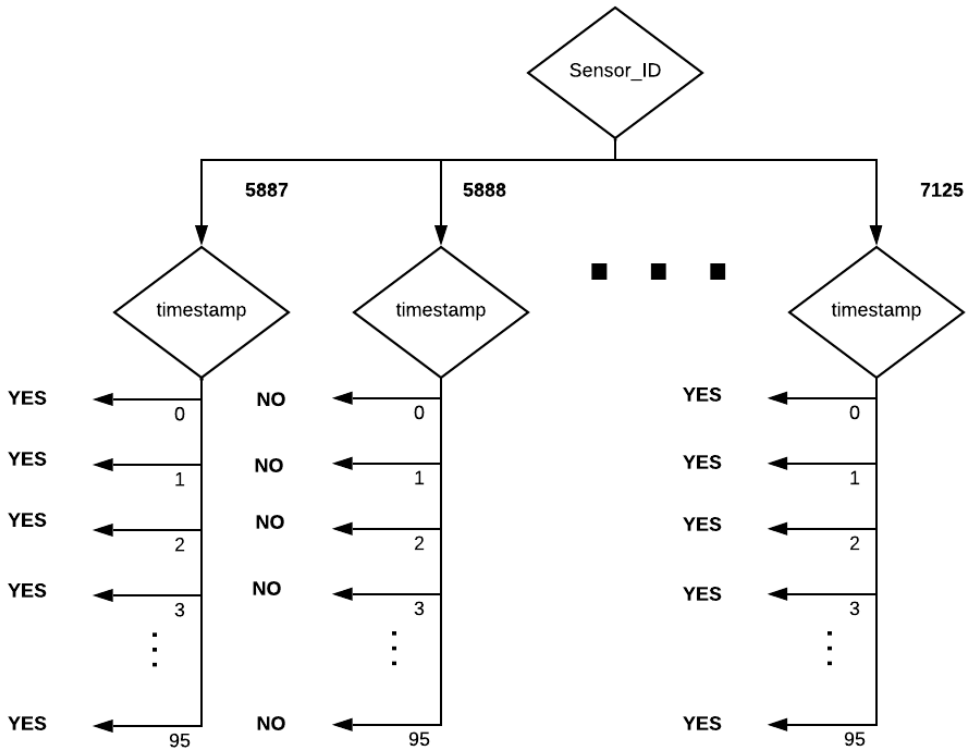

Figura 4.1: Simplificación del árbol de tipo ID3 resultante.

Esta es una versión resumida del árbol en cuestión, puesto que para el primer atributo existen doce ramas (una para cada sensor), y para cada una de ellas existen otras 96 (que corresponden al número total de fracciones de tiempo).

Para comprobar la eficiencia de este algoritmo se define una función que utilice el árbol de decisión para predecir resultados. Se divide el dataframe en dos partes: la que sirva para entrenar el algoritmo y la que sirva para ponerlo a prueba. Con el set de entrenamiento se construye un árbol y con el de prueba se calcula el porcentaje de acierto. El resultado final es de un 90'96 %.

Aunque en un principio pueda resultar un resultado bastante favorable, es necesario tener en cuenta que el porcentaje total de NO en el dataset total es de un 82 %, es decir, en el caso en el que se elaborase un programa que simplemente predijese NO a cualquier entrada de valores estaríamos consiguiendo un 82 % de acierto, por lo que es menester relativizar este valor en cuestión.

Además, según dictamina este árbol en concreto, la mayoría de los sensores sólo pueden tomar un valor, cuando en la práctica se sabe que no es cierto. En teoría, los sensores 5888, 5891, 5892, 5893, 5894, 5895, 6220 y 6686 sólo toman el valor NO.

Este tipo de errores se pueden deber a la poca cantidad de atributos que se relacionen con la variable objetivo y/o a la pérdida de correlación con el valor objetivo sucedida en el preprocesamiento de los datos. Respecto a esto último, si se da por hecho que cualquier persona estaría acostada en su cama a las tres de la madrugada, entonces el sensor de presión de la cama debería medir presencia siempre. Sin embargo, si analizamos la cantidad de 1's y 0's a las 3 a.m. se puede observar que esto no se cumple, encontrándose incluso el SÍ muy cerca del 50 % (figura 4.2).

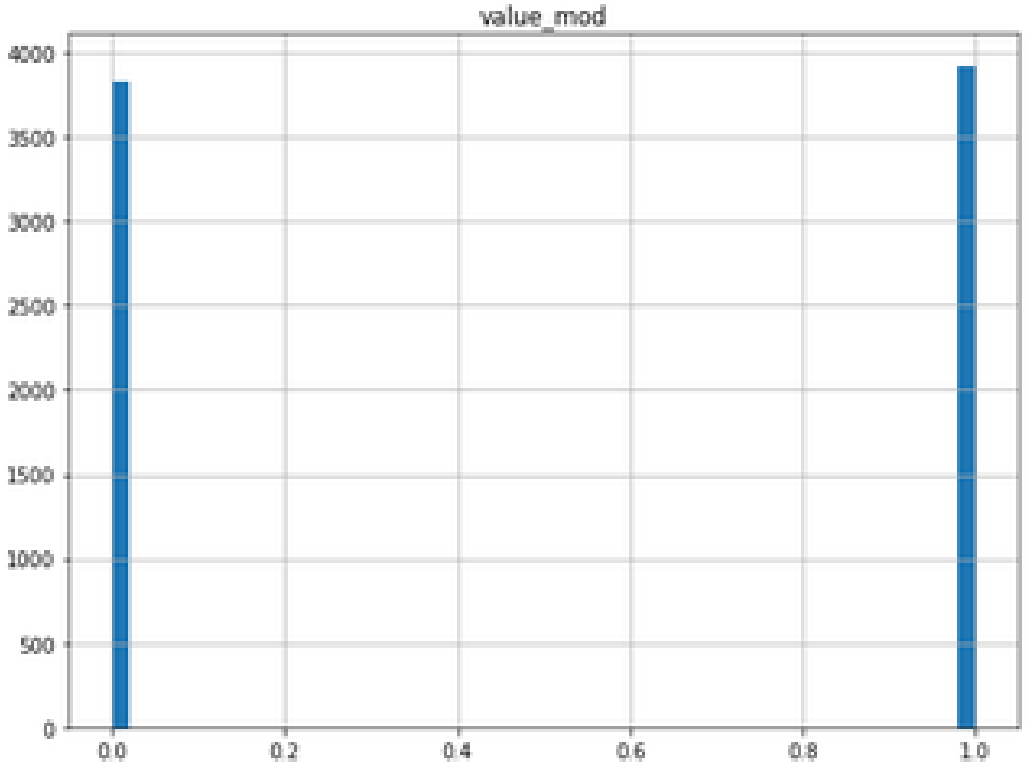

Figura 4.2: Sensor de presión de la cama a las 3 de la mañana.

Un recurso muy utilizado en Machine Learning que permite ahondar más en la precisión de un algoritmo es la matriz de confusión. La matriz de confusión es una herramienta utilizada para saber el tipo y la cantidad de aciertos y errores que está teniendo el algoritmo. En este caso, al existir únicamente dos posibles soluciones (Yes o No), la matriz es 2x2 y muestra la cantidad de verdaderos No, de verdaderos Yes, de falsos No y de falsos Yes.

En este caso la matriz de confusión generada es la mostrada en la figura 4.3.

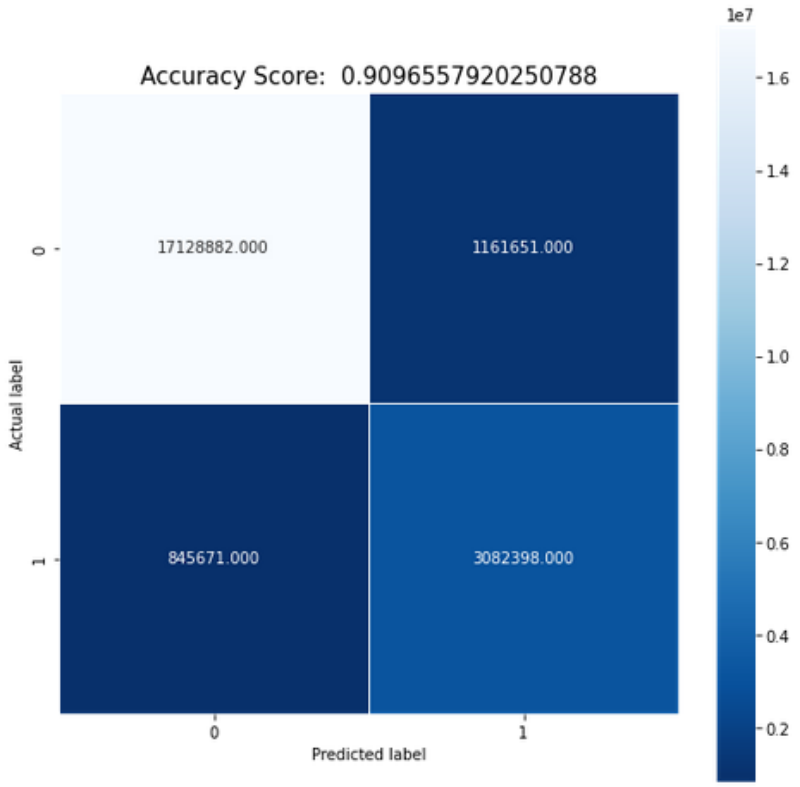

Figura 4.3: Matriz de confusión del árbol de decisión de tipo ID3.

En la diagonal principal se recogen los valores que han sido predecidos correctamente, mientras que en la otra diagonal los erróneos (falsos positivos y falsos negativos). Como se puede observar, existen muy pocos valores que sean Yes (1 en la matriz) y se hayan predecido como No (0 en la matriz), debido a la naturaleza del árbol resultante, explicada anteriormente.

Una vez estudiado dicho árbol, se procede a realizar una ligera modificación en el árbol de decisión, de tal forma que se utilice el timestamp como la primera variable para clasificar, y no el id del sensor. Para ello se modifica el código para que se escoja el atributo con menor ganancia de información. Finalmente el porcentaje de acierto es prácticamente el mismo.

4.2. Regresión logística

A partir de este punto ya no es necesario implementar códigos externos, debido a que la librería "scikit-learn" de Python aporta todas las funciones necesarias para construir cada uno de los árboles.

La construcción del programa sigue la siguiente secuencia:

1. Implementación de las librerías necesarias. Respecto a la que define el algoritmo de regresión, será sklearn.lnear_model.

2. Se carga el dataset y se separan las columnas referentes a los atributos y la variable objetivo en dos columnas distintas.

3. A su vez, se dividen ambas variables en otras dos totalmente independientes. Dos de ellas, que contarán con el 85 % del dataset entero, serán utilizadas para entrenar el algoritmo, mientras que las otras servirán para ponerlo a prueba.

4. Se lleva a cabo un ajuste de escalas para acomodar los valores del dataset a aquellos rangos en los que las funciones operen correctamente.

5. Con las variables independientes y objetivo de entrenamiento se construye la regresión logística.

6. Se calcula la precisión usando las variables independientes y objetivo de prueba.

En este caso, el resultado es de un 82'68 %, cifra que resulta extremadamente baja y se encuentra muy lejos de ser fiable.

Para investigar más en profundidad en el por qué de tan bajo porcentaje de acierto se imprime y analiza la matriz de confusión (figura 4.4).

Según dicha matriz, el algoritmo no es capaz de encontrar ningún tipo de correlación y asume que todos los valores a predecir son 0. Esto se deduce de que no existe ningún valor que haya sido predecido como 1, por lo que el porcentaje de acierto es exactamente igual al número total de 0's que existen en el dataset. Debido a esta razón, se concluye que el algoritmo no funciona adecuadamente y no es indicado para acatar la tarea final.

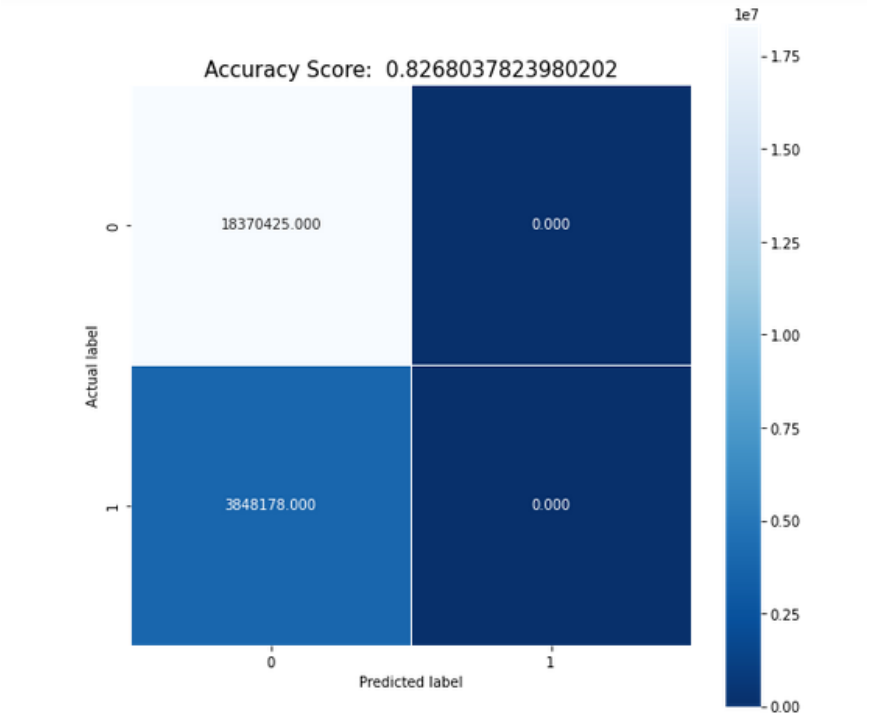

Figura 4.4: Matriz de confusión de la regresión logística.

4.3. Regresión lineal

En este caso se trata de construir un árbol de decisión basado en el concepto de la regresión lineal. El proceso de construcción es bastante similar al anterior: se importan las librerías pertinentes, se carga el dataset y se divide en las cuatro partes descritas, se ajustan las escalas y se entrena el algoritmo.

Sin embargo, en este caso surge un problema: las variables predecidas no son de carácter binario, no responden a 1/0 o a SÍ/NO, sino que al tratarse de una regresión lineal toman valores continuos. En la figura 4.5 se muestran enfrentados un conjunto de valores aleatorios reales (correspondientes al conjunto de prueba) y los predecidos, los cuales son todos números decimales.

Para solucionar esto, para todo valor que sea superior a 0'5 se considerará que el algoritmo ha dictaminado que es más probable que se trate de un 1 que de un 0, por lo que se le asigna dicho valor; y todo el que sea inferior será 0 de facto. Así se consigue transformar una tarea de regresión en una de clasificación.

La matriz de confusión es la mostrada en la figura 4.6, la cual tiene mayor sentido e indica un rendimiento potencialmente aceptable:

	test	pred
19479368	0	0.041522
10771669	0	0.000000
20482605	1	0.943684
5561230	0	0.148705
17251857	0	0.003513
20754466	0	0.037652
19281045	0	0.120821
11233535	0	0.002651
13395725	0	0.408271
2177482	0	0.005119
18529903	0	0.006854
3549785	0	0.018492
19524883	0	0.000000
15585885	0	0.011517

Figura 4.5: Valores reales enfrentados a su predicción.

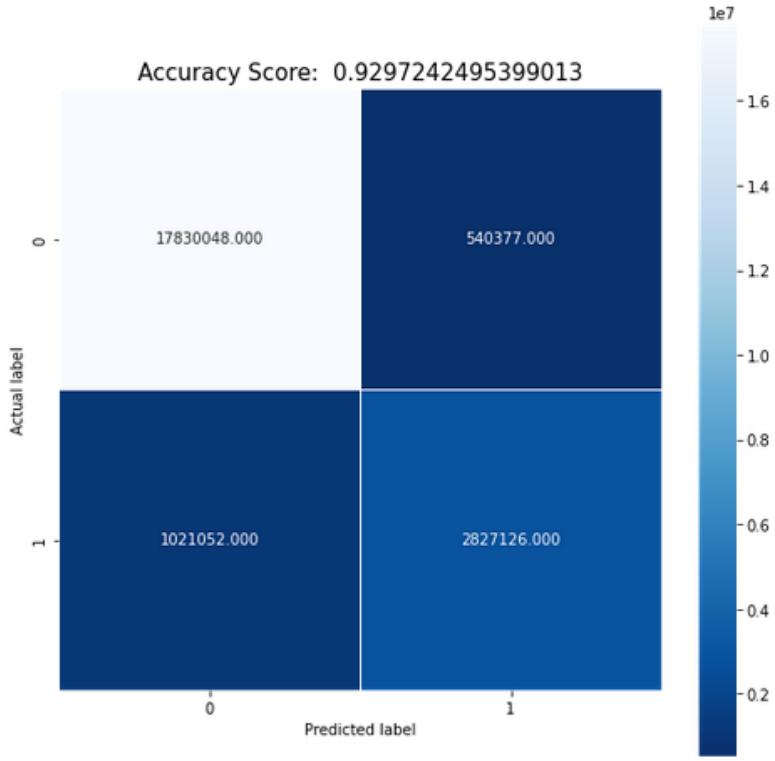

Figura 4.6: Matriz de confusión de la regresión lineal.

El porcentaje de acierto es del 92'97 % aproximadamente, lo que supone una mejora respecto a los anteriores algoritmos.

En cuanto a la forma del árbol, su impresión completa es la mostrada en la figura 4.7.

Debido al tamaño del dataset, el número de relaciones lógicas establecidas es muy elevado, y se materializa en una gran cantidad de nodos y mucha profundidad. Imprimir el árbol entero y mostrarlo adecuadamente sería imposible. Pese a ser posible limitar la profundidad del árbol imprimido, esto se hace desde la misma creación de este, lo cual afectaría al resultado y no sería representativo de la realidad.

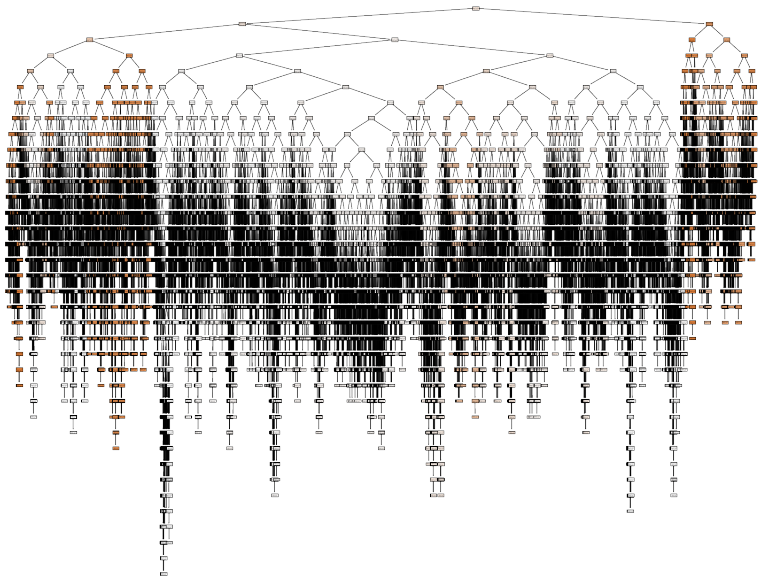

Figura 4.7: Forma completa del árbol de decisión basado en la regresión lineal.

4.4. Árbol de decisión de tipo CART

El proceso seguido para la construcción de este árbol coincide con el de los dos anteriores, y a diferencia del anterior caso, este es un tipo de árbol que puede trabajar tanto con valores continuos como binarios, por lo que no necesitará un paso intermedio de conversión de los datos a binario.

Lo que resulta más sorprendente de este algoritmo es el resultado de precisión que arroja: 92'97 %, es decir, es exactamente el mismo que en el anterior apartado. Si observamos la matriz de confusión (figura 4.8) se puede ver que, utilizando funciones distintas ("DecisionTreeRegressor() en el anterior apartado y "DecisionTreeClassifier(criterion='gini')" en este) los cuatro elementos de la matriz encajan. Sí es cierto que la construcción de los datasets de entrenamiento y prueba se realizan de la misma forma en los dos casos, pero el árbol resultante debería ser distinto y, entonces, el número de aciertos y fallos también.

Como es de esperar, la forma del árbol coincide con la del anterior también (figura 4.7). La explicación más lógica a este suceso es que el árbol destinado a tareas de clasificación es una particularización del árbol destinado a tareas de regresión, y este sufre el mismo proceso de construcción debido a que detecta que el atributo objetivo toma dos valores únicamente.

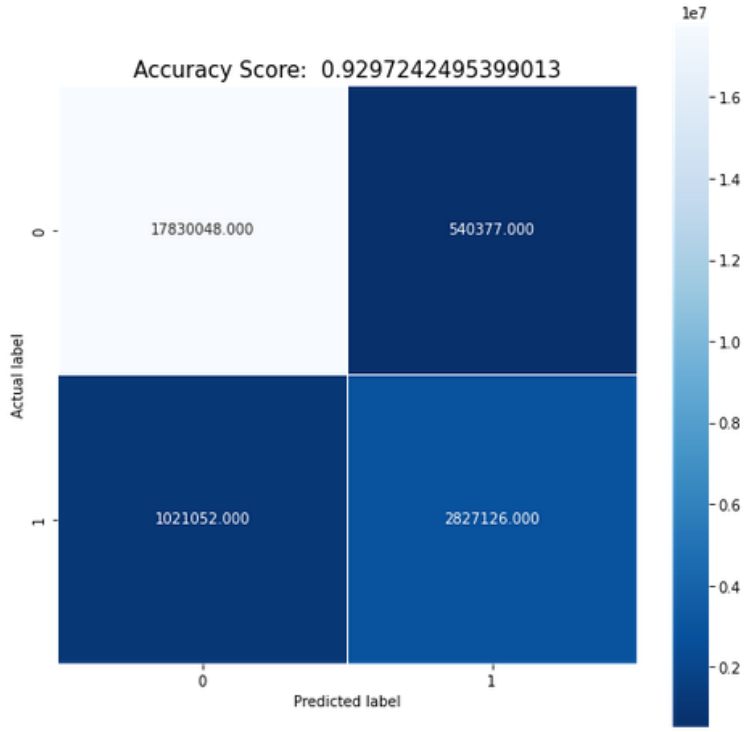

Figura 4.8: Matriz de confusión del árbol de decisión de tipo CART.

4.5. Elección final

Para elegir el algoritmo más adecuado se tomarán como referencia los porcentajes de acierto que han tenido cada uno de ellos en las pruebas a las que fueron sometidos. En la siguiente tabla se plasman cada uno de estos porcentajes:

Tipo de algoritmo	**Porcentaje de acierto**
Árbol de decisión de tipo ID3	90'96 %
Regresión logística	82'68 %
Árbol de decisión basado en la regresión lineal	92'97 %
Árbol de decisión de tipo CART	92'97 %

La regresión logística ha sido la menos eficiente, mientras que los mejores resultados provienen del árbol de decisión de tipo CART y el basado en la regresión lineal, los cuales presentan el mismo porcentaje de acierto.

Ante este suceso, no existiría diferencia alguna a la hora de escoger uno u otro algoritmo, pero en la teoría se sabe que los árboles de decisión de tipo CART son los que mejores resultados suelen arrojar, así que por esta razón será este último el elegido.

5 Elaboración del programa final

Tras escoger el árbol de decisión de tipo CART como el más acertado a la hora de predecir valores, se procede a codificar el programa que servirá para detectar situaciones anómalas en la vivienda.

El funcionamiento de dicho programa es el mostrado en el diagrama de flujos de la figura 5.1.

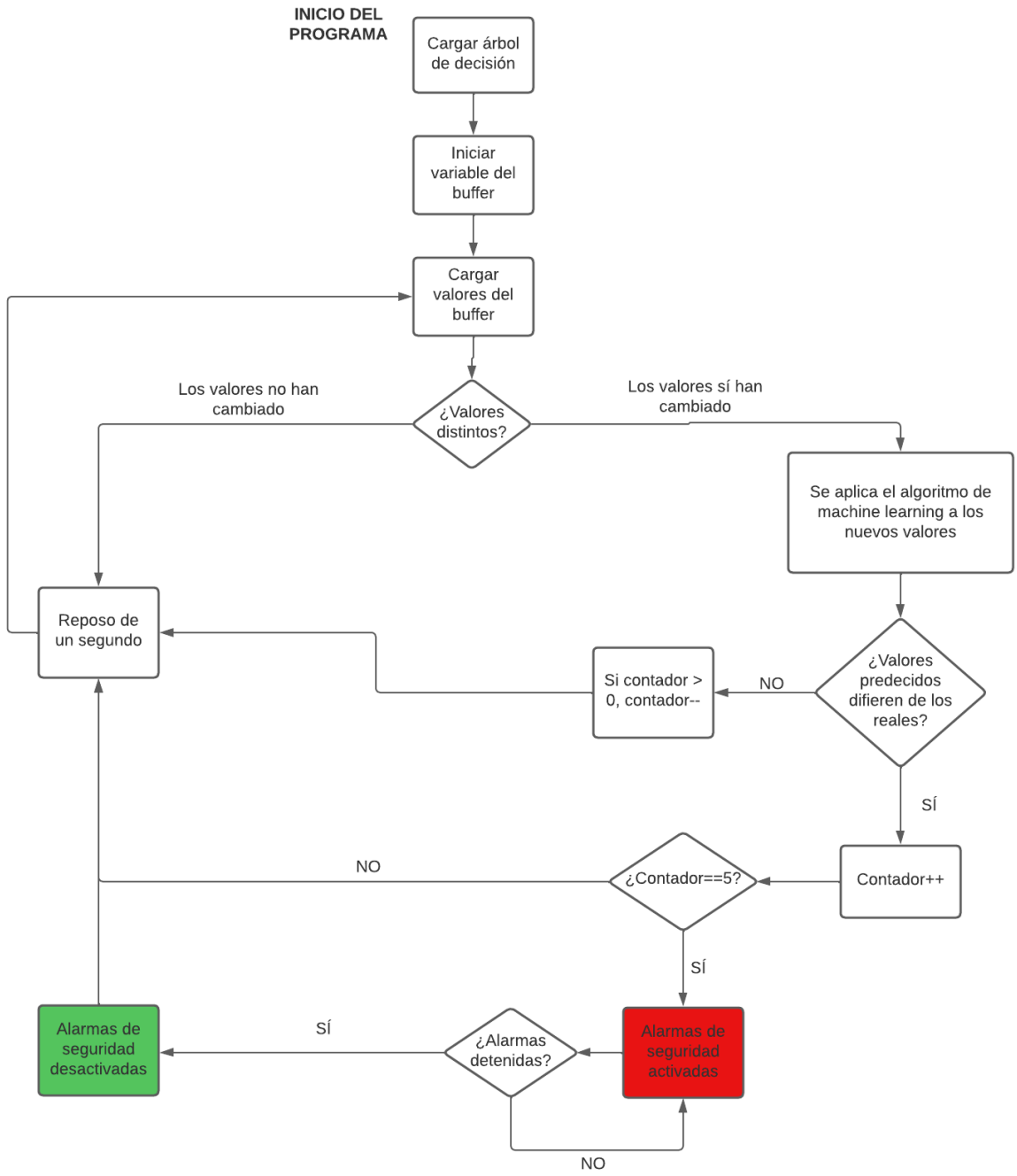

Figura 5.1: Diagrama de flujos del programa creado.

Lo primero que se hace es cargar el árbol de decisión de tipo CART, el cual previamente había sido exportado a un archivo gracias a la librería "pickle" de Python. Posteriormente, se iniciará la variable que contendrá los valores del buffer.

Este buffer es un archivo de extensión ".csv", al igual que el dataset, donde los sensores arrojarían sus datos para que sean recogidos y procesados por el programa. El buffer tiene un formato idéntico al dataset ya preprocesado, es decir, tres columnas: "timestamp", "sensor_id" y "value_mod".

Una vez realizados estos dos pasos, el programa entra en un modo bucle del que sólo podrá salir en caso de que se produjese alguna emergencia.

El programa procede a cargar nuevamente los valores que se encuentran registrados en el buffer para compararlos con los que había previamente. Si no detecta ningún cambio, el programa se suspende durante un segundo. Esto se lleva a cabo con el objetivo de ahorrar potencia de cómputo y ejecutar el algoritmo de Machine Learning sólo si es necesario. De detectar cambios en el buffer, entonces sí se ejecutaría el árbol de decisión y se valorarían los resultados.

En este punto se añade la variable "contador". Su función es la de controlar los posibles errores que pueda cometer el algoritmo en sus predicciones. Por ejemplo, si en un momento determinado el árbol predice que un sensor toma el valor '0' cuando su estado natural en ese momento del día es '1', el contador permitiría que el algoritmo se "equivocase" un número determinado de veces (en este caso particular, cinco veces). También, de ocurrir cualquier error en los sensores, este contador podría ayudar de la misma forma.

Así pues, cuando se estudian los valores predecidos, estos son comparados con los reales, los medidos por los sensores. Si estos difieren, se incrementa el contador y se procede a dejar el programa en reposo durante un segundo. Si el contador llega al valor de cinco, se iniciaría el sistema de seguridad y el programa quedaría suspendido hasta que este fuese detenido manualmente. En ese momento se reiniciaría el valor del contador y el programa volvería a la normalidad.

De coincidir valores reales y predecidos, simplemente se restaría una unidad al valor de contador si este es superior a 0, con la intención de evitar valores infinitamente negativos del contador; se reposaría un segundo y se volvería al punto de inicio.

Para ejemplificar, se añadirán una serie de ejemplos del funcionamiento del programa en la práctica:

Suponiendo que están registrados en el buffer los valores mostrados en la figura 5.2

sensor_id	timestamp	value_mod	nombre
5894	0	0	Sensor de movimiento del pasillo
5895	0	0	Sensor de movimiento del servicio
7125	0	0	Sensor de luz del baño
5896	0	0	Sensor de presion de la cama
6220	0	0	Sensor de contacto de la puerta del balcon
6686	0	0	Sensor de movimiento debajo de la cama
5887	0	0	Sensor de luz de la cocina
5888	0	0	Sensor de contacto de la puerta de entrada
5889	0	0	Sensor de presion del sofa
5891	0	0	Sensor de movimiento del salon
5892	0	0	Sensor de movimiento de la habitacion
5893	0	0	Sensor de movimiento de la cocina

Figura 5.2: Valores del buffer en el instante inicial.

En el momento en el que pasa un minuto y los valores de timestamp cambian de 0 a 1, entonces el archivo buffer.csv se actualizará , por ende, se ejecutará el algoritmo. Por pantalla se imprimiría lo mostrado en la figura 5.3

```
ACTUALIZACION EN LOS VALORES DE LOS SENSORES:
Sensor de movimiento del pasillo: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento del servicio: timestamp=1, valor actual=0, valor esperado=0
Sensor de luz del baño: timestamp=1, valor actual=0, valor esperado=1
Sensor de presion de la cama: timestamp=1, valor actual=0, valor esperado=0
Sensor de contacto de la puerta del balcon: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento debajo de la cama: timestamp=1, valor actual=0, valor esperado=0
Sensor de luz de la cocina: timestamp=1, valor actual=0, valor esperado=0
Sensor de contacto de la puerta de entrada: timestamp=1, valor actual=0, valor esperado=0
Sensor de presion del sofa: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento del salon: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento de la habitacion: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento de la cocina: timestamp=1, valor actual=0, valor esperado=0
Valor de los sensores no esperado
```

Figura 5.3: Resultado impreso por terminal tras actualizarse el buffer.

Como se puede observar, el programa predice que en el baño, cuando "timestamp" vale 1, el valor del sensor de luz ha de ser 1. En este caso es 0, por lo que el programa imprime "Valor de los sensores no esperado", e internamente aumenta el valor del contador en 1.

```
ACTUALIZACION EN LOS VALORES DE LOS SENSORES:
Sensor de movimiento del pasillo: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento del servicio: timestamp=1, valor actual=0, valor esperado=0
Sensor de luz del baño: timestamp=1, valor actual=1, valor esperado=1
Sensor de presion de la cama: timestamp=1, valor actual=0, valor esperado=0
Sensor de contacto de la puerta del balcon: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento debajo de la cama: timestamp=1, valor actual=0, valor esperado=0
Sensor de luz de la cocina: timestamp=1, valor actual=0, valor esperado=0
Sensor de contacto de la puerta de entrada: timestamp=1, valor actual=0, valor esperado=0
Sensor de presion del sofa: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento del salon: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento de la habitacion: timestamp=1, valor actual=0, valor esperado=0
Sensor de movimiento de la cocina: timestamp=1, valor actual=0, valor esperado=0
Valores de los sensores esperados
```

Figura 5.4: El programa compara valores reales y predecidos y estos coinciden.

Si se corrigiese dicho valor de sensor, y efectivamente valiese 1, el programa imprimiría “Valores de los sensores esperados”, tal y como se observa en la figura 5.4.

En el caso en el que dicho valor volviese a ser 0 y pasasen 4 minutos, se habrían producido cinco cambios en el buffer (uno por el cambio del valor del sensor y cuatro por cada minuto). Esto conllevaría cinco ejecuciones del algoritmo y cinco predicciones discrepantes con la realidad, por lo que efectivamente se ejecutarían los sistemas de seguridad y el programa quedaría suspendido hasta que el usuario escribiese STOP en el terminal, lo que detendría las alarmas y reiniciaría el programa (véase la figura 5.5).

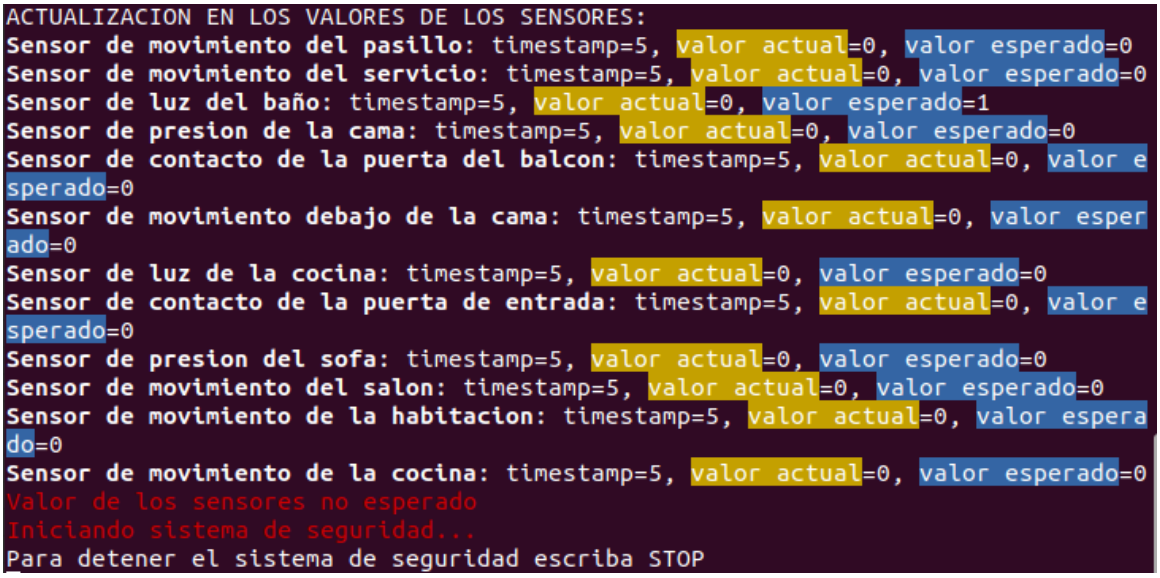

Figura 5.5: Se inician los sistemas de seguridad y el programa queda suspendido hasta que el usuario escriba STOP por línea de comandos.

6 Conclusiones y futuros trabajos

Tras haber realizado un estudio sobre un dataset real que contiene información acerca de diversos sensores distribuidos por una vivienda, se puede concluir que, a pesar de haber logrado crear un script capaz de simular la tarea de detectar situaciones anómalas en dicha vivienda, el resultado ha sido bastante negativo debido a que no se ha conseguido un porcentaje de acierto en las predicciones lo suficientemente fiable para poder llevar este programa a la práctica.

Las razones más destacables por las que ha sucedido esto han sido las siguientes:

- La falta de información acerca del estilo de vida y de los horarios de la persona o personas que residen en el hogar concerniente. El hecho de que no se aporte esta información afecta negativamente a la creación del algoritmo de Machine Learning, pues el poder discriminar entre días laborales/no laborales y horas en las que se encontrarían trabajando o no las personas guarda una fuerte correlación con los valores que puedan tomar los sensores.
- La falta de información acerca de los valores de los sensores. Existen concretamente cuatro sensores que no aportan información binaria, y no se aporta ningún dato en concreto sobre qué criterios sigue el sensor a la hora de dictaminar qué valor tomará, de forma que obliga al programador a realizar ciertas suposiciones que implican una pérdida de precisión en el resultado.

Como propuestas para **trabajos futuros** sería el repetir este mismo trabajo pero con un dataset que contenga una información más precisa y que se adapte a las carencias de información mencionadas anteriormente. En vez de comparar entre distintos tipos de árboles y de regresiones, se debeería realizar una comparación entre el árbol de decisión de tipo CART, que es el más fiable y utilizado, con una red neuronal, escogiendo como algoritmo de Machine Learning aquel que arroje mejores resultados.

7 Presupuesto

El presupuesto necesario para desarrollar este trabajo se limita básicamente al coste del trabajo del ingenierio en cuestión, puesto que no se ha utilizado ningún programa de pago y el PC empleado para la programación es propiedad de la Universidad de Alcalá de Henares.

Mano de obra			Total (€)
	Unidades	Precio por unidad	
Ingeniero de telecomunicaciones (en horas)	300	9€	2.700€
Otros			
Beneficio industrial	1	20 % del gasto total	540€
TOTAL			3.240€

Un ingeniero de telecomunicaciones gana 9€/h de media. Elaborar el trabajo supuso alrededor de 300 horas, por lo que el costo total del ingeniero sería 9*300€, es decir, 2.700€. A esto habría que añadirle el beneficio industrial, que es el 20 % de la suma de todos los costes.

Bibliografía

[1] M. Martínez, "Qué es IoT y cuáles son sus principales aplicaciones." https://blog.orange.es/innovacion/que-es-iot-aplicaciones, 08 2020.

[2] J. Howarth, "80+ Amazing IoT Statistics (2022-2030)." https://explodingtopics.com/blog/iot-stats, 03 2022.

[3] L. G. Serrano, *Grokking Machine Learning*. 2021.

[4] D. Salvador, "Sensores IoT: tipos y aplicaciones." https://www.nespra.net/blog/sensores-iot-tipos-y-aplicaciones/, 06 2019.

[5] R. de ComputerWorld, "IoT, un mercado de más de un billón de dólares en 2024." https://www.computerworld.es/tendencias/iot-un-mercado-de-mas-de-un-billon-de-dolares-en-2024, 06 2021.

[6] Tim, "The internet of things (IoT): 5 reasons why the world needs it." https://medium.com/zeux/the-internet-of-things-iot-5-reasons-why-the-world-needs-it-125fe71195cc, 02 2019.

[7] A. Geron, *Hands–On Machine Learning with Scikit–Learn and TensorFlow*. 2017.

[8] "What Is Machine Learning: Definition, Types, Applications And Examples." https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/#:~:text=These%20are%20three%20types%20of,unsupervised%20learning%2C%20and%20reinforcement%20learning.

[9] "9 Key Machine Learning Algorithms Explained in Plain English." https://www.freecodecamp.org/news/a-no-code-intro-to-the-9-most-important-machine-learning-algorithms-today/.

[10] F. S. Caparrini, "Algoritmos de clustering." http://www.cs.us.es/~fsancho/?e=230, 12 2020.

[11] D. Calvo, "Clasificación de redes neuronales artificiales." https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/, 07 2017.

[12] "Multi-sensor dataset of human activities in a smart home environment." https://www.sciencedirect.com/science/article/pii/S2352340920315122.

[13] profthyagu, "Python-Decision-Tree-Using-ID3." https://github.com/profthyagu/Python-Decision-Tree-Using-ID3/blob/master/3.Decision%20Tree%20Using%20ID3.ipynb, 07 2018.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá